Floating point representation

A scheme for representing a number very small to very large. It is widely used in the scientific world. Consider, the floating point number

	Exponent E	Mantissa or Significand F
+/-	x x x x	y y y y y y y y y y y y

In decimal it means (+/-) **D**. yyyyyyyyyyyy x $10^{\mathbf{xxxx}}$ (D>0)

In binary, it means (+/-) 1. yyyyyyyyyyyy x $2^{\mathbf{xxxx}}$

(The 1 is implied)

IEEE 754 representation

s	xxxxxxxx	yyyyyyyyyyyyyyyyyyyyyyy
1	8	23 bits

Single precision

Largest = 1.111… x 2^{+127} □ 2 x $10^{\mathbf{+38}}$

Smallest = 1.000 … x 2^{-128} □ 1 x $10^{\mathbf{-38}}$

Sign = $(-1)^{S}$

These can be positive and negative, depending on s.

IEEE 754 double precision (64 bits)

S	exponent	significand
1	11 bits	52 bits

Largest = $1.111\ldots \times 2^{+1023}$

Smallest = $1.000\ldots \mathrm{X}\ 2^{-1024}$

Overflow and underflow in FP

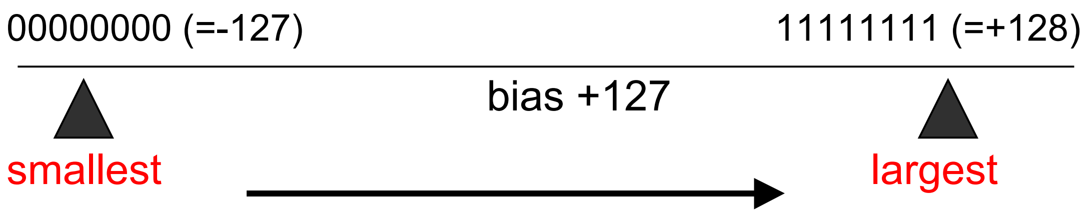
An overflow occurs when the number if too large to fit in the frame. An underflow occurs when the number is too small to fit in the given frame.

Biased Representation

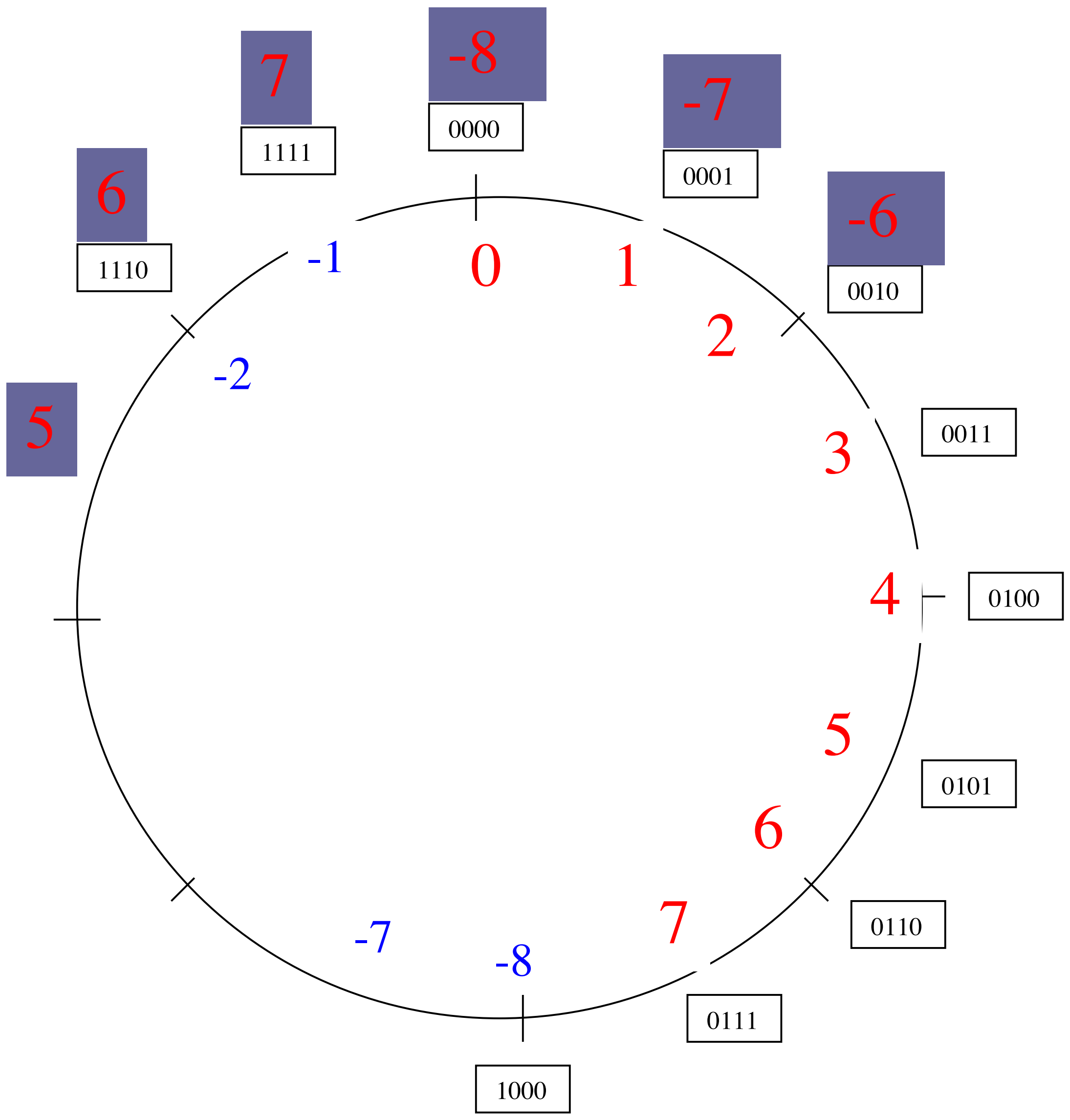
$$\left.\begin{array}{ll}\text{Exponent} = 11111111 & 2^{-1} \\ \text{Exponent} = 00000000 & 2^{0}\end{array}\right\} \quad \text{awkward for sorting}$$

However, to facilitate sorting, IEEE 754 treats 00…0 as the most negative, and 1,11..1 as the most positive exponent.

This amounts to using a bias of 127.



So, value = $(1)^{s} \times (1+\text{significand}) \times 2^{(\text{exponent} - \text{bias})}$



-8
0000
-7
0001
-6
0010
7
1111
6
1110
5
0
1
2
3
4
5
6
7
-1
-2
-7
-8
0011
0100
0101
0110
0111
1000

Floating Point Addition

1. Align (Shift the smaller number to the right until the exponents are equal)
2. Add the significand
3. Normalize
4. **If** underflow or overflow **then**

   round-off to the right number of bits

   else flag exception

Note that these stages can be pipelined.

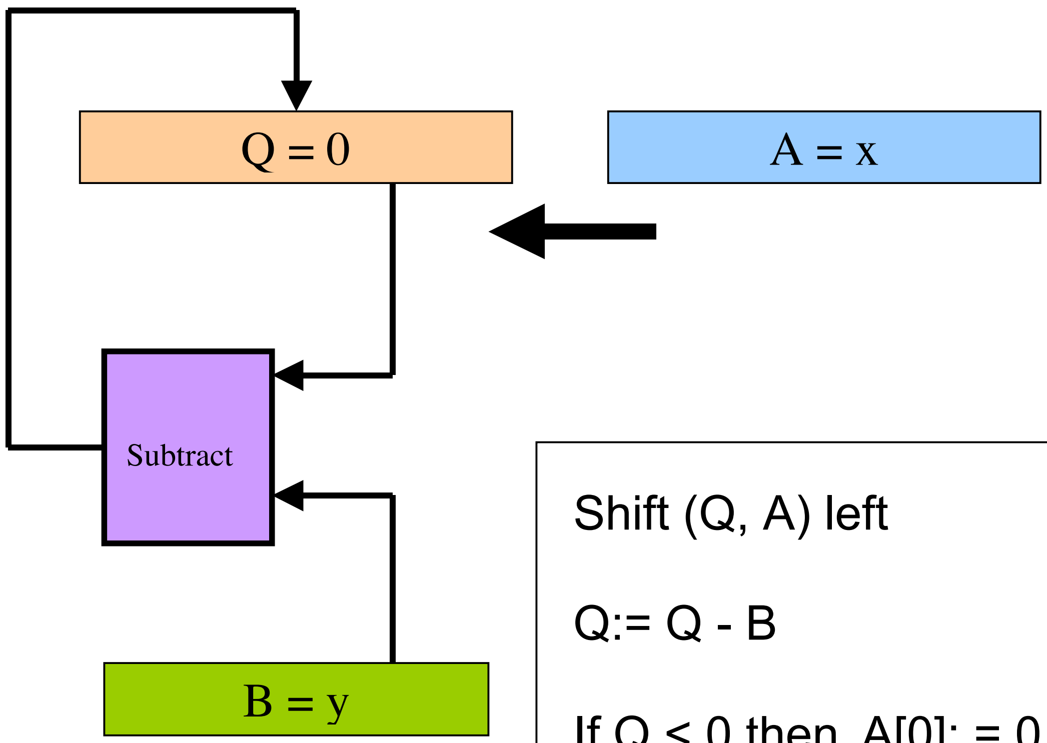
Floating Point Multiplication

1. Add the (biased) exponents and subtract the bias to get the new exponent
2. Multiply the significands
3. Normalize (if necessary)
4. If overflow or underflow then exception else round off the significand
5. Set the sign appropriately

These steps can be pipelined, if necessary

Restoring division algorithm for integer operands

Divide x by y: quotient = q, remainder = r



```
Shift (Q, A) left

Q:= Q - B

If Q < 0 then  A[0]: = 0

                  Q:= Q+B

            else A[0] := 1
```

After N cycles, A = quotient, Q = remainder