MI-ARI

(Computer arithmetics)

winter semester 2017/18

FP. **Floating point**

© **Alois Pluháček Pavel Kubalík**, 2017
Department of digital design
Faculty of Information technology
Czech Technical University in Prague

https://edux.fit.cvut.cz/courses/MIE-ARI/lectures

FP. **Floating point**

- **Definition**
- **Structure of the format**
- **Basic operations**
- **Normalized form**
- **Overflow and underflow**
- **Rounding**
  - **unsigned numbers**
  - **to the nearest value**
  - **signed numbers**
- **Representation of intermediate result (bits G, R and S)**
- **Hidden one**
- **IEEE Std 754-2008**

Floating point — definition

floating point

$$\boxed{A \sim (M, E) \implies A = M \cdot z^{E}}$$

M — **significand or mantissa**
radix point usually **between 1. and 2. digits of significand,**
or **between 1. and 2. digits of its magnitude**

E — **charakteristic or exponent**
always integer (Way ?)

z — **báse of used system**
$z = 2$ or $z = 2^{i}$ or $z = 10$

significand and characteristic
— **some codes** (sign a magnitude, 2's complement, ...)
— **both codes can be the same or different**

important: sign $A =$ **sign** M

floating point $\sim$ so called **semi-logarithmic form of** number
for example $1{,}23 \cdot 10^4$

Structure of format

significand and characteristic
— **some codes (**sign a magnitude, 2's complement, . . . **)**
— **both codes can be the same or different**

format:

characteristic: always integer $\Rightarrow$ **unit** $\varepsilon = 1$

significant: $\mathcal{Z} = 2$ — **„common“ format**
$\mathcal{Z} = 4$ — **format by IEEE Std 754-2008**
$\vdots$
$\varepsilon = 1$ — **„interesting“ format**
(used for decimal representation
by IEEE Std 754-2008)

Basic operations

$\odot$ ☞ **operation** (addition, subtraction, etc)

$\mathcal{F}(X)$ ☞ **representation of number X in floating point**

$$\boxed{A \odot B \to C}$$

$$\Downarrow$$

$$\mathcal{F}(A) \to (M_A, E_A)$$
$$\mathcal{F}(B) \to (M_B, E_B)$$
$$\downarrow$$
$$(M_A, E_A) \odot (M_B, E_B) \to (M_C, E_C)$$
$$\downarrow$$
$$(M_C, E_C) \to \mathcal{F}(C)$$

multiplication: $M_A \cdot z^{E_A} \cdot M_B \cdot z^{E_B} = M_A \cdot M_B \cdot z^{E_A+E_B}$

$$\boxed{(M_A,\, E_A) \cdot (M_B,\, E_B) \rightarrow (M_A \cdot M_B,\; E_A + E_B)}$$

division: $(M_A \cdot z^{E_A})\ /\ (M_B \cdot z^{E_B}) = M_A / M_B \cdot z^{E_A-E_B}$

$$\boxed{(M_A,\, E_A) / (M_B,\, E_B) \rightarrow (M_A / M_B,\; E_A - E_B)}$$

addition and subtraction: $(M_A \cdot z^E) \pm (M_B \cdot z^E) = M_A \pm M_B \cdot z^E$

$$\boxed{\begin{array}{c} (M_A,\, E_A) \pm (M_B,\, E_B) \rightarrow \\ \rightarrow (M'_A,\, E) \pm (M'_B,\, E) \rightarrow \\ \rightarrow (M'_A \pm M'_B,\, E) \end{array}}$$

Normalized form

! **The algorithms of operation are to be arrange so that no accuracy loss occurs if it is not impossible!!** !

simplification of algorithm ➠ normalized form

normalizoved form — **left shift of significand is impossible without overflow**

☛ **It will be supposed!!!**

The result of all operation must be normalized !

normalized form of zero:

significand = 0
characteristic —– as small as possible

normalized form ☛ **algorithm simplification**, example:

addition:

$$
\begin{array}{lll}
E_A \geq E_B & \Rightarrow & E \;\;\, = E_A \\
& & M'_A = M_A \\
& & M'_B = M_B \triangleright (E_A - E_B) \\
E_A < E_B & \Rightarrow & E \;\;\, = E_B \\
& & M'_A = M_A \triangleright (E_B - E_A) \\
& & M'_B = M_B
\end{array}
$$

subtraction: analogically

multiplication:

$$M_A \in \langle 1, 2), \; M_B \in \langle 1, 2) \;\Rightarrow M_A \cdot M_B \in \langle 1, 4)$$

division:

$$M_A \in \langle 1, 2), \; M_B \in \langle 1, 2) \;\Rightarrow M_A / M_B \in \langle \tfrac{1}{2}, 2)$$

comparison:

$$E_A > E_B \quad \Longrightarrow \quad |A| > |B|$$

Overflow and underflow

overflow of partial operation:

- **In some cases it is possible prevent the overflow of partial operation either with shift of significant or with extension of the format.**
- **Some times it is possible to eliminate overflow of partial operation — properly corrected.**

- **It is necessary to compensate processed correction by relevant characteristic.**

result is not possible to compensate or properly corrected (also the result is not possible write to the format):

- **overflow** — **characteristic is too big**
  — **magnitude of the result is too big**
- **underflow** — **characteristic is negative with too big magnitude**
  — **result is near to zero (however not zero)**

Rounding (unsigned numbers)

rounding down:

$A \rightarrow \lfloor A \rfloor_{-m}$ ... **cut off m positions behind of point**

$\lfloor A \rfloor_{-m} \leq A < \lfloor A \rfloor_{-m} + \varepsilon,$

where $\lfloor A \rfloor_{-m}$ is integer multiple ε (unit of format)

e.g.: $\varepsilon = 0{,}001$

$$3{,}141\,925 \doteq 3{,}141 = \lfloor 3{,}141\,925 \rfloor_{-3}$$
$$3{,}141\,025 \doteq 3{,}141 = \lfloor 3{,}141\,025 \rfloor_{-3}$$

rounding up:

$A \rightarrow \lceil A \rceil_{-m}$

$\lceil A - \varepsilon \rceil_{-m} < A \leq \lceil A \rceil_{-m},$

where $\lceil A \rceil_{-m}$ is integer multiple ε (unit of format)

e.g..: $\varepsilon = 0{,}001$

$$3{,}141\,925 \doteq 3{,}142 = \lceil 3{,}141\,925 \rceil_{-3}$$
$$3{,}141\,025 \doteq 3{,}142 = \lceil 3{,}141\,025 \rceil_{-3}$$

Rounding to the nearest value

rounding to the nearest value:

$$A \to \begin{cases} \lfloor A \rfloor_{-m}, & \text{if } A \leq \lfloor A \rfloor_{-m} + \varepsilon/2 \\ \lceil A \rceil_{-m}, & \text{if } A \geq \lfloor A \rfloor_{-m} + \varepsilon/2 \end{cases}$$

ex.: $3{,}141\,925 \doteq 3{,}142$
$3{,}141\,025 \doteq 3{,}141$

In generally *rounding*
(without specification of „down“ or „up“)
is obviously understood as an *rounding to the nearest value*.

What, happen if $A = \lfloor A \rfloor_{-m} + \varepsilon/2$ **?**

e.g.: $3{,}142\,500 \doteq \begin{cases} 3{,}142 \\ 3{,}143 \end{cases}$ **?**

rounding to prefer greater value:

$$A \rightarrow \begin{cases} \lfloor A \rfloor_{-m}, & \text{if } A \leq \lfloor A \rfloor_{-m} + \varepsilon/2 \\ \lceil A \rceil_{-m}, & \text{if } A > \lfloor A \rfloor_{-m} + \varepsilon/2 \end{cases}$$

then: $\boxed{A \rightarrow \lfloor A + \varepsilon/2 \rfloor_{-m}}$

e.g..: $3{,}141\,500 \doteq 3{,}142 = \lfloor 3,141\,500 + 0,000\,500 \rfloor_{-3}$

$3{,}142\,500 \doteq 3{,}143 = \lfloor 3,142\,500 + 0,000\,500 \rfloor_{-3}$

rounding to prefer even last digit:

$$A \rightarrow \begin{cases} \lfloor A \rfloor_{-m}, & \text{je-li } A < \lfloor A \rfloor_{-m} + \varepsilon/2 \\ \lfloor A \rfloor_{-m}, & \text{je-li } A = \lfloor A \rfloor_{-m} + \varepsilon/2 \text{ (*)} \\ \lceil A \rceil_{-m}, & \text{je-li } A = \lfloor A \rfloor_{-m} + \varepsilon/2 \text{ (*)} \\ \lceil A \rceil_{-m}, & \text{je-li } A > \lfloor A \rfloor_{-m} + \varepsilon/2 \end{cases}$$

(*) . . . the last digit after rounding must be even

e.g.: $3{,}141\,500 \doteq 3{,}142$

$3{,}142\,500 \doteq 3{,}142$

Rounding (signed numbers)

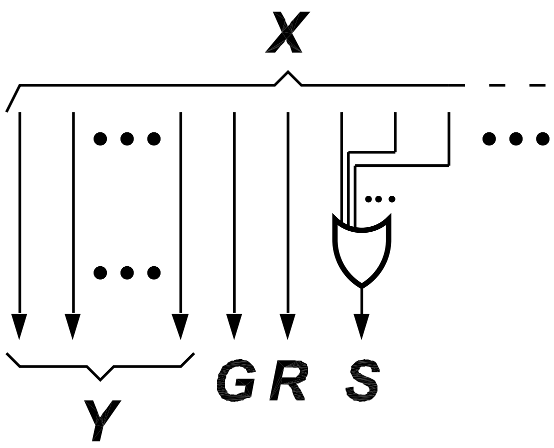
application, combination and modification of previous procedures:

rounding (to the nearest value)	☞	$\begin{cases}\text{to prefer greater value} \\ \text{to prefer even last digit}\end{cases}$

rounding:

- **to** $-\infty$
- **to** $+\infty$
- **to zero**
- **away from zero**

Representation of intermediate result (bits G, R and S)



X . . . **accurate result** (∞ positions)

Y . . . **later used part of result**
using next 3 bits is recomended:

G **[Guard]**
R **[Round]**
S **[Sticky]**

The bits can be utilized G, R **and** S **for**

- **next operation** or
- **result editing** (normalization and rounding).

Hidden one

assumptions:

- **Normalized forms of number are used.**
- **Significand M is represented by sign and magnitude code.**

consequences:

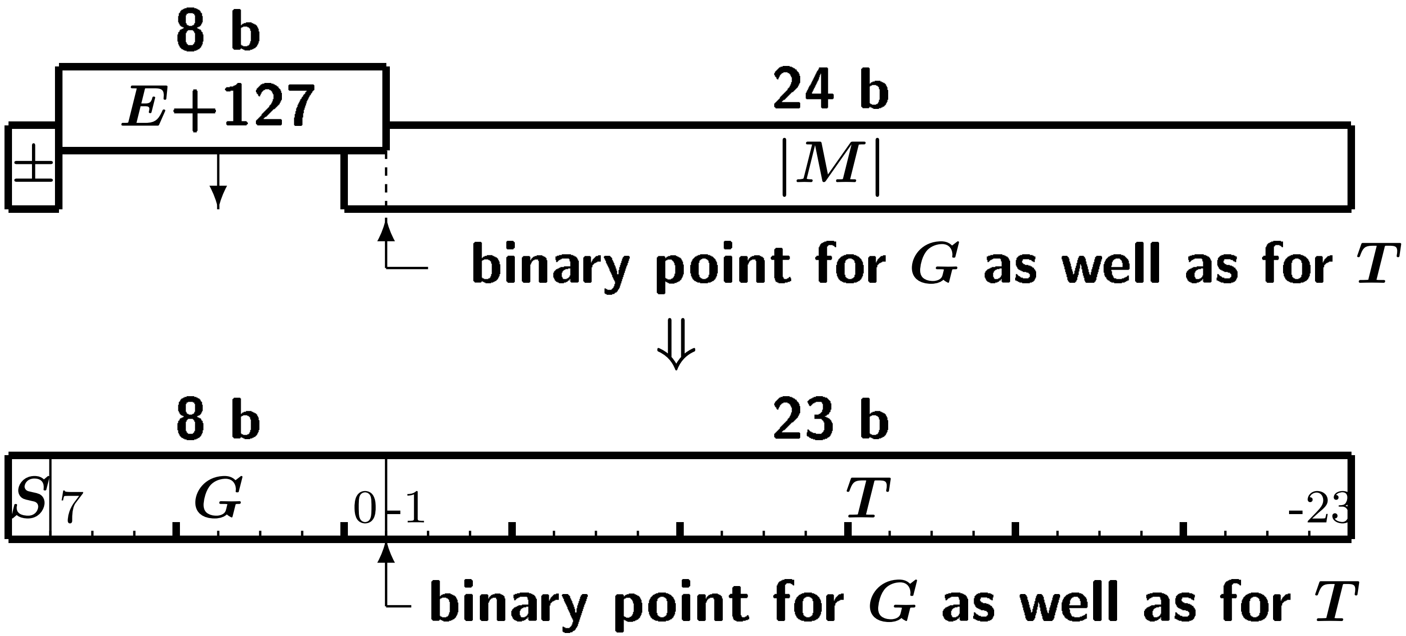
- $\forall M \neq 0$ **MSB**$(|M|) = \mathbf{1}$
- **The bit which is always equal to one can be deleted.**

☞ ***hidden one***

problem: $M = 0$
solution: **using another representation for zero**
— see (FP – 16)

IEEE Std 754-2008

principle of hidden one — example: binary format 32 bits



Example: $-5{,}5_{10} \sim ?$

$5{,}5_{10} = 101{,}1_2 = (1{,}011 \lhd 10)_2$

$T = 0{,}011\,000 \ldots 000_2 \qquad G = (127 + 2)_{10} = 1000\,0001_2$

image of $-5{,}5_{10}$**:** $1\ 1000\,0001\ 011\,000 \ldots 000_2 =$

$= 1100\,0000\,1011\,0000\ \ldots\ 0000_2 =$

$= \mathrm{C0B0\,0000}_{16}$

binary formats

analogues to format for 32 bits

by default: significant — sign and magnitude
characteristic — biased code of type 1

g ☞ **number of bits in part** G

t ☞ **number of bits in part** T

$K = 2^{\,g-1} - 1$

basic format: 32 bits, 64 bits, 128 bits

another formats: 16 bits, k **bits, where** $k = 32 \cdot i \geq 128$**, and other**

format	g	t	accuracy	$K = E_{max}$
16 bits	5	10	11 b	15
32 bits	8	23	24 b	127
64 bits	11	52	53 b	1 023
128 bits	15	112	113 b	16 383

A ☞ **represented number**
$\mathcal{F}(A)$ ☞ **image of number A**

zero and other „specifications“

	A
$\mathsf{G} = 0\ldots0_2$	$(-1)^{S} \cdot T \cdot 2^{-K+1}$
$\mathsf{G} = 1\ldots1_2$ a $T = 0$	$(-1)^{S} \cdot \infty$
$\mathsf{G} = 1\ldots1_2$ a $T \neq 0$	**NaN**
else (viz **FP – 14**)	$(-1)^{S} \cdot (1+T) \cdot 2^{G-K}$

$$\mathcal{F}(A) = \begin{cases} 000\ldots000 \\ 100\ldots000 \end{cases} \quad \Rightarrow \quad A = \pm 0$$

$\pm\infty$ ☞ **overflow — „limit“** (e.g. $5/0 = +\infty$)

NaN [Not a Number] ☞ **unknown result**
(even if the ∞) is used

(e.g. $0/0 =$ NaN)

decimal formats — $A = M \cdot 10^{E}$

- Characteristic E used binary format.
- Significant M is whole number and can be represented as
  - binary or
  - decimal (with some type of compression).
- Normalization is not required.
- Format is divided into parts S, G and T (it is similar to binary formats), however conversion of the parts G and T to numbers E and $|M|$ is very difficult.

basic formats: 64 bits, 128 bits

another format: 32 bits, k bits, where $k = 32 \cdot i \geq 128$,

and other

format	g	t	accuracy	E_{max}
32 bits	11	20	24 b / 7 decimal digits	96
64 bits	13	50	54 b / 16 decimal digits	384
128 bits	17	110	114 b / 34 decimal digits	6 144

