2019-2학기

컴퓨터 SW시스템 개론

Lab #2

20180085 송수민

#1 Previous Week

I. Byte Ordering

-> 메모리에 다수의 바이트를 저장하는 순서. 2가지 방법이 있음

i) Big Endian -> Least significant byte has highest address

ii) Little Endian -> Least significant byte has lowest address

II. Floating Point

2진수로 소수 부분까지 어느정도는 표현이 가능함. 하지만 2진수는 2의 거듭제곱 꼴이므로, 표현의 한계가 있다. 예를 들어, 1/3을 표현하기는 어렵다.

IEEE Floating Point는 1985년에 제작되었으며, 이는 소수점 연산의 기준이다.

$$(-1)^s M 2^E$$

위와 같은 수식으로 표현한다.

이 표현 체계에는 크게 3개의 범위가 있다. NaN(Not a Number), Normalized 와 Denormalized가 있다.

IEEE 표현에는 Rounding이라고 일종의 반올림이 존재한다. 일정 조건을 만족하면, 수 하나를 올려버린다.

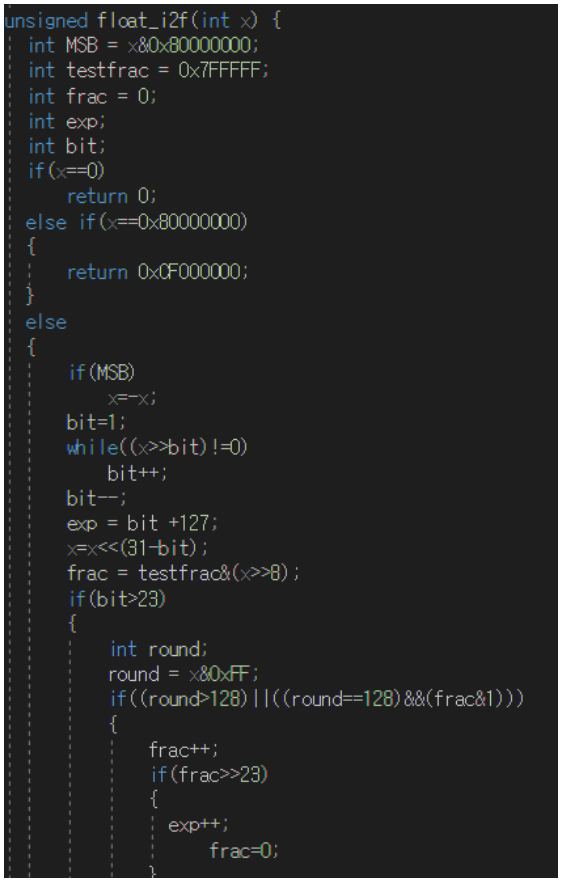
#2 Source Code

```
int isLessOrEqual(int x, int y) {
        int MSBx = (x >> 31) & 1;
        int MSBy = (y >> 31) & 1;
        int cmp = ~x + 1 + y;
        int MSBcmp = (cmp >> 31) & 1;
        return (MSBx & !MSBy) | (!(MSBx^MSBy) & !MSBcmp);
```

위 함수를 구현 할 때, 크게 주의해야할 부분은 부호가 다른 수의 비교이다. 위 코드는 x<=y에서 x를 이항하여 y-x>=0을 사용하는데, y가 양수고 x가 음수이면 Overflow가 발생 할 수 있기 때문이다. 그래서 먼저 두 수의 MSB값을 뽑아내고, 후에 위 부등식을 구현한다. MSB를 비교하고, 나중에 cmp값의 MSB를 통해 두 수의 크기를 비교한다.

```
unsigned float_neg(unsigned uf) {
        int testexp = 0x7F800000 & uf;
        int testfraction = 0x007FFFFF & uf;
        if ((testexp == 0x7F800000) && testfraction)
                return uf;
        return 0x80000000^uf;
}
```

위 함수를 구현할 때에는 NaN을 주의하면 된다. NaN은 exp가 1111...111이고 fraction부분이 0이 아니면 되기 때문에, 그 부분을 예외처리하고 나머지 경우에는 uf의 MSB위치에 있는 값을 처리하여 Return 한다.



```
unsigned float_i2f(int x) {
  int MSB = x&0x80000000;
  int testfrac = 0x7FFFFF;
  int frac = 0;
  int exp;
  int bit;
  if(x==0)
      return 0;
  else if(x==0x80000000)
  {
      return 0xCF000000;
  }
  else
  {
      if(MSB)
          x=-x;
      bit=1;
      while((x>>bit)!=0)
          bit++;
      bit--;
      exp = bit +127;
      x=x<<(31-bit);
      frac = testfrac&(x>>8);
      if(bit>23)
      {
          int round;
          round = x&0xFF;
          if((round>128)||((round==128)&&(frac&1)))
          {
              frac++;
              if(frac>>23)
              {
                  exp++;
                  frac=0;
              }
```

위 함수를 구현하기 위해서는 sign, exp, fraction부분을 생성하여 이를 합쳐서 return 하는 것이 기본적인 구조이다. 하지만 0,-0에 대해서는 미리 예외처리를 하고, 나머지 부분에 대해 sign값을 먼저 뽑아내고, bit단위로 하나씩 옮겨가면서 exp를 결정하고, 나머지 부분에서 fraction을 결정하여 return한다.

```
unsigned float_twice(unsigned uf) {
        int exp = 0x7F800000&uf;
        int MSB = 0x80000000 & uf;
        if (exp == 0x7F800000)//NaN
                return uf;
        else if (exp == 0)
                return MSB | (uf << 1);
        else
                return uf + 0x00800000;
}
```

위 함수를 구현할 때에는 3가지 범위로 나누어서 생각한다. NaN, Normalized와 denormalized이다. 각 경우에 맞게 처리한다.