

## Representing fractions – Fixed point

- The problem:
  - How to represent fractions with finite number of bits ?

## Representing fractions – Fixed point

A number with 10 bits  $a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$

## Representing fractions – Fixed point

A number with 10 bits  $a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$

$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$

Fixing the point

## Representing fractions – Fixed point

Range of representation:

	Number	Fraction
Signed (2 complement)	$-2^7 \dots 2^7 - 1$	Quanta of 0.25
Unsigned	$0 \dots 2^8 - 1$	Quanta of 0.25

## Fixed point : the problem

- Cannot represent wide ranges of numbers.
  - In scientific applications.

## Representing Fractions – Floating point

$10 \longrightarrow 1 * 10^1$

$-0.123 \longrightarrow -1.23 * 10^{-2}$

Base (radix) - r

### Representing Fractions – Floating point

10 →  $1 * 10^1$

-0.123 →  $-1.23 * 10^{-1}$

exponent

### Representing Fractions – Floating point

10 →  $1 * 10^1$

-0.123 →  $-1.23 * 10^{-1}$

Number (Mantissa)

### Representing Fractions – Floating point

10 →  $(-1)^0 * 1 * 10^1$

-0.123 →  $(-1)^1 * 1.23 * 10^{-1}$

Sign bit

### Problem of uniqueness

0.1 →  $100 * 10^{-4}$

0.1 →  $0.001 * 10^2$

Representation is not Unique

### Problem of uniqueness - Normalization

0.61 →  $610 * 10^{-4}$

0.61 →  $0.0061 * 10^2$

Standardization →  $6.1 * 10^{-1}$

One digit to the Left of the point

### Normalized Binary Floating point

$$D = (-1)^{a_0} * (1.a_1a_2a_3\dots) * 2^{b_1b_2b_3\dots}$$

String of bits

### Floating point - Questions

- Representing the (signed) exponent
- How to represent zero?
  - And Nan, infinity ?
- How to add, subtract and multiply?
- Rounding Errors.

### Floating point – Representing the exponent

How to represent signed number ?

### Floating point – Representing the exponent

How to represent signed number ?

### Floating point – Representing the exponent

- We want the exponent to be binary ordered:

0000 < 0001 < .... < 1000 < ... < 1111

### Floating point – Representing the exponent

Number = **Number** - B

Usually B =  $2^{n-1}-1$

We define the following sizes like this:

$e_{min}$	000...0001
$e_{max}$	111...1110

### Floating point – Representing zero, NAN, $\pm\infty$

IEEE754 special values

Exponent	Mantissa	Represent
$e = e_{min}-1$	M = 0	$\pm 0$
$e = e_{min}-1$	M $\neq$ 0	$0.M \cdot 2^{e_{min}}$ ← Denormalized number
$e_{min} \leq e \leq e_{max}$		$1.M \cdot 2^e$ ← normalized number
$e = e_{max}+1$	M = 0	$\pm\infty$
$e = e_{max}+1$	M $\neq$ 0	NaN

## IEEE 754

Parameter	Single	Double
Mantissa	24	53
$e_{\max}$	127	1023
$e_{\min}$	-126	-1022
Exponent width	8	11
Format width	32	64

(Including the sign Bit)

## What is NaN (not a number)

Operation	NaN produced by
+	$\infty + (-\infty)$
*	$0 * \infty$
/	$0/0, \infty/\infty$

Operation
$X \pm \text{NaN}$
$X * \text{NaN}$
$X / \text{NaN}$

Partial list

## Infinity

- Provide a safe way to continue calculation when overflow is encountered.

## Calculations with Floating Point numbers

- Addition:
  - Equalize the exponents (smaller  $\rightarrow$  larger exponent)
  - Sum the mantissa
  - Renormalize if necessary

## Calculations with Floating Point numbers

- Example (in base 10):

$|E| = 1, |M| = 3$

$$91 \rightarrow 9.10 * 10^1$$

$$9.7 \rightarrow 9.70 * 10^0$$

## Calculations with Floating Point numbers

$$9.10 * 10^1 + 9.70 * 10^0$$

Not The same Order.

## Calculations with Floating Point numbers

$$\begin{array}{r}
 9.10 \cdot 10^1 \\
 + 9.70 \cdot 10^0 \\
 \hline
 9.10 \cdot 10^1 \\
 + 0.97 \cdot 10^1 \\
 \hline
 10.7 \cdot 10^1 \\
 \downarrow \text{renormalize} \\
 1.07 \cdot 10^2
 \end{array}$$

## Calculations with Floating Point numbers

- Example II (in base 10):  
|E| = 1, |M| = 3

$$91 \rightarrow 9.10 \cdot 10^1$$

$$9.75 \rightarrow 9.75 \cdot 10^0$$

## Calculations with Floating Point numbers

$$\begin{array}{r}
 9.10 \cdot 10^1 \\
 + 9.75 \cdot 10^0
 \end{array}$$

Not The same Order.

## Calculations with Floating Point numbers

$$\begin{array}{r}
 9.10 \cdot 10^1 \\
 + 9.75 \cdot 10^0 \\
 \hline
 9.10 \cdot 10^1 \\
 + 0.975 \cdot 10^1 \\
 \hline
 10.75 \cdot 10^1 \\
 \downarrow \text{renormalize} \\
 1.07 \cdot 10^2
 \end{array}$$

5 (rounding error)

## Rounding Errors

### The Problem:

Squeezing infinite many real numbers into a finite number of bits

## Measuring Rounding Errors

- Units in last place (Ulp)
- Relative Error

### Measuring Rounding Errors – ULP

If  $\overbrace{d.d\text{ddd}}^{p \text{ digits}} * r^e$  represent  $z$

$$\text{error} = |d.d\text{ddd} - (z/r^e)| * r^{p-1}$$

### Measuring Rounding Errors – ULP

Example I:  $r = 10, p = 3$

The number  $3.14 * 10^{-2}$  represents 0.0314159

$$\text{Error} = 0.159$$

### Measuring Rounding Errors – ULP

- What is the maximum ULP if the rounding is toward the nearest number?

$$0.5 \text{ ULP}$$

### Measuring Rounding Errors – Relative Error

If  $\overbrace{d.d\text{ddd}}^{p \text{ digits}} * r^e$  represent  $z$

$$\text{Relative error} = |d.d\text{ddd} * r^e - z| / z$$

### Measuring Rounding Errors – Relative errors

Example I:  $r = 10, p = 3$

The number  $3.14 * 10^{-2}$  represents 0.0314159

$$\text{Relative Error} \sim 0.0005$$