Báo cáo đồ án 2

Môn Kiến trúc máy tính và Hợp ngữ

Các thành viên:

- Tôn Thất Vĩnh – MSSV: 1512679
- Nguyễn Đào Xuân Trường – MSSV: 1512624
- Vòng Chí Tài – MSSV: 1512474

Mục lục

1. Mô tả các hàm quan trọng

char* Date(int day, int month, int year, char* TIME)

Tương ứng với label date trong MIPS

Input: ngày ($a0), tháng ($a1), năm ($a2), TIME ($a3) là địa chỉ của chuỗi rỗng đã được cấp phát sẵn (11 phần tử)

Output: địa chỉ chuỗi TIME ($v0) đã được chuyển đổi theo định dạng DD/MM/YYYY

Các cài đặt: Lần lượt lấy từng kí tự của ngày ($a0), tháng ($a1), năm ($a2) gán vào vùng nhớ tương ứng trên ($v0)

char* Convert(char* TIME, char type)

Tương ứng với label convert trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY, type ($a1) thuộc 3 loại A/B/C

Output: $v0 là địa chỉ chuỗi kết quả tương ứng với type:

A. MM/DD/YYYY
B. Month DD, YYYY
C. DD Month, YYYY

Cách cài đặt: Hoán đổi vị trí các phần tử trong chuỗi, cài đặt thêm hàm getMonthName để lấy tên tháng và lưu vào chuỗi

int Day(char* TIME)

Tương ứng label day trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY

Output: giá trị nguyên ngày trong chuỗi TIME ($v0)

Cách cài đặt: Lấy từng giá trị ký tự trong chuỗi DD, chuyển đổi thành số nguyên và trả về

int Month(char* TIME)

Tương ứng label month trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY

Output: giá trị nguyên tháng trong chuỗi TIME ($v0)

Cách cài đặt: Lấy từng giá trị ký tự trong chuỗi MM, chuyển đổi thành số nguyên và trả về

int Year(char* TIME)

Tương ứng label year trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY

Output: giá trị nguyên năm trong chuỗi TIME ($v0)

Cách cài đặt: Lấy từng giá trị ký tự trong chuỗi YYYY, chuyển đổi thành số nguyên và trả về

int LeapYear(char* TIME)

Tương ứng label LeapYearC trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY

Output: $v0 = 1 nếu là năm nhuận, $v0 = 0 nếu không là năm nhuận.

Cách cài đặt: Lấy giá trị Year trong chuỗi TIME (bằng hàm Year) và xét nếu (year%4==0 && year%100==0) || year %400 ==0 thì trả về 1, ngược lại trả về 0

int GetTime(char* TIME_1, char* TIME_2)

Tương ứng label getTime trong MIPS

Input: địa chỉ chuỗi TIME_1 ($a0) và TIME_2 ($a1) theo định dạng DD/MM/YYYY

Output: giá trị nguyên số ngày cách biệt giữa thời gian trong chuỗi TIME_1 và TIME_2 ($v0)

Cách cài đặt: Cài đặt thêm hàm getStandardTime để lấy giá trị ngày tuyệt đối (Tính từ ngày 0/0/0) của chuỗi TIME, sau đó lấy giá trị getStandardTime của chuỗi TIME_2 trừ đi giá trị getStandardTime của chuỗi TIME_1

char* WeekDay(char* TIME)

Tương ứng label weekDay trong MIPS

Input: địa chỉ chuỗi TIME ($a0) theo định dạng DD/MM/YYYY

Output: trả về $v0 trỏ đến chuỗi là giá trị thứ trong tuần thuộc tập: {Mon; Tues; Wed; Thurs; Fri; Sat; Sun}

Cách cài đặt: Lấy thứ trong tuần của một ngày làm chuẩn: ví dụ 15/04/2017 là Sat. Sau đó lấy GetTime(TIME) - GetTime(<ngày chuẩn>) ra số ngày chênh lệch. Tìm biểu thức tính thứ trong tuần dựa trên số ngày lệch với thứ trong tuần của ngày chuẩn đó (trong code có chú thích).

2. Quy tắc khi viết và gọi hàm trong MIPS

Quy tắc viết hàm

- Truyền tham biến vào hàm bằng 4 thanh ghi $a, bắt đầu từ $a0. Các thanh ghi $t để chứa dữ liệu tạm dùng để tính toán.
- Trả output bằng 2 thanh ghi $v.
- Kết thúc hàm là lệnh jr $ra.
- Nếu trong hàm có thêm lệnh gọi hàm thì phải lưu $ra vào stack, để tránh chồng lắp mất $ra sẽ không quay lại được. Và trả lại từ stack ở cuối hàm.
- Trong hàm không được làm mất giá trị của các thanh ghi $s và $a. Nếu có thay đổi giá trị $a (thường là khi cần gọi thêm hàm, truyền tham biến vào hàm), thì phải lưu các thanh ghi đó vào stack trước khi gọi hàm. Và trả lại từ stack ở cuối hàm.
- Lệnh lưu và đọc stack cho cùng một thanh ghi phải cùng vị trí trên stack.
- Cấp phát cho stack bao nhiêu thì phải trả lại bấy nhiêu.

Quy tắc gọi hàm

- Trước khi gọi hàm, nếu đang dùng các thanh ghi $t thì phải lưu các thanh ghi đó vào stack để tránh trong hàm làm mất giá trị $t. Và trả lại từ stack sau lời gọi hàm.
- Có 4 thanh ghi $a để truyền tối đa 4 tham biến vào hàm, trường hợp xấu cần dùng nhiều hơn 4 tham biến thì có thể đẩy vào stack. Nhưng khuyến khích gom nhóm lại thành các biến cấu trúc thì đúng đắn hơn.