

Informatyka 1 (ES1F1002)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr I, studia stacjonarne I stopnia
Rok akademicki 2024/2025

Wykład nr 3 (18.10.2024)

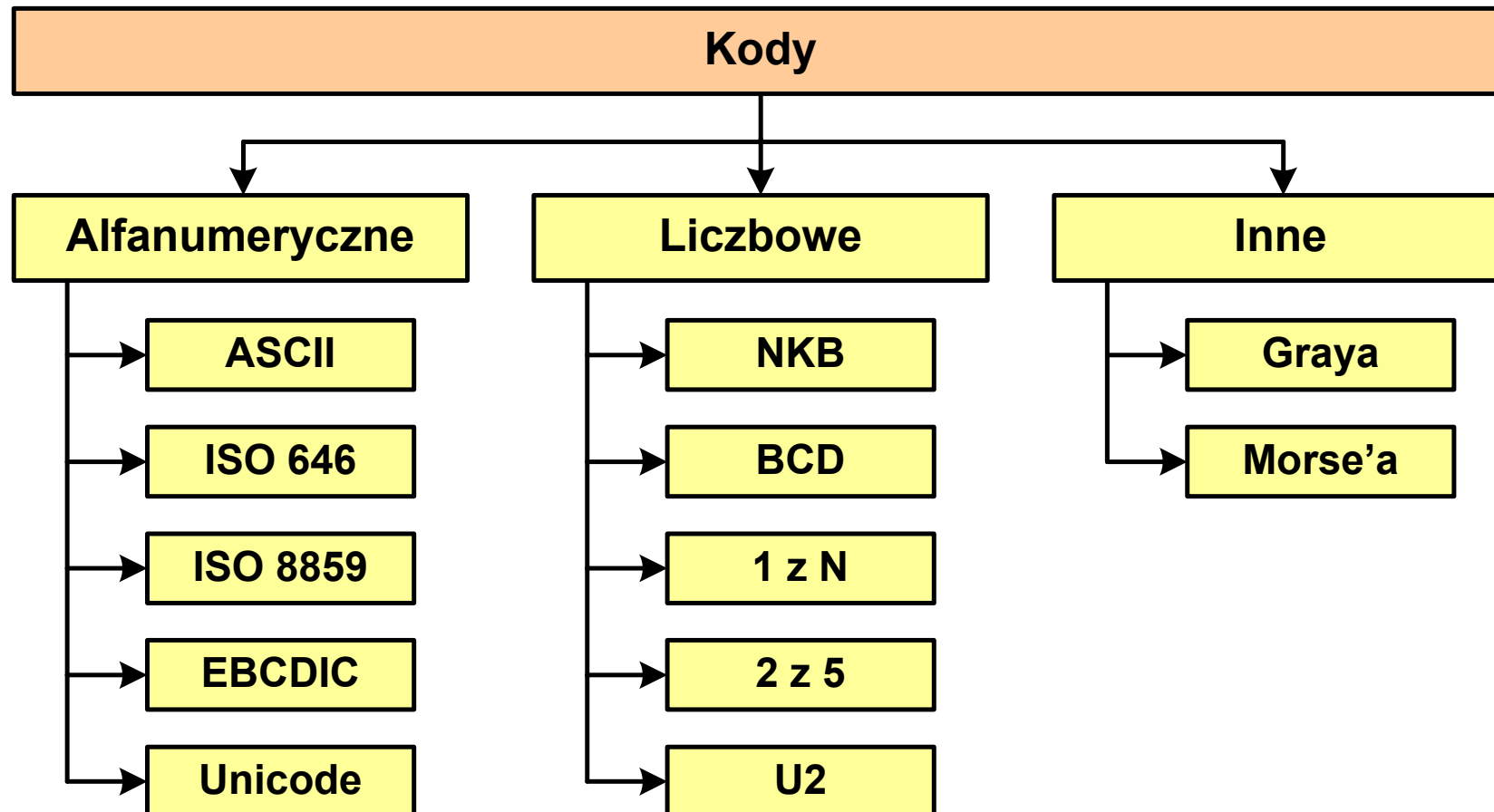
dr inż. Jarosław Forenc

Plan wykładu nr 3

- Kodowanie znaków
 - ASCII, ISO/IEC 646, ISO 8859
 - EBCDIC, Windows-1250, Unicode
- Kodowanie liczb
 - NKB, BCD, kod 2 z 5, kod Graya
- Reprezentacja liczb całkowitych
 - liczby bez znaku i ze znakiem (ZM, U1, U2)
- Reprezentacja zmiennoprzecinkowa
 - zapis, postać znormalizowana
 - zakres liczb zmiennoprzecinkowych

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 transmission)	20	14	DC4 (device control 4)
5	5	ENQ (enquiry)	21	15	NAK (negative acknowledge)
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 (substitute)
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 - $10_{(10)} = 0A_{(16)}$

- Założmy, 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 | plikuDrugi wier
00000020: 73 7A 20 70 6C 69 6B 75|0A 54 72 7A 65 63 69 20 | sz plikuTrzeci
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 plikuDrugi wiersz plikuTrzeci wiersz pliku
```

ISO/IEC 646

- **ISO/IEC 646** - norma definiująca modyfikację 7-bitowego kodowania ASCII, stosowana w latach 70-tych i 80-tych
- W normie określono 10 pozycji na znaki w języku kraju, który przyjął tę normę oraz 2 pozycje na znaki walut

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
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	{		}	~	

- żółty - znaki narodowe
- niebieski - znaki walut

- Wszystkie pozostałe znaki są zgodne z ASCII

ISO/IEC 646 - odmiany narodowe

USA

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
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	{		}	~	

Niemcy

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
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	Ä	Ö	Ü	^	_
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	ä	ö	ü	ß	

Francja

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
10																
20	SP	!	"	£	\$	%	&	\	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	à	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	°	ç	§	^	_
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	é	ù	é	¨	

Polska

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
10																
20	SP	!	"	#	zł	%	&	\	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	ę	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	ź	\	ń	ś	_
60	ą	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	ó	ł	ź	ć	

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 czerwca 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 8859-1

- ISO/IEC 8859-1, Latin-1 („zachodnioeuropejskie”)
- kodowanie używane w Amerykach, Europie Zachodniej, Oceanii i większej części Afryki
- dostępne języki: albański, angielski, baskijski, duński, estoński, fiński, francuski, hiszpański, irlandzki, islandzki, kataloński, łaciński, niderlandzki, niemiecki, norweski, portugalski, retoromański, szkocki, szwedzki, włoski
- 191 znaków łacińskiego pisma.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
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	<i>Nieuzywane</i>															
90																
A0	NBSP	ı	ç	£	¤	¥	ı	§	¨	©	ª	«	¬	SHY	®	—
B0	°	±	²	³	´	µ	¶	·	,	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

SP - spacja

NBSP - twarda spacja

SHY - miękki dywiz (myślnik)

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	<i>Znaki kontrolne</i>															
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	<i>Nie używane</i>															
90																
A0	NB SP	À	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ć	Č	Ď	Ě	Ž	Ž
B0	°	ą	.	ł	´	ł	ś	˘	.	š	š	ť	ž	˘	ž	ž
C0	Ř	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ć	Č	Ď	Ě	Ž	Ž	Ž
D0	Đ	Ń	Ň	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ü	Ý	Ť	ß
E0	ř	á	â	ã	ä	å	ā	ă	ą	ć	č	ď	ě	ž	ž	ž
F0	đ	ń	ň	ó	ô	õ	ö	÷	ř	ů	ú	ů	ü	ý	ť	·

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

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

■ 18 liter:

- Ā - ā
- 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	<i>Znaki kontrolne</i>															
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	<i>Nie używane</i>															
90																
A0	NB SP	Ā	˘	Ł	ł	Ĺ	Ś	ś	˙	Š	š	Ť	Ž	SHY	Ž	Ž
B0	°	ā	˘	ł	ł	ĺ	ś	ś	˙	š	š	ť	ž	ˆ	ž	ž
C0	Ř	Á	Â	Ě	Ä	Í	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ü	Ý	Ŧ	ß
E0	ř	á	â	ě	ä	í	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	õ	ö	÷	ř	ů	ú	ů	ü	ý	ț	·

ISO/IEC 8859-1 i ISO/IEC 8859-2 - porównanie

ISO 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	NB SP	ı	ç	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

ISO 8859-2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	NB SP	Ą	˘	Ł	¤	Ł	Ś	§	¨	Š	Ş	Ť	Ž	SHY	Ž	Ž
B0	°	ą	˙	ł	´	ł	ś	˘	š	ş	ť	ž	˝	ž	ž	
C0	Ř	Á	Â	Ã	Ä	Í	Č	Ç	Č	É	Ě	Ë	Ě	Í	Î	Ď
D0	Ð	Ñ	Ň	Ó	Ô	Õ	Ö	×	Ř	Ú	Ú	Û	Ü	Ý	Ť	ß
E0	ř	á	â	ã	ä	í	č	ç	č	é	ě	ë	ě	í	î	ď
F0	ř	ń	ň	ó	ô	ó	ö	÷	ř	ů	ú	ú	ü	ý	ť	·

EBCDIC

- **EBCDIC** - Extended Binary Coded Decimal Interchange Code
- 8-bitowe kodowanie znaków stworzone jako rozszerzenie kodowania BCD

- używane głównie w systemach IBM w latach 60-tych XX wieku
- umożliwia zapisanie do 256 różnych symboli
- brak zachowania kolejności liter zgodnie z kolejnością kodów, np. po R nie ma S
- kody EBCDIC **nie są zgodne** z ASCII.

EBCDIC

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
10																
20																
30																
40	SP	NB SP	â	ä	à	á	ã	å	ç	ñ	[.	<	(+	!
50	&	é	ê	ë	è	í	î	ï	ì	ß]	\$	*)	;	^
60	-	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ		,	%	_	>	?
70	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
80	∅	a	b	c	d	e	f	g	h	i	<<	>>	ø	ý	þ	±
90	°	j	k	l	m	n	o	p	q	r	^a	°	æ	,	Æ	¤
A0	µ	~	s	t	u	v	w	x	y	z	i	¿	Ð	Ý	þ	®
B0	¢	£	¥	·	©	§	¶	¼	½	¾	¬		¯	¨	'	×
C0	{	A	B	C	D	E	F	G	H	I	SHY	ô	ö	ò	ó	õ
D0	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E0	\	÷	S	T	U	V	W	X	Y	Z	²	ô	ö	ò	ó	õ
F0	0	1	2	3	4	5	6	7	8	9	³	û	ü	ù	ú	

EBCDIC i ISO 8859-1 - porównanie

ISO 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
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	<i>Nieużywane</i>															
90																
A0	NB SP	ı	ç	£	¤	¥	¦	§	¨	©	^a	«	¬	SHY	®	¯
B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

EBCDIC

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	<i>Znaki kontrolne</i>															
10																
20	<i>Znaki kontrolne</i>															
30																
40	SP	NB SP	â	ä	à	á	ã	å	ç	ñ	[.	<	(+	!
50	&	é	ê	ë	è	í	î	ï	ì	ß]	\$	*)	;	^
60	-	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ		,	%	_	>	?
70	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	'	:	#	@	'	=	"
80	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
90	°	j	k	l	m	n	o	p	q	r	^a	°	æ	,	Æ	¤
A0	µ	~	s	t	u	v	w	x	y	z	i	¿	Ð	Ý	Þ	®
B0	ç	£	¥	·	©	§	¶	¼	½	¾	¬		¯	¨	'	x
C0	{	A	B	C	D	E	F	G	H	I	SHY	ô	ö	ò	ó	õ
D0	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E0	\	÷	S	T	U	V	W	X	Y	Z	²	ô	ö	ò	ó	õ
F0	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Windows-1250

- **Windows-1250 (CP-1250)** - strona kodowa używana przez system Microsoft Windows do reprezentacji tekstów w językach środkowoeuropejskich używających alfabetu łacińskiego
- Obsługiwane języki: albański, chorwacki, czeski, polski, rumuński, słowacki, słoweński, węgierski (ale także niemiecki)
- Windows-1250 jest podobny do ISO 8859-2 - posiada wszystkie jego drukowalne znaki (a także kilka dodatkowych), lecz kilka z nich zajmuje inne miejsca.

ISO 8859-2 i Windows-1250 - porównanie

ISO 8859-2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F							
00	<i>Znaki kontrolne</i>																						
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	<i>Nie używane</i>																						
90																							
A0	NBS P	À	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ć	Č	Ĉ	Ď	Ě	Ǽ	Š	Ť	Ž	SHY	Ž	Ž	
B0	°	à	á	â	ã	ä	å	ā	ă	ą	ć	č	ĉ	ď	ě	ǽ	š	ť	ž	ˆ	ž	ž	
C0	Ř	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ć	Č	Ĉ	Ď	Ě	Ǽ	Š	Ť	Ž					
D0	Đ	Ń	Ň	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ů	Ý	Ť	ß							
E0	ř	á	â	ã	ä	å	ā	ă	ą	ć	č	ĉ	ď	ě	ǽ	š	ť	ž					
F0	đ	ñ	ň	ó	ô	õ	ö	÷	ř	ů	ú	ů	ů	ý	ť	·							

Windows-1250

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F							
00	<i>Znaki kontrolne</i>																						
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	€		/		//	...	†	‡		%	Š	<	Ś	Ť	Ž	Ž							
90		\	'	"	"	•	-	-		™	š	>	ś	ť	ž	ž							
A0	NBS P	ˆ	ˆ	Ł	ł	Ą	!	Ś	ˆ	©	Ş	<<	ı	SHY	Ž	Ž							
B0	°	±	.	ł	'	µ	¶	·	.	ş	>>	ı	ˆ	ı	ž								
C0	Ř	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ć	Č	Ĉ	Ď	Ě	Ǽ	Š	Ť	Ž					
D0	Đ	Ń	Ň	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ů	Ý	Ť	ß							
E0	ř	á	â	ã	ä	å	ā	ă	ą	ć	č	ĉ	ď	ě	ǽ	š	ť	ž					
F0	đ	ñ	ň	ó	ô	õ	ö	÷	ř	ů	ú	ů	ů	ý	ť	·							

Problem kodowania polskich liter diakrytycznych

- Problem z wyświetlaniem polskich liter diakrytycznych

- Tekst zapisany w standardzie ISO-8859-2:

Ą Ć Ę Ł Ń Ó Ś Ź Ż
ą ć ę ł ń ó ś ź ż

- Tekst wyświetlony w Notatniku systemu Windows (Windows-1250):

˘ ́ Ć Ę Ł Ń Ó ; ̄ Ż
± ć ę ł ń ó ¶ ℓ ź

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
- Konsorcjum: <http://www.unicode.org>
- Pierwsza wersja: **Unicode 1.0** (październik 1991)
- Ostatnia wersja: **Unicode 16.0.0** (10 września 2024)
 - The Unicode Consortium. The Unicode Standard, Version 16.0.0, (South San Francisco: The Unicode Consortium, 2024. ISBN 978-1-936213-34-4)
 - <https://www.unicode.org/versions/Unicode16.0.0/>
 - Koduje 154.998 znaków



Unicode - Zakresy

<u>Zakres:</u>	<u>Znaczenie:</u>
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-8gdzie: **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 opisanego kodu znaku

A	Ω	語	𐄎	UTF-32
00000041	000003A9	00008A9E	00010384	
A	Ω	語	𐄎	UTF-16
0041	03A9	8A9E	D800 DF84	
A	Ω	語	𐄎	UTF-8
41	CE A9	E8 AA 9E	F0 90 8E 84	



Unicode - kodowanie UTF-32

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

A	Ω	語	卍
00000041	000003A9	00008A9E	00010384

UTF-32

- 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 0x3FFFFFFF - 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	Ð 0110	Ġ 0120	İ 0130	ı 0140	Ō 0150	Š 0160	Ū 0170
1	ā 0101	đ 0111	ġ 0121	ı 0131	Ł 0141	ō 0151	š 0161	ū 0171
2	Ǻ 0102	Ē 0112	Ɔ 0122	IJ 0132	ł 0142	Œ 0152	Ț 0162	Ț 0172
3	ǻ 0103	ē 0113	ġ 0123	ij 0133	Ń 0143	œ 0153	ț 0163	ț 0173
4	Ą 0104	Ě 0114	Ĥ 0124	Ĵ 0134	ń 0144	Ŕ 0154	Ť 0164	Ŵ 0174
5	ą 0105	ě 0115	ĥ 0125	ĵ 0135	Ń 0145	ŕ 0155	ť 0165	ŵ 0175
6	Ć 0106	Ĕ 0116	Ħ 0126	Ɔ 0136	ņ 0146	Ŗ 0156	Ʀ 0166	Ŷ 0176
7	ć 0107	ĕ 0117	ħ 0127	Ɔ 0137	ņ 0147	ŗ 0157	ƣ 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 ħ cyrillic small letter tshe
 ≡ 0063 c 0301 ́

Unicode



27308

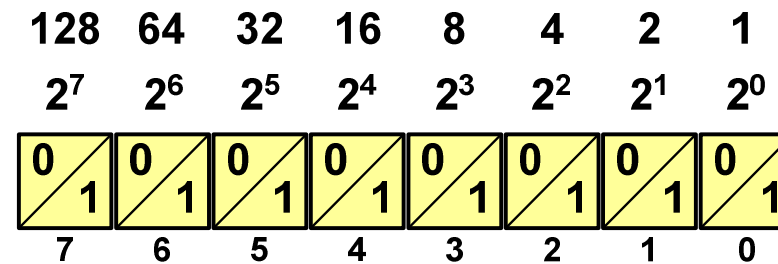
CJK Unified Ideographs Extension B

27342

27308 虫 142.8	𧈧	𧈧	𧈧	2731B 虫 142.8	𧈧	𧈧	𧈧	2732F 虫 142.8	𧈧	𧈧
	UCS2003	GKX-1086.03	T4-4721		UCS2003	GKX-1088.15	T6-617B		UCS2003	GHC
27309 虫 142.8	𧈩	𧈩	𧈩	2731C 虫 142.8	𧈩	𧈩	𧈩	27330 虫 142.9	𧈩	𧈩
	UCS2003	GKX-1086.05	T5-4955		UCS2003	GKX-1088.16	T6-6221		UCS2003	GHC
2730A 虫 142.8	𧈪	𧈪	𧈪	2731D 虫 142.8	𧈪	𧈪	𧈪	27331 虫 142.8	𧈪	𧈪
	UCS2003	GKX-1086.08	T4-467D		UCS2003	GKX-1088.17	T5-4960		UCS2003	G4K
2730B 虫 142.8	𧈫	𧈫	𧈫	2731E 虫 142.7	𧈫	𧈫		27332 虫 142.8	𧈫	𧈫
	UCS2003	GKX-1086.10	T6-6223		UCS2003	GKX-1088.18			UCS2003	GHC
2730C 虫 142.8	𧈬	𧈬	𧈬	2731F 虫 142.8	𧈬	𧈬	𧈬	27333 虫 142.8	𧈬	𧈬
	UCS2003	GKX-1086.12	T5-495F		UCS2003	GKX-1088.19	T6-6174		UCS2003	GHC
2730D 虫 142.8	𧈭	𧈭	𧈭	27320 虫 142.8	𧈭	𧈭	𧈭	27334 虫 142.8	𧈭	𧈭
	UCS2003	GKX-1086.22	T4-4677		UCS2003	GKX-1088.20	T6-617D		UCS2003	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

- **B**inary-**C**oded **D**ecimal - 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 $0 \div 9$
- Pozostałe kombinacje bitowe mogą być stosowane do kodowania znaku liczby lub innych znaczników.

Kody liczbowe - Kod BCD

■ Przykład:

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

$$\begin{array}{ccc} \overbrace{0001}^1 & \overbrace{0110}^6 & \overbrace{1000}^8 \\ 0001 & 0110 & 1000 \end{array}$$

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

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

$$\begin{array}{ccc} \underbrace{1001}_9 & \underbrace{0101}_5 & \underbrace{0011}_3 \\ 1001 & 0101 & 0011 \end{array}$$

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

■ 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.

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

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

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)

Kody liczbowe - Kod 2 z 5

- Kod 5-bitowy: 2 bity zawsze równe 1, a 3 bity zawsze równe 0
- Koduje 10 znaków (cyfry dziesiętne), kody nie są wzajemnie jednoznaczne (ta sama wartość może być zakodowana w różny sposób)

- Kod stałowagowy

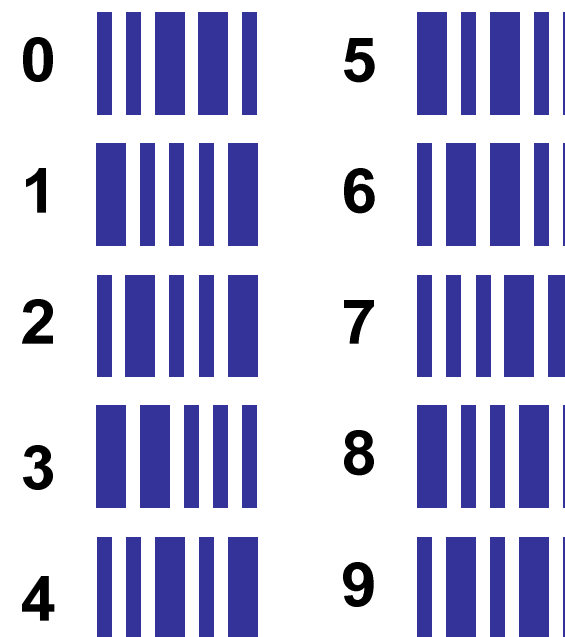
- Kod detekcyjny

- Stosowany głównie w **kodach kreskowych**

Liczba dziesiętna	2 z 5 (01236)	2 z 5 (01234)	2 z 5 (74210)
0	01100	01100	11000
1	11000	11000	00011
2	10100	10100	00101
3	10010	10010	00110
4	01010	01010	01001
5	00110	00110	01010
6	10001	10001	01100
7	01001	01001	10001
8	00101	00101	10010
9	00011	00011	10100

Kody liczbowe - Kod 2 z 5 Industrial (1960 r.)

- Jednowymiarowy kod kreskowy kodujący cyfry: 0 ÷ 9
- Znak to 5 pasków: 2 szerokie i 3 wąskie
- Szeroki pasek jest wielokrotnością wąskiego, szerokości muszą być takie same dla całego kodu
- Struktura kodu:
 - start: 11011010
 - numer
 - stop: 11010110



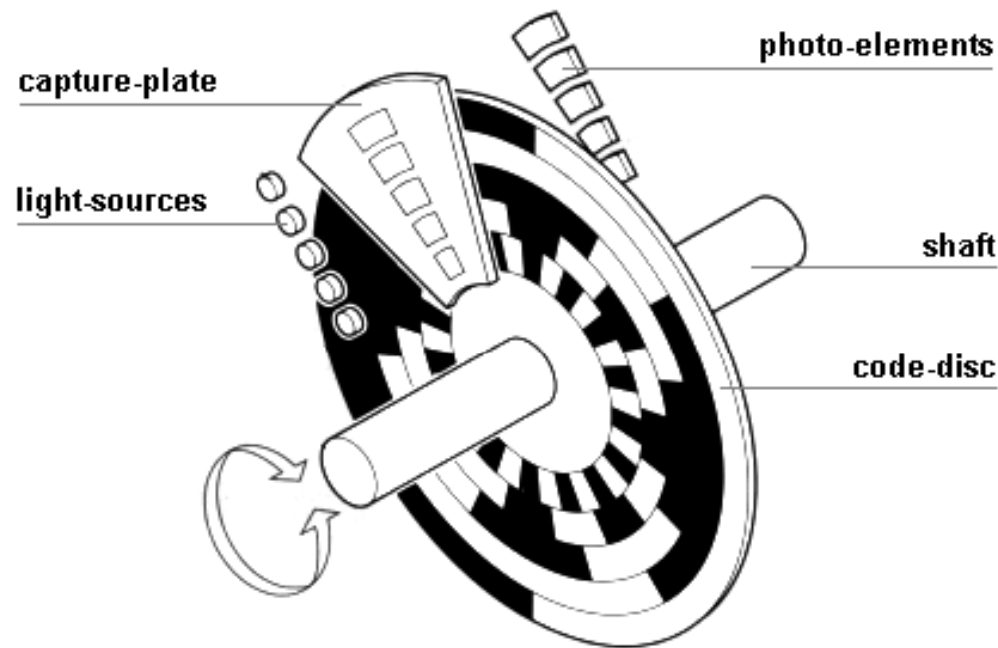
Kod Graya (refleksyjny)

- 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

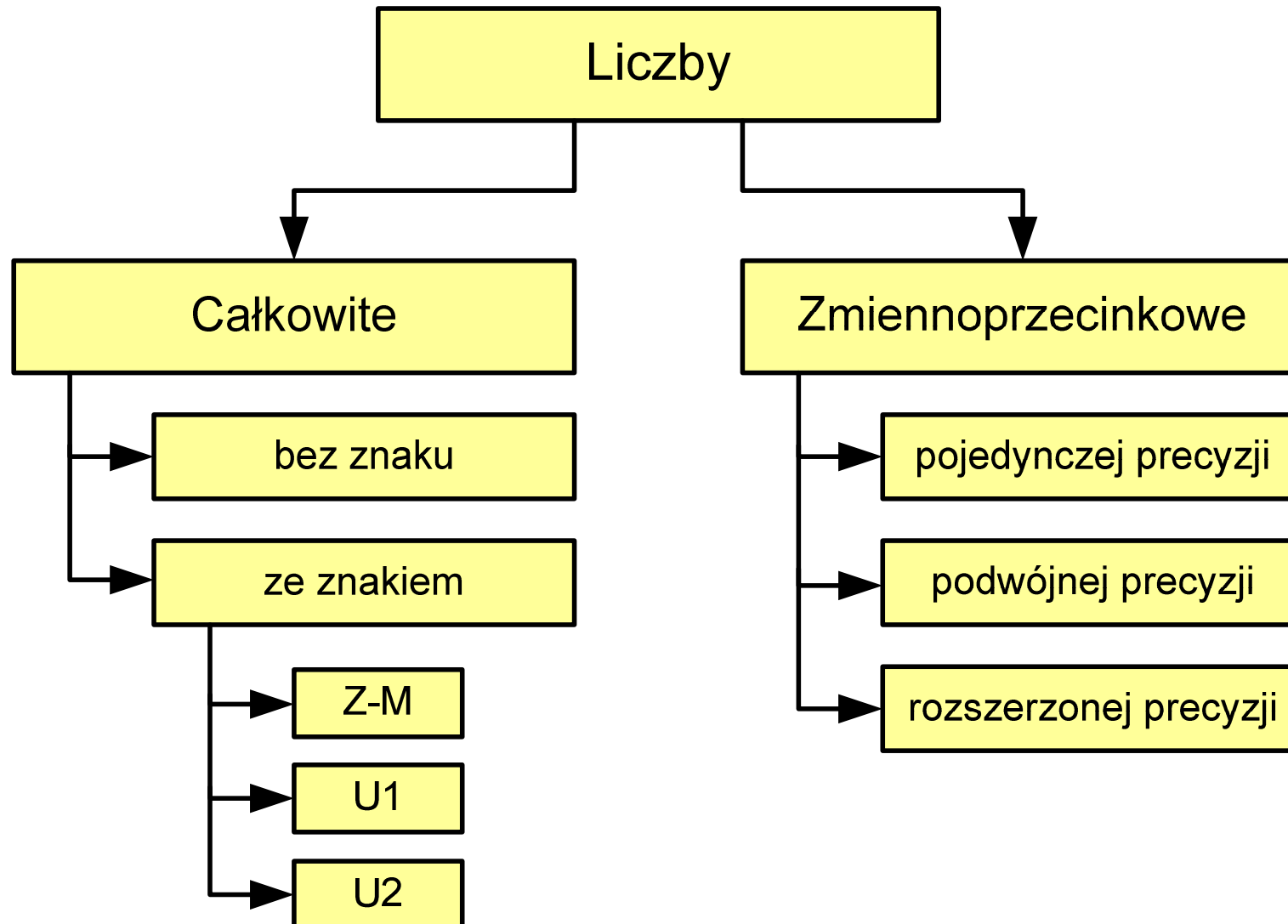
<u>kod 1-bitowy</u>	<u>kod 2-bitowy</u>	<u>kod 3-bitowy</u>
0	00	000
1	01	001
	<u>11</u>	011
	10	<u>010</u>
		110
		111
		101
		100

Kod Graya

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

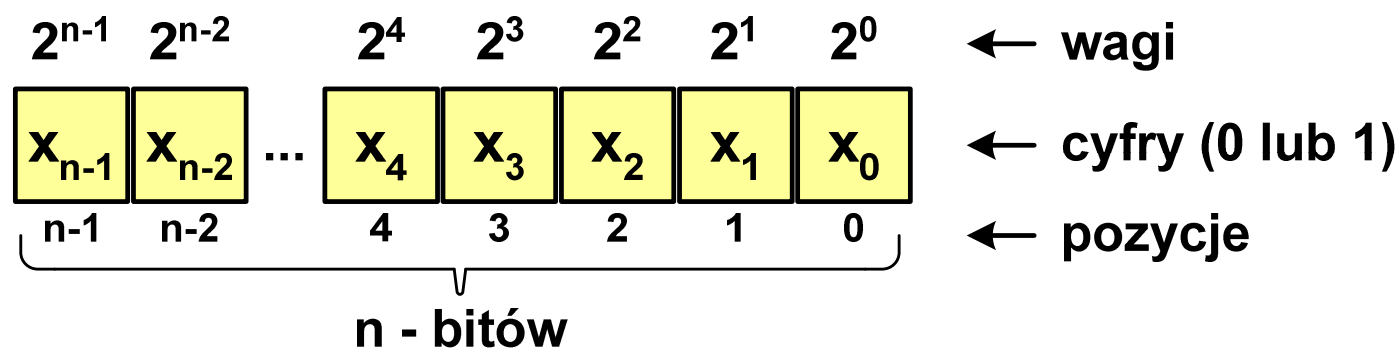


Reprezentacja liczb w systemach komputerowych



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:

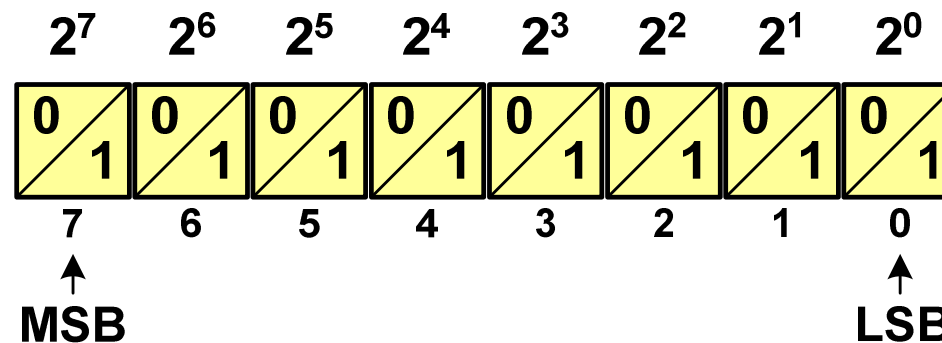
<u>Nazwa typu</u>	<u>Rozmiar (bajty)</u>	<u>Zakres wartości</u>
<code>unsigned char</code>	1 bajt	0 ... 255
<code>unsigned short int</code>	2 bajty	0 ... 65 535
<code>unsigned int</code>	4 bajty	0 ... 4 294 967 295
<code>unsigned long int</code>	4 bajty	0 ... 4 294 967 295
<code>unsigned long long int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

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

<code>unsigned short int</code>	→	<code>unsigned short</code>
<code>unsigned long int</code>	→	<code>unsigned long</code>
<code>unsigned long long int</code>	→	<code>unsigned long long</code>

Liczby całkowite bez znaku w języku C

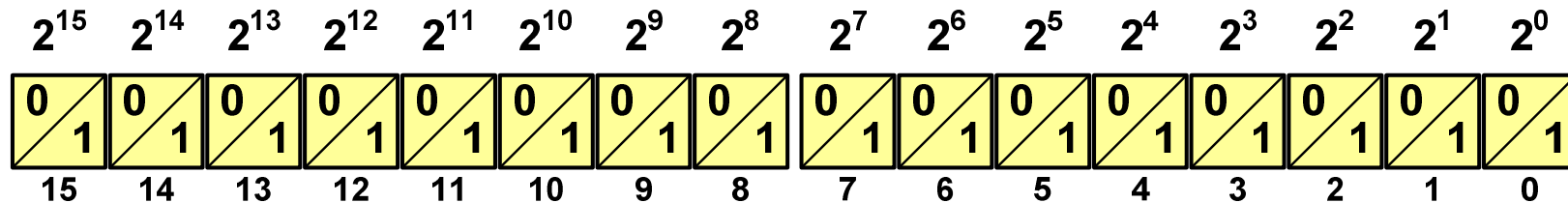
- Typ **unsigned char** (1 bajt):



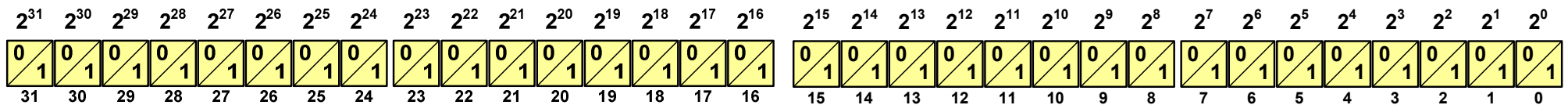
- **MSB** (Most Significant Bit) - najbardziej znaczący bit, najstarszy bit, największa waga
 - **LSB** (Least Significant Bit) - najmniej znaczący bit, najmłodszy bit, najmniejsza waga
- Zakres wartości:
 - dolna granica: $0000\ 0000_{(2)} = 00_{(16)} = 0_{(10)}$
 - górna granica: $1111\ 1111_{(2)} = FF_{(16)} = 255_{(10)}$

Liczby całkowite bez znaku w języku C

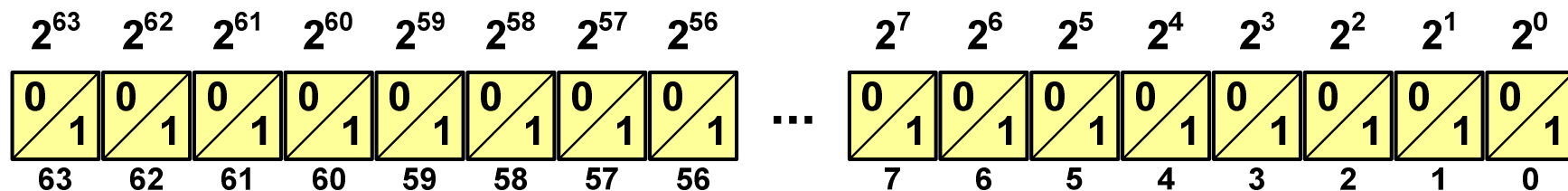
- Typ **unsigned short int** (2 bajty):



- 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
```

```
#include <stdio.h>

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
```

```
#include <stdio.h>

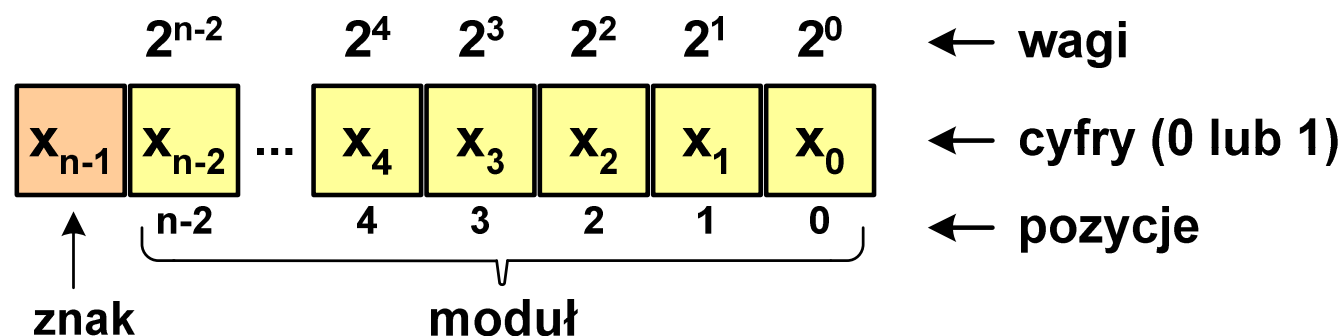
int main() /* przepełnienie zmiennej, ang. integer overflow */
{
    unsigned short int    usi = 1;
    unsigned int          ui = 1;
    unsigned long int     uli = 1;
    unsigned long long int ulli = 1;

    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:

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

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**:

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

dla 8 bitów : $\langle -127 \dots 127 \rangle$

dla 16 bitów : $\langle -32767 \dots 32767 \rangle$

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

$$93_{(10)} = \mathbf{0}1011101_{(ZM)}$$

- liczba ujemna

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

- zamieniamy **moduł** liczby na NKB

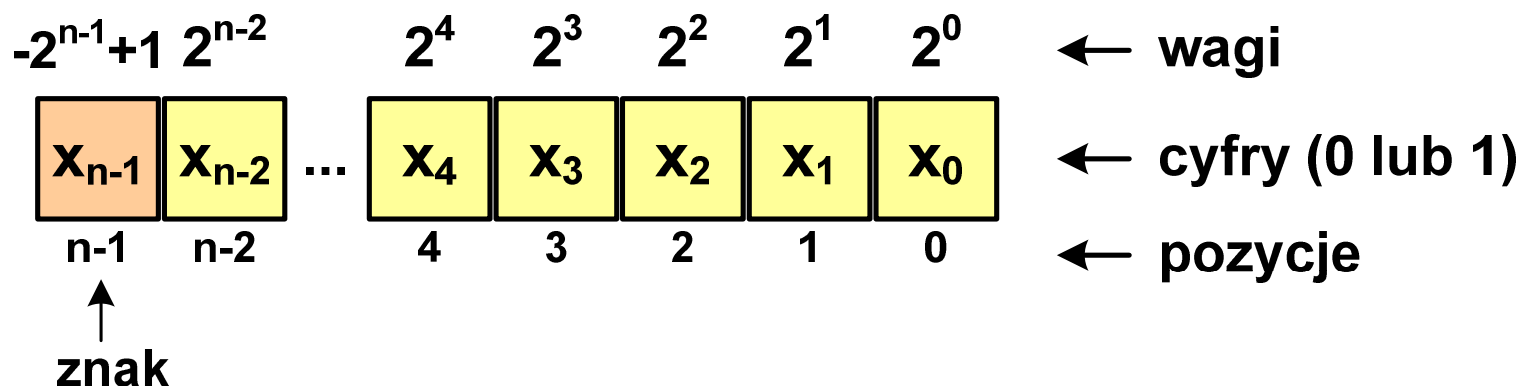
$$|-93_{(10)}| = 93_{(10)} = 1011101_{(NKB)}$$

- dodajemy bit znaku

$$-93_{(10)} = \mathbf{1}1011101_{(ZM)}$$

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:

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

Liczby całkowite ze znakiem - kod U1

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

U1	dziesiętnie	U1	dziesiętnie
0000	+0	1111	-0
0001	1	1110	-1
0010	2	1101	-2
0011	3	1100	-3
0100	4	1011	-4
0101	5	1010	-5
0110	6	1001	-6
0111	7	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**:

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

dla 8 bitów : $\langle -127 \dots 127 \rangle$

dla 16 bitów : $\langle -32767 \dots 32767 \rangle$

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

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

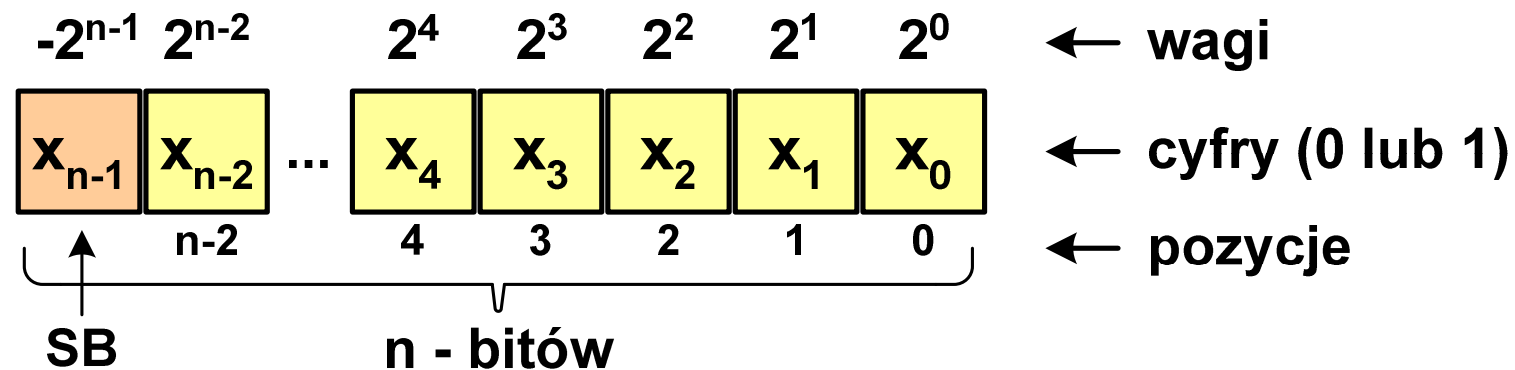
- negujemy wszystkie bity

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

↑
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:

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

- 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	U2	dziesiętnie
0000	0	1111	-1
0001	1	1110	-2
0010	2	1101	-3
0011	3	1100	-4
0100	4	1011	-5
0101	5	1010	-6
0110	6	1001	-7
0111	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**:

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

dla 8 bitów : $\langle -128 \dots 127 \rangle$

dla 16 bitów : $\langle -32768 \dots 32767 \rangle$

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

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

- negujemy wszystkie bity i dodajemy 1

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

Liczby całkowite ze znakiem - kod U2 w języku C

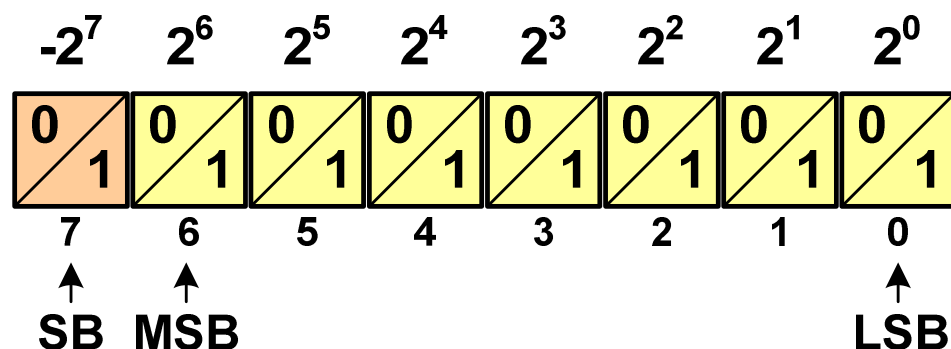
- Typy zmiennych całkowitych ze znakiem stosowane w języku C:

<u>Nazwa typu</u>	<u>Rozmiar (bajty)</u>	<u>Zakres wartości</u>
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):

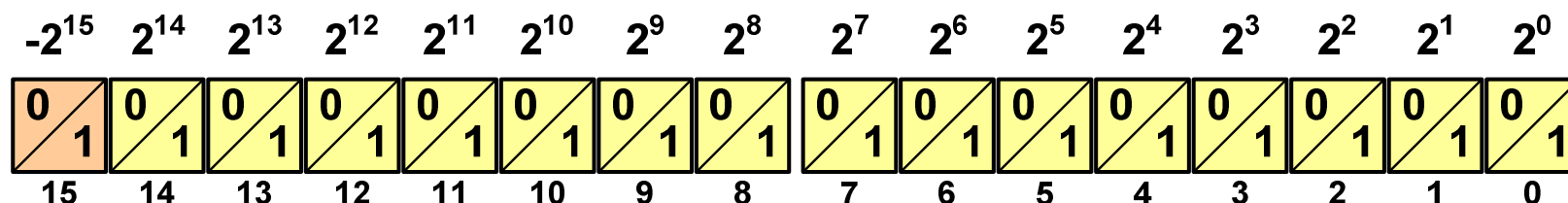


- Zakres wartości:

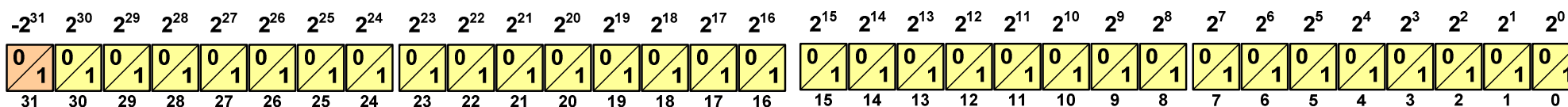
- dolna granica: $1000\ 0000_{(2)} = -128_{(10)}$
- górna granica: $0111\ 1111_{(2)} = 127_{(10)}$
- inne wartości: $1111\ 1111_{(2)} = -1_{(10)}$
 $0000\ 0000_{(2)} = 0_{(10)}$

Liczby całkowite bez znaku w języku C

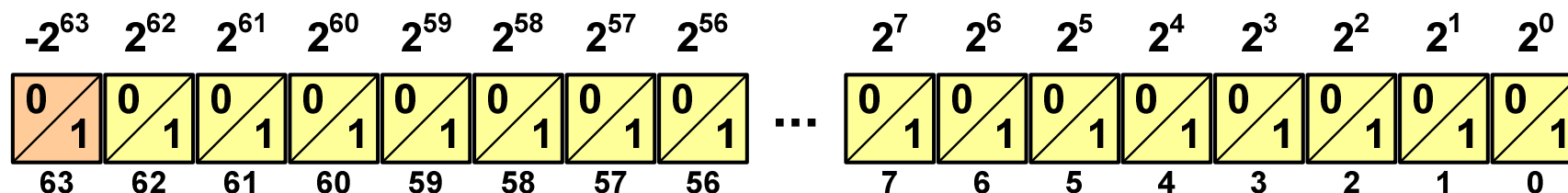
- Typ **short / signed short int** (2 bajty):



- 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 <stdio.h>

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

B - podstawa systemu

M - mantysa

E - wykładnik, cecha

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

Zapis zmiennoprzecinkowy liczby rzeczywistej

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

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

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

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

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

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

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

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

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

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

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

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

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

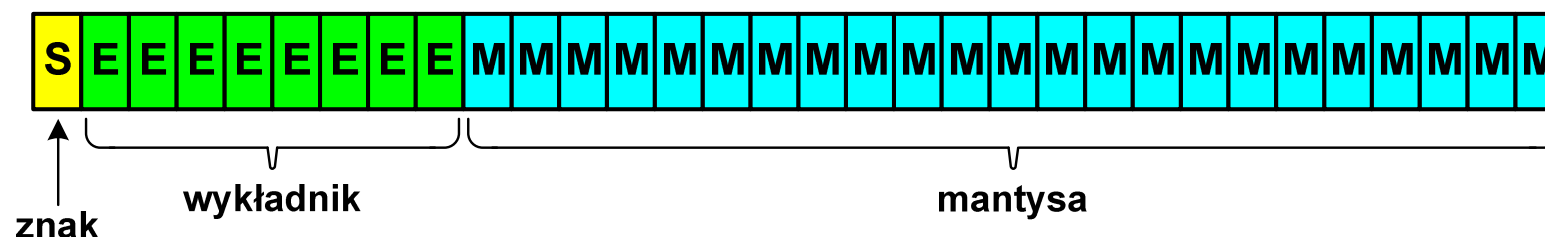
$$B > |M| \geq 1$$

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**)

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

gdzie:

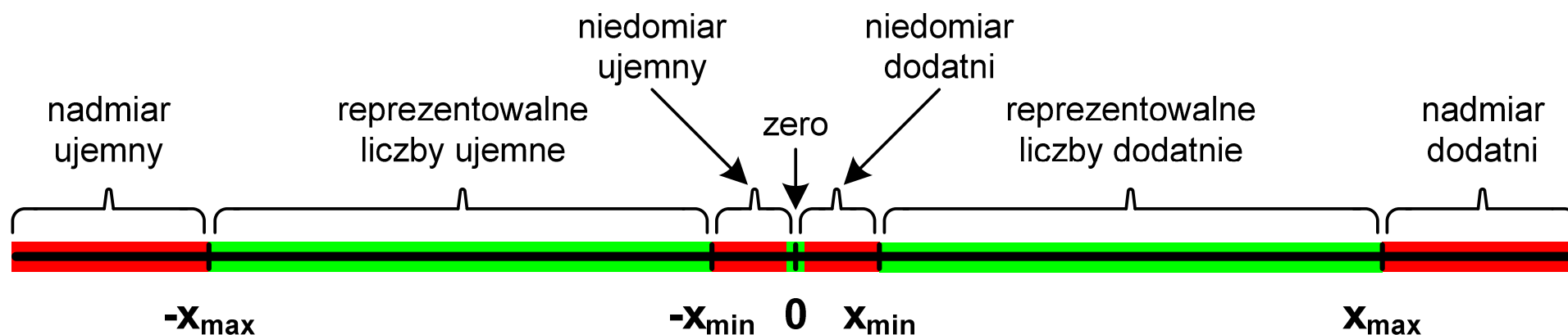
L - wartość liczby **S** - znak liczby **M** - mantysa
E - wykładnik **BIAS** - przesunięcie (nadmiar)

- 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:

$$X_{\min} = M_{\min} \cdot B^{E_{\min}}$$

$$X_{\max} = M_{\max} \cdot B^{E_{\max}}$$

Koniec wykładu nr 3

Dziękuję za uwagę!