Lớp: IT006.O13
MSSV: 22520077 *Tên: Trần Ngọc Ánh*

BÀI TẬP CHƯƠNG 2

Bài 1:

1.1.

a) add f, g, h
add f, f, i
add f, f, j

b) addi f, h, 5
addi f, f, g

1.2.

a) 3

b) 2

1.3.

a) 14

b) 10

1.4.

a) f = g + h

b) addi f, f, 1 → f = f + 1
add f, g, h → f = g + h

1.5.

a) 5

b) 5

Bài 2:

2.1.

a) f = g + h + B[4]
lw $s0, 16($s7)
add $s0, $s0, $s1
add $s0, $s0, $s2

b) f = g – A[B[4]];
lw $t0, 16($s7)
sll $t0, $t0, 2
add $t0, $t0, $s6
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];

2.5. a) Không đổi **b)** Không đổi

2.6. a) Có 5 thanh ghi **b)** Có 2 thanh ghi

Bài 3:

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

```
lw $s0, 4($s7)
sub $s0, $s0, $s1
add $s0, $s0, $s2
```

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

```
sll $t0, $s1, 2
add $t0, $t0, $s7
lw $t0, 0($t0)
addi $t0, $t0, 1
sll $t0, $t0, 2
add $t0, $t0, $s6
lw $s0, 0($t0)
```

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) 10101110000010110000000000000100
$2b_{(hex)}$ 16 11 4
➔ dạng I-type
Lệnh assembly cho câu này là: sw $t3, 4($s0)

b) 10001101000010000000000001000000

23 8 8 64

→ dạng I-type

Lệnh assembly cho câu này là: lw $t0, 64($t0)

5.3. a) 0xAE0B004 b) 0x8D080040

5.4 & 5.5

a) add $t0, $t0, $zero → dạng R-type

opcode	rs	rt	rd	shamt	funct
0	8	0	8	0	20_{hex}

Mã máy của câu lệnh trên: 0000 0001 0000 0000 0100 0000 0010 0000

b) lw $t1, 4($s3) → dạng I-type

opcode	rs	rt	immediate
23	19	9	4

Mã máy của câu lệnh trên: 1000 1110 01110 1001 0000 0000 0000 0100

5.6

a) op = 0x0; rs = 0x8; rt = 0x0; rd = 0x8; func = 0x20

b) op = 0x23, rs = 0x13, rt = 0x9, immediate = 0x4

Bài 6:

6.1. a) 0x57755778 b) 0xFEFFFEDE

6.2. a) 0x0000AAAA b) 0x0000BFCD

Bài 7:

7.1. a) $t0 = 1010 1101 0001 0000 0000 0000 0000 $0010_{(2)}$

$t1 = 0011 1111 1111 1000 0000 0000 0000 0000

Do đó: $t0 < $t1 → $t2 = 1

→ beq điều kiện bằng không xảy ra → lệnh "j DONE" được thực hiện → $t2 = 1

b) Tương tự, $s2 = 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 $2^{15} - 1 = 32767$

→ Vậy X từ -1,391,460,349 tới 32767

b) Tương tự, X từ 0 tới 32767

Bài 8:

a) and $t3, $s0, $s2 → dạng R-type

opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
0	16	18	11	0	24_{hex}

→ 000000 10000 10010 01011 00000 100100

Mã máy theo hệ 16: 0x02125824

b) sll $t1, $t5, 7 → dạng R-type

opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
0	0	13	9	7	00_{hex}

→ 000000 00000 01101 01001 00111 000000

Mã máy theo hệ 16: 0x000D49C0

c) addi $t0, $s3, 25 → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
8	19	8	25

→ 001000 10011 01000 0000000000011001

Mã máy theo hệ 16: 0x22680019

d) addi $t0, $s3, -25 → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
8	19	8	-25

→ 001000 10011 01000 1111111111100111

Mã máy theo hệ 16: 0x2268FFE7

e) lw $t0, 24($s0) → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
23_{hex}	16	8	24

→ 100011 10000 01000 0000000000011000

Mã máy theo hệ 16: 0x8E080018

f) lw $t0, -24($s0) → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
23_{hex}	16	8	-24

→ 100011 10000 01000 1111111111101000

Mã máy theo hệ 16: 0x8E08FFE8

g) sw $t2, 48($s0) → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
$2b_{hex}$	16	10	48

→ 101011 10000 01010 0000000000110000

Mã máy theo hệ 16: 0xAE0A0030

h) sw $t2, -48($s0) → dạng I-type

opcode (6)	rs (5)	rt (5)	immediate (16)
$2b_{hex}$	16	10	-48

→ 101011 10000 01010 1111111111010000

Mã máy theo hệ 16: 0xAE0AFFD0

Bài 9:

a) 0x01304024

Chuyển mã máy (hệ 16) sang dạng nhị phân: 0000 0001 0011 0000 0100 0000 0010 0100

- opcode (6 bits đầu): 0
- funct (6 bits cuối): 24_{hex}

→ Câu lệnh có dạng R-type, lệnh and

Opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
000000	01001	10000	01000	00000	100100
0	9	16	8	0	24_{hex}

Câu lệnh MIPS tương ứng với mã trên: and $t0, $t1, $s0

b) 0x2128FFF3

Chuyển mã máy (hệ 16) sang dạng nhị phân: 0010 0001 0010 1000 1111 1111 1111 0011

- opcode (6 bits đầu): 8_{hex}

→ Câu lệnh có dạng I-type, lệnh addi

Opcode (6)	rs (5)	rt (5)	immediate (16)
001000	01001	01000	1111111111110011

8 9 8 -13

Câu lệnh MIPS tương ứng với mã trên: addi $t0, $t1, -13

c) 0xAD28FFFC

Chuyển mã máy (hệ 16) sang dạng nhị phân: 1010 1101 0010 1000 1111 1111 1111 1100

- opcode (6 bits đầu): $2b_{hex}$

→ Câu lệnh có dạng I-type, lệnh sw

Opcode (6)	rs (5)	rt (5)	immediate (16)
101011	01001	01000	1111111111111100
2b	9	8	-4

Câu lệnh MIPS tương ứng với mã trên: sw $t0, -4($t1)

Bài 10:

a) Biết trước khi chạy: $s0 = 0x12345678; $s1 = 0x00000007

Hỏi sau khi chạy xong đoạn lệnh trên, $s0, $s1, $t0, $t1 bằng bao nhiêu?

❖ ***Trả lời:***

and $t0, $s0, $s1	$t0 = $s0 + $s1 = 0
or $t1, $s0, $s1	$t1 = $s0 - $s1 = 0
nor $t0, $t0, $t1	$t0 = $t0 nor $t1 = 0 → $t1 = 305419903
sll $t0, $t0, 3	Dịch trái $t0 sang 3, → $t0 = -305419904

b) Biết trước khi chạy: $s0 = 0x0000000f

Hỏi sau khi chạy xong đoạn lệnh trên, $s0, $t0 bằng bao nhiêu?

❖ ***Trả lời:***

andi $t0, $s0, 12	$t0 = 12
nor $t0, $t0, $zero	$t0 = -13
ori $t0, $t0, 3	
srl $t0, $t0, 2	$t0 = 1073741820, $s0 = 15

c) Biết trước khi chạy: $t0 = 0x0000008f ; $t1 = 0x0000009f

Hỏi sau khi chạy xong đoạn lệnh trên, $t2 bằng bao nhiêu?

❖ ***Trả lời:***

slt $t2, $t0, $t1 beq $t2, $zero, ELSE add $t2, $t2, $t0 j DONE ELSE: add $t2, $t2, $t1 DONE:	$t2 = 1 $t2 = 144

d) Sau đoạn chương trình này thì giá trị trong thanh ghi $s0 là bao nhiêu?

❖ ***Trả lời:***

addi $s0, $zero, 2
addi $t1, $zero, 6
loop: beq $t1, $zero, end
sll $s0, $s0, 1
addi $t1, $t1, -1
j loop
end: addi $s1, $s0, 2
	$s0 = 128

Bài 11: Chuyển các đoạn lệnh C sau sang assembly của MIPS. Biết i và j tương ứng với các thanh ghi $s0 và $s1. Mảng A là mảng mà các phần tử là số nguyên, mỗi phần tử chiếm 1 từ nhớ (4 bytes) và địa chỉ nền của mảng A lưu trong thanh ghi $s3.

a)
```
if (i < j) {
        A[i] = A[i] + 1;
        A[i+1] = 5;
        }
        else {
        A[i] = A[i] - 1;
        A[i+1] = 10;
        }
    i++;
```

```
slt $t0, $s0, $s1          #i < j
beq $t0, $zero, else       #i >= j, nhảy đến else

lw $t1, 0($s3)             #$t1 = A[i]
addi $t1, $t1, 1           #$t1 = A[i] + 1
sw $t1, 0($s3)             #A[i] = $t1
addi $s3, $s3, 4           #Địa chỉ của A[i + 1]
li $t2, 5                  #$t2 = 5
sw $t2, 0($s3)             #A[i + 1] = 5
j end

else:
        lw $t1, 0($s3)     # $t1 = A[i]
        addi $t1, $t1, -1  # $t1 = A[i] - 1
        sw $t1, 0($s3)     # A[i] = $t1
        addi $s3, $s3, 4   # Địa chỉ của A[i+1]
        li $t2, 10         # $t2 = 10
        sw $t2, 0($s3)     # A[i+1] = 10

end:    addi $s0, $s0, 1   #tăng i lên 1
```

b)
```
if (i <= j && j > 0)
        A[j] = A[i] + A[i+1];
   else
        A[j] = A[i] - A[i+1];
   i++;
```

```
slt $t0, $s1, $s0          # $t0 = (j < i)
slti $t1, $s1, 0           # $t1 = (j > 0)
and $t0, $t0, $t1          # $t0 = ($t0 && $t1)
beq $t0, $zero, else

addi $t2, $s1, 0           # $t2 = Địa chỉ của A[j]
lw $t3, 0($s3)             # $t3 = A[i]
lw $t4, 4($s3)             # $t4 = A[i+1]
add $t5, $t3, $t4          # $t5 = $t3 + $t4
sw $t5, 0($t2)             # A[j] = $t5
j end

else:  addi $t2, $s1, 0    # $t2 = Địa chỉ của A[j]
       lw $t3, 0($s3)      # $t3 = A[i]
       lw $t4, 4($s3)      # $t4 = A[i+1]
       sub $t5, $t3, $t4   # $t5 = $t3 - $t4
       sw $t5, 0($t2)      # A[j] = $t5

end:  addi $s0, $s0, 1
```

c)
```
while (i > 0) {
        A[i+1] =A[i]*8;
        i--;
   }
   A[0] = 5;
```

```
loop_start:
     slt $t0, $s0, 1
     j end_loop

in_loop:
     addi $t0, $s0, 1   #lưu i + 1 vào $t0
     lw $t1, 0($s3)     #lấy giá trị của A[i] vào $t1
     sll $t1, $t1, 3    #nhân A[i] với 8
     sw $t1, 4($s3)     #lưu giá trị $t1 vào A[i +1]
     subi $s0, $s0, 1   #giảm giá trị của i đi 1
     j loop_start
```

	end_loop: li $t1, 5 #gán 5 vào A[0] sw $t1, 0($s3)

d) j = value;
for(i = 1; i < j; i++) {
A[i] = B[i];
j = 0;
}
(Với địa chỉ nền mảng B đang lưu trong thanh ghi $s4 và biến value tương ứng thanh ghi $s5)
	move $t0, $s1
	for_loop:
	slt $t0, $s0, $s1
	beq $t0, $zero, end_for
	lw $t1, 0($s4) #lấy giá trị B[i] vào $t1
	sw $t1, 0($s3)
	addi $s0, $s0, 1
	j for_loop
	
	end_for: li $s1, 0

e) j = value;
max = 0;
for(i = 0; i < j; i++) {
if(A[i] > max) max = A[i];
j = 0;
}
(Với biến max tương ứng với thanh ghi $s4)
	move $t0, $s1
	li $s4, 0
	
	for_loop:
	slt $t0, $s0, $s1
	beq $t0, $zero, end_for
	lw $t1, 0($s3)
	bge $t1, $s4, not_greate #so sánh A[i] và max
	move $s4, $t1
	
	not_greater:
	addi $s0, $s0, 1
	j for_loop
	
	end_for: li $s1, 0