BÁO CÁO THỰC HÀNH

LẬP TRÌNH HỆ THỐNG

Tên bài Thực hành: Lab 01 - Lớp: NT209.P12.ANTT

Giáo viên hướng dẫn: Đỗ Thị Thu Hiền

Họ và tên Sinh viên	MSSV
Nguyễn Trần Minh Khôi	23520780
Lê Đăng Khôi	23520766
Vương Thành Đạt	23520281

Bài tập 1.1: int bitOr(x,y)

```
int bitOr(int x, int y)
{
    x = ~x & ~y;
    x = ~x;
    return x;
}
```

Yêu cầu: or hai bit chỉ sử dụng & và ~.

Giải thích:

Sử dụng công thức DeMorgan: ~((A)|(B))= (~A)&(~B)

Áp dụng công thức, sau đó lật bit của x lại sẽ được (A)|(B)=~((~A)&(~B))

Số phép tính: 6

Bài tập 1.2: int negative(x)

```
int negative(int x)
{
    x = ~x;
    x = x + 1;
    return x;
}
```

Yêu cầu: tính giá trị -x không sử dụng dấu -.

Giải thích:

+ Biểu diễn số nguyên dưới dạng bù 2

+ Sử dụng toán tử ~ để có được bù 1 của x

+ Sau đó cộng thêm 1 bit sẽ có được dạng bù 2 của x à có được số âm của x.

Số phép tính: 4

Bài tập 1.3: flipByte (x,n)

```
int flipByte(int x, int n) {
    int a = x;
    int mask = (0xFF << (n << 3));
    a = ~(a & mask);
    x = (x | mask) & a;
    return x;
}
```

Yêu cầu: lật một byte thứ n của số nguyên x.

Giải thích:

+ Tạo một biến tạm giữ giá trị của x để mang đi tính toán.

+ Tạo một mask để lấy ra những bit cần lật (0b11111111)

+ Shift left n << 3 là đưa những bit 1 vừa được tạo đến vị trí cần lật.

+ Ví dụ: Cần lật ở bit thứ 0 thì mask = 0xFF << (0 << 3), có nghĩa là giữ nguyên vị trí của những bit 1 (vị trí byte thứ 0).

+ Cho biến tạm a and với mask (giữ lại ví trị cần lật) và sau đó lật các bit lại, kết quả thu có byte đã lật, còn những vị trí khác sẽ là 1.

+ Ví dụ: a = 0x12345678, mask = 0x000000FF. Khi thực hiện And bit sẽ cho ra a = 0x00000078. Khi lật bit lại a = 0xFFFFFF**A9**

+ Lúc này đã có những bit cần lật, sau đó đem x or với mask để đưa vị trí lật về 1, cuối cùng cho and lại với a. Khi này những vị trí cần được giữ sẽ được giữ lại và cho ra kết quả cuối cùng.

+ Tiếp tục với ví dụ trên: x = 0x**123456**78, mask = 0x000000FF. Khi thực hiện Or bit sẽ có được x = 0x**123456**FF, sau đó And bit giữa a= 0xFFFFFF**A9** và x ta được x = 0x123456A9 (vị trí in đậm là vị trí cần được giữ lại).

Số phép tính: 10

Câu 1.4: Hàm int mod2n(int x, int n)

```
int mod2n(int x, int n)
{
    int a = x >> n;
    a = a << n;
    x = x ^ a;
    return x;
}
```

Yêu cầu: Tính kết quả phép **lấy dư x%2^n**

Định nghĩa:

+ Trong hệ nhị phân, phép dịch bit sang trái (<<) sẽ nhân 2^n với số bị dịch trong hệ thập phân.

+ Trong hệ nhị phân, phép dịch bit sang phải (>>) sẽ chia 2^n với số bị dịch trong hệ thập phân.

+ Phép XOR: dùng để so sánh 2 bit đầu vào, trả về 0 nếu 2 bit giống nhau, trả về 1 nếu 1 trong 2 bit là 1.

Giải thích: Với n là số bit cần dịch

- Dịch phải n bit tương đương với phép chia nguyên cho 2^n
- Dịch trái n bit tương đương với phép nhân 2^n nhưng chỉ thu được lại phần chia hết
- XOR giá trị ban đầu với giá trị đã dịch trước đó, sẽ loại bỏ phần chia hết và giữ lại bit dư.

Số phép tính: 6

Câu 1.5: Hàm int divpw2(unsigned int x, int n)

```
30 unsigned int divpw2(unsigned int x, int n) {
31     int a=(n>>31);
32     x=((~a)&(x>>n))|((a)&(x<<negative(n)));
33     return x;
34 }
```

Yêu cầu: x >0 && -31<=n<=31. Tính kết quả x/2^n

Giải thích:

- Nếu n dương ta tiến hành dịch phải n bit, nếu n âm thiến hành dịch trái n bit.
- Do kết quả của hàm phụ thuộc vào dấu của tham số n nên ta lấy dấu của n trước. Kết quả của biểu thức lấy dấu sẽ là 0x00000000 hoặc 0xFFFFFFFF tương ứng với trường hợp n dương hoặc âm.
- Ta dùng biểu thức shanon F=~A.B+A.C để biểu diễn (A âm thì lấy B, A dương thì lấy C)
- Biểu diễn công thức trên dưới dạng bit ta được: ((~a)&(x>>n)|(a&(x<<negative(n)).

Số phép tính: 9

Câu 2.1: Hàm int isEqual(int x, int y)

```
int isEqual(int x, int y)
{
    return !(x^y);
}
```

Yêu cầu: Kiếm tra 2 input x và y có bằng nhau không. Trả về 1 nếu bằng, 0 nếu khác.

Giải thích:

- Ta có phép XOR là phép dùng trong mục đích so sánh 2 input về bit, nếu 2 input giống nhau sẽ trả về 0, ngược lại sẽ trả về 1.
- Ở đây, ta chỉ cần dùng phép loại trừ trường hợp 2 input giống nhau để luôn có giá trị trả về là 1 là sẽ như 1 hàm so sánh.

Số phép tính: 2

Bài tập 2.2: is16x(int x)

```
42  int is16x(int x) {
43      int a=x>>4;
44      a=a<<4;
45      x=x^a;
46      x=!x;
47      return x;
48  }
```

Yêu cầu: kiểm tra một số nguyên không âm x có chia hết cho 16 hay không

Giải thích:

- Ta có 16=24 là chia cho 16 đồng nghĩa với phép dịch bit sang phải 4 bit. (a=x>>4)
- Do phép dịch bit sang phải sẽ trả về giá trị chia nguyên của phép tính do đó t đem kết quả dịch bit trên dịch ngược sang trái 4 bit để phục hồi giá trị ban đầu (nhân lại với 16) (a=a<<4)
- Nếu giá trị x ban đầu chia hết cho 16 thì giá trị sau phép dịch sẽ bằng với giá trị x ban đầu, nếu không thì x không chia hết. để kiểm tra xem 2 giá trị có bằng nhau không, ta sử dụng phép toán xor để so sánh từng phần tử, nếu giống nhau, kết quả phép tính sẽ bằng 0, ngược lại thì khác 0. (x^a)
- Do kết quả thu được từ phép xor ngược với yêu cầu đề bài (chia hết trả về 1, ngược lại trả về 0) nên ta phủ định giá trị trên và trả về.

Số phép tính: 8

Bài tập 2.3: isPositive(int x)

```
50 int isPositive(int x) {
51     x=x+negative(1);
52     x=x>>31;
53     x=!(x);
54     return x;
55 }
```

Yêu cầu: trả về 1 nếu x>0.

Giải thích: nếu sử dụng cách xét dấu bình thường (x>>31) thì không thể phân biệt giá trị bằng 0 và lớn hơn không, do đó ta tiến hành lấy x trừ 1 trước để loại trừ trường hợp 0 (x+negative(1)) rồi xét dấu như bình thường.

Số phép tính: 6

Bài tập 2.4: int isGE2n

```
58 int isGE2n(int x, int n) {
59     int a=1<<n;
60     a=~a+1;
61     x=x+a;
62     x=x>>31;
63     x=!(x);
64     return x;
65 }
```

Yêu cầu: trả về 1 nếu $x>=2^n$ (x>0, 30>=n>=0)

Giải thích:

- Trước hết ta cần tính toán giá trị bù 2 của 2^n. (~(1<<n)+1)
- Sau đó lấy x cộng với giá trị vừa tính và đem đi xét dấu, vì giá trị được tính toán là giá trị bù 2 nên không có sự sai lệch về giá trị của kết quả. (x>>31).

Số phép tính: 11

Kết quả

```
1.1 bitOr       Pass.
1.2 negative    Pass.
1.3 flipByte    Pass.
1.4 mod2n       Pass.
1.5 divpw2      Advanced Pass.
2.1 isEqual     Pass.
2.2 is16x       Pass.
2.3 isPositive  Pass.
2.4 isGE2n      Pass.
------
Your score: 10.0
PS D:\vscodeblue\ltht>
```