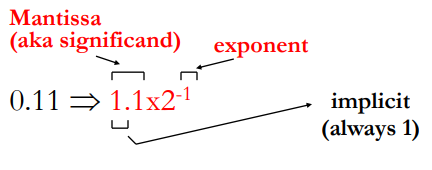
Software Engineering Lecture 3

Real Numbers – Floating point

Binary representation:

* Example 0.75 in base 10 => 0.11 in base 2 (2-1+2-2 = 0.5 + 0.25 = 0.75)
* This isn’t great for calculation in an actual circuit so we use normalization

Normalization:

We normalize for 3 reasons:

* It simplifies machine representation
* It simplifies comparisons (which is bigger 0.000101 or 0.0001? 1.01x2-4 vs 1x2-4)
* It’s more compact for very small/ large numbers ( 0.0000000000000001 = 1x2-16)

Floating Point Conversion

Convert the number 25 to floating point with normalization:

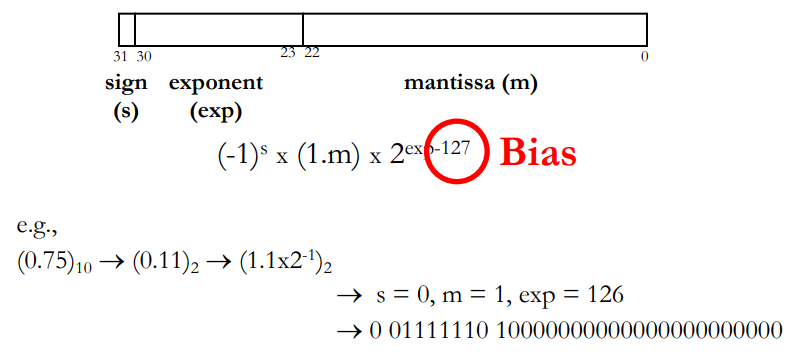
1. 25 in base 10 => 11001 in base 2
2. 11001 to normalized floating point => 1.1001x24

Here the number is normalised with 1.1001 as the mantissa and 4 is the exponent.

The IEEE 754 Floating Point Standard

This is a standard for representing and computing with fixed-width floats.

For 32 bit representation the first bit is the sign, the next 8 bits are the exponent and the remaining 22bits are dedicated to the mantissa:



Note that this representation does not use 2’s complement.

Why bias?

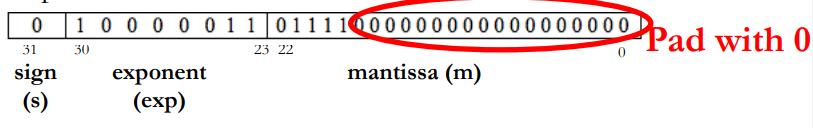
* It avoid the complexity of +/1 exponents
* It simplifies the relative ordering of floating point numbers.

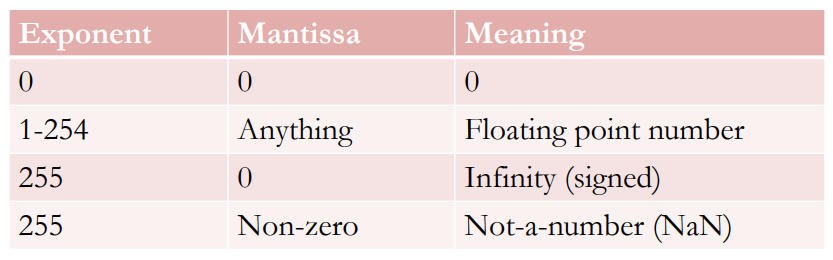
Note that processors usually have specialized floating point units to perform floating point arithmetic.

Example 2

Convert 23.5 to floating point

1. 23 in base 10 => 10111 in base 2
2. 0.5 in base 10 => 0.1 in base 2
3. 23.5 in base 10 => 10111.1 in base 2
4. Normalize = 1.01111x24
5. S = 0, m = 01111 (remember the 1. Is implicit), exp = 4+127 = 131 in base 1- => 10000011 in base 2





Representing Characters

Characters need to be encoded in binary too. Operations on characters have much simpler requirements than on numbers so the encoding choice is not crucial. The most common representation is ASCII where each character is held in a byte. Java uses Unicode which chan encode characters from many (all?) languages, this uses 16 bits per character.

Representing Strings

Words and sentences etc. Are just strings of characters. No common standard exists for identifying the end of a string in memory, different languages use different encodings. In C a special character encoded as 0x00 (called the NULL character) is used. In Java the length of the string is kept with the string itself a string is an object and length is one of its member variables).