컴퓨터 SW시스템개론 Lab2 보고서

20210207 이지현

Ⅰ. 소개

Bits.c의 6가지 문제에 대하여 함수를 작성하는 lab이다. Negate, isless, float_abs, float_twice, float_i2f, float_f2i 5개의 문제가 있고, 각 함수의 예시와 조건을 참고하여 문제를 해결할 수 있다.

Name	Description	Rating	Max Ops
`negate(x)`	`-x` without negation	2	5
`isLess(x,y)`	`x < y`?	3	24

Table 1: Arithmetic Functions

Name	Description	Rating	Max Ops
`float_abs(uf)`	Compute absolute value of `f`	2	10
`float_twice(uf)`	Computer `2*f`	4	30
`float_i2f(x)`	Compute `(float) x`	4	30
`float_f2i(uf)`	Compute `(int) f`	4	30

Table 2: Floating-Point Functions. Value `f` is the floating-point number having the same bit representation as the unsigned integer `uf`.

Ⅱ. 코드 설명

(1) int negate(int x)

~x + 1로 간단하게 표현할 수 있다.

(2) Int isLess (int x, int y)

x<y인 경우는 (1) x 부호가 -, y 부호가 + 인 경우, (2) 두 부호가 같은데 y값이 더 큰 경우이다. (1)은 &를 이용해서 sign이 1 1일때만 1이 됨을 이용해서 (x & ~y)로 나타낼 수 있고, (2)는 x^y를 이용해 부호가 같으며, x-y (x~y+1)을 이용해 총 x<y임을 나타낼 수 있다.

(3) Unsigned float_abs (unsigned uf)

Nan이 되기전 최소 수는 (1<<8 + (-1)) <<23 + 1로 나타낼 수 있는데, 이를 16진법으로나타내면 FF800000이므로 UF가 NAN min 이상이면 uf 반환, 아니라면 sign을 0으로 바꾸어서 반환한다.

(4) Unsigned float_twice(unsigned uf)

Sign과 exp를 설정해놓고, uf가 0이면 0, exp가 11111111이면 uf자체 (overflow), exp가 0이라면 uf를 1칸씩 shift하고 sign 추가, 그 외의 경우라면 exp의 자리인 23번째에 1을 추가한다.

(5) Unsigned float_i2f(int x)

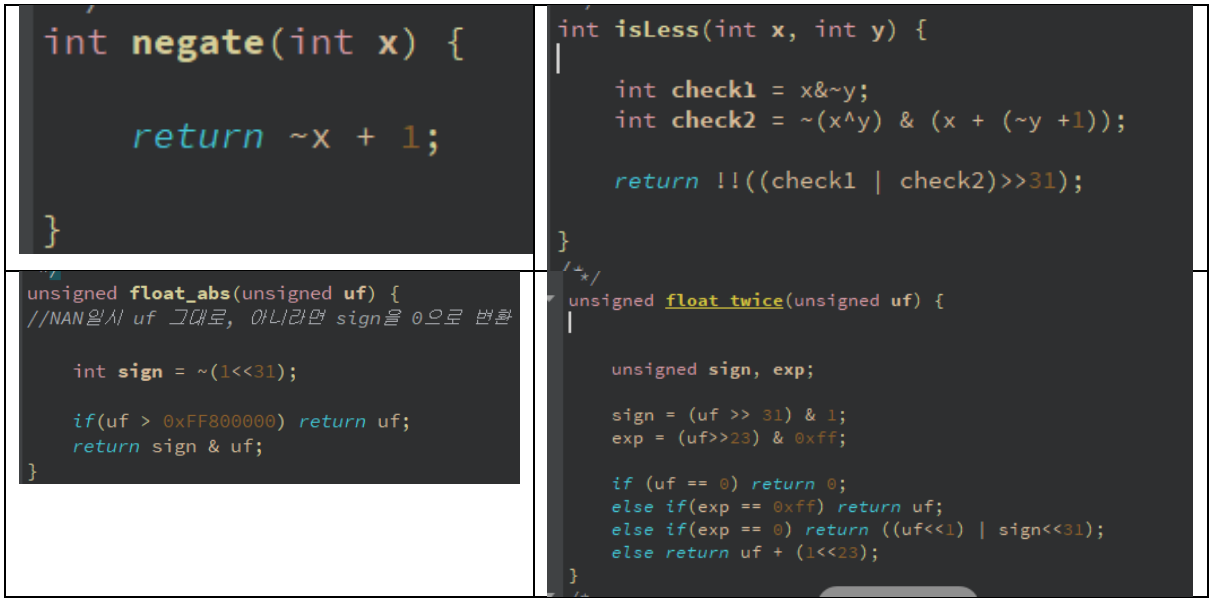
Int를 2진법 비트상 그대로 float로 옮기려면 -2^31은 2^31이 불가능하므로 예외처리를 해주고 음수를 모두 양수로 변화시킨다음 sign, frac, exp를 구한다. 자세히는 exp는 while을 통해, frac은 mask를 씌워 구한다. bias 127 이후의 값들이나 반올림이 필요한 값들을 frac++ 시킨다. 그다음 다 더해 값을 구할 수 있다.

(6) Int float_f2i (unsigned uf)

기존 float sign, exp, frac을 나눈 후, NAN과 infinite 경우의 예외처리를 진행한 후, denormalized, normalized, overflow, exp>22 shift 경우를 나누어 계산한다. 이 때, result 값에서 sign이 1인 경우 다시 음수로 바꾸어주는 작업도 진행한 후 리턴한다.

Ⅲ. 실행 사진 첨부

(1) Problem 1-5



```
int negate(int x) {

    return ~x + 1;

}
```

```
int isLess(int x, int y) {

    int check1 = x&~y;
    int check2 = ~(x^y) & (x + (~y +1));

    return !!((check1 | check2)>>31);

}
```

```
unsigned float_abs(unsigned uf) {
//NAN일시 uf 그대로, 아니라면 sign을 0으로 변환

    int sign = ~(1<<31);

    if(uf > 0xFF800000) return uf;
    return sign & uf;
}
```

```
unsigned float_twice(unsigned uf) {

    unsigned sign, exp;

    sign = (uf >> 31) & 1;
    exp = (uf>>23) & 0xff;

    if (uf == 0) return 0;
    else if(exp == 0xff) return uf;
    else if(exp == 0) return ((uf<<1) | sign<<31);
    else return uf + (1<<23);
}
```

```c
unsigned float_i2f(int x) {
//int를 2진법 비트상 그대로 float로 읽기려면
//-2^31은 2^31이 존재하지 않으므로 예외처리를 해주고
//음수를 모두 양수로 변화시키고 sign, frac, exp로 

    int sign, exp, frac, bias;
    int mask = 1<<31;
    sign = x & mask;
    bias = 127;
    int E = 158;

    if(x==0) return 0;
    if (x==mask) return (mask | (158<<23));
    if(sign) x = ~x + 1;

    while(!(x&mask)){
        x = x <<1;
        E = E - 1;
    }
    frac = (x&(~mask)) >> 8;
    if (x&0x80 && ((x&0x7f) >0 || frac & 1))
        frac = frac + 1;

    return (sign + (E <<23) + frac);

}
```

```c
int float_f2i(unsigned uf) {


    unsigned sign, exp, frac, bias = 127;
    unsigned res;

    sign = (uf >> 31) & 1;
    exp = (uf>>23) & 0xff;
    frac = uf & 0x7fffff;
    int E = exp - bias;

    if(exp == 0xff && frac == 0)
        return 0x80000000u;
    else if(exp == 0xff && frac != 0)
        return 0x80000000u;
    else if(!exp)
        return 0;
    else if(E < 0)
        return 0;
    else if(E >=31)
        return 0x80000000u;


    res = frac | 0x800000;
    if(E> 22)
        res <<= (E -23);
    else res >>= (23-E);

    if (sign)
        res = ~res + 1;

    return res;

}
```

(2) ./btest 실행

```
[io0818@programming2 ~]$ ./btest
Score   Rating  Errors  Function
 2      2       0       negate
 3      3       0       isLess
 2      2       0       float_abs
 4      4       0       float_twice
 4      4       0       float_i2f
 4      4       0       float_f2i
Total points: 19/19
```

IV. 결론 및 고찰

-bit단위의 operator들만을 이용해 코드를 구성하는 기초를 배울 수 있었다.

-float 단위의 bitwise 연산을 다루는 것에 대해 실습을 통해 익숙해졌다.

-리눅스 xshell을 이용해 서버를 이용하는 방법을 익힐 수 있었다.

-exp 표현 방법이 ((1<<8) + ~0)<<23) & uf, 혹은 (uf>>23)&0xff처럼 기준이 달라서 달라지는 코드도 있는 것처럼 여러 expression을 나만의 방식으로 구현하는 것에 익숙해지고 싶다.