BÀI TẬP CHƯƠNG 2

Kiến trúc tập lệnh

---oOo---

Các bài tập chương này được trích dẫn và dịch lại từ:

> ***Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Third Edition, 2011.**

Lưu ý: Các mảng sử dụng trong chương này đều là mảng mà mỗi phần tử chứa 1 word/từ nhớ, mỗi từ nhớ chứa 4 bytes.

Bài 1.

Phần 1.

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

b) *f = g + (h + 5);*

g, h, i, j là những số nguyên 32-bit

1.1 Hãy tìm mã hợp ngữ MIPS tương đương các lệnh C trên.

Answer:

a) add t0,g,h
   add t1, i, j
   add f, t0, t1

b) addi t0, h,5
   add f,g,t0

1.2 Có bao nhiêu lệnh MIPS để hiện thực các lệnh C trên.

Answer:

a) 3 lệnh
b) 2 lệnh

1.3 Nếu các biên *f, g, h, i* và *j* có giá trị tương ứng là 1, 2, 3, 4, 5 thì giá trị của *f* là bao nhiêu?

Answer:

a) F = 14
b) F = 10

Phần 2.

a.

add f, g, h

b.

addi f, f, 1
add f, g, h

1.4 Tìm lệnh C tương đương với các lệnh hợp ngữ MIPS trên.

Answer:

a) f = g+h;
b) f = f +1;
   f = g +h

1.5 Nếu các giá trị *f, g, h* và *i* có giá trị tương ứng 1, 2, 3 và 4 thì giá trị cuối cùng của *f* là bao nhiêu?

Answer:

a) f= 5
b) f= 5

Bài 2.

Phần 1.

a. f = g + h + B[4];
b. f = g – A[B[4]];

f, g, h, i, j được lưu lần lượt ở các thanh ghi *$s0, $s1, $s2, $s3, $s4*
Địa chỉ cơ sở/nền (base address) của mảng A và B được lưu trong các thanh ghi *$s6, $s7*.
Mỗi phần

2.1 Hãy chuyển các câu lệnh C bên trên sang dạng hợp ngữ MIPS.

Answer:

a) lw $t0,16($s7)
   (giải thích: "16" vì mỗi phần tử của mảng B có kích thước là 4 byte (vì đây là mảng các từ), và chúng ta muốn truy cập phần tử thứ 5 (bắt đầu từ 0), nên chúng ta nhân 4 (byte) với 4 (vị trí của phần tử) để có được offset là 16 byte)
   add $t0, $t0, $s2
   add $s0, $t0, $s1
b) lw $t0, 16($s7)
   sll $t0, $t0, 2
   add $t0, $t0, $s6
   lw $s0, 0($t0)
   sub $s0, $s1, $s0

2.2 Cần bao nhiêu lệnh hợp ngữ MIPS để có chức năng tương đương với từng câu lệnh C?

Answer:

a) 3
b) 5

2.3 Có bao nhiêu thanh ghi khác nhau được dùng cho từng câu lệnh C bên trên.

Answer:

a) 4

b) 5

Phần 2.

a.

add $s0, $s0, $s1
add $s0, $s0, $s2
add $s0, $s0, $s3
add $s0, $s0, $s4

b.

lw $s0, 4($s6)

2.4 Hãy tìm câu lệnh C tương đương với các câu lệnh hợp ngữ MIPS bên trên.

Answer:

a) f= f+g+h+i+j;
b) f = A[1];

2.5 Hãy thử rút gọn số lượng lệnh hợp ngữ MIPS trên nếu có thể.

Answer:

Không thay đổi

2.6 Có bao nhiêu thanh ghi được sử dụng trong đoạn hợp ngữ trên? Nếu ta có thể rút gọn số lệnh thì ta cần bao nhiêu thanh ghi?

Answer:

a) 5
b) 2
Vì không thay đổi nên số register vẫn vậy

Bài 3.

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

f, g, h, i, j được lưu lần lượt tại các thanh ghi *$s0, $s1, $s2, $s3, $s4*
Địa chỉ cơ sở của hai chuỗi A và B được lưu trong các thanh ghi *$s6, $s7*

3.1 Hãy tìm các lệnh hợp ngữ MIPS tương đương với các câu lệnh C bên trên.

Answer:

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

3.2 Cần bao nhiêu lệnh hợp ngữ MIPS để có chức năng tương đương với từng câu lệnh C?

Answer:

a) 3 lệnh
b) 7 lệnh

3.3 Có bao nhiêu thanh ghi khác nhau được dùng cho từng câu lệnh C bên trên.

Answer:

a) 4 registers
b) 5 registers

Bài 4.

Các câu hỏi dưới đây liên quan đến mở rộng dấu và tràn.

Thanh ghi *$s0* và *$s1* lưu các giá trị như bảng bên dưới. Hãy trả lời các câu hỏi liên quan đến lệnh hợp ngữ MIPS bên dưới và tính toán các kết quả.

a. $s0 = 0x70000000; $s1 = 0x0FFFFFFF
b. $s0 = 0x40000000; $s1 = 0x40000000

Ghi chú: Khi 0x trước một giá trị thì giá trị đó đang biểu diễn trong hệ 16

4.1 Tính kết quả của *$t0* sau khi thực hiện câu lệnh: *add $t0, $s0, $s1*

Kết quả trong thanh ghi *$t0* đúng như mong muốn của phép toán chưa? Có xảy ra tràn không?

Answer:

a) $t0 = 0x7fffffff (0111…11) và không tràn (31 bits 1)
b) $t0 = 0x80000000 (1000…00) bị tràn vì là số âm (31 bits 0)

4.2

Tính kết quả của $t0 sau khi thực hiện câu lệnh: *sub $t0, $s0, $s1*

Kết quả trong thanh ghi $t0 đúng như mong muốn của phép toán chưa? Có xảy ra tràn không?

Answer:

a) $t0 = 0x60000001 (01100…001) không bị tràn vì có 28 bit 0
b) $t0 = 0x00000000 không bị tràn

4.3

Tính kết quả của $t0 sau khi chạy chuỗi lệnh:

add $t0, $s0, $s1
add $t0, $t0, $s0

Kết quả trong thanh ghi $t0 đúng như mong muốn của phép toán chưa? Có xảy ra tràn không?

Answer:

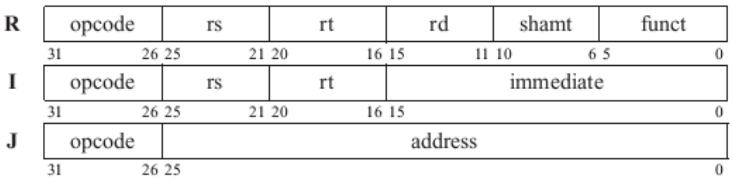
a) $t0 = 0xefffffff bị tràn vì cộng 3 số dương nhưng kết quả ra lại là số âm
b) $t0 = 0xc0000000 bị tràn, sai kết quả

Bài 5.

Chuyển các mã máy sau sang dạng hợp ngữ MIPS

a.	$1010\ 1110\ 0000\ 1011\ 0000\ 0000\ 0000\ 0100_{two}$
b.	$1000\ 1101\ 0000\ 1000\ 0000\ 0000\ 0100\ 0000_{two}$

5.1 & 5.2. Từ các giá trị binary ở bảng trên, hãy xác định chuỗi nhị phân thể hiện là lệnh gì?
Xác định các lệnh trên là thuộc kiểu lệnh gì (I-type, R-type, J-type).



Chú ý: Tham khảo "MIPS reference data" (trang 2, sách tham khảo) để dò tìm opcode của các lệnh.

Answer:

a. 101011 10000 01011 00000 00000 000100 là I-type tại op có giá trị 2b (hex)
b. 100011 01000 01000 00000 00001 000000 là I-type tại op có giá trị 23 (hex)

5.3

Nếu chuỗi nhị phân trên chỉ là dữ liệu đơn thuần. Hãy chuyển chúng sang dạng mã HEX.

Answer:

a. 0xae0b0004
b. 0x8d080040

5.4 & 5.5

Hãy dịch các lệnh sau sang dạng mã máy
a) add $t0, $t0, $zero
b) lw $t1, 4($s3)

Answer:

a. 000000 10000 00000 10000 00000 100000
b. 100011 01001 10010 000000000100

5.6 Hãy trình bày dưới dạng mã HEX của các trường opcode, Rs và Rt của các lệnh trên. Đối với các lệnh kiểu R, hãy trình bày dưới dạng mã HEX của các trường Rd và funct. Đối với các lệnh kiểu I, hãy trình bày dưới dạng mã HEX trường trực tiếp (immediate field).

Answer:

a. Rd: 10000 → hex: 10; funct: 100000 → hex: 20
b. Immediate field: 000000000100 → hex: 4

Bài 6.

Cho giá trị của các thanh ghi sau:

a.	$t0 = 0x55555555, $t1 = 0x12345678
b.	$t0 = 0xBEADFEED, $t1 = 0xDEADFADE

6.1 Hãy cho biết giá trị của thanh ghi $t2 sau khi chạy các lệnh sau:

sll $t2, $t0, 4
or $t2, $t2, $t1

Answer:

a. $t2 = 5 6789 ABC8
b. $t2 = C C98D E9AE

6.2 Hãy cho biết giá trị của thanh ghi $t2 sau khi chạy các lệnh sau:

srl $t2, $t0, 3
andi $t2, $t2, 0xFFEF

Answer:

a. $t2 = AAB AA99
b. $t2 = 17D6 BFCC

Bài 7.

Giá trị của thanh ghi $t0 được cho trong bảng bên dưới

a.	$1010\ 1101\ 0001\ 0000\ 0000\ 0000\ 0000\ 0010_{two}$
b.	$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two}$

7.1

Giả sử rằng thanh ghi $t0 chứa giá trị ở bảng trên và $t1 có giá trị

$0011\ 1111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000_{two}$

Hãy cho biết giá trị của $t2 sau khi chạy các lệnh dưới

```
        slt  $t2, $t0, $t1          # set on less than
        beq  $t2, $zero, ELSE      # go to ELSE if $t2=0
        j    DONE                  # go to DONE
ELSE:   addi $t2, $zero, 2         # $t2 = 0+2
DONE:
```

Answer:

a. Giá trị của $t2 = 0x00000001

Giải thích: vì t0 là một số âm nên nhỏ hơn t1 nên t2 = 1 dẫn đến lệnh beq không xảy ra trường hợp bằng với số 0 nên nhảy tới DONE.

b. Giá trị của t2 giống câu a và cũng tương tự câu a. t0 nhỏ hơn t1 nên t2 = 1.

7.2

Giả sử rằng thanh ghi $t0 chứa giá trị trong bảng trên và được so sánh với giá trị X bằng lệnh MIPS bên dưới. Hãy chú ý cấu trúc của lệnh slti. Tìm giá trị của X (nếu có) để $t2 có giá trị là 1.

slti $t2, $t0, X

Answer:

a. Giá trị của X phải lớn hơn giá trị của t0 thì t2 mới nhận giá trị là 1. Cụ thể t0 = 2.903.506.946 (decimal) nên X phải lớn hơn giá trị này.

b. Trong trường hợp này, không có giá trị nào của X thỏa mãn để t2 bằng 1 vì t0 = ffffffff là lớn nhất**.**

7.3

Giả sử con trỏ PC đang có giá trị 0x0000 0020.

Ta có thể sử dụng lệnh nhảy trong hợp ngữ MIPS (lệnh j) để nhảy đến địa chỉ có giá trị như trong bảng trên hay không.

Tương tự, ta có thể sử dụng lệnh nhảy-nếu-bằng (lệnh beq) để nhảy đến địa chỉ có trong bảng trên hay không.

Answer:

a. Không

b. Không

Vì beq thuộc I-type, khi beq thực hiện lệnh nhảy, PC mới = PC hiện tại + 4 (immediate <<2). Immediate chứa 16 bit, sau khi dịch trái 2 bit thành số 18 bit, giá trị lớn nhất của số 18 bit là 2^18 -1. VÌ vậy PC lớn nhất = PC hiện tại +4+2^18 -1 =0x00000020+4+2^18-1

Giá trị PC lớn nhất này nhỏ hơn giá trị trong bảng ở cả 2 câu a và b. Do đó không thể gán giá trị PC tới cacs giá trị trong bảng bằng lệnh beq

Bài 8. Tìm mã máy (biểu diễn hệ 16) cho các lệnh assembly của MIPS sau:

a. and $t3, $s0, $s2

Answer:

000000 10000 10010 01011 00000 100100

0x02125824

b. sll $t1, $t5, 7

Answer:

000000 11001 01001 00000 00111 000000

0x000D49C0

c. addi $t0, $s3, 25

Answer:

001000 10011 01000 0000000000011001

0x22680019

d. addi $t0, $s3, -25

Answer:

001000 10011 01000 1111111111100111

0x2268FFE7

e. lw $t0, 24($s0)

Answer:

100011 10000 01000 0000000000011000

0x08E08018

f. lw $t0, -24($s0)

Answer:

100011 10000 01000 1111111111101000
0x8E08FFE8

g. *sw $t2, 48($s0)*

Answer:

101011 10000 01010 0000000000110000
0x0AE0A030

h. *sw $t2, -48($s0)*

Answer:

101011 10000 01010 1111111111010000
0x2B82BFD0

Bài 9. Cho các mã máy như sau, hỏi tương ứng với từng mã máy là lệnh assembly gì của MIPS

a. *0x01304024*

Answer:

AND $t0 $t1 $s0

b. *0x2128fff3 (0x2128FFF3)* ***(Trong hệ 16, các chữ từ a tới f có thể viết thường hoặc hoa đều được)***

Answer:

ADDI $t0 $t1 0xFFF3

c. *0xad28fffc*

Answer:

SW $t0 0xFFFC $t1

Bài 10. Cho đoạn lệnh assembly sau, cho biết kết quả sau khi chạy

a. *and $t0, $s0, $s1*
or $t1, $s0, $s1
nor $t0, $t0, $t1
sll $t0, $t0, 3

Biết trước khi chạy: *$s0 = 0x12345678; $s1 = 0x00000007*
Hỏi sau khi chạy xong đoạn lệnh trên, *$s0, $s1, $t0, $t1* bằng bao nhiêu

Answer:
$s0 = 0x12345678
$s1 = 0x00000007
$t0 = 0x6e5d4c00
$t1 = 0x1234567f

b. *andi $t0, $s0, 12*
nor $t0, $t0, $zero
ori $t0, $t0, 3
srl $t0, $t0, 2

Biết trước khi chạy: *$s0 = 0x0000000f*
Hỏi sau khi chạy xong đoạn lệnh trên, *$s0, $t0* bằng bao nhiêu

Answer:
$s0 = 0x0000000f
$t0 = 0x3ffffffc

c.

```
        slt     $t2, $t0, $t1
        beq     $t2, $zero, ELSE
        add     $t2, $t2, $t0
        j       DONE
ELSE:   add     $t2, $t2, $t1
DONE:
```

Biết trước khi chạy: *$t0 = 0x0000008f* ; *$t1 = 0x0000009f*
Hỏi sau khi chạy xong đoạn lệnh trên, *$t2* bằng bao nhiêu
Answer: $t2 = 0x00000090

d.

```
        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
```

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

Answer: $s0 = 0x00000080

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++;
```

Answer:

```
slt $t0, $s0, $s1  # if (i < j)
beq $t0, $zero, Else  # if not true, jump to Else

# Then
sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]
addi $t2, $t2, 1  # A[i] = A[i] + 1
```

```
sw $t2, 0($t1)  # store back to A[i]

addi $t1, $t1, 4  # calculate &A[i+1]
li $t2, 5  # load immediate value 5
sw $t2, 0($t1)  # A[i+1] = 5
j EndIf

# Else
Else:
sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]
subi $t2, $t2, 1  # A[i] = A[i] - 1
sw $t2, 0($t1)  # store back to A[i]

addi $t1, $t1, 4  # calculate &A[i+1]
li $t2, 10  # load immediate value 10
sw $t2, 0($t1)  # A[i+1] = 10

# EndIf
EndIf:
addi $s0, $s0, 1   # i++
```

b.

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

```
Answer:
slt $t0, $s1, $s0  # if (i > j)
bne $t0, $zero, Else  # if true, jump to Else
beq $s1, $zero, Else  # if (j == 0), jump to Else

# Then
sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]

addi $t1, $t1, 4  # calculate &A[i+1]
lw $t3, 0($t1)  # load A[i+1]

add $t2, $t2, $t3  # A[i] + A[i+1]
```

```
sll $t1, $s1, 2  # calculate 4*j
add $t1, $t1, $s3  # calculate &A[j]
sw $t2, 0($t1)  # A[j] = A[i] + A[i+1]
j EndIf

# Else
Else:
sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]

addi $t1, $t1, 4  # calculate &A[i+1]
lw $t3, 0($t1)  # load A[i+1]

sub $t2, $t2, $t3  # A[i] - A[i+1]

sll $t1, $s1, 2  # calculate 4*j
add $t1, $t1, $s3  # calculate &A[j]
sw $t2, 0($t1)  # A[j] = A[i] - A[i+1]

# EndIf
EndIf:
addi $s0, $s0, 1   # i++
```

c.

```
while (i > 0){
A[i+1] =A[i]*8;
i--;
}
A[0] = 5;
Answer:
Loop:
beq $s0, $zero, ExitLoop  # if (i == 0), exit loop

sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]

sll $t2, $t2, 3  # A[i]*8

addi $t1, $t1, 4  # calculate &A[i+1]
sw $t2, 0($t1)  # A[i+1] = A[i]*8

subi $s0, $s0, 1   # i--
j Loop
```

```
ExitLoop:
sw $zero, 0($s3)  # A[0] = 0
addi $t2, $zero, 5
sw $t2, 0($s3)  # A[0] = 5
```

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)

```
Answer:
 $s1, $s5  # j = value

li $s0, 1  # i = 1

Loop:
slt $t0, $s0, $s1  # if (i < j)
beq $t0, $zero, ExitLoop  # if not true, exit loop

sll $t1, $s0, 2  # calculate 4*i
add $t2, $t1, $s3  # calculate &A[i]
add $t3, $t1, $s4  # calculate &B[i]

lw $t4, 0($t3)  # load B[i]
sw $t4, 0($t2)  # A[i] = B[i]

addi $s0, $s0, 1   # i++
j Loop

ExitLoop:
move $s1, $zero  # j = 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)

```
Answer:
move $s1, $s5  # j = value
```

```
move $s4, $zero  # max = 0

li $s0, 0  # i = 0

Loop:
slt $t0, $s0, $s1  # if (i < j)
beq $t0, $zero, ExitLoop  # if not true, exit loop

sll $t1, $s0, 2  # calculate 4*i
add $t1, $t1, $s3  # calculate &A[i]
lw $t2, 0($t1)  # load A[i]

slt $t0, $s4, $t2  # if(max < A[i])
beq $t0, $zero, ContinueLoop  # if not true, continue loop

move $s4, $t2  # max = A[i]

ContinueLoop:
addi $s0, $s0, 1   # i++
j Loop

ExitLoop:
move $s1, $zero  # j = 0
```