Arytmetyka binarna - wykład 6

Adam Szmigielski
aszmigie@pjwstk.edu.pl

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		$\boxed{1}$	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 liczbą bitową o długości $N + 1$. Przeniesienie bitu należy wówczas traktować jako $N + 1$ 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.:

$$
\begin{array}{lccccccccc}
-24_{10} & = & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
118_{10} & = & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
-14_{10} & = & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}
$$

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

- 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. *1's 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 (U2) (ang. *2's complement*)

Najstarszy bit MSB ma wartość 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.:
  $(0, b_{N-2}, \dots, b_1, b_0)_{U2} = \sum_{i=0}^{N-2} 2^i \cdot b_i$
- kod wagowy, najstarszy bit na wartość ujemną. Liczby ujemne można interpretować jako sumę:
  $(1, b_{N-2}, \dots, b_1, b_0)_{U2} = -2^{N-1} + \sum_{i=0}^{N-2} 2^i \cdot b_i$

- 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.:

$$\begin{array}{rcl} 7_{10} & & (00000111) \\ \text{negacja bitów} & & (11111000) \\ \text{dodać bit} & + & (00000001) \\ \hline \text{wynik } -7_{10} & = & (11111001)_{U2} \end{array}$$

Dodawanie i odejmowanie w kodzie U2

- **Dodawanie** wykonywane jak w NKB, niezależnie od znaków argumentów
- **Wartość przeniesienia z najstarszego bitu jest ignorowana**,
- **Przekroczenie zakresu** może wystąpić w dwóch przypadkach:
  - gdy suma dwóch liczb dodatnich przekracza zakres,
  - gdy suma dwóch liczb ujemnych przekracza zakres,
- **Odejmowanie** w $U2$ - dodanie negacji odjemnika tj.:

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

- wystarczą operacje negowania i dodawania.

Odejmowanie w kodzie U2 - przykłady

- Sumowanie liczby dodatniej i ujemnej - wynik dodatni,

$25 + (-1)$:

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

- Sumowanie liczby dodatniej i ujemnej - wynik ujemny

$25 + (-56)$:

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

Dodawanie w kodzie U2 - przykłady

- Sumowanie dwóch liczb dodatnich bez przekroczenia zakresu,

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

- Sumowanie dwóch liczb ujemnych bez przekroczenia zakresu,

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

Przekroczenie zakresu w kodzie U2 - przykłady

- Sumowanie dwóch liczb dodatnich z przekroczeniem zakresu,

$$
\begin{array}{rcl}
112+113: & & \\
112: & & 01110000 \\
113: & + & 01110001 \\
\hline
(c_7=0, c_6=1): & = & 11100001
\end{array}
$$
- przekroczeniem zakresu

- Sumowanie dwóch liczb ujemnych z przekroczeniem zakresu,

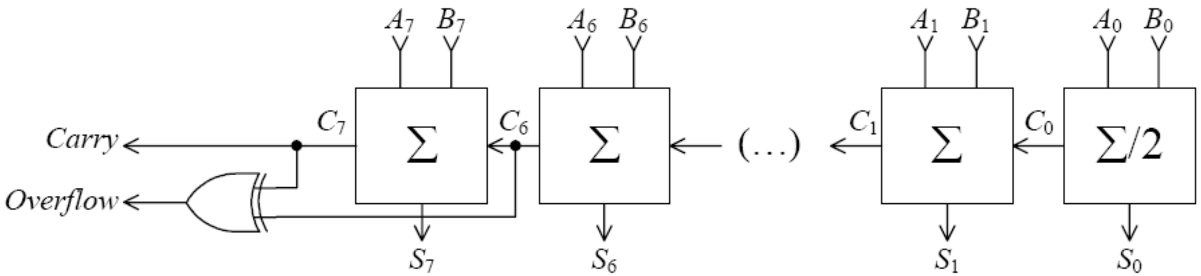
$$
\begin{array}{rcl}
(-75)+(-56): & & \\
-75: & & 10110101 \\
-56: & + & 11001000 \\
\hline
(c_7=1, c_6=0): & = & 01111101
\end{array}
$$
- przekroczeniem zakresu

Sprzętowe wykrywanie przekroczenia zakresu w $U2$

- Przekroczenie zakresu w U2 można zidentyfikować analizując przeniesienia:
  przychodzące C_{IN} i generowane C_{OUT} przez najstarszy bit,

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

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



- Przekroczenie zakresu występuje wtedy i tylko wtedy, gdy oba przeniesienia C_{IN} i C_{OUT} są przeciwnego znaku.

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

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

- np. $0011\ 1001 = 39_{HEX} \Leftrightarrow 39_{10}$,
- Kod niepełny - $2^8 - 10^4 = 156$ kombinacji zabronionych,
- Używany ze względu na prostotę konwersji liczb zapisanych dziesiętnie.

Reprezentacja liczb rzeczywistych

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

Reprezentacja stałoprzecinkowa

- W sposób arbitralny przyjmuje się, że część słowa jest *częścią całkowitą*, a pozostała część słowa *część ułamkową*,
- Przykładowo, dla słowa ośmiobitowego przyjmijmy część całkowitą jako 5 bitów a część ułamkową jako 3 bity:

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

- Liczby stałoprzecinkowe można również interpretować w kodach $U1$, $U2$ czy 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.

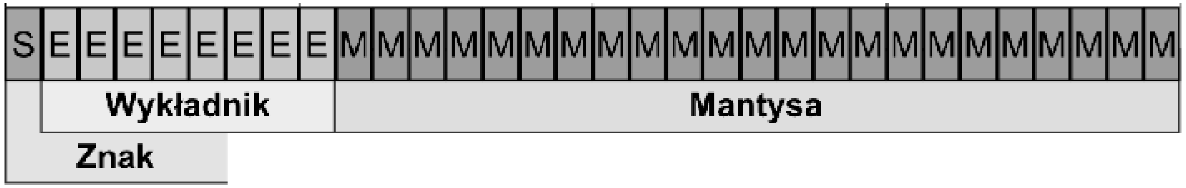
Reprezentacja stałoprzecinkowa - przykład

Podtrzymując założenie że, 5 najstarszych bitów przeznaczone jest na *część całkowitą* a pozostałe 3 bity na *część ułamkową*.
Dodatkowo przyjmujemy, że liczba jest zapisana w kodzie U2:

wartość:	-2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
wartość:	-16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

- Przykładowy ciąg bitów 11001110 jest wówczas równy:
  $-16 + (8 + 1 + \frac{1}{2} + \frac{1}{4}) = -6\frac{1}{4}$,
- Zakres reprezentowanych liczb mieści się w przedziale
  $< -16, 15\frac{7}{8} >$,
- Liczby rzeczywiste są reprezentowane z błędem nie większym od $\frac{1}{8}$.

Reprezentacja zmiennoprzecinkowa



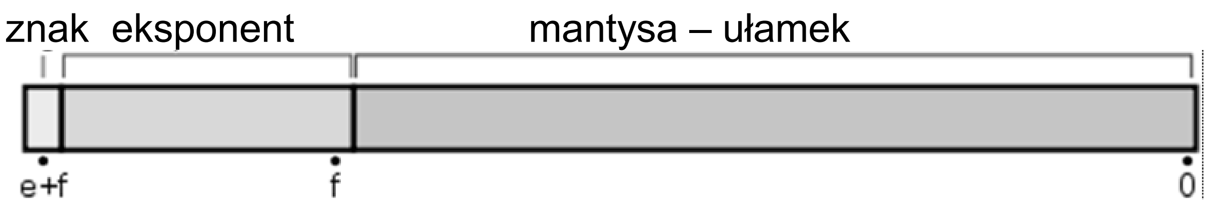
- Liczba zmiennoprzecinkowa jest reprezentowana jako *mantysa* i *eksponent* przy podstawie 2:

$$mantysa \cdot 2^{wykladnik}$$

- *Mantysa* może mieć różne interpretacje (co może być przyczyną nieporozumień),
- *Eksponent* jest liczbą całkowitą dodatnią albo ujemną. Mnożeniu lub dzieleniu przez 2 odpowiada przesuwanie przecinka odpowiednio w prawo i lewo.

Standard reprezentacja zmiennoprzecinkowej IEEE 754

- *Standard IEEE 754* reguluje: format zapisu, sposób konwersji z danego formatu, algorytmy zaokrąglające, operacje arytmetyczne i obsługę wyjątków (np. dzielenie przez zero),
- Przykład Standard *IEEE 754-1985*



$$Wartosc = (-1)^{znak} \cdot 2^{eksponent-stala} \cdot (1 + ulamek)$$

- *Mantysa* w kodzie ZM jest ułamkiem. Najbardziej znaczący bit zawsze =1 nie zapisywany (jest pomijany),
- *Eksponent* Od liczby zapisanej kodzie NKB odejmuje się stałą wartość (połowa zakresu NKB). W ten sposób mogą eksponent może przyjmować wartości ujemne i dodatnie.

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 [a]. Wynik zinterpretuj w kodzie NKB i $U2$,
3. Za pomocą sumatora czterobitowego wykonaj sumowanie liczb w kodzie U2 takie, że:
   - suma dwóch liczb dodatnich powoduje przekroczenie zakresu,
   - suma dwóch liczb ujemnych powoduje przekroczenie zakresu.

   Odczytaj i zinterpretuj otrzymany wynik.
4. Dla przypadków z poprzedniego punktu zrealizuj układ identyfikujący przekroczenie zakresu. Układ powinien również sprawdzać, czy przepełnienie wystąpiło wskutek sumowania dwóch liczb dodatnich czy ujemnych.

---
[a] wskazanych przez prowadzącego

5. Posługując się kodem U2 zaproponuj sposób reprezentacji liczb z częścią ułamkową na ośmiu bitach. Określ przedział, który może być reprezentowany oraz dokładność reprezentacji. Za pomocą sumatora 8-bitowego wykonaj sumowanie dwóch liczb[b]. Określ błąd reprezentacji obu liczb oraz wyniku.

[b] wskazanych przez prowadzącego