Informatyka 1 (EZ1F1002)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2022/2023

Wykład nr 3 (30.10.2022)

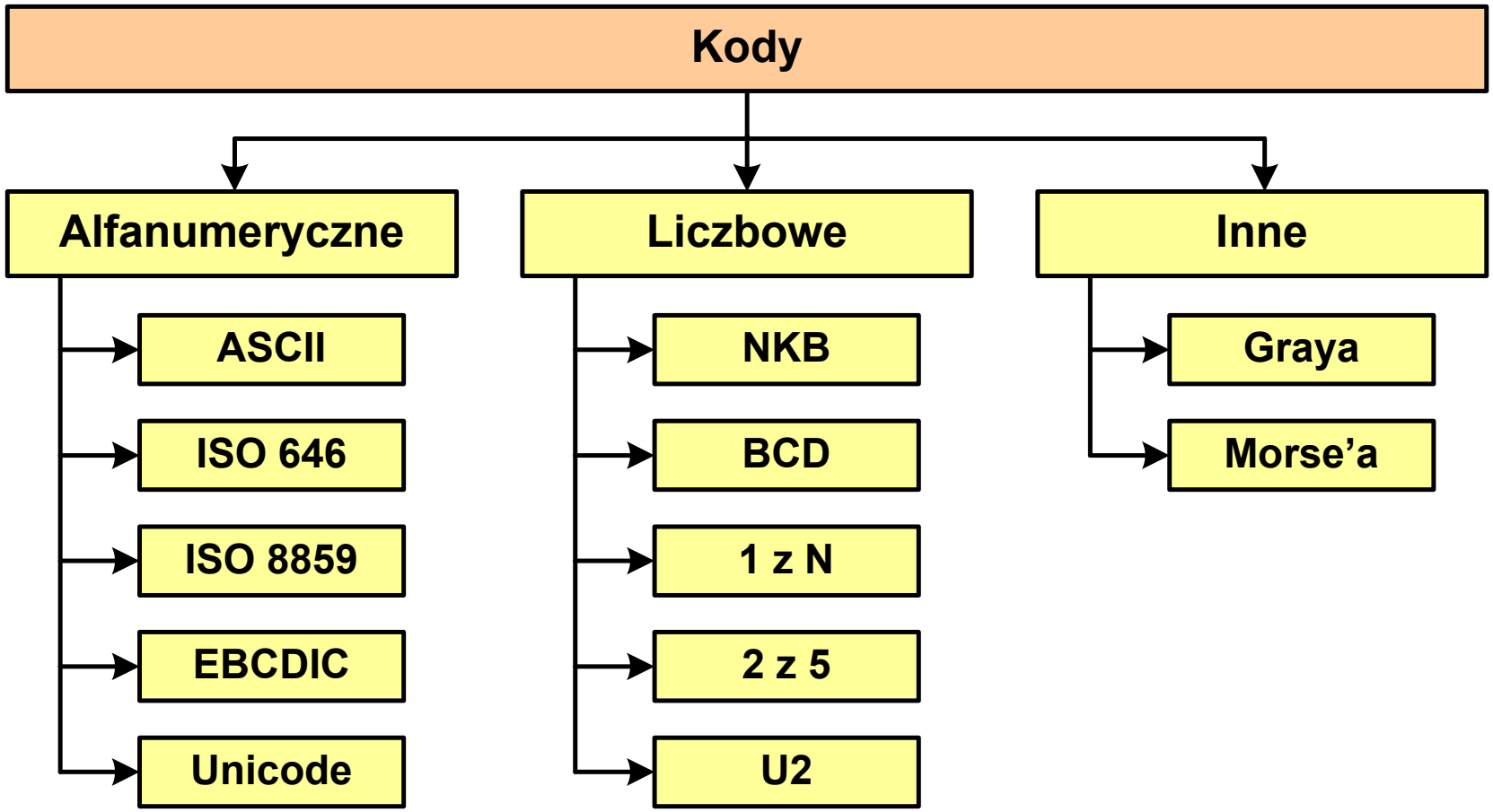
dr inż. Jarosław Forenc

Plan wykładu nr 3

- Kodowanie znaków
  - ASCII, ISO 8859, Unicode
- Kodowanie liczb
  - NKB, BCD, kod Graya
- Reprezentacja liczb całkowitych
  - liczby bez znaku, liczby ze znakiem (ZM, U1, U2)
- Reprezentacja zmiennoprzecinkowa
  - zapis, postać znormalizowana, zakres liczb
- Standard IEEE 754
  - liczby 32-bitowe, liczby 64-bitowe
  - zakres i precyzja liczb, wartości specjalne

Kodowanie

- Kodowanie - proces przekształcania jednego rodzaju postaci informacji na inną postać



Kod ASCII

- ASCII - American Standard Code for Information Interchange
  - 7-bitowy kod przypisujący liczby z zakresu 0-127:
    - literom (alfabet angielski)
    - cyfrom
    - znakom przestankowym
    - innym symbolom
    - poleceniom sterującym.

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

Kod ASCII - Kody sterujące

- Kody sterujące - 33 kody, o numerach: 0-31, 127

Dec	Hex	Char		Dec	Hex	Char	
0	0	NUL	(null)	16	10	DLE	(data link escape)
1	1	SOH	(start of heading)	17	11	DC1	(device control 1)
2	2	STX	(start of text)	18	12	DC2	(device control 2)
3	3	ETX	(end of text)	19	13	DC3	(device control 3)
4	4	EOT	(end of tramsmission)	20	14	DC4	(device control 4)
5	5	ENQ	(enquiry)	21	15	NAK	(negative acknowlege)
6	6	ACK	(acknowledge)	22	16	SYN	(synchronous idle)
7	7	BEL	(bell)	23	17	ETB	(end of trans. block)
8	8	BS	(backspace)	24	18	CAN	(cancel)
9	9	TAB	(horizontal tab)	25	19	EM	(end of medium)
10	A	LF	(NL line feed, new line)	26	1A	SUB	(subtitute)
11	B	VT	(vertical tab)	27	1B	ESC	(escape)
12	C	FF	(NP form feed, new page)	28	1C	FS	(file separator)
13	D	CR	(carriage return)	29	1D	GS	(group separator)
14	E	SO	(shift out)	30	1E	RS	(record separator)
15	F	SI	(shift in)	31	1F	US	(unit separator)
				127	7F	DEL	

- W języku C:

0 (NULL) - \0 7 (BEL) - \a 8 (BS) - \b

9 (TAB) - \t 10 (LF) - \n 13 (CR) - \r

Kod ASCII - Pliki tekstowe

- Elementami pliku tekstowego są wiersze, mogą one mieć różną długość
- W systemie Windows każdy wiersz pliku zakończony jest parą znaków:
  - CR, ang. carriage return - powrót karetki, kod ASCII - $13_{(10)} = 0D_{(16)}$
  - LF, ang. line feed - przesunięcie o wiersz, kod ASCII - $10_{(10)} = 0A_{(16)}$
- Załóżmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
00000000: 50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
00000010: 70 6C 69 6B 75 0D 0A 44|72 75 67 69 20 77 69 65 | pliku■■Drugi wie
00000020: 72 73 7A 20 70 6C 69 6B|75 0D 0A 54 72 7A 65 63 | rsz pliku■■Trzec
00000030: 69 20 77 69 65 72 73 7A|20 70 6C 69 6B 75 0D 0A | i wiersz pliku■■
```

- Wydruk zawiera:
  - przesunięcie od początku pliku (szesnastkowo)
  - wartości poszczególnych bajtów pliku (szesnastkowo)
  - znaki odpowiadające bajtom pliku (traktując bajty jako kody ASCII)

Kod ASCII - Pliki tekstowe

- W systemie Linux znakiem końca wiersza jest tylko LF o kodzie ASCII - $10_{(10)} = 0A_{(16)}$
- Załóżmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
00000000: 50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
00000010: 70 6C 69 6B 75 0A 44 72|75 67 69 20 77 69 65 72 | pliku■Drugi wier
00000020: 73 7A 20 70 6C 69 6B 75|0A 54 72 7A 65 63 69 20 | sz pliku■Trzeci
00000030: 77 69 65 72 73 7A 20 70|6C 69 6B 75 0A          | wiersz pliku■
```

- Podczas przesyłania pliku tekstowego (np. przez protokół ftp) z systemu Linux do systemu Windows pojedynczy znak LF zamieniany jest automatycznie na parę znaków CR i LF
- Błędne przesłanie pliku tekstowego (w trybie binarnym) powoduje nieprawidłowe jego wyświetlanie:

```
Pierwszy wiersz pliku■Drugi wiersz pliku■Trzeci wiersz pliku■
```

ISO/IEC 8859

- ISO/IEC 8859 - zestaw 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)
- Od czerwcu 2004 roku ISO 8859 nie jest rozwijane.

ISO/IEC 8859

- Stosowane standardy ISO 8859:
  - 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 litery
  - ISO 8859-15 (Latin-9)
  - ISO 8859-16 (Latin-10) - łaciński dla Europy środkowej, zawiera polskie litery

ISO/IEC 8859-2

- ISO/IEC 8859-2, Latin-2 („środkowo”, „wschodnioeuropejskie”)
- dostępne języki: bośniacki, chorwacki, czeski, węgierski, polski, rumuński, serbski, serbsko-chorwacki, słowacki, słoweński, górno- i dolnołużycki
- możliwość przedstawienia znaków w języku niemieckim i angielskim
- 191 znaków łacińskiego pisma
- do 02.11.2015 kodowanie to było zgodne z Polską Normą

	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	SHY	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	˙

SP - spacja
NBSP - twarda spacja
SHY - miękki dywiz (myślnik)

ISO/IEC 8859-2 - Litery diakrytyczne w j. polskim

- 18 liter:
  - Ą - ą
  - Ć - ć
  - Ę - ę
  - Ł - ł
  - Ń - ń
  - Ó - ó
  - Ś - ś
  - Ź - ź
  - Ż - ż

	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	SHY	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	˙

Unicode (Unikod)

- 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
- Rozwijany przez konsorcjum utworzone przez firmy komputerowe, producentów oprogramowania oraz grupy użytkowników
  - http://www.unicode.org
- Pierwsza wersja: Unicode 1.0 (październik 1991)
- Ostatnia wersja: Unicode 15.0.0 (13 września 2022)
  - The Unicode Consortium. The Unicode Standard, Version 15.0.0, (Mountain View, CA: The Unicode Consortium, 2022)
  - http://www.unicode.org/versions/Unicode15.0.0/
  - Koduje 149.186 znaków

Unicode - Zakresy

Zakres:	Znaczenie:
U+0000 - U+007F	Basic Latin (to samo co w ASCII)
U+0080 - U+00FF	Latin-1 Supplement (to samo co w ISO/IEC 8859-1)
U+0100 - U+017F	Latin Extended-A
U+0180 - U+024F	Latin Extended-B
U+0250 - U+02AF	IPA Extensions
U+02B0 - U+02FF	Spacing Modifiers Letters
...	
U+0370 - U+03FF	Greek
U+0400 - U+04FF	Cyrillic
...	
U+1D00 - U+1D7F	Phonetic Extensions
U+1D80 - U+1DBF	Phonetic Extensions Supplement
U+1E00 - U+1EFF	Latin Extended Additional
U+1F00 - U+1FFF	Greek Extended
...	

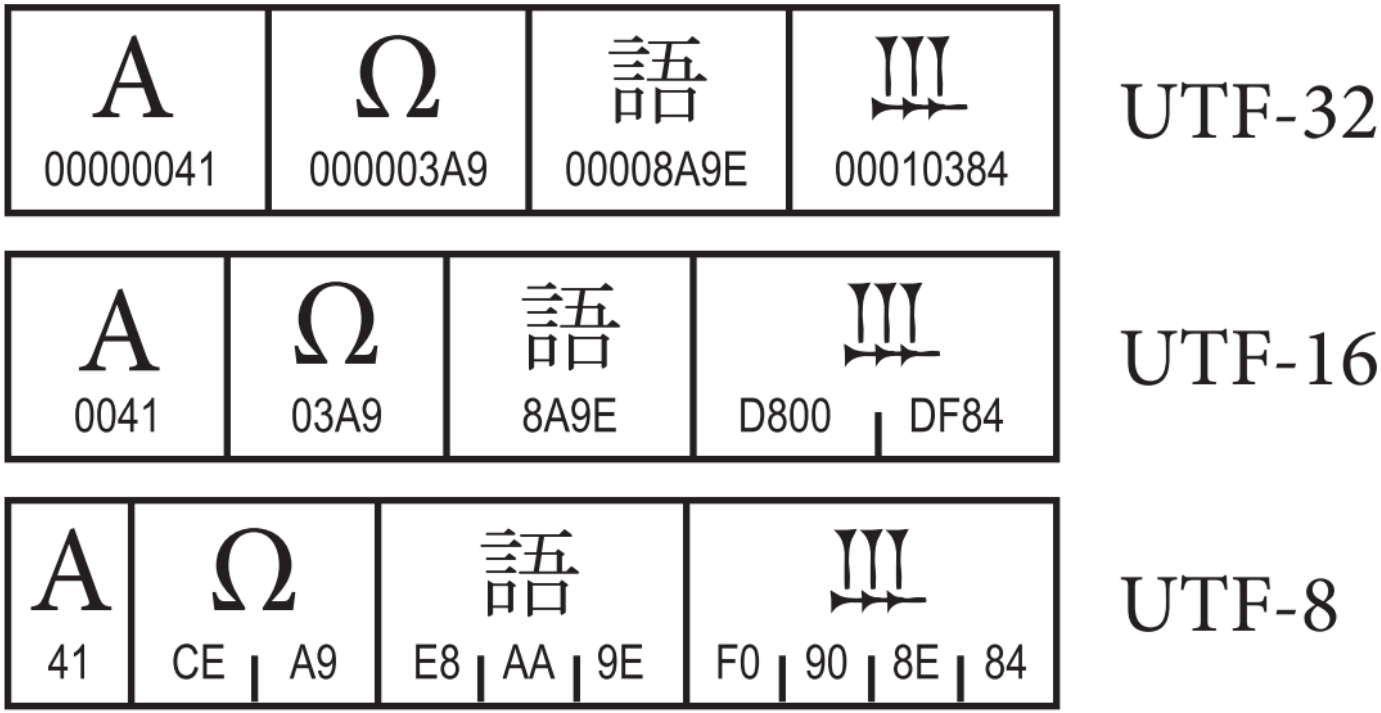
Unicode

- Standard Unicode definiuje nie tylko kody numeryczne przypisane poszczególnym znakom, ale także określa sposób bajtowego kodowania znaków
- Kodowanie określa sposób w jaki znaki ze zbioru mają być zapisane w postaci binarnej
- Istnieją trzy podstawowe metody kodowania:
  - 32-bitowe: UTF-32
  - 16-bitowe: UTF-16
  - 8-bitowe: UTF-8

  gdzie: UTF - UCS Transformation Format
  UCS - Universal Character Set
- Wszystkie metody obejmują wszystkie kodowane znaki w Unicode.

Unicode

- Metody kodowania różnią się liczbą bajtów przeznaczonych do opisania kodu znaku



źródło: The Unicode Consortium. The Unicode Standard, Version 8.0

Unicode - kodowanie UTF-32

- UTF-32 - sposób kodowania standardu Unicode wymagający użycia 32-bitowych słów

- 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.

Unicode - kodowanie UTF-16

- UTF-16 - 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
  - drugie słowo należy do przedziału: U+DC00 - U+DFFF.

Unicode - kodowanie UTF-8

- UTF-8 - kodowanie ze zmienną długością reprezentacji znaku wymagające użycia 8-bitowych słów

- Znaki Unikodu są mapowane na ciągi bajtów
  - 0x00 do 0x7F - bity 0xxxxxxx
  - 0x80 do 0x7FF - bity 110xxxxx 10xxxxxx
  - 0x800 do 0xFFFF - bity 1110xxxx 10xxxxxx 10xxxxxx
  - 0x10000 do 0x1FFFFF - bity 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
  - 0x200000 do 0x3FFFFFF - bity 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
  - 0x4000000 do 0x7FFFFFFF - bity 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

Unicode

	010	011	012	013	014	015	016	017
0	Ā 0100	Đ 0110	Ġ 0120	İ 0130	l· 0140	Ő 0150	Š 0160	Ű 0170
1	ā 0101	đ 0111	ġ 0121	ı 0131	Ł 0141	ő 0151	š 0161	ű 0171
2	Ă 0102	Ē 0112	Ģ 0122	IJ 0132	ł 0142	Œ 0152	Ţ 0162	Ų 0172
3	ă 0103	ē 0113	ģ 0123	ij 0133	Ń 0143	œ 0153	ţ 0163	ų 0173
4	Ą 0104	Ĕ 0114	Ĥ 0124	Ĵ 0134	ń 0144	Ŕ 0154	Ť 0164	Ŵ 0174
5	ą 0105	ĕ 0115	ĥ 0125	ĵ 0135	Ņ 0145	ŕ 0155	ť 0165	ŵ 0175
6	Ć 0106	Ė 0116	Ħ 0126	Ķ 0136	ņ 0146	Ŗ 0156	Ŧ 0166	Ŷ 0176
7	ć 0107	ė 0117	ħ 0127	ķ 0137	Ň 0147	ŗ 0157	ŧ 0167	ŷ 0177

European Latin

0100 Ā LATIN CAPITAL LETTER A WITH MACRON
≡ 0041 A 0304 ◌̄

0101 ā LATIN SMALL LETTER A WITH MACRON
• Latvian, Latin, ...
≡ 0061 a 0304 ◌̄

0102 Ă LATIN CAPITAL LETTER A WITH BREVE
≡ 0041 A 0306 ◌̆

0103 ă LATIN SMALL LETTER A WITH BREVE
• Romanian, Vietnamese, Latin, ...
≡ 0061 a 0306 ◌̆

0104 Ą LATIN CAPITAL LETTER A WITH OGONEK
≡ 0041 A 0328 ◌̨

0105 ą LATIN SMALL LETTER A WITH OGONEK
• Polish, Lithuanian, ...
≡ 0061 a 0328 ◌̨

0106 Ć LATIN CAPITAL LETTER C WITH ACUTE
≡ 0043 C 0301 ◌́

0107 ć LATIN SMALL LETTER C WITH ACUTE
• Polish, Croatian, ...
→ 045B ћ cyrillic small letter tshe
≡ 0063 c 0301 ◌́

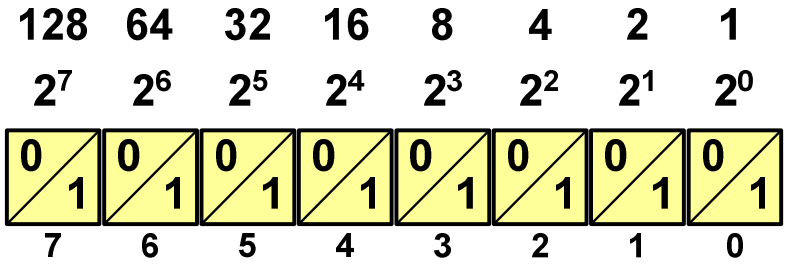
Unicode

27308 **CJK Unified Ideographs Extension B** **27342**

27308 虫 142.8	蜹 UCS2003	蜹 GKX-1086.03	蜹 T4-4721
27309 虫 142.8	蜯 UCS2003	蜯 GKX-1086.05	蜯 T5-4955
2730A 虫 142.8	蜲 UCS2003	蜲 GKX-1086.08	蜲 T4-467D
2730B 虫 142.8	蜵 UCS2003	蜵 GKX-1086.10	蜵 T6-6223
2730C 虫 142.8	蜷 UCS2003	蜷 GKX-1086.12	蜷 T5-495F
2730D 虫 142.8	蜸 UCS2003	蜸 GKX-1086.22	蜸 T4-4677
2731B 虫 142.8	蟗 UCS2003	蟗 GKX-1088.15	蟗 T6-617B
2731C 虫 142.8	蟛 UCS2003	蟛 GKX-1088.16	蟛 T6-6221
2731D 虫 142.8	蟙 UCS2003	蟙 GKX-1088.17	蟙 T5-4960
2731E 虫 142.7	蟎 UCS2003	蟎 GKX-1088.18	
2731F 虫 142.8	蟨 UCS2003	蟨 GKX-1088.19	蟨 T6-6174
27320 虫 142.8	蟲 UCS2003	蟲 GKX-1088.20	蟲 T6-617D
2732F 虫 142.8	蟶 UCS2003	蟶 GHC	
27330 虫 142.9	蠎 UCS2003	蠎 GHC	
27331 虫 142.8	蠻 UCS2003	蠻 G4K	
27332 虫 142.8	蠏 UCS2003	蠏 GHC	
27333 虫 142.8	蠅 UCS2003	蠅 GHC	
27334 虫 142.8	蠔 UCS2003	蠔 T5-4953	

Kody liczbowe - Naturalny Kod Binarny (NKB)

- Jeżeli dowolnej liczbie dziesiętnej przypiszemy odpowiadającą jej liczbę binarną, to otrzymamy naturalny kod binarny (NKB)



Liczba dziesiętna	Kod NKB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Liczba dziesiętna	Kod NKB
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Kody liczbowe - Kod BCD

- Binary-Coded Decimal - dziesiętny zakodowany dwójkowo
- BCD - sposób zapisu liczb polegający na zakodowaniu kolejnych cyfr liczby dziesiętnej w 4-bitowym systemie dwójkowym (NKB)

Cyfra dziesiętna	BCD	Cyfra dziesiętna	BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

- W ogólnym przypadku kodowane są tylko znaki 0 ÷ 9
- Pozostałe kombinacje bitowe mogą być stosowane do kodowania znaku liczby lub innych znaczników.

Kody liczbowe - Kod BCD

- Przykład:

$$168_{(10)} = ?_{(BCD)}$$

$$\overbrace{0001}^{1}\ \overbrace{0110}^{6}\ \overbrace{1000}^{8}$$

$$168_{(10)} = 000101101000_{(BCD)}$$

$$1001\,|\,0101\,|\,0011_{(BCD)} = ?_{(10)}$$

$$\underbrace{1001}_{9}\ \underbrace{0101}_{5}\ \underbrace{0011}_{3}$$

$$100101010011_{(BCD)} = 953_{(10)}$$

- Zastosowania:
  - urządzenia elektroniczne z wyświetlaczem cyfrowym (np. kalkulatory, mierniki cyfrowe, kasy sklepowe, wagi)
  - przechowywania daty i czasu w BIOSie komputerów (także wczesne modele PlayStation 3)
  - zapis części ułamkowych kwot (systemy bankowe).

Kody liczbowe - Kod BCD: przechowywanie liczb

- Użycie 4 najmłodszych bitów jednego bajta, 4 starsze bity są ustawiane na jakąś konkretną wartość:
  - 0000
  - 1111 (np. kod EBCDIC, liczby $F0_{(16)} \div F9_{(16)}$)
  - 0011 (tak jak w ASCII, liczby $30_{(16)} \div 39_{(16)}$)
- Zapis dwóch cyfr w każdym bajcie (starsza na starszej połówce, młodsza na młodszej połówce) - jest to tzw. spakowane BCD
  - w przypadku liczby zapisanej na kilku bajtach, najmniej znacząca tetrada (4 bity) używane są jako flaga znaku
  - standardowo przyjmuje się 1100 ($C_{(16)}$) dla znaku plus (+) i 1101 ($D_{(16)}$) dla znaku minus (-), np.

$$127_{(10)} = 0001\ 0010\ 0111\ 1100 \quad (127C_{(16)})$$

$$-127_{(10)} = 0001\ 0010\ 0111\ 1101 \quad (127D_{(16)})$$

Kody liczbowe - Kod BCD

- Warianty kodu BCD:

Cyfra dziesiętna	BCD 8421	Excess-3	BCD 2421	BCD 84-2-1	IBM 1401 BCD 8421
0	0000	0011	0000	0000	1010
1	0001	0100	0001	0111	0001
2	0010	0101	0010	0110	0010
3	0011	0110	0011	0101	0011
4	0100	0111	0100	0100	0100
5	0101	1000	1011	1011	0101
6	0110	1001	1100	1010	0110
7	0111	1010	1101	1001	0111
8	1000	1011	1110	1000	1000
9	1001	1100	1111	1111	1001

- Podstawowy wariant: BCD 8421 (SBCD - Simple Binary Coded Decimal)

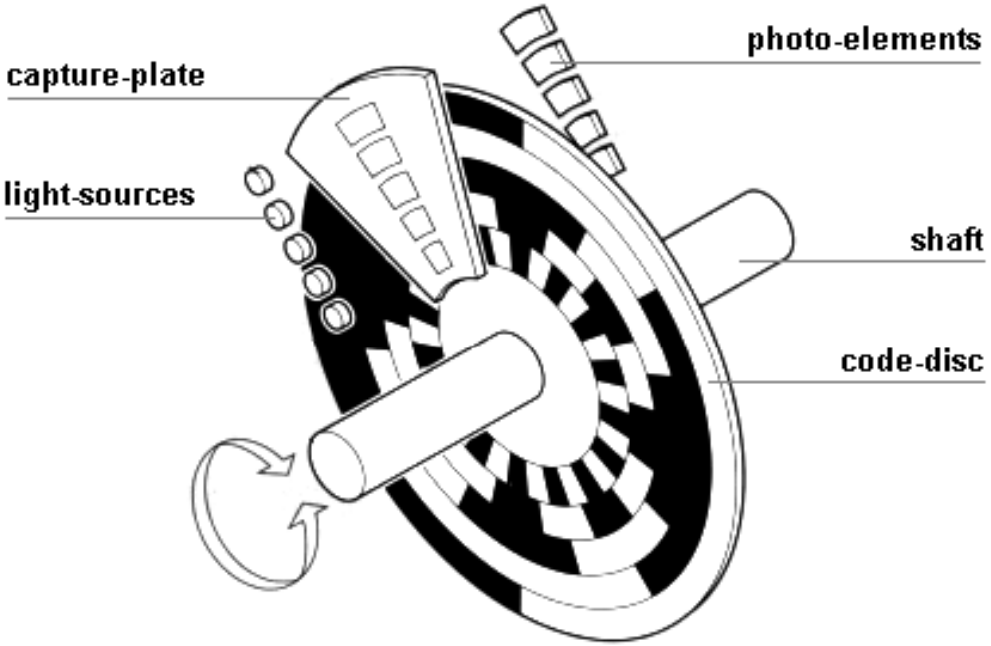
Kod Graya (refleksyjny)

- Kod dwójkowy, bezwagowy, niepozycyjny
- Dwa kolejne słowa kodowe różnią się stanem jednego bitu
- Kod cykliczny - ostatni i pierwszy wyraz również różnią się stanem jednego bitu

kod 1-bitowy	kod 2-bitowy	kod 3-bitowy
0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

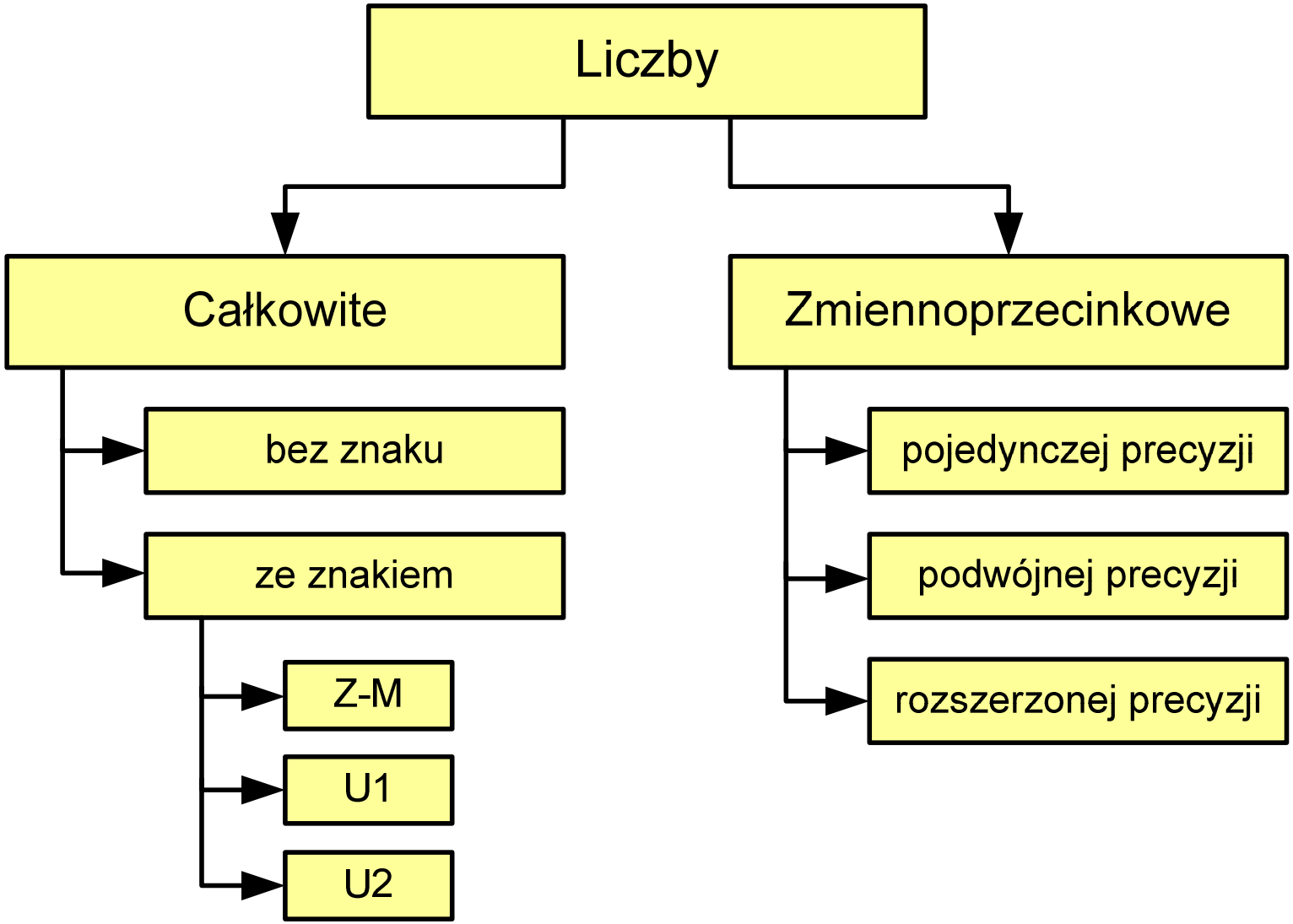
Kod Graya

- Stosowany w przetwornikach analogowo-cyfrowych, do cyfrowego pomiaru analogowych wielkości mechanicznych (np. kąt obrotu)



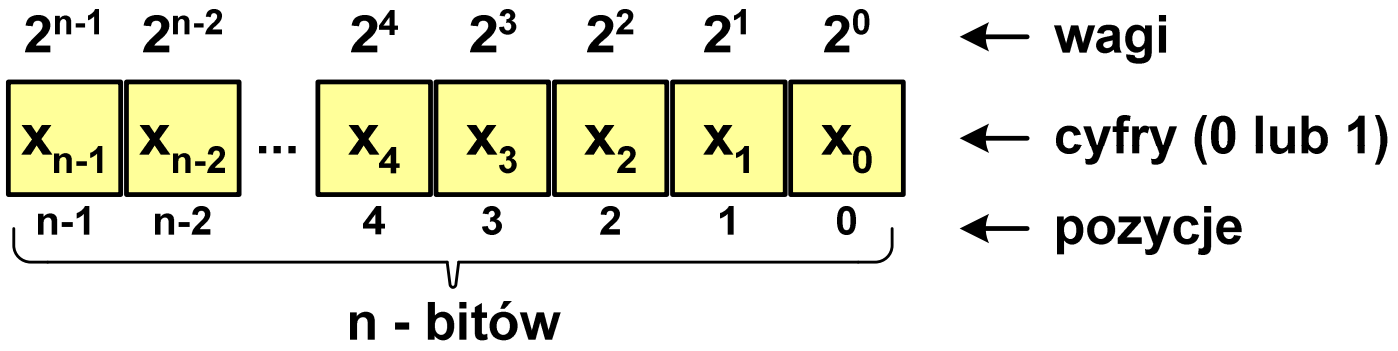
http://tams-www.informatik.uni-hamburg.de
/applets/hades/webdemos/10-gates/15-graycode/dual2gray.html

Reprezentacja liczb w systemach komputerowych



Liczby całkowite bez znaku

- Zapis liczby w systemie dwójkowym:



- Używając n-bitów można zapisać liczbę z zakresu:

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

$8 - bitów$	0 ... 255
$16 - bitów$	0 ... 65 535
$32 - bity$	0 ... 4 294 967 295
$64 - bity$	0 ... 18 446 744 073 709 551 615

18 trylionów 446 biliardów 744 biliony 73 miliardy 709 milionów 551 tysięcy 615

Liczby całkowite bez znaku w języku C

- Typy zmiennych całkowitych bez znaku stosowane w języku C:

Nazwa typu	Rozmiar (bajty)	Zakres wartości
unsigned char	1 bajt	0 … 255
unsigned short int	2 bajty	0 … 65 535
unsigned int	4 bajty	0 … 4 294 967 295
unsigned long int	4 bajty	0 … 4 294 967 295
unsigned long long int	8 bajtów	0 … 18 446 744 073 709 551 615

- W nazwach typów short i long można pominąć słowo int:

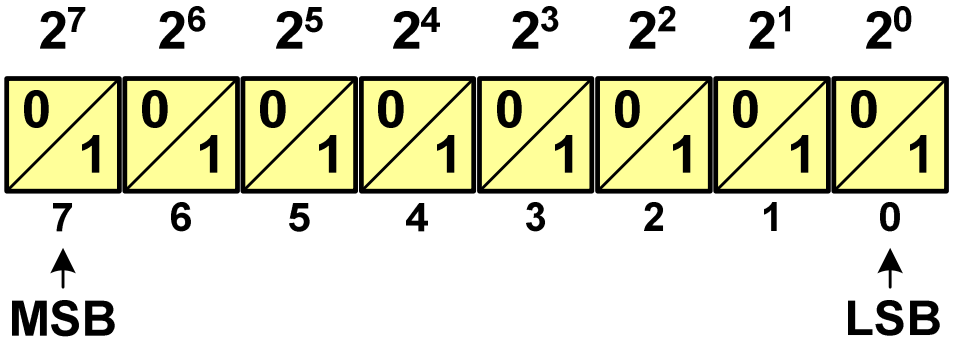
unsigned short int → unsigned short

unsigned long int → unsigned long

unsigned long long int → unsigned long long

Liczby całkowite bez znaku w języku C

- Typ unsigned char (1 bajt):

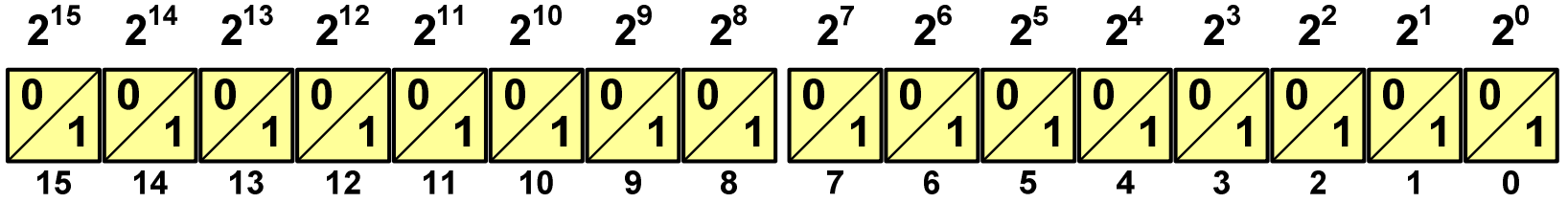


  - MSB (Most Significant Bit) - najbardziej znaczący bit, najstarszy bit, największa waga
  - LSB (Least Significant Bit) - najmniej znaczący bit, najmłodszy bit, najmniejsza waga

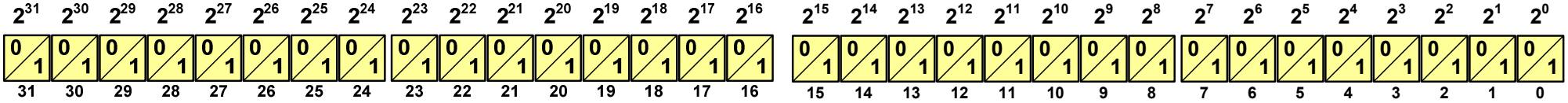
- Zakres wartości:
  - dolna granica: $0000\,0000_{(2)} = 00_{(16)} = 0_{(10)}$
  - górna granica: $1111\,1111_{(2)} = FF_{(16)} = 255_{(10)}$

Liczby całkowite bez znaku w języku C

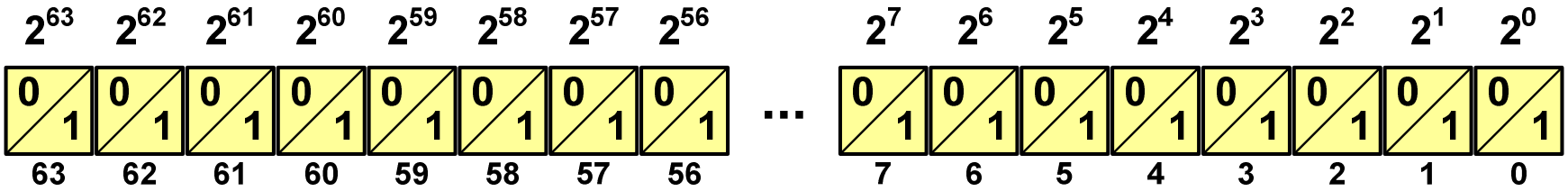
- Typ unsigned short int (2 bajty):



- Typy unsigned int (4 bajty) i unsigned long int (4 bajty):



- Typ unsigned long long int (8 bajtów):



Liczby całkowite bez znaku w języku C

```
unsigned short int:      65535 0 1
unsigned int:            4294967295 0 1
unsigned long int:       4294967295 0 1
unsigned long long int: 18446744073709551615 0 1
```

```c
#include <stdi

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int       usi = 65535;
   unsigned int              ui = 4294967295;
   unsigned long int        uli = 4294967295;
   unsigned long long int ulli = 18446744073709551615;

   printf("unsigned short int:      %hu %hu %hu\n",usi,usi+1,usi+2);
   printf("unsigned int:            %u %u %u\n",ui,ui+1,ui+2);
   printf("unsigned long int:       %lu %lu %lu\n",uli,uli+1,uli+2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli+1,ulli+2);
   return 0;
}
```

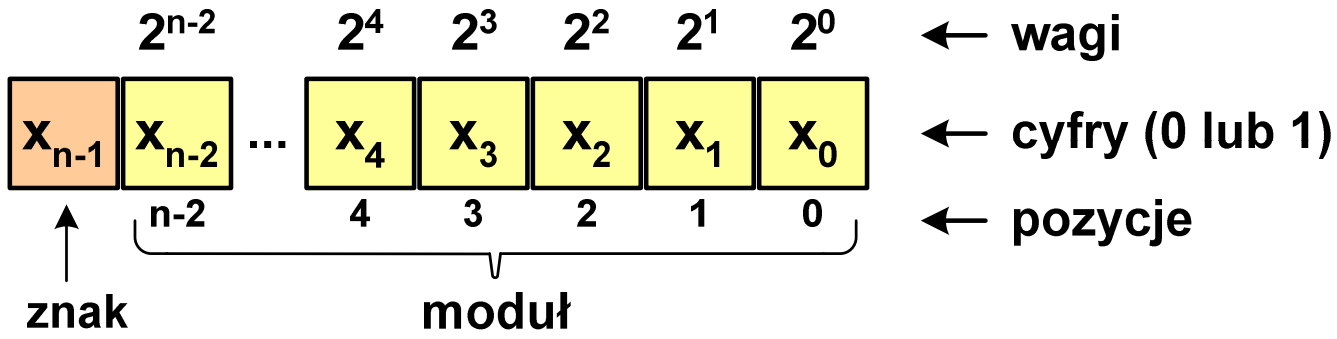
Liczby całkowite bez znaku w języku C

```
unsigned short int:      1 0 65535
unsigned int:            1 0 4294967295
unsigned long int:       1 0 4294967295
unsigned long long int: 1 0 18446744073709551615
```

```c
#include <stdi

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int      usi = 1;
   unsigned int             ui = 1;
   unsigned long int       uli = 1;
   unsigned long long int ulli = 1;

   printf("unsigned short int:      %hu %hu %hu\n",usi,usi-1,usi-2);
   printf("unsigned int:            %u %u %u\n",ui,ui-1,ui-2);
   printf("unsigned long int:       %lu %lu %lu\n",uli,uli-1,uli-2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli-1,ulli-2);

   return 0;
}
```

Liczby całkowite ze znakiem - kod znak-moduł

- Inne nazwy: ZM, Z-M, SM (Signed Magnitude), S+M
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Pozostałe bity mają takie same znaczenie jak w NKB



- Wartość liczby:

$$X_{(10)} = \underbrace{(x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2})}_{\text{moduł}} \cdot \underbrace{(-1)^{x_{n-1}}}_{\text{znak}} = (-1)^{x_{n-1}} \cdot \sum_{i=0}^{n-2} x_i \cdot 2^i$$

Liczby całkowite ze znakiem - kod znak-moduł

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie Z-M:

Z-M	dziesiętnie
0000	+0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Z-M	dziesiętnie
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

- dwie reprezentacje zera

$$+0\ (0000_{ZM})$$

$$-0\ (1000_{ZM})$$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \langle -127 \ldots 127 \rangle$$

$$\text{dla 16 bitów}: \quad \langle -32767 \ldots 32767 \rangle$$

Liczby całkowite ze znakiem - kod znak-moduł

- Zamiana liczby dziesiętnej na kod Z-M:
  - liczba dodatnia

$$93_{(10)} = ?_{(ZM)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$93_{(10)} = 01011101_{(ZM)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(ZM)}$$

  - zamieniamy moduł liczby na NKB

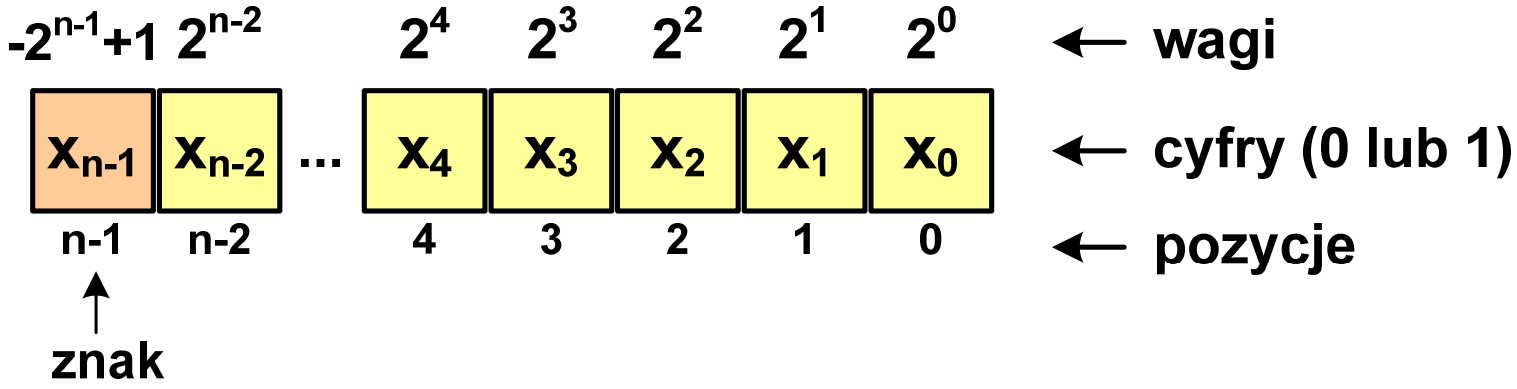
$$\left|-93_{(10)}\right| = 93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$-93_{(10)} = 11011101_{(ZM)}$$

Liczby całkowite ze znakiem - kod U1

- Inne nazwy: U1, ZU1, uzupełnień do jedności
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Wszystkie bity liczby posiadają takie same wagi jak w NKB, oprócz pierwszego bitu, który ma wagę $-2^{n-1} + 1$



- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1} + 1)$$

Liczby całkowite ze znakiem - kod U1

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U1:

U1	dziesiętnie
0000	+0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

U1	dziesiętnie
1111	-0
1110	-1
1101	-2
1100	-3
1011	-4
1010	-5
1001	-6
1000	-7

- liczby dodatnie zapisywane są tak samo jak w NKB
- liczby ujemne otrzymywane są poprzez bitową negację
- dwie reprezentacje zera

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \langle -127 \ldots 127 \rangle$$

$$\text{dla 16 bitów}: \quad \langle -32767 \ldots 32767 \rangle$$

Liczby całkowite ze znakiem - kod U1

- Zamiana liczby dziesiętnej na kod U1:
  - liczba dodatnia

$$93_{(10)} = ?_{(U1)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku: 0

$$93_{(10)} = 01011101_{(U1)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(U1)}$$

  - zamieniamy moduł liczby na U1

$$\left|-93_{(10)}\right| = 93_{(10)} = 01011101_{(U1)}$$

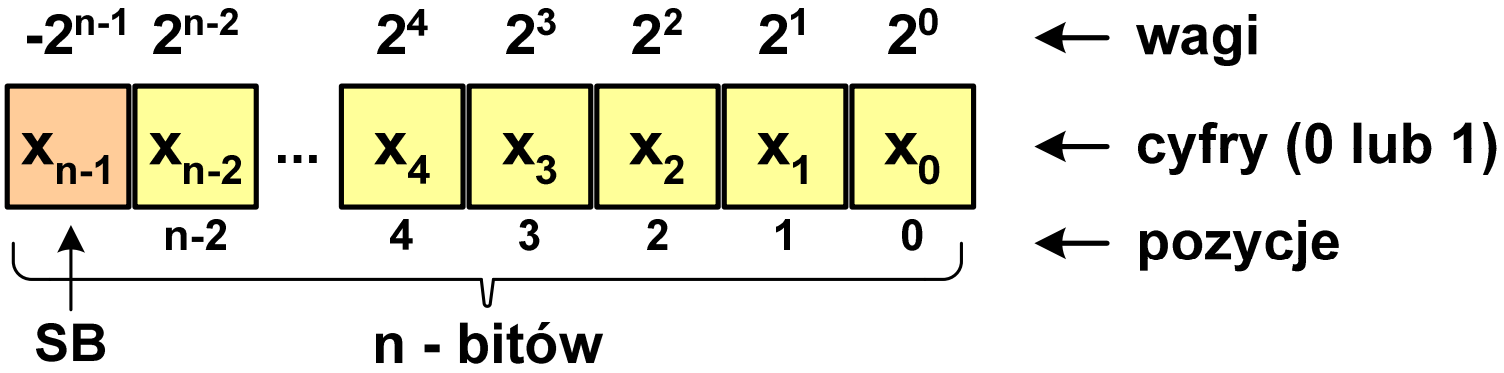
  - negujemy wszystkie bity

$$-93_{(10)} = 10100010_{(U1)}$$

bit znaku

Liczby całkowite ze znakiem - kod U2

- Inne nazwy: ZU2, uzupełnień do dwóch, two's complement
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna



- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1})$$

- Kod U2 jest obecnie powszechnie stosowany w informatyce

Liczby całkowite ze znakiem - kod U2

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U2:

U2	dziesiętnie
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

U2	dziesiętnie
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

- brak podwójnej reprezentacji zera
- liczb ujemnych jest o jeden więcej niż dodatnich
- 00...000 zawsze oznacza $0_{(10)}$
  11...111 zawsze oznacza $-1_{(10)}$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \langle -2^{n-1}, 2^{n-1} - 1 \rangle$$

$$\text{dla 8 bitów}: \quad \langle -128 \ldots 127 \rangle$$

$$\text{dla 16 bitów}: \quad \langle -32768 \ldots 32767 \rangle$$

Liczby całkowite ze znakiem - kod U2

- Zamiana liczby dziesiętnej na kod U2:
  - liczba dodatnia

$$75_{(10)} = ?_{(U2)}$$

  - zamieniamy liczbę na NKB

$$75_{(10)} = 1001011_{(NKB)}$$

  - dodajemy bit znaku: 0

$$75_{(10)} = 01001011_{(U2)}$$

  - liczba ujemna

$$-75_{(10)} = ?_{(U2)}$$

  - zamieniamy moduł liczby na U2

$$\left|-75_{(10)}\right| = 75_{(10)} = 01001011_{(U2)}$$

  - negujemy wszystkie bity i dodajemy 1

$$\begin{array}{rr} & 01001011 \\ \text{negacja}: & 10110100 \\ +1: & \quad\quad 1 \\ \hline -75_{(10)} = & 10110101_{(U2)} \end{array}$$

Liczby całkowite ze znakiem - kod U2 w języku C

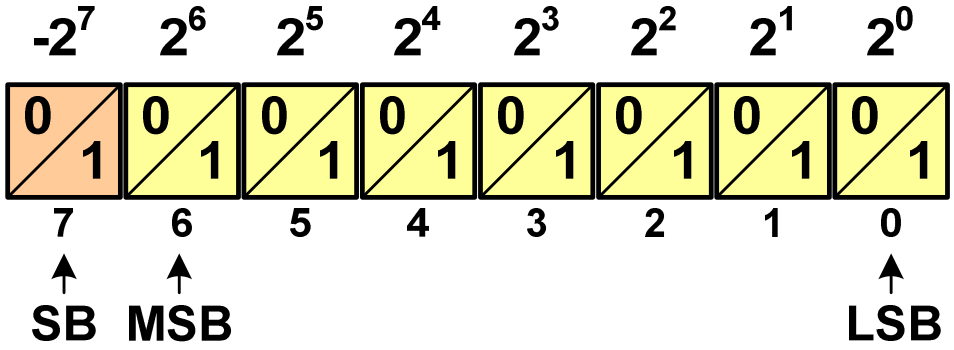
- Typy zmiennych całkowitych ze znakiem stosowane w języku C:

Nazwa typu	Rozmiar (bajty)	Zakres wartości
char	1 bajt	-128 … 127
short int	2 bajty	-32 768 … 32 767
int	4 bajty	-2 147 483 648 … 2 147 483 647
long int	4 bajty	-2 147 483 648 … 2 147 483 647
long long int	8 bajtów	-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807

- Przed nazwą każdego z powyższych typów można dodać signed
  signed char, signed short int, signed int …
- W nazwach typów short i long można pominąć słowo int:
  short int → short, long int → long, long long int → long long

Liczby całkowite ze znakiem - kod U2 w języku C

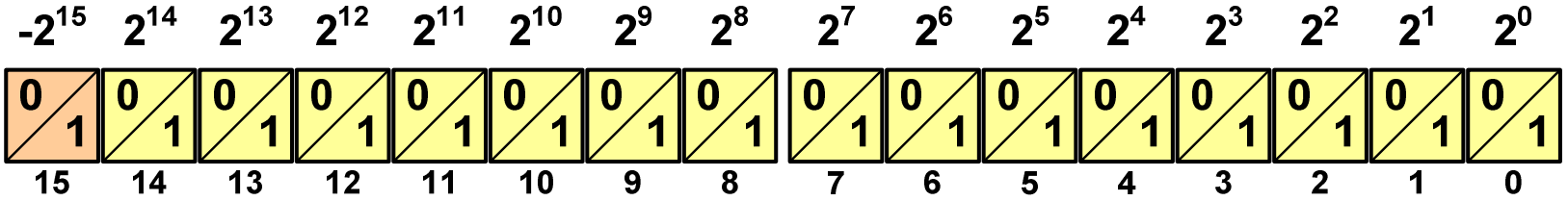
- Typ char / signed char (1 bajt):



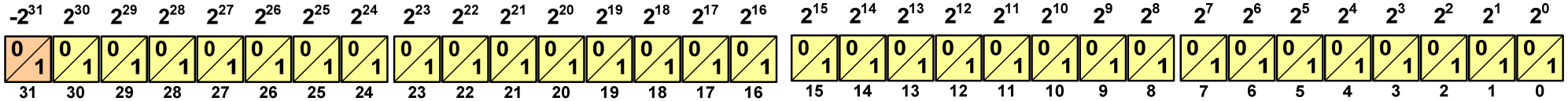
- Zakres wartości:
  - dolna granica: $1000\,0000_{(2)} = -128_{(10)}$
  - górna granica: $0111\,1111_{(2)} = 127_{(10)}$
  - inne wartości: $1111\,1111_{(2)} = -1_{(10)}$
    $0000\,0000_{(2)} = 0_{(10)}$

Liczby całkowite bez znaku w języku C

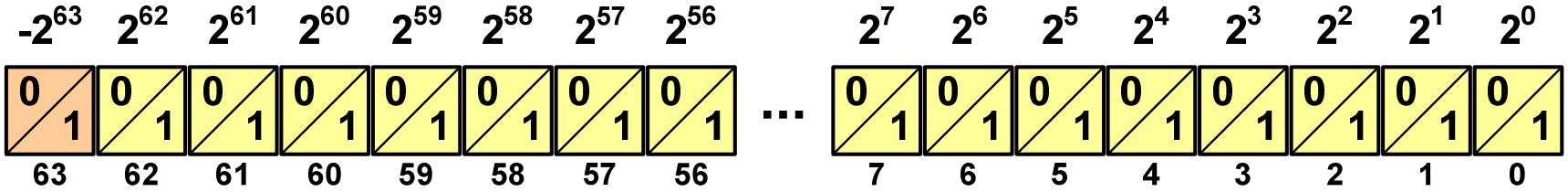
- Typ short / signed short int (2 bajty):



- Typy int / signed int (4 bajty) i long / signed long int (4 bajty):



- Typ long long int / signed long long int (8 bajtów):



Liczby całkowite ze znakiem - kod U2 w języku C

```
short int:      32767 -32768 -32767
int:            2147483647 -2147483648 -2147483647
long int:       2147483647 -2147483648 -2147483647
long long int: 9223372036854775807 -9223372036854775808
```

```
#include

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   short int       si = 32767;
   int              i = 2147483647;
   long int        li = 2147483647;
   long long int lli = 9223372036854775807;

   printf("short int:      %hd %hd %hd\n",si,si+1,si+2);
   printf("int:            %d %d %d\n",i,i+1,i+2);
   printf("long int:       %ld %ld %ld\n",li,li+1,li+2);
   printf("long long int: %lld %lld\n",lli,lli+1);

   return 0;
}
```

Zapis zmiennoprzecinkowy liczby rzeczywistej

- Zapis bardzo dużych lub małych liczb wymaga dużej liczby cyfr
- Znacznie prostsze jest przedstawienie liczb w postaci zmiennoprzecinkowej (ang. floating point numbers)
  - 12 000 000 000 000 = $1{,}2 \cdot 10^{13}$
  - 0,000 000 000 001 = $1{,}0 \cdot 10^{-12}$
- Zapis liczby zmiennoprzecinkowej ma postać:

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

gdzie:

L - wartość liczby
M - mantysa
B - podstawa systemu
E - wykładnik, cecha

- notacja naukowa: 1,2e13 1,2e+13 1,2E13 1,2E+13
- postać wykładnicza: $1{,}2 \cdot 10^{13}$

Postać znormalizowana zapisu liczby

- 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
- W postaci znormalizowanej mantysa spełnia nierówność:

$$B > |M| \geq 1$$

Przykład:

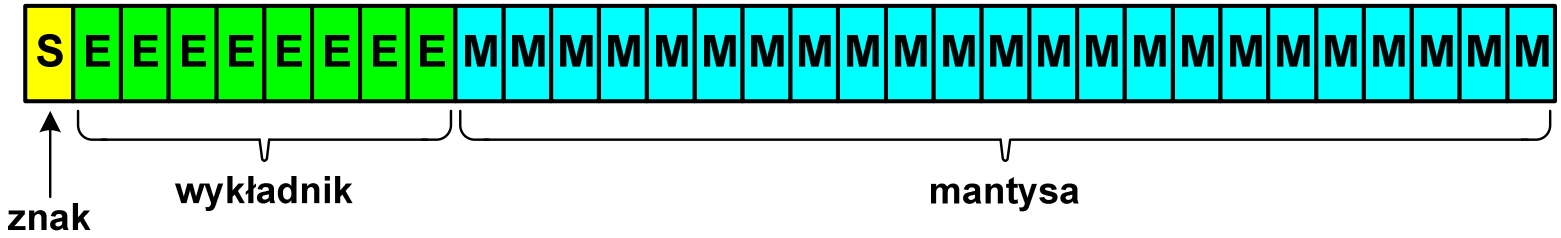
$2{,}43 \cdot 10^3$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

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

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

Liczby zmiennoprzecinkowe w systemie binarnym

- Liczba bitów przeznaczonych na mantysę i wykładnik jest ograniczona



- Wartość liczby L:

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

gdzie:

S - znak liczby (ang. sign), przyjmuje wartość 0 lub 1
M - znormalizowana mantysa (ang. mantissa), liczba ułamkowa
B - podstawa systemu liczbowego (ang. base)
E - wykładnik (ang. exponent), cecha, liczba całkowita

- W systemie binarnym podstawa systemu jest stała: B = 2

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

Przesunięcie wykładnika

- Wykładnik zapisywany jest z przesunięciem (ang. bias)

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

gdzie:

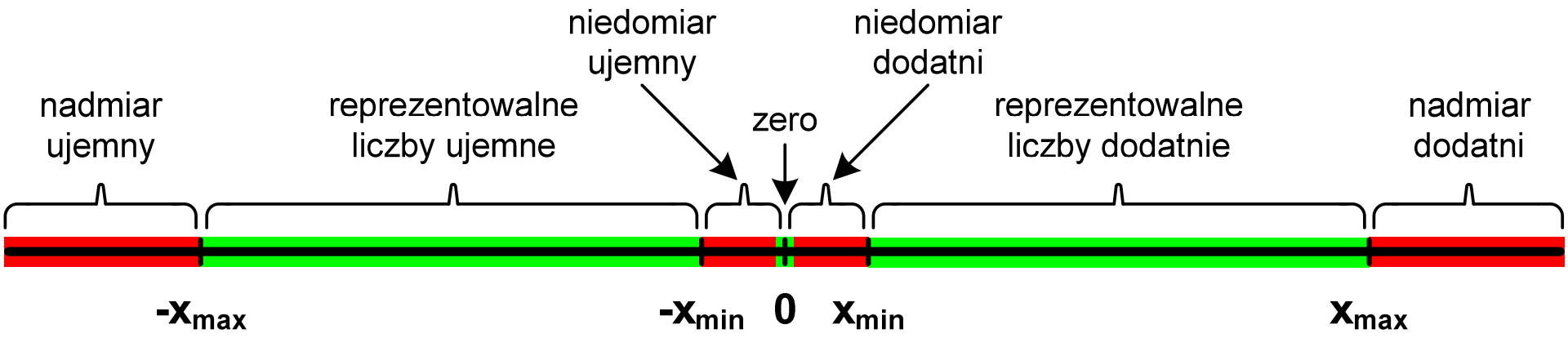
L - wartość liczby S - znak liczby M - mantysa
E - wykładnik BIAS - przesunięcie (nadmiar)

- Typowe wartości przesunięcia (nadmiaru) wynoszą:
  - formatu 32-bitowy: $2^7-1 = 127_{(10)} = 7F_{(16)}$
  - formatu 64-bitowy: $2^{10}-1 = 1023_{(10)} = 3FF_{(16)}$
  - formatu 80-bitowy: $2^{14}-1 = 16383_{(10)} = 3FFF_{(16)}$

Zakres liczb zmiennoprzecinkowych

- Zakres liczb w zapisie zmiennoprzecinkowym:

$$\langle -x_{max}, -x_{min} \rangle \cup \{0\} \cup \langle x_{min}, x_{max} \rangle$$



- Największa i najmniejsza wartość liczby w danej reprezentacji:

$$x_{min} = M_{min} \cdot B^{E_{min}} \qquad x_{max} = M_{max} \cdot B^{E_{max}}$$

Standard IEEE 754

- IEEE Std. 754-2008 - IEEE Standard for Floating-Point Arithmetic
- Standard definiuje następujące klasy liczb zmiennoprzecinkowych:

Precyzja	Długość słowa [bity]	Znak [bity]	Wykładnik		Mantysa	
			Długość [bity]	Zakres	Długość [bity]	Cyfry znaczące
Pojedyncza (Single Precision, binary32)	32	1	8	$2^{\pm 127} \approx 10^{\pm 38}$	23	7
Pojedyncza rozszerzona (Single Extended)	≥ 43	1	≥ 11	$\geq 2^{\pm 1023} \approx 10^{\pm 308}$	≥ 31	≥ 10
Podwójna (Double Precision, binary64)	64	1	11	$2^{\pm 1023} \approx 10^{\pm 308}$	52	16
Podwójna rozszerzona (Double Extended)	≥ 79	1	≥ 15	$\geq 2^{\pm 16383} \approx 10^{\pm 4932}$	≥ 63	≥ 19

źródło: Gryś S.: „Arytmetyka komputerów w praktyce”. PWN, Warszawa, 2007 (str. 116).

Standard IEEE 754

- W przypadku liczb:
  - pojedynczej rozszerzonej precyzji (ang. Single Precision)
  - podwójnej rozszerzonej precyzji (ang. Double Precision)

  standard podaje jedynie minimalną liczbę bitów pozostawiając szczegóły implementacji producentom procesorów i kompilatorów
- Bardzo popularny był 80-bitowy format podwójnej rozszerzonej precyzji (Extended Precision) wprowadzony przez firmę Intel
- W 80-bitowym formacie Intela:
  - długość słowa: 80 bitów
  - znak: 1 bit
  - wykładnik: 15 bitów (zakres: $2^{\pm 16383} \approx 10^{\pm 4932}$)
  - mantysa: 63 bity (cyfry znaczące: 19)

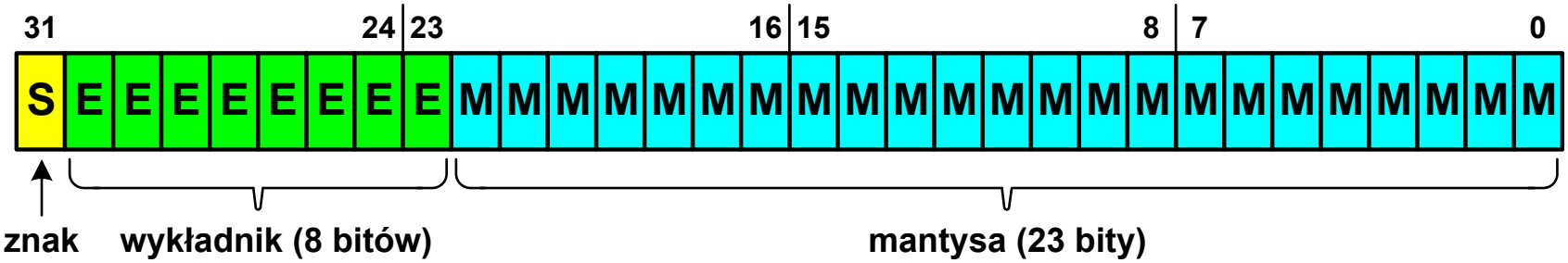
Standard IEEE 754

- Standard IEEE 754 definiuje dziesiętne typy zmiennoprzecinkowe (operujące na cyfrach dziesiętnych):
  - decimal32 (32 bity, 7 cyfr dziesiętnych)
  - decimal64 (64 bity, 16 cyfr dziesiętnych)
  - decimal128 (128 bitów, 34 cyfry dziesiętnych)

- Standard IEEE 754 definiuje:
  - sposób reprezentacji specjalnych wartości, np. nieskończoności, zera
  - sposób wykonywania działań na liczbach zmiennoprzecinkowych
  - sposób zaokrąglania liczb

Standard IEEE 754 - liczby 32-bitowe

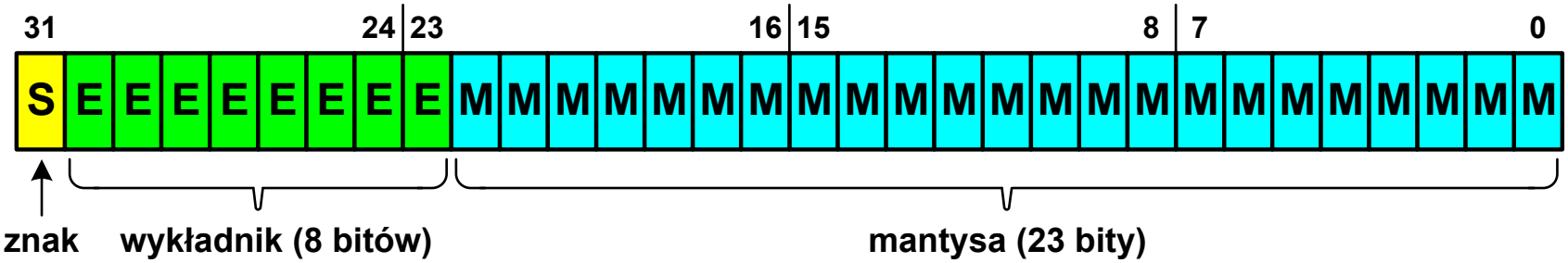
- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- 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)

Standard IEEE 754 - liczby 32-bitowe

- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- Mantysa 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)

Standard IEEE 754 - liczby 32-bitowe

- Przykład:
  - obliczmy wartość dziesiętną liczby zmiennoprzecinkowej

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

  - dzielimy liczbę na części

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

  - określamy znak liczby

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

  - obliczamy wykładnik (nadmiar: 127)

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

Standard IEEE 754 - liczby 32-bitowe

- Przykład (cd.):
  - wyznaczamy mantysę dopisując na początku 1, (część całkowita)

$$M = 1{,}10010000000000000000000 =$$

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

  - wzór na wartość dziesiętną liczby zmiennoprzecinkowej:

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

  - podstawiając otrzymujemy:

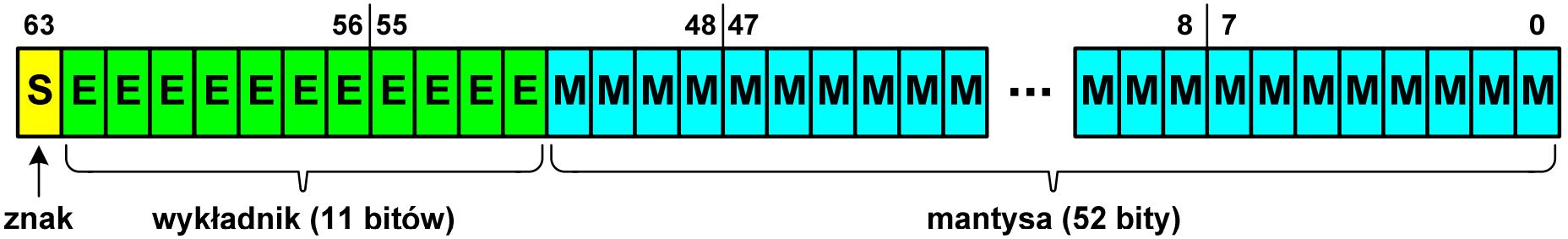
$$S = 0, \qquad E = 6_{(10)}, \qquad M = 1{,}5625_{(10)}$$

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

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

Standard IEEE 754 - liczby 64-bitowe

- Liczba podwójnej precyzji przechowywana jest na 64 bitach:



- Pierwszy bit w zapisie (bit nr 63) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna)
- Wykładnik zapisywany jest na 11 bitach (bity nr 62-52) z nadmiarem o wartości 1023
- Wykładnik może przyjmować wartości od -1023 (wszystkie bity wyzerowane) do 1024 (wszystkie bity ustawione na 1)
- Mantysa zapisywana jest na 52 bitach (pierwszy bit mantysy, zawsze równy 1, nie jest zapamiętywany)

Standard IEEE 754 - zakres liczb

- Pojedyncza precyzja:
  - największa wartość: $\approx 3{,}4 \cdot 10^{38}$
  - najmniejsza wartość: $\approx 1{,}4 \cdot 10^{-45}$
  - zakres liczb: $\langle -3{,}4 \cdot 10^{38} \ldots -1{,}4 \cdot 10^{-45} \rangle \cup \{0\} \cup \langle 1{,}4 \cdot 10^{-45} \ldots 3{,}4 \cdot 10^{38} \rangle$
- Podwójna precyzja:
  - największa wartość: $\approx 1{,}8 \cdot 10^{308}$
  - najmniejsza wartość: $\approx 4{,}9 \cdot 10^{-324}$
  - zakres liczb: $\langle -1{,}8 \cdot 10^{308} \ldots -4{,}9 \cdot 10^{-324} \rangle \cup \{0\} \cup \langle 4{,}9 \cdot 10^{-324} \ldots 1{,}8 \cdot 10^{308} \rangle$
- Podwójna rozszerzona precyzja:
  - największa wartość: $\approx 1{,}2 \cdot 10^{4932}$
  - najmniejsza wartość: $\approx 3{,}6 \cdot 10^{-4951}$
  - zakres liczb: $\langle -1{,}2 \cdot 10^{4932} \ldots -3{,}6 \cdot 10^{-4951} \rangle \cup \{0\} \cup \langle 3{,}6 \cdot 10^{-4951} \ldots 1{,}2 \cdot 10^{4932} \rangle$

Standard IEEE 754 - precyzja liczb

- Precyzja - liczba zapamiętywanych cyfr znaczących w systemie (10)

  4,86452137846 → 4,864521 - 7 cyfr znaczących

- Precyzja liczby zależy od liczby bitów mantysy
- Liczba bitów potrzebnych do zakodowania 1 cyfry dziesiętnej:

$$10^1 = 2^n \quad \rightarrow \quad n = \log_2(10) \approx 3{,}321928$$

- Liczba cyfr dziesiętnych (d) możliwa do zakodowania na m bitach:

  $\log_2(10)$ bitów - 1 cyfra dziesiętna

  m bitów - d cyfr dziesiętnych

$$d = \frac{m}{\log_2(10)}$$

Standard IEEE 754 - precyzja liczb

- Dla formatu pojedynczej precyzji:
  - mantysa: 23 + 1 = 24 bity
  - cyfry znaczące: 7

$$d = \frac{24}{\log_2(10)} = \frac{24}{3{,}321928} = 7{,}2247 \approx 7$$

- Dla formatu podwójnej precyzji:
  - mantysa: 52 + 1 = 53 bity
  - cyfry znaczące: 16

$$d = \frac{53}{\log_2(10)} = \frac{53}{3{,}321928} = 15{,}9546 \approx 16$$

- Dla formatu podwójnej rozszerzonej precyzji:
  - mantysa: 63 + 1 = 64 bity
  - cyfry znaczące: 19

$$d = \frac{64}{\log_2(10)} = \frac{64}{3{,}321928} = 19{,}2659 \approx 19$$

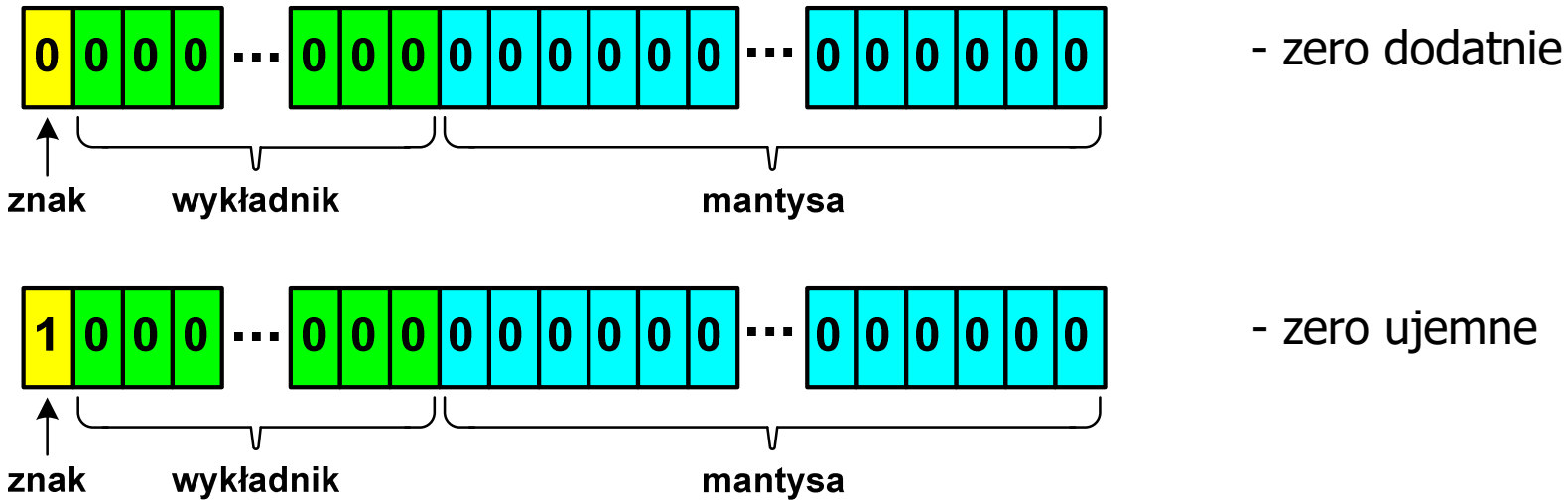
Standard IEEE 754 - precyzja liczb

```c
#include <stdio.h>

int main()
{
    float x;
    double y;

    x = 1234567890.0;    /* 1.234.567.890 */
    y = 1234567890.0;    /* 1.234.567.890 */
    printf("float  -> %f\n",x);
    printf("double -> %f\n\n",y);

    y = 12345678901234567890.0;
    printf("double -> %f\n",y);

    return 0;
}
```

```
float  -> 1234567936.000000
double -> 1234567890.000000

double -> 12345678901234567000.000000
```

Standard IEEE 754 - wartości specjalne

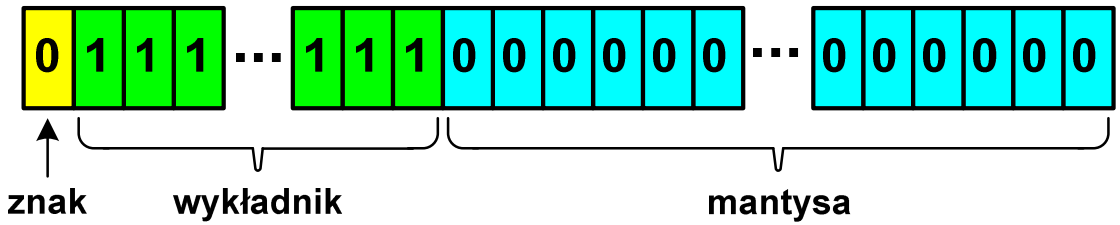
- Zero:



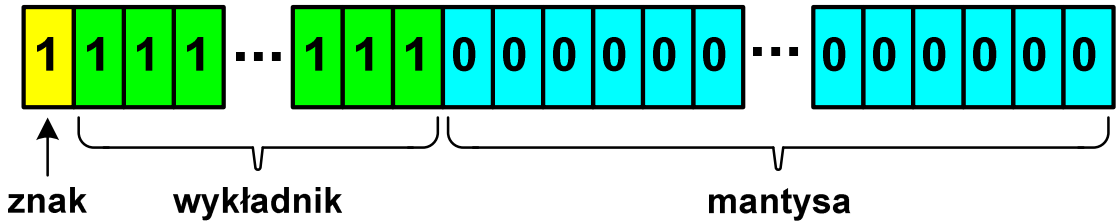
- Podczas porównań zero dodatnie i ujemne są traktowane jako równe sobie

Standard IEEE 754 - wartości specjalne

- Nieskończoność:



- nieskończoność dodatnia

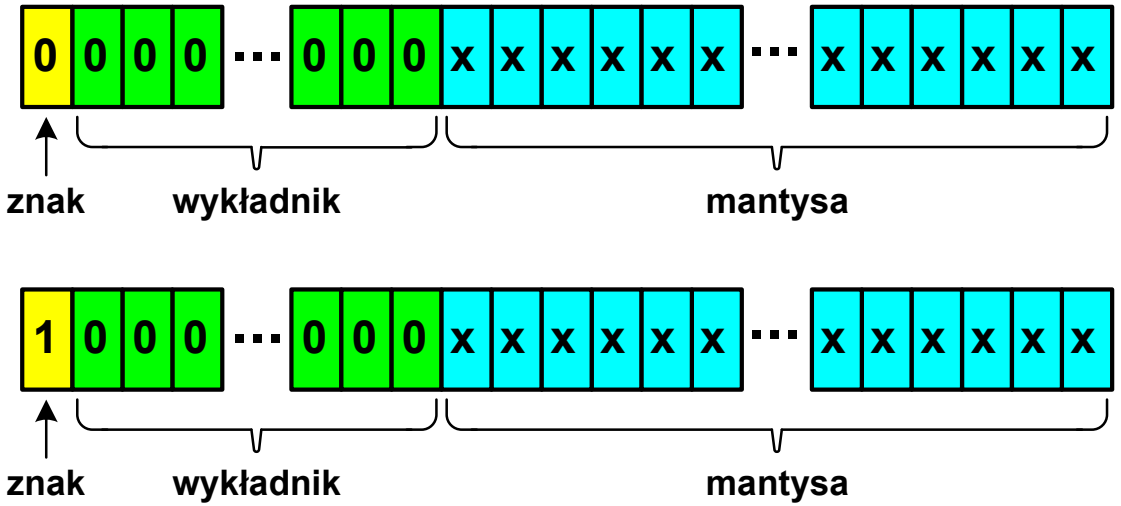


- nieskończoność ujemna

- Nieskończoność występuje w przypadku wystąpienia nadmiaru (przepełnienia) oraz przy dzieleniu przez zero

Standard IEEE 754 - wartości specjalne

- Liczba zdenormalizowana:

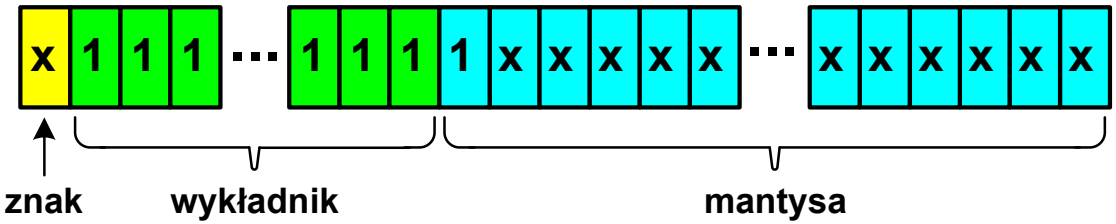


- Pojawia się, gdy występuje niedomiar (ang. underflow), ale wynik operacji można jeszcze zapisać denormalizując mantysę
- Mantysa nie posiada domyślnej części całkowitej równej 1, tzn. reprezentuje liczbę o postaci 0,xxx…xxx, a nie 1,xxx…xxx

Standard IEEE 754 - wartości specjalne

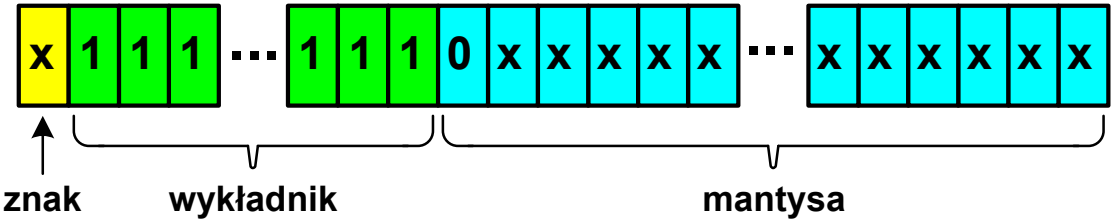
- Nieliczby - NaN (Not A Number) - nie reprezentują wartości liczbowej
- Powstają w wyniku wykonania niedozwolonej operacji

- QNaN (ang. Quiet NaN) - ciche nieliczby



- „przechodzą” przez działania arytmetyczne (brak przerwania wykonywania programu)

- SNaN (ang. Signaling NaN) - sygnalizujące, istotne, głośne nieliczby



- zgłoszenie wyjątku (przerwanie wykonywania programu)

Koniec wykładu nr 3

Dziękuję za uwagę!