CSED211 LAB1 report

20210054 정하우

<negate>

return ~x + 1 : 2의 보수법 계산대로, ~ 연산자 사용 후 1을 더해 준다.

int isLess(int x, int y)

int sign_x = (x >> 31) & 0x00000001 : x의 부호정보를 담는다

int sign_y = (y >> 31) & 0x00000001 : y의 부호정보를 담는다

int sign_result = (((~x + 1) + y) >> 31) & 0x00000001 : -x+y의 부호정보를 담는다

int sign_diff = sign_x ^ sign_y : x 와 y의 부호정보를 nor로 계산한다.

return (sign_diff & sign_x) | (!(sign_diff | sign_result)) & (!!((~x + 1) + y)) : x와 y의 부호가 같을 경우 -x+y를 한 부호의 정보에 따라 true인지 false가 결정된다. 양수면 0이되어 0 or 0이 되어 0이되고 !0이 되서 최종적으로 1이된다. 하지만 x=y인 경우는 제외해 주어야 한다. 이 경우는 overflow를 고려하지 않아도 된다. 부호가 다른 경우에는 or 뒷 부분은 항상 0이 된다. 이 경우는 overflow가 발생하는 상황을 고려 해 주어야 하므로 or 앞은 다음과 같이 적을 수 있다.

<float_abs>

int exp = uf & 0x7F800000 : uf의 exp정보를 담는다

int frac = uf & 0x007FFFFF : uf의 frac정보를 담는다

int abs = uf & 0x7FFFFFFF : uf의 부호를 제외한 나머지 정보를 담는다. Sign은 0(양수)이다.->절댓값이다

if ((exp == 0x7F800000) && (frac != 0x00000000))

return uf : uf가 NaN일 경우 그대로 return 한다

else

return abs : NaN이 아니면 절댓값을 return한다.

<float_twice>

int exp = (uf & 0x7F800000) >> 23 : exp 정보를 담는다

if (exp==0x00000000) : Denormalized 인 경우

exp = 0x000000FF :
uf = (uf & 0x80000000) | (uf << 1) : 어짜피 exp는 0이므로 left shift연산자로 밀고 부호만 그대로 가져간다.

if (!(exp ^ 0x000000FE)) : overflow가 발생한 경우이다.
exp = 0x000000FF : exp를 11111111로 맞춘다(special value 중 infinity value로 맞춘다)
uf = (uf & 0x80000000) | (exp << 23) : 부호는 그대로, exp는 11111111로, frac는 00000000000000000000000로 맞춰서 부호는 그대로 가져가는 무한대로 맞춘다

if (exp ^ 0x000000FF)
uf = uf + (1 << 23) : normalized인 경우 exp에 1만 더해주면 된다.

return uf :

<float_i2f>
int sign = x & 0x80000000 : 부호정보를 가져온다
int frac_temp = 0x7FFFFF
int exp : exp정보ㄹ르 가져온다
int frac : frac 정보를 가져온다
if (x == 0)
return 0 : 0을 그대로 return한다
else if (x == 0x80000000)
return 0xCF000000 : int 최솟값은 2^(-31)이므로 exp만 E+bais만큼 값을 가져가면 된다.

else
if (sign)
x = -x : x 부호를 바꿔준다.

int M = 1 : 입력받은 숫자가 2진법으로 몇 자리인지 알아내야 한다

while ((x >> M) != 0)
M++

M--

exp = M + 127 : 몇 자린지 알아내면 exp를 bais를 통해 구할 수 있다.

x = x << (31 - M) : 가장 높은 bit에 숫자 정보가 올라오도록 한다
frac = frac_temp & (x >> 8) : frac를 구한다

```
        if (M > 23)
            int r = x & 0x000000FF : 반올림 여부를 판단해야한다.
            if ((r > 128) || ((r == 128) && (frac & 0x00000001)))
                frac++ : 반올림이 된 경우이다.
                if ((frac >> 23) != 0x00000000)
                    exp++ : 자릿수가 올라간 경우이다
                    frac = 0




    return sign | (exp << 23) | frac

<float_f2i>
    unsigned sign = uf >> 31 : 부호정보를 가져온다
    unsigned exp = (uf >> 23) & 0x000000FF : exp정보를 가져온다
    unsigned frac = (uf & 0x007FFFFF) : frac정보를 가져온다

    unsigned bias = 0x0000007F : bais
    unsigned int_num = frac : int값을 바꾸기 위해서는 일단 frac정보를 가져와야한다.

    if (exp == 0x000000FF)
        return 0x80000000u : infinity value인 경우
    else if (exp < bias)
        return 0x00000000 : denormalized인 경우

    exp = exp - bias : normalized인경우

    if (exp >= 31)
        return 0x80000000u : overflow 가 발생한 경우

    if (exp > 22) : exp 값만큼 2를 계속 곱해준다
        int_num = frac << (exp - 23)
    else
        int_num = frac >> (23 - exp)

    int_num += 1 << exp
```

if (sign != 0x00000000)

    int_num = -int_num : 부호가 음수인 경우 부호를 바꿔준다

return int_num