Kodowanie liczb

Kodowanie stałopozycyjne liczb całkowitych

Niech liczba całkowita a ma w systemie dwójkowym postać:

$$a = \pm(a_n a_{n-1} \ \dots \ a_1 a_0)_2 .$$

Wtedy może być ona przedstawiona w postaci (n+2)-bitowej przy pomocy trzech niżej zdefiniowanych kodów.

Kod prosty

$$[a]_{pr} = \begin{cases} 0a_n a_{n-1} \ \dots. \ a_1 a_0 , & \text{gdy znakiem liczby jest } + \\ 1a_n a_{n-1} \ \dots. \ a_1 a_0 , & \text{gdy znakiem liczby jest } - \end{cases}$$

Kod odwrotny, 1-kod, kod uzupełniony do 1

$$[a]_{pr} = \begin{cases} 0a_n a_{n-1} \ \dots. \ a_1 a_0 , & \text{gdy znakiem liczby jest } + \\ 1\overline{a}_n \overline{a}_{n-1} \ \dots. \ \overline{a}_1 \overline{a}_0 , & \text{gdy znakiem liczby jest } -, \end{cases}$$

gdzie $\overline{a}_i = 1 - a_i$ dla $i = 0, 1, \dots, n$.

Kod dopełnieniowy, 2-kod, kod uzupełniony do 2

$$[a]_{odw} = \begin{cases} 0a_n a_{n-1} \ \dots. \ a_1 a_0 , & \text{gdy znakiem liczby jest } + \\ 1\overline{a}_n \overline{a}_{n-1} \ \dots. \ \overline{a}_1 \overline{a}_0 + \underbrace{0 \dots 0}_{n+1 \text{ razy}}1, & \text{gdy znakiem liczby jest } -, \end{cases}$$

gdzie $\overline{a}_i = 1 - a_i$ dla $i = 0, 1, \dots, n$.

Uwaga.
Liczba 0 w kodzie prostym i kodzie odwrotnym jest kodowana jako $+0$ lub -0; w systemie dopełnieniowym jest ona kodowana tylko jako $+0$.

Przykłady.
Zakodujemy podane liczby całkowite przy pomocy 7 bitów.

a) $a = +11001_2$.

Mamy $a = +011001_2$, skąd

$$[a]_{pr} = [a]_{odw} = [a]_{dop} = 0011001$$

b) $a = -11001_2$.

Mamy $a = -011001_2$, skąd

$$[a]_{pr} = 1011001;$$

$$[a]_{odw} = 1100110;$$

$$[a]_{dop} = 1100110 + 0000001 = 1100111$$

c) $a = -111111_2$.

Wtedy

$$[a]_{pr} = 1111111;$$

$$[a]_{odw} = 1000000;$$

$$[a]_{dop} = 1000000 + 0000001 = 1000001.$$

d) $a = 0_2$.

Jeżeli przyjmiemy, że $a = +000000_2$, to

$$[a]_{pr} = [a]_{odw} = [a]_{dop} = 0000000\,.$$

W kodzie prostym i w kodzie odwrotnym można założyć, że $a = -000000_2$, i wtedy

$$[a]_{pr} = 1000000\,;$$

$$[a]_{odw} = 1111111\,.$$

<u>*Przykłady*</u>

Rozkodujemy przykładowe liczby, tzn. znajdziemy ich reprezentacje dwójkowe na podstawie wartości ich kodów 7-bitowych.

a) $[a]_{kod} = 0001101 \Rightarrow kod = pr = odw = dop \Rightarrow a = +001101_2 = +1101_2$.

b) $[a]_{kod} = 110111|1$.

$$kod = pr \Rightarrow a = -101111_2\,;$$

$$kod = odw \Rightarrow a = -010000 = -10000_2\,;$$

$$kod = dop \Rightarrow [a]_{odw} = 1101111 - 0000001 = 1101110 \Rightarrow a = -010001_2 = -10001_2\,.$$

c) $[a]_{kod} = 1010|100$

$$kod = pr \Rightarrow a = -10100_2\,;$$

$$kod = odw \Rightarrow a = -0101011_2 = -101011_2\,;$$

$$kod = dop \Rightarrow [a]_{odw} = 1010100 - 0000001 = 1010011 \Rightarrow a = -101100_2\,.$$

d) $[a]_{kod} = 1000000$

$$kod = pr \Rightarrow a = -00000_2 = 0_2\,;$$

$$kod = odw \Rightarrow a = -111111_2\,;$$

$$kod = dop \Rightarrow [a]_{odw} = 1000000 - 0000001 = 0111111 \Rightarrow a = +111111_2 \Rightarrow \text{sprzeczność, bo}$$

$$[+111111]_{dop} = 0111111.$$

A więc zgodnie z przyjętym określeniem kodu dopełnieniowego równość nie jest możliwa!. Dlatego na podstawie dodatkowej umowy przyjmuje się, że $[-1000000_2]_{dop} = 1000000$.

Kodowanie stałopozycyjne a dodawanie i odejmowanie liczb całkowitych

Dodawanie

<u>*Przypadek kodu odwrotnego*</u>

Prześledzimy zagadnienie na reprezentatywnych przykładach liczb kodowanych przy pomocy 8 bitów.

<u>*Przykłady*</u>

a)

$$\begin{array}{rl} a & = +0101101_2 \\ b & = +0100000_2 \\ \hline a+b & = +1001101_2 \\ [a+b]_{odw} & = 0\,1001101 \end{array} \qquad \begin{array}{rl} [a]_{odw} & = 00101101 \\ [b]_{odw} & = 00100000 \\ \hline [a]_{odw} + [b]_{odw} & = 01001101 \end{array}$$

b)

$$\begin{array}{rl} a & =+1101101_2 \\ b & =-1011010_2 \\ \hline a+b & =+0010011_2 \\ [a+b]_{odw} & =0001001\ 1 \end{array}
\qquad
\begin{array}{rr} [a]_{odw} = & 01101101 \\ [b]_{odw} = & 10100101 \\ \hline & \boxed{1}00010010 \\ + & 1 \\ \hline & 00010011 \end{array}$$

c)

$$\begin{array}{rl} a & =+1011010_2 \\ b & =-1101101_2 \\ \hline a+b & =-0010011_2 \\ [a+b]_{odw} & =1110110\ 0 \end{array}
\qquad
\begin{array}{rl} [a]_{odw} & =01011010 \\ [b]_{odw} & =10010010 \\ \hline [a]_{odw}+[b]_{odw} & =11101100 \end{array}$$

d)

$$\begin{array}{rl} a & =-0101101_2 \\ b & =-0100000_2 \\ \hline a+b & =-1001101_2 \\ [a+b]_{odw} & =1011001\ 0 \end{array}
\qquad
\begin{array}{rr} [a]_{odw} = & 11010010 \\ [b]_{odw} = & 11011111 \\ \hline & \boxed{1}10110001 \\ + & 1 \\ \hline & 10110010 \end{array}$$

Wniosek.
Aby otrzymać kod odwrotny sumy dwóch liczb, należy dodać ich kody odwrotne i w przypadku powstania bitu przepełnienia sumę tę powiększyć o 1.

Przypadek kodu dopełnieniowego
Prześledzimy zagadnienie na reprezentatywnych przykładach liczb kodowanych przy pomocy 8 bitów.

Przykłady
a)

$$\begin{array}{rl} a & =+0101101_2 \\ b & =+0100000_2 \\ \hline a+b & =+1001101_2 \\ [a+b]_{dop} & =0100110\ 1 \end{array}
\qquad
\begin{array}{rl} [a]_{dop} & =00101101 \\ [b]_{dop} & =00100000 \\ \hline [a]_{dop}+[b]_{dop} & =01001101 \end{array}$$

b)

$$\begin{array}{rl} a & =+1101101_2 \\ b & =-1011010_2 \\ \hline a+b & =+0010011_2 \\ [a+b]_{dop} & =0001001\ 1 \end{array}
\qquad
\begin{array}{rr} [a]_{dop} = & 01101101 \\ [b]_{dop} = & 10100110 \\ \hline & \boxed{1}00010011 \\ \text{odrzucić!} & \\ \hline & 00010011 \end{array}$$

c)

$$\begin{array}{rl} a & =+1011010_2 \\ b & =-1101101_2 \\ \hline a+b & =-0010011_2 \\ [a+b]_{dop} & =1110110\ 1 \end{array} \qquad \begin{array}{rl} [a]_{dop} & =01011010 \\ [b]_{dop} & =10010011 \\ \hline [a]_{dop}+[b]_{dop} & =11101101 \end{array}$$

d)

$$\begin{array}{rl} a & =-0101101_2 \\ b & =-0100000_2 \\ \hline a+b & =-1001101_2 \\ [a+b]_{dop} & =1011001\ 1 \end{array} \qquad \begin{array}{rl} [a]_{dop} & =11010011 \\ [b]_{dop} & =11100000 \\ \hline & \boxed{1}10110011 \\ \text{odrzucić!} & \\ \hline & 10110011 \end{array}$$

Wniosek.
Aby otrzymać kod dopełnieniowy sumy dwóch liczb, należy dodać ich kody dopełnieniowe i ewentualny bit przepełnienia odrzucić.

Odejmowanie

Z uwagi na równość

$$a-b=a+(-b)$$

odejmowanie może być zastąpione dodawaniem liczby a i liczby przeciwnej do liczby b. Dlatego powstaje problem uzyskiwania kodu liczby przeciwnej z kodu liczby danej. Prześledzimy zagadnienie na reprezentatywnych przykładach liczb kodowanych przy pomocy 8 bitów.

Przypadek kodu odwrotnego

Przykłady

a)

$a=+101101_2 \Rightarrow [a]_{odw}=0101101$

$-a=-101101_2 \Rightarrow [a]_{odw}=1010010=[0101101]_{dop1}$

b)

$a=+010010_2 \Rightarrow [a]_{odw}=0010010$

$-a=-010010_2 \Rightarrow [a]_{odw}=1101101=[0010010\ ]_{dop1}$

c)

$a=-100111_2 \Rightarrow [a]_{odw}=1011000$

$-a=+100111_2 \Rightarrow [a]_{odw}=0100111=[1011000\ ]_{dop1}$

d)

$a=-011000_2 \Rightarrow [a]_{odw}=1100111$

$-a=+011000_2 \Rightarrow [a]_{odw}=0011000=[1100111\ ]_{dop1}$

Wniosek.
Aby otrzymać kod odwrotny liczby przeciwnej należy obliczyć tzw. *dopełnienie do 1* kodu liczby danej.

Przykłady
Oto przykłady operacji, o której mowa w ostatnim wniosku.
a)
$[0011000]_{dop1} = 1100111;$
b)
$[1000000]_{dop1} = 0111111.$

Przypadek kodu dopełnieniowego

Przykłady
a)
$a = +101101_2 \Rightarrow [a]_{dop} = 0101101$
$-a = -101101_2 \Rightarrow [a]_{dop} = 1010011 = [010110|1]_{dop2}$

b)
$a = +010010_2 \Rightarrow [a]_{dop} = 0010010$
$-a = -010010_2 \Rightarrow [a]_{dop} = 1101110 = [00100|10]_{dop2}$

c)
$a = -100111_2 \Rightarrow [a]_{dop} = 1011001$
$-a = +100111_2 \Rightarrow [a]_{dop} = 0100111 = [101100|1]_{dop2}$

d)
$a = -011000_2 \Rightarrow [a]_{dop} = 1101000$
$-a = +011000_2 \Rightarrow [a]_{dop} = 0011000 = [110|1000]_{dop2}$

Wniosek.
Aby otrzymać kod dopełnieniowy liczby przeciwnej należy obliczyć tzw. *dopełnienie do 2* kodu liczby danej.

Przykłady
Oto przykłady operacji, o której mowa w ostatnim wniosku.
a)
$[001|1000]_{dop1} = 1101000$
Inna metoda:
$$\begin{array}{r} 1100111 \\ +1 \\ \hline 1101000 \end{array}$$

b)

$\left[101011|1\right]_{dop2} = 0101001$

Inna metoda:

$$\begin{array}{r} 0\,1\,0\,1\,0\,0\,0 \\ +1 \\ \hline 0\,1\,0\,1\,0\,0\,1 \end{array}$$

c)

$\left[|1000000\right]_{dop2} = 1000000$

Inna metoda:

$$\begin{array}{r} 0\,1\,1\,1\,1\,1\,1 \\ +1 \\ \hline 1\,0\,0\,0\,0\,0\,0 \end{array}$$

Otrzymaliśmy sprzeczność, bo liczba przeciwna do liczby o rozważanym kodzie jest poza zakresem!

<u>Ćwiczenie</u>.

Wykonać poniższe dodawanie i odejmowanie przy pomocy kodu odwrotnego i kodu dopełnieniowego argumentów tych operacji. Sprawdzić otrzymane wyniki przy pomocy tradycyjnego dodawania i tradycyjnego odejmowania.

a) $s = \left(-1001110_2\right) + \left(-0101011_2\right).$

<u>Kod odwrotny</u>

$\left[-1001110_2\right]_{odw} = 10110001$

$\left[-0101011_2\right]_{odw} = 11010100$

$$\Rightarrow \left\{ \begin{array}{r} 1\,0\,1\,1\,0\,0\,0\,1 \\ +1\,1\,0\,1\,0\,1\,0\,0 \\ \hline 1\;1\,0\,0\,0\,0\,1\,0\,1 \\ + \quad +1 \\ \hline 1\,0\,0\,0\,0\,1\,1\,0 \end{array} \right. \Rightarrow s = -1111001_2 .$$

<u>Kod dopełnieniowy</u>

$\left[-1001110_2\right]_{dop} = 10110010$

$\left[-0101011_2\right]_{dop} = 11010101$

$$\Rightarrow \left\{ \begin{array}{r} 1\,0\,1\,1\,0\,0\,1\,0 \\ +1\,1\,0\,1\,0\,1\,0\,1 \\ \hline 1\;1\,0\,0\,0\,0\,1\,1\,1 \\ 1\,0\,0\,0\,0\,1\,1\,1 \end{array} \right. \Rightarrow s = -1111001_2 .$$

Odrzucić!

<u>Sprawdzenie</u>

$$\begin{array}{r} -1\,0\,0\,1\,1\,1\,0_2 \\ -0\,1\,0\,1\,0\,1\,1_2 \\ \hline -1\,1\,1\,1\,0\,0\,1_2 \end{array}$$

b) $r = \left(-1110001_2\right) - \left(-0101100_2\right)$.

Kod odwrotny

$\left[-1110001_2\right]_{odw} = 10001110$

$\left[-0101100_2\right]_{odw} = 11010011 \Rightarrow \left[11010011\right]_{dop1} = 00101100$

$$\Rightarrow \begin{cases} \;\;10001110 \\ +00101100 \\ \hline \;\;10111010 \end{cases} \Rightarrow r = -1000101_2 .$$

Kod dopełnieniowy

$\left[-1110001_2\right]_{dop} = 10001111$

$\left[-0101100_2\right]_{dop} = 11010100 \Rightarrow \left[11010100\right]_{dop2} = 00101100$

$$\Rightarrow \begin{cases} \;\;10001111 \\ +00101100 \\ \hline \;\;10111011 \end{cases} \Rightarrow r = -1000101_2 .$$

Sprawdzenie

$$\begin{array}{l} -1110001_2 \\ +0101100_2 \\ \hline -1000101_2 \end{array}$$

Kodowanie dwójkowo-dziesiętne

Istota tego systemu polega na kodowaniu przy pomocy 4 bitów każdej cyfry zapisu dziesiętnego liczby. Można stworzyć wiele systemów tego typu, ale w praktyce stosuje się kod BCD (Binary Coded Decimal) zwany także kodem 8-4-2-1, kod Aikena zwany także kodem 2-4-2-1 i kod z nadmiarem 3, oznaczany dalej symbolem +3.

Oto definicja tych systemów:

Cyfra dziesiętna	Kod 8-4-2-1	Kod 2-4-2-1	Kod +3
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

Przykład

$\left[18970\right]_{8-4-2-1} = 0001\,|\,1000\,|\,1001\,|\,0111\,|\,0000;$

$\left[18970\right]_{2-4-2-1} = 0001\,|\,1110\,|\,1111\,|\,1101\,|\,0000;$

$\left[18970\right]_{+3} = 0100\,|\,1011\,|\,1100\,|\,1010\,|\,0011.$

Ćwiczenie
Zakodować przy pomocy 16 bitów poniższe liczby dziesiętne:

a) 7925.
Rozwiązanie.
$[7925]_{8-4-2-1} = 0111|1001|0010|0101;$
$[7925]_{2-4-2-1} = 1101|1111|0010|1011;$
$[7925]_{+3} = 1010|1100|0101|1000.$

b) $348 = 0348$.
Rozwiązanie.
$[0348]_{8-4-2-1} = 0000|0011|0100|1000$;
$[0348]_{2-4-2-1} = 0000|0011|0100|1110$;
$[0348]_{+3} = 0011|0110|0111|1011.$

Ćwiczenie
Rozkodować liczby dziesiętne zakodowane przy pomocy 16 bitów:

a) $[x]_{kod} = 0011|0100|1100|1011.$
Rozwiązanie.
$kod = "8-4-2-1" \Rightarrow x = 34(12)(11) \Rightarrow$ sprzeczność!
$kod = "2-4-2-1" \Rightarrow x = 3465$
$kod = "+3" \Rightarrow x = 0198.$

b) $[x]_{kod} = 1001|0111|1000|0011.$
Rozwiązanie.
$kod = "8-4-2-1" \Rightarrow x = 9783$
$kod = "2-4-2-1" \Rightarrow x = 3723 \Rightarrow$ sprzeczność!
$kod = "+3" \Rightarrow x = 6450.$

Kodowanie zmiennopozycyjne liczb rzeczywistych

Jeżeli ustalona jest podstawa systemu liczbowego $p \geq 2$, to każda liczba rzeczywista a może być zapisana przy pomocy tzw. *notacji naukowej* w postaci:

$$a = \pm m \cdot p^c,$$

gdzie współczynnik m nazywa się *mantysą*, a wykładnik c będący liczbą całkowitą – *cechą liczby* a. Zakładać będziemy, że część ułamkowa mantysy ma rozwinięcie skończone, co oznacza, że opisana reprezentacja niekiedy przedstawia liczbą a w sposób przybliżony.

Ponieważ notacja naukowa nie jest jednoznaczna, więc w przypadku, gdy $a \neq 0$, częstą stosuje się tzw. *notację znormalizowaną* polegającą na tym, że m oraz c są tak dobrane, aby

$$m = (0.m_1 m_2 \, .. \, m_k)_p,$$

przy czym $m_1 \neq 0$. Gwarantuje to zachodzenie nierówności

$$\frac{1}{p} \leq m < 1.$$

Innym stosowanym warunkiem normalizacyjnym jest przedstawianie m w postaci

$$m = (1.m_1 m_2 \, .. \, m_k)_p,$$

tak, aby zachodziła nierówność

$$1 \leq m < p.$$

Opisane niżej systemy kodowania liczb rzeczywistych zwane <u>*reprezentacjami zmiennoprzecinkowymi*</u> w sposób istotny wykorzystują opisane notacje naukowe.

<u>*Standard IBM*</u>.
Występuje tu tzw. <u>*reprezentacja krótka*</u> 32-bitowa i <u>*reprezentacja długa*</u> 64-bitowa. Jeżeli kodowana liczba rzeczywista a nie jest zerem, to daje się ona przedstawić w postaci

$$a = (-1)^S \cdot 16^{C-64} \cdot (0.M)_2 ,$$

gdzie parametr S równy 0 lub 1 określa znak liczby a, parametr C nazywa się *charakterystyką* a parametr M jest częścią ułamkową wyrażonej w systemie dwójkowym mantysy liczby a. Jeżeli C zostanie przedstawione w systemie dwójkowym przy pomocy 7 bitów, to liczba a jest kodowana w następujący sposób:

Reprezentacja krótka: $\boxed{0}\boxed{1 \rightarrow 7}\boxed{8 \rightarrow 31}$

$$S \updownarrow \qquad C \updownarrow \qquad M \updownarrow$$

Reprezentacja długa: $\boxed{0}\boxed{1 \rightarrow 7}\boxed{8 \rightarrow 63}$

<u>*Przykłady*</u>
Rozkodujemy liczby zmiennoprzecinkowe zakodowane w standardzie IBM.

a) $[a]_{IBM} = 0\,|\,1000001\,|\,111010 \ldots 0 = (-1)^0 \cdot 16^{65-64} \cdot \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{32} \right) = +16 \cdot \frac{29}{32} = +\frac{29}{2} = +15.5.$

b) $[a]_{IBM} = 1\,|\,0111111\,|\,10100 \ldots 0 = (-1)^1 \cdot 16^{63-64} \cdot \left(\frac{1}{2} + \frac{1}{8} \right) = -\frac{1}{16} \cdot \frac{5}{8} = -\frac{5}{128} = -0.0390625.$

<u>*Przykłady*</u>
Zakodujemy liczbę w standardzie IBM.
a)

$$a = -917504 = -16^5 \cdot \frac{917504}{16^5} = -16^5 \cdot 0.875 = (-1)^1 \cdot 16^{69-64} \cdot 0.111_2 ,$$

gdyż

$$\begin{array}{r} 0.875 \\ \times 2 \\ \hline \boxed{1}.750 \end{array} \qquad \begin{array}{r} 0.75 \\ \times 2 \\ \hline \boxed{1}.50 \end{array} \qquad \begin{array}{r} 0.5 \\ \times 2 \\ \hline \boxed{1}.0 \end{array}$$

Ponadto

$$69 = 64 + 4 + 1 = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1000101_2$$

i dlatego

$$[a]_{IBM} = 1\,|\,1000101\,|\,1110 \ldots 0$$

b)

$$a = 0 = +16^{0-64} \cdot 0.0 \ldots 0_2$$

i dlatego

$$[a]_{IBM} = 0\,|\,000000\,|\,00 \ldots 0$$

Standard IEEE

Tu również występuje *reprezentacja krótka* 32-bitowa i *reprezentacja długa* 64-bitowa. Na potrzeby tej pierwszej przedstawiamy liczbę $a \neq 0$ w postaci

$$a = (-1)^S \cdot 2^{E-127} \cdot (1.F)_2$$

(symbole E oraz F oznaczają pewne liczby naturalne a nie cyfry sytemu pozycyjnego) i definiujemy reprezentację w następujący sposób zakładając, że E jest wyrażone w systemie dwójkowym:

0	$1 \to 8$	$9 \to 31$
S	E	F

W przypadku reprezentacji długiej punktem wyjścia jest przedstawienie

$$a = (-1)^S \cdot 2^{E-1023} \cdot (1.F)_2$$

a kod ma postać

0	$1 \to 11$	$12 \to 63$
S	E	F

Przykłady. Rozkodujemy liczby zmiennoprzecinkowe zakodowane w systemie krótkim *IEEE*.

a)

$$[a]_{IEEE} = 3F800000_{16} = 0\,|\,011\ 1111\ 1\,|\,000\ 0000\ 0000\ 0000\ 0000\ 0000 = (-1)^0 \cdot 2^{127-127} \cdot 1.0_2 = +1.$$

b)

$$[a]_{IEEE} = BF800000_{16} = 1\,|\,011\,1111\ 1\,|\,000\ 0000\ 0000\ 0000\ 0000\ 0000 = (-1)^1 \cdot 2^{127-127} \cdot 1.0 = -1.$$

c)

$$[a]_{IEEE} = 40480000_{16} = 0\,|\,100\ 0000\ 0\,|\,100\ 1000\ 0000\ 0000\ 0000\ 0000 = (-1)^0 \cdot 2^{128-127} \cdot 1.1001_2 =$$

$$11.001_2 = 3 + \frac{1}{8} = 3.125.$$

Przykład. Zakodujemy liczbę $a = -30.25$ w krótkim systemie *IEEE*. Mamy

$$a = -30.25 = -\left(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}\right) = -11110.01_2 =$$

$$-2^4 \cdot 1.111001 = (-1)^1 \cdot 2^{131-127} \cdot 1.111001_2,$$

oraz

$$131 = 128 + 2 + 1 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 10000011_2.$$

Stąd

$$[a]_{IEEE} = 1100\,|\,0001\,|\,1111\,|\,0010\,|\,00\ \ldots\ 0 = C1F20000_{16}.$$

Przykład. Rozkodujemy liczbę zmiennoprzecinkową a zakodowaną w systemie długim *IEEE*.

Niech

$$[a]_{IEEE} = C03E\underbrace{0\ \ldots\ 0}_{12 razy}{}_{16} = 1\,|\,100\ 0000\ 0011\,|\,1110\,0000\ \ldots\ 0.$$

Stąd

$$a = (-1)^1 \cdot 2^{1027-1023} \cdot 1.111_2 = -2^4 \cdot 1.111_2 = -11110_2 = -(16+8+4+2) = -30.$$

Przykład. Zakodujemy liczbę $a = 7.625$ w długim kodzie *IEEE*.
Mamy

$$a = 7 + \frac{625}{1000} = 7 + \frac{25}{40} = 7 + \frac{5}{8} = \frac{61}{8} = \frac{32+16+8+4+1}{8} = 4+2+1+\frac{1}{2}+\frac{1}{8} = 111.101_2 =$$
$$2^2 \cdot 1.11101_2 = (-1)^0 \cdot 2^{1025-1023} \cdot 1.11101_2,$$

przy czym

$$1025 = 1024 + 1 = 100\,0000\,0001_2 .$$

Stąd

$$[a]_{IEEE} = 0100\,|\,0000\,|\,0001\,|\,1110\,|\,1000\,|\,0\,...\,0 = 401E\,8000\,0000\,0000_{16} .$$

Uwaga. W przypadku standardu *IEEE*, zarówno krótkiego jak i długiego, następujące przypadki szczególne wymagają wyjaśnienia.

a) Jeżeli $E = 0\,...\,0$ oraz $F = 0\,...\,0$, to a jest najmniejszą liczbą, co do wartości bezwzględnej, którą można w danym systemie zakodować i dlatego przyjmuje się w zależności od tego, czy bit S jest zerem, czy jedynką, że jest to reprezentacja $+0$ albo -0.

b) Jeżeli $E = 1\,...\,1$ oraz $F = 0\,...\,0$, to czynnik 2^{E-127} albo 2^{E-1023} osiąga największą możliwą wartość i dlatego przyjmuje się w zależności od tego, czy bit S jest zerem, czy jedynką, że jest to reprezentacja $+\infty$ albo $-\infty$.