

ĐẠI HỌC QUỐC GIA TP.HCM
ĐẠI HỌC KHOA HỌC TỰ NHIÊN



KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ

BÁO CÁO ĐỒ ÁN
BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC
TRÊN MÁY TÍNH

Giảng viên hướng dẫn :

Thầy Lê Viết Long

Sinh Viên :
Nguyễn Duy Thảo
Lê Văn Thi
Nguyễn Ngọc Triệu

MSSV:
1612639
1612647
1612739

Mục lục

Mục lục	2
1. Dữ liệu kiểu số nguyên lớn.....	4
2. Thao tác bit	4
• <i>Thao tác trên 1 byte (8 bit):</i>	4
a. Bật bit:	5
b. Tắt bit:	5
c. Đọc bit:	5
• <i>Thao tác với kiểu dữ liệu QInt - 16 bytes (128 bit):</i>	5
3. Nhập xuất dữ liệu	6
• <i>Các phép xử lý số nguyên dạng chuỗi:</i>	6
a. Loại bỏ các số 0 dư ở đầu chuỗi:	6
b. Nhân, chia số nguyên dạng chuỗi cho 2:	6
c. Cộng 2 số thập phân dạng chuỗi:	7
• <i>Số nhị phân dạng bù 1, bù 2:</i>	7
a. Số bù 1:	7
b. Số bù 2:	7
• <i>Chuyển đổi số nguyên dạng chuỗi sang QInt:</i>	7
a. Hàm nhập:	8
b. Hàm xuất:	8
4. Chuyển đổi giữa các hệ cơ số:	9
a. Thập phân sang nhị phân :	9
b. Nhị phân sang thập phân :	9
c. Nhị phân sang thập lục phân :	9
d. Thập phân sang thập lục phân:	9
5. Các operator toán tử +, -, *, /	10
a. Phép cộng (+):	10
b. Phép trừ (-):	10
c. Phép nhân:	10
d. Phép chia:	11

6. Các toán tử trên bit:	11
a. Phép AND (&):	11
b. Phép OR ():	11
c. Phép XOR (^):	11
d. Phép NOT (~):	12
e. Phép dịch trái (<<):	12
f. Phép dịch phải (>>):	12
7. Test case:	14
a. Nhập xuất:	14
b. Chuyển đổi giữa các hệ đếm:	14
c. Các phép toán +, -, *, /:	15
c. Toàn tử dịch trái (<<), dịch phải (>>):	18
8. Đánh giá:	19

1. Dữ liệu kiểu số nguyên lớn

Số nguyên trong máy tính được biểu diễn dưới dạng nhị phân. Ví dụ:

$$10101_2 = 21_{10}$$
$$101001101_2 = 333_{10}$$

Kiểu dữ liệu số nguyên int trong máy tính có độ lớn là 4 bytes (32 bits) => có thể biểu diễn được giá trị từ $-2^{32} \rightarrow 2^{32} - 1$.

Vậy, nếu muốn lưu trữ số nguyên có giá trị lớn hơn (nằm ngoài miền giá trị của int thì sao?

- ⇒ Ta có thể thiết kế một kiểu dữ liệu số nguyên có độ lớn trên 4 byte. Trong bài này ta sẽ thiết kế kiểu dữ liệu QInt có độ lớn 16 bytes (128 bits). Miền giá trị của kiểu dữ liệu này trong khoảng từ -2^{127} đến $2^{127} - 1$. lớn hơn.

Cấu trúc kiểu dữ liệu: Ta có thể sử dụng mảng int 4 phần tử để biểu diễn 16 byte của QInt.

```
// QInt 16 byte = 128 bit
// Phạm vi biểu diễn :          -2^127 : (2^127) - 1
// -170141183460469231731687303715884105726 : 170141183460469231731687303715884105725
struct QInt
{
    // kiểu int 4 bytes, mảng 4 phần tử => 16 bytes
    int data[4];
};
```

Việc nhập xuất, khởi tạo, tính toán giá trị của kiểu dữ liệu này đều thông qua thao tác trên các bit nhị phân.

2. Thao tác bit

Số nguyên trong máy tính được lưu trữ dưới dạng dãy các bit nhị phân. Bằng cách khởi tạo, thay đổi giá trị các dãy bit trong vùng nhớ của biến dữ liệu, ta có thể khởi tạo, thay đổi giá trị số nguyên của biến.

Có 3 thao tác chính là đọc bit, bật bit (gán bit 1 tại vị trí tương ứng trong dãy bit) và tắt bit (gán bit 0 tại vị trí tương ứng trong dãy bit).

Thao tác trên 1 byte (8 bit):

Để thực hiện bật, tắt bit, ta sử dụng dãy bit sau (dãy bit có giá trị thập phân là 1):

	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1

a. Bật bit:

Ta dịch chuyển bit 1 của dãy bit trên đến vị trí tương ứng và thực hiện phép toán **OR** (phép cộng logic) với dãy bit cần bật.

Ví dụ, bật bit thứ 4 của dãy bit 0:

	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1
<< 4	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0

b. Tắt bit:

Ta dịch chuyển 1 đến vị trí tương ứng và thực hiện đảo bit (phép toán **NOT**) dãy bit trên để được bit 0 ở vị trí cần tắt và bit 1 ở các vị trí còn lại. Sau đó thực hiện phép toán **AND** (phép nhân logic) với dãy bit cần bật.

Ví dụ, tắt bit thứ 4 của dãy bit 1:

	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1
<< 4	0	0	0	1	0	0	0	0
~	1	1	1	0	1	1	1	1
&	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1

c. Đọc bit:

Ta dịch phải bit tới cuối dãy rồi thực hiện phép nhân logic với 1. Giá trị cuối cùng của dãy bit sẽ là 1 nếu đó là bit 1, là 0 nếu đó là bit 0.

Ví dụ: đọc bit 4 của dãy

	7	6	5	4	3	2	1	0
	1	1	1	0	0	1	1	0
>>4	0	0	0	0	1	1	1	0
&	0	0	0	0	0	0	0	1
0 ₁₀	0	0	0	0	0	0	0	0

Thao tác với kiểu dữ liệu QInt - 16 bytes (128 bit):

Do kiểu dữ liệu được xây dựng bằng mảng 4 phần tử nên không thể thao tác trực tiếp trên 128 bit mà phải xác định vị trí của bit đó ở phần tử thứ bao nhiêu rồi thao tác riêng trên phần tử đó.

Mỗi phần tử trong mảng có độ lớn 4 byte (32 bit), để xác định vị trí muốn thao tác nằm ở phần tử nào ta lấy vị trí đó chia 32. Rồi lấy hiệu của 4 và kết quả.

0	1	2	3
32bit	32bit	32bit	32bit

Ví dụ: ta muốn thao tác ở vị trí bit thứ 78. Ta có $78 \div 32 = 2 \rightarrow$ Bit cần thao tác nằm ở phần tử $4 - 2 = 2$. (Vị trí bit được tính từ phải sang trái – Mang giá trị từ 0 \rightarrow 127)

Tiếp theo để thao tác bit trong từng phần tử, ta cần xác định vị trí của bit đó trong phần tử. Vị trí được tính bằng cách lấy phần dư của phép chia phía trên (vị trí đầu vào chia 32). Theo ví dụ trên, ta có vị trí của bit cần thao tác trong phần tử thứ 2 là $78 \% 32 = 14$.

\Rightarrow Bit cần thao tác là bit thứ 14 của phần tử $a[2]$ trong mảng.

3. Nhập xuất dữ liệu

Các phép xử lý số nguyên dạng chuỗi:

Dữ liệu sẽ được nhập vào từ bàn phím ở dạng chuỗi (có dấu hoặc không dấu) nên ta cần các hàm xử lý chuỗi này phục vụ cho việc nhập xuất, tính toán.

a. Loại bỏ các số 0 dư ở đầu chuỗi:

Chuỗi do người dùng nhập vào có thể có những số 0 dư ở đầu chuỗi. Để tiện cho việc tính toán, ta cần xóa bớt các số 0 này đi:

- Ta xét vị trí đầu chuỗi, nếu gặp dấu thì ta bỏ qua dấu và bắt đầu xét từ vị trí kế tiếp.
- Duyệt xem và xóa đi chuỗi số 0 dư ở đầu chuỗi, lưu ý nếu gặp trường hợp chuỗi toàn số 0, cần tránh xóa luôn số 0 cuối cùng.

b. Nhân, chia số nguyên dạng chuỗi cho 2:

Trong mã ASCII, số = kí tự - 48. Bằng công thức trên, ta có thể lấy giá trị thập phân của từng chữ số trong chuỗi đầu vào để xử lý các phép tính trên chuỗi.

Việc xử lý phép nhân chia theo như nhân chia thập phân thông thường.

- Nhân 2: Nhân từng chữ số và lưu kết quả từ phải sang trái. Nếu kết quả > 9 thì lưu chữ số hàng đơn vị và cộng thêm biến nhớ phép nhân chữ số kế tiếp.
- Chia 2: Dùng 1 biến lưu số chia ở các bước. Chia từng chữ số và lưu kết quả từ trái sang phải.
 - Nếu chữ số đó chia hết cho 2, thì số chia bước tiếp theo chính là *chữ số kế tiếp*.
 - Nếu chữ số đó không chia hết cho 2 (có dư), số chia kế tiếp bằng số $dư * 10 + \text{chữ số tiếp theo}$.
- Lũy thừa 2: Chạy vòng lặp gọi hàm nhân 2 n lần. (n là số mũ đầu vào)

c. Cộng 2 số thập phân dạng chuỗi:

Trước tiên, ta cần điền thêm các bit 0 vào chuỗi có độ dài ngắn hơn (nếu 2 chuỗi có độ dài khác nhau).

Xét 2 trường hợp:

- 2 số dương cộng nhau: Ta xử lí như phép cộng thông thường. Cộng từng chữ số từ phải sang trái, lưu biến nhớ nếu kết quả >9 .
- 1 trong 2 số là số âm: Ta xử lí như phép trừ, trừ từng chữ số từ phải sang trái. Nếu số trừ $<$ số bị trừ, 'mượn' (+10) vào số trừ để tính và lưu biến mượn.

Số nhị phân dạng bù 1, bù 2:

a. Số bù 1:

Số bù 1 có được bằng cách đảo bit số dấu lượng. Số bù 1 có thể dùng để biểu diễn số âm.

Ví dụ :

00101011 \Rightarrow 11010100 (Bù 1)

0111001100 \Rightarrow 1000110011 (Bù 1)

C++ có định nghĩa sẵn toán tử NOT cho kiểu int. Qint cấu trúc bởi mảng 4 phần tử int nên ta chỉ việc chạy vòng lặp đảo bit từng phần tử một.

b. Số bù 2:

Số bù 2 bằng số bù 1 cộng thêm 1, ta thực hiện cộng nhị phân y như phép cộng thập phân. Số bù 2 có thể được sử dụng để biểu diễn số âm.

Ví dụ:

00101011 \Rightarrow 11010100 (Bù 1) \Rightarrow 11010101 (Bù 2)

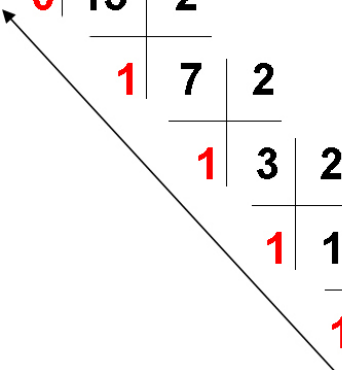
0111001100 \Rightarrow 1000110011 (Bù 1) \Rightarrow 1000110100 (Bù 2)

Để chuyển Qint sang bù 2 ta chỉ cần gọi hàm chuyển số đầu vào sang dạng bù 1 rồi cộng thêm một.

Chuyển đổi số nguyên dạng chuỗi sang Qint:

Chuỗi được chuyển từ thập phân sang nhị phân bằng cách chia 2 lấy phần dư. Ví dụ:

Dòng						
1		30		2		
2		0		15		2
3				1		7
4						1
5						1
6						1



Trước hết ta khởi tạo 1 biến có giá trị 0 để lưu kết quả.

Ta bắt đầu tính các bit của chuỗi nhị phân, tức phần dư của phép chia 2: (phần dư này do số cuối cùng quyết định). Chạy vòng lặp duyệt chuỗi từ phải sang trái:

- Đổi chữ số cuối cùng sang số (-48)
- Chia giá trị này cho 2 lấy phần dư
- Ghi bit này vào biến Qint
- Chia 2 chuỗi

Vòng lặp kết thúc khi giá trị của chuỗi trở thành 0.

Nếu dữ liệu nhập là số âm, ta chuyển số đó sang dạng bù 2.

a. Hàm nhập:

Số nguyên được nhập dưới dạng chuỗi từ bàn phím, sau đó chuyển đổi sang kiểu dữ liệu QInt.

b. Hàm xuất:

Đọc từng bit (bỏ qua bit đầu tiên) của biến đầu vào và chuyển chuỗi nhị phân thành thập phân (tính toán và lưu giá trị thập phân này bằng chuỗi) theo công thức sau:

$$x_{n-1}...x_1x_0 = x_{n-1}.2^{n-1} + ... + x_1.2^1 + x_0.2^0$$

Ví dụ: $100100_2 = 2^5 + 2^2 = 36_{10}$

Phép cộng trên được xử lý ở dạng chuỗi.

Tiếp theo ta xét bit đầu tiên (bit dấu), nếu đây là số âm (bit dấu bằng 1), ta thêm dấu trừ vào trước chuỗi.

Cuối cùng, ta xóa đi những số 0 thừa ở đầu chuỗi và xuất chuỗi ra màn hình.

Áp dụng cho QInt, vì số mũ ở đây rất lớn (max = 127), do đó ta viết 1 hàm tính mũ và trả về chuỗi (hàm có sẵn không thể tính được số mũ lớn như 127, cũng không có kiểu dữ liệu nào ngoài chuỗi có thể lưu trữ một số cực lớn như vậy).

Sau khi tính toán được các giá trị 2^x , ta tiến hành cộng các chuỗi lại với nhau và cuối cùng xuất QInt ra ở dạng chuỗi.

4. Chuyển đổi giữa các hệ cơ số:

a. Thập phân sang nhị phân :

Do dữ liệu trong máy tính được lưu dưới dạng nhị phân nên ta chỉ cần đọc từng bit của biến và ghi vào chuỗi kết quả.

b. Nhị phân sang thập phân :

Ta khởi tạo một biến có QInt có toàn giá trị 0 để lưu kết quả, ta xét từng vị trí của chuỗi nhị phân, nếu là 1 thì ta gán vào bit có vị trí tương ứng của QInt.

c. Nhị phân sang thập lục phân :

Quy tắc chuyển đổi từ nhị phân sang thập lục phân: Gom chuỗi nhị phân thành từng nhóm 4 bit , mỗi nhóm tương ứng với 1 giá trị thập phân và 1 ký số trong hệ thập lục phân.

Ta đổi các nhóm 4 bit thành giá trị thập phân rồi dựa vào đó xác định ký số thập lục.

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Ví dụ:

$$0011\ 0010\ 1101\ 1001_2 = 32D9_{16}$$

$$0110\ 1100\ 0101_2 = 6C5_{16}$$

Kích thước của số thập lục phân: Mỗi ký số trong thập lục phân tương ứng với 4 bit => QInt hệ thập lục phân có $128 \div 4 = 32$ ký số, tức 32 byte.

d. Thập phân sang thập lục phân:

Để đổi từ thập phân sang thập lục phân, ta chỉ cần gọi hàm đổi từ thập phân sang nhị phân rồi đổi từ nhị phân sang thập lục.

5. Các operator toán tử +, -, *, /

a. Phép cộng (+):

Với phép cộng, ta thực hiện phép cộng nhị phân dãy bit của hai số hạng. Cách cộng nhị phân như thập phân bình thường, ta cộng từng cặp bit từ phải sang trái.

Ví dụ:

$$\begin{array}{r} 1 \\ 11101 \\ + 10001 \\ \hline 10110 \end{array}$$

Với mỗi cặp bit $value1$, $value2$, ta cộng 2 bit với nhau và cộng thêm bit nhớ $membit$. (bit nhớ được khởi tạo bằng 0)

- Nếu $value1 + value2 + membit = 3$ thì ta ghi kết quả là 1 và lưu $membit = 1$
- Nếu $value1 + value2 + membit = 2$ thì ta ghi kết quả là 0 và lưu $membit = 1$
- Nếu tổng của chúng là 0 hoặc 1 thì ghi kết quả tương ứng và $membit = 0$

b. Phép trừ (-):

Ta cũng thực hiện phép trừ nhị phân hai dãy bit.

Quy tắc trừ nhị phân bằng cách lấy số trừ cộng cho số bù 2 của số bị trừ. Cần lưu ý do ta sử dụng số bù 2 để biểu diễn số âm nên nếu số bị trừ là số âm thì phép trừ sẽ biến thành phép cộng.

c. Phép nhân:

Áp dụng thuật toán nhân có dấu cho Qint:

- Khởi tạo $A = 0$, $Q_{-1} = 0$, gọi 2 toán hạng lần lượt là Q và M.
- Vòng lặp $k = 128 \rightarrow 0$:
 - Xét 2 bit Q_0Q_{-1} (Q_0 là bit 127 của Q):
 - Nếu 2 bit $Q_0Q_{-1} = 10$ thì $A = A - M$
 - Nếu 2 bit $Q_0Q_{-1} = 01$ thì $A = A + M$
 - Nếu 2 bit $Q_0Q_{-1} = 00$ hoặc 11 thì A không đổi
 - Dịch phải dãy $[A, Q, Q_{-1}]$
- Kết quả: $[A, Q]$

d. Phép chia:

Áp dụng thuật toán chia không dấu, nhưng vì QInt là số có dấu. Do đó ta chuyển số chia và số bị chia về số dương nếu chúng là số âm. Tùy từng trường hợp mà ta sẽ quyết định dấu của kết quả

6. Các toán tử trên bit:

Với các toán tử này, ta đọc và thao tác trên từng bit /cặp bit của toán hạng và lưu từng bit một vào biến kết quả.

a. Phép AND (&):

Còn gọi là phép nhân logic, phép AND chỉ cho ra giá trị 1 nếu cả hai bit toán hạng đều có giá trị 1.

Bảng chân trị:

AND	0	1
0	0	0
1	0	1

C++ có định nghĩa sẵn toán tử & cho các kiểu dữ liệu có sẵn. Ta có thể sử dụng 1 biến kiểu bool để lưu kết quả từng cặp bit sau đó ghi vào biến kết quả cuối cùng.

b. Phép OR (|):

Còn gọi là phép cộng logic, phép OR chỉ cho kết quả 0 nếu cả hai bit toán hạng đều có giá trị 0.

Bảng chân trị:

OR	0	1
0	0	1
1	1	1

C++ có định nghĩa sẵn toán tử | cho các kiểu dữ liệu có sẵn. Ta có thể sử dụng 1 biến kiểu bool để lưu kết quả từng cặp bit sau đó ghi vào biến kết quả cuối cùng.

c. Phép XOR (^):

Còn gọi là phép so sánh khác, phép XOR cho kết quả 1 nếu hai bit của toán hạng khác nhau, cho kết quả 0 nếu hai bit giống nhau.

Bảng chân trị:

XOR	0	1
0	0	1
1	1	0

C++ có định nghĩa sẵn toán tử ^ cho các kiểu dữ liệu có sẵn. Ta có thể sử dụng 1 biến kiểu bool để lưu kết quả từng cặp bit sau đó ghi vào biến kết quả cuối cùng.

d. Phép NOT (~):

Còn gọi là phép đảo bit, phép phủ định, phép NOT biến tất cả các bit 0 thành 1 và tất cả các bit 1 thành 0.

Bảng chân trị:

NOT	0	1
	1	0

⇒ Ta chỉ cần gọi lại hàm đảo bit đã viết.

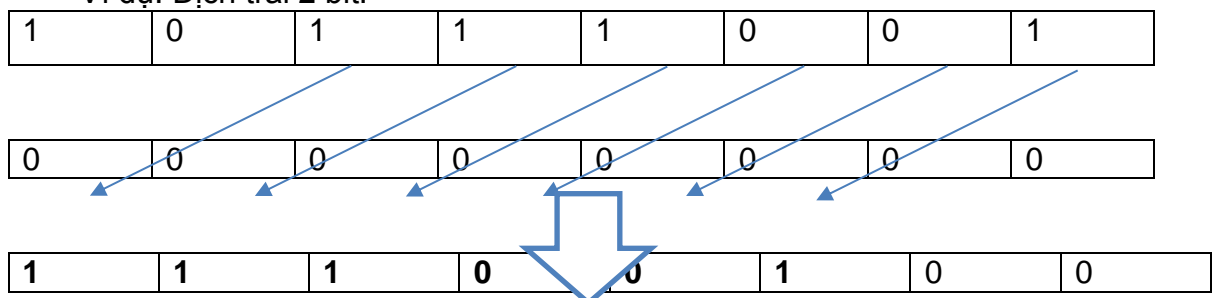
e. Phép dịch trái (<<):

Phép dịch trái chuyển tất cả các bit sang trái, bỏ bit trái nhất và thêm 0 ở bit phải nhất. Bit dấu không đổi.

Cách dịch:

- Khởi tạo một biến có giá trị đầu bằng 0 để lưu kết quả.
- Duyệt chuỗi từ trái sang phải, bắt đầu từ bit thứ n (n là số bit cần dịch), chép từng bit một vào biến kết quả.

Ví dụ: Dịch trái 2 bit.



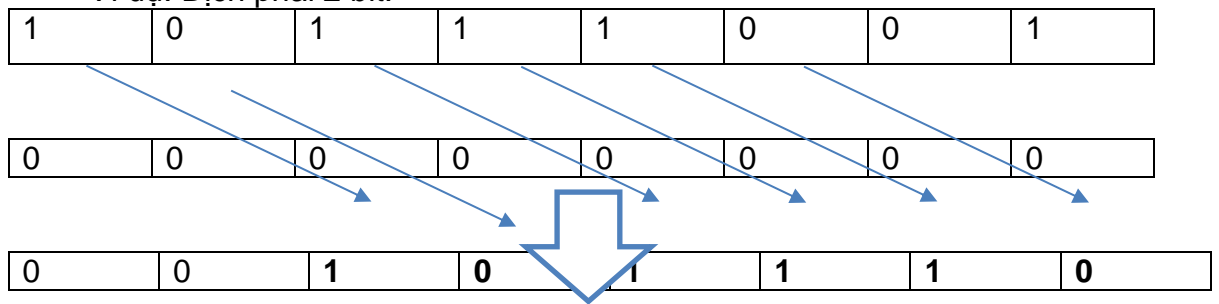
f. Phép dịch phải (>>):

Phép dịch phải chuyển tất cả các bit sang phải, bỏ bit phải nhất và thêm 0 ở bit trái nhất. Bit dấu không đổi.

Cách dịch:

- Khởi tạo một biến có giá trị đầu bằng 0 để lưu kết quả.
- Duyệt chuỗi từ phải sang trái, bắt đầu từ bit thứ n (n là số bit cần dịch), chép từng bit một vào biến kết quả (chép từ phải sang trái).

Ví dụ: Dịch phải 2 bit.



7. Test case:

a. Nhập xuất:

```
Input: 12345678987654321
Output: 12345678987654321
Press any key to continue . . .
```

```
Input: -478145821357823
Output: -478145821357823
Press any key to continue . . .
```

b. Chuyển đổi giữa các hệ đếm:

➤ Thập phân sang nhị phân:

```
Decimal:
123
Binary
1111011
Press any key to continue . . .
```

➤ Nhị phân sang thập phân:

```
Binary
1111011
Decimal:
123
Press any key to continue . . .
```

➤ Nhị phân sang thập lục phân:

```
Binary
1001111001101
Hexadecimal: 13CD
Press any key to continue . . .
```

- *Thập phân sang thập lục phân:*

```
Decimal:
5069
Hexadecimal: 13CD
Press any key to continue . . .
```

c. Các phép toán +, -, *, /:

- *Phép + : Ta xét từng trường hợp âm dương của a và b*

```
a = 315802163598203
b = 93237931372
a + b = 315895401529575
Press any key to continue . . .

-----
a = 332999750312721982
b = -872221208242693291325130
a + b = -872220875242942978603148
Press any key to continue . . .

a = -2300553388277
b = 23692738
a + b = -2300529695539
Press any key to continue . . .

a = -63573938001446
b = -33409578863
a + b = -63607347580309
Press any key to continue . . .
```

➤ *Phép -* : Ta xét từng trường hợp âm dương của a và b

```
a = 7044424580711
b = 2033587235192
a - b = 5010837345519
Press any key to continue . . .
```

```
a = 50000000000000
b = -555555555555
a - b = 50055555555555
Press any key to continue . . .
```

```
a = -235784400488316
b = 907220125703357
a - b = -1143004526191673
Press any key to continue . . .
```

```
a = -76230237770146
b = -508332061483283235
a - b = 508255831245513089
Press any key to continue . . .
```

➤ *Phép ** :

```
a = 111111111
b = 111111111
a * b = 12345678987654321
Press any key to continue . . .
```

```
a = 4820163720356
b = -50235723
a * b = -242144409470453477388
Press any key to continue . . .
```



```
a = -632570157325
b = -3110438802
a * b = 1967570762330924524650
Press any key to continue . . .
```

➤ *Phép /:*

```
a = 327502395225
b = 5225
a / b = 62679884
Press any key to continue . . .
```

```
a = 5168620779310
b = -8743600135
a / b = -591
Press any key to continue . . .
```

```
a = -9947662682359
b = 69253296
a / b = -143641
Press any key to continue . . .
```

```
a = -366973016592
b = -672357
a / b = 545800
Press any key to continue . . .
```

c. Toàn tử dịch trái (<<), dịch phải (>>):

➤ *Phép dịch phải:*

a: 283652010 >>20: 270 >>1: 141826005 >>1: 70913002 >>1: 35456501 >>1: 17728250 >>1: 8864125 >>1: 4432062 >>1: 2216031 >>1: 1108015 >>1: 554007 >>1: 277003 >>1: 138501 >>1: 69250 >>1: 34625 >>1: 17312 >>1: 8656 >>1: 4328 >>1: 2164 >>1: 1082 >>1: 541 >>1: 270 Press any key to continue . . .	a: 17 <<20: 17825792 <<1: 34 <<1: 68 <<1: 136 <<1: 272 <<1: 544 <<1: 1088 <<1: 2176 <<1: 4352 <<1: 8704 <<1: 17408 <<1: 34816 <<1: 69632 <<1: 139264 <<1: 278528 <<1: 557056 <<1: 1114112 <<1: 2228224 <<1: 4456448 <<1: 8912896 <<1: 17825792 Press any key to continue . . .
--	--

➤ *Phép dịch trái:*

a: -42 <<20: -44040192 <<1: -84 <<1: -168 <<1: -336 <<1: -672 <<1: -1344 <<1: -2688 <<1: -5376 <<1: -10752 <<1: -21504 <<1: -43008 <<1: -86016 <<1: -172032 <<1: -344064 <<1: -688128 <<1: -1376256 <<1: -2752512 <<1: -5505024 <<1: -11010048 <<1: -22020096 <<1: -44040192 Press any key to continue . . .	a: -74952810 >>20: -72 >>1: -37476405 >>1: -18738203 >>1: -9369102 >>1: -4684551 >>1: -2342276 >>1: -1171138 >>1: -585569 >>1: -292785 >>1: -146393 >>1: -73197 >>1: -36599 >>1: -18300 >>1: -9150 >>1: -4575 >>1: -2288 >>1: -1144 >>1: -572 >>1: -286 >>1: -143 >>1: -72 Press any key to continue . . .
--	--

8. Nguồn tài liệu tham khảo :

Slide bài giảng môn Kiến trúc máy tính và hợp ngữ - lớp CQ2016/4

9.Đánh giá:

Nội dung	Hoàn thành
Hàm nhập	✓
Hàm xuất	✓
Hàm chuyển đổi Qint thập phân sang nhị phân	✓
Hàm chuyển đổi Qint nhị phân sang thập phân	✓
Hàm chuyển đổi Qint nhị phân sang thập lục phân	✓
Hàm chuyển đổi Qint thập phân sang thập lục phân	✓
Các operator +, -, *, /	✓
Các toán tử: AND "&", OR " ", XOR "^", NOT "~"	✓
Các toán tử: dịch trái "<<", dịch phải ">>"	✓
Toàn đồ án:	100%