FPA (Floating Point Arithmetics)
(aritmetika pohyblivej rádovej čiarky)

Zobrazenie racionálnych čísel – principiálne stačí pracovať s **pevnou rádovou čiarkou (Fixed Point Arithmetic)** (v n-tici bitov stačí zvoliť polohu desatinnej čiarky)

$3.625_{10} = \ 11.101 \qquad 0011.1010$

$6.5_{10} \ = 110.1 \qquad 0110.1000$

Napríklad súčet je možné formálne realizovať ako pri celých číslach

$$\begin{array}{rl} & 00111010 \\ + & \underline{01101000} \\ & 10100010 \end{array} \qquad 1010.0010_2 = 10.0125_{10}$$

Nevýhody pevnej rádovej čiarky (malý rozsah zobraziteľných čísel) (napr. pre „súčasné“ zobrazenie hmotnosti slnka a atómu by sme potrebovali 201 bitov (26 bytov)).

Pohyblivá rádová čiarka

1234.56=**0.123456*10^{4}** - zápis čísla v pohyblivej rádovej čiarke (vedecká anotácia)
podobne pre binárne čísla
11001100.0011011=**0.11001100011011*2^{01000}**
Obidva zápisy sa dajú vyjadriť v tvare

$$N = M * z^{E} \quad |M| \leq 1 \quad z \in Z$$

M – mantisa (argument), E – exponent (characteristic), základ (radix)

Na zobrazenie má vplyv
- celkový počet bitov na zobrazenie
- spôsob zobrazenia mantisy
- spôsob zobrazenia exponentu
- rozdelenie bitov pre mantisu a exponent
- poradie mantisy a exponentu....

Pri voľbe počtu bitov pre zobrazenie mantisy a exponentu rozhodujú nasledovné atribúty :

-rozsah (range) hovorí, o najmenšom a najväčšom čísle v absolútnej hodnote rôzne od nuly, ktoré sa dá zobraziť

- **počet platných číslic - rozlišovacia schopnosť (precision)** – je to vlastne presnosť zobrazenia čísla napr. číslo π môže byť zobrazené ako 3.142 alebo 3.141592

-presnosť (accuracy) hovorí o blízkosti čísla k správnej hodnote napr. číslo π môže byť zobrazené ako 3.142 alebo 3.241592. Aj keď prvé číslo je zobrazené pomocou menšieho počtu platných číslic, jeho hodnota je bližšia k hodnote čísla π.

Je dôležité sa zaoberať presnosťou najmä aritmetických operácií napr.

$$A = 1432.52 \quad B = 1432.51$$

$$C = \frac{A+B}{A-B} = \frac{1432.52+1432.51}{1432.52-1432.51} = \frac{2865.03}{0.01}$$

bude hodnota výsledku C stanovená s chybou 100%.

Kódovanie FPA čísel

Kódovanie podľa IEEE 754-1985

- **normalizovaná mantisa** (maximalizácia počtu platných číslic - bitov (precision) = počet platných bitov mantisy)

$$-2 < x \leq 1 \quad alebo \quad x = 0 \quad alebo \quad 1 \leq x < 2$$

- posunutý exponent, ak je m-bitový exponent potom je exponent z rozsahu

$$-2^{m-1} \leq e \leq 2^{m-1} - 1$$

Ak k exponentu pripočítame posun 2^{m-1}

potom rozsah hodnôt mantisy bude (1.00000 ...00, 1.1111...11)

- usporiadanie bitov pre znamienko, exponent a mantisu

[1]str.188 Obr. 4.28

- reprezentácia nuly

[1]str.187 Obr. 4.27

	Single precision	Double precision	Quad precision
Field width in bits			
S = sign	1	1	1
E = exponent	8	11	15
L = leading bit	1	1	1
F = fraction	23	52	111
Total width	32	64	128
Exponent			
Maximum E	255	2047	32 767
Minimum E	0	0	0
Bias	127	1023	16 383

Notes

S = sign bit (0 for a negative number, 1 for a positive number)

L = leading bit (always 1 in a normalized, non-zero mantissa)

F = fractional part of the mantissa

The range of exponents is from Min E + 1 to Max E − 1.

The number is represented by $-1^S \times 2^{E - \text{exponent}} \times L.F.$

A signed zero is represented by the minimum exponent, $L = 0$, and $F = 0$, for all three formats.

The maximum exponent has a special function that represents signed infinity for all three formats.

Príklad prevodu do 32-bitového formátu IEEE:
Majme číslo –19.375
Najprv prevedieme celočíselnú časť, a desatinnú časť

19	1
9	1
4	0
2	0
1	1
0	

$19_{10} = 10011_2$

0.375	0
0.750	1
0.500	1
0.000	koniec

$0.375_{10} = 0.011_2$

$$-19.375_{10} = -10011.011_2 = -1.0011011x2^4$$

$$S = 1$$

$$E = 4 + 127 = 131_{10} = 10000011_2$$

$$M = 1.0011011$$

S	E	M		
1	1000001	1 0011011	00000000	00000000
C1		9B	00	00

Číslo –19.375 sa zobrazí v 32-bitovom formáte IEEE do 32-bitového slova s obsahom **C19B0000**$_{\mathrm{H}}$.

Pre 32-bitové číslo v IEEE formáte je :

Minimálna hodnota
posunutého exponentu $E_{pmin} = 1$ $E_{min} = -126$

Maximálna hodnota
posunutého exponentu $E_{pmax} = 254$ $E_{max} = 127$

Zvláštne hodnoty

$E_p = 0 \wedge M = 0$ (NULA)

$E_p = 0 \wedge M \neq 0$ ($-\infty$) – podtečenie

$E_p = 255 \wedge M = 0$ (NaN)

$E_p = 255 \wedge M \neq 0$ (∞) - pretečenie

NaN – Not a Number

Možné hodnoty 32-bitového FPA čísla IEEE formátu :

- záporné čísla $\langle -(1-2^{-24})x2^{127}, -0.5x2^{-128} \rangle$

- kladné čísla $\langle 0.5x2^{-128}, (1-2^{-24})x2^{127} \rangle$

- záporné čísla menšie ako $-(1-2^{-24})x2^{127}$
  nazývame záporné pretečenie

- kladné čísla väčšie ako $(1-2^{-24})x2^{127}$
  nazývame kladné pretečenie

- nula, ak všetky bajty sú nulové

Podtečenie nie je problém – výsledok sa vyjadrí „čistou nulou“.
Pretečenie sa signalizuje dvojakým spôsobom :
-ak je NaN potom generuje sa „výnimka“ (nezmyselný výsledok) (je to vec operačného systému)
- ak je signalizované pretečenie je na programátorovi, ako danú situáciu vyrieši

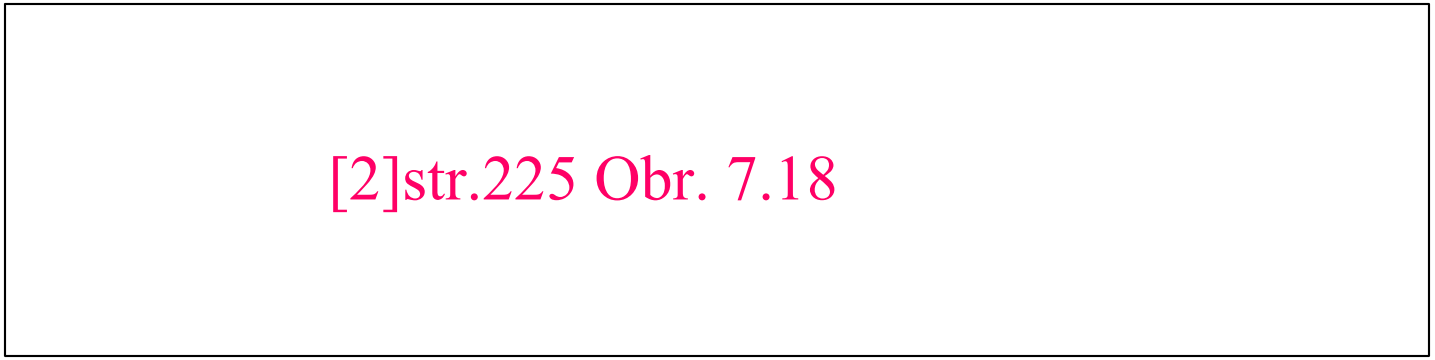
Iné formáty FPA

IBM S/370

-základ 16 (32-bit a 64-bit)

-dôraz na počet platných bitov

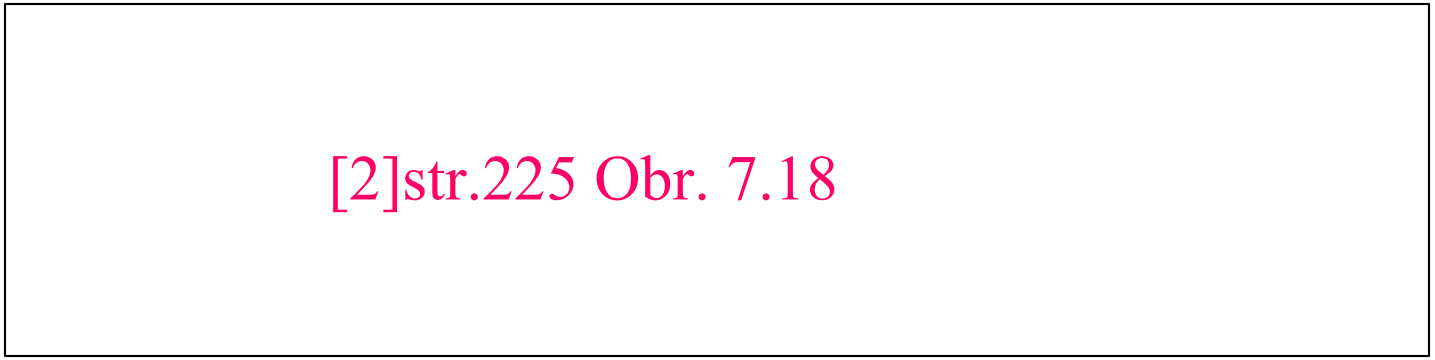
-nula = nulové bajty



Formáty FPA VAX-11

-dve varianty veľkosti exponentu

-nula = nulové bajty



FPA aritmetika

FPA aritmetika nie je asociatívna !!!!. Neplatí pre čísla x,y,z v pohyblivej rádovej čiarke

$$(x + y) + z \neq x + (y + z)$$
$$(x \cdot y) \cdot z \neq x \cdot (y \cdot z)$$

nie je ani vždy distributívna !!!!

$$x \cdot (y + z) \neq (x \cdot y) + (x \cdot z)$$

Algoritmy treba starostlivo analyzovať, či aplikácia pravidiel nemôže viesť k nezmyselnému výsledku. Napr.

$$(1.0e100 - 1.0e100) + 1.0 = 1.0 \quad ale$$
$$(1.0e100 + 1.0) - 1.0e100 = 0.0$$

Súčet a rozdiel

Princíp :
Majme čísla A=12345 a B=567.89, ktoré sa dajú zapísať v tvare $A = 0.12345x10^5 \quad B = 0.56789x10^3$

Súčet v dekadickom vyjadrení je jednoduchý

$$\begin{array}{r} 12345 \\ +\ 567.89 \\ \hline 12912.89 \end{array}$$

ale vo FPA formáte

$$\begin{array}{r} 0.12345x10^5 \\ +\ 0.56789x10^3 \end{array}$$

je potrebné normovať exponenty (zväčšiť menší exponent)

$$0.1234500x10^5$$
$$+\ 0.0056789x10^5$$
$$0.1291289x10^5$$

V prípade binárnych čísel

$$1.1001x2^4$$
$$+\ 1.0001x2^3$$

normujeme menší exponent

$$1.10010x2^4$$
$$+\ 0.10001x2^4$$
$$10.00011x2^4$$

pretože mantisa nie je normalizovaná, musíme upraviť výsledok

$$10.00011x2^4 \quad na \quad 1.000011x2^5$$

a upraviť počet platných miest zaokrúhlením

$$1.000011x2^5 \quad na \quad (1.000011+0.00001)x2^5 = 1.0001x2^5$$

Postup sčítania a odčítania môžeme formulovať nasledovne :

2. Ak je niektorý z operandov nulový, výsledok je rovný nenulovému operandu s príslušným znamienkom
3. Ak rozdiel exponentov operandov je väčší ako počet bitov matnisy výsledok je rovný operandu s väčším exponentom s príslušným znamienkom
4. Inak

$$A = M_A * z^{E_A} \qquad B = M_B * z^{E_B}$$

Potom

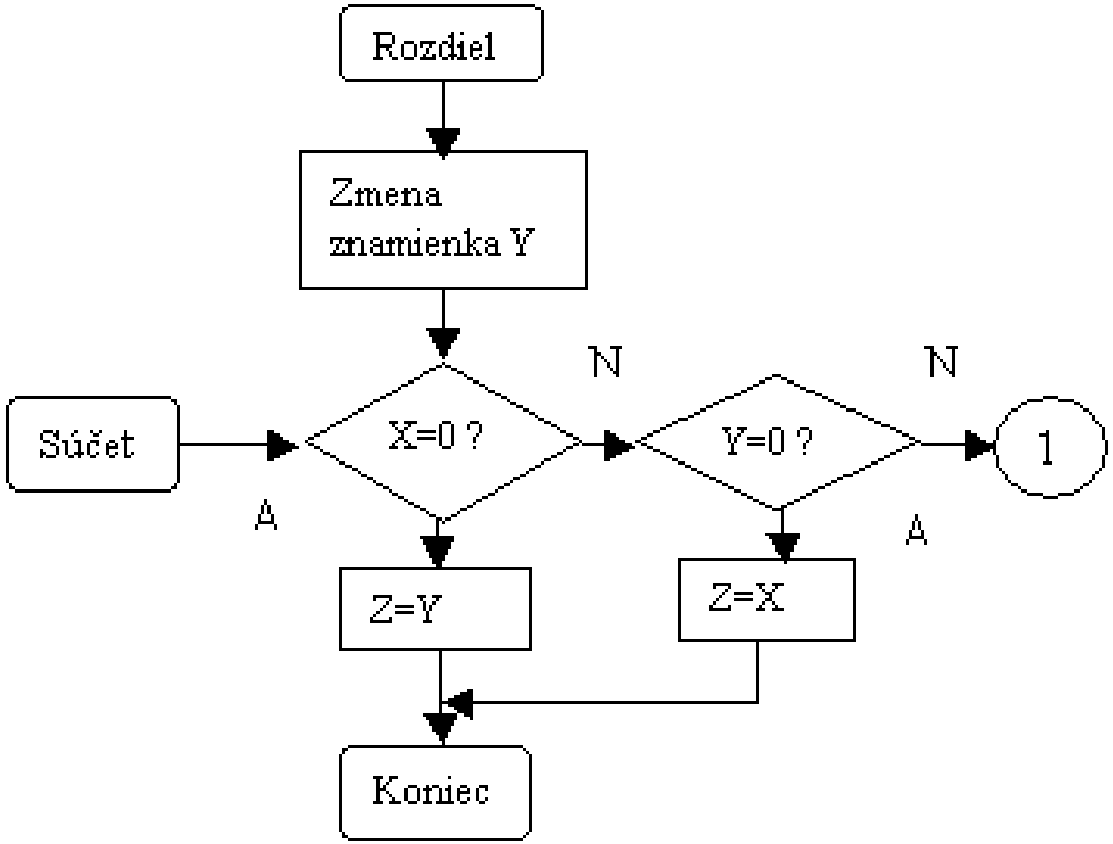
$$A + B = \left(M_A * z^{(E_A - E_B)} + M_B\right) * z^{E_B} \qquad pre \quad E_A \le E_B$$

$$A - B = \left(M_A * z^{(E_A - E_B)} - M_B\right) * z^{E_B} \qquad pre \quad E_A \le E_B$$

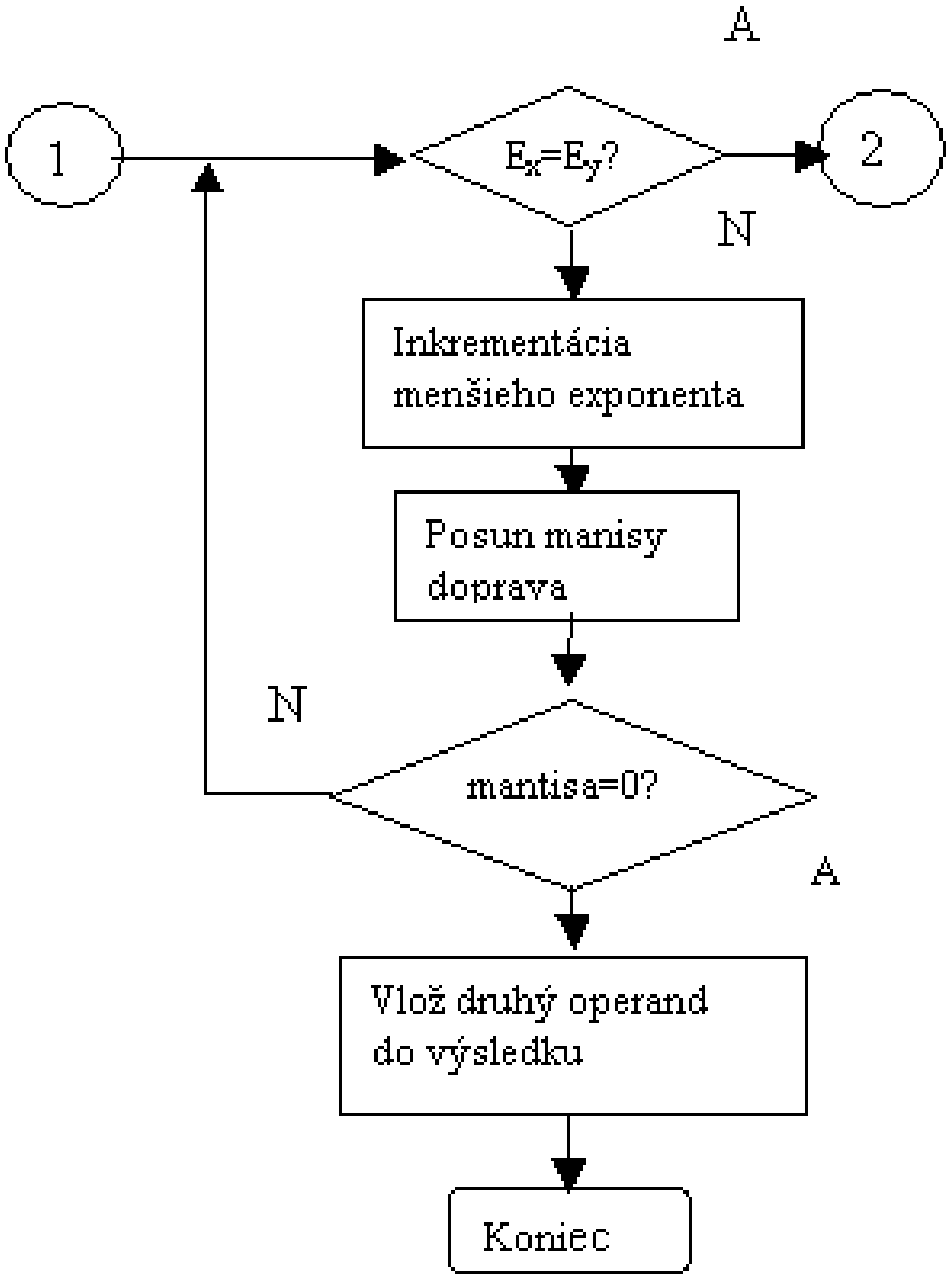
4. Vykoná sa normalizácia výsledku s príslušným zaokrúhlením
5. Testuje sa pretečenie (podtečenie) výsledku

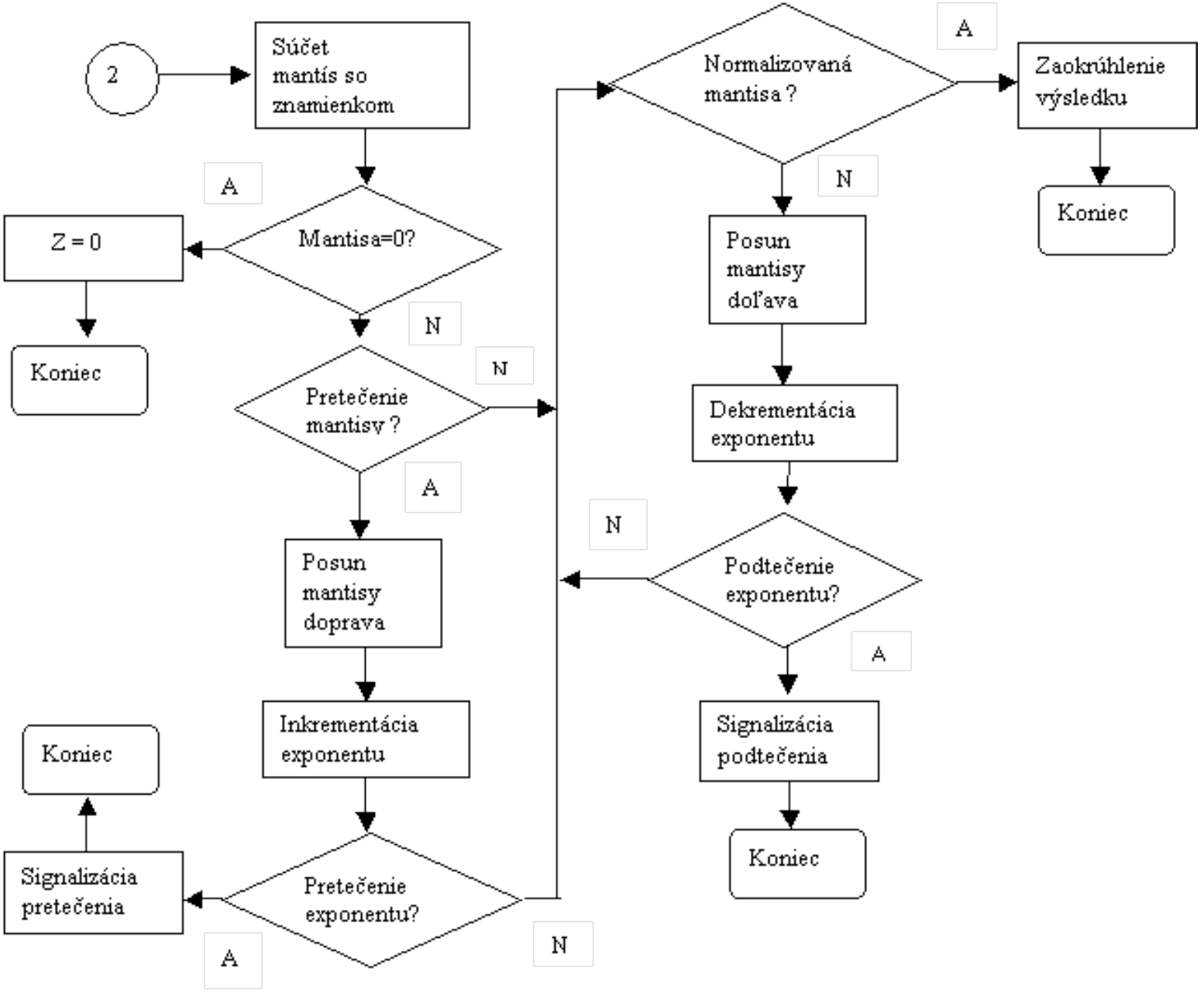
Bloková schéma súčtu/rozdielu

Vyhodnotenie operandov



Úprava mantisy operandu s menším exponentom




2
Súčet mantís so znamienkom
A
Mantisa=0?
Z = 0
Koniec
N
Pretečenie mantisy ?
N
A
Posun mantisy doprava
Inkrementácia exponentu
Pretečenie exponentu?
A
Signalizácia pretečenia
Koniec
N
N
Podtečenie exponentu?
A
Signalizácia podtečenia
Koniec
Dekrementácia exponentu
Posun mantisy doľava
N
Normalizovaná mantisa ?
A
Zaokrúhlenie výsledku
Koniec

Zaokrúhlenie a orezanie čísla

Orezanie čísla na počet platných rádov

Orezaním čísla 0.1101101 (čo je 0,8515625) na 4 platné bity dostaneme 0.1101 (0,8125) s chybou (-0.0390625). Orezaním dostaneme najbližšie číslo zdola. Lepšie je zaokrúhlenie, vykonáme pripočítaním čísla 1 s váhou o 1 menej, ako je posledný platný rád, pre tento prípad je to číslo 0.00001

$$\begin{array}{r} 0.1101101 \\ +\,0.0000100 \\ \hline 0.1110001 = 0.1110 \end{array}$$

čo je 0,875 s chybou (0.0234375).

Zaokrúhlenie
-zabezpečuje menšiu chybu výsledkov FPA operácií oproti orezaniu.
- vyžaduje dodatočné operácie s výsledkom
- vyžaduje medzivýpočty s väčším počtom platných číslic (registre ALU pre FPA obsahujú ochranné bity aby pre normalizáciu a zaokrúhlenie bolo k dispozícii dosť bitov)

Chyby spôsobené konečnou dĺžkou čísla FPA sa prejavia vo výsledkoch operácií

Napr.

```
float a=0.1, b=0.1;
for( int i=0;I<100000;i++)a*=b;
```

dáva výsledok a=9998.557, čo je chyba 0,014%

Obecne fungujú zákony štatistiky, zaokrúhľovaním sú generované chyby s premenlivým znamienkom, čo v strednej hodnote sa môže udržiavať v prijateľných hraniciach. Orezávanie generuje chyby len s jedným znamienkom.

Násobenie a delenie

$$A = M_A * z^{E_A} \qquad B = M_B * z^{E_B}$$

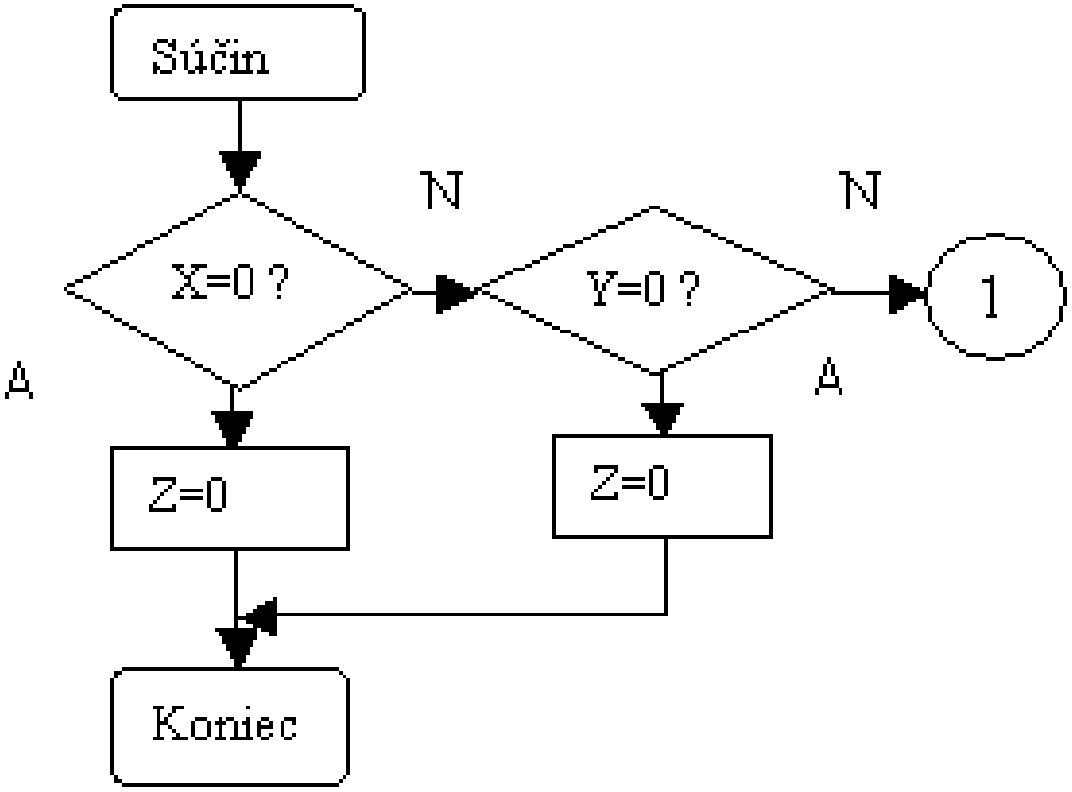
Potom

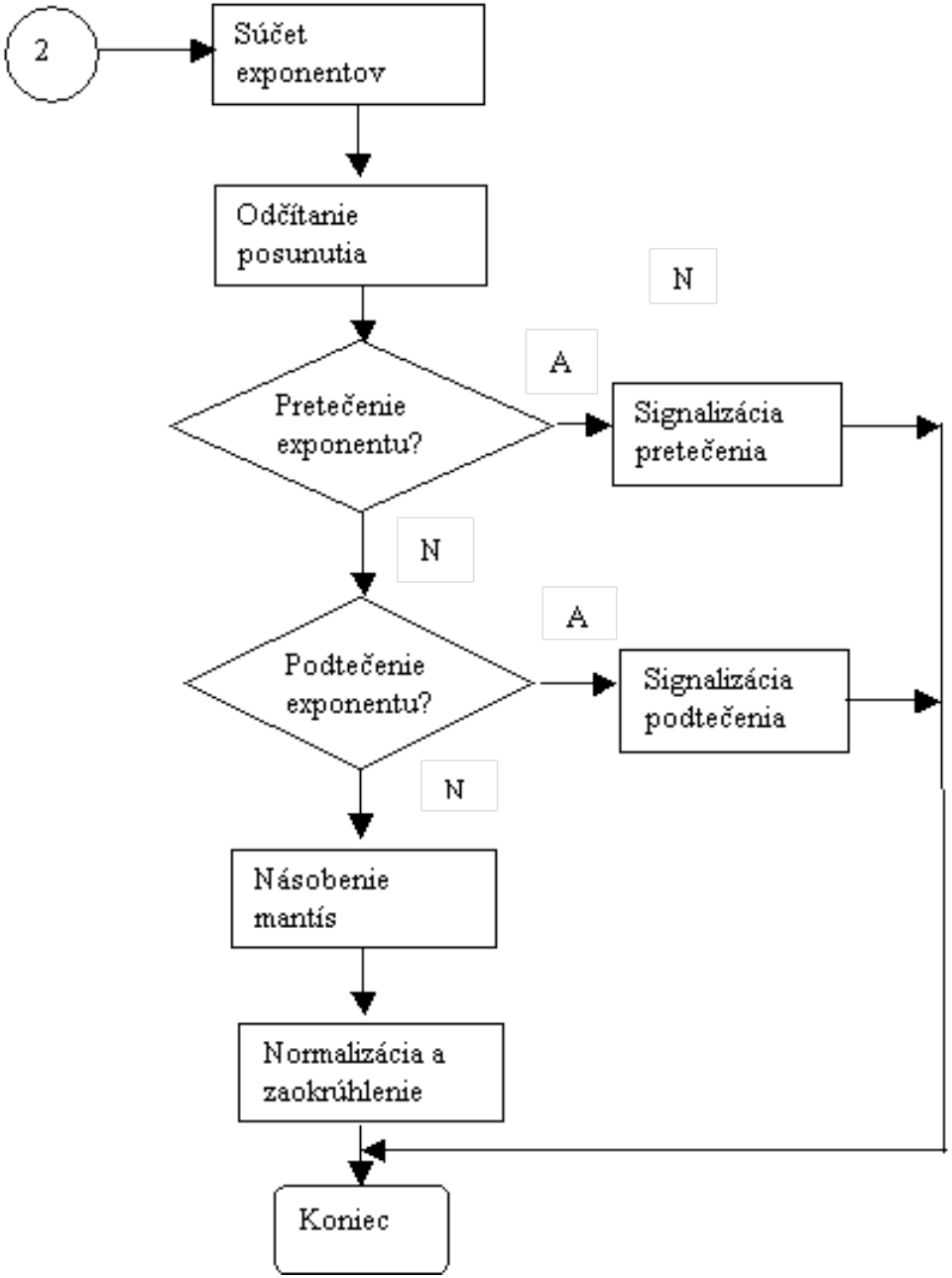
$$A * B = \left(M_A * M_B\right) * z^{\left(E_A + E_B\right)}$$

$$A / B = \left(M_A / M_B\right) * z^{\left(E_A - E_B\right)}$$

Bloková schéma násobenia X*Y

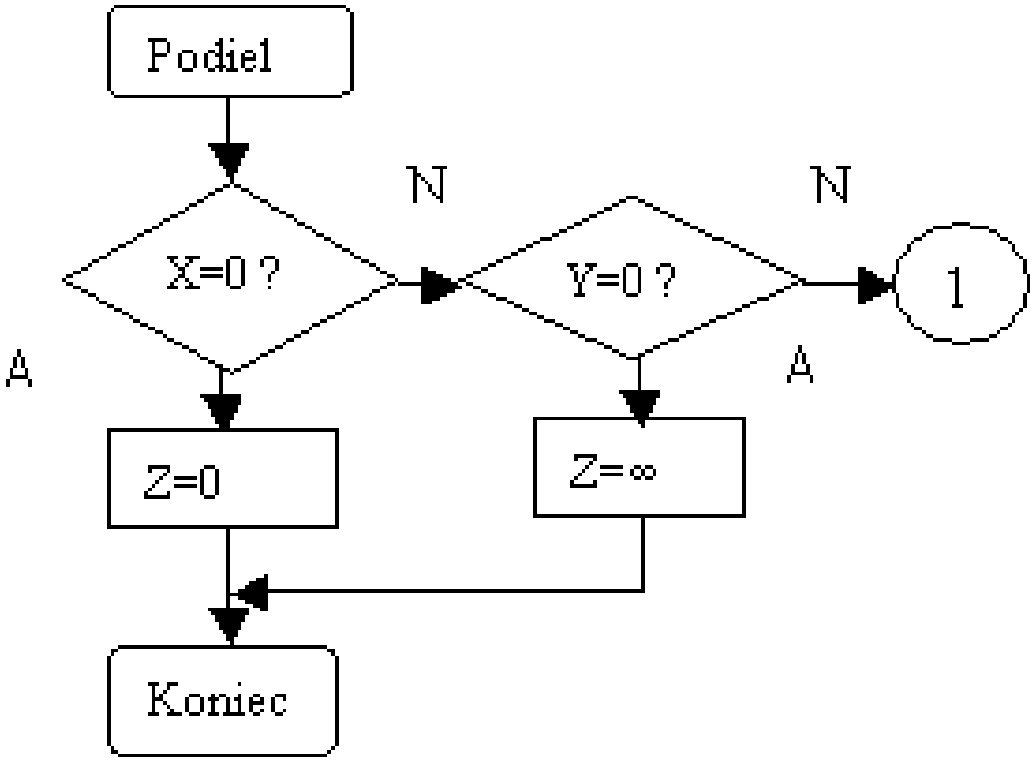
Vyhodnotenie operandov

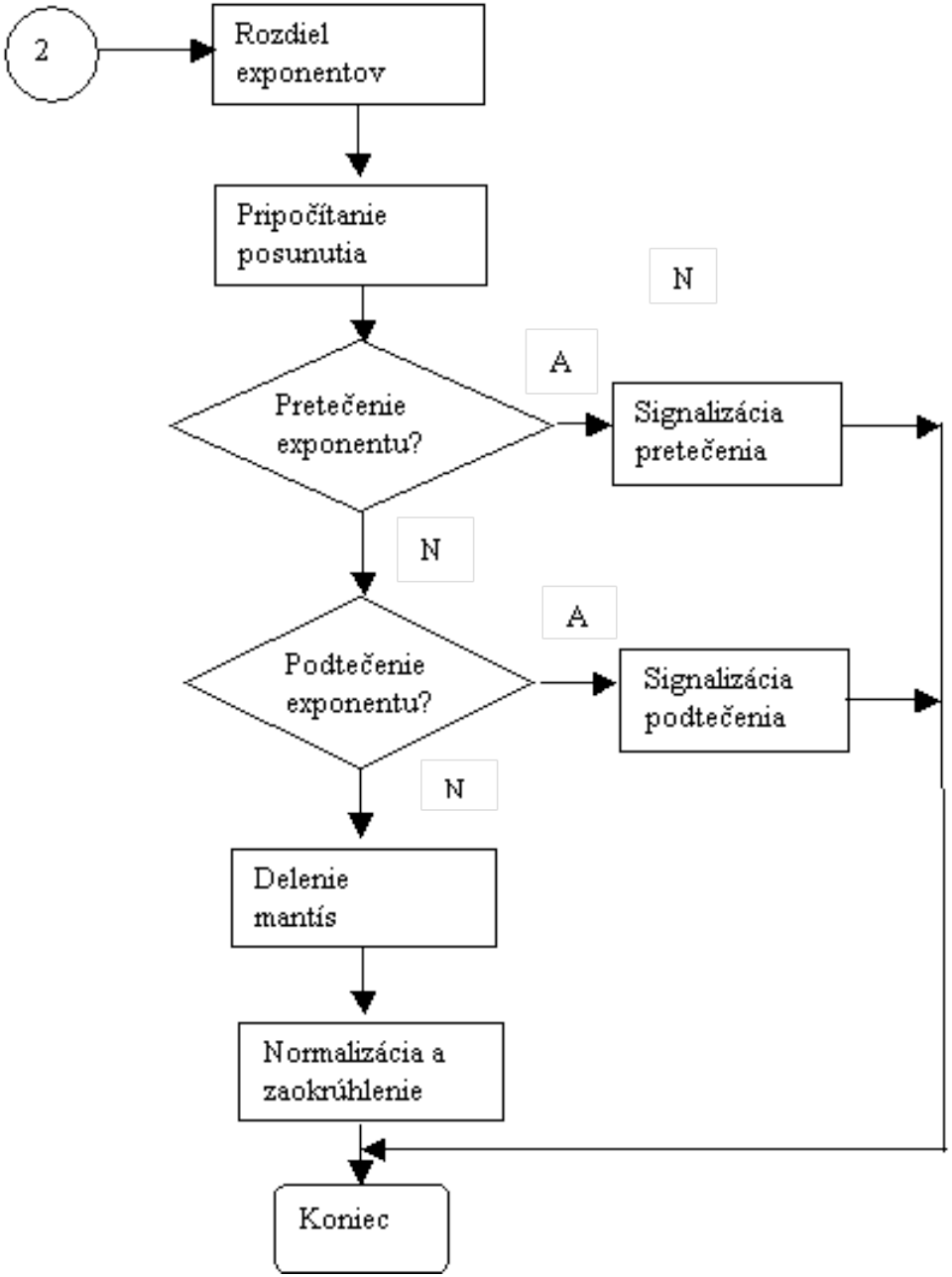



2
Súčet exponentov
Odčítanie posunutia
N
A
Pretečenie exponentu?
Signalizácia pretečenia
N
A
Podtečenie exponentu?
Signalizácia podtečenia
N
Násobenie mantís
Normalizácia a zaokrúhlenie
Koniec

Bloková schéma delenia X/Y

Vyhodnotenie operandov




2
Rozdiel exponentov
Pripočítanie posunutia
N
A
Pretečenie exponentu?
Signalizácia pretečenia
N
A
Podtečenie exponentu?
Signalizácia podtečenia
N
Delenie mantís
Normalizácia a zaokrúhlenie
Koniec

Delenie násobením

Nech $C = \frac{A}{B}$ upravíme B tak, aby

$$C = \frac{A*L}{B*L} = \frac{A^*}{B^*} \quad , tak\ aby \quad \frac{1}{2} \le B^* < 1$$

$$Definujme\ Z = 1 - B^*\ potom\ 0 < Z \le \frac{1}{2}$$

$$a\ K = 1 + Z$$

$$C = \frac{A^* * K}{B^* * K} = \frac{A^* * (1+Z)}{(1-Z) * (1+Z)} = \frac{A^* * (1+Z)}{1-Z^2}$$

$$C = \frac{A^* * (1+Z)}{1-Z^2} = \frac{A^* * (1+Z) * (1+Z^2)}{(1-Z^2)(1+Z^2)} = \frac{A^* * (1+Z) * (1+Z^2)}{1-Z^4}$$

$$\vdots$$

$$C = \frac{A^* * (1+Z) * (1+Z^2) * \cdots * (1+Z^{2^{n-1}})}{1-Z^{2^n}} \quad pre\ n \to \infty \quad Z^{2^n} \to 0$$

pre 32 platných bitov stačí n=5.

Realizácia elementárnych funkcií

Použitie tabuliek

-hodnota argumentu funkcie = adresa do tabuľky (podobný princíp ako pri rýchlych násobičkách) – vhodné pre systémy používajúce čísla s malým počtom bitov

- tabelované sú vybrané body funkcie, hodnota funkcie sa získa interpoláciou medzi dvomi hodnotami uloženými v tabuľke

Aproximácia pomocou polynómov (podielu polynómov)

Napríklad **Taylorov rad**

$$f(x) \cong \sum_{n=0}^{k} (x-a)^n \frac{f^{(n)}(a)}{n!} \quad resp. \quad f(x) \cong \sum_{n=0}^{k} x^n \frac{f^{(n)}(0)}{n!}$$

požadovaná presnosť sa dosiahne voľbou n. Napr.

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

Iteračné algoritmy

Založené na postupnom výpočte funkčnej hodnoty podľa algoritmu, ktorý zaručuje konvergenciu vypočítaných hodnôt k požadovanej hodnote. Počet krokov určuje presnosť výpočtu.

Príklad (výpočet funkčných hodnôt funkcie 1/x, tento princíp využíva aj algoritmus delenia)

Výpočet hodnoty funkcie

$$g(x) = \frac{1}{x}$$

pre konkrétnu hodnotu argumentu b to bude

$$g(b) = \frac{1}{b}$$

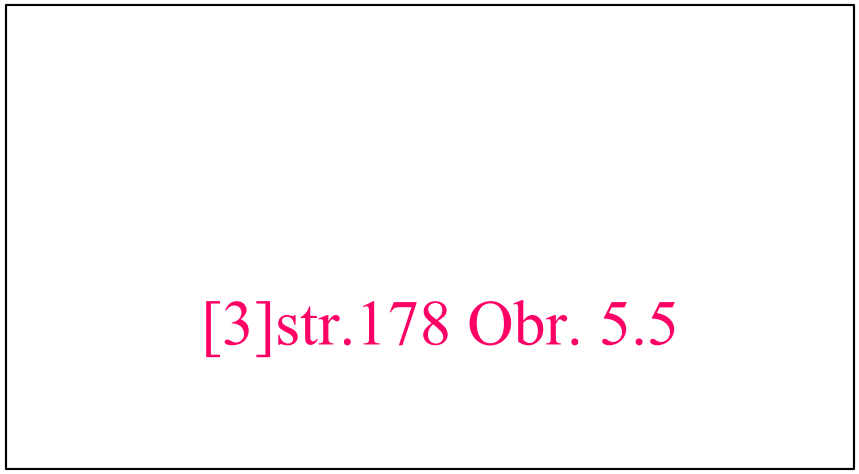
čo sa dá vyriešiť napr. delením, alebo riešením rovnice

$$f(x) = \frac{1}{x} - b = 0$$

alebo riešením úlohy nájdenia takého argumentu f(x), pre ktorý je hodnota funkcie rovná 0.
Na tento typ úloh je vhodná Newtonova iteračná metóda. Iterácia je založená na výpočte

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{\frac{1}{x_i} - b}{\frac{-1}{x_i^2}} = x_i(2 - bx_i)$$



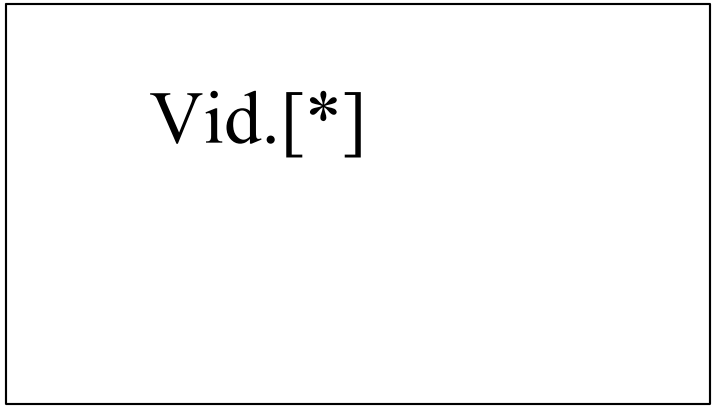
Podmienkou konvergencie je normovanie parametra b na interval <1,2), čo sa dá zabezpečiť úpravou argumentu pred výpočtom, a úpravou výsledku.

Nech b=1.5 a x_0=1, potom sú postupne medzivýsledky

1.0000000000000000
0.5000000000000000
0.6250000000000000
0.6640625000000000
0.6666564941406250
0.6666666665114462
0.6666666666666666

Algoritmy využívajúce špecifické vlastnosti funkcií

CORDIC (for **CO**ordinate **R**otation **DI**gital **C**omputer)
http://en.wikipedia.org/wiki/CORDIC - column-one [*]
je jednoduchý a efektívny algoritmus na výpočet trigonometrických a hyperbolických funkcií
Princíp je založený na postupnej rotácii jednotkového vektora v rovine



$$v_{n+1} = Rv_n$$

kde v_n , v_{n+1} sú súradnice jednotkového vektora pred a po natočení, R je rotačná matica.

$$R = \begin{pmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{pmatrix}$$

$$v_{n+1} = Rv_n = cos\gamma \begin{pmatrix} 1 & -\sigma\tan\gamma \\ \sigma\tan\gamma & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

kde γ je uhol natočenia a σ nadobúda hodnoty 1 alebo –1 (riadenie smeru natáčania).

$$v_{n+1} = Rv_n = K \begin{pmatrix} 1 & -\sigma 2^{-n} \\ \sigma 2^{-n} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x - \sigma 2^{-n} y \\ x\sigma 2^{-n} + y \end{pmatrix} K$$

volíme také uhly aby $\tan(\gamma)=2^{-n}$. Uhly sú tabelované
Problém je mierka

$$K_n = \prod_{i=0}^{n-1} \cos(\arctan(2^{-i})) = \prod_{i=0}^{n-1} \sqrt{1+2^{-2i}}$$

pre konštatný počet krokov je vždy rovnaká. Po skončení výpočtu x-vá zložka sa rovná kosínusu a y-vá zložka sínusu uhla.

Literatúra:

[1] Clements,A: The Principles of Computer Hardware, Oxford

[2] Stalling, W.: Computer Organization and Architecture, principles ...,

[3] Jelšina, M.: Architektúry počítačových systémov, ...