Kiến trúc máy tính và hợp ngữ:

Đồ án 2: Thư viện TIME

Nguyễn Sĩ Hùng (1612226) - Trần Quang Minh (1612374)
Đoàn Minh Hiếu (1612198)

Ngày 15 tháng 4 năm 2018

Mục lục

1 Đánh giá mức độ hoàn thành

Đánh giá mức độ hoàn thành của QInt	
Yêu cầu	Mức độ hoàn thành
Hàm Date	100%
Hàm Convert	100%
Hàm Day	100%
Hàm Month	100%
Hàm Year	100%
Hàm LeapYear	100%
Hàm GetTime	100%
Hàm Weekday	100%
Hàm Weekday	100%
Tổng cộng	100%

2 Cách thức cài đặt

2.1 Hàm Date

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

Input: $a0, $a1, $a2 lần lượt là các số nguyên day, month, year (đã hợp lệ) $a3 là địa chỉ vùng nhớ TIME.

Output: $v0 là địa chỉ trỏ đến vùng nhớ lưu trữ kết quả chuỗi ngày tháng năm theo định dạng DD\MM \YYYY.

Cách làm: Day và month đều có tối đa 2 chữ số nên ta lấy giá trị day, month chia cho 10. Chữ số đầu tiên là phần nguyên, chữ số thứ 2 là phần dư. Lấy 2 kí tự ngày store byte vào 0($a3) và 1($a3) , tiếp tục ta store byte vào 2($a3) kí tự '/' (47). Cứ như vậy đến tháng. Năm thì ta làm tương tự nhưng chia cho 1000 rồi 100 rồi 10 để lấy đủ 4 chữ số và store byte đến 9($a3). Sau đó trả về kết quả là địa chỉ của thanh ghi $a3.

2.2 Hàm Convert

```
char* Convert(char* TIME, char type)
```

Input: $a0 là địa chỉ của TIME (lưu theo định dạng DD\MM \YYYY), $a1 là địa chỉ của kí tự type.

Output: $v0 là địa chỉ của TIME đã thay đổi định dạng.

Cách làm: Ta tạo ra một mảng các chuỗi lưu tên của tháng (kích thước một chuỗi là 10 byte). Tiếp theo, ta tách TIME ra làm 3 phần ngày, tháng, năm riêng biệt để tiện xử lý. Sau khi xác định yêu cầu người dùng, ta sẽ thay đổi giá trị TIME lần lượt từng byte. Nếu type là A thì ta trả về chuỗi TIME ban đầu;

nếu type là B thì ta đọc vào chuỗi TIME tên tháng, sau đó đọc ngày và cuối cùng là năm theo đúng định dạng Month DD, YYYY; nếu type là C thì tương tự, ta đọc ngày trước, sau đó là tháng và cuối cùng là năm theo đúng định dạng DD Month, YYYY. Cuối cùng là gán $v0 là TIME.

2.3 Hàm Day

```
int Day(char* TIME)
```

Input: $a0 là địa chỉ lưu trữ chuỗi TIME (DD\MM \YYYY).

Output: $v0 là giá trị ngày trong chuỗi TIME.

Cách làm: Ta load byte lên ký tự ở vị trí 0($a0) và 1($a0) sau đó trừ đi 48 để được số nguyên. Kí tự đầu $*$ 10 + Kí tự thứ 2 là kết quả gán vào $v0.

2.4 Hàm Month

```
int Month(char* TIME)
```

Input: $a0 là địa chỉ lưu trữ chuỗi TIME (DD\MM \YYYY).

Output: $v0 là giá trị tháng trong chuỗi TIME.

Cách làm: Ta load byte lên ký tự ở vị trí 3($a0) và 4$a0) sau đó trừ đi 48 để được số nguyên. Kí tự đầu $*$ 10 + Kí tự thứ 2 là kết quả gán vào $v0.

2.5 Hàm Year

```
int Year(char* TIME)
```

Input: $a0 là địa chỉ lưu trữ chuỗi TIME (DD\MM \YYYY).

Output: $v0 là giá trị năm trong chuỗi TIME.

Cách làm: Ta load từng byte đến khi gặp kí tự NULL, sau đó trừ đi 48 để được số nguyên. Mỗi lần lặp, ta sẽ lấy giá trị trước đó $*$ 10 rồi + giá trị lấy được. Sau khi thực hiện vòng lặp, ta thu được giá trị cuối cùng sẽ gán vào $v0.

2.6 Hàm LeapYear

```
int LeapYear(char* TIME)
```

Input: $a0 là là địa chỉ vùng nhớ TIME.

Output: $v0: một trong 2 giá trị 0 (không là năm nhuận), 1 (là năm nhuận).

Cách làm: Gọi hàm Year để lấy ra giá trị năm từ chuỗi TIME, sau đó gọi hàm LeapYearNum.

2.7 Hàm GetTime

```
int GetTime(char* TIME_1, char* TIME_2)
```

Input: $a0, $a1: địa chỉ lưu trữ chuỗi TIME1, TIME2.

Ouput: $v0, khoảng cách năm giữa 2 khoảng thời gian.

Cách làm: Đầu tiên tìm ra khoảng thời gian trễ hơn. Gọi y1, y2, là năm của hai thời gian theo thứ tự tăng dần, tương tự cho d1, d2, m1, m2. Nếu $y2 - y1 = 0$, kết quả trả về sẽ là 0. Khi $y2 - y1 > 0$, gọi $h = y2 - y1$, ta xét m2 và m1, nếu $m2 > m1$, kết quả trả về sẽ là h, $m2 < m1$, kết quả trả về sẽ là $h\check{}1$, $m2 = m1$ sau đó xét đến d2 và d1, $d2 >= d1$, kết quả là h, $d2 < d1$, kết quả sẽ là $h\check{}1$.

2.8 Hàm Weekday

```
char* Weekday(char* TIME)
```

Input: $a0 địa chỉ chuỗi TIME.

Output: $v0, cho biết TIME là thứ mấy trong tuần.

Cách làm: Sử dụng công thức $h = [d + t[m-1] + y + floor(\frac{y}{4}) - floor(\frac{y}{100}) + floor(\frac{y}{400})] \bmod 7$. Với $t[] = \{0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4\}$ và nếu $m < 3$ thì $y = y - 1$. d, m, y tương ứng là ngày tháng năm.

2.9 Các hàm phụ trợ

2.9.1 Hàm isNumber

```
int isNumber(char *string)
```

Input: $a0 là địa chỉ một chuỗi.

Output: Nếu chuỗi chứa kí tự ngoài (0-9) thì gán $v0 là 0. Ngược lại trả về giá trị nguyên mà số đó biểu diễn.

Cách làm: Duyệt lần lượt các kí tự trong chuỗi string, nếu chứa kí tự ngoài (0-9) thì $v0 = 0. Ngược lại dùng hàm string_to_num để đổi sang số nguyên tương ứng.

2.9.2 Hàm string_to_num

```
int string_to_num(char *string)
```

Input: $a0 là chuỗi cần chuyển sang số. (chuỗi chỉ bao gồm kí tự từ 0-9).

Output: $v0 là giá trị số của chuỗi.

Cách làm: Duyệt lần lượt chuỗi từ trái sang phải đến khi gặp kí tự NULL. Sau đó ta chuyển từ kí tự sang số bằng cách − 48. Mỗi lần lặp, ta sẽ lấy giá trị

trước đó $*$ 10 rồi + giá trị lấy được. Sau khi thực hiện vòng lặp, ta thu được giá trị cuối cùng sẽ gán vào $v0.

2.9.3 Hàm num_to_string

```
char* num_to_string(int num)
```

Input: $a0 là số cần chuyển sang chuỗi.

Output: $v0 là giá trị chuỗi của số.

Cách làm: Ta lần lượt chia số cho 10, phần dư sẽ được lưu vào một chuỗi tạm (chuỗi tạm này ban đầu chỉ có kí tự NULL), phần nguyên được làm số bị chia cho lần lặp tới. Quá trình lặp kết thúc khi số bị chia = 0, khi đó, ta sẽ đảo ngược chuỗi tạm để được kết quả lưu ra $v0.

2.9.4 Hàm string_modify

```
char* string(char* string)
```

Input: $a0 là chuỗi cần định dạng lại.

Output: $v0 là chuỗi đã được định dạng bỏ đi kí tự \n.

Cách làm: Duyệt lần lượt chuỗi cho đến khi gặp kí tự \n, thay đổi kí tự đó bằng kí tự NULL. Gán $v0 là địa chỉ của chuỗi mới định dạng.

2.9.5 Hàm check

```
int check(char* day, char* month, char* year)
```

Input: $a0 là chuỗi lưu ngày, $a1 là chuỗi lưu tháng, $a2 là chuỗi lưu năm.

Output: $v0 là 1 khi ngày tháng năm hợp lệ, ngược lại là 0.

Cách làm: Đầu tiên, ta kiểm tra các chuỗi ngày, tháng và năm có toàn kí tự từ 0-9 hay không bằng hàm isNumber, nếu đúng thì ta làm tiếp, ngược lại thì gán $v0 là 0. Sau đó, ta kiểm tra từng ngày có thỏa điều kiện trong tháng đó hay không (ta sử dụng một mảng số để lưu số ngày tối đa của 12 tháng). Chú ý tháng 2 có 2 trường hợp năm nhuận và năm không nhuận. Nếu thỏa mãn hết các điều kiện thì gán $v0 = 1, ngược lại là 0.

2.9.6 Hàm LeapYearNum

```
int LeapYearNum(int Year)
```

Input: $a0 giá trị kiểu int ứng với năm cần x.

Output: $v0: một trong 2 giá trị 0 (không là năm nhuận), 1 (là năm nhuận).

Cách làm: Sử dụng câu điều kiện sau:

```
if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))
   return 1;
else return 0;
```

2.9.7 Hàm getTwoNearestLeapYear

```
pair<int, int> getTwoNearestLeapYear(char* TIME)
```

Input: $a0 là là địa chỉ vùng nhớ TIME.

Output: $v0, $v1, hai năm nhuận gần nhất lơn hơn year(TIME).

Cách làm: Từ input $a0: địa chỉ chuỗi TIME ta dùng hàm Year để lấy ra năm hiện tại, tăng năm hiện tại lên 1 rồi cho vào vòng lặp đến khi nào gặp năm nhuận đầu tiên thì lưu vào $v0, rồi lại tăng năm lên 1 và cho vào vòng lặp đến khi nào gặp năm nhuận thứ 2 thì lưu vào $v1.

3 Quy tắc gọi hàm

Đầu hàm: Cấp phát vùng nhớ trong stack để lưu địa chỉ trả về của $ra. Cấp phát vùng nhớ trong stack để lưu tạm các thanh ghi khác nếu cần.
Cuối hàm: Load lại địa chỉ $ra để return. Giải phóng vùng nhớ stack đã cấp phát.

4 Các nguồn tài liệu tham khảo

1. MARS - Mips Assembly and Runtime Simulator
2. MIPS architecture Wikipedia
3. Slide bài giảng Kiến Trúc Máy Tính và Hợp Ngữ (Phạm Tuấn Sơn)