

ASSIGNMENT #2

Computer Hardware

ECE 314/13CTT

Nguyễn Tuấn Nam - Lương Việt Thắng - Nguyễn Văn Linh

Ngày 11 tháng 3 năm 2015

1 Mục tiêu

- Lập trình cơ bản trong MIPS
- Hiểu các quy tắc gọi hàm, sử dụng mảng, chuỗi, stack ... trong MIPS.
- Thao tác với file trong MIPS.
- Cài đặt một số giải thuật tìm kiếm và sắp xếp sử dụng giải thuật đệ quy trong MIPS.

2 Nội dung

Cài đặt các hàm tìm kiếm và sắp xếp chuỗi số nguyên dưới dạng ngôn ngữ MIPS .

```
int linearsearch ( int * arr, int numofelements, int val ) ;
```

- Mô tả:
 - Tìm kiếm vị trí xuất hiện đầu tiên của giá trị `val` trong mảng `arr` theo kiểu tuyến tính.
- Tham số:
 - `arr`: con trỏ trỏ tới mảng số nguyên.
 - `numofelements`: số lượng phần tử trong mảng `arr`.
 - `val`: giá trị cần tìm kiếm.
- Giá trị trả về:
 - là vị trí của `val` trong mảng, ngược lại nếu giá trị `val` không xuất hiện trong mảng thì trả về -1.

```
1 int binarysearch ( int * arr, int numofelements, int val ) ;
```

- Mô tả:
 - Tìm kiếm vị trí xuất hiện đầu tiên của giá trị `val` trong mảng `arr` theo kiểu nhị phân (**Giả định mảng đã được sắp xếp tăng dần**).
 - **Yêu cầu:** Cài đặt bằng đệ quy.
- Tham số:
 - `arr`: con trỏ trỏ tới mảng số nguyên.
 - `numofelements`: số lượng phần tử trong mảng `arr`.
 - `val`: giá trị cần tìm kiếm.
- Giá trị trả về:
 - là vị trí của `val` trong mảng, ngược lại nếu giá trị `val` không xuất hiện trong mảng thì trả về -1.

```
1 int * bubblesort ( int * arr, int numofelements, bool asc ) ;
```

- Mô tả:
 - Sắp xếp mảng `arr` theo quy định của biến `asc` bằng giải thuật Bubble Sort.
- Tham số:
 - `arr`: con trỏ trỏ tới mảng số nguyên.
 - `numofelements`: số lượng phần tử trong mảng `arr`.
 - `asc`: `true` - sắp xếp tăng dần; `false` - giảm dần.
- Giá trị trả về:
 - Trả về mảng số nguyên đã được sắp xếp.

```
1 int * insertionsort ( int * arr, int numofelements, bool asc ) ;
```

- Mô tả:
 - Sắp xếp mảng `arr` theo quy định của biến `asc` bằng giải thuật Insertion Sort.
- Tham số:
 - `arr`: con trỏ trỏ tới mảng số nguyên.

- `numofelements`: số lượng phần tử trong mảng `arr`.
- `asc: true` - sắp xếp tăng dần; `false` - giảm dần.

- Giá trị trả về:

- Trả về mảng số nguyên đã được sắp xếp.

```
1 int * mergesort ( int * arr, int numofelements, bool asc ) ;
```

- Mô tả:

- Sắp xếp mảng `arr` theo quy định của biến `asc` bằng giải thuật Merge Sort.
- **Yêu cầu:** Cài đặt bằng đệ quy.

- Tham số:

- `arr`: con trỏ trỏ tới mảng số nguyên.
- `numofelements`: số lượng phần tử trong mảng `arr`.
- `asc: true` - sắp xếp tăng dần; `false` - giảm dần.

- Giá trị trả về:

- Trả về mảng số nguyên đã được sắp xếp.

```
1 int * quicksort ( int * arr, int numofelements, bool asc ) ;
```

- Mô tả:

- Sắp xếp mảng `arr` theo quy định của biến `asc` bằng giải thuật Quick Sort.
- **Yêu cầu:** Cài đặt bằng đệ quy.

- Tham số:

- `arr`: con trỏ trỏ tới mảng số nguyên.
- `numofelements`: số lượng phần tử trong mảng `arr`.
- `asc: true` - sắp xếp tăng dần; `false` - giảm dần.

- Giá trị trả về:

- Trả về mảng số nguyên đã được sắp xếp.

Sử dụng các hàm trên để giải các bài toán sau:

1. Sắp xếp một mảng số nguyên có n phần tử tăng dần bằng giải thuật Bubble Sort.
2. Sắp xếp một mảng số nguyên có n phần tử giảm dần bằng giải thuật Insertion Sort.
3. Sắp xếp một mảng số nguyên có n phần tử sao cho: đầu mảng là số nguyên âm giảm dần, cuối mảng là số nguyên không âm tăng dần bằng giải thuật Merge Sort.
4. Sắp xếp một mảng số nguyên có n phần tử sao cho: số nguyên âm và không âm xen kẽ nhau và cùng tăng dần bằng giải thuật Quick Sort.

Cấu trúc file input: Cấu trúc file **nhị phân** dùng để chạy chương trình như sau:

- $n + 2$ số nguyên đầu tiên:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên.
 3. Giá trị cần tìm kiếm
- $n + 2$ số nguyên tiếp theo:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên sắp xếp tăng dần.
 3. Giá trị cần tìm kiếm.
- $n + 1$ số nguyên tiếp theo:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên.
- $n + 1$ số nguyên tiếp theo:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên.
- $n + 1$ số nguyên tiếp theo:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên.
- $n + 1$ số nguyên tiếp theo:
 1. Số nguyên đầu tiên chỉ ra n số phần tử trong mảng.
 2. Danh sách n số nguyên.

Lưu ý: File input mẫu đính kèm.

Cài đặt chương trình minh hoạ các hàm đã cài đặt. Chương trình minh hoạ có giao diện như sau:

- Nhập đường dẫn đến file input:
- Xuất ra các kết quả sau:
 - Kết quả tìm kiếm tuyến tính.
 - Kết quả tìm kiếm nhị phân.
 - Kết quả bài toán 1.
 - Kết quả bài toán 2.
 - Kết quả bài toán 3.
 - Kết quả bài toán 4.

Lưu ý: Không được tạo menu chọn, chỉ cần xuất ra 6 kết quả theo thứ tự đã nêu trên.

3 Yêu cầu

- Sử dụng chương trình MARS.
- Các hàm trên không được sử dụng biến toàn cục để lưu trữ giá trị tính toán trung gian, phải sử dụng biến cục bộ.
- Sinh viên nộp bài theo cấu trúc sau:
 - Thư mục Source: chứa mã nguồn của chương trình (file .asm).
 - Thư mục Doc: chứa file mô tả cách thức cài đặt các hàm quan trọng.