Naturalny kod binarny (NKB)

pozycja	7	6	5	4	3	2	1	0
wartość	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
wartość	128	64	32	16	8	4	2	1
bity	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- System pozycyjny o podstawie systemu 2
- Liczby określone są bez znaku
- Wartość liczby binarnej (N- długość słowa kodowego)
  $Wartosc = \sum_{i=0}^{N-1} 2^i \cdot b_i$
- Wartość cyfry zależy od pozycji $b_i = 2^i$ (numerowanie od zera)
- 2^N różnych wartości kodu (kod pełny)

Sumowanie

76_{10}		0	1	0	0	1	1	0	0
188_{10}	+	0	1	1	1	0	1	1	0
194_{10}	=	1	1	0	0	0	0	1	0
przeniesienie		0	1	1	1	1	1	0	0

- Sumowanie dwóch a, b bitów: $a_i, b_i, c_i \Rightarrow s_i, c_{i+1}$ (c - przeniesienie, s - wynik sumowania)

Przekroczenie zakresu

152_{10}		1	0	0	1	1	0	0	0
118_{10}	+	0	1	1	1	0	1	1	0
14_{10} ?	=	0	0	0	0	1	1	1	0
przeniesienie		1	1	1	1	0	0	0	0

- Przeniesienie z najstarszego bitu ($c_{N-1} = 1$) oznacza przekroczenie zakresu dla słowa N-bitowego,
- Alternatywnie: Wystąpienie przeniesienia oznacza, że wynik jest jest $N + 1$-liczbą bitową. Przeniesienie bitu należy wówczas traktować $N + 1$ bit wyniku.

Reprezentacja "znak-moduł" ZM

Najstarszy bit słowa b_{N-1} (MSB - ang. *Most Significant Bit*) pełni rolę znaku (tj. jeśli $b_{N-1} = 1$ to liczna jest ujemna, gdy $b_{N-1} = 0$ dodatnia) np.:

-24_{10}	=	1	0	0	1	1	0	0	0
118_{10}	=	0	1	1	1	0	1	1	0
-14_{10}	=	1	0	0	0	1	1	1	0

$wrtosc = (-1)^{b_{N-1}} \cdot \sum_{i=0}^{N-2} 2^i \cdot b_i$

- Ze względu na najstarszy bit kod nie jest wagowy,
- zakres kodu $< -(2^{N-1} - 1), 2^{N-1} - 1 >$,
- $2^N - 1$ kombinacji - zero posiadałoby dwie reprezentacje (kombinacja 10000000 (minus zero) jest zabroniona),
- Kłopotliwe sprawdzanie bitu znaku i wykonywanie operacji na modułach.
- Idea bitu znaku jest wykorzystywana w innych reprezentacjach (np. w eksponencie liczb zmiennoprzecinkowych)

kod uzupełnień do 1 (U1) (ang. 1s complement)

- W zapisie tym najbardziej znaczący bit jest także bitem znaku (0 – liczba dodatnia, 1 – liczba ujemna), ale w zależności od jego wartości dalsze bity zapisu maja różne znaczenie.
  - Jeśli bit znaku jest 0 (liczba dodatnia), to dalsze bity reprezentują liczby dodatnie w ZM.
  - Natomiast gdy bit znaku jest 1 (liczba ujemna), to dalsze bity reprezentują moduł liczby ujemnej, w taki sposób, że zanegowane ich wartości odpowiadają modułowi tej liczby w kodzie ZM.
- Zapis U1 dla liczb dodatnich jest taki sam jak zapis ZM.
- Różnice w zapisie występują jedynie dla liczb ujemnych.
- Zakres liczb tego zapisu jest taki sam jak dla zapisu ZM.

Kod uzupełnień do 1

- W zapisie U1 występują także dwie reprezentacje zera: 000000...00 i 111111...11.
- Sposób przeliczenia liczby ujemnej w zapisie ZM na zapis U1: Zanegować bity oznaczające moduł liczby (bit znaku pozostaje 1). Np. dla liczb 8-bitowych:

  zapis ZM: 11010110 (dziesiętnie -86)

  zapis U1: 10101001

Kod uzupełnień do 2

Najstarszy bit MSB ma wartośc ujemną pozostałe bity są dodatnie tj.:

$$wartosc = -2^{N-1} \cdot b_{N-1} + \sum_{i=0}^{N-2} 2^i \cdot b_i$$

- Najstarszy bit identyfikuje czy liczba jest dodatnia czy ujemna.
- Zakres kodu: $< -2^{N-1}, 2^{N-1} - 1 >$,
- 2^N kombinacji (kod pełny), zero ma tylko jedną reprezentację,
- Liczby dodatnie z przedziału $< 0, 2^{N-1} - 1 >$ mają identyczną reprezentacje w U2 co w NKB tj.:
  $(0, b_{N-2}, \ldots, b_1, b_0)_{U2} = \sum_{i=0}^{N-2} 2^i \cdot b_i$
- kod wagowy, najstarszy bit na wartość ujemną. Liczby ujemne można interpretować jako sumę:
  $(1, b_{N-2}, \ldots, b_1, b_0)_{U2} = -2^{N-1} + \sum_{i=0}^{N-2} 2^i \cdot b_i$

- wada kodu $U2$: zakres kodu jest niesymetryczny, negacja liczby -2^{N-1} prowadzi do błędu (np. dla $N = 128$ liczba -128 mieści się w zakresie, ale 128 już nie).
- Przekroczenie zakresu przy sumowaniu, np. dla $N = 8$:
  $(127)_{U2} + (4)_{U2} = (-125)_{U2}$ - błąd
- Inkrementacja liczby 127 daje wynik -128.

Negowanie liczb w kodzie U2

$$-(wartosc)_{U2} = \overline{(wartosc)_{U2}} + 1$$

Aby obliczyć liczbę przeciwną do danej w kodzie $U2$ należy zanegować wszystkie bity i do wyniku dodać jedynkę np.:

7_{10}		(00000111)
negacja bitów		(11111000)
dodać bit	+	(00000001)
wynik -7	=	$(11111001)_{U2}$

Dodawanie i odejmowanie w kodzie U2

- **Dodawanie** wykonywane jak w NKB, niezależnie od znaków argumentów
- Wartość przeniesienia z sumowania najstarszego bitu jest ignorowana
- Przekroczenie zakresu (nadmiar) $\Longleftrightarrow$ suma dwóch liczb dodatnich jest ujemna lub suma dwóch liczb ujemnych jest dodatnia
- **Odejmowanie** w $U2$ **-** dodanie negacji odjemnika tj.:

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

- wystarczą operacje negowania i dodawania.

Odejmowanie w kodzie U2 - przykłady

$$
\begin{array}{rcl}
25 + (-1): & & \\
25: & & 00011001 \\
-1: & + & 11111111 \\
\hline
(c_7 = 1): & = & 00011000_{U2} = 24_{10}
\end{array}
$$

$$
\begin{array}{rcl}
25 + (-56): & & \\
25: & & 00011001 \\
-56: & + & 11001000 \\
\hline
(c_7 = 0): & = & 11100001_{U2} = -31_{10}
\end{array}
$$

Dodawanie w kodzie U2 - przykłady

$$
\begin{array}{rcl}
25+1 & & \\
25: & & 00011001 \\
+1: & + & 00000001 \\
\hline
(c_7=0): & = & 00011010_{U2}=26_{10}
\end{array}
$$

$$
\begin{array}{rcl}
(-25)+(-56): & & \\
-25: & & 11100111 \\
-56: & + & 11001000 \\
\hline
(c_7=1): & = & 10101111_{U2}=-81_{10}
\end{array}
$$

Przekroczenie zakresu w kodzie U2 - przykłady

$112 + 113$:

112 :		01110000
113 :	+	01110001
$(c_7 = 0, c_6 = 1)$:	=	11100001 **- przepełnienie**

$(-75) + (-56)$:

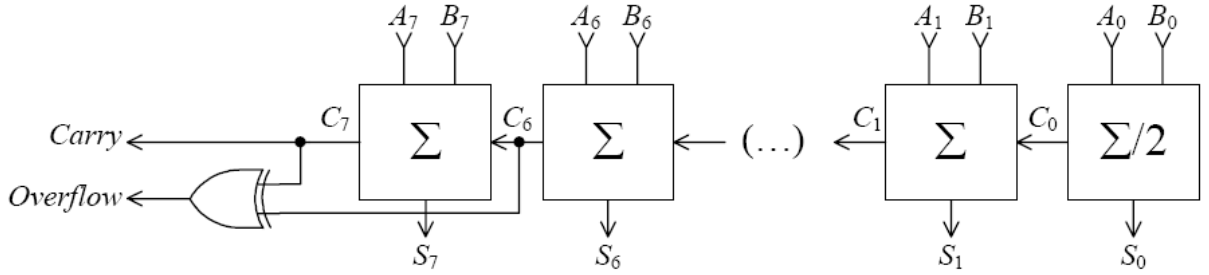
-75 :		10110101
-56 :	+	11001000
$(c_7 = 1, c_6 = 0)$:	=	01111101 **- przepełnienie**

Sprzętowe wykrywanie przekroczenia zakresu w $U2$

Sumowanie najstarszego bitu:

A	B	C_{IN}	C_{OUT}	S	OFL
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

$\Rightarrow \quad OFL = C_{IN} \oplus C_{OUT}$



Kod BCD

Packed Binary Coded Decimal w dwóch tetrada przechowywane są dwie cyfry dziesiętne $(0, \ldots, 9)$

wartość	80	40	20	10	8	4	2	1
bity	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

$$wartosc = \sum_{i=0}^{7} 10^{\frac{i}{4}} \cdot 2^{imod4} \cdot b_i$$

- np. $0011\ 1001 = 39_{HEX} \Leftrightarrow 39_{10}$
- Kod niepełny - 156 kombinacji zabronionych
- Używany ze względu na prostotę konwersji liczb zapisanych dziesiętnie

Dekodowanie w kodzie BDC

Proste sumowanie binarne (możliwe trzy przypadki)

1)

```
   36   0011 0110
 + 42   0100 0010
 ====  ==========
   78   0111 1000
```

– wynik poprawny bez korekcji

2)

```
   36   0011 0110
 + 45   0100 0101
 ====  ==========
   81   0111 1011
           7    11
```

– młodsza cyfra > 9,
korekcja: –10 & +1 do starszej cyfry ⇔ +6:

```
        0111 1011
      +       110
      ===========
        1000 0001
           8    1
```

– wynik poprawny

Dekodowanie w kodzie BDC

3)

```
    38   0011 1000
  + 49   0100 1001
  ====  ==========
    87   1000 0001
            8     1  – przeniesienie pomiędzy tetradami (AC, z b3 do b4)
                     korekcja: +6
         1000 0001
       +      110
       ===========
         1000 0111  – wynik poprawny
            8     7
```

- Analogiczne przypadki przy przepełnieniu starszej tetrady.

> Korekcja BCD jest konieczna, gdy którakolwiek cyfra jest niepoprawna (> 9) lub zostało wygenerowane z niej przeniesienie (przeniesienie połówkowe AC z bitu b_3 lub przeniesienie C z bitu b_7); polega na dodaniu 6 do korygowanej tetrady.

Reprezentacja liczb rzeczywistych

- **Reprezentacja stałoprzecinkowa** (ang. *fixed point*)
- **Reprezentacja zmiennoprzecinkowa** (ang. *floating point*)

Reprezentacja stałoprzecinkowa

W sposób arbitralny przyjmuje się, że część słowa reprezentuje *część całkowitą*, a pozostała część słowa *część ułamkową* np. dla słowa ośmiobitowego przyjmijmy część całkowitą jako 5 bajtów a część ułamkową jako 3 bajty

pozycja:	7	6	5	4	3	2	1	0
wartość:	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
wartość:	16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
bity:	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Reprezentacja stałoprzecinkowa

- w Interpretacji stałoprzecinkowej można również przyjąć interpretację $U1$, $U2$, ZM (najstarszy bit będzie miał znaczenie jak w tych kodowaniach)
- Kodowanie stałoprzecinkowe może powodować błąd,
- Dokładność kodowania zależna jest od długości słowa,
- Niektóre liczby całkowite i wymierne nie mają swojej dokładnej reprezantacji w skończonym kodowaniu,
- Liczby niewymierne zawsze kodowane są z błędem.

Reprezentacja zmiennoprzecikowa

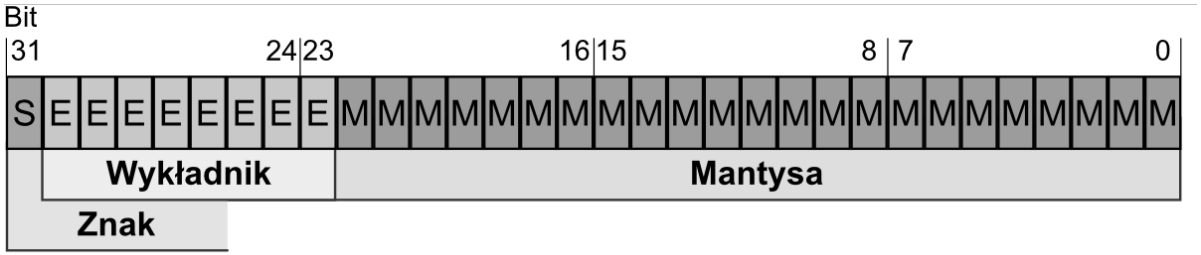
Ogólnie: Liczba zmienna przecinkowa jest reprezentowana jako mantysa i wykładnik

mantysa	wykładnik

Przykład:

	mantysa	wykładnik
dziesiętny:	$2,14$	10^3
dwójkowy:	$0,10001$	2^{010}

Reprezentacja zmiennoprzecikowa - przykłady



- Wykładnik reprezentowany jest w kodzie ZM,
- Mantysa jest ułamkowa.

Zadania na ćwiczenia

1. Zbuduj z bramek NAND sumator jednobitowy. Sprawdź jego działanie.
2. Za pomocą sumatora czterobitowego przeprowadź operację sumowania dwóch czterobitowych liczb dwójkowych bez przepełnienia (wskazanych przez prowadzącego). Wynik zinterpretuj w kodzie NKB i $U2$,
3. Wykonaj operacje sumowania dwóch czterobitowych liczb dodatnich generujących przepełnienie (wskazanych przez prowadzącego). Wynik oraz przeprowadzone operacje zinterpretuj.
4. Wykonaj operacje sumowania dwóch czterobitowych liczb ujemnych generujących przepełnienie (wskazanych przez prowadzącego). Wynik oraz przeprowadzone operacje zinterpretuj.
5. Zaprojektuj i sprawdź działanie układu do identyfikacji

przepełnienia. Układ powinien również sprawdzać, czy przepełnienie wystąpiło wskutek sumowania dwóch liczb dodatnich czy dwóch liczb ujemnych.

6. Zaproponuj reprezentacje ujemnych i dodatnich liczb rzeczywistych z częścią ułamkową za pomocą 8-bitów. Określ przedział liczbowy, który może być reprezentowany oraz dokładność reprezentacji. Za pomocą sumatora 8-bitowego wykonaj sumowanie dwóch liczb rzeczywistych (dodatniej i ujemnej) wskazanych przez prowadzącego. Określ błąd reprezentacji obu liczb oraz wyniku. Wynik oraz przeprowadzone operacje zinterpretuj.