Section 4 Worksheet: Bit Manipulation

Andrew Benson

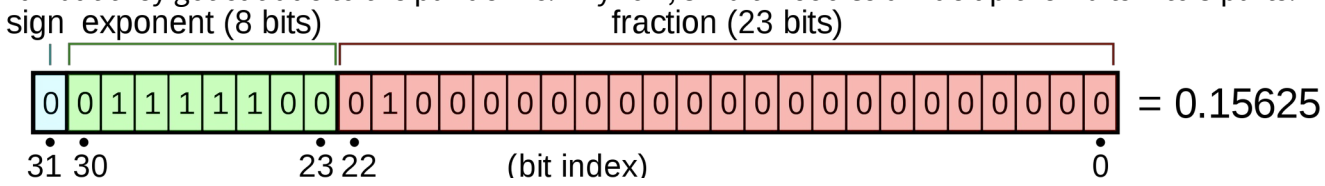
Bitwise Operator Practice

Write out, **in decimal**, the result of the following expressions. If possible, try to do these in your head, visualizing their binary representations!

```
3 & 4                   ~1

3 | 4                   ~(-1)

3 ^ 4                   3 << 2                  0xe >> 2

1 << 63                 (1 << 31) >> 31 (← don't actually do this in your code!)
```

Bitwise Coding Practice

In C, the type of fractional numbers is called a `float`. They're usually part of the curriculum of CS 107 but they got cut due to the pandemic. Anyhow, 32-bit `float`s divide up their bits into 3 parts:



https://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/Float_example.svg/2880px-Float_example.svg.png

For simplicity, let's pretend we're given values of type `int` that use this same structure.

For the following exercises, `ssh` into the Myth machines, `cd` into your cs107a folder, and run `git clone /afs/ir/class/cs107a/WWW/git/section4` to get the starter code.

1) Write a C function with the following prototype that takes in an `int` and returns the 8 bits of the exponent as a `char`. For example, when run on an `int` with the same bit pattern as the image above, `get_exponent` should return `0x7c (0b01111100)`.

```
char get_exponent(int f);
```

2) [optional] Write a C function with the following prototype that takes in an int and prints out the exponent's bits. For example, when run on an `int` with the same bit pattern as the image above, `print_exponent` should print `01111100`.

```
void print_exponent(int f);
```

More Optional Practice

Write out, **in hexadecimal**, the result of the following expressions. Assume integers are signed.

```
1) 0xdead & 0xbeef

2) 0xdead | 0xbeef

3) 0xdead ^ 0xbeef

4) 0xfaceb00c >> 4

5) 0xfaceb00c >> 1

6) 0xfaceb00c << 1
```