**Bài tập chương 2:**

2.1. Cho đoạn chương trình hợp ngữ trên MIPS như sau:

# lui $s0, 0x4050

ori $s0, $s0, 251

sll $s1, $s0, 4

add $s2, $s0, $s1

# andi $s3, $s1, 0x0FE7

Hãy xác định giá trị nội dung của các thanh ghi $s0, $s1, $s2, $s3 theo dạng Hexa sau khi thực hiện đoạn chương trình trên.

# lui $s0, 0x4050

$s0: 0x40500000

ori $s0, $s0, 251

$s0: 0x405000FB

sll $s1, $s0, 4

$s1: 0x05000FB0

add $s2, $s0, $s1

$s2: 0x455010AB

# andi $s3, $s1, 0x0FE7

$s3: 0x00000FA0

2.2. Chuyển các câu lệnh C sau đây về dạng hợp ngữ MIPS. Giả thiết rằng các biến f, g, h, i, j được gán tương ứng trong các thanh ghi $s0, $s1, $s2, $s2, $s3, $s4; A và B là hai mảng dữ liệu các phần tử số nguyên 32-bit có địa chỉ cơ sở tương ứng nằm trong các thanh ghi $s6 và $s7.

1. f = - g - A[4]
2. B[8] = A[i-j]

a)

sub $t0, $zero, $s1

lw $t1, 16($s6)

sub $s0, $t0, $t1

b)

sub $t0, $s3, $s4 # t0 = i - j

sll $t0, $t0, 2 # t0 = t0 \* 4

add $t0, $t0, $s6 # t0 = address(A[i-j])

lw $t0, 0($t0) # t0 = A[i-j]

sw $t0, 32($s7) # B[8] = A[i-j]

2.3. Hãy viết lời giải thích cho các lệnh của các đoạn chương trình hợp ngữ MIPS dưới đây, rồi chuyển đoạn chương trình sang câu lệnh C tương ứng. Lời giải thích cho hợp ngữ cần sát với ý nghĩa của câu lệnh C. Giả thiết rằng các biến f, g, h, i, j được gán tương ứng trong các thanh ghi $s0, $s1, $s2, $s2, $s3, $s4; A và B là hai mảng dữ liệu các phần tử số nguyên 32-bit có địa chỉ cơ sở tương ứng nằm trong các thanh ghi $s6 và $s7.

a.

sll $s2, $s4, 1 # h = 2 \* j

add $s0, $s2, $s3 # f = h + i

add $s0, $s0, $s1 # f = f + g

f = g + i + 2\*j;

b.

sll $t0, $s0, 2 # t0 = 4\*f

add $t0, $s6, $t0 # t0 = Address(A[f])

sll $t1, $s1, 2 # t1 = 4\*g

add $t1, $s7, $t1 # t1 = Address(B[g])

lw $t3, 0($t0) # t3 = A[f]

addi $t2, $t0, 4 # t2 = Address(A[f+1])

lw $t4, 0($t2) # t4 = A[f+1]

add $t5, $t3, $t4 # t5 = A[f]+A[f+1]

sw $t5, 0($t1) # B[g] = A[f]+A[f+1]

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

2.4. Cho đoạn chương trình vòng lặp viết bằng hợp ngữ của MIPS sau đây:

addi $t1, $zero, 8 #t1=8

add $s2, $zero,$zero #s2=0

LOOP: slt $t2, $zero, $t1 #t1 > 0 ?

beq $t2, $zero, DONE

addi $s2, $s2, 3

sll $s2, $s2, 1

addi $t1, $t1, -1

j LOOP

DONE:

a. Tính số lệnh được thực hiện khi chạy đoạn chương trình trên. 52

b. Xác định giá trị thanh ghi $s2 sau khi thực hiện đoạn chương trình trên. 1530

2.5. Cho đoạn chương trình vòng lặp viết bằng hợp ngữ của MIPS sau đây:

LOOP: slt $t2, $zero, $t1

bne $t2, $zero, ELSE

j DONE

ELSE: addi $s2, $s2, 2

addi $t1, $t1, -1

j LOOP

DONE:

1. Giả thiết các thanh ghi $t1, $s2 được khởi tạo các giá trị ban đầu là $t1 = 18, $s2 = 0, hãy xác định giá trị thanh ghi $s2 sau khi thực hiện đoạn chương trình trên. s2 = 36 / 0x24
2. Với vòng lặp hợp ngữ trên, giả sử thanh ghi $t1 được khởi tạo giá trị bằng N (với N nguyên dương), hãy xác định khi thực hiện đoạn chương trình trên thì có bao nhiêu lệnh được thực hiện? 6N+3

2.6. Cho đoạn mã C sau đây:

while (i < 10) {

M[i] = a + i;

i += 1;

}

Hãy dịch đoạn mã C ở trên thành đoạn mã hợp ngữ của MIPS với số lệnh tối thiểu (*yêu cầu không sử dụng lệnh giả*). Trong đó M là mảng dữ liệu các phần tử 32-bit. Giả thiết rằng các giá trị i, a nằm thanh ghi $s0, $s1 tương ứng, thanh ghi $s2 chứa địa chỉ cơ sở của mảng M. (*Lưu ý: các lệnh của hợp ngữ cần viết lời giải thích*).

LOOP:

addi $t0, $s0, -10 # t0 = i – 10

beq $t0, $zero, DONE

sll $t0, $s0, 2 # t0 = 4 \* i

add $t0, $t0, $s2 # t0 = address(M[i])

add $t1, $s0, $s1 # t2 = a + i

sw $t1, 0($t0) # M[i] = a + i

j LOOP

DONE:

2.7. Cho M là mảng các phần tử số nguyên có dấu 32-bit, có địa chỉ cơ sở nằm trong thanh ghi $s1 và số phần tử của mảng M được cho trong thanh ghi $s2. Viết đoạn chương trình hợp ngữ MIPS (yêu cầu không sử dụng lệnh giả) để tìm phần tử lớn nhất của mảng M cất vào thanh ghi $t0 theo thuật giải viết bằng C dưới đây. (*Lưu ý: lời giải thích cho các lệnh hợp ngữ cần mô tả sát với thuật giải*).

max = M[0];

for (i=1; i < n; i ++) {

if ( M[i] > max ) max = M[i];

}

lw $t0, 0($s1) # t0 = M[0]

addi $t1, $zero, 1 # i = 1

LOOP:

slt $t2, $t1, $s2 # Check i < n

beq $t2, $zero, DONE

sll $t2, $t1, 2 # $t2 = 4 \* i

add $t2, $t2, $s2 # $t2 = Address(M[i])

lw $t2, 0($t2) # $t2 = M[i]

IF:

slt $t3, $t0, $t2 # Check max < M[i]

beq $t3, $zero, ELSE

add $t0, $zero, $t2 # max = M[i]

ELSE:

addi $t1, $t1, 1

j LOOP

DONE: