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.

Trả lời:
a. add f, g, h
add f, f, i
add f, f, j

b. addi f, h, 5
add f, f, g

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

Trả lời:
Tại câu a có 3 lệnh MIPS.
Tại câu b có 2 lệnh MIPS.

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?

Trả lời:

a. f = 2 + 3 + 4 + 5 = 14

b. f = 2 + (3 + 5) = 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.

Trả lời:

a. f = g + h

b. addi f, f, 1 => f = f + 1
add f, g, h => f = g + h
=> 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?

Trả lời:

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 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.

Trả lời:

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

Trả lời:

Tại câu a có 3 lệnh MIPS.
Tại câu b có 5 lệnh MIPS.

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

Trả lời:

Tại câu a có 4 thanh ghi khác nhau là $s0, $s1, $s2, $s7.
Tại câu b có 5 thanh ghi khác nhau là $s0, $t0, $s1, $s6, $s7.

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.

a. add $s0, $s0, $s1 => f = f + g
add $s0, $s0, $s2 => f = f + g + h
add $s0, $s0, $s3 => f = f + g + h + i
add $s0, $s0, $s4 => f = f + g + h + i + j
=> f = f + g + h + i + j;
b. lw $s0, 4($s6) => f = A[1];

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

a. Không được.
b. Không được.

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:
Câu a có 5 thanh ghi là $s0, $s1, $s2, $s3, $s4 (không thể rút gọn nhỏ hơn).
Câu b có 2 thanh ghi là $s0 và $s6 (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 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.

Trả lời:
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)
```

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

Trả lời:
Tại câu a cần có 3 lệnh hợp ngữ MIPS.
Tại câu b cần có 7 lệnh hợp ngữ MIPS.

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

Trả lời:
Câu a có 4 thanh ghi là $s0, $s1, $s2, $s7.
Câu b có 6 thanh ghi là $t0, $s0, $s1, $s2, $s6, $s7.

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?

Trả lời:
a. **$s0 = 0x70000000; $s1 = 0x0FFFFFFF**
add $t0, $s0, $s1 => $t0 = 0x7FFFFFFF
Kết quả đúng như mong muốn của phép toán, không tràn.

b. **$s0 = 0x40000000; $s1 = 0x40000000**
add $t0, $s0, $s1 => $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 (0x80000000 đã vượt quá giới hạn số nguyên dương của máy tính => số âm).

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?

Trả lời:
a. $t0 = 0x70000000 – 0x0FFFFFFF
= 1879048192 – 268435455 = 1610612737 = 0x60000001
=> Đúng với mong muốn, không tràn.

b. $t0 = 0x40000000 – 0x40000000 = 0 => Đúng với mong muốn, không 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?

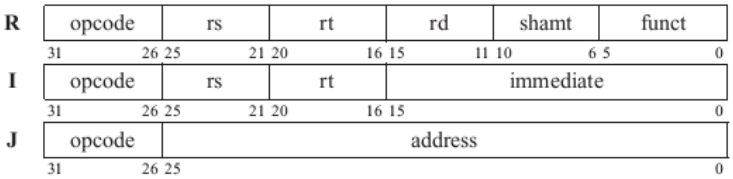
Trả lời:
a. $t0 = 0x70000000 + 0x70000000 + 0x0FFFFFFF = 0xEFFFFFFF => Tràn do cộng hai số dương nhưng kết quả là số âm (0xEFFFFFFF vượt quá giới hạn số nguyên dương).

b. $t0 = 0x40000000 + 0x40000000 + 0x40000000 = 0xC0000000 => Tràn do cộng hai số dương nhưng kết quả là số âm (0xC0000000 vượt quá giới hạn số nguyên dương).

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.

Trả lời:
a) op = 101011 (bin) = 2B (hex) => Lệnh sw.
=> Dạng I – type.
rs = 10000 (bin) = 16 (dec): vậy địa chỉ nền được lưu trong thanh ghi thứ 16, là thanh ghi $s0.
rt = 01011 (bin) = 11 (dec): thanh ghi thứ 11, là $t3.
Immediate = 0000000000000100 (bin) = 4 (dec).
=> Lệnh assembly này là: **sw $t3, 4($s0)**

b) op = 100011 (bin) = 23 (hex) => Lệnh lw
=> **Dạng I – type.**
rs = 01000 (bin) = 8 (dec) => Thanh ghi $t0.
rt = 01000 (bin) = 8 (dec) => Thanh ghi $t0.
Immediate (16 bit) = 0000000001000000 (bin) = 64 (dec).
=> Lệnh assembly: **lw $t0, 64($t0)**

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

Trả lời:
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)

Trả lời:
a) **add $t0, $t0, $zero**
Lệnh add => Dạng R – type.
opcode = 0 (hex) = 000000 (bin).
rs = 8 (dec) = 01000 (bin) ($t0 là thanh ghi thứ 8).
rt = 0 (dec) = 00000(bin) ($zero là thanh ghi thứ 0).
rd = 8 (dec) = 01000 (bin) ($t0 là thanh ghi thứ 8).
shamt = 0 (dec) = 00000 (bin).
func = 20 (hex) = 100000 (bin).
=> **Mã máy:** 0000 0001 0000 0000 0100 0000 0010 0000 (bin) = **01004020 (hex).**

b) **lw $t1, 4($s3)**
Lệnh lw => Dạng I – type.
opcode = 23 (hex) = 100011 (bin).
rs = 19 (dec) = 10011 (bin) ($s3 là thanh ghi thứ 19).
rt = 9 (dec) = 01001 (bin) ($t1 là thanh ghi thứ 9).
Immediate = 4 (dec) = 0000 0000 0000 0100 (bin).
=> **Mã máy:** 1000 1110 0110 1001 0000 0000 0000 0100 (bin). = **8E690004 (hex).**

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).

Trả lời:
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

Trả lời:

a) **sll $t2, $t0, 4** => $t2 = 0x55555550
or $t2, $t2, $t1 => $t2 = 0x57755778

b) **sll $t2, $t0, 4** => $t2 = 0xEADFEED0
or $t2, $t2, $t1 => $t2 = 0xFEFFFEDE

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:

a) **srl $t2, $t0, 3** => $t2 = 0x0AAAAAAA
andi $t2, $t2, 0xFFEF => $t2 = 0x0000AAAA

b) **srl $t2, $t0, 3** => $t2 = 0x17D5BFDD
andi $t2, $t2, $t1 => $t2 = 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:
```

Trả lời:

a. $t0 = 1010 1101 0001 0000 0000 0000 0000 0010 (bin) => Là một số âm
Mà $t1 là một số dương (0011 1111 1111 1000 0000 0000 0000 0000)
⇨ $t1 > $t0 => $t2 = 1
⇨ lệnh "j DONE" được thực hiện => Giá trị của $t2 sau khi chạy các lệnh dưới là 1.

b. $t0 = 1111 1111 1111 1111 1111 1111 1111 1111 (bin) => Là một số âm
Mà $t1 là một số dương (0011 1111 1111 1000 0000 0000 0000 0000)
⇨ $t1 > $t0 => $t2 = 1
⇨ lệnh "j DONE" được thực hiện => Giá trị của $t2 sau khi chạy các lệnh dưới là 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

Trả lời:

a. $t0 = 1010 1101 0001 0000 0000 0000 0000 0010 (bin) = -1391460350 (dec)
Để $t2 = 1 => $t0 < X => X phải từ -1391460349 tới 32767.
b. $t0 = 1111 1111 1111 1111 1111 1111 1111 1111 (bin) = -1 (dec)
Để $t2 =1 => $t0 < X => X phải từ 0 tới 32767.

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.

Trả lời:

a. Không
b. Không
Giải thích:

- Địa chỉ nhảy tới của lệnh j được tính bằng cách:
  (5) JumpAddr = { PC+4[31:28], address, 2'b0}
- Tức vùng address trong lệnh j được lấy ra, dịch trái 2 bit, sau đó gán 4 bit cao nhất (từ 28-31) của PC hiện tại cho vùng address mới
- PC hiện tại = 0x00000020 => PC nhảy tới = 0x00000020 + 4 = 0x00000024
- 4 bit cao nhất của PC nhảy tới = 0000
- ⇨ Vậy PC nhảy tới phải có 4 bit cao nhất là 0000, nhưng thanh ghi $t0 của cả a và b đều lớn hơn 0000 nên không thể sử dụng lệnh nhảy

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

a) Không
b) Không
- Địa chỉ nhảy tới của lệnh beq:

- Khi thực hiện lệnh beq, PC = PC hiện tại + 4 + immediate x 4 (dịch 2 bit)
- Immediate có 16 bit, sau khi dịch trái 2 bit thì có 18 bit, giá trị lớn nhất của số 18 bit này là 2^18 – 1
- PC mới = 0x00000020 + 4 + 2^18 -1

⇨ Giá trị lớn nhất của PC mới nhỏ hơn PC của $t0 cả a và b nên không thể dùng beq để nhảy tới địa chỉ của a và b

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

a. and $t3, $s0, $s2
b. sll $t1, $t5, 7
c. addi $t0, $s3, 25
d. addi $t0, $s3, -25
e. lw $t0, 24($s0)
f. lw $t0, -24($s0)
g. sw $t2, 48($s0)
h. sw $t2, -48($s0)

Trả lời:

a) **and $t3, $s0, $s2**
Lệnh and => Dạng R – type.
opcode = 0 (bin) = 000000 (hex).
rs = 16 (dec) = 10000 (bin) ($s0 là thanh ghi thứ 16).
rt = 18 (dec) = 10010 (bin) ($s2 là thanh ghi thứ 18).
rd = 11 (dec) = 01011 (bin) ($t3 là thanh ghi thứ 11).
shamt = 0000 (bin).
funct =23 (hex) = 100100 (bin).
=> **Mã máy:** 0000 0010 0001 0010 0101 1000 0010 0100 = **0x02125824**

b) **sll $t1, $t5, 7**
=> **Mã máy:** 0000 0000 0000 1101 0100 1001 1100 0000 = **0x000D49C0**

c) **addi $t0, $s3, 25**
addi => Dạng I – type.
opcode = 8 (hex) = 001000 (bin).
rs = 19 (dec) = 10011 (bin).
rt = 8 (dec) = 01000 (bin).
Immediate = 25 = 0000 0000 0001 1001
=> **Mã máy:** 0010 0010 0110 1000 0000 0000 0001 1001 = **0x22680019**

d) **addi $t0, $s3, -25**
Immediate = -25 (dec) = 1111 1111 1110 0111
=> **Mã máy:** 0010 0010 0110 1000 1111 1111 1110 0111 = **0x2268FFE7**

e) **lw $t0, 24($s0)**
lw => Dạng I – type.
opcode = 23 (hex) = 100011
rs = 16 (dec) = 10000
rt = 8 (dec) = 01000
Immediate = 24 = 0000 0000 0001 1000
=> **Mã máy:** 1000 1110 0000 1000 0000 0000 0001 1000 = **0x8E080018**

f) **lw $t0, -24($s0)**
Immediate = -24 (dec) = 1111 1111 1110 1000
=> **Mã máy**: 1000 1110 0000 1000 1111 1111 1110 1000 = **0x8E08FFE8**

g) **sw $t2, 48($s0)**
sw => Dạng I – type.
opcode = 2B (hex) = 101011 (bin).
rs = $s0 = 10000 (bin).
rt = $t2 = 01010 (bin).
Immediate = 48 (dec) = 0000 0000 0011 0000
=> **Mã máy:** 1010 1110 0000 1010 0000 0000 0011 0000 = **0xAE0A0D30**

h) **sw $t2, -48($s0)**
Immediate = -48 = 1111 1111 1101 0000
=> **Mã máy:** 1010 1110 0000 1010 1111 1111 1101 0000 = **0xAE0AFFD0**

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

a. *0x01304024*
b. *0x2128fff3 (0x2128FFF3)* ***(Trong hệ 16, các chữ từ a tới f có thể viết thường hoặc hoa đều được)***
c. *0xad28fffc*

Trả lời:

a) **0x01304024**
=> Binary = 0000 0001 0011 0000 0100 0000 0010 0100
opcode = 0 => Dạng R – type.
funct = 0010 0100 = 24 (hex) => Lệnh and.
rs = 9 => Thanh ghi $t1.
rt = 16 => Thanh ghi $s0.
rd = 8 => Thanh ghi $t0.
=> Lệnh MIPS: **and $t0, $t1, $s0**

b) **0x2128FFF3**
=> Binary = 0010 0001 0010 1000 1111 1111 1111 0011
opcode = 001000 = 8 (hex) => Lệnh addi, dạng I – type.
rs = 9 => Thanh ghi $t1.
rt = 8 => Thanh ghi $t0.
Immediate = 1111 1111 1111 0011 = -13 (dec).
=> Lệnh MIPS: **addi $t0, $t1, -13**

c) **0xAD28FFFC**
=> Binary = 1010 1101 0010 1000 1111 1111 1111 1100
opcode = 101011 = 2B (hex) => Lệnh sw, dạng I – type.
rs = 9 => Thanh ghi $t1.

rt = 8 => Thanh ghi $t0.
Immediate = 1111 1111 1111 1100 = -4 (dec)
=> Lệnh MIPS: **sw $t0, -4($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
$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
$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
$t2 = 0x000000090

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?
$t0 = 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++;
```

- *MIPS:*

```
sll $t0, $s0,2          #$t0 = i*4
add $t0,$t0,$s3     #$t0 = A  + i*4
lw $t0,0($t0)         #$t0 = A[i]

addi $t1, $s0,1          #$t1 = i +1
sll $t1,$t1,2            #$t1 =$t1*4
add $t1,$t0,$s3        #$t1 = A + (i +1)*4
lw $t1,0($t1)          #A[i+1]

blt $s0,$s1, true
addi $t2,$t0,-1       #$t2 = A[i] - 1
sw $t2,0($t0)        #A[i] = A[i] – 1

addi $t3,$t3,10     #$t3 = 10
sw $t3,0($t1)       #A[i+1] = 10
j exit
true:

addi $t3,$t3,5         #$t3= 5
sw $t3,0($t1)          #A[i +1] =5
exit:
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++;
```

- *MIPS:*

```
        sll $t0, $s0,2          #$t0 = i*4
        add $t0,$t0,$s3     #$t0 = A  + i*4
        lw $t0,0($t0)         #$t0 = A[i]

        addi $t1, $s0,1          #$t1 = i +1
        sll $t1,$t1,2           #$t1 = $t1*4
        add $t1,$t0,$s3         #$t1 = A + (i +1)*4
        lw $t1,0($t1)          #$t1 = A[i+1]

        sll $t2, $s1,2          #$t2 = j*4
        add $t2,$t2,$s3     #$t2 = A  + j*4
        lw $t2,0($t2)         #$t2 = A[j]

        ble $s0,$s1,continue
        j false
         continue: bgt $s1,0, true
        j false
          true:
           add $t3,$t0,$t1     # $t3= A[i] + A[i+1]
           sw $t3, 0($t2)       # A[j] = A[i] + A[i+1]
          j exit
        false:
           sub $t3,$t0,$t1     #$t0 = A[i] - A[i+1]
          sw $t3, 0($t2)      #A[j] = A[i] - A[i+1]

        exit:
addi $s0,$s0,1
```

c.

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

- *MIPS:*

```
        bgt $s0,0,loop
        j exit
loop:
        sll $t0, $s0,2          #$t0 = i*4
        add $t0,$t0,$s3      #$t0 = A + i*4
        lw $s3,0($t0)         #A[i]
        sll $t0,$s3,3       #$t0 = A[i]*8

        addi $t1,$s0,1       #$t1 = i + 1
        sll $t1,$t1,2          #$t1 = $t1*4
```

```
        add $t1,$t1,$s3      #$t1 = A + $t1*4
        lw $t1,0($t1)        #$t1 = A[i+1]

        sw $t0,0($t1)     #A[i+1] = A[i]*8
        addi $s0,$s0,-1   #i --
        bgt $s0,0,loop
        j exit
exit: lw $t0,0($s3)         #A[0]
      addi $t1,$t1,5         #$t1= 5
      sw $t1, 0($t0)         #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)

- MIPS:

```
        sw $s5, 0($s1)
        blt $s0,1,exit
loop:
        sll $t0, $s0,2          #$t0 = i*4
        add $t1,$t0,$s3     #$t1 = A  + i*4
        lw $t1,0($t1)         #$t1 = A[i]

        add $t0,$t0,$s4     #$t0 = B  + i*4
        lw $t0,0($t0)         #$t0 = B[i]

        sw $t0, 0($t1)        #A[i] = B[i]

        addi $s0,,$s0,1
        blt $s0,$s1,loop     #Kiem tra i<j ?
        j exit

exit:
       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)

- *MIPS:*

```
        sw $s5, 0($s1)
        li $s4,0
        blt $s0,0,exit
loop:
        sll $t0, $s0,2          #$t0 = i*4
        add $t0,$t0,$s3     #$t0 = A  + i*4
        lw $t0,0($t1)         #$t0 = A[i]

        bgt $t0,$s4,true
        j loop
        true:
        sw $s4, 0($t0)         #max= A[i]

        addi $s0,,$s0,1
        bge $s0,$s1,exit     #Kiem tra i<j ?
        j loop
exit:
       li $s1,0
```