ARYTMETYKA ZMIENNOPRZECINKOWA

Reprezentacja stałoprzecinkowa v/s zmiennoprzecinkowa

Reprezentacja stałoprzecinkowa – odwzorowanie zapisu pozycyjnego:

$$X = \sum_{-\infty}^{\infty} x_i \beta^i$$

- jednorodna struktura kodu
- stała **bezwzględna** dokładność reprezentacji – waga pozycji najniższej
- dowolny zakres i dokładność
- łatwe rozszerzanie rozmiaru reprezentacji

Reprezentacja zmiennoprzecinkowa – odwzorowanie zapisu wykładniczego:

$$F = M\beta^E = \pm\beta^E \sum_{-m}^{p} x_i \beta^i$$

- niejednorodna struktura kodu – złożenie rekordów o różnej interpretacji
- stała **względna** dokładność reprezentacji – liczba pozycji ułamka
- ograniczone rozszerzanie zakresu – tylko w ramach standardu
- reprezentacja pseudorzeczywista (myląca nazwa "*real*") – niespełniony postulat *continuum*, reprezentowany jest podzbiór liczb rzeczywistych

Arytmetyka stałoprzecinkowa v/s zmiennoprzecinkowa

Arytmetyka stałoprzecinkowa:

- nieograniczony zakres – łatwa rozszerzalność
- dowolna dokładność bezwzględna – waga przypisana najniższej pozycji
- dowolny rozmiar reprezentacji
- możliwość dowolnej programowej interpretacji kodu

Arytmetyka zmiennoprzecinkowa:

- brak łączności dodawania w ustalonym formacie (skutek zaokrągleń)
- ograniczony zakres – ograniczone rozmiary formatu
- zmienna i ograniczona dokładność bezwzględna
- dokładność względna – zależna od formatu liczba pozycji ułamka
- nieprzydatna do dokładnych obliczeń
- wygodna w zastosowaniach, gdzie wystarczy ograniczona dokładność (np. DSP, przetwarzanie dźwięku lub obrazu, obliczenia przybliżone)

Postulaty dotyczące kodowania

$$F = M\beta^{E}$$

M – *mnożnik* (*significand*), d. mantysa (*mantissa*) – liczba z zapisie znak-moduł,
β – ustalona podstawa reprezentacji, baza (*radix*), $\beta \geq 2$,
E – *wykładnik* (*exponent*), d. cecha (*characteristic*) liczba całkowita.

$\Rightarrow$ wiele różnych reprezentacji liczby, np. $314159{,}267\ldots\cdot 10^{-5} = 0{,}0314159267\ldots\cdot 10^{2}$

postulaty

- duża dokładność $\rightarrow$ rozmiar (liczba bitów) mnożnika M
- duży zakres $\rightarrow$ rozpiętość wykładnika
- łatwe wykonanie podstawowych działań arytmetycznych i porównanie
- jednoznaczność reprezentacji $\rightarrow$ *normalizacja mnożnika*:

$$\beta^{p-1} \leq |M| < \beta^{p}$$

! normalizacja *uniemożliwia* reprezentację **zera**

- potrzebna specjalna reprezentacja zera i liczb zdenormalizowanych
- potrzebna reprezentacja $\pm\infty$ i takich obiektów jak np. ln 0, 1/0, sqrt(–1)

Normalizacja mnożnika

Dodawanie i odejmowanie

Suma lub różnica znormalizowanych mnożników może być dwa razy większa od górnej granicy przedziału normalizacji, może też być mniejsza od dolnej granicy.

Mnożenie i dzielenie

Jeśli $F_1 = M_1\beta^{E_1}, F_2 = M_2\beta^{E_2}$ oraz $\beta^{p-1} \le |M_{1,2}| < \beta^p$, to iloraz jest zawsze w przedziale $\beta^{-1} \le |M_1 / M_2| < \beta^1$. Rozsądnym wyborem jest więc p=0 lub 1. Wtedy tylko część ilorazów wymaga normalizacji, pozostałe wymagają prostego skalowania przez β. Podobnie jest w mnożeniu.

Wybór p=1 odpowiada odręcznemu zapisowi wykładniczemu, jest też intuicyjny i lepszy gdy jest narzucone ograniczenie zakresu wykładnika (w kodowaniu), bo umożliwia dokładny zapis odwrotności najmniejszej liczby znormalizowanej

Reprezentacja zera

Niezależnie od wyboru przedziału normalizacji nie jest możliwa znormalizowana reprezentacja zera. Ograniczony jest też od dołu zakres reprezentacji.

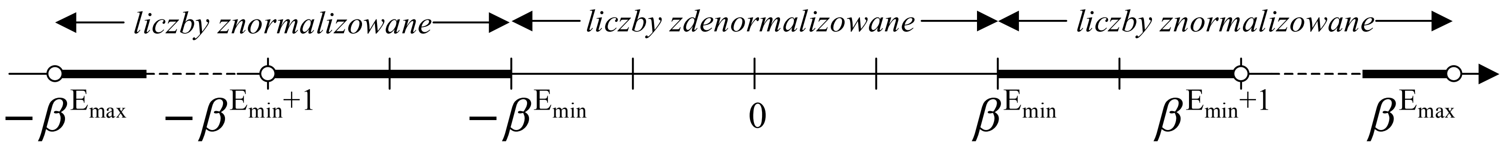
Zero i liczby bardzo małe

- *liczby znormalizowane* $(1 \le |M| < \beta)$

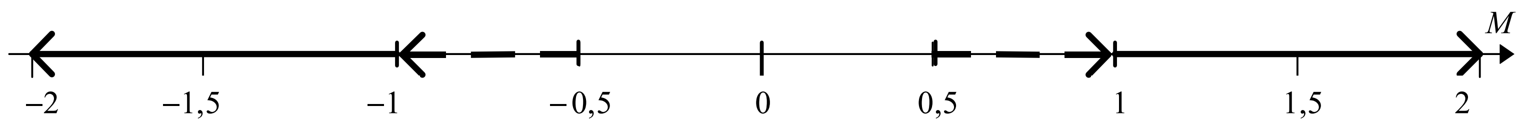
$$F = \pm(d + f)\beta^{E}, \quad 0 \le f < 1, \;\; d = 1,2,\ldots,\beta - 1$$

- niemożliwa znormalizowana reprezentacja zera
$\rightarrow$ liczby *zdenormalizowane* $|F| < \beta^{E_{\min}} \rightarrow F = M_f \beta^{E_{\min}}, \;\; |M_f| < 1$

$$F = \pm(0 + f)\beta^{E_{\min}}, \quad 0 \le f < 1$$



Liczby znormalizowane i zdenormalizowane (p=1)



Znormalizowane wartości mnożnika M przy β=2, p=0 (- - -) oraz p=1 (—)

Realizacja podstawowych działań arytmetycznych

argumenty znormalizowane – $F_i = M_i \beta^{E_i}$, $1 \le |M_i| < \beta$, $E_{\min} \le E_i \le E_{\max}$

znormalizowany wynik działania $F = f(F_1, F_2, \ldots) = M\beta^E$, $1 \le |M| < \beta$

- *nadmiar zmiennoprzecinkowy* (*exponent overflow*) – $E > E_{\max}$
- *niedomiar zmiennoprzecinkowy* (*exponent underflow*) – $E < E_{\min}$

Mnożenie

$$F_1 F_2 = (M_1 \beta^{E_1})(M_2 \beta^{E_2}) = (M_1 M_2)\beta^{E_1+E_2}$$

- $1 \le M = M_1 M_2 < \beta^2$ – normalizacja potrzebna, gdy $M \ge \beta$ i wtedy:

$$M^* = (M_1 M_2)\beta^{-1},\ E^* = (E_1 + E_2) - 1$$

Dzielenie

$$F_1 F_2 = (M_1 \beta^{E_1})/(M_2 \beta^{E_2}) = (M_1 / M_2)\beta^{E_1-E_2}$$

- $\beta^{-1} \le M = M_1 / M_2 < \beta$ – normalizacja potrzebna, gdy $M < 1$ i wtedy:

$$M^* = (M_1 / M_2)\beta,\ E^* = (E_1 - E_2) + 1$$

Dodawanie i odejmowanie

$$F_1 \pm F_2 = (M_1\beta^{E_1}) \pm (M_2\beta^{E_2}) = (M_1 \pm M_2\beta^{-(E_1-E_2)})\beta^{E_1}$$

- *wyrównanie* wykładników jeśli $E_1 \neq E_2$
- *denormalizacja* operandu o *mniejszym* wykładniku ($|M_\#|\beta^{-i} < \beta^p$)
- *utrata dokładności* operandu denormalizowanego (F_2 jeśli $E_1 > E_2$)
- *normalizacja wyniku*:

$$1 \leq |M_1|, |M_2| < \beta \Rightarrow |M_W| = |M_1 \pm M_2\beta^{-(E_1-E_2)}| < 2\beta$$

  - skutkiem normalizacji jest konieczność korekty wykładnika (jeśli $|M_W| \geq \beta$ albo $|M_W| < 1$
  - $|M_W| < 1 \Rightarrow$ *utrata dokładności wyniku, potrzebne dodatkowe cyfry* M_W,
  - możliwe wystąpienie *nadmiaru* lub *niedomiar*

F_1	0,1 0 0 0 0 0 0 0	..×16^4	0,1 0 0 0 0 0 0 0 0 0 ..	×16^4
F_2 (*wyrównane*)	0,0 F F F F F F F C	..×16^4	0,0 F F F F F F F C0 ..	×16^4
$F_1 – F_2$	0,0 0 0 0 0 0 0 1	×16^4	0,0 0 0 0 0 0 0 0 4 0	×16^4
(*postnormalizacja*)	0,1 0 0 0 0 0 0 0	×16^{-3}	0,4 0 0 0 0 0 0 0	×16^{-4}

Dokładność wyników działań – schematy zaokrąglania

normalizacja wyniku wymaga skalowania (przesunięcia arytmetycznego)

- wynikowy mnożnik ilorazu lub sumy może być zbyt mały ($|M|<1$)
  – możliwa *utrata dokładności* → *ochrona*: użycie dodatkowych cyfr
- wynikowy mnożnik iloczynu może być zbyt duży ($|M|\geq\beta$)
  – konieczne zaokrąglenie skalowanego wyniku

$Fl(X)=M_X\beta^{E_X}$ – reprezentacja zmiennoprzecinkowa liczby X

$$X\leq Y\Rightarrow Fl(X)\leq Fl(Y)$$

$$M\beta^{E_X}\leq X<(M+ulp)\beta^{E_X}\Rightarrow |Fl(X)-X|<ulp\cdot\beta^{E_X}$$

zaokrąglanie – przybliżanie z założoną dokładnością

- *obcięcie* (*truncation, chopping*) – ignorowanie cyfr (bitów) nadmiarowych
- *zaokrąglanie „do najbliższej"* (*round-off*) – minimalizacja błędu lokalnego
- *zaokrąglanie „do parzystej"* (*round to even*) – minimalizacja błędu średniego

Realizacja zaokrąglania wymaga dodatkowych cyfr wyniku.

Standaryzacja dwójkowej reprezentacji zmiennoprzecinkowej

Struktura kodu – postulat: łatwe i szybkie porównywanie liczb znakowanych
→ uporządkowanie liczb zgodne z naturalną interpretacją kodów
→ wykładnik na wyższych, znacznik na niższych pozycjach

Kody specjalne –

- kod wykładnika **0…00** – *zero* (f=0) albo liczby zdenormalizowane (f≠0)
- kod wykładnika **1…11**, f=0 – *nieskończoności* ±∞,
- kod wykładnika **1…11**, f≠0 – *nie-liczby*, *NaN* (wyniki, które nie są liczbami)

Kodowanie wykładnika liczb znormalizowanych – wartości dodatnie i ujemne

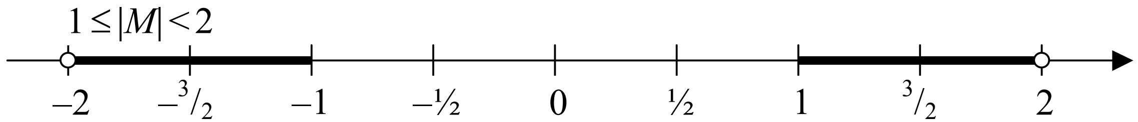
× uzupełnieniowy – problem uporządkowania liczb, asymetria ujemna
wyklucza obliczenie odwrotności liczb bardzo małych.

√ spolaryzowany +N – uporządkowanie naturalne, $N=2^{k-1}-1$, asymetria dodatnia

Liczby znormalizowane i zdenormalizowane

liczba znormalizowana (ukryty bit „1”) – $1{,}00\ldots00_2 \le |M| \le 1{,}11\ldots11_2$

$$F = (-1)^s 2^E (1+f), \quad 0 \le f < 1$$

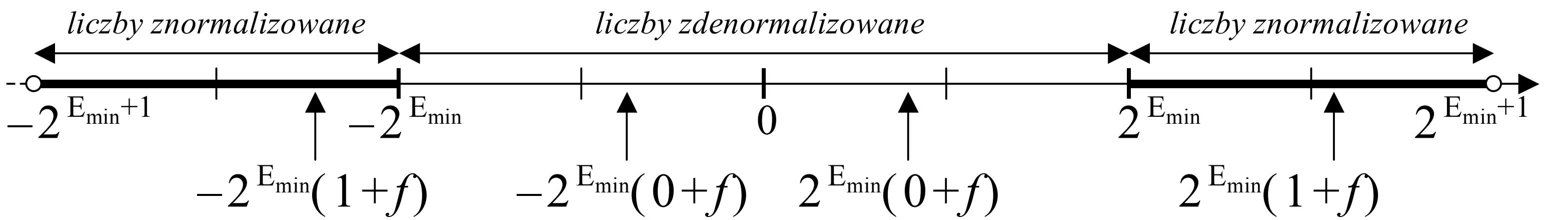


liczba zdenormalizowana (ukryty bit „0”) – kod wykładnika: 00…0

$$F = (-1)^s 2^{E_{\min}} (0+f), \quad 0 \le f < 1$$

najmniejszy wykładnik $E_{\min}$ ma 2 kody: **0...01** – liczba znormalizowana $2^{E_{\min}}(1+f)$

0...00 – liczba zdenormalizowana $2^{E_{\min}}(0+f)$



Liczby znormalizowane i zdenormalizowane

Format zmiennoprzecinkowy IEEE 754-2008

bit ukryty

z	*e*	*e*	*e*	…	*e*	*e*	*e*	*e*	*b*	*b*	*b*	*b*	…	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*
znak	*E* – wykładnik ***k* bitów**								*f* – część ułamkowa modułu mnożnika |*M*| ***m* bitów**																	

32b	SINGLE – [s_{31} | | $E_{30:23}$ | | $f_{22:0}$]	k=8, m=23
64b	DOUBLE – [s_{63} | | $E_{62:52}$ | | $f_{51:0}$]	k=11, m=52
128b	QUADRUPLE – [s_{127+} | | $E_{126:112}$ | | $f_{111:0}$]	k=15, m=112
n×32b	VOID …	k= … , m=32n –k–1
	EXTENDED 32/64/128	k≥11/15/17, m≥32/64/128

Wzorce kodów obiektów standardu IEEE 754-2008

Wykładnik	Ułamek	Kod binarny	Wielkość
$E=0\ldots00$	—	$s\ 0\ldots00\ b\ldots bb$	$F=(-1)^s 2^{E_{\min}}(0,bb\ldots b)$
$E_{\min}\le E\le E_{\max}$	—	$s\ e\ldots ee\ b\ldots bb$	$F=(-1)^s 2^{E}(1,bb\ldots b)$
$E=1\ldots11$	$f=0$	$s\ 1\ldots11\ 0\ldots00$	$\pm\infty$
$E=1\ldots11$	$f\neq0$	$s\ 1\ldots11\ b..bb$	NaN
$E=E_{\max}$	$f=1\ldots11$	$s\ 1\ldots10\ 1\ldots11$	$F_{\max}=(-1)^s 2^{E_{\max}+1}(1-2^{-m-1})$

Cyfry chroniące

Ile dodatkowych cyfr wyniku potrzeba do poprawnego zaokrąglania i postnormalizacji?

- normalizacja w dzieleniu (bez przybliżania)
  → jedna dodatkowa cyfra wyniku – *cyfra chroniąca* (*guard digit*, *G*)
- normalizacja w dodawaniu lub odejmowaniu (bez przybliżania)
  → jeśli $|M|<1$, to jedna dodatkowa cyfra wyniku nie wystarczy:

		G R S	…
F_1	0,1 0 1 1 1 0 0 1 0 1 1 1 0 1 1	0 0 1	..$\times 2^4$
F_2	0,1 0 1 1 1 0 0 1 0 1 1 1 0 1 0	0 0 0	..$\times 2^4$
$F_1–F_2$	0,0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 1	$\times 2^4$
(*postnormalizacja*)	0,1 0 0 ? ? ? ? ? ? ? ? ? ? ? ?	? ? ?	$\times 2^{-10}$

! tylko podwojenie rozmiaru ułamka jest gwarancją utrzymania dokładności

- normalizacja z zaokrąglaniem zwykłym
  → potrzebna dodatkowa cyfra zaokrąglania (*round digit*, *R*)
- normalizacja z zaokrąglaniem symetrycznym – problem gdy $R=½\beta$
  → potrzebny wskaźnik zer na pozostałych pozycjach,
  tzw. „lepki" bit *S* (*sticky bit*): gdy *S*=1, zaokrąglenie w górę

Maszynowe tryby zaokrąglania – standard IEEE754-2008

Standardy zaokrąglania IEEE754-2008

- *do zera* – obcinanie
- *do nieskończoności* – arytmetyka przedziałowa (*interval arithmetic*)
  dodatnie zawsze w górę, ujemne zawsze w dół (lub odwrotnie)
- *do najbliższej* (parzystej) – symetryczne (środek …xx0)

Propagacja poprawki – może być bardzo czasochłonna:

$$
\begin{array}{rl}
1,\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ldots 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 & 1\ 0\ 1 \\
1 & \hookleftarrow \\
\hline
1\ 0,\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & 0\ 0\ 0
\end{array}
$$

- pamięć ROM $2^{l+d} \times l$ (l+d wejść oraz l bitów wyjściowych), $l<m$
  - $M = ...x11...11$ obcinanie ostatnich d bitów
    - błąd zaokrąglania $=-(1-2^{-d})$ zamiast 2^{-d}
    - średni standaryzowany błąd zaokrąglania $=-2^{-(d+l)}$

średnia wartość błędu standaryzowanego – $2^{-l}(2^{l}-2)2^{-d-1} = (1-2^{-l+1})2^{-d-1}$.

Kumulacja błędów podczas działań arytmetycznych*

wynik działania przybliżony ($Fl(X) = X(1+\varepsilon_X)$) $\rightarrow$ ryzyko kumulacji błędów

- błąd względny mnożenia lub dzielenia – niewielka kumulacja

$$\frac{Fl(X)Fl(Y)-XY}{XY} = (1\pm\varepsilon_X)(1\pm\varepsilon_Y)-1 = \varepsilon_X \pm \varepsilon_Y \pm \varepsilon_X\varepsilon_Y \cong \varepsilon_X \pm \varepsilon_Y$$

$$\frac{Fl(X)/Fl(Y)-X/Y}{X/Y} = \frac{(1\pm\varepsilon_X)}{(1\pm\varepsilon_Y)}-1 = \frac{(\varepsilon_X \pm \varepsilon_Y)}{(1\pm\varepsilon_Y)} \cong \varepsilon_X \pm \varepsilon_Y$$

- błąd względny dodawania lub odejmowania

$$\frac{Fl(X)\pm Fl(Y)-(X\pm Y)}{X\pm Y} = \frac{X\varepsilon_X \pm Y\varepsilon_Y}{X\pm Y} = \varepsilon_X \pm \frac{Y}{X\pm Y}(\varepsilon_Y - \varepsilon_X)$$

$\rightarrow$ błąd wyniku jest średnią ważoną błędów argumentów

$\rightarrow$ krytyczna sytuacja w odejmowaniu, gdy $X \cong Y$ oraz $\varepsilon_X = \varepsilon_Y$

utrata dokładności (*cancellation*)

- *łagodna* (*benign*) – argumenty dokładne (zapobieganie – cyfry chroniące)
- *katastroficzna* (*catastrophic*) – argumenty obarczone błędem zaokrąglania

Zapobieganie katastroficznej utracie dokładności i kumulacji błędów*

Zmiana algorytmu – przykłady

1. Obliczanie pierwiastków równania $ax^2+bx+c=0$ według znanej formuły

$$x_{1,2} = \frac{1}{2a}(-b \pm \sqrt{b^2 - 4ac})$$

może spowodować bardzo dużą niedokładność jednego z nich, gdy $b^2 >> 4ac$.
Alternatywa – algorytm wykorzystujący wzory Viety ($x_1x_2=c/a$)

$$x_1 = \frac{w}{2a}, \qquad x_2 = \frac{c}{ax_1} = \frac{2c}{w}, \qquad w = -\mathrm{sgn}(b)(|b| + \sqrt{b^2 - 4ac}).$$

2. Obliczanie pola trójkąta według wzoru Herona $S = \sqrt{q(q-a)(q-b)(q-c)}$, gdzie ($q = \frac{1}{2}(a+b+c)$) powoduje bardzo duży błąd przybliżenia, gdy trójkąt jest bardzo „płaski", tzn. gdy $a \approx b + c$, bo wtedy $q-a \approx 0$.
Kahan zaproponował modyfikację tego wzoru do postaci ($a \geq b \geq c$)

$$S = \tfrac{1}{4}\sqrt{[a+(b+c)][c-(a-b)][c+(a-b)][(a+(b-c)]},$$

Nie wystąpi katastroficzna kumulacja błędu, bo gdy $a \approx b$, to $a-b$ jest rzędu c.

Wyjątki, obsługa nadmiaru i niedomiaru

wyjątki zmiennoprzecinkowe – sytuacje zagrożenia poprawności wyniku

- nadmiar
  √ przejściowy → skalowanie $\times 2^{k-1}$ i zapamiętanie skalowania
  √ permanentny (nadmiar po skalowaniu) → sygnalizacja
- niedomiar
  √ przejściowy skalowanie $\times 2^{-(k-1)}$ i zapamiętanie
  √ permanentny (niedomiar po skalowaniu) → sygnalizacja
- utrata dokładności
  → zmiana algorytmu
- niedozwolona operacja
  → sygnalizacja, zmiana algorytmu
- argument lub wynik nie jest liczbą
  → cicha NaN (*quiet NaN*) – kontynuacja
  → sygnalizowana NaN (*signalling NaN*) – sygnalizacja błędu
- błąd zaokrąglenia

Działania na kodach wykładników

*Kodowanie wykładnika – (*k *– liczba bitów,* e *– kod,* E *– wartość wykładnika)*

- kod spolaryzowany "$+2^{k-1}-1$" ($e_{\min}=00\ldots01_2$, $e_{\max}=11\ldots10_2$)
- liczba zdenormalizowana – $e=00\ldots00_2$, $E=E_{\min}$
- nieskończoności i nie-liczby (*NaN*) – $e=11\ldots11_2$
- *łatwa konwersja kodu spolaryzowanego na kod U2 i odwrotnie*

$$\left|\{x_{k-1},x_{k-2},\ldots,x_1,x_0\}_{+(2^{k-1}-1)}\right| = -\left|\{x_{k-1},\overline{x}_{k-2},\ldots,\overline{x}_1,\overline{x}_0\}_{\mathrm{U2}}\right|$$

W działaniach na wykładnikach (dodawanie, odejmowanie, przesunięcie, zmiana znaku) *operacyjna* zmiana znaku argumentów jest nieistotna.

Uniwersalna procedura:

0. Jeśli argument zdenormalizowany, kod 00...00 zmień na 00...01
1. Przekoduj wykładniki na kod U2
2. Wykonaj działanie w kodzie U2
3. Przekoduj wynik na kod spolaryzowany $+(2^{k-1}-1)$
4. Jeśli potrzebna jest normalizacja skoryguj kod wynikowy

Dodawanie i odejmowanie

Niech $E_1 \geq E_2$ oraz $F_1 \geq F_2$. Wtedy sumą/różnicą jest:

$$F_1 \pm F_2 = (-1)^{z_1}(1+f_1)2^{E_1} \pm (-1)^{z_2}(1+f_2)2^{E_2} =$$
$$= (-1)^{z_1} 2^{E_1}\{(1+f_1) \pm (-1)^{z_2 - z_1}(1+f_2)2^{-(E_1-E_2)}\} = (-1)^{z_1} 2^{E_1} S$$

Odległość wykładników $E_1 - E_2 = [E_1 + (2^{k-1}-1)] - [E_2 + (2^{k-1}-1)] = e_1 - e_2$ można obliczyć używając ich kodów e_1 i e_2, zbędne jest obliczanie wartości.

Obliczona suma S może być nieznormalizowana:

- jeśli $S \geq 2$, to należy zwiększyć wykładnik wyniku o 1
- jeśli $\frac{1}{2} \leq S < 1$ (S=0,1xx…), to bity GRS wystarczą do normalizacji, a wykładnik wyniku należy zmniejszyć o 1
- jeśli $\frac{1}{4} \leq S < \frac{1}{2}$ (S=0,01xx…), to zaokrąglenie jest niepewne, a wykładnik wyniku należy zmniejszyć o 2
- jeśli $S < \frac{1}{4}$ (0,00xx…), to wystąpi katastroficzna utrata dokładności
- zwiększanie i zmniejszanie wykładnika można wykonać traktując kod wykładnika jak kod binarny, bo $(E + q + (2^{k-1}-1))_2 = e + q$;
- może wystąpić nadmiar lub niedomiar

Mnożenie i dzielenie

$$F_1 \cdot F_2 = (-1)^{z_1}(1+f_1)2^{E_1} \cdot (-1)^{z_2}(1+f_2)2^{E_2} = (-1)^{z_1+z_2}(1+f_1)(1+f_2)2^{E_1+E_2}$$

$$F_1 / F_2 = \frac{(-1)^{z_1} 2^{E_1}\{(1+f_1)}{(-1)^{z_2} 2^{E_2}\{(1+f_2)} = (-1)^{z_1-z_2} \frac{(1+f_1)}{(1+f_1)} 2^{E_1-E_2}$$

Obliczenie sumy/różnicy wykładników – najłatwiej po przekodowaniu na U2:

$$\left|\{x_{k-1}, x_{k-2}, \ldots, x_1, x_0\}_{+(2^{k-1}-1)}\right| = -\left|\{x_{k-1}, \overline{x}_{k-2}, \ldots, \overline{x}_1, \overline{x}_0\}_{\mathrm{U2}}\right|$$

- łatwe wykrycie nadmiaru, niedomiaru

Mnożenie:

- jeśli $(1+f_1)(1+f_2) \geq 2$, zmniejsz e_1
- przekoduj e_1 i e_2 na U2, wykonaj dodawanie, przekoduj na $+(2^{k-1}-1)$

Dzielenie:

- jeśli $(1+f_1)/(1+f_2) < 1$, zwiększ e_1
- przekoduj e_1 i e_2 na U2, wykonaj odejmowanie, przekoduj na $+(2^{k-1}-1)$

Obliczanie odwrotności liczby zmiennoprzecinkowej

Jeśli kod liczby zawiera wykładnik *E* oraz ułamek *f*, czyli

$$F = (-1)^s (1 + f) 2^E,$$

to odwrotnością tej liczby jest

$$F^{-1} = (-1)^s (1 + f)^{-1} 2^{-E} = (-1)^s \frac{2}{1+f} 2^{-(E+1)},$$

Iloraz 2/(1+*f*) jest znormalizowany, więc wykładnik odwrotności to –(*E*+1)

Zmiana znaku liczby w kodzie spolaryzowanym +(2^{k-1}–1):

Uzupełnienie liczby U2 polega na dopełnieniu bitów i dodaniu 1 na pozycji najniższej, a ponieważ $\left|\{x_{k-1}, x_{k-2}, \ldots, x_1, x_0\}_{+(2^{k-1}-1)}\right| = -\left|\{x_{k-1}, \overline{x}_{k-2}, \ldots, \overline{x}_1, \overline{x}_0\}_{\mathrm{U2}}\right|$ więc kod spolaryzowany "+2^{k-1}–1" liczby przeciwnej do danej powstanie jako dopełnienie bitów i odjęcie 1 na pozycji najniższej:

$$-X_{2^{k-1}-1} = -\left(\sum_{i=0}^{k-1} x_i 2^i - (2^{k-1} - 1)\right) + (2^{k-1} - 1) = \sum_{i=0}^{k-1} (1 - x_i) 2^i - 1$$

WNIOSEK: Kod wykładnika –(*E*+1) powstanie analogicznie przez odjęcie 10_2

Obliczanie pierwiastka kwadratowego

$$\sqrt{(1+f)2^E} = 2^{\lfloor E/2 \rfloor}\sqrt{(1+f)2^{E \bmod 2}}$$

czyli

$$\sqrt{(1+f)2^E} = \begin{cases} 2^{E/2}\sqrt{1+f} & \text{gdy } E \text{ jest parzyste} \\ 2^{(E-1)/2}\sqrt{2(1+f)} & \text{gdy } E \text{ jest nieparzyste} \end{cases}$$

Procedura:

- jeśli wykładnik jest parzysty oblicz $\sqrt{1+f}$
  jeśli wykładnik jest nieparzysty oblicz $\sqrt{2(1+f)}$
- jeśli wykładnik jest nieparzysty (kod „…xx0”) zmniejsz kod o 1.
- utwórz kod połowy wykładnika parzystego (E lub E–1) „…xx1”:

$$\left|\{x_{k-1}, x_{k-2}, \ldots, x_1, 1\}_{+(2^{k-1}-1)}\right| = -\left|\{x_{k-1}, \overline{x}_{k-2}, \ldots, \overline{x}_1, 0\}_{\mathrm{U2}}\right|$$

Ponieważ $\frac{1}{2}\left|\{x_{k-1}, \overline{x}_{k-2}, \ldots, \overline{x}_1, 0\}_{\mathrm{U2}}\right| = \left|\{x_{k-1}, x_{k-1}, \overline{x}_{k-2}, \ldots, \overline{x}_2, \overline{x}_1\}_{\mathrm{U2}}\right|$ więc kod E/2 jest:

$$\tfrac{1}{2}\left|\{x_{k-1}, x_{k-2}, \ldots, x_1, 1\}_{+(2^{k-1}-1)}\right| = \left|\{x_{k-1}, \overline{x}_{k-1}, x_{k-2}, \ldots, x_1\}_{+(2^{k-1}-1)}\right|$$

Szybkie obliczenie wartości wykładnika

W każdym kodzie binarnym $+(2^{k-1}-1)$ mamy odpowiednio:

- kodem zera jest

$$011\ldots11$$

Stąd wynika, że wartością liczby ujemnej o kodzie $0x_{k-2}x_{k-3}\ldots x_1x_0$ jest

$$\left|\{0x_{k-2}x_{k-3}\ldots x_1x_0\}_{2^{k-1}-1}\right| = -\left|\{0\overline{x}_{k-2}\overline{x}_{k-3}\ldots\overline{x}_1\overline{x}_0\}_2\right|,$$

na przykład wartością ciągu 01111110101 jest –10 (minus dziesięć)

- kodem wartości jeden jest

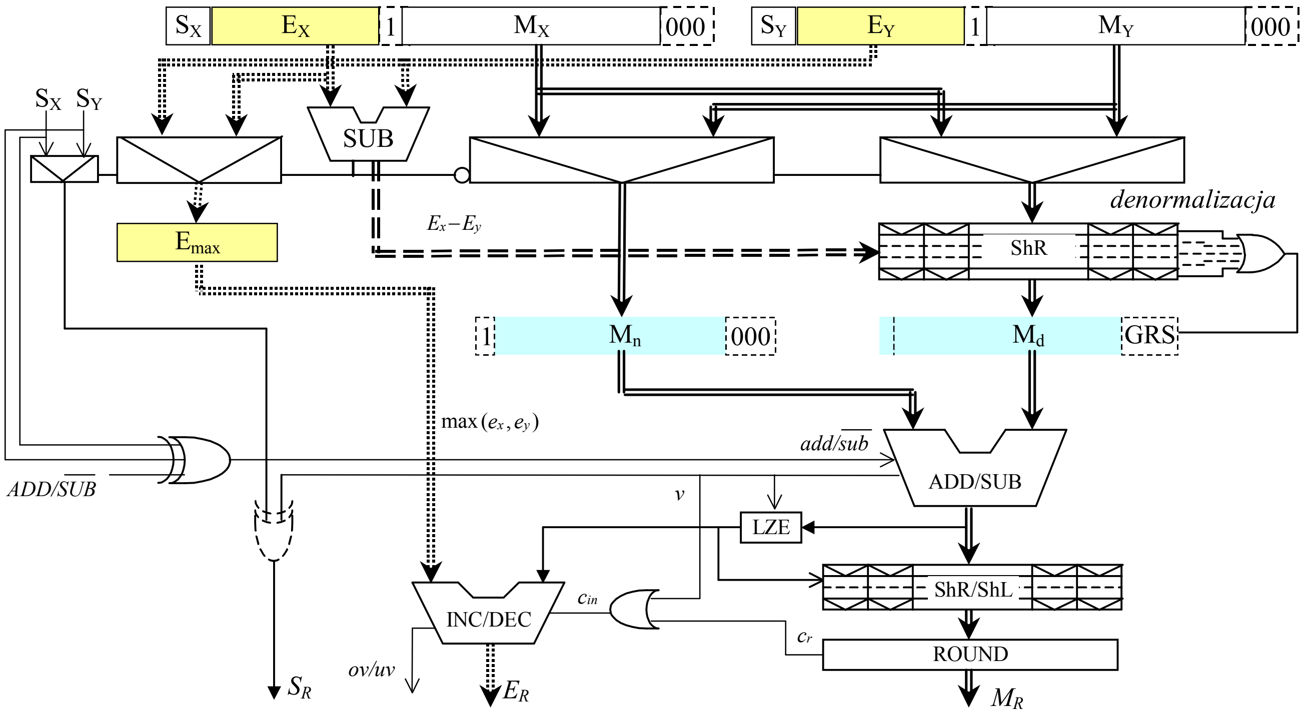
$$100\ldots00$$

Stąd wynika, że wartością liczby dodatniej o kodzie $1x_{k-2}x_{k-3}\ldots x_1x_0$ jest

$$\left|\{1x_{k-2}x_{k-3}\ldots x_1x_0\}_{2^{k-1}-1}\right| = 1 + \left|\{0x_{k-2}x_{k-3}\ldots x_1x_0\}_2\right|,$$

na przykład wartością ciągu 10000001101 jest +14 (plus czternaście)

Sumator zmiennoprzecinkowy – denormalizacja

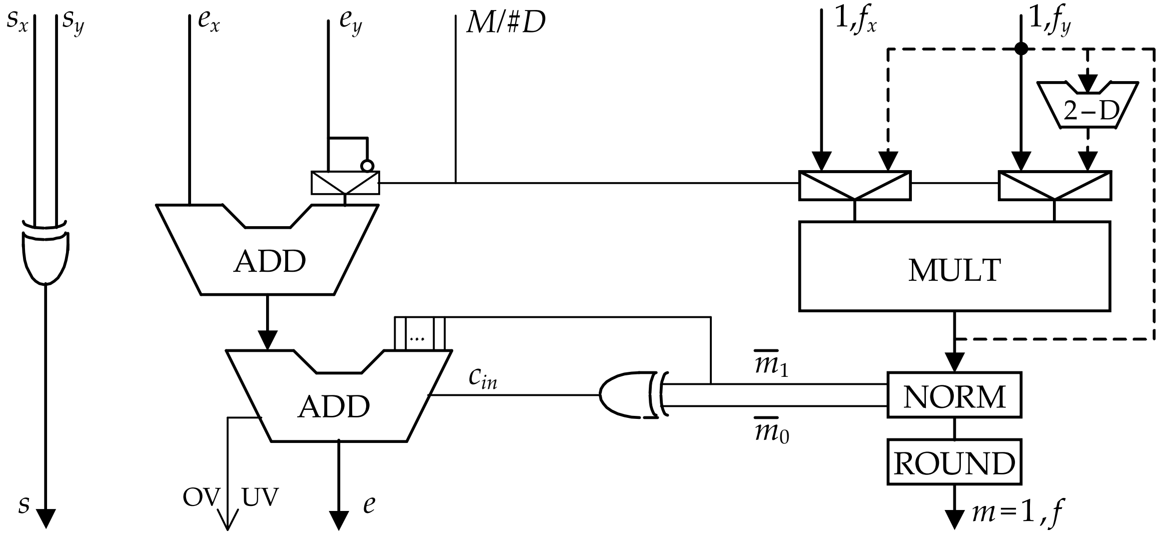


Moduł wykładnika: SIGN – generator znaku wyniku, SUB – układ odejmujący wykładniki, *Moduł mnożnika*: ShR – układ denormalizacji (przesunięcia w prawo), ADD/SUB – sumator mnożników, LZE – koder wiodących zer, ShR/ShL – układ postnormalizacji, INC/DEC – układ korekcji wykładnika, ROUND – układ zaokrąglania (GRS – bity dodatkowe).

Zmiennoprzecinkowy układ mnożąco-dzielący

Mnożenie i dzielenie (# – mnożenie lub dzielenie)

$$F_1 \# F_2 = M_1\beta^{E_1} \# M_2\beta^{E_2} = (M_1 \# M_2)\beta^{E_1 \pm E_2}$$



Mnożenie zmiennoprzecinkowe (——) i obliczanie odwrotności dzielnika (- - -)
(2–D – uzupełnianie przybliżenia, MULT – matryca mnożąca, NORM – przesuwnik, ADD – sumator, ROUND – układ zaokrągleń, m_1, m_0 – bity części całkowitej iloczynu)

- problem – obsługa liczb denormalizowanych

Sumator zmiennoprzecinkowy – normalizacja i zaokrąglanie*

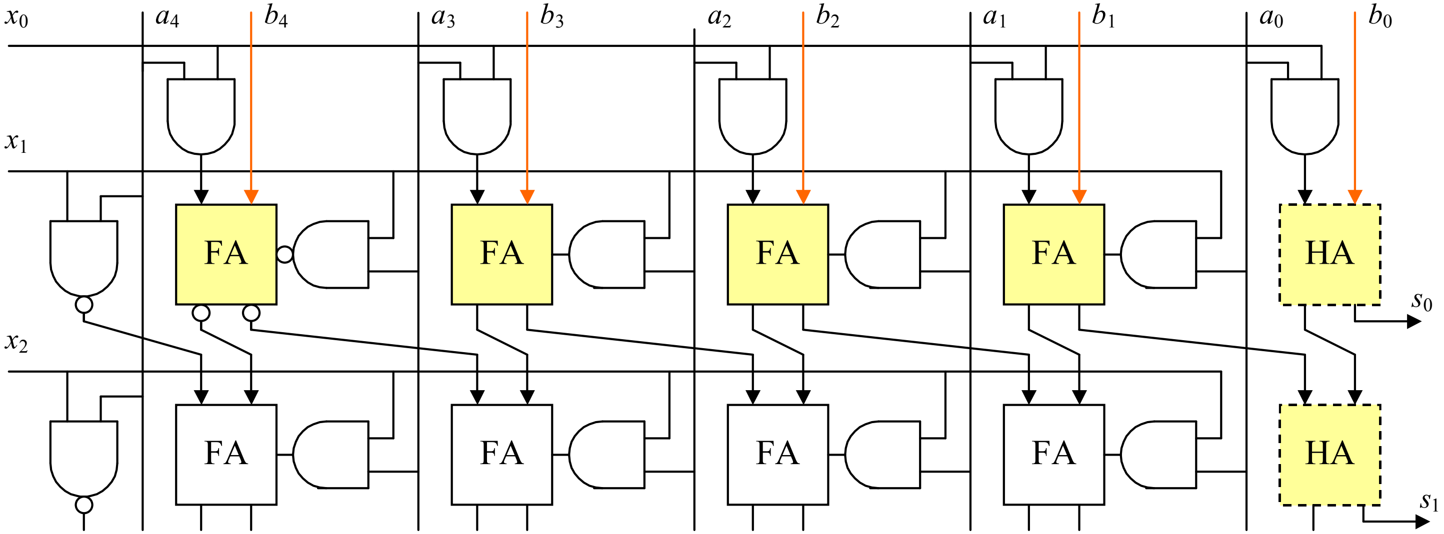
- przynajmniej jeden argument znormalizowany
  jeśli suma $2^{-2}<A<4$, możliwe jest zaokrąglenie i normalizacja bez utraty dokładności – przesunięcie w prawo ($A>2$) lub w lewo o 1 lub 2 pozycje co jest równoważne przesuwaniu zawsze w lewo o 0, 1, 2 lub 3 pozycje i przekierowaniu wyjść o 1 pozycję w prawo! (prostszy przesuwnik)
  jeśli suma $<2^{-2}$ musi nastąpić utrata dokładności (lepki bit S sumy jest bitem znaczącym ułamka!)
- wykładniki jednakowe (lub denormalizowane) – sterowanie komutatora: może wystarczyć porównanie pary najwyższych bitów ułamka: jeśli są identyczne $((f_{X1}\oplus f_{Y1})+(f_{X2}\oplus f_{Y2})=0)$ nastąpi katastroficzna utrata precyzji
- problem normalizacji gdy suma zawiera długi ciąg jedynek na niższych bitach czas normalizacji się wydłuża (propagacja)
- działania mogą być wykonane w rozszerzonej precyzji

Mnożenie akumulacyjne

arytmometr zmiennoprzecinkowy

- *wystarczy układ mnożenia akumulacyjnego (matryca, drzewo Wallace'a) z dodatkowym wejściem dla 3. argumentu*

$$M = X * A + B$$



Modyfikacja matrycy mnożącej umożliwiająca mnożenie akumulacyjne

METODY NUMERYCZNE*)

Przybliżanie ilorazu wymiernego jego skończonym rozwinięciem

$$D \neq 0 \Rightarrow \exists\{R_i\} : D_m = D\prod_{i=0}^{m} R_i \to 1 \Rightarrow Q = \frac{X}{D} = \frac{X}{D_m}\prod_{i=0}^{m} R_i \approx X\prod_{i=0}^{m} R_i$$

- dokładność ilorazu – określona precyzją wyznaczenia liczby $R_0 R_1 \ldots R_m$

standaryzacja: (ujemny dzielnik → zmienić znaki *D* oraz *X*)

$$D := D \operatorname{sgn} D, \quad X := X \operatorname{sgn} D$$

normalizacja: $\beta^{m-1} \le D < \beta^m \Rightarrow \beta^{-1} \le d = D\beta^{-m} < 1 \ \& \ x = X\beta^{-m}$

$$0 < q < 1 \Rightarrow (1-q)(1+q)(1+q^2)(1+q^4)\ldots(1+q^{2^n}) = (1-q^{2^{n+1}}) \cong 1$$

procedura:

$$d_i = 1 - z \Rightarrow d_{i+1} = d_i(1+z) = (1-z^2) = d_i(2-d_i)$$

zbieżność procedury – kwadratowa

$$1 - d_i = z < \beta^{-s} \Rightarrow 1 - d_{i+1} = z^2 < \beta^{-2s}$$

Przybliżanie ilorazu skończonym rozwinięciem w systemie dwójkowym

$$\{0_2,1_1,0_0,0_{-1},0_{-2},\ldots\}_{U2}+\{1_2,1_1,1_0,x_{-1},x_{-2},x_{-3},\ldots\}_{U2}=\{0_2,0_1,1_0,x_{-1},x_{-2},x_{-3},\ldots\}_{U2}$$

$$D>0\Rightarrow 2+\left|\{1,x_{-1},x_{-2},x_{-3},\ldots\}_{U2}\right|=\left|\{1,x_{-1},x_{-2},x_{-3},\ldots\}_{NB}\right|$$

Procedura

0° $R_0: 1-2^{-s}\le D_1=R_0D<1,\quad Q_1=R_0X$ (np. $R_0=2^{-m}$: $1-2^{-s}\le 2^{-m}D\le 1$)

1° **obliczaj** $R_i=2-D_i$ oraz $Q_{i+1}=Q_iR_i$ **aż** $1-2^{-n}<D_{i+1}=D_iR_i\le 1$

- liczba wiodących jedynek liczb D_i zostaje podwojona w każdej iteracji

$$1-2^{-p}\le D_i<1\Rightarrow 1-2^{-2p}\le D_{i+1}<1$$

- $1-2^{-s}\le D_1<1\Rightarrow$ względną dokładność 2^{-n} ilorazu $Q\approx XR_0R_1\ldots$ zapewnia $m=\lceil\log_2 n/s\rceil+1$ iteracji

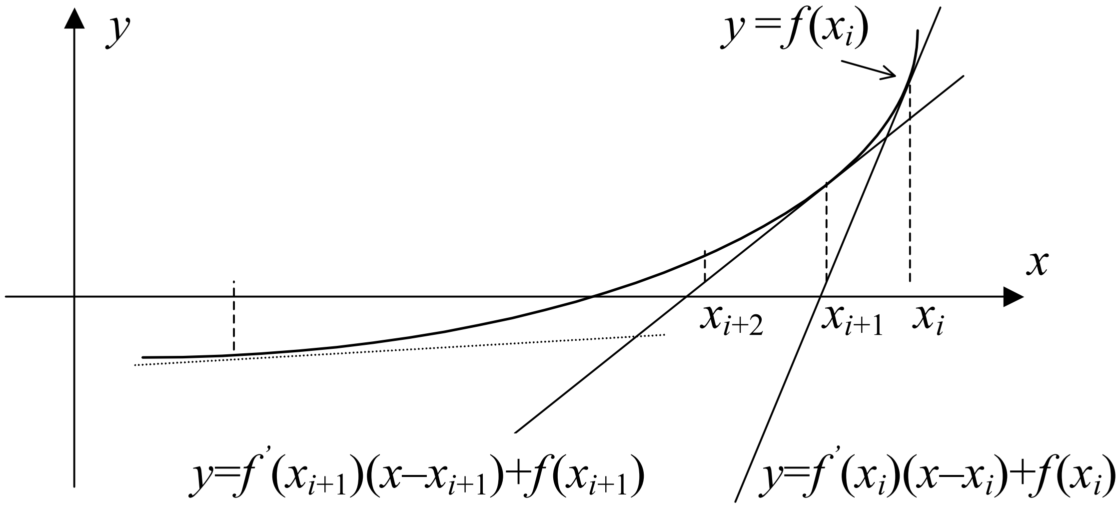
  $\Rightarrow$ pierwszy mnożnik R_0 – wyznaczony z dokładnością $s+\log_2 s$ bitów

  $R_0=f(D)$ – z matrycy ROM o rozmiarze $2^s\times(s+\log_2 s)$ bitów

    - przyśpieszenie mnożenia → użycie krótszych mnożników R_i

Obliczanie odwrotności dzielnika

metoda iteracyjna Newtona-Raphsona



kolejne przybliżenia miejsca zerowego $f(x)$ określa równanie rekurencyjne

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

w odniesieniu do funkcji $f(x) = x^{-1} - D$ przybiera postać

$$x_{i+1} = x_i(2 - Dx_i)$$

Zbieżność metody mnożenia przez odwrotność dzielnika

$$\text{Niech } x_0 = a \text{ oraz } Da = 1-q \Rightarrow x_1 = a(1+q) = a(1-q)^{-1}(1-q^2)$$

$$x_i = a(1+q)\ldots(1+q^{2^{i-1}}) = a(1-q)^{-1}(1-q^{2^i})$$

$$x_{i+1} = a(1-q)^{-1}(1-q^{2^i})\{2 - Da[(1+q)\ldots(1+q^{2^{i-1}})]\} = a(1-q)^{-1}(1-q^{2^i})(1+q^{2^i})$$

$$|q|<1 \Rightarrow \lim_{i\to\infty} x_i = \lim_{i\to\infty} a(1-q)^{-1}(1-q^{2^i}) = a(1-q)^{-1} = D^{-1}$$

- dzielnik znormalizowany $\frac{1}{2} \leq |D| < 1 \Rightarrow$ zbieżność, jeżeli $|a|<2$ i $aD>0$.
- zbieżność kwadratowa – jeśli $\delta_i = D^{-1} - x_i$, to
  $$\delta_{i+1} = D^{-1} - x_{i+1} = D^{-1} - (D^{-1} - \delta_i)[2 - D(D^{-1} - \delta_i)] = D\delta_i^2 < \delta_i^2$$
- szybkość zbieżności zależy od dokładności pierwszego przybliżenia
  $(k-1)2^{-j} \leq D < k2^{-j} \Rightarrow$ optymalne $x_0(k) = 2^{j+1}[2^j + k - \frac{1}{2}]^{-1}$

wada – mniejsza dokładność niż uzyskiwana w dzieleniu sekwencyjnym
niezbędna korekcja wyniku → dodatkowe działania arytmetyczne

Obliczanie pierwiastka i odwrotności pierwiastka kwadratowego

Liczba pierwiastkowana jest znormalizowana $\frac{1}{4} \le A < 1$.

metoda iteracyjna Newtona – równanie rekurencyjne: $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$

Obliczanie pierwiastka kwadratowego:

Jeśli $f(x)=x^2-A$, to $x=sqrt(A)$ ($\sqrt{A}$) i wtedy $f'(x)=2x$, więc

$$x_{i+1} = x_i - \frac{x_i^2 - A}{2x_i} = \tfrac{1}{2}x_i + \frac{\tfrac{1}{2}A}{x_i}$$

wada: konieczność dzielenia

Obliczanie odwrotności pierwiastka kwadratowego:

$f(x) = x^{-2} - A$ i wtedy $f'(x) = -2x^{-3}$ oraz

$$x_{i+1} = x_i - \frac{(x_i^{-2} - A)}{-2x_i^{-3}} = \tfrac{1}{2}x_i(3 - x_i^2 A)$$

Obliczanie odwrotności pierwiastków wyższych stopni

Jeżeli $f(x) = x^{-k} - A$, to $x = \sqrt[k]{A^{-1}}$

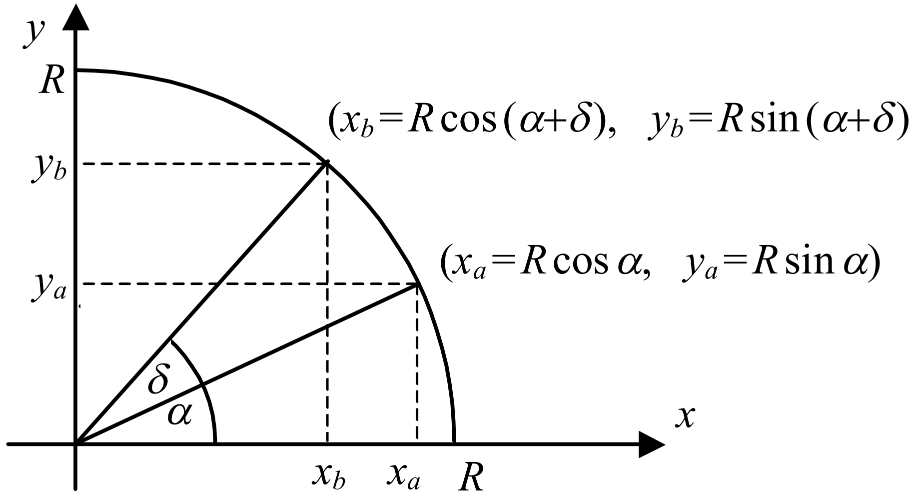
Ponieważ wtedy $f'(x) = -kx^{-k-1}$, więc otrzymujemy

$$x_{i+1} = x_i - \frac{(x_i^{-k} - A)}{-kx_i^{-k-1}} = \tfrac{1}{k} x_i (k + 1 - x_i^k A)$$

Obliczenia wymagają wielokrotnego mnożenia / potęgowania jeśli $k>2$.

CORDIC – algorytm Voldera (1)

Obrót wektora zaczepionego w punkcie (0,0) przestrzeni kartezjańskiej



Z tożsamości trygonometrycznych

$$\cos(\alpha+\delta)=\cos\alpha\cos\delta-\sin\alpha\sin\delta$$

$$\sin(\alpha+\delta)=\sin\alpha\cos\delta+\cos\alpha\sin\delta$$

wynika, że:

$$x_b=R\cos(\alpha+\delta)=R\cos\alpha\ \cos\delta-R\sin\alpha\ \sin\delta=\ x_a\cos\delta-y_a\sin\delta$$

$$y_b=R\sin(\alpha+\delta)=R\sin\alpha\ \cos\delta+R\ \cos\alpha\ \sin\delta=\ y_a\cos\delta+x_a\sin\delta$$

CORDIC – algorytm Voldera (2)

J.Volder (1956, sterowanie samolotu B-58)

Podstawiając $t = \mathrm{tg}\,\delta$ $(\cos^{-2}\delta = 1 + t^2)$, otrzymamy dla kątów w I ćwiartce:

$$\sqrt{(1+t^2)^{-1}}\,R\cos(\alpha + \mathrm{arctg}\,t) = R\cos\alpha - tR\sin\alpha$$

$$\sqrt{(1+t^2)^{-1}}\,R\sin(\alpha + \mathrm{arctg}\,t) = R\sin\alpha + tR\cos\alpha$$

W krokach iteracji, to wynikiem obrotu wektora (x_i, y_i) o kąt arctg t_i jest:

$$\sqrt{(1+t_i^2)^{-1}}\; x_{i+1} = x_i - t_i y_i, \quad i=0,1,\ldots$$

$$\sqrt{(1+t_i^2)^{-1}}\; y_{i+1} = y_i + t_i x_i, \quad i=0,1,\ldots$$

Obie współrzędne są jednakowo skalowane, więc w pojedynczym kroku można zignorować wydłużenie wektora, dokonując korekty w ostatnim kroku obliczeń

$$x_{i+1}^* = x_i^* - t_i y_i^* \;\text{ oraz }\; y_{i+1}^* = y_i^* + t_i x_i^*$$

gdzie $x_0^* = x_0,\; y_0^* = y_0$ oraz

$$x_n^* = x_n \prod_{i=0}^{n-1} \sqrt{1+t_i^2}, \quad y_n^* = y_n \prod_{i=0}^{n-1} \sqrt{1+t_i^2}\,.$$

CORDIC – algorytm Voldera (3)

Podobne tożsamości dotyczą funkcji hiperbolicznych sinh i cosh (wzór Eulera)

$$\cosh(\alpha+\delta)=\cosh\alpha\,\cosh\delta-\sinh\alpha\,\sinh\delta$$

$$\sinh(\alpha+\delta)=\sinh\alpha\,\cosh\delta+\cosh\alpha\,\sinh\delta$$

gdzie (wzór Eulera: $\exp ix=\cos x+i\sin x$)

$$2\sinh x=-2i\sin x=\exp x-\exp(-x)$$

$$2\cosh x=2\cos x=\exp x+\exp(-x)$$

$$\exp x=\cosh x+\sinh x$$

Wartości $t=\mathrm{tg}\,\delta=\pm 2^{-n}$, można łatwo tablicować i wtedy wszystkie obliczenia można wykonać za pomocą dodawania, odejmowania i przesunięcia.

Zależnie od znaku kąta wyróżnia się obliczenia

- w *trybie obrotu* (*rotation mode*), gdy kąt jest dodatni,
- w *trybie normowania* (*vectoring mode*), gdy kąt jest ujemny – jego wynikiem jest obliczenie długości wektora

CORDIC – algorytm Voldera (4)

trzecia zmienna – z_i odległość kątowa wektora od osi $[0,x)$:

$$x_{i+1} = x_i + m\sigma_i t_i y_i$$

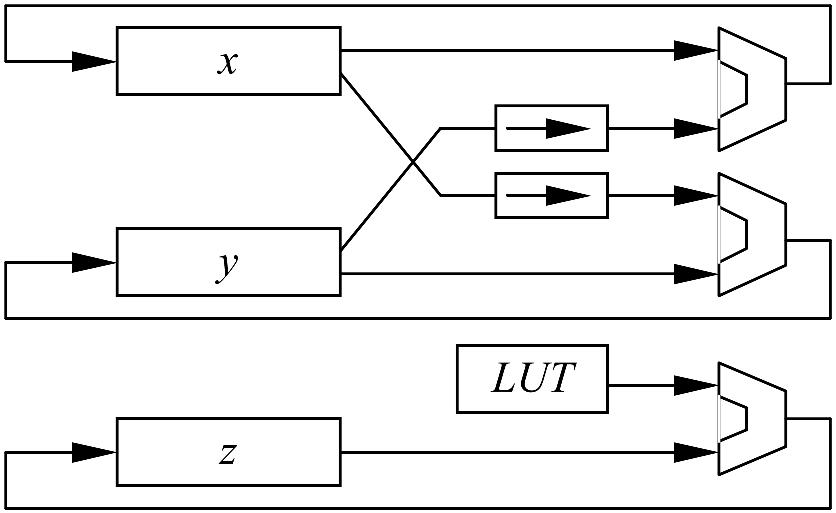
$$y_{i+1} = y_i - \sigma_i t_i x_i$$

$$z_{i+1} = z_i + \sigma_i (1/\sqrt{m}) \operatorname{arctg}\sqrt{m}\, t_i$$

gdzie $t_i = 2^{-S(m,i)}$ – przyjęta sekwencja iteracji przyrostów

$K = \prod_{i=0}^{n-1} \sqrt{1+t_i^2}$			tryb obrotu ($z_i \to 0$) $\sigma_i = -\operatorname{sign} z_i$	tryb normowania ($y_i \to 0$) $\sigma_i = \operatorname{sign}(x_i y_i)$
trygonometr.	$m=1$	$\operatorname{arctg} 2^{-k}$	$x_n \to K(x_0 \cos z_0 - y_0 \sin z_0)$ $y_n \to K(y_0 \cos z_0 - x_0 \sin z_0)$ $z_n \to 0$	$x_n \to K\sqrt{x_0^2 + y_0^2}$ $y_n \to 0$ $z_n \to z_0 - \operatorname{arctg} 2(y_0/x_0)$
hiperboliczny	$m=-1$	$\tanh^{-1} 2^{-k}$	$x_n \to K(x_0 \cosh z_0 - y_0 \sinh z_0)$ $y_n \to K(y_0 \cosh z_0 - x_0 \sinh z_0)$ $z_n \to 0$	$x_n \to K\sqrt{x_0^2 - y_0^2}$ $y_n \to 0$ $z_n \to z_0 - \tanh^{-1} 2(y_0/x_0)$

CORDIC – realizacja układowa



Zalety algorytmu CORDIC

obliczanie funkcji elementarnych za pomocą prostych działań arytmetycznych

prosta implementacja układowa algorytmu (Cyrix, procesory DSP)

Wada – wolna zbieżność, konieczność wykonania dużej liczby obliczeń,

$\rightarrow\rightarrow$ wersja ulepszona CORDIC–2.

Inne metody obliczania wartości funkcji przestępnych

tablica odniesień (*look-up table*)

- zapamiętanie wartości funkcji jednej zmiennej z dokładnością do n bitów – matryca ROM o rozmiarze $n \times 2^n$ bitów (dla $n > 23$ rozmiar > 128 Mb)

rozwinięcie w szereg Taylora

- różne algorytmy dla poszczególnych funkcji
- wolna zbieżność szeregu Taylora (zależy silnie od wartości argumentu)

rozwinięcia funkcji przestępnych w postaci ułamków wymiernych

- powszechnie stosowane w implementacjach programowych
- mogą być bardzo skuteczne w realizacjach sprzętowych, jeśli w dyspozycji są szybkie zmiennoprzecinkowe sumatory i układy mnożące

algorytmy oparte na przybliżeniach wielomianowych z użyciem tablic odniesień

- domena argumentu funkcji $f(x)$ podzielona na przedziały równej długości
- wartości graniczne $f(x_i)$ w punktach podziału x_i są w tablicy odniesień
- wartości wewnątrz przedziałów obliczane na podstawie aproksymacji wielomianowej $p(x - x_i)$ funkcji $f(x - x_i)$

Dodatek

Zakres wykładnika*)

- skrajne wartości wykładnika są potrzebne do zakodowania:
  – zera i ewentualnie liczb zdenormalizowanych
  – nieskończoności i obiektów specjalnych, tzw. nie-liczb (*NaN*)

⇒ rozpiętość zakresu k-pozycyjnego wykładnika: $E_{max}-E_{min}=(\beta^k-1)-2$

- odwrotność bezwzględnie najmniejszej liczby znormalizowanej powinna być obliczalna, więc musi być znormalizowana:

$$\left|F_{\min}^{-1}\right| = (\beta^{E_{\min}}\beta^{p-1})^{-1} = \beta^{-E_{\min}}\beta^{-p+1} < \beta^{E_{\max}}\beta^{p} \Rightarrow E_{\max}+E_{\min} > -2p+1$$

Jeśli podstawa β jest parzysta, to rozpiętość zakresu jest nieparzysta, więc także $E_{max}+E_{min}$ musi być nieparzyste.

Zakładając najmniejszą asymetrię, czyli $E_{max}+E_{min}=-2p+3$, mamy

- odwrotność bezwzględnie największej liczby znormalizowanej:

$$\left|F_{\max}^{-1}\right| \geq (\beta^{E_{\max}}\beta^{p})^{-1} = \beta^{-E_{\max}}\beta^{-p} = \beta^{E_{\min}+2p-3}\beta^{-p} = \beta^{E_{\min}}\beta^{p-3}$$

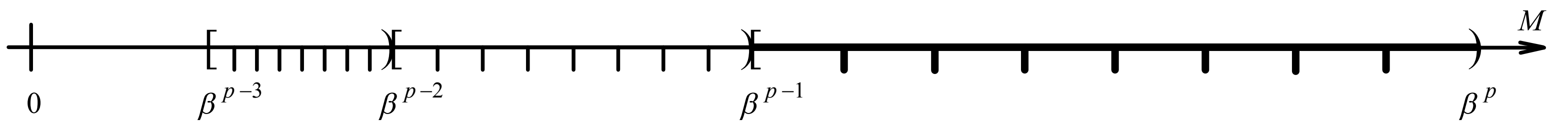
Dokładność przybliżenia rośnie ze zwiększaniem p, więc $p=1$ jest lepszym wyborem niż $p=0$.

Dokładność reprezentacji (przybliżenia liczby rzeczywistej)*)

$Fl(X)$ – reprezentacja zmiennoprzecinkowa liczby rzeczywistej X

$$M\beta^E \le X \le (M+ulp)\beta^E \;\Rightarrow |\,Fl(X)-X\,| \le \tfrac{1}{2}ulp\cdot\beta^E$$

- odległość dwóch kolejnych liczb – zależy od wykładnika



- *względny błąd* reprezentacji, lokalny – $\varepsilon(X)$ i maksymalny

$$|\,\varepsilon(X)\,| = \frac{|\,Fl(X)-X\,|}{X} \le \frac{ulp\beta^E}{2M\beta^E} = \frac{ulp}{2M} \le \tfrac{1}{2}\beta^{-(m+p)+1} = \max_M |\,\varepsilon(X)\,| = MRRE$$

- *średni błąd względny* ARRE (*average relative representation error*) (rozkład błędu jest logarytmiczny …)

$$\mathrm{ARRE} = \int_{\beta^{p-1}}^{\beta^p} \frac{\varepsilon(X)}{X\ln\beta}\,\mathrm{d}X = ulp\,\frac{\beta-1}{4\ln\beta}\,\beta^{-p} = \frac{\beta-1}{4\ln\beta}\,\beta^{-(p+m)}$$

Kodowanie mnożnika*)

w kodzie U2:

warunek $1 \le \left|M \cdot 2^{-(p-1)}\right| < 2$ *rozdziela się na rozłączne warunki:*

$$01{,}00\ldots00 \le M \cdot 2^{-(p-1)} \le 01{,}11\ldots11, \ \text{gdy } M > 0$$

$$10{,}00\ldots01 \le M \cdot 2^{-(p-1)} \le 11{,}00\ldots00, \ \text{gdy } M < 0$$

- *warunek normalizacji trudny do sprawdzenia*
- *trudne porównanie liczb – porządek kodów liczb nie może być zgodny z porządkiem liczb*

w kodzie znak-moduł:

warunek normalizacji $1 \le \left|M \cdot 2^{-(p-1)}\right| < 2$ *upraszcza się do postaci*

$$1{,}00\ldots00 \le |M| \cdot 2^{-(p-1)} \le 1{,}11\ldots11 \Rightarrow |M| \cdot 2^{-(p-1)} = 1 + f = 1{,}b_1 b_2 b_3 \ldots b_m$$

- *łatwe porównanie liczb – porządek kodów liczb może być zgodny z porządkiem liczb*
- *nie trzeba zapisywać wiodącej „1”* (*„**bit ukryty**”*)
- *gdy* p=1, *zapisany jest kod ułamka*

Obcięcie*)

d – liczba bitów obcinanych

$$Fl(Y) = T(Y) = M \Leftrightarrow M \leq Y < M + ulp,$$

- standaryzowany błąd obcinania ($Y = M + i2^{-d}ulp,\ \ 0 \leq i \leq 2^d - 1$)

$$\frac{T(Y) - Y}{ulp} = -i2^{-d}, \quad 0 \leq i \leq 2^d - 1, \quad 0 \leq i \leq 2^d - 1$$

- średni standaryzowany błąd obcinania (rozkład Y równomierny)

$$\delta_T = 2^{-d} \sum_{i=0}^{2^d-1} (-i)2^{-d} = -(2^{d-1} - \tfrac{1}{2})2^{-d} = -(\tfrac{1}{2} + 2^{-d-1})$$

- błąd względny i średni jest zawsze ujemny, bowiem
- skutkiem obcinania jest zawsze niedoszacowanie
- estymator $T(Y)$ jest ujemnie obciążony (*negative biased*).

Zaokrąglanie zwykłe – przyciąganie do najbliższej*)

$$Fl(Y) = R(Y) = \begin{cases} M, & \text{gdy} \quad Y < M + \tfrac{1}{2}ulp, \\ M + ulp, & \text{gdy} \quad Y \geq M + \tfrac{1}{2}ulp. \end{cases}$$

(możliwe przeciwne przypisanie $R(Y)$ przy $Y = M + \frac{1}{2}ulp$)

- standaryzowany błąd zaokrąglania ($Y = M + i2^{-d}ulp, \;\; 0 \leq i \leq 2^d - 1$)

$$\frac{R(Y) - Y}{ulp} = \begin{cases} -i2^{-d}, & \text{gdy} \quad 0 \leq i < 2^{d-1} \;\; (0 \leq Y - M < \tfrac{1}{2}ulp), \\ 1 - i2^{-d}, & \text{gdy} \quad 2^{d-1} \leq i < 2^d \;\; (\tfrac{1}{2}ulp \leq Y - M < ulp). \end{cases}$$

- średni standaryzowany błąd zaokrąglania (rozkład Y równomierny)

$$\delta_R = 2^{-d} \left\{ \sum_{i=0}^{2^{d-1}-1} (-i)2^{-d} + \sum_{i=2^{d-1}}^{2^d - 1} (1 - i2^{-d}) \right\} = \tfrac{1}{2} 2^{-d}$$

- średni błąd zaokrąglania jest bardzo bliski 0
- estymator $R(Y)$ obciążony dodatnio (lub ujemnie) (zależnie od definicji)
- przypisanie $R(Y)$ przy $Y = M + \frac{1}{2}ulp$ zależne od znaku M ($R\downarrow 0$ lub $R\uparrow\infty$)

Zaokrąglanie symetryczne (do parzystej)*)

$$Fl(Y)=S(Y)=\begin{cases} M-ulp, & \text{gdy} \quad -ulp\le Y-M<-\tfrac{1}{2}ulp,\\ M, & \text{gdy} \quad -\tfrac{1}{2}ulp\le Y-M\le+\tfrac{1}{2}ulp,\\ M+ulp, & \text{gdy} \quad +\tfrac{1}{2}ulp<Y-M<+ulp,\end{cases}$$

- standaryzowany błąd przybliżenia ($Y=M+i2^{-d}ulp,\ \ -2^{d}\le i\le 2^{d}-1$)

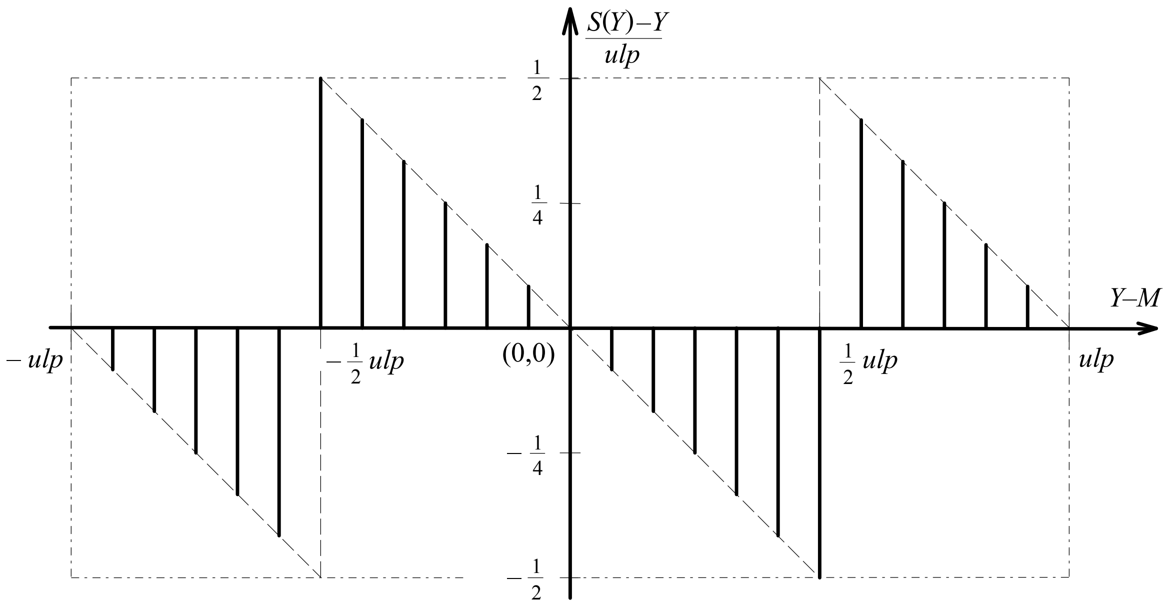
$$\frac{S(Y)-Y}{ulp}=\begin{cases} -i2^{-d}-1, & \text{gdy} \quad -2^{d}\le i<-2^{d-1},\\ -i2^{-d}, & \text{gdy} \quad -2^{d-1}\le i\le 2^{d-1},\\ -i2^{-d}+1, & \text{gdy} \quad 2^{d-1}<i<2^{d},\end{cases}$$

- średni standaryzowany błąd zaokrąglania symetrycznego

$$\delta_S=2^{-2d}\left\{\left(-\sum_{i=-2^{d}}^{-2^{d-1}-1}(i2^{-d}+1)-\sum_{i=-2^{d-1}}^{-1}i2^{-d}\right)+\left(-\sum_{i=0}^{2^{d-1}}i2^{-d}-\sum_{i=2^{d-1}+1}^{2^{d}-1}(i2^{-d}-1)\right)\right\}=0$$

- estymator $S(Y)$ nieobciążony (średni błąd zaokrąglania równy 0)
- zaokrąglanie do parzystej (*nearest-even*) lub nieparzystej (*nearest-odd*)

Niedokładność przybliżenia*)



- propagacja przeniesienia podczas zaokrąglania

M	$\ldots x0\cancel{00}$	$\ldots x0\cancel{01}$	$\ldots x0\cancel{10}$	$\ldots x0\cancel{11}$	$\ldots x1\cancel{00}$	$\ldots x1\cancel{01}$	$\ldots x1\cancel{10}$	$\ldots x1\cancel{11}$
$T(M)$	$\ldots x0$	$\ldots x0$	$\ldots x0$	$\ldots x0$	$\ldots x1$	$\ldots x1$	$\ldots x1$	$\ldots x1$
$R(M)$	$\ldots x0$	$\ldots x0$	$\ldots x1$	$\ldots x1$	$\ldots x1$	$\ldots x1$	$\ldots x1+1$	$\ldots x1+1$
$S(M)$	$\ldots x0$	$\ldots x0$	$\ldots x0$	$\ldots x1$	$\ldots x1$	$\ldots x1$	$\ldots x1+1$	$\ldots x1+1$

Reprezentacja dwójkowa*)

Kodowanie mnożnika – kod znak-moduł

- *moduł mnożnika znormalizowanego* ma postać $1,b_{-1}b_{-2}b_{-3}\ldots b_{-m}$

$$1{,}00\ldots00 \le M \le 1{,}11\ldots11 \Rightarrow M = 1{,}b_{-1}b_{-2}b_{-3}\ldots b_{-m}$$

  - *nie trzeba zapisywać wiodącej „1” („bit ukryty”)*

- *moduł mnożnika zdenormalizowanego* ma postać $0,b_{-1}b_{-2}b_{-3}\ldots b_{-m}$
  - *nie trzeba zapisywać wiodącego „0” („bit ukryty”)*

*Kodowanie wykładnika – (*k *– liczba bitów,* e *– kod,* E *– wartość wykładnika)*

- kod spolaryzowany "$+2^{k-1}-1$" ($e_{\min}=00\ldots01_2$, $e_{\max}=11\ldots10_2$)
- liczba zdenormalizowana – $e=00\ldots00_2$, $E=E_{\min}$
- nieskończoności i nie-liczby (*NaN*) – $e=11\ldots11_2$
- zakres $E_{\min}=-(2^{k-1}-2)$, $E_{\max}=2^{k-1}-1$ (asymetria dodatnia)

standard IEEE 754-2008 – arytmetyka dwójkowa i dziesiętna *zastąpił*:
- IEEE 754 (1985) – arytmetyka dwójkowa
- IEEE 854 (1987) – arytmetyka w dowolnej podstawie