Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Podstawy informatyki

Kod przedmiotu: **EKS1C1007**

(studia stacjonarne)

SYSTEMY LICZBOWE, JEDNOSTKI INFORMACJI, KODOWANIE LICZB I ZNAKÓW

Numer ćwiczenia

PINF01

Autor:
dr inż. Jarosław Forenc

Białystok 2023

Spis treści

1. Opis stanowiska 3
1.1. Stosowana aparatura 3
2. Wiadomości teoretyczne 3
2.1. Pozycyjne systemy liczbowe 3
2.2. Konwersje pomiędzy systemami liczbowymi 5
2.3. Jednostki informacji cyfrowej 9
2.4. Kodowanie 10
2.5. Kody alfanumeryczne 11
2.6. Kody liczbowe 16
2.7. Liczby zmiennoprzecinkowe 18
3. Przebieg ćwiczenia 24
4. Literatura 25
5. Pytania kontrolne 26
6. Wymagania BHP 26

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

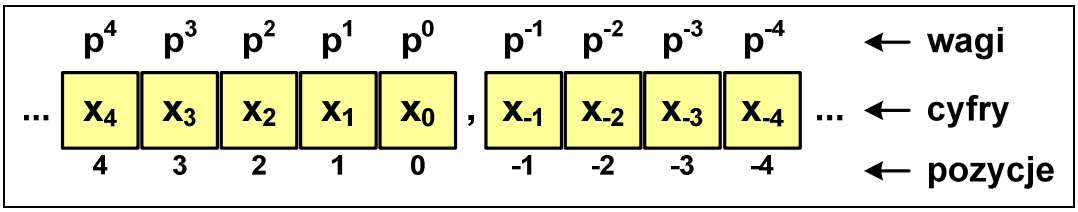
Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11.

2. Wiadomości teoretyczne

2.1. Pozycyjne systemy liczbowe

System liczbowy jest to zbiór zasad umożliwiających zapis liczb za pomocą cyfr oraz wykonywanie działań na tych liczbach. Systemy liczbowe dzielą się na pozycyjne i niepozycyjne. W systemie pozycyjnym znaczenie cyfr jest zależne od miejsca (pozycji), które zajmują one w liczbie (np. w systemie dziesiętnym, w liczbie 111 każda cyfra ma inne znaczenie - setki, dziesiątki, jedności). W systemie niepozycyjnym znaczenie cyfr jest niezależne od miejsca położenia w liczbie (np. w systemie rzymskim, w liczbie III, każda cyfra ma taką samą wartość).

Konstrukcja wszystkich pozycyjnych systemów liczbowych jest bardzo podobna. Załóżmy, że mamy system o podstawie **p** (Rys. 1).



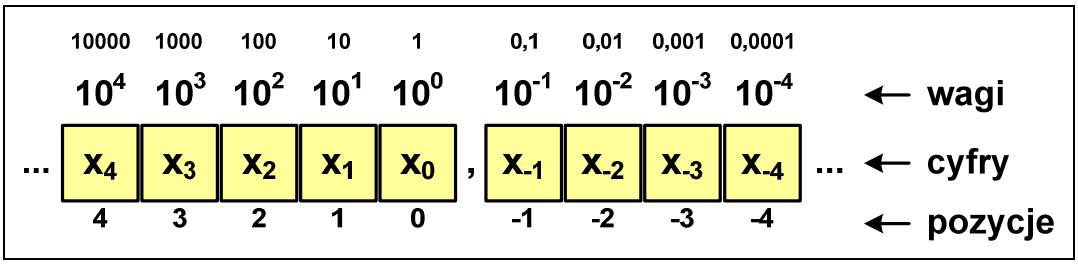
Rys. 1. System pozycyjny o podstawie **p**

W systemie takim stosowana jest ograniczona liczba cyfr o kolejnych wartościach **0,1,2,..., p-1**. Zatem liczba cyfr jest równa podstawie systemu, zaś wartość największej cyfry jest o 1 mniejsza od podstawy **p**.

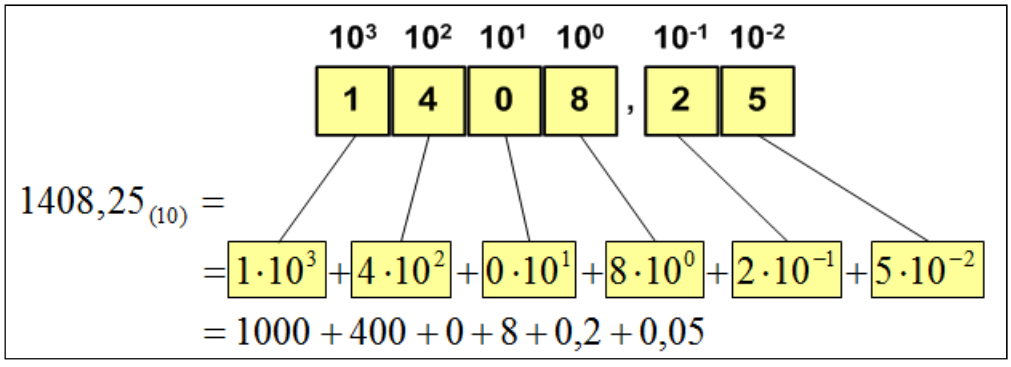
Liczba w systemie o podstawie **p** składa się z cyfr części całkowitej (x_0, x_1, x_2, x_3, x_4,…) i cyfr części ułamkowej (x_{-1}, x_{-2}, x_{-3}, x_{-4}, …). Cyfry ustawiane są na kolejnych pozycjach. Wartość cyfry w zapisie zależy od jej pozycji. Każda pozycja posiada swoją wagę równą podstawie systemu podniesionej do potęgi o wartości pozycji. Wartość liczby (w systemie dziesiętnym) powstaje poprzez wymnożenie danej cyfry przez odpowiadającą jej wagę i zsumowanie otrzymanych iloczynów:

$$X_{(10)} = \ldots + x_4 \cdot p^4 + x_3 \cdot p^3 + x_2 \cdot p^2 + x_1 \cdot p^1 + x_0 \cdot p^0 + \\ + x_{-1} \cdot p^{-1} + x_{-2} \cdot p^{-2} + x_{-3} \cdot p^{-3} + x_{-4} \cdot p^{-4} + \ldots \quad (1)$$

W systemie **dziesiętnym** (dziesiątkowym) podstawa systemu **p = 10**, zaś dozwolone cyfry to: **0,1,2,…,9**. Konstrukcję systemu dziesiętnego pokazuje Rys. 2, zaś sposób obliczenia wartości liczby w tym systemie - Rys. 3.

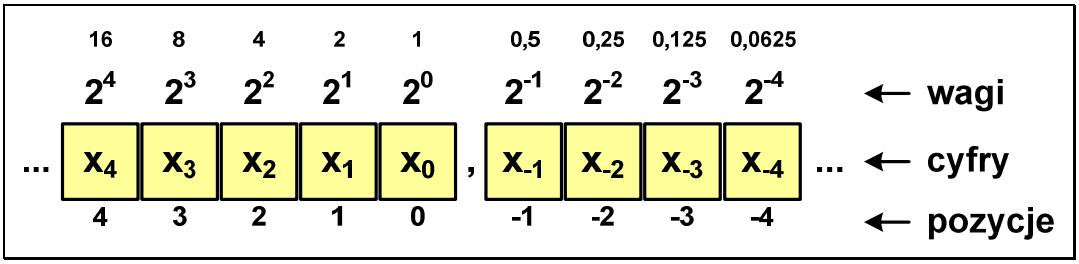


Rys. 2. System dziesiętny

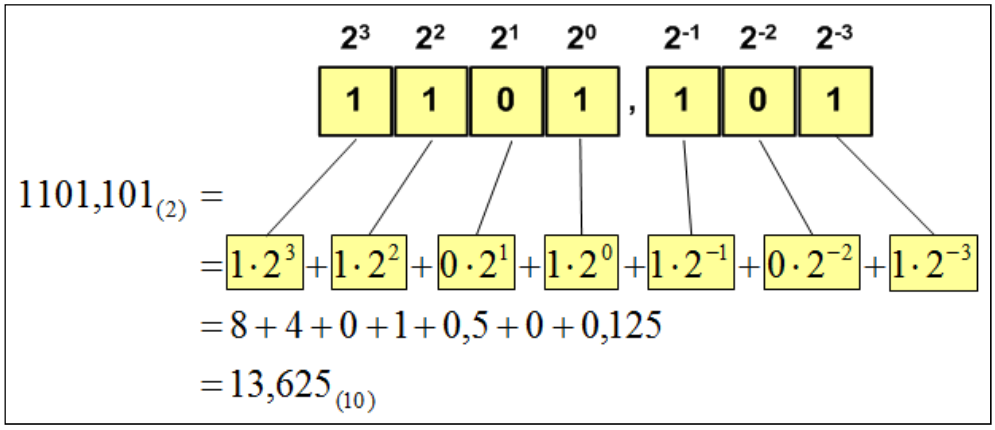


Rys. 3. Określenie wartości liczby w systemie dziesiętnym

W systemie **dwójkowym** (binarnym) podstawa systemu **p = 2**. W systemie tym można stosować tylko dwie cyfry: **0** i **1**. Wagi są natomiast kolejnymi potęgami liczby 2. Konstrukcję systemu dwójkowego pokazuje Rys. 4, natomiast sposób obliczenia wartości liczby w tym systemie - Rys. 5.



Rys. 4. System dwójkowy



Rys. 5. Określenie wartości liczby w systemie dwójkowym

2.2. Konwersje pomiędzy systemami liczbowymi

Konwersja liczby całkowitej z systemu dziesiętnego na system o innej podstawie najczęściej wykonywana jest zgodnie z algorytmem Hornera. W algorytmie tym dzielimy zamienianą liczbę przez podstawę systemu docelowego, otrzymując część całkowitą i resztę z dzielenia. Następnie ponownie dzielimy część całkowitą przez podstawę systemu docelowego, itd. Operację dzielenia kończymy, gdy otrzymana część całkowita jest równa zero. Reszty z dzielenia, zapisane od ostatniej do pierwszej, tworzą kolejne cyfry liczby w systemie docelowym.

Przykład: zamiana liczby $626_{(10)}$ na system siódemkowy

$626_{(10)} = ?_{(7)}$ $\qquad$ $626_{(10)} = 1553_{(7)}$

$626 / 7 = 89$ *reszta* 3
$89 / 7 = 12$ *reszta* 5
$12 / 7 = 1$ *reszta* 5
$1 / 7 = 0$ *reszta* 1

kolejność odczytywania cyfr liczby w systemie siódemkowym

Przykład: zamiana liczby $626_{(10)}$ na system dwójkowy

$$626_{(10)} = ?_{(2)} \qquad\qquad 626_{(10)} = 1001110010_{(2)}$$

$$\begin{array}{lcrlr}
626/2 & = & 313 & reszta & 0 \\
313/2 & = & 156 & reszta & 1 \\
156/2 & = & 78 & reszta & 0 \\
78/2 & = & 39 & reszta & 0 \\
39/2 & = & 19 & reszta & 1 \\
19/2 & = & 9 & reszta & 1 \\
9/2 & = & 4 & reszta & 1 \\
4/2 & = & 2 & reszta & 0 \\
2/2 & = & 1 & reszta & 0 \\
1/2 & = & 0 & reszta & 1
\end{array}$$

Przykład: zamiana liczby $626_{(10)}$ na system czternastkowy

$$626_{(10)} = ?_{(14)} \qquad\qquad 626_{(10)} = 32\mathrm{A}_{(14)}$$

$$\begin{array}{lcrlrl}
626/14 & = & 44 & reszta & 10 & \rightarrow \mathrm{A} \\
44/14 & = & 3 & reszta & 2 & \\
3/14 & = & 0 & reszta & 3 &
\end{array}$$

W przypadku zamiany liczby z dowolnego systemu liczbowego na system dziesiętny korzystamy z wzoru (1), czyli z wymnożenia danej cyfry zamienianej liczby przez odpowiadającą jej wagę i zsumowanie otrzymanych iloczynów.

Przykład: zamiana liczby $21302_{(4)}$ na system dziesiętny

$$21302_{(4)} = ?_{(10)}$$
$$21302_{(4)} = 2 \cdot 4^0 + 0 \cdot 4^1 + 3 \cdot 4^2 + 1 \cdot 4^3 + 2 \cdot 4^4$$
$$21302_{(4)} = 2 \cdot 1 + 0 \cdot 4 + 3 \cdot 16 + 1 \cdot 64 + 2 \cdot 256$$
$$21302_{(4)} = 2 + 0 + 48 + 64 + 512 = 626_{(10)}$$

Przykład: zamiana liczby $AC24_{(17)}$ na system dziesiętny

$$AC24_{(17)} = ?_{(10)}$$
$$AC24_{(17)} = 4 \cdot 17^0 + 2 \cdot 17^1 + 12 \cdot 17^2 + 10 \cdot 17^3$$
$$AC24_{(17)} = 4 \cdot 1 + 2 \cdot 17 + 12 \cdot 289 + 10 \cdot 4913$$
$$AC24_{(17)} = 4 + 34 + 3468 + 49130 = 52636_{(10)}$$

W bardzo szybki sposób można zamienić liczbę z systemu dwójkowego na czwórkowy, ósemkowy i szesnastkowy oraz z tych systemów na system dwójkowy.

Przy zamianie liczby z systemu **dwójkowego** na **czwórkowy** dzielimy liczbę dwójkową na dwucyfrowe grupy(od prawej strony). Jeśli w pierwszej grupie od lewej znajdzie się tylko jedna cyfra, to dopisujemy przed nią 0. Następnie każdą dwucyfrową grupę, zapisaną w systemie dwójkowym, zamieniamy na jedną cyfrę w systemie czwórkowym. Otrzymane w ten sposób cyfry określają końcową wartość liczby w systemie czwórkowym.

Przykład: zamiana liczby $11110110011011_{(2)}$ na system czwórkowy

$$11110110011011_{(2)} = ?_{(4)}$$
$$\underbrace{11}_{3} \mid \underbrace{11}_{3} \mid \underbrace{01}_{1} \mid \underbrace{10}_{2} \mid \underbrace{01}_{1} \mid \underbrace{10}_{2} \mid \underbrace{11}_{3}$$
$$11110110011011_{(2)} = 3312123_{(4)}$$

Przy zamianie liczby z systemu **dwójkowego** na **ósemkowy** postępujemy w analogiczny sposób, ale tym razem tworzymy trzycyfrowe grupy.

Przykład: zamiana liczby $11110110011011_{(2)}$ na system ósemkowy

$$11110110011011_{(2)} = ?_{(8)}$$
$$\underbrace{011}_{3} \mid \underbrace{110}_{6} \mid \underbrace{110}_{6} \mid \underbrace{011}_{3} \mid \underbrace{011}_{3}$$
$$11110110011011_{(2)} = 36633_{(8)}$$

W przypadku zamiany liczby z systemu **dwójkowego** na **szesnastkowy** tworzymy czterocyfrowe grupy.

Przykład: zamiana liczby $11110110011011_{(2)}$ na system szesnastkowy

$$11110110011011_{(2)} = ?_{(16)}$$
$$\underbrace{0011}_{3} \mid \underbrace{1101}_{D} \mid \underbrace{1001}_{9} \mid \underbrace{1011}_{B}$$
$$11110110011011_{(2)} = 3D9B_{(16)}$$

Przy zamianie liczby z systemu **czwórkowego** na **dwójkowy** kolejne cyfry liczby w systemie czwórkowym zapisujemy jako dwie cyfry w systemie dwójkowym.

Przykład: zamiana liczby $12303_{(4)}$ na system dwójkowy

$$12303_{(4)} = ?_{(2)}$$
$$\overbrace{01}^{1}\ \overbrace{10}^{2}\ \overbrace{11}^{3}\ \overbrace{00}^{0}\ \overbrace{11}^{3}$$
$$12303_{(4)} = 110110011_{(2)}$$

Przy zamianie liczby z systemu **ósemkowego** na **dwójkowy** kolejne cyfry liczby w systemie ósemkowym zapisujemy jako trzy cyfry w systemie dwójkowym.

Przykład: zamiana liczby $263_{(8)}$ na system dwójkowy

$$263_{(8)} = ?_{(2)}$$
$$\overbrace{010}^{2}\ \overbrace{110}^{6}\ \overbrace{011}^{3}$$
$$263_{(8)} = 10110011_{(2)}$$

Przy zamianie liczby z systemu **szesnastkowego** na **dwójkowy** kolejne cyfry liczby w systemie szesnastkowym zapisujemy jako cztery cyfry w systemie dwójkowym.

Przykład: zamiana liczby $5A_{(16)}$ na system dwójkowy

$$5A_{(16)} = ?_{(2)}$$
$$\overbrace{0101}^{5}\ \overbrace{1010}^{A}$$
$$5A_{(16)} = 1011010_{(2)}$$

2.3. Jednostki informacji cyfrowej

Podstawową jednostką informacji stosowaną w informatyce i telekomunikacji jest **bit** (ang. binary digit). Określa on najmniejszą ilość informacji potrzebną do stwierdzenia, który z dwóch możliwych stanów przyjął układ. Bit przyjmuje jedną z dwóch wartości: **0** (zero) lub **1** (jeden). Bit jest więc tożsamy z cyfrą w systemie dwójkowym. Bity oznaczane są małą literą „b" (standard IEEE 1541, 2002 r.) lub słowem „bit" (standard IEC 60027).

W praktyce często stosowane są wielokrotności bitów. Zgodnie z układem SI w przedrostkach powinien być stosowany mnożnik 1000 (10^3), ale w informatyce przyjął się mnożnik 1024 (2^{10}). Najczęściej stosowane wielokrotności bitów przedstawia Tabela 1.

Tabela 1. Wielokrotności bitów

Nazwa	Symbol	Mnożnik
bit	b	---
kilobit	kb	$2^{10} = 1024^1$
megabit	Mb	$2^{20} = 1024^2$
gigabit	Gb	$2^{30} = 1024^3$
terabit	Tb	$2^{40} = 1024^4$
petabit	Pb	$2^{50} = 1024^5$
eksabit	Eb	$2^{60} = 1024^6$
zettabit	Zb	$2^{70} = 1024^7$
jottabit	Yb	$2^{80} = 1024^8$

Najmniejszą adresowalną jednostką informacji pamięci komputerowej jest **bajt** (ang. byte). Bajt składa się z bitów. W praktyce przyjmuje się, że jeden bajt to 8 bitów. Za pomocą jednego bajtu można zapisać $2^8 = 256$ różnych wartości. W pierwszych komputerach bajt mógł mieć inną liczbę bitów: 4, 6, 7, 9, 12. 8-bitowy bajt po raz pierwszy został zastosowany pod koniec 1956 r., natomiast został uznany za standard w 1964 r. (po zastosowaniu w komputerze IBM System/360). Inna nazwa 8-bitowego bajtu to oktet. Najczęściej stosowanym skrótem dla bajtu jest wielka litera „B".

W praktyce często stosowane są wielokrotności bajtów. Najczęściej stosowane wielokrotności bajtów (mnożnik $2^{10} = 1024$) przedstawia Tabela 2.

Tabela 2. Wielokrotności bajtów

Nazwa	Symbol	Mnożnik
bajt	B	---
kilobajt	kB	$2^{10} = 1024^1$
megabajt	MB	$2^{20} = 1024^2$
gigabajt	GB	$2^{30} = 1024^3$
terabajt	TB	$2^{40} = 1024^4$
petabajt	PB	$2^{50} = 1024^5$
eksabajt	EB	$2^{60} = 1024^6$
zettabajt	ZB	$2^{70} = 1024^7$
jottabajt	YB	$2^{80} = 1024^8$

Przykłady przejścia pomiędzy różnymi wielokrotnościami bajtów:

kB → B: 1 kB = 1 024 B

MB → B: 1 MB = 1024 × 1024 = 1 048 576 B

GB → B: 1 GB = 1024 × 1024 × 1024 = 1 073 741 824 B

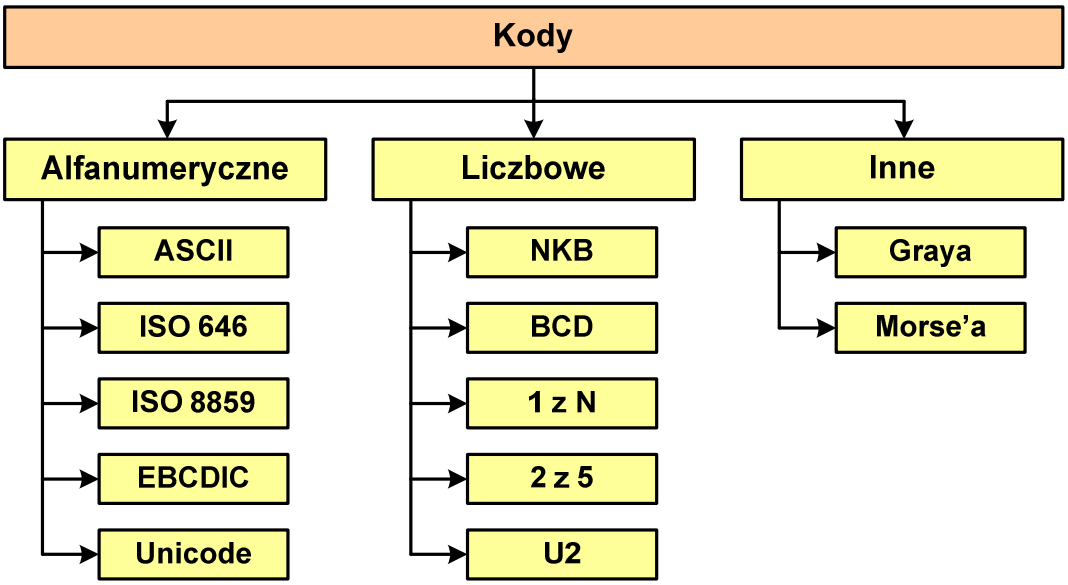
MB → kB: 2 MB = 2 × 1024 = 2 048 kB

GB → kB: 3 GB = 3 × 1024 × 1024 = 3 145 728 kB

TB → kB: 4 TB = 4 × 1024 × 1024 × 1024 = 4 294 967 296 kB

2.4. Kodowanie

Informacje przetwarzane przez komputer to liczby, ale także inne obiekty, np. litery, wartości logiczne, obrazy. Każda taka przetwarzana informacja musi być reprezentowana za pomocą tylko dwóch stanów: wysokiego (1 - jedynka) i niskiego (0 - zero). Konieczne są zatem reguły przekształcania różnych postaci informacji na informację binarną (zero-jedynkową). Proces przekształcania jednego rodzaju postaci informacji na inną postać nazywamy **kodowaniem**. Na Rys. 6 przedstawiony jest podział kodów.



Rys. 6. Podział kodów

2.5. Kody alfanumeryczne

Kody alfanumeryczne przeznaczone są do kodowania znaków (liter, cyfr, znaków przestankowych, itp.). Ogólnie mówiąc, kodowanie znaków polega na tym, że każdemu znakowi przypisywana jest określony numer (liczba). Istnieje wiele różnych kodów. Różnią się one między sobą liczbą bitów (bajtów) przeznaczonych do przechowywania kodów, a tym samym różnią się zbiorami kodowanych znaków.

Kod **ASCII** (ang. American Standard Code for Information Interchange) jest 7-bitowym kodem przypisującym liczbom z zakresu 0-127: litery (alfabet angielski), cyfry, znaki przestankowe, inne symbole i polecenia sterujące. 33 kody o numerach 0-31 i 127 służą do sterowania urządzeniami typu drukarka czy terminal. Pozostałe 95 kodów (numery 32-126) tworzy zbiór znaków ASCII.

Na Rys. 7 przedstawiona jest tabela kodów ASCII. Tabela ta zawiera dla każdego kodu: numer kodu w postaci liczby w systemie dziesiętnym, numer kodu w postaci liczby w systemie szesnastkowym oraz kodowany znak lub oznaczenie polecenia sterującego. Tablica kodów może być przedstawiona także w bardziej zwartej postaci (Rys. 8).

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20	Space	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	″	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	`	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Rys. 7. Tabela kodów ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
20	SP	!	"	#	$	%	&	‘	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Rys. 8. Tabela kodów ASCII w systemie szesnastkowym

Na podstawie powyższej tabeli można określić kod danego znaku w postaci liczby w systemie szesnastkowym. W tym celu należy dodać numer wiersza

i numer kolumny, na przecięciu których znajduje się rozpatrywany znak. Przykład: kod litery „E" to $40_{(16)} + 5_{(16)} = 45_{(16)}$, zaś kod znaku „=" to $30_{(16)} + D_{(16)} = 3D_{(16)}$.

ISO/IEC 8859 jest zestawem standardów służących do kodowania znaków za pomocą 8-bitów. Wszystkie zestawy ISO 8859 mają znaki $0_{(10)}$-$127_{(10)}$ ($00_{(16)}$-$7F_{(16)}$) takie same jak w kodzie ASCII. Pozycjom $128_{(10)}$-$159_{(10)}$ ($80_{(16)}$-$9F_{(16)}$) przypisane są dodatkowe kody sterujące, tzw. C1 (obecnie nie są używane). W czerwcu 2004 roku, grupa robocza odpowiedzialna za utrzymanie zestawów znaków kodowanych ośmiobitowo została rozwiązana. Prace związane rozwojem ISO 8859 zostały wstrzymane, a skoncentrowano się na standardzie Unicode. W skład ISO 8859 wchodzą następujące standardy:

- ISO 8859-1 (Latin-1) - alfabet łaciński dla Europy zachodniej;
- ISO 8859-2 (Latin-2) - łaciński dla Europy środkowej i wschodniej;
- ISO 8859-3 (Latin-3) - łaciński dla Europy południowej;
- ISO 8859-4 (Latin-4) - łaciński dla Europy północnej;
- ISO 8859-5 (Cyrillic) - dla cyrylicy;
- ISO 8859-6 (Arabic) - dla alfabetu arabskiego;
- ISO 8859-7 (Greek) - dla alfabetu greckiego;
- ISO 8859-8 (Hebrew) - dla alfabetu hebrajskiego;
- ISO 8859-9 (Latin-5);
- ISO 8859-10 (Latin-6);
- ISO 8859-11 (Thai) - dla alfabetu tajskiego;
- ISO 8859-12 - brak;
- ISO 8859-13 (Latin-7);
- ISO 8859-14 (Latin-8) - zawiera polskie znaki;
- ISO 8859-15 (Latin-9);
- ISO 8859-16 (Latin-10) - łaciński dla Europy środkowej, zawiera polskie znaki.

Kodowanie **ISO/IEC 8859-2** (Latin-2, „środkowo", „wschodnioeuropejskie") (Rys. 9) obsługuje języki: bośniacki, chorwacki, czeski, węgierski, polski, rumuński, serbski, serbsko-chorwacki, słowacki, słoweński, górno- i dolnołużycki.

W standardzie tym można przedstawić także wszystkie znaki z języka niemieckiego i angielskiego. Kodowanie Latin-2 było zgodne z Polską Normą PN-T-42118:1993: „Technika informatyczna - Znormalizowane zbiory znaków graficznych przeznaczone do stosowania w kodach 8-bitowych”, która została wycofana w 2015 roku.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
20	SP	!	"	#	$	%	&	‘	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80	Nieużywane															
90																
A0	NB SP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SH Y	Ž	Ż
B0	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Rys. 9. Tabela kodów ISO/IEC 8859-2

Unicode (Unikod) to komputerowy zestaw znaków mający obejmować wszystkie pisma i inne znaki (symbole techniczne, wymowy) używane na świecie. Unicode przypisuje unikalny numer każdemu znakowi, niezależny od używanej platformy, programu czy języka. Unicode rozwijany przez konsorcjum utworzone przez firmy komputerowe, producentów oprogramowania oraz grupy użytkowników. Pierwsza wersja Unicode 1.0 została opublikowana w październiku 1991 roku. Ostatnia wersja, Unicode 11.0.0, została udostępniona 5 czerwca 2018 roku. Wersja ta koduje 137 374 znaki.

Pierwsze 128 kodów (kody od U+0000 do U+007F, Basic Latin) zawiera dokładnie to samo co kod ASCII. Kolejne 128 znaków (kody od U+0080 do U+00FF, Latin-1 Supplement) odpowiada standardowi ISO/IEC 8859-1. Litery z polskimi znakami diakrytycznymi znajdują się w kolejnym zakresie kodów (kody od U+0100 do U+017F, Latin Extended-A).

Standard Unicode definiuje nie tylko kody numeryczne przypisane poszczególnym znakom, ale także określa sposób bajtowego kodowania znaków. Kodowanie to określa sposób w jaki znaki ze zbioru mają być zapisane w postaci binarnej. Istnieją trzy podstawowe metody kodowania: UTF-32, UTF-16, UTF-8. Metody te różnią się liczbą bajtów przeznaczonych do opisania kodu znaku.

Kodowanie **UTF-32** (Rys. 10) wymaga użycia 32-bitowych słów kodowych. Kod znaku ma zawsze stałą długość 4 bajtów i przedstawia numer znaku w tabeli Unikodu. Kody obejmują zakres od 0 do 0x10FFFF (od 0 do 1 114 111). Kodowanie to jest jednak bardzo nieefektywne - zakodowane ciągi znaków są 2-4 razy dłuższe niż ciągi tych samych znaków zapisanych w innych kodowaniach.

Rys. 10. Kodowanie UTF-32

UTF-16 (Rys. 11) to sposób kodowania standardu Unicode wymagający użycia 16-bitowych słów. Dla znaków z przedziału od U+0000 do U+FFFF używane jest jedno słowo, którego wartość jest jednocześnie kodem znaku w Unicode. Dla znaków z wyższych pozycji używa się dwóch słów: pierwsze słowo należy do przedziału: U+D800 - U+DBFF, zaś drugie słowo należy do przedziału: U+DC00 - U+DFFF.

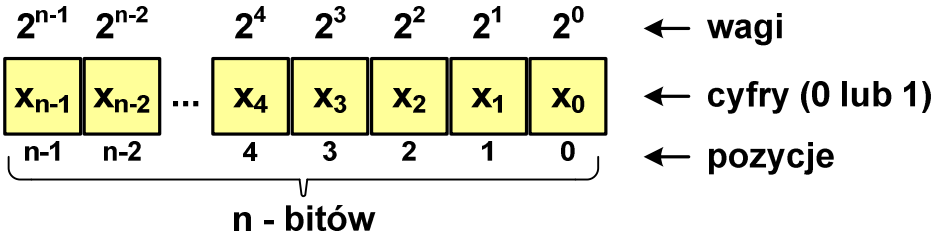
Rys. 11. Kodowanie UTF-16

UTF-8 (Rys. 12) to kodowanie ze zmienną długością reprezentacji znaku wymagające użycia 8-bitowych słów.

Rys. 12. Kodowanie UTF-8

2.6. Kody liczbowe

Kody liczbowe służą do reprezentacji liczb w systemach komputerowych. W przypadku **liczb całkowitych bez znaku** stosowany jest naturalny kod binarny (**NKB**). Kod NKB (Rys. 13) powstaje poprzez przypisanie dowolnej liczbie dziesiętnej odpowiadającej jej liczby binarnej.



Rys. 13. Naturalny kod binarny (NKB)

Używając n-bitów można w **NKB** zapisać liczbę z zakresu:

$$X_{(2)} = \langle 0,\ 2^n - 1 \rangle \tag{1}$$

Standardowo w systemach komputerowych stosowane są następujące liczby bitów, z których wynikają poniższe zakresy liczb:

- 8 bitów, zakres liczb: 0 … 255;
- 16 bitów, zakres liczb: 0 … 65 535;
- 32 bity, zakres liczb: 0 … 4 294 967 295;
- 64 bity, zakres liczb: 0 … 18 446 744 073 709 551 615.

Do konwersji liczby dziesiętnej na kod NKB wystarczy jej zwykła zamiana na system dwójkowy.

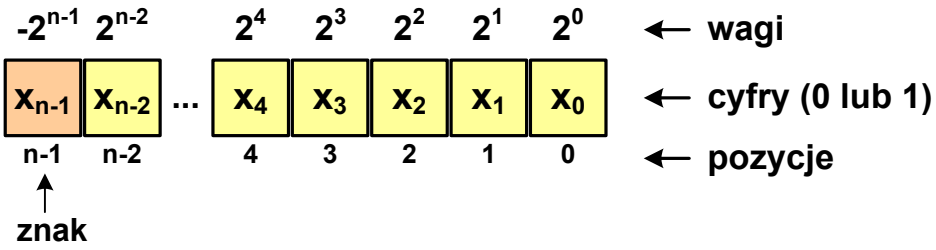
Przykład: zamiana liczby $93_{(10)}$ na kod NKB

$$93_{(10)} = ?_{(NKB)} \qquad 93_{(10)} = 1011101_{(NKB)}$$

$$\begin{aligned} 93/2 &= 46 \quad reszta \quad 1 \\ 46/2 &= 23 \quad reszta \quad 0 \\ 23/2 &= 11 \quad reszta \quad 1 \\ 11/2 &= 5 \quad reszta \quad 1 \\ 5/2 &= 2 \quad reszta \quad 1 \\ 2/2 &= 1 \quad reszta \quad 0 \\ 1/2 &= 0 \quad reszta \quad 1 \end{aligned}$$

W przypadku liczb całkowitych ze znakiem stosowane są trzy metody kodowania: **ZM**, **U1** i **U2**. W systemach komputerowych powszechnie stosowany jest kod **U2**.

W kodzie **U2** (ZU2, uzupełnień do dwóch, two's complement) najstarszy bit jest bitem znaku liczby: 0 - liczba dodatnia, 1 - liczba ujemna. Wszystkie bity liczby posiadają takie same wagi jak w NKB, oprócz najstarszego bitu, który ma wagę -2^{n-1} (Rys. 14).



Rys. 14. Kod U2

W kodzie U2 liczby dodatnie zapisywane są tak samo jak w NKB, zaś liczby ujemne otrzymywane są poprzez bitową negację i dodanie wartości 1.

Przykład: zamiana liczby $93_{(10)}$ na kod U2

- zamieniamy liczbę na NKB: $93_{(10)} = 1011101_{(NKB)}$
- dopisujemy bit znaku (0): $93_{(10)} = 01011101_{(U2)}$

Przykład: zamiana liczby $-93_{(10)}$ na kod U2

- zamieniamy moduł liczby na U2: $|-93_{(10)}| = 93_{(10)} = 01011101_{(U2)}$
- negujemy wszystkie bity: $-93_{(10)} = 10100010_{(U1)}$
- dodajemy 1 do wyniku: $-93_{(10)} = 10100011_{(U2)}$

2.7. Liczby zmiennoprzecinkowe

Zapis bardzo dużych lub małych liczb rzeczywistych wymaga dużej liczby cyfr. Znacznie prostsze jest przedstawienie liczb w postaci zmiennoprzecinkowej (ang. floating point numbers), np.

$12\ 000\ 000\ 000\ 000 = 1{,}2 \cdot 10^{13}$

$0{,}000\ 000\ 000\ 001 = 1{,}0 \cdot 10^{-12}$

Ogólny zapis liczby zmiennoprzecinkowej ma postać:

$$L = M \cdot B^E \tag{2}$$

gdzie: L - wartość liczby, B - podstawa systemu, M - mantysa, E - wykładnik.

Przykłady:

$$2{,}43 \cdot 10^{3}_{(10)} = 2{,}43 \cdot 1000 = 2430_{(10)}$$

$$6{,}59 \cdot 10^{-2}_{(10)} = 6{,}59 \cdot 0{,}01 = 0{,}0659_{(10)}$$

$$1{,}011 \cdot 10^{101}_{(2)} = ?_{(10)}$$

$$M = 1{,}011_{(2)} = 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1{,}375_{(10)}$$

$$B = 10_{(2)} = 0 \cdot 2^0 + 1 \cdot 2^1 = 2_{(10)}$$

$$E = 101_{(2)} = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 = 1 + 4 = 5_{(10)}$$

$$1{,}011 \cdot 10^{101}_{(2)} = 1{,}375 \cdot 2^5 = 1{,}375 \cdot 32 = 44_{(10)}$$

$$3{,}121 \cdot 10^{32}_{(4)} = ?_{(10)}$$

$$M = 3{,}121_{(4)} = 3 \cdot 4^0 + 1 \cdot 4^{-1} + 2 \cdot 4^{-2} + 1 \cdot 4^{-3} = 3{,}390625_{(10)}$$

$$B = 10_{(4)} = 0 \cdot 4^0 + 1 \cdot 4^1 = 4_{(10)}$$

$$E = 32_{(4)} = 2 \cdot 4^0 + 3 \cdot 4^1 = 2 + 12 = 14_{(10)}$$

$$3{,}121 \cdot 10^{32}_{(4)} = 3{,}390625 \cdot 4^{14} = 910\ 163\ 968_{(10)}$$

Położenie przecinka w mantysie nie jest ustalone i może się zmieniać. Poniższe zapisy oznaczają tę samą liczbę (system dziesiętny):

$$243 \cdot 10^{1} = 24{,}3 \cdot 10^{2} = 2{,}43 \cdot 10^{3} = 0{,}243 \cdot 10^{4}$$

Dla ujednolicenia zapisu i usunięcia wielokrotnych reprezentacji tej samej liczby, przyjęto tzw. **postać znormalizowaną** zapisu liczby zmiennoprzecinkowej. W postaci znormalizowanej mantysa spełnia nierówność:

$$B > |M| \geq 1 \tag{3}$$

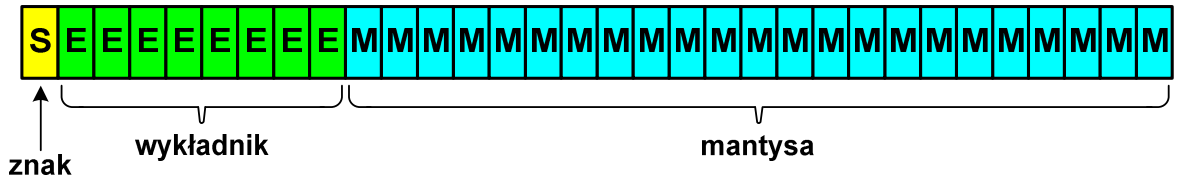
Przykład:

$2{,}43 \cdot 10^{3}$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

$0{,}243 \cdot 10^{4}$ - to nie jest postać znormalizowana

$24{,}3 \cdot 10^{2}$ - to nie jest postać znormalizowana

Jeśli liczba zmiennoprzecinkowa przechowywana jest w systemie binarnym to liczba bitów przeznaczonych na mantysę i wykładnik jest ograniczona. W systemie binarnym przechowywany jest znak liczby, wykładnik i mantysa (Rys. 15). Podstawa systemu jest stała (**B = 2**) i nie musi być przechowywana.



Rys. 15. Reprezentacja liczby zmiennoprzecinkowej w systemie binarnym

W praktycznych implementacjach liczb zmiennoprzecinkowych wykładnik zapisywany jest z przesunięciem (ang. bias). Właściwą wartość wykładnika otrzymuje się poprzez odjęcie od zakodowanego wykładnika wartości przesunięcia. Zatem wzór określający wartość liczby zmiennoprzecinkowej ma postać:

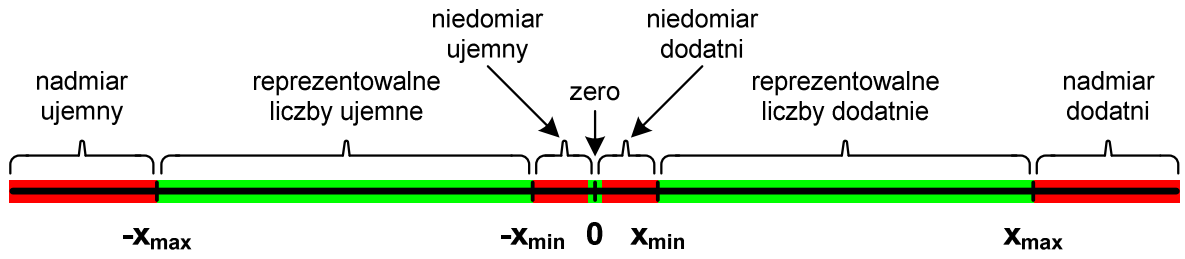
$$L = (-1)^{S} \cdot M \cdot 2^{E-BIAS} \tag{4}$$

gdzie: L - wartość liczby, S - znak liczby, M - mantysa, E - wykładnik, BIAS - przesunięcie (nadmiar).

Typowe wartości przesunięcia wynoszą:

- format 32-bitowy: $2^7-1 = 127_{(10)} = 7F_{(16)}$
- format 64-bitowy: $2^{10}-1 = 1023_{(10)} = 3FF_{(16)}$
- format 80-bitowy: $2^{14}-1 = 16383_{(10)} = 3FFF_{(16)}$

W zapisie zmiennoprzecinkowym nie można reprezentować wszystkich liczb. Wynika to z ograniczonej liczby bitów przeznaczonych na wykładnik i mantysę. Na Rys. 16 kolorem zielonym oznaczono przedział liczb, które można reprezentować, zaś kolorem czerwonym - przedział, którego reprezentacja nie jest możliwa. Na poniższym rysunku x_{min} jest to najmniejsza wartość możliwa do zapisania w danej reprezentacji, zaś x_{max} - największa wartość możliwa do zapisania w danej reprezentacji.



Rys. 16. Zakres liczb w zapisie zmiennoprzecinkowym

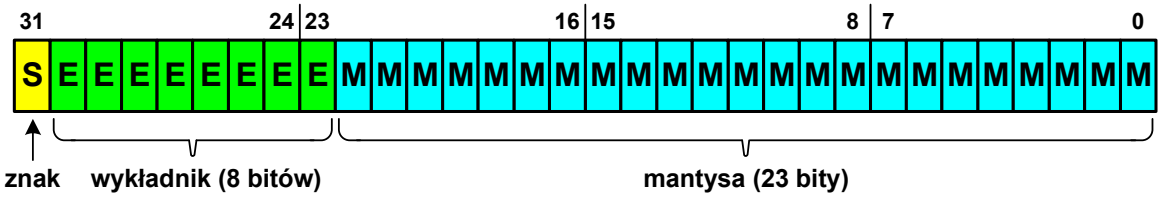
Niedomiar występuje, gdy wielkość ułamkowa jest zbyt mała - zazwyczaj jest wtedy aproksymowana przez zero. Nadmiar występuje, gdy wynik operacji zmiennoprzecinkowej jest większy od x_{max} lub mniejszy od $-x_{max}$.

W celu ujednolicenia operacji na liczbach zmiennoprzecinkowych na różnych platformach sprzętowych opracowano odpowiedni standard - IEEE Std. 754-2008 - IEEE Standard for Floating-Point Arithmetic. Obecnie praktycznie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych oparte są o ten standard. Tabela 3 przedstawia klasy liczb zmiennoprzecinkowych zdefiniowane w standardzie IEEE 754.

Tabela 3. Klasy liczb zmiennoprzecinkowych w standardzie IEEE 754

Precyzja	Długość słowa (bity)	Znak (bity)	Wykładnik		Mantysa	
			Długość (bity)	Zakres	Długość (bity)	Cyfry znaczące
Pojedyncza (ang. single)	32	1	8	$2^{\pm 127} \approx 10^{\pm 38}$	23	7
Pojedyncza rozszerzona (ang. single extended)	≥ 43	1	≥ 11	$\geq 2^{\pm 1023} \approx 10^{\pm 308}$	≥ 31	≥ 10
Podwójna (ang. double)	64	1	11	$2^{\pm 1023} \approx 10^{\pm 308}$	52	16
Podwójna rozszerzona (ang. double extended)	≥ 79	1	≥ 15	$\geq 2^{\pm 16383} \approx 10^{\pm 4932}$	≥ 63	≥ 19

Liczba **pojedynczej precyzji** przechowywana jest na 32 bitach (Rys. 17). Pierwszy bit w zapisie (bit nr 31) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna). Wykładnik zapisywany jest na 8 bitach (bity nr 30-23) z nadmiarem o wartości 127. Wykładnik może przyjmować wartości od -127 (wszystkie bity wyzerowane) do 128 (wszystkie bity ustawione na 1). Mantysa zapisywana jest na 23 bitach w kodzie U1, w większości przypadków jest znormalizowana. Wartość mantysy zawiera się pomiędzy 1 a 2, a zatem w zapisie liczby pierwszy bit jest zawsze równy 1. Powyższy bit nie jest zapamiętywany, natomiast jest automatycznie uwzględniany podczas wykonywania obliczeń. Dzięki pominięciu tego bitu zyskujemy dodatkowy bit mantysy (zamiast 23 bitów mamy 24 bity).



Rys. 17. Liczba pojedynczej precyzji w standardzie IEEE 754

Przykład: obliczenie wartość dziesiętnej liczby zmiennoprzecinkowej

$$01000010110010000000000000000000_{(\text{IEEE}\,754)} = ?_{(10)}$$

- dzielimy liczbę na części:

$$\underbrace{0}_{\text{S}-\text{bit znaku}} \quad \underbrace{10000101}_{\text{E}-\text{wykladnik}} \quad \underbrace{10010000000000000000000}_{\text{M}-\text{mantysa (tylko czesc ulamkowa)}}$$

- określamy znak liczby:

$$S = 0 \quad - \text{liczba dodatnia}$$

- obliczamy wykładnik pamiętając, że w reprezentacji 32-bitowej nadmiar to 127:

$$E = 10000101_{(2)} = 128 + 4 + 1 = 133 - \underbrace{127}_{\text{nadmiar}} = 6_{(10)}$$

- wyznaczamy mantysę dopisując na początku 01 (0 - znak liczby w kodzie U1, 1 - część całkowita) i stawiając przecinek:

$$M = 01{,}10010000000000000000000_{(U1)} =$$
$$= 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-4} = 1 + 0{,}5 + 0{,}0625 = 1{,}5625_{(10)}$$

- wartość dziesiętną liczby zmiennoprzecinkowej obliczamy według wzoru:

$$L = (-1)^S \cdot M \cdot 2^E$$

- podstawiając otrzymujemy:

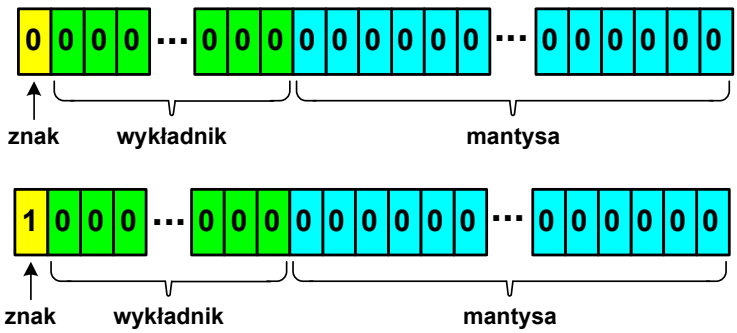
$$L = (-1)^0 \cdot 1{,}5625 \cdot 2^6 = 100_{(10)}$$

- ostatecznie:

$$01000010110010000000000000000000_{(\text{IEEE}\,754)} = 100_{(10)}$$

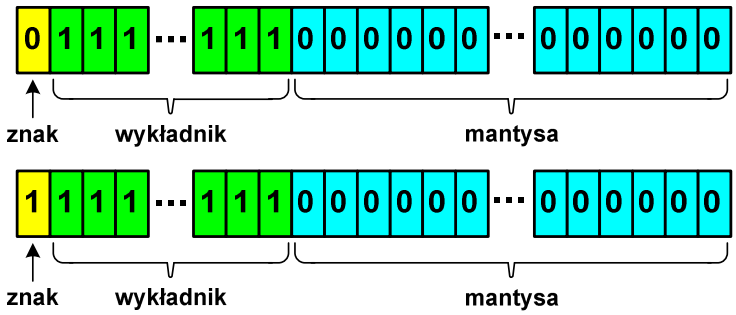
W standardzie IEEE 754 zdefiniowano także wartości specjalne takie jak: zero, nieskończoność, liczba zdenormalizowana, nieliczba.

W przypadku **zera** (Rys. 18) bit znaku może przyjmować dowolną wartość (zero dodatnie i ujemne). Wszystkie bity wykładnika i mantysy są równe zeru. Przy porównaniach zero dodatnie i ujemne są traktowane jako równe sobie.



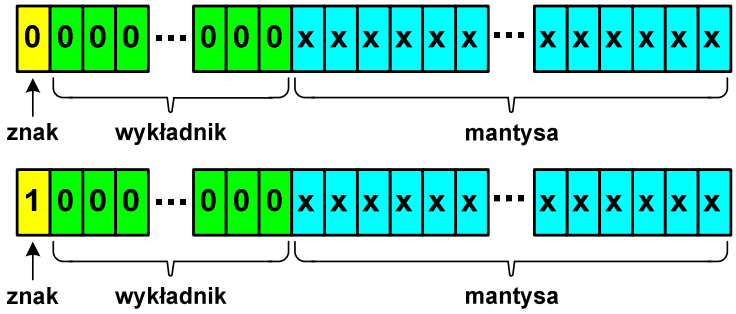
Rys. 18. Reprezentacja zera w standardzie IEEE 754

Bit znaku określa czy mamy **nieskończoność** dodatnią czy ujemną (Rys. 19). Wszystkie bity wykładnika są równe jeden, zaś wszystkie bity mantysy - zero. Nieskończoność występuje w przypadku wystąpienia nadmiaru (przepełnienia) oraz przy dzieleniu przez zero.



Rys. 19. Reprezentacja nieskończoności w standardzie IEEE 754

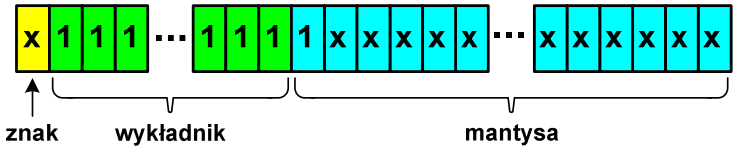
W przypadku **liczby zdenormalizowanej** (Rys. 20) bit znaku może być równy zero lub jeden, wszystkie bity wykładnika są równe zeru, zaś bity mantysy przyjmują dowolne wartości. Liczba ta pojawia się, gdy występuje niedomiar (ang. underflow), ale wynik operacji można jeszcze zapisać denormalizując mantysę. Wtedy mantysa nie posiada domyślnej części całkowitej równej 1, tzn. reprezentuje liczbę o postaci 0,xxx…xxx, a nie 1,xxx…xxx.



Rys. 20. Reprezentacja liczby zdenormalizowanej w standardzie IEEE 754

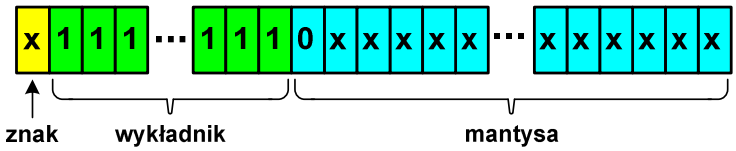
W standardzie IEEE 754 zdefiniowane są także dwie specjalne wartości, które nie reprezentują wartości liczbowej. Wartości te nazywane są **NaN** (ang. Not A Number - nie liczba). Powstają w wyniku wykonania niedozwolonej operacji, np. obliczania pierwiastka z liczby ujemnej, dzielenie zera przez zero. Wyróżnia się dwa rodzaje nieliczb.

QNaN (ang. Quiet NaN) są to tzw. ciche nieliczby (Rys. 21). Wszystkie bity wykładnika oraz pierwszy bit mantysy są równe jeden, pozostałe bity mantysy mają dowolną wartość. Ciche nieliczby „przechodzą" przez działania arytmetyczne (ich wystąpienie nie powoduje przerwania wykonywania programu). Najczęściej oznaczają wartość niezdefiniowaną.



Rys. 21. Cicha nieliczba w standardzie IEEE 754

SNaN (ang. Signaling NaN) czyli sygnalizujące, istotne, głośne nieliczby (Rys. 22). Wszystkie bity wykładnika są równe jeden, pierwszy bit mantysy ma wartość zero, zaś pozostałe bity - dowolną wartość. Powodują powstanie wyjątków w operacjach arytmetycznych i przerwanie wykonywania programu. Najczęściej oznaczają wartość niedozwoloną.



Rys. 22. Głośna nieliczba w standardzie IEEE 754

3. Przebieg ćwiczenia

Wykonaj podane poniżej zadania. Rozwiązania zadań oddaj na koniec zajęć w formie sprawozdania. Szablon sprawozdania wskaże prowadzący zajęcia.

1. Podaj postać zapisu liczb od 0 do 15 w siódemkowym i trzynastkowym systemie liczbowym.

2. Na podstawie liczby liter swojego nazwiska (x) i liczby liter swojego imienia (y) utwórz i zapisz liczbę zgodnie ze wzorem: $x \cdot 10+y$. Podaj postać tej liczby w systemie dwójkowym i dwunastkowym.

3. Dana jest liczba $765_{(8)}$. Podaj postać tej liczby w systemie dwójkowym, czwórkowym, dziesiętnym i szesnastkowym.

4. Podaj wartość następujących liczb w systemie dziesiętnym: $100101_{(2)}$, $231_{(5)}$, $654_{(8)}$, $A91_{(12)}$.

5. Stosując mnożnik używany w informatyce (1024) zamień 5 GB i 5 MB na kilobajty.

6. Odczytaj poniższe zdanie zapisane w postaci kodów standardu ISO-8859-2 (system szesnastkowy).

Kod	5A	61	BF	F3	B3	E6	20	67	EA	B6	6C	B1	20	6A	61	BC	F1	2E
Znak																		

   Umieść w sprawozdaniu powyższą tabelę.

7. Podaj kody standardu ISO-8859-2 odpowiadające znakom Twojego imienia i nazwiska (w systemie dziesiętnym lub szesnastkowym). Przykład:

Imię	J	a	n
Kod			

Nazwisko	K	o	W	a	l	s	k	i
Kod								

8. Liczbę z Zadania nr 2 zamień na ujemną i zapisz ją w systemie dwójkowym w kodzie U2.

9. Zapisz podane liczby w postaci zmiennoprzecinkowej znormalizowanej (system dziesiętny): 1234,56 0,000786 -0,0234 -10000.

4. Literatura

[1] Gryś S.: Arytmetyka komputerów w praktyce. PWN, Warszawa, 2013.

[2] Coldwin G.: Zrozumieć programowanie. PWN, Warszawa, 2020.

[3] Kawa R., Lembas J.: Wykłady z informatyki. Wstęp do informatyki. PWN, Warszawa, 2021.

[4] Pochopień B.: Arytmetyka systemów cyfrowych. Wydawnictwo Politechniki Śląskiej, Gliwice, 2003.

[5] Kwiatkowski W.: Wprowadzenie do kodowania. BEL Studio, Warszawa, 2010.

5. Pytania kontrolne

1. Wyjaśnij pojęcie system liczbowy. Jaka jest różnica pomiędzy pozycyjnym i niepozycyjnym systemem liczbowym?
2. Opisz konstrukcję liczb w dowolnym pozycyjnym systemie liczbowym.
3. Omów metody konwersji liczb z systemu dziesiętnego na system o dowolnej podstawie i odwrotnie.
4. Scharakteryzuj podstawowe jednostki informacji: bit i bajt.
5. Scharakteryzuj kody: ASCII, ISO 8859, Unicode.
6. Opisz sposób zamiany liczby ujemnej na kod U2.
7. Omów sposób przechowywania liczb zmiennoprzecinkowych w standardzie IEEE 754.
8. Wymień wartości specjalne występujące w standardzie IEEE 754.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.

- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.