Arytmetyka dwójkowa

Konwersja liczb w kodach naturalnych

Kod liczbowy naturalny

$$L(A_p) = a_{n-1}p^{n-1} + a_{n-2}p^{n-2} + \ldots + a_1 p^1 + a_0 p^0 + a_{-1}p^{-1} + a_{-2}p^{-2} + \ldots$$

$$A_{10} = 245{,}67 = 2\cdot 10^2 + 4\cdot 10^1 + 5\cdot 10^0 + 6\cdot 10^{-1} + 7\cdot 10^{-2} \qquad ND$$

$$A_2 = 101.101 = 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-3} \qquad NB$$

Uzupełnienie liczb do podstawy ***p*** i długości części całkowitej ***n***

$$Up(L) = p^n - L \quad dla \quad L > 0$$

$$U10(3480) = 10^4 - 3480 = 6520$$

$$U10(0{,}4947) = 1 - 0{,}4947 = 0{,}5053$$

$$U2(10110) = 01010$$

$$U2(01.1101 = 10.0011$$

<u>Reguła praktyczna</u> Uzupełnienie U2 liczby nieujemnej binarnej otrzymuje się przez:

• pozostawienie wszystkich mniej znaczących zer bez zmiany i pierwszej najmniej znaczącej jedynki

• a następnie negację wszystkich pozostałych bitów

Zapis liczb dwójkowych w kodzie ZM i ZU2

Znak wprowadzany jest w postaci odrębnego bitu 1 oznacza (-) minus

0 oznacza (+) plus

W **kodzie ZM** liczby dodatnie i ujemne o tych samych wartościach bezwzględnych różnią się tylko bitem znaku

$$+12_{10} \mapsto 0.1100_2 \qquad +0{,}75_{10} \mapsto 0.1100_2$$

$$-12_{10} \mapsto 1.1100_2 \qquad -0{,}75_{10} \mapsto 1.1100_2$$

W **kodzie ZU2** liczby dodatnie zapisywane są jak liczby w kodzie ZM
liczby ujemne reprezentowane są przez bit znaku równy 1i przez uzupełnienie modułu do 2

$$+12_{10} \mapsto 0.1100_2 \qquad +0{,}75_{10} \mapsto 0.1100_2$$

$$-12_{10} \mapsto 1.0100_2 \qquad -0{,}75_{10} \mapsto 1.0100_2$$

Arytmetyka na liczbach dwójkowych bez znaku

Dodawanie

Reguły dodawania

Argumenty			Wynik	
c_i	p_i	q_i	f_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\begin{array}{lrl} P = & 1011 & L(P)=11 \\ Q = & 1101 & L(Q)=13 \\ \hline Y = & 11000 & L(Y)=24 \end{array}$$

Odejmowanie

$$\begin{array}{r} 54 \\ -30 \\ \hline 24 \end{array} \qquad \begin{array}{r} 54 \\ +U(30) \end{array} \qquad \begin{array}{r} 54 \\ +70 \\ \hline (1)24 \end{array} \qquad \begin{array}{r} 110110 \\ +100010 \\ \hline (1)011000 \end{array} \mapsto +24$$

Mnożenie

$$\begin{array}{lr} L(P) & 23 \\ L(Q) & \times\ 11 \\ \hline & 253 \end{array} \qquad \begin{array}{r} 10111 \\ \times\ 1011 \\ \hline 10111 \\ +10111\ \\ \hline 1000101 \\ +10111\ \ \ \\ \hline 11111101 \end{array} \mapsto 253$$

Dzielenie

$$\begin{array}{l} \ \ 1000100.11 \\ \hline 101011000 \end{array} \div 101 \qquad \begin{array}{l} \mapsto 68{,}75 \\ 344 \div 5 = 68{,}8 \end{array}$$

$$\begin{array}{r} 011\ \ \ \ \ \ \ \\ \hline (1)0000110 \\ 011\ \ \\ \hline (1)001000 \\ 011\ \\ \hline (1)0110 \\ 011 \\ \hline (1)001 \end{array}$$

Dodawanie i odejmowanie na liczbach ze znakiem

Działania w kodzie ZU2

$$\begin{array}{r} +11 \\ \underline{-36} \\ -25 \end{array} \quad \begin{array}{l} ZU2(+11) \\ ZU2(+36) = 0.100100 \end{array} \quad ZU2(-36) \mapsto \begin{array}{r} 0.001011 \\ \underline{+1.011100} \\ 1.100111 \\ \overleftarrow{0}\ \overleftarrow{0}\qquad\quad \end{array}$$

$$\begin{array}{r} +36 \\ \underline{-11} \\ +25 \end{array} \quad \begin{array}{l} ZU2(+36) \\ ZU2(+11) = 0.001011 \end{array} \quad ZU2(-11) \mapsto \begin{array}{r} 0.100100 \\ \underline{+1.110101} \\ 10.011001 \\ \overleftarrow{1}\ \overleftarrow{1}\qquad\quad \end{array}$$

$$\begin{array}{r} +68 \\ \underline{+102} \\ +170 \end{array} \quad \begin{array}{r} 0.1000100 \\ \underline{0.1100110} \\ 1.0101010 \\ \overleftarrow{0}\ \overleftarrow{1}\qquad\quad \end{array} \quad \mapsto ZU2(-86)\ ???$$

$$c_Z = 0 \quad c_m = 1 \qquad\qquad c_Z \oplus c_m = 1$$

Nadmiar powodujący błąd gdy

mnożenie liczb ZM

$Xz \oplus Yz = 1 \quad \textit{liczba ujemna}$

mnożna przepisywana jest
pod jedynki mnożnika

ExOR dla
bitów znaku

$$\frac{13}{16} \cdot \frac{9}{16} = 0{,}457031$$

X	-	1	1	0	1	X= - 13
Y	+	1	0	0	1	Y=+9

1. 1 1 0 1
0. 1 0 0 1
1 1 0 1
1 1 0 1
1. 0 1 1 1 0 1 0 1

w przód

1. 1 1 0 1
0. 1 0 0 1
1 1 0 1
1 1 0 1
1. 0 1 1 1 0 1 0 1

wstecz

```
 1.   1 1 0 1
 0.   1 0 1 1
      ---------
       1 1 0 1
           1 1 0 1
---------------------
    1 0 0 0 0 0 1
             1 1 0 1
   ---------------------
1.  1 0 0 0 1 1 1 1
```

$$\frac{13}{16}\cdot\frac{11}{16}=\frac{143}{256}$$

sprawdzenie

$$x \bullet y = \frac{128+8+4+2+1}{16 \bullet 16} = \frac{143}{256}$$

mnożenie liczb ZU2

X - 1 1 0 1

Y + 1 0 0 1

metoda Bootha

	X^d	1.	0	0	1	1				
	Y^d	0.	1	0	0	1	(0)			
$y_4y_5=10$		0.	1	1	0	1				
		0.	0	1	1	0	1			
$y_3y_4=01$		1.	0	0	1	1				
		1.	1	0	0	1	1			
		1.	1	1	0	0	1	1		
$y_2y_3=00$		1.	1	1	1	0	0	1	1	
$y_1y_2=10$		0.	1	1	0	1				
		0.	1	0	1	1	0	1	1	
		0.	0	1	0	1	1	0	1	1
$y_0y_1=01$		1.	0	0	1	1				
		1.	1	0	0	0	1	0	1	1
		-.	0	1	1	1	0	1	0	1

$y_ny_{n+1}=0\ 1$ ▶ $+X^d$ →

$y_ny_{n+1}=1\ 0$ ▶ $-X^d$ →

$y_ny_{n+1}=0\ 0$ ▶ →

$y_ny_{n+1}=1\ 1$ ▶ →

Mnożenie
(+6)*(+4)

$y_n y_{n+1}$=0 1 ▶ $+X^d$ →

$y_n y_{n+1}$=1 0 ▶ $-X^d$ →

$y_n y_{n+1}$=0 0 ▶ →

$y_n y_{n+1}$=1 1 ▶ →

	X	+0 1 1 0
	Y	+0 1 0 0
	X^d	0.0 1 1 0
	Y^d	0.0 1 0 0 (0)
y_4y_5=00		0
y_3y_4=00		0 0
y_2y_3=10		1.1 0 1 0 0 0
		1.1 1 0 1 0 0 0
y_1y_2=01		0.0 1 1 0
		0.0 0 1 1 0 0 0
		0.0 0 0 1 1 0 0 0
y_0y_1=00		brak przesunięcia wynik +24

Mnożenie
(+6)*(-4)

$y_n y_{n+1}$=0 1 ► $+X^d$ →

$y_n y_{n+1}$=1 0 ► $-X^d$ →

$y_n y_{n+1}$=0 0 ► →

$y_n y_{n+1}$=1 1 ► →

X +0 1 1 0
Y - 0 1 0 0

X^d 0.0 1 1 0
Y^d 1.1 1 0 0 (0)

y_4y_5=00	0
y_3y_4=00	0 0
y_2y_3=10	1.1 0 1 0 0 0
	1.1 1 0 1 0 0 0
y_1y_2=11	1.1 1 1 0 1 0 0 0
y_0y_1=11	brak przesunięcia =-24

Mnożenie
(+6)*(-6)

$y_n y_{n+1}=0\ 1$ ▶ $+X^d$ →

$y_n y_{n+1}=1\ 0$ ▶ $-X^d$ →

$y_n y_{n+1}=0\ 0$ ▶ →

$y_n y_{n+1}=1\ 1$ ▶ →

X +0 1 1 0
Y - 0 1 1 0

X^d 0.0 1 1 0
Y^d <u>1.1 0 1 0 (0)</u>

$y_4y_5=00$	0
$y_3y_4=10$	1.1 0 1 0 0
	1.1 1 0 1 0 0
$y_2y_3=01$	<u>0.0 1 1 0</u>
	0.0 0 1 1 0 0
	0.0 0 0 1 1 0 0
$y_1y_2=10$	<u>1.1 0 1 0</u>
	1.1 0 1 1 1 0 0
	1.1 1 0 1 1 1 0 0
$y_0y_1=11$	brak operacji =-36

Zmiennoprzecinkowa reprezentacja liczb

W stałoprzecinkowej reprezentacji liczb położenie punktu dziesiętnego (kropki) jest stałe niezależne od wielkości liczb

np. format dziesięciocyfrowy XXXXX.XXXXX

A= 47567,81

B= 0,00007599998 wówczas B= 0,00007

w przypadku mnożenia (A*B) błąd względny wynosi wówczas 7,89%

a dla liczby B=0,00007999998 błąd względny wynosi 12,5%

Należy przeprowadzić skalowanie liczby
A= 47567,81= 0,47567*10^5
B=0,00007999998= 0,799999 *10^-4
błąd względny = 0,0029%

Zmiennoprzecinkowa reprezentacja liczb

Liczbę L w formacie zmiennoprzecinkowym przedstawia się jako

$L=L(M*W) =L(M) *p^{dL(W)}$

gdzie: - <u>słowo M</u>- mantysa, liczba ułamkowa ze znakiem przedstawiona w jednym z trzech kodów ZM, ZU1, ZU2

<u>słowo W</u>- wykładnik (cecha) liczba całkowita ze znakiem przedstawiona w jednym z trzech kodów (niekoniecznie tym samym co M)

p – wspólna podstawa kodów dla słów W i M

d - liczba naturalna zwykle równa 1

Liczba zmiennoprzecinkowa jest znormalizowana kiedy mantysa spełnia warunek

$$p^{-d} \le \left| L(M) \right| < 1$$

Zmiennoprzecinkowa reprezentacja liczb

Przykład Liczbę L = - 4,25 przedstawić w dwójkowym zapisie zmiennoprzecinkowym przyjmując słowa jednobajtowe

ZM(L) = 1.100.01

ZU2(L) = 1.011.11

normalizacja $2^{-1} = 0{,}5 \leq |L(M)| < 1$

$L=L(1.0111100)*2^3$

$M*W= \underbrace{\underline{1.0111100}}_{M}\ \underbrace{0.0000011}_{W}$

Sprawdzenie

$L(M)*p^{L(W)=}$ - $[1-(1/4+1/8+1/16+1/32)]*2^3$= - (1-0,46875)*8=-4,25

Przykład mnożenia dużej i małej liczby

Mnożenie liczb zmiennoprzecinkowych

743,81*0,0007599=0,565221219

10111001111,11   * ,0000 0000 0001 1000 1110=

.1011100111111 *2^(11)   * .1 1000 111*2^(-11)=

Przyjmując, że do zapisu liczby dostępnych jest 8 pól, mantysy przyjmują postać i w celu wykonania mnożenia są one **mnożone**

Natomiast wykładniki są **sumowane** 11+(-11)=0

2^0=1

```
   0, 1 0 1 1 1 0 0 1
   0, 1 1 0 0 0 1 1 1
-------------------------------------
      1 0 1 1 1 0 0 1
+       1 0 1 1 1 0 0 1
-------------------------------------
    1 0 0 0 1 0 1 0 1 1
+               1 0 1 1 1 0 0 1
-------------------------------------
    1 0 0 0 1 1 0 1 1 0 1 0 0 1
+                 1 0 1 1 1 0 0 1
-------------------------------------
    1 0 0 0 1 1 1 1 0 0 0 1 0 1 1
+                   1 0 1 1 1 0 0 1
-------------------------------------
 0, 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1
```

0,	1/2	0	1/8	0	1/32	1/64	1/128	1/256

=0,558594

Dodawanie liczb zmiennoprzecinkowych

W celu dodania liczb zmiennoprzecinkowych należy sprowadzić je do zapisu stałoprzecinkowego i dodać

101 1100 1111 ,1100 0000 0000 0000 0000

+ 000 0000 0000 ,0000 0000 0001 1000 1110

= 101 1100 1111 ,1100 0000 0001 1000 1110

W praktyce wynikiem dodawania dużej i małej liczby **jest duża liczba**

743,81+0,0007599=743,8107599

czyli w zapisie zmiennoprzecinkowym dla powyższego przykładu

znak mantysy

znak wykładnika

$$\underbrace{0{,}10111001}_{\text{mantysa}} * \underbrace{0{,}0000\ 1011}_{\text{wykładnik}}$$

Format IEEE 754

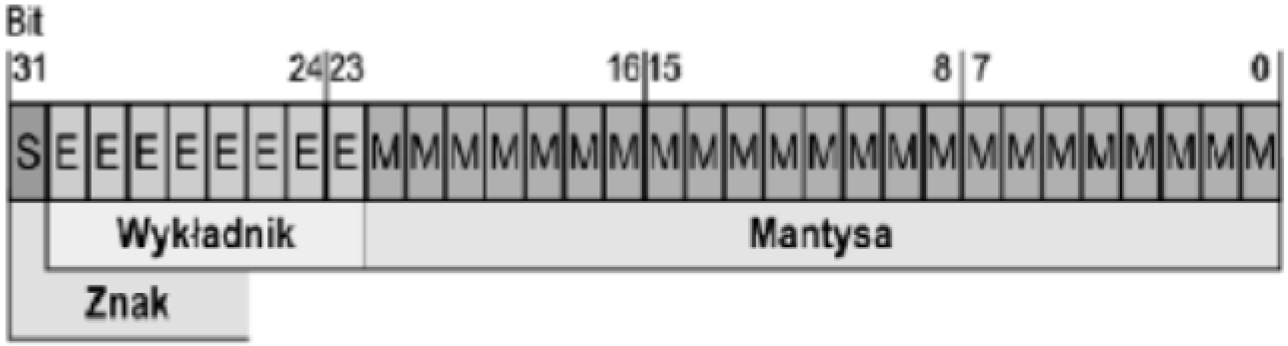
Standard IEEE 754(-1985, -2008) definiuje dwie podstawowe klasy binarnych liczb zmiennoprzecinkowych:
←binary 32 - pojedynczej precyzji (ang. single-precision)
←binary 64 - podwó jnej precyzji (ang. double-precision)

Format	Bit znaku	Bity cechy	Bity mantysy
32 bity - pojedyncza precyzja	1 bit	8 bitów	23 bity
64 bity - podwójna precyzja	1 bit	11 bitów	52 bity

Standard IEEE 754-2008 rozszerza też wersję IEEE 754-1985 o definicje kilku formatów dziesiętnych liczb zmiennoprzecinkowych

Format IEEE 754

Reprezentacja zmiennoprzecinkowa IEEE 754 pojedynczej precyzji



- bit znaku: 0 oznacza liczbę dodatnią, 1 ujemną

- cecha zapisywana jest w kodzie z nadmiarem (dla 8-mio bitowego zapisu nadmiar wynosi 127, zatem w polu cechy można zapisać wartości od -127 do 128)
- mantysa zapisywana jest w stałoprzecinkowym kodzie U1.