Họ và tên: Đoàn Vũ Phú Minh

MSSV: 22520859

Bài 1:

Phần 1.

1.1

a.

add f, g, h

add f, f, i

add f, f, j

b.

addi f, h, 5

add f, f, g

1.2

a. 3

b. 2

1.3

a. 14

b. 10

Phần 2.

1.4

a. f = g + h

b. addi f, f, 1 ⎝ f = f + 1

add f, g, h ⎝ f = g + h

⎝ Cuối cùng f = g + h

1.5

a. 5

b. 5

Bài 2:

2.1 Lệnh hợp ngữ MIPS tương ứng:

a) f = g + h + B[4];

lw $s0, 16($s7) # Bởi vì MIPS định địa chỉ theo byte và một word chứa 4 byte

add $s0, $s0, $s1

add $s0, $s0, $s2

b) f = g – A[B[4]];

lw $t0, 16($s7) # $t0=B[4]

sll $t0, $t0, 2 # $t0=B[4] \* 4

add $t0, $t0, $s6 # $t0=&A[B[4]]

lw $s0, 0($t0)

sub $s0, $s1, $s0

2.2

a) 3

b) 5

2.3

a) 4

b) 5

2.4

Tìm lệnh C tương đương với chuỗi lệnh hợp ngữ:

add $s0, $s0, $s1 ◊ f=f+h

add $s0, $s0, $s2 ◊ f=f+h+g

add $s0, $s0, $s3 ◊ f=f+h+g+i

add $s0, $s0, $s4 ◊ f=f+h+g+i+j

a) f = f+g+h+i+j;

b) f = A[1];

(lw $s0, 4($s6): Lệnh này đọc dữ liệu của từ nhớ cách địa chỉ nền đang chứa trong thanh ghi $s6 4 byte. Nếu xem nội dung thanh ghi $s6 là địa chỉ nền của mảng A, lệnh này đọc nội dung phần tử thứ 2 của mảng, tức phần tử A[1])

2.5

a) Không đổi

b) Không đổi

2.6

a) Có 5 thanh ghi (không thể rút gọn nhỏ hơn)

b) Có 2 thanh ghi 2 (không thể rút gọn nhỏ hơn)

3.1

a. f = -g + h + B[1];

lw $s0, 4($s7)

sub $s0, $s0, $s1

add $s0, $s0, $s2

b. f = A[B[g] + 1];

# tìm B[g]

sll $t0, $s1, 2

add $t0, $t0, $s7

lw $t0, 0($t0) # $t0 = B[g]

# tính B[g] + 1

addi $t0, $t0, 1 # $t0 = B[g] + 1

# tìm A[B[g] + 1]

sll $t0, $t0, 2

add $t0, $t0, $s6

lw $s0, 0($t0)

3.2 Số lệnh MIPS cần để có chức năng tương đương với từng câu lệnh C:

a. 4 lệnh MIPS

b. 4 lệnh MIPS

3.3 Số thanh ghi khác nhau được dùng cho từng câu lệnh C bên trên:

a. 3 thanh ghi ($t0, $t1, $t2)

b. 3 thanh ghi ($t0, $t1, $s0)

Bài 4:

4.1

a. $s0 = 0x70000000;

$s1 = 0x0FFFFFFF;

add $t0, $s0, $s1

Sau khi thực hiện câu lệnh trên, $t0 = 0x7FFFFFFF

Kết quả này đúng như mong muốn, không tràn.

b. $s0 = 0x40000000;

$s1 = 0x40000000;

add $t0, $s0, $s1

Sau khi thực hiện câu lệnh trên, $t0 = 0x80000000

Phép toán add được thực hiện trên số có dấu (dùng bù hai). Phép cộng trên thực hiện cộng hai số dương, nhưng kết quả 0x80000000 rõ ràng là số âm ⎝ phép toàn bị tràn

4.2

sub $t0, $s0, $s1

a) $t0 = 0x60000001, không tràn

b) $t0 = 0, không tràn.

4.3

add $t0, $s0, $s1

add $t0, $t0, $s0

a) $t0 = 0x70000000 + (0x70000000 + 0x0FFFFFFF) = 0xEFFFFFFF Tràn, bởi vì cộng hai số dương nhưng bit dấu của kết quả lại là âm, không đúng.

b) $t0 = 0x40000000 + (0x40000000 + 0x40000000) = 0xC0000000 Tràn, bởi vì cộng hai số dương nhưng bit dấu của kết quả lại là âm, không đúng.

Bài 5:

5.1 & 5.2

a) op = 101011(2) = 2B(hex)

⎝ dạng I-type

rs = 10000(2) = 16(10): vậy địa chỉ nền được lưu trong thanh ghi thứ 16, chính là thanh ghi $s0.

rt = 01011(2) = 11(10): thanh ghi thứ 11, chính là thanh ghi $t3. Immediate(16 bit) = 100(2) = 4(10)

 Lệnh assembly này là: sw $t3, 4($s0)

b) op = 100011(2) = 23(hex) ⎝ Lệnh lw

⎝ I-type instruction

rs = 01000(2) = 8(10): thanh ghi thứ 8, là thanh ghi $t0

rt = 01000(2) = 8(10): thanh ghi thứ 8, là thanh ghi $t0

Immediate (16 bit) = 1000000(2) = 64(10)

=>Lệnh assembly: lw $t0, 64($t0)

5.3

a) 0xAE0B0004

b) 0x8D080040

5.4 & 5.5

a) add $t0, $t0, $zero

add: R-type format

opcode = 0hex

rs = 8ten ($t0 là thanh ghi thứ 8/ $t0 là thanh ghi có chỉ số 8)

rt = 0ten ($zero là thanh ghi thứ 0)

rd = 8ten ($t0 là thanh ghi thứ 8)

shamt = 0ten (không sử dụng)

func = 20hex 1= 00000(2)

 Machine code: 0000 0001 0000 0000 0100 0000 0010 0000bin = 01004020hex

1. lw: I-type format

opcode = 23hex

rs = 19ten = 10011bin ($s3 là thanh ghi thứ 19)

rt = 9ten ($t1 là thanh ghi thứ 9)

immediate = 4ten

 Machine code: 1000 1110 0110 1001 0000 0000 0000 0100bin = 8E690004hex

5.6

a) op=0x0;

rs=0x8;

rt=0x0;

rd=0x8;

shamt=0x0;

func=0x20

1. op=0x23; rs=0x13; rt=0x9; imm=0x4

Bài 6:

6.1

Tìm giá trị của $t2

a) 0x57755778

b) 0xFEFFFEDE

6.2 Tìm giá trị của $t2

a) 0x0000AAAA

b) 0x0000BFCD

Bài 7:

7.1

a. “slt” làm việc vơí số có dấu: Nếu thanh ghi $t0 nhỏ nhon thanh ghi $t1 thì thanh ghi $t2 nhận giá trị là 1; nếu ngược lại $t2 nhận giá trị 0

$t0 = 1010 1101 0001 0000 0000 0000 0000 0010(2)

$t1 = 0011 1111 1111 1000 0000 0000 0000 0000

Rõ ràng $t0 là một số âm, thanh ghi $t1 là một số dương, do đó $t0 < $t1

⎝ $t2 = 1

⎝ beq điều kiện bằng không xảy ra

⎝ lệnh “j DONE” được thực hiện

⎝ $t2 = 1

b. Tương tự $t2=1

7.2

a) $t0 = 1010 1101 0001 0000 0000 0000 0000 0010(2)

 $t0 = - 1,391,460,350

Xét lệnh: slti $t2, $t0, X

Nếu $t0 < X thì $t2 nhận giá trị là 1; ngược lại $t2 nhận giá trị là 0. Vậy theo như yêu cầu đề bài, để sau lệnh này, giá trị thanh ghi $t2 = 1 thì X phải có giá trị lớn hơn $t0. Mặt khác, X chỉ có được biểu diễn tối đa trong 16 bits, có dấu theo bù 2, nên giá trị của nó không thể vượt quá 215-1 = 32767

 Vậy X từ -1,391,460,349 tới 32767

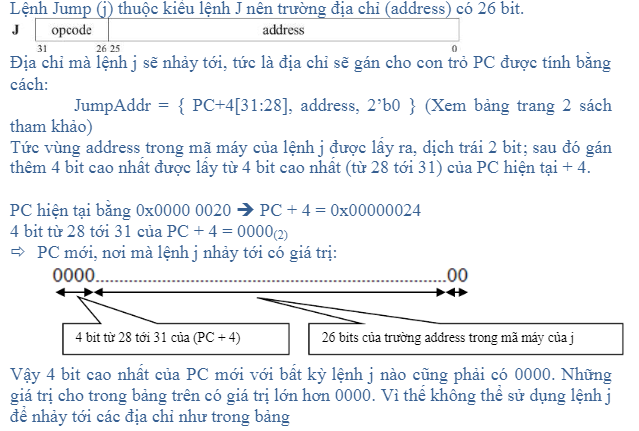
1. Tương tự, X từ 0 tới 32767

7.3

PC = 0x0000 0020

• Ta có thể sử dụng lệnh nhảy j trong hợp ngữ MIPS để con trỏ PC trỏ tới địa chỉ được ghi trong bảng trên không? Giải thích?

1. Không
2. Không



• Ta có thể sử dụng lệnh “nhảy nếu bằng” (beq) của hợp ngữ MIPS để PC trỏ tới địa chỉ được cho trong bảng trên không? Giải thích?

a) Không

b) Không

