Binary

Contents

Numbers

Unsigned positive integers are represented in binary use place values of powers of two.

128	64	32	16	8	4	2	1	
0	0	1	1	0	1	0	1	$= 53_{10}$

Sign and magnitude

Negative integers can be represented by using the first bit as a sign bit and using the other bits as their normal values. For instance a `u8` becomes an `i7` with a leading sign bit. The sign bit represents the sign only and has no numerical value. It is very hard to do arithmetic on sign and magnitude numbers.

$+/-$	64	32	16	8	4	2	1	
0	0	1	1	0	1	0	1	$= 53_{10}$

$+/-$	64	32	16	8	4	2	1	
1	0	1	1	0	1	0	1	$= -53_{10}$

Two's complement

In two's complement, the place value of the leading bit is the negative of what it would be in an unsigned integer.

-128	64	32	16	8	4	2	1	
0	0	1	1	0	1	0	0	$= 52_{10}$

-128	64	32	16	8	4	2	1	
1	1	0	0	1	1	0	0	$= -52_{10}$

Take the 1 furthest to the right. Keep it and everything to the right of it. Flip all the bits to the left.

Fixed point reals

Fixed point decimals have a fixed decimal point in a pre-determined position. The placement of the point is an trade-off between magnitude and precision.

-32	16	8	4	2	1	·	$^1/_2$	$^1/_4$	
0	0	1	1	0	0	·	1	0	$= 12.5$

-32	16	8	4	2	1	·	$^1/_2$	$^1/_4$	
1	1	0	0	1	1	·	1	0	$= 12.5$

Floating point reals

Floating point numbers are encoded as $m \cdot 2^n$ where m is the *mantissa* and n is the *exponent*. This means that fewer bits are needed to store the number and greater magnitude and precision are possible.

A floating point number is represented as a mantissa section, and an exponent section. Both are written in two's complement.

Converting floating point to decimal

Write down the mantissa. If the sign bit is negative, then apply the two's complement flip the mantissa and note that it's negative. Then find the exponent (respecting two's complement) and shift the point right by that amount.

Here's an example:

Consider the given binary floating point number with an 8-bit mantissa and a 4-bit exponent:
1.0101011 0101

We will first apply the two's complement flip to the mantissa and convert the exponent to decimal:
-0.1010101 5

Now we can shift the point 5 places to the right:
-010101.01

Now we have a fixed point representation with a separate negative sign and we get -21.25 as our answer.

Here are some more examples:

0.1100000 1110
0.1100000 -2
0.0011000
0.125 + 0.0625 = 0.1875

1.1100000 1111
-0.0100000 -1
-0.0010000
0.125

Normalisation

Providing the most precision by making the first digit after the point significant.

In normalised form, the first digit of the mantissa after the sign bit is the opposite of the sign bit. Use the exponent to adjust the position of the point in the mantissa to achieve this. If you move the point to the right to normalise, then you need to *subtract* that number from the exponent and vice versa.

0.0011010 0100
0.1101000 0010

To normalise a negative number, move the point to the right until it is between a 1 followed by a 0 and subtract that many from the exponent. Leading 1s in a negative number have the same effect as leading 0s in a positive number.

1.1100011 0011
111.00011 0001
1.0001100 0001

Converting decimal to floating point

As an example, lets convert 88 to floating point.

First convert to fixed point binary: 01011000.0

Move the point left until it is between a 0 followed by a 1 and set the exponent to the number of places you moved:
0.1011000 0111

Now for 17.25:
010001.01
0.1000101 0101

Now for -17.75:
- 010001.11 0000
Apply two's complement and we get 101110.01 0000
1.0111001 0101

Arithmetic

To add two floating point numbers, convert them both to fixed point and add like normal. If you want to subtract, then make one of them negative first, using two's complement.