

# **Podstawy Systemów Liczbowych**

---

***Wersja: 1.0***

***Będzin, dn. 03-11-2010 r.***

© Copyright by Krzysztof Kryczka (gSystem)

Data: 03.11.2010

## Wydanie I

**Darmowy poradnik**, dostarczony dla serwisu **HAKERZY.NET**

### UWAGA!!!

Niniejszy poradnik jest objęty prawem autorskim i nie może być kopiowany lub rozpowszechniany na innych stronach bez wiedzy i zgody autora.

Zabrania się również jego odsprzedaży.

Jeśli ktoś znajdzie błąd w poradniku „Podstawy Systemów Liczbowych” proszę mnie o tym fakcie poinformować na mój adres mailowy, wpisując w temacie maila kod **22ce9303** i opisując w treści błąd. Maile, które będą niepoprawnie nazwane lub będą dotyczyły innych kwestii będę ignorował!

Kontakt:

WWW: <http://www.hakerzy.net>

EMAIL: [gsystem@hakerzy.net](mailto:gsystem@hakerzy.net)

Facebook: [facebook.com/krzysztof.kryczka](https://www.facebook.com/krzysztof.kryczka)

## Spis Treści

<b>Od Autora</b>	<b>5</b>
<b>Teoria informacji</b>	<b>5</b>
<b>Tworzenie liczb</b>	<b>6</b>
<b>System dwójkowy</b>	<b>8</b>
<b>Konwersja liczb dziesiętnych</b>	<b>13</b>
1. Zamiana liczby $185,625_{(10)}$ na dwójkową (bin)	13
2. Zamiana liczby $185,625_{(10)}$ na ósemkową (oct)	14
3. Zamiana liczby $185,625_{(10)}$ na szesnastkową (hex)	15
<b>Konwersja liczb binarnych</b>	<b>16</b>
1. Zamiana liczby $10111001,101_{(2)}$ na dziesiętną	16
2. Zamiana liczby $110111001_{(2)}$ na ósemkową	17
3. Zamiana liczby $111110111000101,11_{(2)}$ na szesnastkową	18
<b>Działania arytmetyczne</b>	<b>19</b>
1. Dodawanie binarne	19
2. Odejmowanie binarne	20
3. Mnożenie binarne	21
4. Dzielenie binarne	22
<b>ARYTMETYKA STAŁOPRZECINKOWA I ZMIENNOPRZECINKOWA</b>	<b>23</b>
<b>NOTATKI</b>	<b>24</b>

## Od Autora

---

Wiele osób związanych zawodowo z dziedziną informatyki nie potrafi posługiwać się systemami liczbowymi – binarnymi, oktalnymi czy heksadecymalnymi o innych nie wspominając. Korzystają z kalkulatorów lub specjalnych programów jednak nie wiedzą jak to policzyć na „kartce”. Na forach internetowych można znaleźć bardzo wiele wątków gdzie padają pytania, jak dodawać czy odejmować binarnie, jak przekonwertować wybraną liczbę między systemami i inne elementarne pytania.

Aby pomóc wszystkim naszym czytelnikom będą systematycznie wydawane poradniki, które pozwolą uzupełnić luki w wiedzy, a także usystematyzować nabytą wiedzę. W lewym górnym rogu okładki znajduje się numer poradnika oraz kolor. Tło zielone to poradniki dla początkujących, pomarańczowe dla osób średnio-zaawansowanych natomiast na tle czerwonym będą omawiane zagadnienia zaawansowane.

W pierwszym poradniku z serii dla **początkujących** omówimy:

- systemy liczbowe (dwójkowe, ósemkowe i szesnastkowe),
- strukturę reprezentacji danych w pamięci komputera,
- zasady realizacji wybranych działań arytmetycznych.

## Teoria informacji

---

Ludzie posługują się informacją. Informacja jest przekazywana za pomocą słów i symboli. Symbolami są litery, cyfry, znaki specjalne czy rysunki. Rozróżniamy ważniejsze informacje i te mniej ważne, podobnie komputer również operuje na informacji, a konkretniej wykonuje pewne zadania, które mu wydamy. Jednak nie rozumie naszego języka dlatego powstają różne języki programowania (łatwiejsze i trudniejsze), które pozwalają nam za pomocą pewnych kluczowych zarezerwowanych słów porozumieć się z komputerem. Słowa te są umieszczane przez nas w algorytmach, które są następnie kompilowane do programów, czyli tłumaczone na język maszyny cyfrowej. Ułatwia to znacznie korzystanie z komputera ponieważ

gotowy program jest w stanie obsłużyć prawie każdy użytkownik komputera - natomiast zaprojektować już niekoniecznie.

Informacja jest zapisywana w postaci binarnej. Dlatego też, aby zakodować informacje lub zaprezentować je wykorzystuje się do tego celu systemy liczbowe. Można stworzyć praktycznie dowolny system pozycyjny o podstawie np. 2, 3, 4, ..., 8, 9, 10, ..., 16, itd. Dla nas ludzi najbardziej popularny jest system liczbowy o podstawie 10, tzw. dziesiętny. Jednak był on bardzo trudny do implementacji w maszynach cyfrowych, dlatego też komputery używają:

- systemu dwójkowego (binarnego) dla przechowywania i przetwarzania danych przez układy elektroniczne;
- systemu szesnastkowego (heksadecymalnego) do prezentacji niektórych danych np. adresów komórek pamięci.

## Tworzenie liczb

W celu utworzenia liczby w dowolnym systemie liczbowym pozycyjnym wykorzystuje się wzór:

$$X = \sum_{i=-m}^{n-1} x_i p^i = (x_{n-1} \cdot p^{n-1} + x_{n-2} \cdot p^{n-2} \dots + x_0 \cdot p^0, + x_{-1} \cdot p^{-1} + \dots + x_{-m} \cdot p^{-m}) = C_x, U_x$$

gdzie:

$p$  – podstawa systemu liczbowego  $p \in \{2, 3, \dots, 8, \dots, 16, \dots\}$

$x_i$  – cyfra  $i$ -tej pozycji  $x_i \in \{0, 1, 2, \dots, p-1\}$

$n-1$  – ilość cyfr części całkowitej liczby  $X$

$m$  – ilość cyfr części ułamkowej liczby  $X$

$x_{n-1}$  – cyfra najbardziej znacząca

$x_{-m}$  – cyfra najmniej znacząca

$C_x$  – część całkowita liczby  $X$

$U_x$  – część ułamkowa liczby  $X$

, – przecinek oddziela te części od siebie

Końcowa wartość liczby zależy od tego, gdzie znajduje się cyfra, która tworzy liczbę.

**Przykład wykorzystania wzoru:**

➤ **System dziesiętny:**

$p = 10$  ;  $x_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  ;

gdzie:

$p$  – podstawa systemu liczbowego

$x_i$  – zakres cyfr, na których operuje wybrany system

Liczbę  $377,75_{(10)}$  można rozpisać następująco:

$_{(10)}$  – liczba w systemie o podstawie  $p$

$10^2$	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$
$x_2$	$x_1$	$x_0$	$x_{-1}$	$x_{-2}$
↓	↓	↓	↓	↓
3	7	7	7	5

$C_x$  – część całkowita liczby  $X$        $U_x$  – część ułamkowa liczby  $X$

$$X = 3 * 10^2 + 7 * 10^1 + 7 * 10^0 + 7 * 10^{-1} + 5 * 10^{-2} = 300 + 70 + 7 + 0,7 + 0,05 = 377,75_{(10)}$$

➤ **System dwójkowy (binarny):**

$p = 2$  ;  $x_i \in \{0, 1\}$  ;

Liczbę  $101111001,11_{(2)}$  można rozpisać następująco:

$x_8$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$x_{-1}$	$x_{-2}$
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	0	1	1	1	1	0	0	1	1	1

$$X = 1*2^8 + 0*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} = 256 + 64 + 32 + 16 + 8 + 1 + 0,5 + 0,25 = 377,75_{(10)}$$

## System dwójkowy

W systemie o podstawie równej 2 mamy do dyspozycji dwa symbole.

$$p = 2 ; x_i \in \{0, 1\} ;$$

Tworzą one tzw. bit  $[b]$ , który przyjmuje tylko dwie wartości np. prawdę, fałsz. Za pomocą jednego bitu można przekazać dwie różne informacje. Aby przekazać więcej informacji musimy utworzyć grupy bitów.

1 bit => 2 informacje ( $2^1=2$ )		
	$b_0$	
stan bitu 0	0	Informacja 1
stan bitu 1	1	Informacja 2

2 bity => 4 informacje ( $2^2=4$ )			
	$b_1$	$b_0$	
stan bitu 00	0	0	Informacja 1
stan bitu 01	0	1	Informacja 2
stan bitu 10	1	0	Informacja 3
stan bitu 11	1	1	Informacja 4

$(2^3=8 \text{ informacji})$				
	$b_2$	$b_1$	$b_0$	
stan bitu 000	0	0	0	Informacja 1
stan bitu 001	0	0	1	Informacja 2
stan bitu 010	0	1	0	Informacja 3
stan bitu 011	0	1	1	Informacja 4
stan bitu 100	1	0	0	Informacja 5
stan bitu 101	1	0	1	Informacja 6
stan bitu 110	1	1	0	Informacja 7
stan bitu 111	1	1	1	Informacja 8

Wybrane grupy bitów mają nadane nazwy co zostało przedstawione w tabeli.

Nazwa	Ilość bajtów	Ilość bitów	Ilość informacji	Zakres
<b>Tetrada (nibble)</b>	0,5	4	$2^4 = 16$	
<b>Bajt</b>	1	8	$2^8 = 256$	-128 ... +127 (ze znakiem) 0 ... +255 (bez znaku)
<b>Słowo</b>	2	16	$2^{16} = 65\,536$	-32 768 ... +32 767 (ze znakiem) 0 ... +65 535 (bez znaku)
<b>Podwójne słowo</b>	4	32	$2^{32} = 4\,294\,967\,296$	-2 147 483 648 ... +2 147 483 647 (ze znakiem) 0 ... +4 294 967 295 (bez znaku)
<b>Poczwórne słowo</b>	8	64	$2^{64} = 18\,446\,744\,073\,709\,551\,616$	-9 223 372 036 854 775 808 ... +9 223 372 036 854 775 807 (ze znakiem) 0 ... +18 446 744 073 709 551 615 (bez znaku)

**Tabela 1** - Wybrane jednostki pamięci i zakres zmiennych

W komputerach najczęściej operujemy na wewnętrznych rejestrach. Pozwalają one zapamiętywać i operować na grupach bitów. Najbardziej znany z tej grupy jest bajt. Z tabeli możemy wyczytać, że zawiera on 8 bitów, czyli składa się z dwóch tetrad (nibble). Za jego pomocą możemy przedstawić aż 256 różnych wartości, czyli reprezentuje on wartości od 0000 0000 do 1111 1111 binarnie. Wymaga to aż dwóch cyfr heksadecymalnych, a ponieważ największą cyfrą w heksach jest \$0F czyli 15 to wartość \$FF daje nam 255.

Bajty są grupowane w kolejne grupy co zaprezentowano w tabeli 2. Tworzą one jednostki danych i są bardzo często wykorzystywane przy określaniu pojemności nośników danych takich jak dyski twarde, płyty DVD, pendrive, karty pamięci, itp.



## Wielokrotności bajtów

Nazwa	Symbol	Mnożnik
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	$2^{50}=1024^5$

Tabela 2 – Bajty w grupach

Wiemy już, że w bajcie można zapamiętać zbiór 255 informacji (symboli) to znaczy, że składając bajty w pewnej długości ciąg można zapamiętywać nie tylko liczby, ale również litery, znaki lub całe zdania. Wymyślono tablicę ASCII (*American Standard Code for Information Interchange*), w której został zapisany podstawowy zbiór symboli. Niestety nie zawiera on narodowych symboli diaktrycznych. Aby zakodować tego typu znaki dla języka polskiego są to np. ą, ę, ż, ź, ć, ń, ś ... należy posłużyć się tablicami UNICODE. Są to specjalne tablice, które zawierają wszystkie tego typu znaki ze wszystkich języków świata.

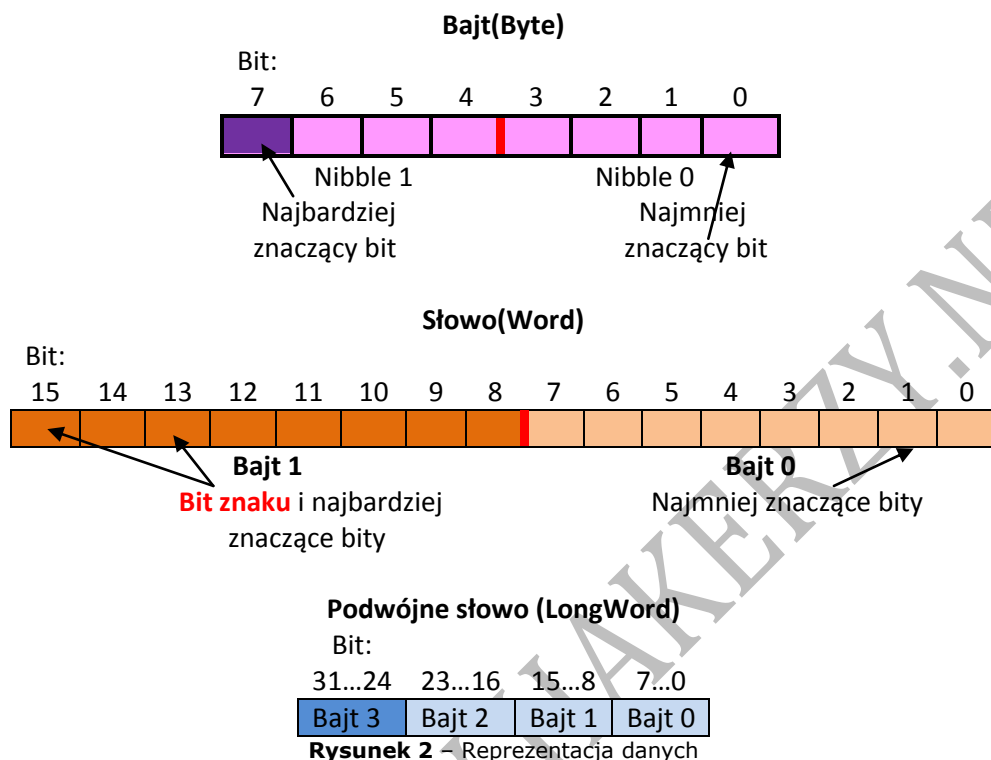
Przykład tabeli ASCII z zapisem podstawowych symboli został przedstawiony na rysunku

1. Oprócz znaczenia (char, chr) jest również przedstawiona wartość w systemach (Dec - 10), (HeX-16), (Oct-8) i niektóre w języku opisu strony jakim jest HTML.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	&	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	70	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	71	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	72	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	73	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	74	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	75	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	76	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	77	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	78	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	79	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	80	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	81	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	82	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	83	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	84	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	85	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	86	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	:	91	5B	133	&#91;	[	123	87	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	88	174	&#124;	}
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	89	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	90	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	91	177	&#127;	DEL

Rysunek 1 – Tablica ASCII

Natomiast poniżej zestawiono kilka przykładów wewnętrznej reprezentacji danych.



Na rysunku zaznaczono bit zerowy, otóż jest to najmniej znaczący bit. Każdy kolejny bit na lewo jest o numer większy. Bit, który znajduje się najbardziej z lewej strony jest najbardziej znaczący. Ma to ostatecznie wpływ na wartość bajtu, czy słowa. Można to łatwo zapamiętać, gdyż zapewne każdy woli 1000 zł na koncie niż 0001 zł.

Oczywiście na pojedynczych bitach możemy wykonywać różne operacje, służą nam do tego operatory bitowe takie jak:

- << przesunięcie w lewo,
- >> przesunięcie w prawo,
- & Bitowy iloczyn (AND),
- | Bitowa suma (OR),
- ^ Bitowa różnica symetryczna (XOR)
- ~ Bitowa negacja (NOT)

<b>AND</b>	<b>Jeśli jeden z bitów jest zerem to wynik jest równy 0</b>
$0 \& 0 = 0$	
$0 \& 1 = 0$	
$1 \& 0 = 0$	
$1 \& 1 = 1$	
<b>OR</b>	<b>Jeśli jeden z bitów jest jedynką to wynik jest równy 1</b>
$0   0 = 0$	
$0   1 = 1$	
$1   0 = 1$	
$1   1 = 1$	
<b>XOR</b>	<b>Jeśli oba bity są równe to wynik jest równy 0</b>
$0 \wedge 0 = 0$	
$0 \wedge 1 = 1$	
$1 \wedge 0 = 1$	
$1 \wedge 1 = 0$	
<b>NOT</b>	<b>Zero staje się jedynką Jedynka zerem</b>
$\sim 0 = 1$	
$\sim 1 = 0$	
<b>przesunięcie w lewo</b>	<b>Bity zostają przesunięte o 3 miejsca w lewo, Bity z prawej strony zostają uzupełnione zerami, Bity z lewej strony zostają odrzucone.</b>
0100 0000 1111 0010 << 3	
0100 0000 1111 0010 000 0000 0111 1001 0000	
<b>przesunięcie w prawo</b>	<b>Bity zostają przesunięte o 3 miejsca w prawo, Bity z lewej strony zostają uzupełnione zerami, Bity z prawej strony zostają odrzucone.</b>
0001 1111 1111 0000 >> 3	
0000 0011 1111 1100 000 0000 0011 1111 1100	

Tabela 3 – Operatory bitowe

Komputer wykonuje także operacje dodawania, odejmowania, mnożenia, dzielenia służą do tego operatory arytmetyczne. W zasadzie odbywa się to w sposób zautomatyzowany, dlatego warto poznać podstawy arytmetyki.

## Konwersja liczb dziesiętnych

### 1. Zamiana liczby $185,625_{(10)}$ na dwójkową (bin)

Liczbę dzielimy przez 2 zapisując część całkowitą i resztę z dzielenia. Strzałka wskazuje kierunek odczytu. Reszta z dzielenia = 1, brak reszty = 0.

a) Liczmy część całkowitą:

Działanie:	Reszta:
$185 : 2 = 92$	r 1 ↑
$92 : 2 = 46$	r 0
$46 : 2 = 23$	r 0
$23 : 2 = 11$	r 1
$11 : 2 = 5$	r 1
$5 : 2 = 2$	r 1
$2 : 2 = 1$	r 0
$1 : 2 = 0$	r 1
0 (KONIEC)	

$$185_{(10)} = 10111001_{(2)}$$

b) Liczmy część ułamkową:

Działanie:	Część ułamkowa	Część całkowita
$0,625 * 2 = 1,25$	0,25	1
$0,25 * 2 = 0,50$	0,50	0
$0,5 * 2 = 1,00$	0,00	1
0 (KONIEC)		



$$0,625_{(10)} = 101_{(2)}$$

$$\text{Ostatecznie: } 185,625_{(10)} = 10111001,101_{(2)}$$

## 2. Zamiana liczby $185,625_{(10)}$ na ósemkową (oct)

Podobnie jak w pierwszym przykładzie musimy podzielić liczbę, tylko że tym razem nie przez 2, a przez 8.

a) Liczymy część całkowitą:

Działanie:	Część ułamkowa:	Dopełnienie:	Reszta:
$185 : 8 = 23$	0,125	$23 \cdot 8 + 1$	r 1
$23 : 8 = 2$	0,875	$2 \cdot 8 + 7$	r 7
$2 : 8 = 0$	0,25	$0 \cdot 8 + 2$	r 2
0 (KONIEC)			

$$185_{(10)} = 271_{(8)}$$

b) Liczymy część ułamkową:

Działanie:	Część ułamkowa	Część całkowita
$0,625 \cdot 8 = 5,00$	0,00	5
0 (KONIEC)		

$$0,625_{(10)} = 5_{(8)}$$

$$\text{Ostatecznie: } 185,625_{(10)} = 271,5_{(8)}$$

### 3. Zamiana liczby $185,625_{(10)}$ na szesnastkową (hex)

a) Liczymy część całkowitą:

Działanie:	Część ułamkowa:	Dopełnienie:	Reszta:
$185 : 16 = 11$	0,5625	$11 * 16 + 9$	r 9
$11 : 16 = 0$	0,6875	$0 * 16 + 11$	r B
0 (KONIEC)			



$$185_{(10)} = B9_{(16)}$$

b) Liczymy część ułamkową:

Działanie:	Część ułamkowa	Część całkowita
$0,625 * 16 = 10,00$	0,00	10 = A
0 (KONIEC)		



$$0,625_{(10)} = A_{(16)}$$

$$\text{Ostatecznie: } 185,625_{(10)} = B9,A_{(16)}$$

## Konwersja liczb binarnych

### 1. Zamiana liczby $10111001,101_{(2)}$ na dziesiętną

Rozpisujemy liczbę na pojedyncze bity i odczytujemy wartości potęg z tabeli.

$$\begin{array}{cccccccccccc}
 x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & & x_{-1} & x_{-2} & x_{-3} \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\
 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & , & 1 & 0 & 1
 \end{array}$$

$$10111001,101_{(2)} = 128 + 32 + 16 + 8 + 1 + 0,5 + 0,125 = 185,625_{(10)}$$

n-index	$x_n$	$x_n$
0	1	1,0
1	2	0,5
2	4	0,25
3	8	0,125
4	16	0,0625
5	32	0,03125
6	64	0,015625
7	128	
8	256	
9	512	
10	1024	
11	2048	
12	4096	
13	8192	
14	16384	
15	32768	
16	65536	

**Tabela 4** - Wybrane wartości potęg dla systemu binarnego

## 2. Zamiana liczby $110111001_{(2)}$ na ósemkową

Wiemy, że  $2^3 = 8$  warto więc rozbić liczbę  $1111001_{(2)}$  na 3-cyfrowe bloki co ułatwi nam odczyt liczb z przygotowanej tabelki. Zaczynamy rozpisywać od prawej strony. Niestety w pewnym momencie dzielona liczba na bloki ma tylko 1 bit, musimy dopisać brakujące bity – dopisujemy 0.

$$001\ 111\ 001_{(2)} = 171_{(8)}$$

$0_{(8)}$	$000_{(2)}$
$1_{(8)}$	$001_{(2)}$
$2_{(8)}$	$010_{(2)}$
$3_{(8)}$	$011_{(2)}$
$4_{(8)}$	$100_{(2)}$
$5_{(8)}$	$101_{(2)}$
$6_{(8)}$	$110_{(2)}$
$7_{(8)}$	$111_{(2)}$

**Tabela 5** - Wybrane wartości dla systemu ósmkowego



### 3. Zamiana liczby 11110111000101,11<sub>(2)</sub> na szesnastkową

Podobnie jak system ósemkowy, system szesnastkowy także pozwala nam skrócić zapis binarny.  $2^4=16$  więc przyjmujemy tę koncepcję i rozpisujemy naszą liczbę binarną na czterocyfrowe bloki.

Zaczynamy rozpisywać od wartości ułamkowej, ponieważ blok na prawo ma tylko 2 cyfry więc brakujące cyfry dopełniamy zerami, następnie rozpisujemy wartość całkowitą liczby na bloki, dochodzimy do momentu gdy blok wysunięty najbardziej na lewo ma jedynie 3 cyfry więc dopisujemy brakujące zero, a następnie tak podzieloną liczbę na równe bloki możemy przekonwertować wg gotowych wartości z tabelki:

Zamiana liczb w systemach pozycyjnych o podstawie 2, 16			
<b>0</b> <sub>(16)</sub>	0000 <sub>(2)</sub>	<b>8</b> <sub>(16)</sub>	1000 <sub>(2)</sub>
<b>1</b> <sub>(16)</sub>	0001 <sub>(2)</sub>	<b>9</b> <sub>(16)</sub>	1001 <sub>(2)</sub>
<b>2</b> <sub>(16)</sub>	0010 <sub>(2)</sub>	<b>A</b> <sub>(16)</sub>	1010 <sub>(2)</sub>
<b>3</b> <sub>(16)</sub>	0011 <sub>(2)</sub>	<b>B</b> <sub>(16)</sub>	1011 <sub>(2)</sub>
<b>4</b> <sub>(16)</sub>	0100 <sub>(2)</sub>	<b>C</b> <sub>(16)</sub>	1100 <sub>(2)</sub>
<b>5</b> <sub>(16)</sub>	0101 <sub>(2)</sub>	<b>D</b> <sub>(16)</sub>	1101 <sub>(2)</sub>
<b>6</b> <sub>(16)</sub>	0110 <sub>(2)</sub>	<b>E</b> <sub>(16)</sub>	1110 <sub>(2)</sub>
<b>7</b> <sub>(16)</sub>	0111 <sub>(2)</sub>	<b>F</b> <sub>(16)</sub>	1111 <sub>(2)</sub>

**Tabela 6** - Wybrane wartości dla systemu heksadecymalnego

Po odczycie liczba przybiera następującą postać:

<sup>7</sup>0111   <sup>D</sup>1101   <sup>C</sup>1100   <sup>5</sup>0101   ,   <sup>C</sup>1100   => 7DC5,C

## Działania arytmetyczne

Wykonywanie operacji arytmetycznych na cyfrach  $x$  i  $y$ :

- Wynik operacji dla dodawania, odejmowania, mnożenia,
- Przeniesienie dla dodawania
- Pożyczka dla odejmowania

Zależności te zostały przedstawione w poniższej tabeli:

x y	x+y		x-y		x*y
	przeniesienie	wynik	pożyczka	wynik	wynik
0 0	-	0	-	0	0
0 1	-	1	1	1	0
1 0	-	1	-	1	0
1 1	1	0	-	0	1

**Tabela 7** – Zestawienie operacji arytmetycznych

### 1. Dodawanie binarne

	1111	
	↓↓↓↓	
1000 = 8 <sub>(10)</sub>	01101,100 = 13,50 <sub>(10)</sub>	
+ 0011 = 3 <sub>(10)</sub>	+ 00111,010 = 7,25 <sub>(10)</sub>	
1011 = 11 <sub>(10)</sub>	10100,110 = 20,75 <sub>(10)</sub>	

Należy pamiętać, że przy wykonywaniu operacji dodawania na liczbach binarnych 1+1 występuje przeniesienie.

**Uwaga!** Jeśli wykonujemy dodawanie w obrębie zmiennych to trzeba uważać bo np. jeśli mamy zmienną typu byte i do wartości 255 (wartość graniczna) dodamy 1 to nasz wynik wyjdzie poza zakres wielkości zmiennej. Niektóre języki programowania jak C# poinformują nas o tym fakcie na etapie kompilacji projektu. ☺

## 2. Odejmowanie binarne

### a) Sposób A

$  \begin{array}{r}  0 \ 1 \ 1 \ 0 = 6_{(10)} \\  - 0 \ 1 \ 0 \ 1 = 5_{(10)} \\  \hline  0 \ 0 \ 0 \ 1 = 1_{(10)}  \end{array}  $	$  \begin{array}{r}  1 \ 0 \ 1 \ 0 \ 0, \ 1 \ 1 \ 0 = 20,75_{(10)} \\  - \quad \quad 1 \ 1 \ 1, \ 0 \ 1 \ 0 = 7,25_{(10)} \\  \hline  0 \ 1 \ 1 \ 0 \ 1, \ 1 \ 0 \ 0 = 13,55_{(10)}  \end{array}  $
---	---

Podczas odejmowania 0-1 należy pożyczyć jedynkę z pozycji sąsiedniej, podobnie jak to ma miejsce w systemie dziesiętnym jak odejmujemy pod kreską.

### b) Sposób B - U2 (system uzupełnień do dwóch)

Najczęściej wykorzystywanym sposobem odejmowania liczb binarnych jest system uzupełnień do dwóch (U2). Polega on na negacji wszystkich bitów (liczba2) oraz dodaniu do uzyskanego wyniku jedynki. Czyli zamieniamy zera na jedynki, a jedynki na zera, a następnie dodajemy 1. Po czym dodajemy uzyskaną liczbę do (liczba1).

Dodawanie odbywa się łącznie z bitem znaku, jednak przeniesienia poza najstarszy bit ignorujemy - co zostanie pokazane w przykładach. Można także wykorzystać do zamiany liczb na U2 tabelę 8.

<p>Liczba 1: <math>1000 = 8_{(10)}</math></p> <p>Liczba 2: <math>0011 = 3_{(10)}</math></p> <p>Po negacji Liczby 2: <math>1100</math></p> <p>Dodajemy 1: <math>+0001</math></p> <p style="text-align: right;"><math>1101_{(U2)}</math></p> <p><math>1000 \Rightarrow 8</math></p> <p><math>+ 1101_{(U2)} \Rightarrow -3</math></p> <p><math>\cancel{1}0101 \Rightarrow 5</math></p> <p>(najstarszy bit pomijamy)</p>	<p>Liczba 1: <math>010100 = 20_{(10)}</math></p> <p>Liczba 2: <math>000111 = 7_{(10)}</math></p> <p>Po negacji Liczby 2: <math>111000</math></p> <p>Dodajemy 1: <math>+000001</math></p> <p style="text-align: right;"><math>111001_{(U2)}</math></p> <p><math>010100 \Rightarrow 20</math></p> <p><math>+ 111001_{(U2)} \Rightarrow -7</math></p> <p><math>\cancel{1}001101 \Rightarrow 13</math></p> <p>(najstarszy bit pomijamy)</p>
--	---

<p>Liczba 1: 00001001 = <math>9_{(10)}</math></p> <p>Liczba 2: 00001110 = <math>14_{(10)}</math></p> <p>Po negacji Liczby 2: 11110001</p> <p>Dodajemy 1: <u>+00000001</u></p> <p style="text-align: right;"><math>11110010_{(U2)}</math></p> <p>00001001 <math>\Rightarrow 9</math></p> <p>+ <u><math>11110010_{(U2)}</math></u> <math>\Rightarrow -14</math></p> <p>= <math>11111011_{(U2)}</math> <math>\Rightarrow -5</math></p>	
--	--

Reprezentacja dziesiętna	Reprezentacja znak-moduł	Reprezentacja U2
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000
-0	1000	-
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	-	1000

Tabela 8 – Reprezentacja liczb

### 3. Mnożenie binarne

Jedną z najłatwiejszych metod mnożenia jest metoda mnożenia bezpośredniego. Można z niej skorzystać wtedy gdy liczby są zapisane w kodzie znak-moduł. Pozwala to mnożyć w identyczny sposób jak w dobrze nam znanym systemie dziesiętnym pod kreską, czyli wielokrotnym dodawaniem odpowiednio przesuniętej mnożnej.

$\begin{array}{r} 101 = 5_{(10)} \\ * 111 = 7_{(10)} \\ \hline 101 \\ +101 \\ \hline 100011 = 35_{(10)} \end{array}$	$\begin{array}{r} 1001 = 9_{(10)} \\ * 101 = 5_{(10)} \\ \hline 1001 \\ +0000 \\ \hline 101101 = 45_{(10)} \end{array}$
--	---

$  \begin{array}{r}  1 \ 1 \ 1 = 7_{(10)} \\  * \ 1 \ 1 \ 1 = 7_{(10)} \\  \hline  1 \ 1 \ 1 \\  +1 \ 1 \ 1 \\  +1 \ 1 \ 1 \\  \hline  1 \ 1 \ 0 \ 0 \ 0 \ 1 = 49_{(10)}  \end{array}  $	
--	--

#### 4. Dzielenie binarne

Dzielenie binarne polega na odejmowaniu dzielnika od dzielnej.

Wartości jakie przyjmuje iloraz przedstawiono w tabeli.

Dzielna	Dzielnik	Iloraz
1	0	0
1	1	1

Tabela 9 – Wartości ilorazu

Dzielenie może powodować, że powstanie reszta i można je zakończyć w momencie, gdy pojawi się reszta. Krótki przykład dzielenia dla części całkowitej:

<b>1011</b>	- wynik dzielenia $11_{(10)}$
$111010 : 101$	$[58_{(10)} : 5_{(10)}]$
$111010$	(dzielna)
$-101$	(przesunięty dzielnik)
$10010$	(nowa wartość dzielnej)
$-101$	(przesunięty dzielnik o 1 pozycję)
$1000$	(nowa wartość dzielnej)
$-101$	(przesunięty dzielnik)
$11$	← reszta

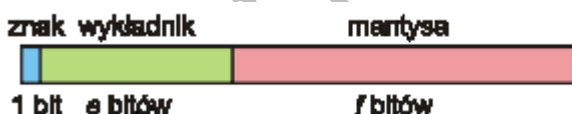
Na tym etapie zakończyłem dzielenie jednak gdybyśmy je dalej kontynuowali otrzymamy wtedy część ułamkową, a tym samym nasza liczba będzie miała większą dokładność.

## ARYTMETYKA STAŁOPRZECINKOWA I ZMIENNOPRZECINKOWA

Warto zauważyć, że liczby zapisywane są zawsze z określoną dokładnością! Zapisywane są one w postaci wykładniczej. Więc przyjmuje się pewien zakres na mantysę i wykładnik.

Dlatego też, liczby rzeczywiste można zapisywać jako:

- Liczby stałoprzecinkowe, gdzie wyróżnia się:
  - Pole znakowe (bit reprezentujący znak liczby)
  - Pole liczbowe (bity reprezentujące cyfry liczbowe)
  - Reprezentacja stałopozycyjna oznacza, że z góry określamy, ile bitów przeznaczamy na część całkowitą, a ile na ułamkową.
  - typy 4-bajtowe posiadają (3 bajty mantysy i 1 bajt na wykładnik w tym 1 bit na znak liczby)
- Liczby zmiennoprzecinkowe – Norma IEEE-754 ([czytaj więcej](#))



Rysunek 3 – Reprezentacja liczby rzeczywistej

## NOTATKI

---

[HTTP://WWW.HAKERZY.NET](http://www.hakerzy.net)

*Poradnika tego nie można traktować jako jedyne źródła wiedzy, ponieważ wiele rzeczy zostało uproszczonych w celu lepszego zrozumienia materiału, a część rzeczy po prostu pominiętych.*

*Być może w przyszłości zostanie wydany suplement, który poruszy nowe zagadnienia, ale niczego nie obiecuję natomiast warto zaznaczyć, że zostało omówione niezbędne minimum dla tych użytkowników komputera, którzy mają zamiar zacząć swoją przygodę z informatyką.*

*Krzysztof Kruczka*