

## Laboratory Exercise 2

### Instruction Set, Basic Instructions, Directives

#### Goals

Sau bài thực hành này, sinh viên sẽ nắm được nguyên lý cơ bản về tập lệnh của bộ xử lý MIPS; sử dụng được các lệnh hợp ngữ cơ bản và sử dụng công cụ gỡ rối để kiểm nghiệm lại các kiến thức về tập lệnh và hợp ngữ. Sinh viên cũng thành thạo với các chỉ thị biên dịch (Directives) để công cụ MARS có thể dịch hợp ngữ thành mã máy một cách đúng đắn.

#### Literature

- Tài liệu tóm tắt về Kiến trúc MIPS<sup>1</sup>, file pptx
- Bảng tra cứu tập lệnh MIPS<sup>2</sup>, file .doc

#### Assignments at Home and at Lab

##### Home Assignment 1

Đọc tài liệu về Kiến trúc MIPS và ghi nhớ các kiến thức cơ bản sau

- Tên và ý nghĩa của 32 thanh ghi
- Các thanh ghi đặc biệt PC, HI, LO
- Khuôn dạng của 3 loại lệnh I, J, R

#### MiniMIPS Instruction Formats

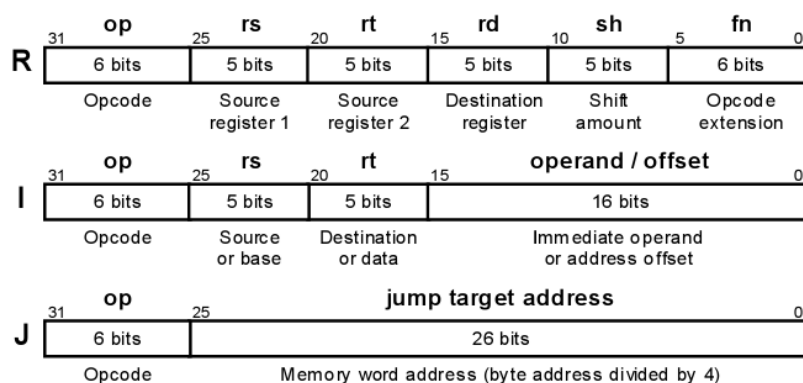


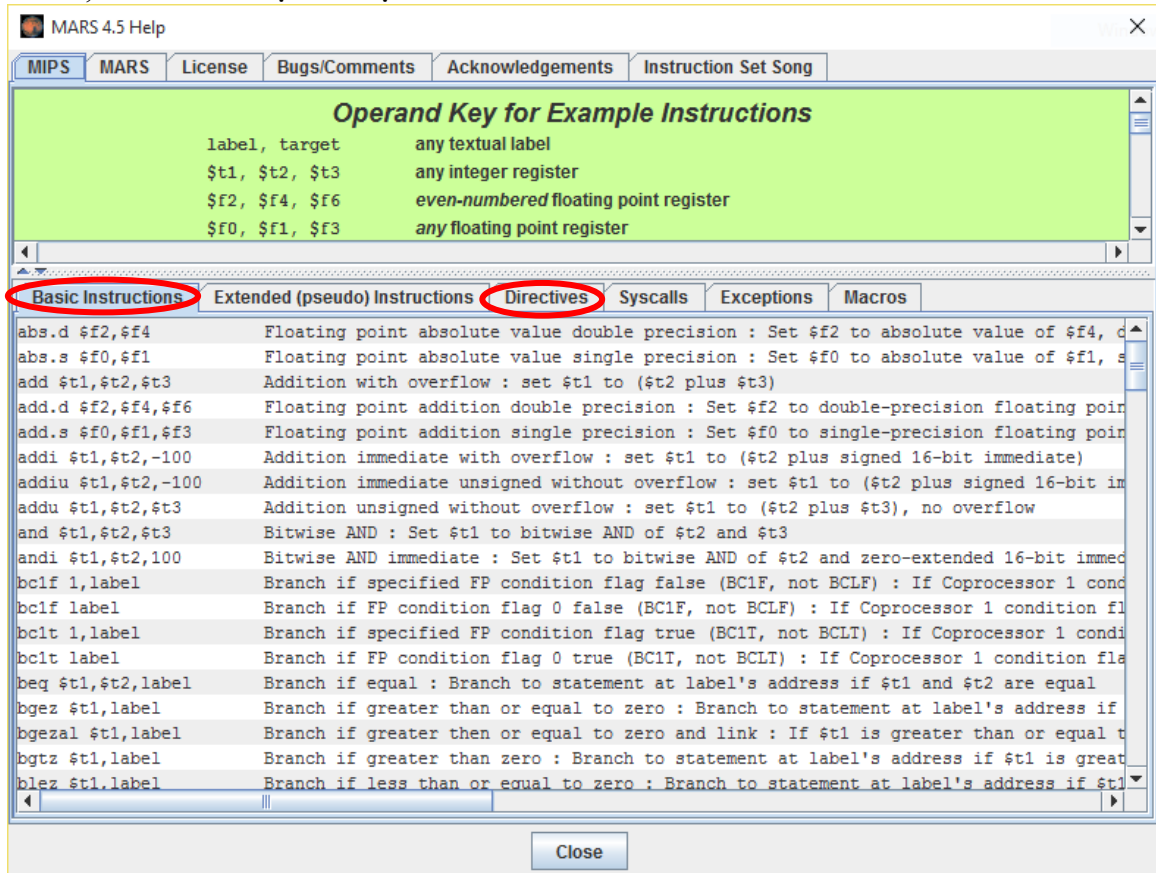
Figure 5.4 MiniMIPS instructions come in only three formats: register (R), immediate (I), and jump (J).

<sup>1</sup> Download tại: [ftp://dce.hust.edu.vn/tiennd/ict4/References/Kien truc MIPS.pptx](ftp://dce.hust.edu.vn/tiennd/ict4/References/Kien%20truc%20MIPS.pptx)

<sup>2</sup> Download tại: <ftp://dce.hust.edu.vn/tiennd/ict4/References/MIPS32-InstructionSet-QuickReference.pdf>

## Home Assignment 2

Sử dụng công cụ MARS, tra cứu Help và tìm hiểu về các lệnh cơ bản trong MIPS, và các chỉ thị biên dịch




### Assignment 1: lệnh gán số 16-bit

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 1
.text
    addi    $s0, $zero, 0x3007 # $s0 = 0 + 0x3007 = 0x3007 ;I-type
    add     $s0, $zero, $0      # $s0 = 0 + 0 = 0 ;R-type
```

Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
  - Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
    - o Sự thay đổi giá trị của thanh ghi \$s0
    - o Sự thay đổi giá trị của thanh ghi \$pc
  - Ở cửa sổ Text Segment, hãy so sánh mã máy của các lệnh trên với khuôn dạng lệnh để chứng tỏ các lệnh đó đúng như tập lệnh đã qui định
  - Sửa lại lệnh lui như bên dưới. Chuyện gì xảy ra sau đó. Hãy giải thích
- ```
addi    $s0, $zero, 0x2110003d
```

### Assignment 2: lệnh gán số 32-bit

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 2
```

```
.text
    lui    $s0, 0x2110          #put upper half of pattern in $s0
    ori    $s0, $s0, 0x003d     #put lower half of pattern in $s0
```

Sau đó:



- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại,
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
  - o Sự thay đổi giá trị của thanh ghi \$s0
  - o Sự thay đổi giá trị của thanh ghi \$pc
- Ở cửa sổ Data Segment, hãy click vào hộp combo để chuyển tới quan sát các byte trong vùng lệnh .text.
  - o Kiểm tra xem các byte đầu tiên ở vùng lệnh trùng với cột nào trong cửa sổ Text Segment.

### Assignment 3: lệnh gán (giả lệnh)

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 3
.text
    li      $s0, 0x2110003d #pseudo instruction=2 basic instructions
    li      $s1, 0x2        #but if the immediate value is small, one ins
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?

### Assignment 4: tính biểu thức $2x + y = ?$

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 4
.text
    # Assign X, Y
    addi    $t1, $zero, 5      # X = $t1 = ?
    addi    $t2, $zero, -1     # Y = $t2 = ?
    # Expression Z = 2X + Y
    add     $s0, $t1, $t1      # $s0 = $t1 + $t1 = X + X = 2X
    add     $s0, $s0, $t2      # $s0 = $s0 + $t2 = 2X + Y
```

Sau đó:



- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại,
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
  - o Sự thay đổi giá trị của các thanh ghi
  - o Sau khi kết thúc chương trình, xem kết quả có đúng không?
- Ở cửa sổ Text Segment, xem các lệnh **addi** và cho biết điểm tương đồng với hợp ngữ và mã máy. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh I




```
Y : .word    -1           # Variable Y, word type, init value =
Z : .word                    # Variable Z, word type, no init value

.text                # DECLARE INSTRUCTIONS
# Load X, Y to registers
la    $t8, X          # Get the address of X in Data Segment
la    $t9, Y          # Get the address of Y in Data Segment
lw    $t1, 0($t8)      # $t1 = X
lw    $t2, 0($t9)      # $t2 = Y

# Calculate the expression Z = 2X + Y with registers only
add   $s0, $t1, $t1    # $s0 = $t1 + $t1 = X + X = 2X
add   $s0, $s0, $t2    # $s0 = $s0 + $t2 = 2X + Y

# Store result from register to variable Z
la    $t7, Z          # Get the address of Z in Data Segment
sw    $s0, 0($t7)      # Z = $s0 = 2X + Y
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment.
  - o Lệnh **la** được biên dịch như thế nào?
- Ở cửa sổ Label và quan sát địa chỉ của X, Y, Z.
  - o So sánh chúng với hằng số khi biên dịch lệnh **la** thành mã máy
  - o Click đúp vào các biến X, Y, Z để công cụ tự động nhảy tới vị trí của biến X, Y, Z trong bộ nhớ ở cửa sổ Data Segment. Hãy bảo đảm các giá trị đó đúng như các giá trị khởi tạo.
- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
  - o Sự thay đổi giá trị của các thanh ghi
  - o Xác định vai trò của lệnh **lw** và **sw**
- Ghi nhớ qui tắc xử lý
  - o Đưa tất cả các biến vào thanh ghi bằng cặp lệnh **la, lw**
  - o Xử lý dữ liệu trên thanh ghi
  - o Lưu kết quả từ thanh ghi trở lại biến bằng cặp lệnh **la, sw**
- Tìm hiểu thêm các lệnh **lb, sb**