Họ tên: Nguyễn Long Nhật

MSSV: 20176840

**BÁO CÁO BÀI TẬP CHƯƠNG 2**

Exercise 2.1:

Đoạn chương trình cần thực thi :

lui $s0, 0x4050

ori $s0, $s0, 251

sll $s1, $s0, 4

add $s2, $s0, $s1

andi $s3, $s1, 0x0FE7

Giá trị của các thanh ghi (dạng Hexa) sau khi thực hiện đoạn chương trình trên

\* $s0 = 0x405000fb

Giải thích : 251(10) = 00fb(16)

Lệnh lui lưu 16 bit cao, lệnh ori lưu 16 bit thấp => 0x405000fb

\* $s1 = 0x05000fb0

Giải thích : Lệnh sll là lệnh dịch trái, ở phần i là 4 nên dịch 4 bit, mà cứ 4 bit lại biểu diễn 1 số trong 8 số trong số 32 bit, nên khi dịch bit sang trái 4 bit đơn giản là xóa số đầu và thêm số 0 vào cuối.

\* $s2 = 0x455010ab

Giải thích : Lệnh add thực hiện phép cộng 2 thanh ghi $s0 và $s1 lưu vào thanh ghi $s2

\* $s3 = 0x00000fa0

Giải thích : Lệnh andi thực hiện logic AND giữa $s1 và 0x00000fe7 và lưu vào $s3 , vậy 5 chữ số đầu nghiễm nhiên là 00000, vì AND với 0 luôn là 0, và cho

0xfe7 (1111 1110 0111) AND 0xfb0 (1111 1011 0000) ta được 0xfa0(1111 1010 0000)

Exercise 2.2:

a. f = - g - A[4]

Chương trình MIPS được viết lại như sau:

#Exercise 2a

.data

.text

sub $s0,$0,$s1 # f = -g

lw $t0,16($s6) # read A[4]

sub $s0,$s0,$t0 #f = -g -A[4]

b. B[8] = A[i-j]

Chương trình MIPS được viết lại như sau:

#Exercise 2b

.data

.text

sub $t1,$s3,$s4 #t1 = i - j

sll $t1,$t1,2 #t1 = 4(i-j)

add $t1,$t1,$s6 #take address A[i-j]

lw $t0,0($t1) #load t0 = A[i-j]

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

Exercise 2.3:

a.

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

add $s0, $s2, $s3 #f = 2\*j + i

add $s0, $s0, $s1 #f = g + i + 2\*j

=> Câu lệnh C tương ứng là f = g + i + 2\*j

b.

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

add $t0, $s6, $t0 #t0 = address of A[f]

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

add $t1, $s7, $t1 #t1 = address of B[g]

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

addi $t2, $t0, 4 #t2 = address of 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]

=> Câu lệnh C tương ứng là B[g] = A[f] + A[f+1]

Exercise 2.4:

a. Số lệnh được thực hiện khi thực hiện chương trình trên: 2 + 6\*8 + 2 = 52 lệnh.

Giải thích: Ta biết rằng, 2 lệnh đầu tiên không nằm trong vòng lặp nên chỉ tính 1 lần là 2 lệnh.

Tiếp đến, xét vòng lặp loop với điều kiện lặp là t1 > 0, mà sau mỗi vòng lặp, t1 -= 1 vậy nên với giá trị ban đầu của t1 = 8, vòng lặp sẽ kết thúc sau 8 vòng lặp

Tuy nhiên, còn cần cộng them 2 lệnh kiểm tra điều kiện thoát vòng lặp nữa nên cuối cùng chúng ta có 52 vòng lặp.

b. Sau 1 lần, giá trị của thanh ghi $s2 là (0+3)\*2 = 6

Sau 2 lần, giá trị của thanh ghi $s2 là (6+3)\*2 = 18

….

Sau 8 lần, giá trị của thanh ghi $s2 là 0x5fa hay 1530

Vì chương trình kết thúc sau 8 vòng lặp nên khi kết thúc chương trình trên, thanh ghi $s2 mang giá trị 0x5fa

Exercise 2.5:

a.

Ta thấy, trong block LOOP, khi t1 > 0 thì chỉ có 2 lệnh đầu chạy và nhảy sang block else, nên có thể coi như 18 vòng lặp cho đến khi t1 <= 0 thì có 5 lệnh luôn chạy liên tục => 18\*5

Tiếp theo, khi t1 <= 0, thì toàn bộ block LOOP chạy hết và jump tới done => 3 lệnh

=> Số lệnh được thực hiện khi thực hiện chương trình trên: 18\*5 + 3 = 93 lệnh.

b.

Với vòng lặp hợp ngữ trên, giả sử thanh ghi $t1 được khởi tạo giá trị bằng N (với N nguyên dương), thì ta luôn có N vòng lặp 5 lệnh liên tục gồm 2 lệnh đầu của LOOP và 3 lệnh của ELSE. Tiếp đó là 3 lệnh kiểm tra và thoát vòng lặp

=> Số lệnh được thực hiện khi thực hiện chương trình trên với $t1 được khởi tạo giá trị bằng N là: 5\*N + 3.

Exercise 2.6:

Đoạn mã C cần chuyển :

while (i < 10) {

M[i] = a + i;

i += 1;

}

Chương trình hợp ngữ MIPS sau chuyển đổi là:

.text

LOOP:

slti $t0,$s0,10 # i < 10 ? 1:0

beq $t0,$0,EXIT # if not, exit

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

sll $t2,$s0,2 # t2 = 4\*i

add $t2,$t2,$s2 # t2 = address of M[i]

sw $t1,0($t2) # M[i] = a + i

addi $s0,$s0,1 # i += 1

j LOOP

EXIT:

Exercise 2.7

Đoạn mã C cần chuyển:

max = M[0];

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

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

}

Chương trình hợp ngữ MIPS sau chuyển đổi là:

.text

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

addi $s3,$0,1 # i = 1

LOOP:

slt $t1,$s3,$s2 # i < n ? 1:0

beq $t1,$0,EXIT # if not, exit

sll $t2,$s3,2 # t2 = 4\*i

add $t2,$t2,$s1 # t2: address of M[i]

lw $t3,0($t2) # t3 = M[i]

slt $t4,$t0,$t3 # max < M[i] ? 1:0

bne $t4,$0,MAX # if true, set MAX

addi $s3,$s3,1 # i += 1

j LOOP

MAX:

add $t0,$0,$t3 # max = M[i]

addi $s3,$s3,1 # i += 1

j LOOP

EXIT: