KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ

GỢI Ý LÀM PROJECT 1

BIỂU DIỄN SỐ HỌC TRÊN MÁY TÍNH

GVHD: LVLONG

1. Biểu diễn Số nguyên lớn

Để biểu diễn số nguyên lớn hơn phạm vi 4byte có nhiều cách. Trong hướng dẫn này sẽ gợi ý cách biểu diễn trong phạm vi 8 byte không dấu. Từ đó sinh viên có thể phát triển để biểu diễn các số nguyên có dấu, trong phạm vi lớn hơn tùy ý.

Một cách thông thường và đơn giản để biểu diễn số nguyên 8byte không dấu. Sử dụng cấu trúc sau:

```
typedef struct BigInt
{
    char bit[64];
}
```

- Cấu trúc trên sử dụng mảng 1 chiều kiểu char gồm 64 phần tử để lưu 64 bit của số nguyên 8 byte.

- Với cách này việc xử lý tính toán khá đơn giản do việc truy xuất dãy bit trên mảng 1 chiều khá dễ dàng.

- Tuy nhiên, cách biểu diễn này có hạn chế về mặt lưu trữ: sử dụng mảng char 64 phần tử để biểu diễn 64bit của số nguyên → dùng 64 byte để biểu diễn 8 byte → **lãng phí vùng nhớ.**

- Để khắc phục vấn đề lãng phí vùng nhớ, ta có thể khai báo cấu trúc kiểu dữ liệu mới như sau:

```
typedef struct BigInt
{
    Int data[4];
}
```

(Sinh viên có thể tự định nghĩa lại 2 kiểu dữ liệu Dint (8byte) và Qint (16byte) cấu trúc tương tự như trên.)

- Kiểu dữ liệu trên sử dụng 64 bit tương ứng với mảng 2 int để biểu diễn 64 bit của số nguyên 8 byte.
- Vậy làm sao để biểu diễn số nguyên 8 byte thành kiểu dữ liệu BigInt. Ta cần giải quyết lần lượt các vấn đề sau:

+ Do số nguyên lớn hơn 4 byte nên không thể dùng lệnh scanf (“%d”,…) để đọc giá trị vừa nhập → chỉ có thể đọc dãy số vừa nhập bằng cách đọc theo chuỗi.

+ Làm sao để chuyển đổi dãy số ở dạng chuỗi về kiểu BigInt? Ta có thể giải quyết bằng thuật toán đã học (Tính các bit bằng cách đem dãy số chia 2 và lấy phần dư), ta có hàm để chuyển đổi chuỗi số sang BigInt như sau:

```
void StrToBigInt(string X, BigInt &a)
{
    int i = 63; // bắt đầu set bit tại vị trí cuối cùng
    while (X ! = "0")
    {
           int bit  = (X[X.length()-1] - 48) % 2; // Tính phần dư
           SetBit(a, i, bit); // Tạo bit (biến bit) tại vị trí bit i của biến a
           X=StrDiv2(X); // Chia chuỗi số X cho 2
           i--;

    }
}
```

- Hàm tạo giá trị bit tại vị trí bit i của Kiểu dữ liệu BigInt

```
void SetBit(BigInt &a, int bit, int i)
{
    // Tùy theo vị trí i ta quyết định thao tác xử lý bit trên phần tử nào của
    // biến BigInt.
    // VD: i = 31 thao tác bit thứ 31 của a.data[0],
    // i = 32 thao tác bit thứ 1 của a.data[1]
}
```

- Hàm chia một dãy số nguyên dạng chuỗi cho 2

```
string StrDiv2(string X)
{
   //Sinh viên tự viết
}
```

- Để xem được giá trị của biến BigInt, ta xây dựng hàm xuất giá trị biến BigInt ra màn hình như sau:

```
void PrintBigInt(BigInt a)
{
    /*
    - Truy xuất dãy bit của a và đổi nó sang thập phân (ở dạng chuỗi) theo thuật toán
      đã học:
    - S += ∑_{i=0}^{63} a_i * 2^{63-i} (tính ở dạng chuỗi, có thể định nghĩa thêm 2 hàm: tính kết quả
      lũy thừa 2^n ở dạng chuỗi, và hàm nhân 2 chuỗi với nhau).
    - Trong đó: S là số thập phân của dãy bit, bit_i là bit thứ i của kiểu dữ liệu
      BigInt.
    - Print dãy số thập phân (ở dạng chuỗi) tìm được ra màn hình.
    */

}
```

(Sinh viên đọc thêm file hướng dẫn các thao tác xử lý bit trên moodle)

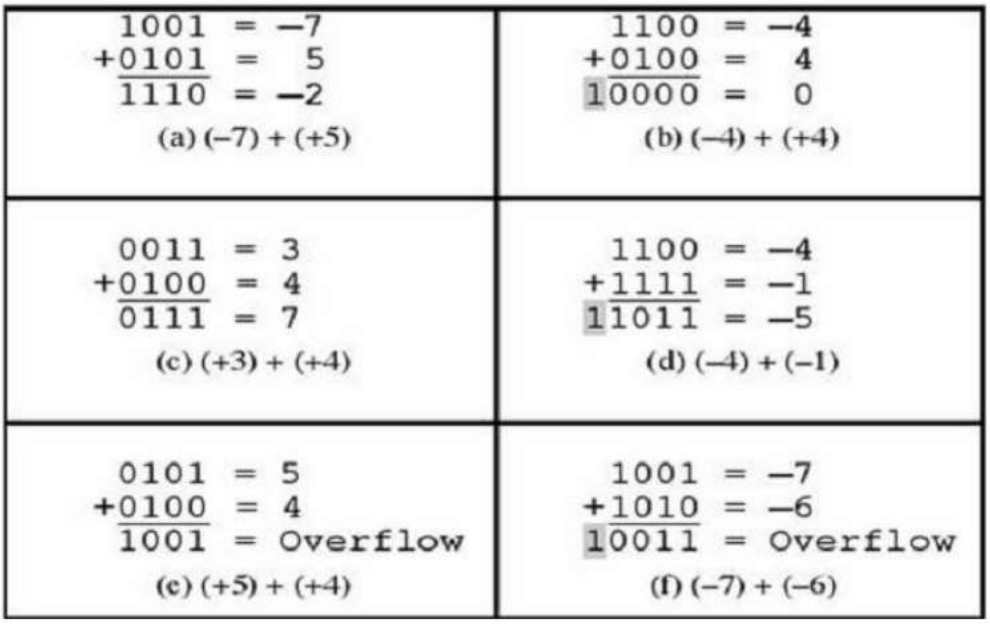
2. Các thao tác trên số nguyên lớn

- Cac hàm tính toán được viết theo phương thức Operator như sau:

a. Hàm tổng 2 số BigInt

```
Bigint operator + (BigtInt a, Bigint b)
{
   //...
}
```

Thực hiện cộng tương ứng trên từng bit, ví dụ cộng 2 số 4 bit:



b. Hàm hiệu 2 số BigInt

```
Bigint operator - (BigtInt a, Bigint b)
{
   //...
}
```

Nguyên tắc cơ bản: đưa về phép cộng: A – B = A + (-B) = A + (số bù 2 của B)

c. Hàm tích 2 số BigInt

```
Bigint operator / (BigtInt a, Bigint b)
{
   //...
}
```

THUẬT TOÁN NHÂN 2 SỐ KHÔNG DẤU

- Giả sử ta muốn thực hiện phép nhân M x Q với
  - Q có n bit
- Ta định nghĩa các biến:
  - C (1 bit): đóng vai trò bit nhớ
  - A (n bit): đóng vai trò 1 phần kết quả nhân ([C, A, Q]: kết quả nhân)
  - [C, A] (n + 1 bit) ; [C, A, Q] (2n + 1 bit): coi như các thanh ghi ghép
- Thuật toán:

```
Khởi tạo: [C, A] = 0; k = n
Lặp khi k > 0
{
    Nếu bit cuối của Q = 1 thì
        Lấy (A + M) → [C, A]

    Shift right [C, A, Q]
    k = k – 1
}
```

CẢI TIẾN NHÂN CÓ DẤU:

- Chuyển thừa số 2 thành số dương
- Nhân theo thuật toán nhân không dấu
- Nếu khác dấu, đổi dấu

a. Hàm thương 2 số BigInt

```
Bigint operator * (BigtInt a, Bigint b)
{
   //...
}
```

THUẬT TOÁN CHIA KHÔNG DẤU

Khởi tạo: A = n bit 0 nếu Q > 0; A = n bit 1 nếu Q < 0; k = n
Lặp khi k > 0
{
    Shift left (SHL) [A, Q]

    A – M → A
        # Nếu A < 0: Q_0 = 0 và A + M → A
        # Ngược lại: Q_0 = 1

    k = k – 1
}
Kết quả: Q là thương, A là số dư

CẢI TIẾN CHIA CÓ DẤU:

- Thực hiện như phép chia không dấu
- Nếuso chiavà sốbị chia khác dấu thì đổi dấu thương