MIPS Assembly Language

1 Chương trình MIPS đơn giản

```
################################################################
#   Description:
#       Simple example program
################################################################

################################################################
#   Main program
################################################################

# Variables for main
  .data
greeting: .asciiz "Hello, world!\n"

# Main body
  .text
main:
  li      $v0, 4
  la      $a0, greeting
  syscall
```

Một chương trình MIPS đơn giản có 2 phần, phần .data dùng để khai báo dữ liệu dùng trong chương trình, phần .text dùng để viết lệnh, hiện thực các giải thuật của chương trình.

2 Memory segments

Bộ nhớ được chia ra thành nhiều phần: phần .text dùng để chứa code, phần .data dùng để chứa các biến, dữ liệu cần khai báo.

```
   .kdata
      #variables for kernel
   .ktext
      # text for kernel

   .data
      # Variables for main

   .text
      # Main body
      # Your code start here
```

3 Cấu trúc lệnh

Một lệnh MIPS được cấu trúc như sau:

```
label:  opcode/directive    operand, operand, operand    # comment
```

Ví dụ của 1 lệnh cộng thanh ghi t2 và t3 chứa vào thanh ghi t1

```
KTMT:  add $t1, $t2, $t3    # add t1 = t2 + t3
```

3.1 Label (nhãn)

- Nhãn là một thành phần không bắt buộc của lệnh. Nếu dùng nhãn thì nó phải đứng ở đầu câu lệnh và kết thúc bằng một dấu ":" (không có khoảng trắng). Dấu ":" không phải là 1 phần của nhãn, ":" chỉ dùng để phân biệt nó là một nhãn.
- Mỗi nhãn đại diện cho một địa chỉ bộ nhớ trong hợp ngữ. Nó có thể là địa chỉ dữ liệu trong vùng nhớ dữ liệu hoặc là địa chỉ lệnh trong vùng text.

- Nhãn đại diện cho địa chỉ dữ liệu/lệnh trên cùng một dòng hoặc ở dòng tiếp theo nếu nhãn đứng 1 mình.
- Nhãn định nghĩa trong vùng .data xem như là tên biến. Nó phải bắt đầu bằng chữ cái hoặc gạch dưới "_". Nó chỉ chứa chữ cái, gạch dưới, và các con số. Tên biến không được đặt trùng với keyword.
- Nhãn định nghĩa trong vùng .text đại diện cho địa chỉ của lệnh. Nó dùng làm đối số cho các lệnh nhảy (jump), rẽ nhánh (Branch), hoặt return về tại vị trí lệnh đó.

3.2 Comments

- Comment dòng bắt đầu bằng ký tự "#" đến hết dòng.

  Example: `# this is a comment`

- Comment khối(block) đơn giản là gộp nhiều comment dòng lại với nhau.

  Example:
  ```
  ################################################################
  # This is a block comment
  ################################################################
  ```

3.3 opcode

Opcode (mã lệnh) tham khảo tập lệnh "MIPS32™ Architecture For Programmers Volume II: The MIPS32™ Instruction Set"

3.4 operand

Operand (toán hạng) có thể là thanh ghi (register) hoặc toán hạng trực tiếp (immediate). Thanh ghi tham khảo ở bảng 1.
Chú ý: Operand 1 luôn là thanh ghi.

Bảng. 1: Danh sách 32 thanh ghi

REGISTERS		
0	zero	Always equal to zero
1	at	Assembler temporary; used by the assembler
2-3	v0-v1	Return value from a function call
4-7	a0-a3	First four parameters for a function call
8-15	t0-t7	Temporary variables; need not be preserved
16-23	s0-s7	Function variables; must be preserved
24-25	t8-t9	Two more temporary variables
26-27	k0-k1	Kernel use registers; may change unexpectedly
28	gp	Global pointer
29	sp	Stack pointer
30	fp/s8	Stack frame pointer or subroutine variable
31	ra	Return address of the last subroutine call

3.5 Directives

- Directives (tạm dịch là chỉ thị) nó không phải là lệnh, do đó nó không được thực thi trong quá trình chạy. Nó thông báo cho trình biên dịch hợp ngữ phải xử lý một số việc trong khi chuyển từ ngôn ngữ cấp cao sang mã máy như: cấp phát không gian vùng nhớ, khởi tạo các giá trị trong vùng nhớ ...
- Directives dùng ký tự '.' ở đầu để phân biệt với các lệnh.

Bảng. 2: Các directive thông dụng trong MIPS

Directive	Mô tả
.data <addr>	Chỉ định phần theo sau sẽ nằm trong vùng .data cho đến khi gặp .text, .ktext, .kdata hoặc cuối file. Nếu thông số add được đưa vào, nó dùng để xác định địa chỉ bắt đầu trong vùng data mà người lập trình muốn sử dụng
.text <addr>	Chỉ định phần theo sau sẽ nằm trong vùng .text cho đến khi gặp .data, .ktext, .kdata hoặc cuối file. Nếu thông số add được đưa vào, nó dùng để xác định địa chỉ bắt đầu trong vùng data mà người lập trình muốn sử dụng
.kdata <addr>	Chỉ định phần theo sau sẽ nằm trong vùng .data của kernel cho đến khi gặp .text, .ktext, .data hoặc cuối file. Nếu thông số add được đưa vào, nó dùng để xác định địa chỉ bắt đầu trong vùng data mà người lập trình muốn sử dụng
.ktext <addr>	Chỉ định phần theo sau sẽ nằm trong vùng .text của kernel cho đến khi gặp .data, .text, .kdata hoặc cuối file. Nếu thông số add được đưa vào, nó dùng để xác định địa chỉ bắt đầu trong vùng data mà người lập trình muốn sử dụng
.word	32-bit integer
.half	16-bit integer
.byte	08-bit integer
.float	số thực chính xác đơn IEEE 754
.double	số thực chính xác kép IEEE 754
.space	Cấp phát số byte trống trong bộ nhớ
.ascii	Khai báo chuỗi/phần tử là mã ascii và không có ký tự kết thúc chuỗi theo sau.
.asciiz	Khai báo chuỗi/phần tử là mã ascii, tự động thêm ký tự kết thúc chuỗi theo sau.
.align	canh lề địa chỉ cho phần tử tiếp theo.

3.6 Các ví dụ về việc dùng directives

```
   .data
age:    .word 30      # khai bao integer 32-bit, khoi tao gia tri la 30
class:  .half 0x10    # khai bao integer 16-bit, khoi tao gia tri 0x10 (HEX)
grade:  .byte 08      # Khai bao 1 byte, khoi tao gia tri 8 (OCT)
        .align  2     # Canh lai dia chi memory:
                      #  0 byte     : dia chi tiep theo chia het cho 1
                      #  1 half word: dia chi tiep theo chia het cho 2
                      #  2 word     : dia chi tiep theo chia het cho 4

gpa:    .float  3.65  # 32-bit floating point

# Mang 6 so nguyen
ArrayInt:   .word   1, 2, 3, 54, 24, 120

# Mang char
SubName:       .byte 'C', 'A', '2', '0', '1', '6'

# Cap phat vung nho 1024 byte
ArrayBytr:  .space  1024    # 1 KB buffer

# Chuoi "Hello, world!" CO ky tu ket thuc chuoi theo sau (ASCIIZ)
Greeting:  .asciiz "Hello, world!"

# chuoi KHONG co ky tu ket thuc chuoi theo sau (ASCII)
String:   .ascii  "This string is without a null byte"
   .text
```