Bài Tập Chương 2

---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, Revised Fourth 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.
1.2 Có bao nhiêu lệnh MIPS để hiện thực các lệnh C trên.
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?

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.
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?

Trả lời:
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.

Phần 1.

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

f, g, h, i, j được lưu 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.
2.2 Cần bao nhiêu lệnh hợp ngữ MIPS để có chức năng tương đương với từng câu lệnh C?
2.3 Có bao nhiêu thanh ghi khác nhau được dùng cho từng câu lệnh C bên trên.

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

Trả lời:

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)

Bài 3.

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

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

f, g, h, i, j được lưu 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.

3.2 Cần bao nhiêu lệnh hợp ngữ MIPS để có chức năng tương đương với từng câu lệnh C?
3.3 Có bao nhiêu thanh ghi khác nhau được dùng cho từng câu lệnh C bên trên.

Trả lời:
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)
```


⇨ *Có thể viết gọn hơn với ít lệnh MIPS hơn không? Có thể sử dụng ít thanh ghi hơn không?*

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?
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?
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?

Trả lời:

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

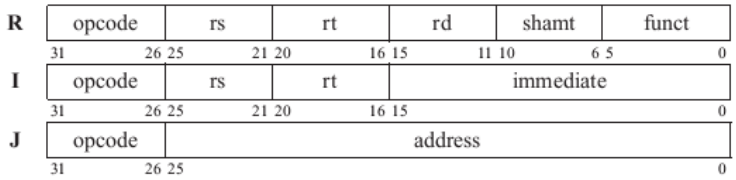
add $t0, $s0, $s1
add $t0, $t0, $s0
a) $t0 = 0x70000000 + (0x70000000 + 0x0FFFFFFF) = 0xEFFFFFFF
Tràn, bởi vì cộng hai số dương nhưng bit dấu của kết quả lại là âm, không đúng.
b) $t0 = 0x40000000 + (0x40000000 + 0x40000000) = 0xC0000000
Tràn, bởi vì cộng hai số dương nhưng bit dấu của kết quả lại là âm, không đúng.

Bài 5.

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.

5.3
Nếu chuỗi nhị phân trên chỉ là dữ liệu đơn thuần. Hãy chuyển chúng sang dạng mã HEX.

5.4 & 5.5
Hãy dịch các lệnh sau sang dạng mã máy
a) add $t0, $t0, $zero
b) lw $t1, 4($s3)

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:

5.1 & 5.2

a) op = $101011_{(2)}$ = $2B_{(hex)}$ ➔ Lệnh *sw* (tra bảng "MIPS reference data", trang 2, sách tham khảo)
➔ 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 = $100011_{(2)}$ = $23_{(10)}$ ➔ 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, $zero
add: R-type format
opcode = 0_{hex}
rs = 8_{ten} ($t0 là thanh ghi thứ 8/ $t0 là thanh ghi có chỉ số 8)

rt = 0_{ten} ($zero là thanh ghi thứ 0)
rd = 8_{ten} ($t0 là thanh ghi thứ 8)
shamt = 0_{ten} (không sử dụng)
func = 20_{hex} 1= $00000_{(2)}$
⇨ Machine code: 0000 0001 0000 0000 0100 0000 0010 0000_{bin} = **01004020_{hex}**

a) ***lw***: I-type format
opcode = 23_{hex}
rs = 19_{ten} = 10011_{bin} ($s3 là thanh ghi thứ 19)
rt = 9_{ten} ($t1 là thanh ghi thứ 9)
immediate = 4_{ten}
⇨ Machine code: 1000 1110 0110 1001 0000 0000 0000 0100_{bin} = **$8E690004_{hex}$**

5.6
a) op=0x0; rs=0x8; rt=0x0; rd=0x8; shamt=0x0; func=0x20
b) op=0x23; rs=0x13; rt=0x9; imm=0x4

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
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

Trả lời:
6.1 Tìm giá trị của $t2
a) 0x57755778
b) 0xFEFFFEDE

6.2 Tìm giá trị của $t2
a) 0x0000AAAA
b) 0x0000BFCD

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:
```

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

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:

7.1

a.

"slt" làm việc vơi 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 $2^{15}-1 = 32767$

⇨ Vậy X từ -1,391,460,349 tới 32767

b) 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?

  a) Không

b) Không

Lệnh Jump (j) thuộc kiểu lệnh J nên trường địa chỉ (address) có 26 bit.

Địa chỉ mà lệnh j sẽ nhảy tới, tức là địa chỉ sẽ gán cho con trỏ PC được tính bằng cách:

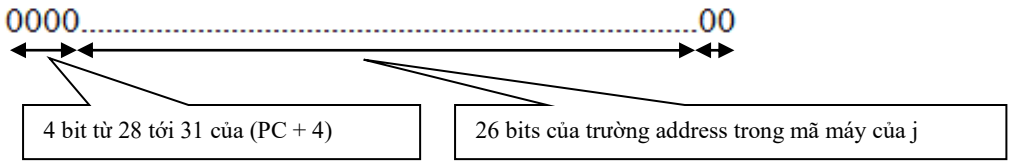
JumpAddr = { PC+4[31:28], address, 2'b0 } (Xem bảng trang 2 sách tham khảo)

Tức vùng address trong mã máy của lệnh j được lấy ra, dịch trái 2 bit; sau đó gán thêm 4 bit cao nhất được lấy từ 4 bit cao nhất (từ 28 tới 31) của PC hiện tại + 4.

PC hiện tại bằng 0x0000 0020 ➔ PC + 4 = 0x00000024

4 bit từ 28 tới 31 của PC + 4 = $0000_{(2)}$

⇨ PC mới, nơi mà lệnh j nhảy tới có giá trị:



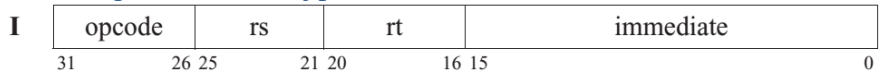
Vậy 4 bit cao nhất của PC mới với bất kỳ lệnh j nào cũng phải có 0000. Những giá trị cho trong bảng trên có giá trị lớn hơn 0000. Vì thế không thể sử dụng lệnh j để nhảy tới các địa chỉ như trong bảng

- 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

Lệnh beq thuộc kiểu I-type



Ví dụ lệnh beq $t0, $t1, 4

Thì 4 được lưu vào trường immediate

Khi beq thực hiện lệnh nhảy, giá trị mới gán cho PC = PC hiện tại + 4 + (immediate x 4)

Hay: PC mới = PC hiện tại + 4 + (immediate <<2) ***(<<2: dịch trái 2 bit)***

Immediate là số 16 bit, sau khi dịch trái 2 bit thành số 18 bit, giá trị lớn nhất của số 18 bit này là $2^{18} - 1$.

Vì vậy, PC lớn nhất chỉ có thể nhận giá trị = PC hiện tại + 4 + 2^{18}-1

= 0x0000 0020 + 4 + 2^{18} - 1

Giá trị lớn nhất này nhỏ hơn các giá trị được cho trong bảng trên.

Nên ta không thể gán giá trị PC tới các giá trị trong bảng bằng cách sử dụng lệnh beq