시스템소프트웨어(CB2400108-059) - HW1 : Datalab

정보컴퓨터공학부 202255513 김대욱

(1) bitAnd

De Morgan 법칙에 따르면, ~(x & y) = (~x) | (~y) 가 성립한다. 즉, x & y = ~(~x | ~y) 가 성립한다. 따라서, ~ 과 | 만을 사용하여 bitAnd를 계산하면 아래와 같다.

(2) getByte

n번째 바이트를 추출하기 위해서는 해당 바이트를 기준으로 오른쪽에 있는 모든 바이트를 버리고, 해당 바이트만 남겨야 한다. 먼저, shift 변수를 통해 n번째 바이트의 위치를 계산하고, x를 shift만큼 right logical shift하여 n번째 바이트를 오른쪽 끝으로 이동시킨다. 마지막으로, &연산자를 사용하여 x와 0xFF를 AND연산하면 오른쪽 끝 1바이트만 남길 수 있다.

(3) logicalShift

right shift의 경우, logical과 arithmetic 연산이 모두 가능하다. signed 정수의 경우 arithmetic 연산으로 수행하기 때문에, >> 연산을 수행한 이후 왼쪽에 0을 채워주어야한다. mask 변수를 통해 오른쪽 끝에서부터 n개의 비트가 0인 값을 갖는다. x를 n만큼 오른쪽 shift를 한 다음, ~mask와 AND연산을 수행하면, 좌측에 생긴 불필요한 1을 제거해줄 수 있다.

(4) bitCount

먼저 0xF0F로 mask를 만들어주고, 비트카운트를 수행하기 위한 카운트는 0x11111111로 만들어준다. 생성된 마스크와 카운트 값을 이용하여, x를 1비트씩 shift하면서 4비트 내의 1의 개수를 모두 합쳐준다. 그리고 16비트 shift하여 좌측 16비트와 우측 16비트를 합친 후, 4비트씩 묶어 mask와 AND연산을 수행하고, 다시 8비트씩 묶어 mask와 AND연산하여 값을 구한다.

(5) bang

주어진 수와 2의 보수값을 OR 연산한 후, Sign Bit를 추출하여 1을 더한다.

(6) tmin

첫 비트가 1인 수가 tmin 값이므로, 1을 31번 left shift한 값이 된다.

(7) fitsBits

n비트의 최댓값을 구한 후, 주어진 수를 n비트 크기로 왼쪽 shift한 이후 다시 right shift하여 원래 수와 같은지 확인한다. 같으면 1, 다르면 0을 반환한다.

(8) divpwr2

먼저 주어진 x값이 음수인지 판별하여 isNeg 변수에 저장하고, 음수일 경우, 2의 n제곱 값을 구한 후 temp 변수에 저장한다. isNeg & 1을 하게 되면, x 값이 음수이면 1, 양수이면 0이 반환되므로 이 결과에 n을 shift하여 2의 n제곱 값을 게산한다. x 값에 temp값을 더한 후 n비트만큼 right shift하여 2의 n제곱으로 나눈 몫을 계산한다.

(9) negate

2의 보수 값을 반환해주면 되므로, 1의 보수를 구해주고 1을 더한 값을 return한다.

(10) isPositive

x를 오른쪽으로 31비트 이동시켜 부호 비트만 남긴 후, 부호 비트가 1이면 x가 음수이므로 결과값은 0이 되어야 한다. ! 연산자를 이용해 0을 반환한다. 반대로, 부호 비트가 0인 경우, x가 양수이므로, 1을 반환해야한다.

(11) isLessOrEqual

x와 y의 부호를 각각 xSign과 ySign에 저장한다(음수는 1, 양수는 0). x와 y의 부호가 다르고 x가 음수일 경우 1을 return한다. 그리고, x와 y의 부호값이 다른 경우 diff 변수에 1을 저장한 다음, 부호 값이 같고 계산 값이 0인 경우를 고려하여 return한다. y가 양수이고, x가 음수라면 overflow에 의해 0이 나오는 것을 고려해야 한다.

(12) ilog2

leftmost 1부터 LSB까지 모두 1로 채운 후, 해당 숫자를 1의 개수를 bitCount하고 1을 빼준 값을 출력해준다. 4번 문제와 상당히 유사한 과정을 거치게 된다.

(13) float_neg

unsigned uf를 floating point expression으로 생각하고 sign Bit만 수정해주는 함수이다. 그러나 만약 exp Bit가 모두 1이고, frac Bit가 0이 아닌 경우, Not a Number이므로 if문을 통해 예외처리 할 수 있다.

(14) float_i2f

정수 x를 32비트 부동소수점으로 변환하는 함수이다. 먼저 x가 0인 경우와 음수인 경우를 처리하고, leftmost 1을 찾아 지수값을 계산한다. 이후, x의 소수부분을 계산하고 반올림을 수행한다. 그리고 exp, s, frac을 조합하여 비트로 표현한 후 반환한다.

(15) float_twice

uf 는 denormalized, normalized, NaN 값을 가질 수 있는데, uf가 denormalized라면 sign Bit를 제외한 나머지 비트를 1칸 left shift하는 것이 2를 곱하는 것과 같은 결과를 얻을 수 있다. 그리고 만약 uf가 normalized인 경우, exp에 1을 더해주면 2를 곱하는 것과 같은 결과를 얻는다.