CSED211 LAB1 report

20210054 정하우

<bitNor>

return ~x & ~y

=>~(x|y)=(~x)&(~y)=>드모르간의 법칙

<isZero>

return !x

=>0이면 1을, 0이 아닌 숫자면 0을 return한다

<addOK>

int sum = x + y

=>x+y를 한 값을 sum에 대입한다

int sum_check = sum >> 31

int x_check = x >> 31

int y_check = y >> 31

=> 양수면 msb가 0이므로 0x00000000가 되고, 음수면 msb가 1이므로 0xFFFFFFFF 가 된다

return !(((sum_check & (~x_check) & (~y_check))) | (((~sum_check) & x_check & y_check)))

=>x와 y 둘 다 양수인 경우에는 위의 논리에 따라 x_check, y_check 둘 다 0x00000000가 된다. 이때 overflow가 발생하면 sum은 음수가 되므로 sum_check는 위의 논리에 따라 0xFFFFFFFF가 된다. 이때 x_check와 y_check 를 ~연산자를 통해 계산하면 ~x_check, ~y_check 둘 다 0xFFFFFFFF 가 된다. 이 경우에 &로 ~x_check, ~y_check, sum_check를 연산하면 결과값은 0xFFFFFFFF가 된다.

위의 경우를 제외하고 overflow가 나오는 경우는 딱 하나 더 있다. 바로 x와 y 둘 다 음수이지만 sum값이 양수인 경우이다. x와 y 둘 다 음수인 경우에는 위의 논리에 따라 x_check, y_check 둘 다 0xFFFFFFFF가 된다. 이때 overflow가 발생하면 sum은 양수가 되므로 sum_check는 위의 논리에 따라 0x00000000가 된다. 이때 sum_check를 ~연산자를 통해 계산하면 ~sum_check는 0x00000000 가 된다. 이 경우에 &로 x_check, y_check, ~sum_check를 연산하면 결과값은 0xFFFFFFFF가 된다.

이 두 경우를 | 연산자를 통해 계산하면, 위의 두 경우 중 한 경우에서 결과값이 무조건 0xFFFFFFFF가 된다. 이때 전체 값을 ! 연산자를 통해 계산하면 overflow가 발생하면 0을, overflow가 발생하지 않으면 음수를 return한다.

<absVal>

int msb = x >> 31

=>msb는 x가 양수이면 0xFFFFFFFF, 음수이면 0x00000000가 된다

int re = (msb & (1 + (~x))) | ((~msb) & x)

return re

=>만약 x가 음수이면 msb가 0xFFFFFFFF가되어 1 + (~x)를 계산하게 된다. 1 + (~x)는 곧 -x이므로 양수값이 return됨을 알 수 있다.

만약 x가 양수이면 msb가 0x00000000가되어 x를 계산하게 된다. 이 경우 양수인 x가 그대로 return됨을 알 수 있다.

<logicalShift>

x >>= n

=>x를 n만큼 right shift로 민다

int check = ((1 << 31) >> n) << 1

=>가장 큰 bit 부터 n까지 1, 나머지 bit들은 0으로 이루어진 bit를 check에 대입한다.

check = ~check

=>check를 !연산자를 통해 계산하면 x에서 shift 연산을 통해 밀려 들어온 가장 큰 bit부터 n bit까지 0, 나머지 bit가 1인 숫자가 대입된다.

return x & check

=>새로 밀려 들어온 bit들은 0와 &계산을 하여 0이되고, 나머지 bit들은 1과 &계산을 하여 그대로 나온다.