ICS Homework 2

Cao Shengcao

2.88

Format A		Format B	
Bits	Value	Bits	Value
1 01110 001	$-\frac{9}{16}$	1 0110 0010	$-\frac{9}{16}$
0 10110 101	208	0 1110 1010	208
1 00111 110	$-\frac{7}{1024}$	1 0000 0111	$-\frac{7}{1024}$
0 00000 101	$\frac{5}{2^{17}} = \frac{5}{131072}$	0 0000 0001	$\frac{1}{1024}$
1 11011 000	$-2^{12} = -4096$	1 1110 1111	-248
0 11000 100	768	0 1111 0000	$+\infty$

Be careful about "rounding toward $+\infty$".

This requirement is quite ambiguous. If we replace "rounding toward $+\infty$" with "rounding to even", we'll get a different result. "0 00000 101 ($\frac{5}{131072}$)" would be "0 0000 0000 (0)", and "1 11011 000 (-4096)" would be "1 1111 0000 ($-\infty$)".

2.92

```
float_bits float_negate(float_bits f)
{
    float_bits s = f & 0x80000000;
    float_bits e = f & 0x7F800000;
    float_bits r = f & 0x007FFFFF;
    if (e == 0x7F800000 && r)
        return f;
    return f ^ 0x80000000;
}
```

The idea is not difficult, when `exp` bits are `1111 1111` and `frac` bits are not all zeros, the float is a `NaN`.

I tested this function and compared with the following float-point operations `float_negate_test` on my machine, and the result is somewhat different.

```
float_bits float_negate_test(float_bits f)
{
    float x = -*(float*)&f;
    return *(float_bits*)&x;
}
```

These two functions treat `NaN` differently. For example, when `f` is `7f800001`, the former function returns `7f800001` as the exercise requires. The latter returns `ff800001`, which simply reverses the sign bit.

2.96

```
int float_f2i(float_bits f)
{
    int s = f >> 31;
    int e = ((f >> 23) & 0xFF) - 127;
    int r = (f & 0x7FFFFF) | (1 << 23);
    int i;
    if (e > 30) return 0x80000000;
    else if (e > 23) i = r << (e - 23);
    else if (e > -1) i = r >> (23 - e);
    else i = 0;
    if (s) i = -i;
    return i;
}
```

First we need to extract each part of the float ($(-1)^S \times M \times 2^E$), and then especially, test the exponent E. When $E > 30$, the float is too large or ∞ or even "not a number", and the result would be a special integer. When $E < 0$, the float is too small, and the result would be 0. Otherwise, we can simply perform bit shift on M.