

Bài thực hành số 1

Mục tiêu:

Thực hành nhằm hiểu rõ các vấn đề cơ bản liên quan đến MIPS trên phần mềm mô phỏng MARS:

- Cách viết và chạy các lệnh MIPS trên phần mềm mô phỏng MARS
- Cách lưu trữ lệnh và dữ liệu trong bộ nhớ
- Cách truy xuất dữ liệu từ bộ nhớ
- Mảng và các thao tác
- Cách chuyển từ các cấu trúc điều khiển, vòng lặp trong ngôn ngữ lập trình cấp cao xuống hợp ngữ MIPS

Nội dung:

1. Phần 1

Cài đặt và tìm hiểu cách sử dụng phần mềm mô phỏng MARS

Hướng dẫn: đọc file “1. Hướng dẫn sử dụng MARS” đính kèm trong bài thực hành

2. Phần 2

a. Tìm hiểu các nội dung cơ bản khi lập trình hợp ngữ theo kiến trúc MIPS

- ✓ Cấu trúc một chương trình như thế nào.
- ✓ Giới hạn, cách lưu trữ dữ liệu, cách khai báo biến.
- ✓ Cách gọi các hàm hệ thống

Hướng dẫn: đọc file “2. Tổng quan hợp ngữ và MIPS” đính kèm trong bài thực hành

b. Sử dụng MARS để kiểm tra lại chức năng các lệnh MIPS cơ bản đã học

Hướng dẫn: đọc file “3. Các lệnh MIPS cơ bản” đính kèm trong bài thực hành

3. Phần 3 – Tổng hợp cách lập trình một đoạn code MIPS trên MARS

Sử dụng MARS chạy các ví dụ sau và cho biết ý nghĩa

Ví dụ 1:

```

.data
var1: .word 23

.text
__start:
    lw    $t0, var1
    li    $t1, 5
    sw    $t1, var1
  
```

Ví dụ 2:

```

array1:      .data
             .space 12
             .text
__start:     la      $t0, array1
             li      $t1, 5
             sw      $t1, ($t0)
             li      $t1, 13
             sw      $t1, 4($t0)
             li      $t1, -7
             sw      $t1, 8($t0)

```

Ví dụ 3:

```

li      $v0, 5

syscall

```

Ví dụ 4:

```

string1:     .data
             .asciiz "Print this.\n"

             .text
main:        li      $v0, 4

             la      $a0, string1
             syscall

```

4. Phần 4 – Thực hành với cấu trúc điều khiển và vòng lặp

Chuyển đoạn code trong bảng theo sau sang MIPS và sử dụng MARS để kiểm tra lại kết quả:

```

if (i == j)
    f = g + h;
else
    f = g - h;

```

(Với giá trị của i, j, f, g, h lần lượt chứa trong các thanh ghi $\$s0, \$s1, \$s2, \$t0, \$t1$)

```

int Sum = 0
for (int i = 1; i <= N; ++i){
    Sum = Sum + i;
}

```

(Với giá trị của i , N , Sum lần lượt chứa trong các thanh ghi $\$s0$, $\$s1$, $\$s2$)

5. Phần 5 – Thao tác với mảng

Mảng với n phần tử là một chuỗi n phần tử liên tiếp nhau trong bộ nhớ. Thao tác với mảng trong MIPS là thao tác trực tiếp với byte/word trong bộ nhớ.

- Để cấp phát chuỗi word hoặc byte trong bộ nhớ, có giá trị khởi tạo sử dụng “.word” hoặc “.byte” trong “.data”
- Để cấp phát chuỗi byte không có giá trị khởi tạo trước, sử dụng “.space” trong “.data”

Cho ba mảng với cấp phát dữ liệu trong bộ nhớ như sau:

```

.data
array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
size1:  .word 10

array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
size2:  .word 16

array3: .space 8
size3:  .word 8

```

Mảng array1 có 10 word, kích thước được lưu trong size1; Mảng array2 có 16 byte, kích thước được lưu trong size2; Mảng array3 có 8 byte, kích thước được lưu trong size3.

Viết code trong phần “.text” thực hiện riêng từng phần việc:

- ✓ In ra cửa sổ I/O của MARS tất cả các phần tử của mảng array1 và array2
- ✓ Gán các giá trị cho mảng array3 sao cho

$$\text{array3}[i] = \text{array2}[i] + \text{array2}[\text{size2} - 1 - i]$$
- ✓ Người sử dụng nhập vào mảng thứ mấy và chỉ số phần tử cần lấy trong mảng đó, chương trình xuất ra phần tử tương ứng.

6. Phần 6 – Bài tập tổng hợp

1. Hãy viết chương trình hợp ngữ MIPS để giải quyết các bài toán sau:

a. Nhập vào một chuỗi, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

- ✓ Xuất ra lại đúng chuỗi đã nhập

Ví dụ:

Nhap: Ky Thuat May Tinh

Xuat: Ky Thuat May Tinh

- ✓ Xuất ra chuỗi ngược với chuỗi đã nhập

Ví dụ:

Nhap: Ky Thuat May Tinh

Xuat: hniT yaM tauhT yK

- ✓ Xuất ra chiều dài của chuỗi (tính luôn khoảng trắng)

b. Nhập vào một ký tự, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

- ✓ Ký tự liền trước và liền sau của ký tự nhập vào

Ví dụ:

Nhap ky tu (chỉ một ký tự): b

Ky tu truoc: a

Ky tu sau: c

- ✓ Ký tự nhập vào chỉ được phép là ba loại: số, chữ thường và chữ hoa. Nếu ký tự nhập vào rơi vào một trong ba loại, xuất ra cửa sổ đó là loại nào; nếu ký tự nhập không rơi vào một trong ba loại trên, xuất ra thông báo “invalid type”

c. Nhập vào một số nguyên dương, xuất ra cửa sổ I/O của MARS:

Nếu số nhập vào không là số nguyên dương, chương trình kết thúc với thông báo “invalid Entry”; nếu số nhập vào là nguyên dương, tên của từng chữ số được in ra và cách nhau một khoảng trắng

Ví dụ: Nếu số nhập vào “728”, in ra cửa sổ sẽ là “Seven Two Eight”

d. Nhập từ bàn phím 2 số nguyên, in ra cửa sổ I/O của MARS theo từng yêu cầu sau:

- ✓ Số lớn hơn
- ✓ Tổng, hiệu, tích và thương của hai số

e. Nhập một mảng các số nguyên n phần tử (nhập vào số phần tử và giá trị của từng phần tử), xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

- ✓ Xuất ra giá trị lớn nhất và nhỏ nhất của mảng

- ✓ Tổng tất cả các phần tử của mảng
 - ✓ Người sử dụng nhập vào chỉ số của một phần tử nào đó và giá trị của phần tử đó được in ra cửa sổ
2. Nhìn lại các chương trình trên, chương trình có sử dụng lệnh giả hay không, nếu có thay thế để không sử dụng các lệnh giả này.
 3. Nhập một mảng các số nguyên n phần tử (nhập vào số phần tử và giá trị của từng phần tử). Mảng này gọi là A.

Chuyển dòng lệnh C dưới đây sang mã assembly của MIPS. Với các biến nguyên i, j được gán lần lượt vào thanh ghi $\$s0, \$s1$; và địa chỉ nền của mảng số nguyên A được lưu trong thanh ghi $\$s3$

```
if (i < j) A[i] = i;
else A[i] = j;
```

4. Nhập vào hai mảng số nguyên A và B chứa lần lượt n_A và n_B phần tử (nhập vào số phần tử và giá trị của từng phần tử cho từng mảng).

Chuyển dòng lệnh C dưới đây sang mã Assembly của vi xử lý MIPS, với A, B là các mảng số nguyên, i là biến nguyên. Giả sử biến i được gán vào thanh ghi $\$s0$, địa chỉ nền của mảng A và B lần lượt được lưu trong thanh ghi $\$s1$ và $\$s2$.

```
for (i = 2; i < 10; i++)
    A[i] = B[A[i-2]];
```

-----Hết-----