Chapter 2

Floating Point Numbers

Computer Application

Floating Point Numbers

- Computer representation inspired by scientific notation
  - Examples
    - 3.15576×10^9
    - 1.001101×2^4
- Floating point representation encodes
  - Sign
  - Exponent
  - Significand

IEEE-754 representation

Single Precision - 32 bits

s	exponent	significand
1 bit	8 bits	23 bits

Double Precision - 64 bits

s	exponent	significand
1 bit	11 bits	52 bits

IEEE-754 fields

- **Sign** bit: 1 if -ve
- **Exponent**: The value of exponent is 8 bit two's compliment offseted by 127 (-126 to +127 for single precision). Special exponent values 0 and 255 used to code for 0, Nan and Infinity.
- **Significand**: Leading bit is implicit - significand field contains the bits *after* the binary point.

Examples

- -0.75 = -0.11 = -1.1×2^{-1}
- IEEE-754 single precision representation
  - 10111111010000000000000000000000
- 1.00 = 1.0×2^0
  - 00111111100000000000000000000000
- -52.0 = -110100 = -1.101×2^5
  - 11000010010100000000000000000000

Elaboration (float)

Exponent	Significand	*Object represented*
0	0	0
1-254	anything	floating-point number
255	0	infinity
255	nonzero	Nan
0	nonzero	denormalized number

Why Offset

- To keep the order in integer form.
- If (x > y) <=> (int)x > (int) y.
- Comparing integer is faster and simpler.
- Example:

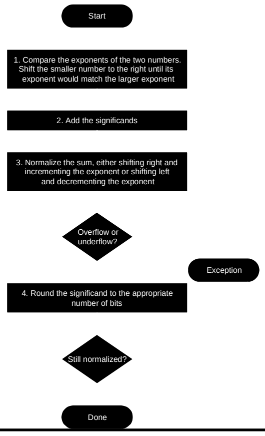
With Offset	2.0 =	01000000000000000000000000000000
	0.5 =	00111111000000000000000000000000
No Offset	2.0 =	00000000100000000000000000000000
	0.5 =	01111111100000000000000000000000

C Examples

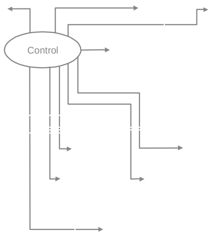
```
#include <conio.h>
#include <stdio.h>
union dami {
            float f;
            long  l;
            } tt[4];
void main()
 {
 FILE *fp;
 fp= fopen ("xxx.txt", "w");
 clrscr();
 tt[0].f = 1.0;
 tt[1].f = -1.0;
 tt[2].f = 0.0;
 tt[3].f = 1.25;
 fprintf(fp,"%08lx\n", tt[0].l);
 fprintf(fp,"%08lx\n", tt[1].l);
 fprintf(fp,"%08lx\n", tt[2].l);
 fprintf(fp,"%08lx\n", tt[3].l);
 }
```

Result:
3f800000
bf800000
00000000
3fa00000

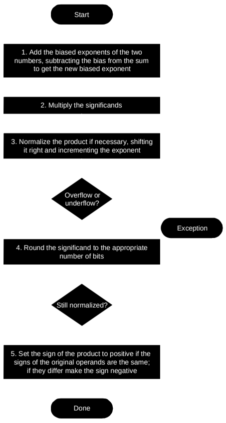
Floating Point Addition



Floating Point Adder



Floating Point Multiplication



IEEE-754 Rounding

- Two extra bits, guard and round, are maintained in intermediate results to do rounding properly.
- Rounding options
  - Truncation
  - Round up
  - Round Down
  - Round to nearest

Finite Precision Arithmetic

- IEEE-754 only captures a finite number of members of the infinite set of reals
- Floating point operations produce approximate, not exact results this can have profound consequences when you want to perform a long sequence of arithmetic operations. Egs weather prediction, nuclear weapons simulations.