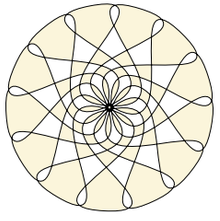
WSTĘP DO INFORMATYKI
I ROK MATEMATYKI

Część 8

Przetwarzanie liczb



Marta Tyran-Kamińska

Systemy pozycyjne

Przykład 1. Weźmy liczbę 352. Co ten napis reprezentuje?
Nauczono nas, że jest to

$$352 = 3 * 100 + 5 * 10 + 2 * 1$$

lub z użyciem potęg

$$352 = 3 * 10^2 + 5 * 10^1 + 2 * 10^0.$$

- **Wartość reprezentowana przez cyfrę w liczbie zależy od jej pozycji w tej liczbie.**
- **Z każdą pozycją związana jest waga.**

Przykład 1 (ciąg dalszy)**.** Cyfra 2 jest na pozycji z wagą $1 = 10^0$ (pozycja 0), cyfra 5 na pozycji z wagą $10 = 10^1$ (pozycja 1), a 3 na pozycji z wagą $100 = 10^2$ (pozycja 2).

- **Używamy dziesięciu cyfr:** $0, 1, \ldots, 9$ **i dlatego mówimy o systemie dziesiętnym (dziesiątkowym).**

Zapis liczb

- ***Systemy liczbowe***
  - System pozycyjny: dziesiętny, dwójkowy, szesnastkowy
  - Konwersja liczb dziesiętnych: część całkowita i część ułamkowa
  - Konwersja z systemu pozycyjnego na dziesiętny, algorytm Hornera
  - Konwersja liczb dwójkowych na szesnastkowe i odwrotnie
  - Arytmetyka binarna
- ***Kodowanie liczb całkowitych:*** *ZM, U1, U2, BIAS*
- ***Zapis zmiennoprzecinkowy (zmiennopozycyjny):*** *notacja naukowa, działania arytmetyczne, standard IEEE-754*

Systemy pozycyjne

- **Pozycyjne systemy liczbowe budujemy według schematu:**
  d **- podstawa**
  c_i $(0 \leqslant c_i < d)$ **- ustalone cyfry systemu liczbowego**
- **Ogólny zapis**

$$c_{n-1}c_{n-2}\ldots c_0,c_{-1}\ldots c_{-m(d)}$$

- **Konwersja na system dziesiętny**

$$c_{n-1}d^{n-1}+c_{n-2}d^{n-2}+\ldots+c_0d^0+c_{-1}d^{-1}+\ldots+c_{-m}d^{-m}$$

System liczbowy	Podstawa	Ustalone cyfry
system dwójkowy	2	0,1
system trójkowy	3	0,1,2
system ósemkowy	8	0,1,2,3,4,5,6,7
system dziesiętny	10	0,1,2,3,4,5,6,7,8,9
system szesnastkowy (heksadecymalne)	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F (litery A-F to liczby 10-15)

Systemy pozycyjne

Dziesiętny	Dwójkowy	Trójkowy	Ósemkowy	Szesnastkowy
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	10	3	3
4	100	11	4	4
5	101	12	5	5
6	110	20	6	6
7	111	21	7	7
8	1000	22	10	8
9	1001	100	11	9
10	1010	101	12	A
11	1011	102	13	B
12	1100	110	14	C
13	1101	111	15	D
14	1110	112	16	E
15	1111	120	17	F
16	10000	121	20	10
17	10001	122	21	11
18	10010	200	22	12
19	10011	201	23	13
20	10100	202	24	14

Konwersja na liczby dziesiętne

Przykład 2. Zamienimy liczbę $101011_{(2)}$ na liczbę w systemie dziesiętnym

Cyfry dwójkowe	1	0	1	0	1	1
Wagi	2^5	2^4	2^3	2^2	2^1	2^0
Wartości wag	32	16	8	4	2	1

Mamy

$$101011_{(2)} = 32+8+2+1 = 43$$

Przykład 3. Konwersja z systemu szesnastkowego liczby $1AE4B_{(16)}$

Cyfry dwójkowe	1	A	E	4	B
Wartości cyfr	1	10	14	4	11
Wagi	16^4	16^3	16^2	16^1	16^0
Wartości wag	65 536	4 096	256	16	1

$$1AE4B_{(16)} = 65536+10*4096+14*256+4*16+11 = 110\,155$$

Konwersja części ułamkowych

Przykład 4. A teraz liczbę $0{,}10011_{(2)}$

Cyfry dwójkowe	1	0	0	1	1
Wagi	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
Wartości wag	0,5	0,25	0,125	0,0625	0,03125

Po zamianie wag na ułamki dziesiętne dodajemy te ułamki, zatem wartość dziesiętna liczby wynosi

$$0,10011_{(2)} = 0{,}5 + 0{,}0625 + 0{,}03125 = 0{,}59375$$

Można też dodać ułamki zwykłe

$$0,10011_{(2)} = \frac{1}{2} + \frac{1}{16} + \frac{1}{32} = \frac{19}{32}$$

a następnie wykonać dzielenie $19 : 32$.

Algorytm Hornera

Dany jest wielomian stopnia $n-1$ o współczynnikach $c_0, c_1, \ldots, c_{n-1}$. Wartość tego wielomianu w punkcie x_0

$$c_{n-1} * x_0^{n-1} + c_{n-2} * x_0^{n-2} \ldots + c_2 * x_0^2 + c_1 * x_0 + c_0$$

wyznaczamy w kolejnych krokach:

c_{n-1}
$c_{n-1} * x_0 + c_{n-2}$
$(c_{n-1} * x_0 + c_{n-2}) * x_0 + c_{n-3}$
$((\ldots((c_{n-1} * x_0 + c_{n-2}) * x_0 + c_{n-3}) * x_0 + \ldots + c_1) * x_0 + c_0)$

Przykład 5. Algorytm Hornera można stosować dla $x_0 = d$ lub $x_0 = d^{-1}$.
Wykorzystamy algorytm Hornera do konwersji liczby $101011_{(2)}$
1
$1 * 2 + 0 = 2$
$2 * 2 + 1 = 5$
$5 * 2 + 0 = 10$
$10 * 2 + 1 = 21$
$21 * 2 + 1 = 43$

Algorytm Hornera

Dany jest wielomian stopnia $n-1$ o współczynnikach $c_0, c_1, \ldots, c_{n-1}$. Wartość tego wielomianu w punkcie x_0

$$c_{n-1} * x_0^{n-1} + c_{n-2} * x_0^{n-2} \ldots + c_2 * x_0^2 + c_1 * x_0 + c_0$$

wyznaczamy w kolejnych krokach:

c_{n-1}

$c_{n-1} * x_0 + c_{n-2}$

$(c_{n-1} * x_0 + c_{n-2}) * x_0 + c_{n-3}$

$((\ldots((c_{n-1} * x_0 + c_{n-2}) * x_0 + c_{n-2}) * x_0 + \ldots + c_1) * x_0 + c_0)$

Przykład 6. Algorytm Hornera można stosować dla $x_0 = d$ lub $x_0 = d^{-1}$.
Wykorzystamy algorytm Hornera do konwersji liczby $101011_{(2)}$

	1	0	1	0	1	1
2		2	4	10	20	42
	1	2	5	10	21	43

Konwersja liczb dziesiętnych

Twierdzenie o dzieleniu z resztą. *Jeżeli* $a, d \in \mathbb{Z}$, $d \neq 0$, *to istnieje dokładnie jedna para liczb całkowitych* k, r *spełniająca warunki:*

$$a = k * d + r, \quad 0 \leqslant r < |d|.$$

Przy czym $d|a \iff r = 0$.

Algorytm dla liczb całkowitych:

- **dzielimy liczbę dziesiętną przez d i zapamiętujemy resztę z dzielenia**
- **algorytm kończy sie, gdy uzyskamy dzielną równą zero**
- **zapisujemy reszty z dzielenia w odwrotnej kolejności**

Możemy posłużyć się tabelką:

Kolejna dzielna	**Reszta z dzielenia przez d**

Konwersja liczb dziesiętnych

Przykład 7. Zamiana liczby dziesiętnej 43 na postać dwójkową

Kolejna dzielna	Reszta z dzielenia przez 2
43	
21	1
10	1
5	0
2	1
1	0
0	1

$43_{(10)} = 101011_{(2)}$

Zamiana tej samej liczby na postać trójkową

Kolejna dzielna	Reszta z dzielenia przez 3
43	
14	1
4	2
1	1
0	1

$43_{(10)} = 1121_{(3)}$

Konwersja ułamków dziesiętnych

Algorytm dla liczb ułamkowych:

- **mnożymy część ułamkową przez d i wydzielamy cyfry przed przecinkiem**
- **algorytm może się nie zakończyć.**

Przykład 8. Zamiana 0,63 na postać dwójkową:

Kolejna mnożna	Cześć całkowita z mnożenia przez 2
0,63	0,
(1),26	1
(0),52	0
(1),04	1
(0),08	0
(0),16	0
(0),32	0
(0),64	0
(1),28	1
(0),56	0

$$0{,}63_{(10)} = 0,101000010._{(2)}$$

System dwójkowy i szesnastkowy

Binarnie	Heksadecymalnie
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Przykład 9. Zamieniamy liczbę z systemu szesnastkowego na system dwójkowy zamieniając każdą cyfrę na zapis jak w tabeli
Dla liczby $A4CD_{(16)}$ mamy:
$A4CD_{(16)}$=1010 0100 1100 1101

Przykład 10. Zamieniamy liczbę z systemu dwójkowego na system szesnastkowy: uzupełniamy zerami, aby długość liczby była wielokrotnością 4, a 4 cyfry dwójkowe mogą być zastąpione 1 cyfrą szesnastkową.
Dla liczby $110101_{(2)}$ mamy:
$110101_{(2)}$=0011 0101 =$35_{(16)}$

Przydatność systemu szesnastkowego: pozwala zapisać w zwięzłej postaci długie ciągi bitów.

Arytmetyka binarna

Dodawanie

- 0+0=0
- 0+1=1
- 1+0=1
- 1+1=0 z przeniesieniem
- przeniesienie +0+0=1
- przeniesienie +0+1=0 z przeniesieniem
- przeniesienie +1+0=0 z przeniesieniem
- przeniesienie +1+1=1 z przeniesieniem

Przykład 11.

```
          p       p       p
 0101    0101    0101    0101    0101
+0011   +0011   +0011   +0011   +0011
-----   -----   -----   -----   -----
    0       0      00     000    1000
```

Arytmetyka binarna

Odejmowanie

- 0-0=0
- 0-1=1 z pożyczką
- 1-0=1
- 1-1=0
- 0-0-pożyczka = 1 z pożyczką
- 0-1-pożyczka = 0 z pożyczką
- 1-0-pożyczka = 0
- 1-1-pożyczka = 1 z pożyczką

```
 0101    0101     0101     0101     0101     0101
-0011   -0011    -0011    -0011    -0011    -0011
-----   -----    -----      p      -----    -----
             0       10    -----      010     0010
                             10
```

Arytmetyka binarna

Mnożenie

- 0*0=0
- 0*1=0
- 1*0=0
- 1*1=1

```
      0101
     *1010
     -----
      0000
     0101
    0000
   0101
---------
  0110010
```

Dzielenie

```
 1001
------
11011:11
11
-----
 00
 00
 ----
  01
  00
  ----
   11
   11
   ----
    0
```

Liczby ze znakiem

Jak zakodować za pomocą bitów liczbę ujemną (całkowitą)?

Różne sposoby

- Kod znak-moduł (*sign-magnitude*)
- Kod uzupełnień do 1 (*one's complement*)
- Kod uzupełnień do 2 (*two's complement*)

Własności tych kodów:

- Liczby dodatnie w tych kodach mają zawsze taki sam kod.
- Kody te różnią się zapisem liczb ujemnych.
- Należy z góry ustalić długość kodu n – liczbę bitów.

Na $n-1$ bitach można zakodować liczby naturalne z zakresu: od 0 do $2^{n-1}-1$.

Kod znak-moduł (ZM)

- **Najbardziej znaczący bit zamiera znak liczby: 0 – liczba dodatnia, 1 – liczba ujemna**
- **Na pozostałych $n-1$ bitach kodujemy wartość bezwzględną liczby.**
- **Liczba 2^{n-1} ma postać dwójkową złożoną z 1 i $n-1$ zer; zera zamieniamy na bity występujące w module.**

  Formalnie regułę można zapisać:

$$x_{(\mathrm{ZM})} = \begin{cases} x & \textbf{dla } x \geqslant 0 \\ 2^{n-1} - x & \textbf{dla } x \leqslant 0. \end{cases}$$

Przykład 12. Liczba $11_{(10)} = 1011_{(2)}$ w 5-cio bitowym kodzie ZM jest postaci $01011_{(\mathrm{ZM})}$ a liczba $(-11)_{(10)} = 11011_{(\mathrm{ZM})}$

$n = 16$	$d = 2$	$d = 16$	$d = 10$
Zero	**0000 0000 0000 0000**	**0000**	$+0$
	1000 0000 0000 0000	**8000**	-0
Liczba największa	**0111 1111 1111 1111**	**7FFF**	$2^{15} - 1$
Liczba najmniejsza	**1111 1111 1111 1111**	**FFFF**	$-(2^{15} - 1)$

Kod uzupełnień do 1 (U1)

- **Do liczby dodatniej dokładamy bit zero**
- **Liczbę ujemną kodujemy poprzez zanegowanie wszystkich bitów z modułu, czyli zamieniamy 0 na 1, 1 na 0 – uzupełnienie do 1**
- **Liczba $2^n - 1$ ma postać dwójkową złożoną z samych jedynek, $(2^n - 1) - |x|$ odejmowanie zamienia bity na przeciwne. Formalnie regułę można zapisać:**

$$x_{(\mathrm{U1})} = \begin{cases} x & \textbf{dla } x \geqslant 0 \\ (2^n - 1) + x & \textbf{dla } x \leqslant 0. \end{cases}$$

Przykład 13. Liczba $11_{(10)} = 1011_{(2)}$ w 5-cio bitowym kodzie U1jest postaci $01011_{(\mathrm{U1})}$ a liczba $(-11)_{(10)} = 10100_{(\mathrm{U1})}$

$n = 16$	$d = 2$	$d = 16$	$d = 10$
Zero	**0000 0000 0000 0000** **1111 1111 1111 1111**	**0000** **FFFF**	$+0$ -0
Liczba największa	**0111 1111 1111 1111**	**7FFF**	$2^{15} - 1$
Liczba najmniejsza	**1000 0000 0000 0000**	**8000**	$-(2^{15} - 1)$

Kod uzupełnień do 2 (U2)

- **Liczbę ujemną kodujemy jako uzupełnienie dwójkowe modułu: negujemy bity moduły, dodajemy 1**
- **Liczba $2^n - 1$ ma postać dwójkową złożoną z samych jedynek, a odejmowanie zamienia bity na przeciwne. Formalnie regułę można zapisać:**

$$x_{(\mathrm{U2})} = \begin{cases} x & \textbf{dla } x \geqslant 0 \\ 2^n + x & \textbf{dla } x < 0. \end{cases}$$

- **Dla liczb ujemnych mamy**

$$x_{(\mathrm{U2})} = x_{(\mathrm{U1})} + 1$$

Przykład 14. Liczba $11_{(10)} = 1011_{(2)}$ w 5-cio bitowym kodzie U2 jest postaci $01011_{(\mathrm{U2})}$, a dla liczby $(-11)_{(10)}$ mamy $10100_{(\mathrm{U1})} + 00001 = 10101_{(\mathrm{U2})}$

$n = 16$	$d = 2$	$d = 16$	$d = 10$
Zero	**0000 0000 0000 0000**	**0000**	$+0$
Liczba największa	**0111 1111 1111 1111**	**7FFF**	$2^{15} - 1$
Liczba najmniejsza	**1000 0000 0000 0000**	**8000**	-2^{15}

Kod przesunięty (BIAS)

- **W kodzie przesuniętym $+k$ można zakodować k liczb ujemnych przyjmując definicję**

$$x_{(+k)} = x + k$$

- **Zakres przedstawianych liczb może nie być symetryczny; zbliżony do symetrycznego, gdy przyjmiemy $k = 2^{n-1}$.**

Porównanie kodów

Kod	Wartość:	ZM	U1	U2	+5	+8
0000	0	0	0	0	-5	-8
0001	1	1	1	1	-4	-7
0010	2	2	2	2	-3	-6
0011	3	3	3	3	-2	-5
0100	4	4	4	4	-1	-4
0101	5	5	5	5	0	-3
0110	6	6	6	6	1	-2
0111	7	7	7	7	2	-1
1000	8	-0	-7	-8	3	0
1001	9	-1	-6	-7	4	1
1010	10	-2	-5	-6	5	2
1011	11	-3	-4	-5	6	3
1100	12	-4	-3	-4	7	4
1101	13	-5	-2	-3	8	5
1110	14	-6	-1	-2	9	6
1111	15	-7	-0	-1	10	7

Liczby rzeczywiste

- **zapis stałoprzecinkowy - przecinek stoi na ustalonym miejscu**
- **zapis zmiennoprzecinkowy (ang. *floating point*) - przecinek może być przesuwany ze zmianą wykładnika**
- **Różne liczby rzeczywiste:**
  $9,11 \times 10^{-31}$ **kg - masa elektronu**
  $5,98 \times 10^{24}$ **kg - masa Ziemi**
- **Postać znormalizowana liczby w systemie o podstawie** d

$$\pm 0, b_1 b_2 \ldots * d^E, \text{ gdzie } b_1 \neq 0$$

$$\pm M \times d^E$$

  Dla masy elektronu mamy
  M **- mantysa, 0,911**
  d **- podstawa systemu, 10**
  E **- wykładnik, -30**
- **Notacja naukowa liczby:**
  zapisujemy jako $9,11E-31$

Przykład 15. Rozważmy liczbę

$$1 - (333,4 - 333,3)^2 * 100$$

Wyznaczając wartość tej liczby widzimy, że

$$1 - (0,1)^2 * 100 = 1 - \frac{1}{100} * 100 = 0$$

Wynik w Excelu:

$$6,82121E - 13$$

Arytmetyka zmiennoprzecinkowa

- Działania na liczbach zmiennoprzecinkowych wykonuje się na mantysach i na wykładnikach.
- Dodawanie i odejmowanie: wyrównujemy wykładniki obu liczb, trzeba więc przesunąć mantysę jednej z liczb (dokonać jej denormalizacji)
- Wyrównanie zawsze do większego wykładnika – mantysa mniejszej liczby jest przesuwana w prawo z możliwą utratą dokładności
- Mnożenie: mnożymy mantysy i dodajemy wykładniki.
- Dzielenie: dzielenie mantys i odejmowanie wykładników.

Arytmetyka zmiennoprzecinkowa

Własności arytmetyki zmiennoprzecinkowej

- **Ustala się długość liczby cyfr mantysy i wykładnika**
- **Brak łączności działań: kolejność wykonywanych działań ma wpływ na wynik**
- **Brak też rozdzielności mnożenia:**
- **Występują zatem**
  zaokrąglenia
  nieprawidłowe operacje
  przepełnienie
  niedomiar

  Przykład 16. Dla długości 4 cyfr mantysy mamy

  $0{,}3176 * 10^3 * 0{,}2504 * 10^5 = 0,07952704 * 10^8 = 0,7953 * 10^7$

  Przy zapisie stałoprzecinkowym byłoby

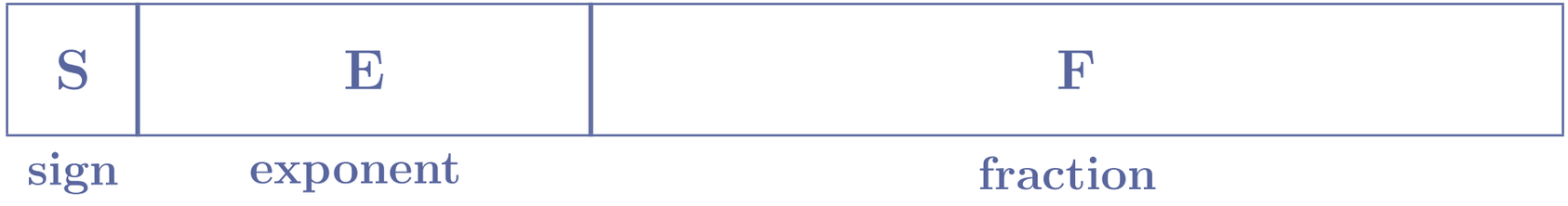
  $317{,}6 * 25040 = 7\,952\,704$

Standard IEEE-754

- Pełna nazwa - 754 IEEE Standard for Binary Floating-Point Arithmetic - ustanowiony w 1985 (*IEEE - Institute of Electrical and Electronics Engineers*)
- Dwa formaty podstawowe:
  - **z pojedynczą precyzją (single) - 32-bitowy**
  - **z podwójną precyzją (double) - 64-bitowy**
- Słowo binarne może przedstawiać:
  - **liczbę znormalizowaną**
  - **liczbę nieznormalizowaną**
  - **zero**
  - **nieskończoność**
  - **tak zwane nie liczby NaN (not-a-number)**

Standard IEEE-754

- Sposób kodowania:
  - **znak - zawsze 1 bit: 0 - liczba dodatnia, 1 - liczba ujemna**
  - **wykładnik - kod dwójkowy z przesunięciem**
  - **ułamek - kod dwójkowy**



	S	E	F		
	Znak	**Wykładnik**	**Ułamek**		**BIAS**
single	**1 [31]**	**8 [30-23]**	**23 [22-0]**	**32**	**127**
double	**1 [63]**	**11 [62-52]**	**52 [51-0]**	**64**	**1023**

- Interpretacja słowa zależy od wartości pól E i F.
  - **Skrajne wartości** $E = E_m = 00..00$ **i** $E = E_M = 11..11$ **zarezerwowane.**
  - **Gdy** $E_m < E < E_M$ **to wartość=** $(-1)^S * 2^{E-BIAS} * 1, F$

Standard IEEE-754

S	E	F	Wartość
0	00..00	00..00	$+0$
0	00..00	00..01	dodatnie
	⋮	⋮	nieznormalizowane
0	00..00	11..11	$2^{(-BIAS+1)} * 0.F$
0	00..01	XX..XX	dodatnie
	⋮	⋮	znormalizowane
0	11..10	XX..XX	$2^{(E-BIAS)} * 1.F$
0	11..11	00..00	$+\infty$
0	11..11	00..01	nieliczby
0	⋮	⋮	aktywne
0	11..11	01..11	SNaN
0	11..11	10..00	nieliczby
0	⋮	⋮	pasywne
0	11..11	11..11	QNaN

S	E	F	Wartość
1	00..00	00..00	-0
1	00..00	00..01	ujemne
	⋮	⋮	nieznormalizowane
1	00..00	11..11	$-2^{(-BIAS+1)} * 0.F$
1	00..01	XX..XX	ujemne
	⋮	⋮	znormalizowane
1	11..10	XX..XX	$-2^{(E-BIAS)} * 1.F$
1	11..11	00..00	$-\infty$
1	11..11	00..01	nieliczby
1	⋮	⋮	aktywne
1	11..11	01..11	SNaN
1	11..11	10..00	nieliczby
1	⋮	⋮	pasywne
1	11..11	11..11	QNaN

- **Efektywny zakres:**

	nieznormalizowane	znormalizowane	dziesiętnie
32	$\pm 2^{-149}$ **do** $(1 - 2^{-23}) * 2^{-126}$	$\pm 2^{-126}$ **do** $(2 - 2^{-23}) * 2^{127}$	$\pm \sim 10^{-44,85}$ **do** $10^{38,53}$
64	$\pm 2^{-1074}$ **do** $(1 - 2^{-52}) * 2^{-1022}$	$\pm 2^{-1022}$ **do** $\pm(2 - 2^{-52}) * 2^{1023}$	$\pm \sim 10^{-323,3}$ **do** $10^{308,25}$

Standard IEEE-754

- **Np. w 32 bitowym formacie IEEE-754**
  - najmniejsza dodatnia liczba znormalizowana $(+L_m)$ ma postać binarną

    $$00000000100000000000000000000000$$

    i wartość:

    $$2^{(1-127)} = 2^{-126} \approx 1,2 * 10^{-38}$$

  - największa dodatnia liczba znormalizowana $(+L_M)$ ma postać binarną

    $$01111111011111111111111111111111$$

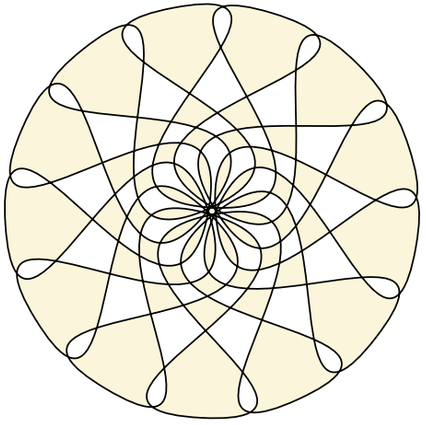
    i wartość:

    $$2^{(254-127)} * (2 - 2^{-23}) = 2^{128} - 2^{104} \approx 3,4 * 10^{38}$$

  - Przykład: zapiszemy liczbę $x = -118,625_{(10)}$ w 32-bitowym IEEE-754.
    $118_{(10)} = 1110110$; $0,625_{(10)} = 0,101_{(2)}$; $1110110,101 = 1,110110101 * 2^6$;
    kodujemy wykładnik w kodzie BIAS=127: $6 + 127 = 133_{(10)} = 10000101_{(2)}$

S	E	F
1	10000101	11011010100000000000

Koniec



Narzędzia: LaTeX, `wrapfig`, `texpower`
Rysunki: *Inkscape*