**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**


# ĐỒ ÁN 02 MÔN KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ

Đề tài:

# THƯ VIỆN TIME – MIPS

Người thực hiện:

Nguyễn Hoàng Thái Dương (18120336)

Trần Thanh Quang (18120230)

Trương Trọng Lộc (18120197)

Giảng viên hướng dẫn:

Thầy Phạm Tuấn Sơn

Thành phố Hồ Chí Minh – Tháng 06, 2020

# MỤC LỤC

# 1. THÀNH VIÊN VÀ PHÂN CHIA CÔNG VIỆC:

| STT | MSSV | Tên thành viên | Công việc |
|---|---|---|---|
| 1 | 18120336 | Nguyễn Hoàng Thái Dương | Hàm Convert, Weekday, GetTime và Menu |
| 2 | 18120230 | Trần Thanh Quang | Hàm lấy giá trị Day, Month, Year |
| 3 | 18120197 | Trương Trọng Lộc | Hàm LeapYear, NearestLeapYear, Date, nhập và kiểm tra input |

# 2. QUY TẮC VIẾT VÀ GỌI HÀM TRONG MIPS

## 2.1. Nhắc lại các quy ước khi thao tác với hàm (thủ tục) trong MIPS:

Trong MIPS, các giá trị được lưu trữ trên các thanh ghi. Không giống như C/C++, các biến tĩnh được khởi tạo trong hàm sẽ được giải phóng khi kết thúc hàm, các biến này được gọi là biến cục bộ. Còn MIPS, các thanh ghi như là các biến toàn cục, tức là nếu ở bất kì dòng lệnh nào ta thay đổi giá trị của thanh ghi thì sự thay đổi đó là vĩnh viễn, chỉ có thể khôi phục lại nếu được lưu giữ trước đó, chứ không tự động giữ nguyên không thay đổi giá trị. Để dễ thống nhất, người ta quy ước 32 thanh ghi vào các loại sau:

| Tên | Thanh ghi | Ý nghĩa |
|---|---|---|
| $zero | 0 | Thanh ghi này luôn chứa giá trị 0 |
| $at | 1 | Được dành cho Assembler |
| $v0, $v1 | 2, 3 | Lưu giá trị trả về của hàm |
| $a0 - $a3 | 4 – 7 | Lưu tham số truyền vào của hàm |
| $t0 - $t7 | 8 - 15 | Lưu biến tạm |
| $s0 - $s7 | 16 – 23 | Lưu biến (có thể hiểu lưu giá trị bền vững) |
| $t8, $t9 | 24, 25 | Lưu biến tạm (More temporary) |
| $k0, $k1 | 26, 27 | Được dùng cho nhân Hệ điều hành (Kernel) sử dụng |
| $gp | 28 | Global Pointer |
| $sp | 29 | Stack Pointer |
| $fp | 30 | Frame Pointer |

| | | |
|---|---|---|
| $ra | 31 | Trả về địa chỉ, sử dụng cho việc gọi hàm |

Như đã nói ở trên, do các thanh ghi trong MIPS mang ý nghĩa là toàn cục nên ta cần lưu trữ giá trị của thanh ghi trước khi thực hiện các câu lệnh và khôi phục nó, nên người ta đặt ra quy tắc **Stack.** Stack theo quy chế Last In First Out (LIFO), nên ta dễ dàng thực hiện các thao tác "push" và "pop" khi lưu trữ.

Trong các thanh ghi, các thanh ghi $a, $ra, $s cần được đưa vào Stack và được phục hồi sau khi thực hiện xong các câu lệnh. Các thanh ghi $t, $v mang ý nghĩa là thanh ghi tạm / trả về nên ta chỉ thực hiện lưu trữ và phục hồi lại khi thật sự cần thiết (khi muốn quá trình thực hiện các câu lệnh trong hàm không làm thay đổi giá trị của các thanh ghi này trước đó).

### 2.2. Quy tắc viết hàm trong MIPS:

Ở mỗi hàm (thủ tục) ta thực hiện 4 công đoạn: Đầu thủ tục, Thân thủ tục, Cuối thủ tục và Trả về. Cụ thể:

- Đầu thủ tục: "push" các thanh ghi cần thiết vào Stack.
- Thân thủ tục: thực hiện các câu lệnh để giải quyết vấn đề của hàm đó.
- Cuối thủ tục: phục hồi lại các thanh ghi trong Stack và trả Stack về trạng thái ban đầu.
- Trả về: Trả về địa chỉ thanh ghi $ra.

Cấu trúc của một thủ tục trong MIPS:

```
label:
    # Đầu thủ tục
    addi $sp, $sp, -framesize # khai báo kích thước cho stack
    sw $ra, framesize - 4($sp) # cất địa chỉ trả về của thủ tục trong $ra vào ngăn xếp
    Lưu tạm các thanh ghi khác (nếu cần)
    # Thân thủ tục ...
    (có thể gọi các thủ tục khác...)
    Lưu giá trị trả về vào thanh ghi $v
```

```
# Cuối thủ tục

Phục hồi các thanh ghi khác

lw $ra, framesize  -4($sp) # lấy địa chỉ trả về ra $ra

addi $sp,$sp, framesize

# Trả về

jr $ra

#Gọi thủ tục:

jal label
```

### 2.3. Quy tắc gọi hàm trong MIPS:

- Nếu hàm cần tham số truyền vào, cần lưu các tham số vào các thanh ghi $a trước khi gọi hàm.
- Xác định các thanh ghi nào vẫn giữ giá trị như cũ sau khi gọi hàm để tiến hành đưa vào Stack.
- Nhảy đến hàm cần thực hiện (sử dụng *jal*)

## 3. CÁCH THỨC CÀI ĐẶT CÁC HÀM QUAN TRỌNG

### 3.1. Nhập dữ liệu (kiểm tra đầu vào):

Với Nhap ngay DAY:

- Người dùng nhập vào một chuỗi ký tự.
- Lưu lại chuỗi ký tự đó và nhảy đến hàm *isDigit* để kiểm tra chuỗi nhập vào có phải là toàn số hay không. Hàm này trả về 1 nếu toàn là số, ngược lại trả về 0.
    - Nếu chuỗi nhập vào là toàn số, ta lưu trạng thái này bằng cách cộng dồn vào một biến (nghĩa là biến đó được cộng thêm 1) và tiếp tục nhảy đến hàm chuyển chuỗi ký tự số đó sang dạng số nguyên.
    - Nếu chuỗi nhập không là toàn số, lưu trạng thái này bằng cách cộng dồn vào một biến (nghĩa là biến đó được cộng thêm 0) và nhảy đến nhãn *NhapMonth*.

Với Nhap thang MONTH và Nhap nam YEAR thực hiện tương tự như Nhap ngay DAY. Sau khi nhập xong DAY, MONTH, YEAR nếu biến lưu trạng thái bằng 3, nghĩa là 3 chuỗi DAY, MONTH, YEAR nhập vào đều toàn là số. Ta tiếp tục nhảy đến hàm kiểm tra tính hợp lệ về mặt logic của ngày, tháng, năm (sẽ được trình bày ở mục 3.2). Ngược lại, nếu biến lưu trạng thái khác 3, ta thực hiện thông báo cho người dùng nhập không hợp lệ và lặp cho người dùng nhập lại.

### 3.2. Kiểm tra tính hợp lệ ngày, tháng, năm:

Sau khi thỏa 3.1, ta thực hiện:

- Xét tháng (lúc này kiểu số nguyên nên dễ dàng trong việc sử dụng *beq* và *addi*):
  - Nếu rơi vào các tháng 1, 3, 5, 7, 8, 10, 12, nhảy đến nhãn *Check31Days*.
  - Nếu rơi vào các tháng 4, 6, 9, 11, nhảy đến nhãn *Check30Days*.
  - Nếu rơi vào tháng 2, nhảy đến nhãn *CheckThang2*.
- Ở các nhãn *Check31Days* và *Check30Days*, thực hiện kiểm tra ngày có vượt quá [1, 31] hay [1, 30] hay không. Nếu vượt ra khỏi đoạn này, dữ liệu nhập vào không hợp lệ.
- Ở nhãn *CheckThang2*, ta kiểm tra ngày có vượt quá [1, 28] hay không. Nếu vượt ra 28, ta kiểm tra năm đó có phải năm nhuận hay không, nếu có thì kiểm tra ngày có bằng 29 hay không. Nếu ngày không bằng 29, nghĩa là không hợp lệ.
- Vì do xét toàn số, nên nếu dưới dùng nhập giá trị âm thì sẽ có ký tự '-', ký tự này đã được kiểm tra ở 3.1 vì nó không phải là số.

### 3.3. Xuất TIME theo định dạng DD/MM/YYYY:

Tương ứng C/C++:

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

- Với ngày và tháng, do định dạng gồm 2 chữ số, nên ta thực hiện chia cho 10, lấy phần nguyên cho phần chục và phần dư cho phần đơn vị.
- Với năm, do định dạng gồm 4 chữ số, nên ta thực hiện chia cho 1000, ta được phần ngàn, sau đó lấy phần dư chia cho 100, ta được phần trăm. Cứ như vậy đến khi nhận được phần đơn vị.
- Các số sẽ được cộng thêm 48 để biến thành ký tự lưu vào mảng. Đồng thời, ta thêm ký tự '/' vào các vị trí thích hợp để phân cách ngày, tháng, năm.

### 3.4. Chuyển đổi giữa các định dạng:

### 3.5. Các hàm lấy ngày, tháng, năm từ chuỗi:

Tương ứng C/C++:

**int Day (char* TIME)**

- Hàm nhận vào 1 tham số là chuỗi TIME.
- Giá trị Day là 2 bit đầu của chuỗi time. Do đó, ta thực hiện lấy 2 bit đầu của chuỗi TIME, chuyển từ kiểu chuỗi thành số nguyên, rồi trả kết quả về số nguyên Day.

Tương ứng C/C++:

**int Month (char* TIME)**

- Hàm nhận vào 1 tham số là chuỗi TIME.
- Giá trị Mont là 2 bit thứ 4 và thứ 5 của chuỗi TIME(TIME(3), TIME(4)). Do đó, ta thực hiện lấy 2 bit thứ 4 và thứ 5 của chuỗi TIME, chuyển từ kiểu chuỗi thành số nguyên, rồi trả kết quả về số nguyên Month

Tương ứng C/C++:

**int Year (char* TIME)**

- Hàm nhận vào 1 tham số là chuỗi TIME.
- Giá trị Year là bit thứ 6 đến thứ 9 của chuỗi TIME(TIME(6) đến TIME(9)). Do đó, ta thực hiện lấy 4 bit thứ 6 đến thứ 9 của chuỗi TIME, chuyển từ kiểu chuỗi thành số nguyên, rồi trả kết quả về số nguyên Year

### 3.6. Kiểm tra năm nhuận:

Tương ứng C/C++:

**int LeapYear(char* TIME)**

- Thực hiện gọi hàm lấy YEAR từ chuỗi TIME. Khi đó ta được YEAR ở dạng số nguyên. Ta gọi lại CheckLeapYear để kiểm tra năm nhuận hay không với tham số là *int Year* (C++).

- Hàm CheckLeapYear kiểm tra xem năm đó có chia hết cho 400 hay không. Nếu có đó là năm nhuận.
- Ngược lại kiểm tra năm đó có chia hết cho 4 không, nếu có kiểm tra tiếp có chia hết cho 100 hay không. Nếu chia hết cho 4 nhưng không chia hết cho 100 thì đó là năm nhuận.
- Các trường hợp còn lại, không phải là năm nhuận.
- Trả về 0 nếu không là năm nhuận và 1 nếu là năm nhuận.

### 3.7. Xác định thứ trong tuần:

### 3.8. Khoảng cách giữa 2 TIME:

### 3.9. Hai năm nhuận gần nhất:

Ở đây, ta xét năm nhuận gần nhất về phía 2 bên. Giả sử, gọi Y là năm nhập vào, khi đó X và Z là 2 năm nhuận gần nhất thỏa X < Y < Z.

Tương ứng C/C++:

**pair<int, int> NearestLeapYear(char* TIME)**

- Đầu tiên ta tìm X bằng cách giảm dần Y xuống, mỗi lần giảm 1, đến khi gặp năm nhuận.
- Tương tự, ta tìm Z bằng cách tăng Y lên, mỗi lần tăng 1, đến khi gặp năm nhuận.
- Trả về $v0 và $v1 đại diện cho X và Z.

## 4. ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

| Công việc | Mức độ hoàn thành |
|---|---|
| Nhập Input | 100% |
| Kiểm tra Input | 100% |
| Chức năng 1: Xuất định dạng DD/MM/YYYY | 100% |
| Chức năng 2: Xuất định dạng theo yêu cầu | 100% |

| | |
|---|---|
| Chức năng 3: Xác định thứ trong tuần | 100% |
| Chức năng 4: Kiểm tra năm nhuận | 100% |
| Chức năng 5: Khoảng cách giữa hai TIME | 100% |
| Chức năng 6: Hai năm nhuận gần nhất | 100% |
| **Tổng thể đồ án** | **100%** |

## 5. TÀI LIỆU THAM KHẢO

[1] Slides bài giảng Kiến trúc máy tính và hợp ngữ, Thầy ThS Phạm Tuấn Sơn, trường Đại học Khoa học Tự nhiên.