BUỔI THỨ 6 – Địa chỉ, mảng

Thời lượng: 3 tiết.

Biên soạn	*Trần Trung Tín*
Phiên bản	21.08.2016
Mọi góp ý đều được đón nhận chân thành, xin gửi email cho tôi đến tttin@tdt.edu.vn	

Nhãn

* Trong MIPS, nhãn ngoài việc định vị một câu lệnh, nó còn dùng cho định vị một vị trí trong bộ nhớ cho một hằng số hay biến số.

```
.data
N: .word 10
.text
main:
lw $t0, N # $t0 <-- Mem[N] (10)
la $t1, N # $t1 <-- N (address)
. . .
exit: li $v0, 10
syscall
```

Với *register* mode, địa chỉ truy cập đến là giá trị đang chứa trong thanh ghi

```
lw $t0, ($s3) # giá trị đang lưu trong bộ nhớ có địa chỉ chứa trong $s3
lw $t0, $s3 # giá trị đang lưu trong $s3
```

Với *immediate* mode, có thể truyền trực tiếp địa chỉ vào cho một thanh ghi. Lưu ý: cách này không nên dùng và nhiều rủi ro.

```
lw $t0, 0 # nạp giá trị lưu trong bộ nhớ tại địa chỉ 0 vào thanh ghi $t0
li $t0, 0 # nạp giá trị 0 vào thanh ghi $t0
la $t0, var # nạp địa chỉ bộ nhớ của nhãn var vào cho thanh ghi $t0
```

Với chế độ hỗn hợp *base* + *register,* địa chỉ là tổng của số nguyên và số chứa trong thanh ghi.

```
lw $t0, 100($s3) # address is $s3 + 100
```

Ngoài ra còn một số cách sử dụng *label* khác:

```
lw $t0, absval # absval is a label
lw $t0, absval + 100
lw $t0, absval + 100($s3)
```

Mảng

*** Khai báo**

```
.asciiz "a string" # khai báo chuỗi kí tự
.byte 13, 14, -3 # yêu cầu cấp phát 3 bytes và lưu trữ các giá trị liệt kê
.space 16 # yêu cầu cấp phát 16 bytes
.word 13, 14, -3 # yêu cầu cấp phát 3 words (1 word = 4 bytes) và lưu trữ
các giá trị liệt kê.
```

Lưu ý: trong MIPS không có khái niệm "kiểu dữ liệu", tất cả câu lệnh hay các giá trị đều được lưu trữ là chuỗi 32 bits. Việc khai báo kiểu là để thể hiện cho người lập trình cách sử dụng các biến số.

*** Xuất mảng**

```
.data
list: .word 2, 3, 5, 7, 11, 13, 17, 19, 23, 29
size: .word 10

lw $t3, size # $t3 chứa số phần tử của mảng (N = 10)
la $t1, list # địa chỉ của "list" truyền và cho $t1
li $t2, 0 # biến đếm vòng lặp i
print_loop:
beq $t2, $t3, print_loop_end # lặp cho đến khi i == N
lw $a0, ($t1) # in giá trị phần tử i ra màn hình
li $v0, 1
syscall
addi $t2, $t2, 1 # i++
addi $t1, $t1, 4 # tăng địa chỉ chứa trong $t1 lên 4
j print_loop # lặp
print_loop_end:
```

Câu hỏi: trong lệnh addi $t1, $t1, 4 thì tại sao là hằng số 4? Nếu giá trị khác thì điều gì sẽ xảy ra?

*** Nhập mảng**

```
sw $t0, ($t1) # lưu giá trị $t0 vào vị trí địa chỉ là giá trị của $t1 trong
bộ nhớ
sw $t2, -12($t3) # lưu giá trị $t2 vào vị trí địa chỉ là giá trị của $t3 -
12 trong bộ nhớ.
```

Bài tập:

1. Khai báo chuỗi kí tự "Hello TDT" và xuất từng kí tự ra màn hình, xuống dòng cho mỗi kí tự.
2. Khai báo mảng chứa 10 số nguyên, tìm số lớn nhất và số bé nhất rồi in ra màn hình.
3. Viết đoạn code khai báo mảng có tối đa 10 phần tử. Cho phép người dùng nhập số phần tử (1 <= N <= 10) và nhập từng giá trị vào mảng.
4. Sắp xếp mảng có 10 phần tử đã khai báo sẵn.
5. Hoán vị 2 phần tử a và b của một mảng có 10 phần tử đã khai báo sẵn với a, b nhập từ bàn phím. Chương trình cũng cần kiểm tra tính hợp lệ của giá trị a, b . Sau đó xuất mảng ra màn hình.
6. Mảng 2 chiều n hàng, m cột: hãy đề xuất cách quản lý bộ nhớ, nhập và xuất mảng.