4 lệnh add, sub, addu, subu dùng để cộng/trừ giá trị của 2 thanh ghi, và lưu kết quả vào thanh ghi đích

2 lệnh addi, addiu dùng để cộng một thanh ghi với 1 hằng số, rồi lưu vào thanh ghi đích

add $s0, $s1, $s2 # $s0 = $s1 + $s2

sub $s0, $s1, $s2 # $s0 = $s1 - $s2

addi $s0, $s0, 123 # $s0 = $s0 + 123

addi $s0, $s2, -123 # $s0 = $s2 - 123

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]\

Có 3 lệnh: and, or, nor. NOR là thao tác “NOT OR”: A nor B = not (A or B). Cú pháp của 3 lệnh này tương tự như lệnh add ở trên.

Tương tự, ta cũng có lệnh andi và ori để tính AND/OR của một thanh ghi với một hằng số.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]\

“Địa chỉ vùng nhớ cần truy cập phải chia hết cho kích thước cần truy cập”. Ví dụ, đọc 4 byte bắt đầu từ ô nhớ có địa chỉ 10 là không hợp lệ.

\_MIPS lưu trữ dữ liệu theo dạng Big Endian, tức là byte cao sẽ được lưu ở địa chỉ thấp. Ví dụ, số 12345678h (thập lục phân) khi được lưu trong bộ nhớ thì byte đầu tiên sẽ là 12h, byte tiếp theo là 34,…

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]\

* lw (load word), lh (load halfword), lb (load byte): Đọc 4/2/1 byte. Đối với lh và lb, vì thanh ghi có độ dài 4 byte, nhiều hơn lượng dữ liệu đọc được nên các bit trống sẽ được gán bằng bit dấu của số đọc được.
* lhu (load halfword unsigned), lbu (load byte unsigned): tương tự như trên, tuy nhiên các bit trống được gán bằng 0.
* sw (store word), sh (store halfword), sb (store byte): lưu 4/2/1 byte dữ liệu trong thanh ghi vào bộ nhớ.

tên lệnh r1, offset(r2)

Trong đó:

* r1: thanh ghi cần nạp dữ liệu vào / lấy dữ liệu ra.
* r2: thanh ghi lưu địa chỉ gốc.
* offset: hằng số nguyên (16 bit), giá trị này sẽ được cộng với giá trị của r2 để được địa chỉ cần nạp vào / lấy ra.

Một số lưu ý:

* Các lệnh trên đều phải tuân theo quy tắc Alignment Restriction ở trên.
* Đối với sh và sb sẽ lưu các byte thấp trong thanh ghi vào bộ nhớ.
* Dữ liệu trong thanh ghi và bộ nhớ đều tuân theo quy tắc Big Endian.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]\

Để so sánh lớn hơn/bé hơn, MIPS đưa thêm lệnh slt (set on less than). Cú pháp:

slt rt, rs, rd

Với rt, rs, rd là các thanh ghi. Lệnh này sẽ gán rt bằng 1 khi rs < rd, bằng 0 trong trường hợp ngược lại.

So sánh trong lệnh trên là so sánh có dấu (bù 2). Để so sánh không dấu, MIPS hỗ trợ lệnh stlu, cách dùng tương tự như trên.

Ngoài ra, cũng có lệnh để so sánh với một hằng số, là slti và sltiu. Cú pháp tương tự như các lệnh tính toán với hằng số ở trên.

Kết hợp các lệnh đã tìm hiểu, ta có thể dịch đoạn chương trình C sau sang hợp ngữ MIPS:

int n = 1000;

int s = 0;

for (int i=1; i<n; i++) s += i;

addi $s0, $0, 1000 # n = 1000

addi $s1, $0, 0 # s = 0

addi $s2, $0, 1 # i = 1

FOR:

slt $t0, $s2, $s0 # $t0 = i < n?

bne $t0, $0, END # if !(i < n) goto END

add $s1, $s1, $s2 # s = s + i

addi $s2, $s2, 1 # i = i + 1

j FOR

END:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_--]\