[CSED211] 컴퓨터 SW시스템 개론  
Lab Session Howework #1

학번 : 20180551

학과 : 컴퓨터공학과

이름 : 이준석

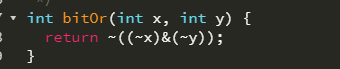
**Problem 1**

~와 &만을 이용하여 비트 연산자 OR을 구현하는 문제이다.

x or y = not( not ( (x or y) ) ) = not ( (not x) and (not y) )

이므로 코드로 다음과 같이 구현할 수 있다.

( ‘not’은 ‘~’에 대응하고 ‘and’는 ‘&’에 대응한다. )



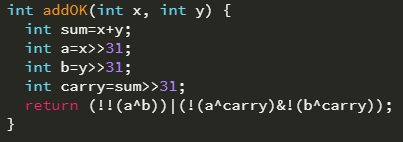
**Problem 2**

두 피연산자를 더하였을 때 overflow가 발생하는지 발생하지 않는지 확인해주는 함수를 구현하는 문제이다. 덧셈이 정상적으로 가능하면 true 이고 아니라면 false이다.

Overflow는 다음과 같은 조건에서 발생하지 않는다. 즉, 정상적으로 덧셈이 수행된다.

1. 둘의 MSB가 다를 때
2. 둘의 MSB가 같고 그것들과 MSB 자리로 들어오는 carry의 값도 같을 때

1번 또는 2번을 충족하면 덧셈이 가능하다. 따라서 코드로 다음과 같이 구현된다.



a = x>>31와 b = y >> 31은 각각 MSB들로 모든 bit를 채우기 위함이다. Sum과 carry는 carry 값으로 모든 bit를 채우기 위함이다.

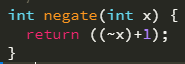
Return type이 bool이 아니라 int여서 True나 false로 값을 반환할 수 없기 때문에, 덧셈이 정상적으로 가능하다면 1을 반환하고 overflow가 발생한다면 0을 반환하기 위해서 ‘!’ 연산을 두 번 수행하였다.

**Problem 3**

절대값은 같고 부호는 반대인 수를 반환하는 함수이다.

입력된 값의 2의 보수를 구하면 그것이 바로 구하려고 하는 값이다.

X의 2의 보수를 구하기 위해서는 x의 모든 0을 1로 바꾸고 모든 1을 0으로 바꾼 다음에 1을 더해주어야 한다. 그래서 코드로 다음과 같이 구현하였다.



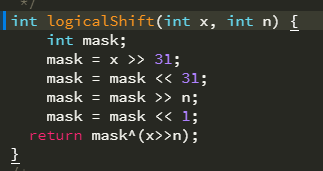
**Problem 4**

Logical shift를 구현하는 문제이다. 기본적으로 ‘>>’ 연산자는 arithmetic shift인데, 이것은 비어지는 칸을 MSB로 채우는 방식이다. 그러므로 0으로 채울 수 있게 구현해야 한다.

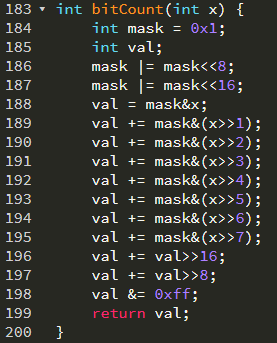
n칸 만큼 이동한 후에 logical shift에서는 0으로 채워져야할 칸에 MSB로 채워지는데, 이것을 해소하기 위한 방법으로는 0과의 AND를 이용하여 0으로 바꾸거나 MSB와 같은 비트를 가진 mask를 만들어서 XOR을 이용하여 0으로 바꾸는 방법이 있다.

후자의 방법을 이용하여 구현하였다. 왼쪽부터 n칸만큼 MSB로 채워진 mask를 만들어야 한다.

1. 우선 전체를 MSB로 채웠다. ( mask = x >> 31; )
2. n칸만 MSB로 채워지고 나머지는 0이어야한다. 그래서 우선 가장 왼쪽 1칸만 MSB로 채운다. 그리고 나머지는 0으로 채운다. ( mask = mask << 31; )
3. n칸 만큼 오른쪽으로 arithmetic shift를 하면 왼쪽 (n+1)칸은 MSB, 나머지 칸은 0으로 채워진다. ( mask = mask >> n; )
4. n칸만 채워져야 하는데 원래 있던 1칸으로 인해 n+1칸이 채워졌으므로 다시 왼쪽으로 한 칸 이동시킨다. ( mask = mask << 1; )
5. 그렇게 만든 mask과 n칸 만큼의 arithmetic shift를 수행한 것을 xor하면 logical shift가 이루어진다.



**Problem 5**



Mask에 00000….000(0이 31개)1(1이 1개)를 대입한다.

그리고 총 32비트이므로 8비트 단위로 끊어서 x의 1의 개수를 세기 위하여

00000001 이 4개 있는 mask로 만들어준다. (line 186과 line 187)

X를 비트 1칸씩 이동시키며 더해준다. (line 188 ~ line 195) 8비트 단위로 끊었으므로 총 8번의 덧셈이 수행된다.

8비트 단위로 총 4개의 구역에 덧셈 결과가 저장되어 있으므로 다시 그 4개 구역들을 모두 합쳐준다. (line 196 ~ line 197)

마지막으로 연산하는데 사용한 칸들에 남아있는 1을 무시하기 위해 필요한 칸만 선택하는 연산을 수행하도록 한다. (line 198) 그러면 1의 개수가 모두 count된 결과가 val에 저장된다.