Tổng quát về hợp ngữ và kiến trúc MIPS

1. Dữ liệu

Lưu trữ:

- Byte (8 bits), nữa word (2 bytes), word (4 bytes)
- Một lệnh sử dụng 32 bits để lưu trữ
- Một ký tự (character) sử dụng 1 byte để lưu trữ
- Một số nguyên sử dụng 1 word (bytes) để lưu trữ

Cách viết số, chữ và chuỗi:

- Số nguyên hệ 10 viết bình thường, số hệ 16 đặt 0x vào đầu

  Ví dụ: Số 17 hệ 10: 17

  Số 17 hệ 16: 0x17

- Các ký tự được đặt trong dấu nháy đơn

  Ví dụ: 'b'

- Một chuỗi được đặt trong dấu nháy kép

  Ví dụ: "Đại học Công Nghệ Thông Tin"

Thanh ghi:

- Có 32 thanh ghi đa dụng (general-purpose registers)
- Thanh ghi được viết với ký tự $ đi trước. Mỗi thanh ghi đều có một chỉ số và một tên đi kèm, vì thể có thể sử dụng hai cách gọi thanh ghi:
  - ✓ Sử dụng chỉ số cụ thể của từng thanh ghi: từ $0 tới $31
  - ✓ Sử dụng tên gọi của từng ghi, ví dụ: $t1, $sp
- Hai thanh ghi đặc biệt "Lo" và "Hi" được sử dụng để lưu trữ kết quả của phép nhân và chia. Truy xuất dữ liệu từ hai thanh ghi này sử dụng *mfhi* ("move from Hi") and *mflo* ("move from Lo")

Lưu ý: Các thanh ghi nên sử dụng theo quy ước như trong hình 1.

- ✓ *Thanh ghi $at phục vụ cho assembler, thanh ghi $k0, $k1 phục vụ cho hệ điều hành*
- ✓ *Các thanh ghi $gp, $sp, $fp, $ra được dùng trong các mục đích đặc biệt, không nên sử dụng như biến khi lập trình*

NAME	NUMBER	USE
$zero	0	The Constant Value 0
$at	1	Assembler Temporary
$v0-$v1	2-3	Values for Function Results and Expression Evaluation
$a0-$a3	4-7	Arguments
$t0-$t7	8-15	Temporaries
$s0-$s7	16-23	Saved Temporaries
$t8-$t9	24-25	Temporaries
$k0-$k1	26-27	Reserved for OS Kernel
$gp	28	Global Pointer
$sp	29	Stack Pointer
$fp	30	Frame Pointer
$ra	31	Return Address

Hình 1. Các thanh ghi và quy ước sử dụng

Stack:

- Stack phát triển từ bộ nhớ cao tới bộ nhớ thấp

2. Cấu trúc chương trình

Dữ liệu khai báo được đặt sau nhãn **.data** và sẽ được lưu trong bộ nhớ chính (main memory – RAM)

Code đặt sau nhãn **.text**

Điểm bắt đầu của chương trình tại **main**

Các ghi chú/chú giải đặt sau dấu "#"

Mẫu của chương trình:

```
# Comment giving name of program and description of function
# Template.s
# Bare-bones outline of MIPS assembly language program

            .data       # variable declarations follow this line
                        # ...

            .text       # instructions follow this line

main:                   # indicates start of code (first instruction to execute)
            ...         # ...
            ...
            ...
```

- Như thế nào để khai báo biến sau .data

Việc khai báo biến được sử dụng theo công thức sau:

```
            name:       storage_type    value(s)
```

Ví dụ:

```
# Tạo 1 biến số nguyên tên var1, chiếm 1 word lưu trữ, với giá trị khởi tạo bằng 3.

        var1:           .word   3

# Tạo một mảng tên array1 có hai phần tử, mỗi phần tử chiếm 1 byte, được khởi tạo với hai ký tự 'a' và 'b'.

        array1:         .byte   'a','b'

# Tạo một mảng tên array2 có 40 bytes liên tục nhau với lưu trữ chưa khởi tạo (có thể xem đây là 1 mảng của 40 ký tự hoặc 10 số nguyên, v.v.)

        array2:         .space  40

# Tạo một chuỗi tên myString với khởi tạo "a string"

        myString:       .asciiz  "a string"
```

Lưu ý: Việc khai báo có thể bỏ qua tên biến, dữ liệu vẫn được cấp phát bình thường. Viêc truy xuất dữ liệu sẽ không dùng tên mà dùng trực tiếp địa chỉ nơi dữ liệu được cấp phát.
Ví dụ:

```
        .data
        .word 13, 14
        .space 16
        .byte 2, 4, 1, 5
```

Giả sử quy định cho .data bắt đầu từ địa chỉ 0x10000, định địa chỉ theo byte, một word 4 bytes, ngay lập tức word 0x10000 sẽ nhận giá trị 13, word 0x10004 sẽ nhận giá trị 14. Tiếp theo, 4 word, tức 16 byte từ word 0x10008 đến 0x10014 dành cho cấp phát .space. Cuối cùng word 0x10018 được cấp phát với từng byte tương ứng 2, 4, 1, 5.

Tài liệu tham khảo
http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm