Podstawy informatyki

Izabela Szczęch

Politechnika Poznańska

KOMPUTEROWA REPREZENTACJA ZNAKÓW I LICZB

Plan wykładu

- Reprezentacja informacji w systemie komputerowym
- Podstawowe jednostki informacji
- Komputerowa reprezentacja znaków
  - kod ASCII i różne strony kodowe
  - Unicode
- Komputerowa reprezentacja liczb
  - Binarna reprezentacja liczb naturalnych
  - Podstawowe działania na binarnych liczbach naturalnych
  - Stałoprzecinkowa reprezentacja liczb rzeczywistych
  - Zmiennoprzecinkowa reprezentacja liczb rzeczywistych

Idea działania układów cyfrowych

- W układach cyfrowych wszelka informacja i wszelkie wielkości przetwarzane przez te układy reprezentowane są przez dwa stany:
  - stan niski (L), nazywany też poziomem logicznym niskim lub zerem (0)
  - stan wysoki (H), nazywany też poziomem logicznym wysokim lub jedynką (1)
- W praktycznej realizacji układów cyfrowych należy określić jakie wartości lub zakresy wartości oznaczają poziom logiczny niski lub wysoki

Jednostki informacji

- bit – podstawowa, najmniejsza i niepodzielna jednostka informacji cyfrowej, jaka może być przetwarzana przez komputer
- bit może przechowywać informację o jednym z dwóch możliwych stanów – przyjmuje wartości oznaczane jako 0 albo 1
- bit to skrót terminu BInary DigiT
- bit to po angielsku kawałek
- oznaczenie: b

Jednostki informacji

- 1 bit: 0, 1, rozróżnia 2 znaki
- 2 bity: 00, 01, 10, 11, rozróżniają 4 znaki
- 3 bity: 000, 001, 010, 011, 100, 101, 110, 111, rozróżniają 8 znaków
- 4 bity: 0000 ... 1111, rozróżniają 16 znaków
- 8 bitów (=bajt B) pozwala odróżnić $2^8 = 16 \times 16 = 256$ znaków

- Słowo komputerowe - ilość informacji przetwarzanej przez komputer
- Komputer 8–, 16–, 32– (64–, 128-) bitowy oznacza wielkość grupy danych, którą komputer może operować jako całością

Jednostki informacji

- W informatyce nazwy przedrostków nie odpowiadają tym z układu SI
  - kilo = 1 000 = 10^3
  - mega = 1 000 000 = 10^6 = kilo x 1000
  - giga = 1 000 000 000 = 10^9 = mega x 1000
  - tera = 1 000 000 000 000 = 10^{12} = giga x 1000

  - Kilo = 1 024 = 2^{10}
  - Mega = 1 048 576 = 2^{20} = Kilo x 1024
  - Giga = 1 073 741 824 = 2^{30} = Mega x1024
  - Tera = 1 099 511 627 776 = 2^{40} = Giga x 1024

Jednostki informacji

- 2^{10}=1024=1K kilobajt typowa strona tekstu to kilka KB;
- 2^{20}=1024K=1M megabajt książka bez grafiki lub minuta muzyki;
- 2^{30}=1024M=1G gigabajt film cyfrowy, sporo grafiki, ludzki genom;
- 2^{40}=1024G=1T terabajt duża biblioteka, szerokoekranowy film w kinie;
- 2^{50}=1024T=1P petabajt ludzka pamięć;

- Rozróżnienie B i b:
  B = bajty, KB = kilobajty, MB = megabajty, GB = gigabajty
  b = bity, Kb = kilobity, Mb = megabity ....

Dlaczego komputery przetwarzają bity?

- Bity są odporne na zakłócenia
    - w czasie przekazu należy wykryć tylko dwa poziomy, wysoki H - 1 i niski L – 0
- Brak wartości pośrednich
    - w przypadku innych jednostek sygnał musi mieć więcej poziomów, a więc jest bardziej podatny na przekłamania w trakcie transmisji
- Binarny system pozycyjny da się bezpośrednio zakodować w postaci bitów
    - każdej cyfrze binarnej odpowiada jeden bit (na bitach można wykonywać dowolne operacje arytmetyczne, a więc liczyć)

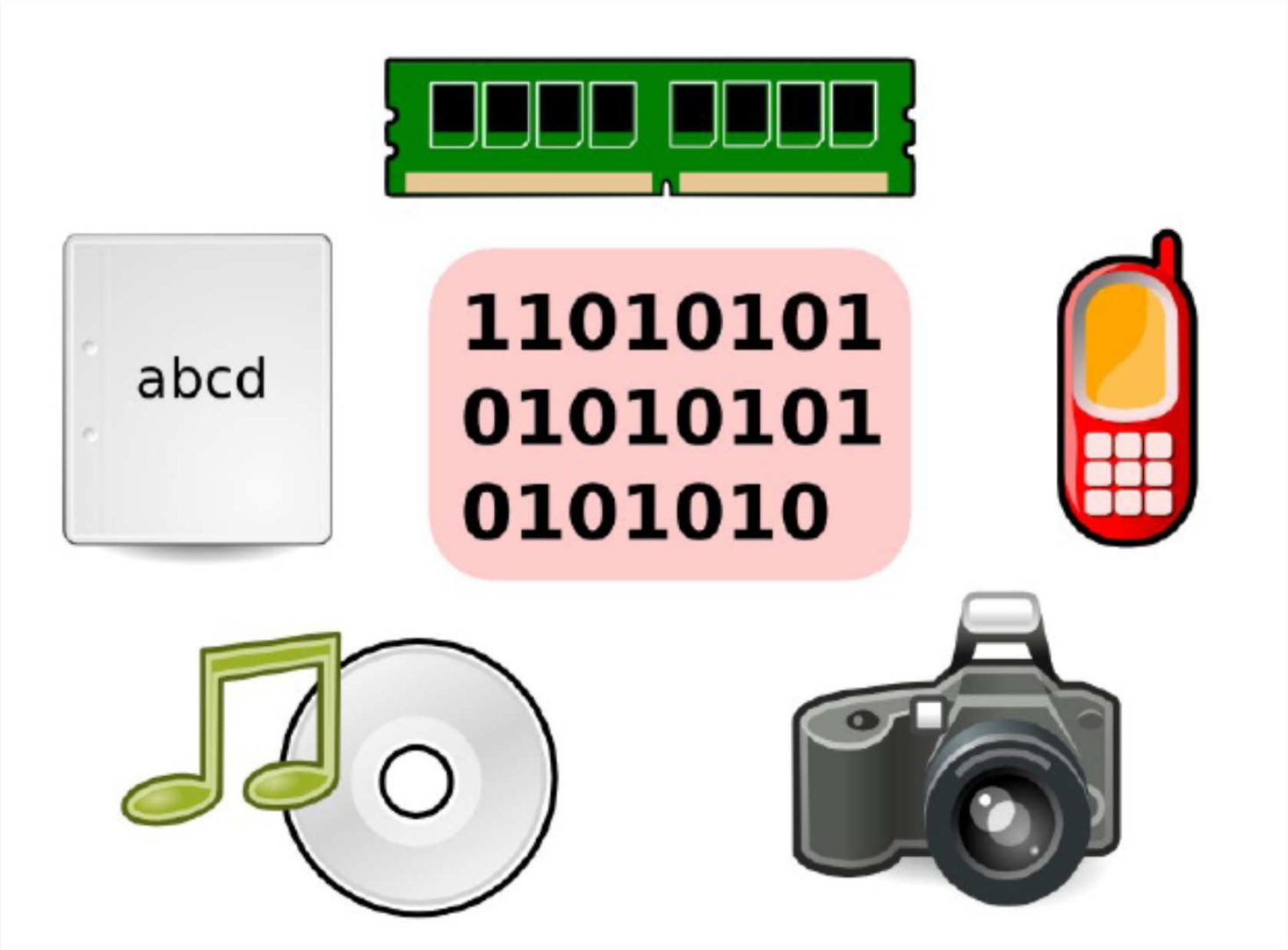
Reprezentacja informacji w komputerze

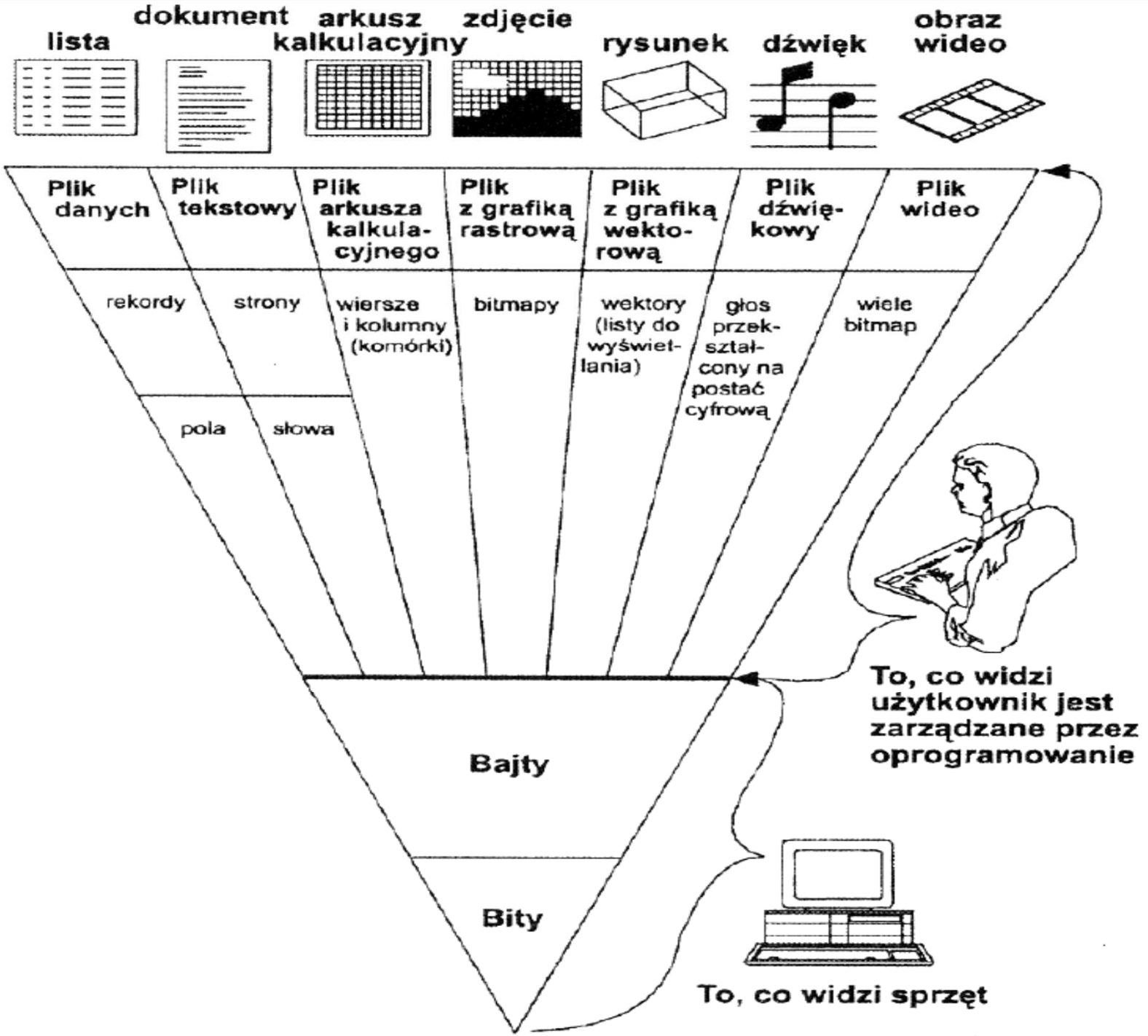
- Jak to się dzieje że w pamięci komputera można przechowywać teksty, obrazy, dźwięki i liczby?

Dzięki kodowaniu informacji

- Kodowanie informacji jest to przedstawienie informacji w postaci komunikatu zrozumiałego przez odbiorcę. Do kodowania używamy określonego zbioru, np. cyfr, znaków, impulsów

Reprezentacja informacji w komputerze




lista
dokument
arkusz kalkulacyjny
zdjęcie
rysunek
dźwięk
obraz wideo
Plik danych
Plik tekstowy
Plik arkusza kalkula-cyjnego
Plik z grafiką rastrową
Plik z grafiką wekto-rową
Plik dźwię-kowy
Plik wideo
rekordy
strony
wiersze i kolumny (komórki)
bitmapy
wektory (listy do wyświet-lania)
głos prze-kształ-cony na postać cyfrową
wiele bitmap
pola
słowa
To, co widzi użytkownik jest zarządzane przez oprogramowanie
Bajty
Bity
To, co widzi sprzęt

KOMPUTEROWA REPREZENTACJA ZNAKÓW

Kodowanie znaków

- Każdemu elementowi alfabetu przyporządkowuje ciąg binarny
  np. a -> 0001, b -> 0010, itd.
- Inaczej K(a)=0001, K(b)=0010, gdzie K to kod.
- Słowo kodowe:
  Jeśli K(a)=0001, to 0001 jest słowem kodowym a.

Kod ASCII

- ASCII (American Standard Code for Information Interchange)
- Opracowany dla urządzeń dalekopisowych, później przyjęty dla komputerów
- 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom (alfabetu angielskiego), cyfrom, znakom przestankowym i innym symbolom oraz poleceniom sterującym
- litery, cyfry oraz inne znaki drukowane tworzą zbiór znaków ASCII - jest to 95 znaków o kodach 32-126
- pozostałe 33 kody (0-31 i 127) to tzw. kody sterujące służące do sterowania urządzeniem odbierającym komunikat, np. drukarką czy terminalem

Kod ASCII - przykłady

Bin	Dec	Hex	Znak
010 0000	32	20	Spacja
010 0001	33	21	!
010 0010	34	22	"
010 0011	35	23	#
100 0000	64	40	@
100 0001	65	41	A
100 0010	66	42	B
100 0011	67	43	C
100 0100	68	44	D
110 0001	97	61	a
110 0010	98	62	b
110 0011	99	63	c
110 0100	100	64	d

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Kod ASCII – kody sterujące

- 33 kody: 0-31 i 127
- znaki niedrukowalne: sygnały dźwiękowe, białe znaki (tabulacja, spacja, znak końca wiersza, powrót na początek wiersza)

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	

Kod ASCII – informacja tekstowa

- W systemie DOS/Windows każdy wiersz pliku zakończony jest parą znaków:
  - CR, kod ASCII - $13_{(10)} = 0D_{(16)}$ - powrót na początek wiersza
  - LF, kod ASCII - $10_{(10)} = 0A_{(16)}$ - przesunięcie o wiersz
- w systemie Linux znakiem końca wiersza jest tylko:
  - LF, kod ASCII - 10(10) = 0A(16)
- Podczas przesyłania pliku tekstowego z systemu Linux do systemu DOS/Windows pojedynczy znak LF zamieniany jest automatycznie na parę znaków CR i LF
- Błędne przesłanie pliku tekstowego (w trybie binarnym) powoduje nieprawidłowe jego wyświetlanie

Kod ASCII – informacja tekstowa

Pierwszy wiersz pliku

Drugi wiersz pliku

Trzeci wiersz pliku

- Windows

```
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■■
```

- Linux

```
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 0A       | wiersz pliku■■
```

- Nieprawidłowy wygląd przy błędnej transmisji

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

Kod ASCII – strony kodowe

- Kod ASCII w wersji podstawowej jest 7-bitowy
- Większość komputerów pracuje na 8-bitowych bajtach
- Dodatkowy bit jest wykorzystywany do powiększenia zbioru kodowanych znaków
- Powstało wiele różnych rozszerzeń ASCII wykorzystujących ósmy bit, nazywanych stronami kodowymi (np. norma ISO 8859, rozszerzenia firm IBM lub Microsoft)

Kod ASCII – strony kodowe

- Strony kodowe – wersje kodu ASCII różniące się interpretacją symboli o numerach od 128 do 255
- Różne strony kodowe mogą przyjąć odmienne znaki dla tego samego kodu
- Różne strony kodowe mogą różnić się wyborem znaków
- Duża liczba dostępnych stron kodowych wynika z faktu, że na 8 bitach można zakodować tylko 256 różnych znaków, co jest niewystarczające do zmieszczenia w jednym zestawie znaków wszystkich alfabetów
- Przykłady stron kodowych uwzględniające polskie litery: ISO 8859-2 (Latin-2), ISO 8859-16 (Latin-10)

ISO 8859

- Zestaw standardów służących do kodowania znaków za pomocą ośmiu bitów
- Standardy te zostały utworzone przez European Computer Manufactures' Association (ECMA) w połowie lat osiemdziesiątych, a następnie uznane przez ISO
- Wszystkie zestawy ISO 8859 mają znaki 0-127 takie same jak ASCII
- Pozycjom 128-159 przypisane są dodatkowe kody sterujące, tzw. C1 (faktycznie są nieużywane).

Wybrane standardy ISO 8859

- ISO 8859-1 (Latin-1) - alfabet łaciński dla Europy zachodniej
- ISO 8859-2 (Latin-2) – alfabet łaciński dla Europy środkowej i wschodniej, również odpowiednia Polska Norma
- ISO 8859-3 (Latin-3) – alfabet łaciński dla Europy południowej
- ISO 8859-4 (Latin-4) – alfabet łaciński dla Europy północnej
- ISO 8859-5 (Cyrillic) – alfabet dla cyrylicy
- ISO 8859-6 (Arabic) - dla alfabetu arabskiego
- ISO 8859-7 (Greek) - dla alfabetu greckiego
- ISO 8859-8 (Hebrew) - dla alfabetu hebrajskiego
- ISO 8859-11 (Thai) - dla alfabetu tajskiego
- ISO 8859-13 (Latin-7)
- ISO 8859-15 (Latin-9) - z ISO 8859-1 usunięto kilka rzadko używanych znaków i wprowadzono znak Euro oraz litery Š, š, Ž, ž, Œ, œ oraz Ÿ
- ISO 8859-16 (Latin-10) – alfabet łaciński dla Europy środkowej - zmodyfikowany ISO 8859-2 ze znakiem Euro i dodatkowymi literami dla kilku języków

Kod ISO 8859-1

- 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
- składa się ze 191 znaków łacińskiego pisma
- po rozszerzeniu o dodatkowe przypisania znaków jest podstawą dla dwóch mapowań znaków ISO-8859-1 i Windows-1252

Kod ISO 8859-2

- dostępne języki: bośniacki, chorwacki, czeski, węgierski, polski, rumuński, serbski, serbsko-chorwacki, słowacki, słoweński, górno- i dolnołużycki
- możliwość przedstawienia znaków w języku niemieckim i angielskim
- składa się ze 191 znaków łacińskiego pisma
- kody z przedziałów $00_{(16)}$-$1F_{(16)}$ oraz $7F_{(16)}$-$9F_{(16)}$ nie są używane w ISO-8859-2
- kodowanie to jest zgodne z Polską Normą

ISO 8859-1 i 8859-2 – porównanie

	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	¡	¢	£	¤	¥	¦	§	¨	©	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)

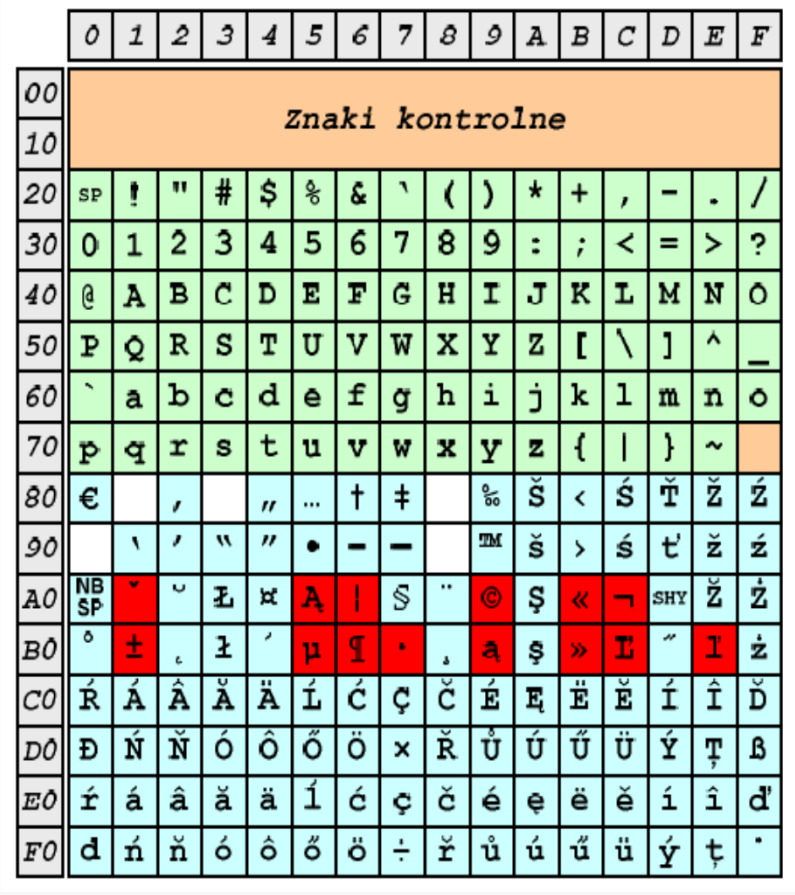
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80	Nieużywane															
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)

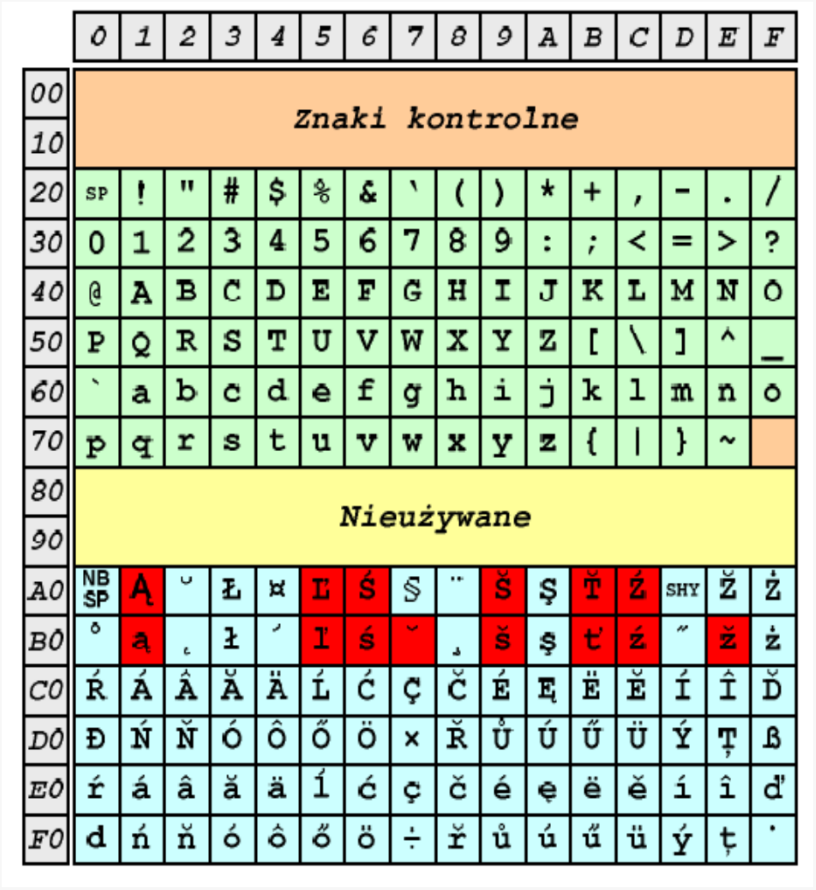
Windows 1250

- Standard kodowania używany przez system Microsoft Windows do reprezentacji tekstów w językach środkowoeuropejskich używających alfabetu łacińskiego, takich jak albański, chorwacki, czeski, polski, rumuński, słowacki, węgierski
- jest podobny do ISO 8859-2 — posiada wszystkie jego drukowalne znaki (a także kilka dodatkowych), lecz kilka z nich zajmuje inne miejsca

Windows 1250 i ISO 8859-2 - porównanie



	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	€		‚		„	…	†	‡		‰	Š	‹	Ś	Ť	Ž	Ź
90		‘	’	“	”	•	–	—		™	š	›	ś	ť	ž	ź
A0	NB SP	ˇ	˘	Ł	¤	Ą	|	§	¨	©	Ş	«	¬	SHY	Ž	Ż
B0	°	±	˛	ł	´	µ	¶	·	¸	ą	ş	»	Ľ	˝	ľ	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	d	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙



	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	d	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Windows 1250 i ISO 8859-2 - kodowanie polskich znaków

- 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 edytorze systemu Windows (Windows 1250)

ˇ Ć Ę Ł Ń Ó ¦ ¬ Ż
± ć ę ł ń ó ¶ Ľ ż

Standardy kodowania polskich znaków

- W Polsce stosowanych było ok. 20 „standardów” kodowania polskich liter (http://www.ogonki.agh.edu.pl)
- Próby wprowadzania standardu:
  - Mazovia - promowany przez społeczność informatyczną w Polsce (nie był pełną stroną kodową, ale określał sposób kodowania polskich liter)
  - IBM-Latin-2 (CP-852) - stosowany w DOS, OS/2 i części systemu Windows
  - CP-1250 (Windows-1250) – stosowany w MS Windows PL
  - ISO-Latin-2 (ISO 8859-2) - stosowany w Internecie
    Standard ISO 8859-2 jest zgodny z Polską Normą PN-93 T-42118

	(A) Ą	(C) Ć	(E) Ę	(L) Ł	(N) Ń	(O) Ó	(S) Ś	(X) Ź	(Z) Ż	(a) ą	(c) ć	(e) ę	(l) ł	(n) ń	(o) ó	(s) ś	(x) ź	(z) ż
ISO-8859-2	161	198	202	163	209	211	166	172	175	177	230	234	179	241	243	182	188	191
Windows-EE	165	198	202	163	209	211	140	143	175	185	230	234	179	241	243	156	159	191
IBM (CP852)	164	143	168	157	227	224	151	141	189	165	134	169	136	228	162	152	171	190
Mazovia	143	149	144	156	165	163	152	160	161	134	141	145	146	164	162	158	166	167

Unicode

- Komputerowy zestaw znaków mający w zamierzeniu obejmować wszystkie pisma i inne znaki (symbole muzyczne, techniczne, wymowy) używane na świecie
- Unicode przypisuje unikalny numer każdemu znakowi, niezależnie od używanej platformy, programu czy języka
- Standard Unicode zaimplementowany został w wielu technologiach, np. XML, Java, Microsoft .NET Framework, systemy operacyjne

Unicode

- Rozwijany jest przez konsorcjum, w skład którego wchodzą firmy komputerowe, producenci oprogramowania oraz grupy użytkowników
- http://www.unicode.org
- Strona polska: http://www.unikod.pl
- Pierwsza wersja: Unicode 1.0 (październik 1991)
- Ostatnia wersja: Unicode 9.0 (czerwiec 2017)

http://www.unicode.org

- Version 10.0 adds exactly 8,518 characters, for a total of 136,690 characters. These additions include four new scripts, for a total of 139 scripts, as well as 56 new emoji characters.
- The new scripts and characters in Version 10.0 add support for lesser-used languages and unique written requirements worldwide, including:
  - Masaram Gondi, used to write Gondi in Central and Southeast India
  - Nüshu,used by women in China to write poetry and other discourses until the late twentieth century
  - Soyombo and Zanabazar Square, used in historic Buddhist texts to write Sanskrit, Tibetan, and Mongolian
  - …
- Important symbol additions include:
  - Bitcoin sign
  - 56 emoji characters

http://www.unicode.org

- Version 9.0 adds exactly 7,500 characters, for a total of 128,172 characters. These additions include six new scripts and 72 new emoji characters.
- The new scripts and characters in Version 9.0 add support for lesser-used languages worldwide, including:
  - Osage, a Native American language
  - Nepal Bhasa, a language of Nepal
  - Fulani and other African languages
  - The Bravanese dialect of Swahili, used in Somalia
  - The Warsh orthography for Arabic, used in North and West Africa
  - Tangut, a major historic script of China
- Important symbol additions include:
  - 19 symbols for the new 4K TV standard
  - 72 emoji characters such as the following

Podstawowe właściwości Unicode

- Jednoznaczność – każdemu znakowi zakodowanemu za pomocą Unicode odpowiada zawsze ta sama wartość liczbowa i odwrotnie
- Uniwersalność – dostępne znaki obejmują wszystkie powszechnie używane języki i symbole
- 16-bitowa reprezentacja – każdy znak reprezentowany jest w postaci 16-bitowej liczby, czyli można zakodować 216=65 536 różnych znaków
- Efektywność – ułatwienie manipulowania tekstami, gdyż identyfikacja znaku nie zależy od sekwencji sterujących czy znaków następujących bądź poprzedzających

Sposoby kodowania Unicode

- Standard Unicode definiuje kody numeryczne przypisane poszczególnym znakom, nie określa natomiast sposobu bajtowego kodowania znaków
- Kodowanie określa sposób w jaki znaki ze zbioru mają być zapisane w postaci binarnej
- Przykłady metod kodowania:
  - UTF-8 - kodowanie 8-bitowe ze zmienną długością kodowania, zgodne z ASCII
  - UTF-16 - kodowanie 16-bitowe ze zmienną długością kodowania
  - UTF-32/UCS-4 - kodowanie 32-bitowe

Sposoby kodowania Unicode

- UTF-8 – Unicode ośmiobitowy. Pierwsze 128 znaków pokrywa się z tablicą ASCII i jest zapisywana za pomocą jednego bajta, natomiast znaki dodatkowe (np. polskie literki) są zapisywane za pomocą specjalnych kilkubajtowych sekwencji

- UTF-16 - kodowanie dwubajtowe. Każdemu znakowi odpowiadają dwa bajty

- UTF-32 – kodowanie czterobajtowe rozwiązujące problem języków ideograficznych (np. chiński liczący tysiące znaków). Cztery bajty dają 232 = 4 miliardy kombinacji

KOMPUTEROWA REPREZENTACJA LICZB NATURALNYCH

System liczbowy

- System liczbowy to zbiór reguł do jednolitego zapisywania i nazywania liczb
- Do zapisywania liczb używa się pewnego skończonego zbioru znaków, zwanych cyframi (np. arabskimi, rzymskimi), które można zestawiać ze sobą na różne sposoby otrzymując nieskończoną liczbę kombinacji

System liczbowy

- Najbardziej prymitywnym systemem liczbowym jest jedynkowy system liczbowy, w którym występuje tylko jeden znak (np. 1 albo pionowa kreska). W systemie tym kolejne liczby są tworzone przez proste powtarzanie tego znaku.
- Przykładowo, 3 w tym systemie jest zapisywana jako 111, a pięć jako 11111

- Ogólnie, wśród systemów liczbowych można wyróżnić:
  - systemy addytywne (niepozycyjne)
  - systemy pozycyjne

- Zadanie domowe: określić czy system jedynkowy jest systemem addytywnym czy pozycyjnym

Systemy addytywne

- Systemy addytywne (zwane też niepozycyjnymi)
  to systemy liczbowe, w których wartość przedstawionej liczby jest sumą wartości jej znaków (cyfr)

- Przykład - rzymski system liczbowy

  Jeśli "X"=10,"V"=5,"I"=1

  to XVI = 10+5+1 = 16

Aby wyznaczyć wartość dziesiętną liczby zapisanej w systemie rzymskim należy odejmować wartości znaków stojących przed znakami większymi, a dodawać wartości znaków stojących za znakami większymi lub równymi co do wartości.

Systemy pozycyjne

- Pozycyjnym systemem liczbowym nazywamy parę (P; C),
  gdzie P jest liczbą naturalną zwaną podstawą systemu,
  a C jest skończonym zbiorem znaków {0, 1, …, P-1} zwanych cyframi
- W systemie pozycyjnym liczbę przedstawia się jako ciąg cyfr, przy czym wartość tej liczby zależy zarówno od cyfr jak i miejsca, na którym się one znajdują w tym ciągu
- Jeśli P=10, to otrzymujemy dziesiętny system liczbowy,
  w którym występują cyfry ze zbioru {0,…,9},
  dla P = 2 dwójkowy (binarny) z cyframi ze zbioru {0, 1},
  dla P=8 ósemkowy (oktalny) z cyframi ze zbioru {0,…,7}, itd.

Systemy pozycyjne

- Liczba całkowita *L* zapisana w systemie pozycyjnym o podstawie P w postaci ciągu cyfr

$$c_{n-1}\ldots c_1 c_0 \ (P)$$

ma wartość liczbową

$$w = c_{n-1}P^{n-1} + c_{n-2}P^{n-2} + \ldots + c_1 P^1 + c_0$$

$$w = \sum_{i=0}^{n-1} c_i P^i$$

gdzie $c_{n-1},\ldots,c_1,c_0 \in C$, oraz P jest podstawą systemu.

- Przykład

Liczba 5004 w dziesiętnym systemie liczbowym:

$5 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 4 \times 10^0 = 5 \times 1000 + 4 \times 1$

System dziesiętny (decymalny, arabski)

- Podstawę systemu dziesiętnego stanowi liczba 10
- Zapis liczby tworzymy za pomocą cyfr o przypisanych wartościach od 0 do 9

- Przykład

$126_{(10)} = 1x10^2 + 2x10^1 + 6x10^0$

System dwójkowy (binarny)

- Podstawę systemu binarnego stanowi liczba 2
- Zapis liczby tworzymy za pomocą cyfr o przypisanych wartościach 0 i 1

- Przykład

$$1010_{(2)} = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 8 + 2 = 10_{(10)}$$

Zadania

- Z definicji wyznacz dziesiętną wartość poniższych liczb:
  - $421_{(7)} = ???_{(10)}$
  - $2102_{(3)} = ???_{(10)}$
  - $AF1_{(16)} = ???_{(10)}$

Konwersja z systemu dziesiętnego

- Sformułowanie problemu:
  znalezienie kolejnych cyfr zapisu liczby dziesiętnej L
  w dowolnym systemie docelowym o podstawie P
- Zgodnie z definicją wartość liczby dziesiętnej L wynosi:

$$L = c_{n-1} P^{n-1} + c_{n-2} P^{n-2} + \ldots + c_2 P^2 + c_1 P + c_0$$

- Przykładowo, konwersja liczby $L = 6_{(10)}$ z systemu dziesiętnego na dwójkowy polega na znalezieniu kolejnych cyfr c_i czyli cyfr $110_{(2)}$
  $6_{(10)} = \mathbf{1}*2^2 + \mathbf{1}*2^1 + \mathbf{0}*2^0 = \mathbf{110}_{(2)}$

Konwersja z systemu dziesiętnego

- Do wydobycia poszczególnych cyfr c_i, i = 0, 1, 2, ..., n-1,
  są nam potrzebne dwa działania:
  - div - dzielenie całkowitoliczbowe,
    określa ile całkowitą ilość razy dzielnik mieści się w dzielnej,
    np.: 9 div 4 = 2, gdyż 4 mieści się w 9 dwa razy
  - mod - reszta z dzielenia całkowitoliczbowego,
    np: 9 mod 4 = 1, gdyż 4 mieści się w 9 dwa razy,
    co daje 8 i pozostawia resztę 1

Konwersja z systemu dziesiętnego

Algorytm konwersji liczby z systemu dziesiętnego
na system o podstawie P:

- *w* - wartość liczby w systemie dziesiętnym
- P - podstawa docelowego systemu pozycyjnego
- c_i - cyfra na i-tej pozycji w systemie pozyc. o podstawie P

1. i ← 0
2. c_i ← *w* mod P
3. *w* ← *w* div P
4. jeśli *w* = 0, to kończymy, wszystkie cyfry znalezione
5. i ← i + 1
6. wróć do punktu 2.

Konwersja z systemu dziesiętnego

Przykład:

Przeliczyć wartość $175_{(10)}$ na system binarny.

$w \leftarrow 175$ div $2 = 87$ i reszta 1 (zatem $c_0=1$)

$w \leftarrow 87$ div $2 = 43$ i reszta 1 (zatem $c_1=1$)

$w \leftarrow 43$ div $2 = 21$ i reszta 1 (zatem $c_2=1$)

$w \leftarrow 21$ div $2 = 10$ i reszta 1

$w \leftarrow 10$ div $2 = 5$ i reszta 0

$w \leftarrow 5$ div $2 = 2$ i reszta 1

$w \leftarrow 2$ div $2 = 1$ i reszta 0

$w \leftarrow 1$ div $2 = 0$ i reszta 1 (koniec obliczeń, ponieważ otrzymaliśmy wartość 0 jako wynik operacji div)

$175_{(10)} = 10101111_{(2)}$

Konwersja z systemu dziesiętnego

Przykład:

Przeliczyć wartość $12786_{(10)}$ na system piątkowy.

$w \leftarrow 12786$ div $5 = 2557$ i reszta 1 (zatem $c_0=1$)

$w \leftarrow 2557$ div $5 = 511$ i reszta 2 (zatem $c_1=2$)

$w \leftarrow 511$ div $5 = 102$ i reszta 1 (zatem $c_2=1$)

$w \leftarrow 102$ div $5 = 20$ i reszta 2

$w \leftarrow 20$ div $5 = 4$ i reszta 0

$w \leftarrow 4$ div $5 = 0$ i reszta 4

$12786_{(10)} = 402121_{(5)}$

Konwersja z systemu dziesiętnego

Zadania:

1. Przedstawić w systemie dwójkowym liczbę $32_{(10)}$.
2. Przedstawić w systemie dwójkowym liczbę $240_{(10)}$.
3. Przedstawić w systemie czwórkowym liczbę $2743_{(10)}$.

DODAWANIE LICZB BINARNYCH

Dodawanie liczb binarnych

- Do wykonywania dodawania potrzebna jest znajomość wyników sumowania wszystkich kombinacji cyfr:
  - $0_{(2)} + 0_{(2)} = 0_{(2)}$
  - $0_{(2)} + 1_{(2)} = 1_{(2)}$
  - $1_{(2)} + 0_{(2)} = 1_{(2)}$
  - $1_{(2)} + 1_{(2)} = 10_{(2)}$

- Wyjaśnienie:

1+1 w systemie dwójkowym daje w wyniku 0 na pewnej pozycji, a jedność jest przenoszona na następną pozycję w liczbie.
Jest to podoba sytuacja jak w przypadku dodawania 1 + 9 w systemie dziesiętnym - otrzymujemy w wyniku 0, a jedność jest przenoszona na następną pozycję.

Dodawanie liczb binarnych

- $0101 = 5_{(10)}$
  $+\ 0110 = 6_{(10)}$
  $1011 = 11_{(10)}$
- $1011 = 11_{(10)}$
  $+\ 0011 = 3_{(10)}$
  $1110 = 14_{(10)}$
- $1010 = 10_{(10)}$
  $+\ 1010 = 10_{(10)}$
  $10100 = 20_{(10)}$
- $1111 = 15_{(10)}$
  $+\ 0001 = 1_{(10)}$
  $10000 = 16_{(10)}$

$0_{(2)} + 0_{(2)} = 0_{(2)}$

$0_{(2)} + 1_{(2)} = 1_{(2)}$

$1_{(2)} + 0_{(2)} = 1_{(2)}$

$1_{(2)} + 1_{(2)} = 10_{(2)}$

Dodawanie liczb binarnych

Zadania:

Wykonaj poniższe dodawania:

1. $1111001_{(2)} + 10010_{(2)} = ???_{(2)}$
2. $01111111_{(2)} + 1_{(2)} = ???_{(2)}$

Dla sprawdzenia poprawności obliczeń przekonwertuj liczby binarne na zapis w systemie dziesiętnym.

Dodawanie liczb binarnych - nadmiar

- W pamięci komputera liczby binarne przechowywane są w postaci ustalonej liczby bitów (np. 8, 16, 32 bity).
- Jeśli, zakładając np. 8-mio bitowy format, wynik sumowania dwóch liczb 8 bitowych jest większy niż 8 bitów, to najstarszy bit (dziewiąty) zostanie utracony.
- Sytuacja taka nazywa się **nadmiarem** (ang. overflow) i występuje zawsze, gdy wynik operacji arytmetycznej jest większy niż górny zakres danego formatu liczb binarnych (np. dla 8 bitów wynik większy od 2^8 - 1, czyli większy od 255):

$11111111_{(2)} + 00000001_{(2)} = 1|00000000_{(2)}$
$(255_{(10)}+1_{(10)}=0_{(10)})$

ODEJMOWANIE LICZB BINARNYCH

Odejmowanie liczb binarnych

- Do wykonywania odejmowania potrzebna jest znajomość wyników odejmowania wszystkich kombinacji cyfr:
  - $1_{(2)} - 0_{(2)} = 1_{(2)}$
  - $1_{(2)} - 1_{(2)} = 0_{(2)}$
  - $0_{(2)} - 0_{(2)} = 0_{(2)}$
  - $0_{(2)} - 1_{(2)} = 1_{(2)}$ i pożyczka z następnej pozycji

- Pożyczka oznacza konieczność odjęcia 1 od wyniku odejmowania cyfr w następnej kolumnie

Odejmowanie liczb binarnych

- **1**
  11001010 = $202_{(10)}$
  - 01110100 = $116_{(10)}$
  110
- **1 1**
  11001010 = $202_{(10)}$
  - 01110100 = $116_{(10)}$
  10110
- **11 1**
  11001010 = $202_{(10)}$
  - 01110100 = $116_{(10)}$
  010110
- **111 1**
  11001010 = $202_{(10)}$
  - 01110100 = $116_{(10)}$
  01010110 = $86_{(10)}$

$1_{(2)} - 0_{(2)} = 1_{(2)}$

$1_{(2)} - 1_{(2)} = 0_{(2)}$

$0_{(2)} - 0_{(2)} = 0_{(2)}$

$0_{(2)} - 1_{(2)} = 1_{(2)}$ i pożyczka z następnej pozycji

Odejmowanie liczb binarnych

Zadania:

Wykonaj poniższe odejmowania:

1. $10000000_{(2)}$ - $0000001_{(2)}$ = $???_{(2)}$
2. $10101010_{(2)}$ - $01010101_{(2)}$ = $???_{(2)}$

Dla sprawdzenia poprawności obliczeń przekonwertuj liczby binarne na zapis w systemie dziesiętnym.

Odejmowanie liczb binarnych - niedomiar

- Przy operacjach na liczbach naturalnych, jeśli od liczby mniejszej odejmiemy większą, to wynik będzie ujemny,
  a zatem niemożliwy do reprezentacji jako liczba naturalna.

```
 11111111
  00000000
-  00000001
  11111111
```

- Otrzymujemy same jedynki, a pożyczka nigdy nie zanika.
- Sytuacja taka nazywa się niedomiarem (ang. underflow)
  i występuje zawsze, gdy wynik operacji arytmetycznej jest mniejszy od dolnego zakresu formatu liczb binarnych
  (dla naturalnego kodu dwójkowego wynik mniejszy od 0).

STAŁOPRZECINKOWY ZAPIS LICZB RZECZYWISTYCH

Zapis stałoprzecinkowy

- Podejście stałoprzecinkowe – zakłada określenie ile bitów tworzy część całkowitą liczby, a ile część ułamkową. Przecinek jest zatem umieszczany w stałym dla danego formatu miejscu
- Dla liczb całkowitych w zapisie binarnym otrzymujemy zawsze dokładne wartości
- Kodowanie liczb niecałkowitych w ogólności obarczone jest błędem
- Rozróżniamy dwa zapisy stałoprzecinkowe:
  - bez znaku
  - ze znakiem

Dziesiętny zapis stałoprzecinkowy bez znaku

- Zapis pozycyjny można w prosty sposób rozszerzyć na liczby ułamkowe wprowadzając pozycje o wagach ułamkowych

Wagi pozycji	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Cyfry zapisu	3	5	7	9	,	8	2	9	1	4
			Część całkowita					Część ułamkowa		

Binarny zapis stałoprzecinkowy bez znaku

Wagi pozycji	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
Cyfry zapisu	1	0	1	0	,	1	1	0	0	1

Część całkowita

Część ułamkowa

$$1010{,}11001_{(2)} = 2^3 + 2^1 + 2^{-1} + 2^{-2} + 2^{-5}$$
$$= 8 + 2 + {}^1/_2 + {}^1/_4 + {}^1/_{32}$$
$$= 10\ {}^{25}/_{32}$$

Zapis stałoprzecinkowy bez znaku (bz)

- Liczba wymierna *L* zapisana w systemie pozycyjnym o podstawie P w postaci ciągu cyfr

$$c_{n-1}...c_1c_0\ c_{-1}...c_{-m}\ (P)$$

$$w = \sum_{i=-m}^{n-1} c_i P^i$$

ma wartość liczbową

$$w = c_{n-1}P^{n-1} + c_{n-2}P^{n-2} + \ldots + c_1P^1 + c_0 + c_{-1}P^{-1} + \ldots + c_{-m}P^{-m}$$

gdzie

c - cyfra dwójkowa 0 lub 1,

n - liczba bitów części całkowitej liczby

m - liczba bitów części ułamkowej liczby

- Przykład

$1010{,}11001_{(2)} = 2^3 + 2^1 + 2^{-1} + 2^{-2} + 2^{-5} = 8 + 2 + {}^1/_2 + {}^1/_4 + {}^1/_{16}$
$= 10\ {}^{13}/_{16}$

Konwersja zapisu stałoprzecinkowego bz na system dziesiętny

- Obliczamy z definicji wartość liczby w zapisie stałoprzecinkowym sumując iloczyny wag pozycji razy podstawa systemu.
- Należy pamiętać, że wagi części ułamkowej są ujemne.
- Obliczyć wartość dziesiętną liczby stałoprzecinkowej (bez znaku) $110100{,}111011_{(2)}$.

$110100{,}111011_{(2)}$

$= 2^5+2^4+2^2+2^{-1}+2^{-2}+2^{-3}+2^{-5}+2^{-6}$

$=32+16+4+ {}^1/_2+{}^1/_4+{}^1/_8 +{}^1/_{32}+{}^1/_{64}$

$=52 + {}^{32}/_{64} + {}^{16}/_{64} + {}^8/_{64} + {}^2/_{64} + {}^1/_{64}$

$= 52\ {}^{59}/_{64\ (10)}$

Konwersja zapisu stałoprzecinkowego bz na system dziesiętny

Zadania:

Oblicz wartość dziesiętną następujących liczb stałoprzecinkowych bez znaku:

1. $110{,}0011_{(2)} = ???_{(10)}$
2. $213{,}132_{(4)} = ???_{(10)}$

Konwersja z systemu dziesiętnego

- Załóżmy, że chcemy znaleźć zapis liczby dziesiętnej w systemie o podstawie P z dokładnością do *m* pozycji ułamkowych
- Liczbę rozdzielamy na część całkowitą oraz część ułamkową
- Część całkowitą przeliczamy na system o podstawie P
- Część ułamkową przeliczamy wg następujących kroków:
  - Dopóki nie otrzymamy w wyniku zera lub nie wyznaczymy zadanej ilości cyfr ułamkowych
    - część ułamkową mnożymy przez P
    - część całkowita wyniku mnożenia jest kolejną cyfrą zapisu stałoprzecinkowego w systemie o podstawie P
    - do następnego mnożenia bierzemy jedynie część ułamkową wyniku

Konwersja z systemu dziesiętnego

- Przykład: Przeliczyć na system dwójkowy bez znaku liczbę dziesiętną $2{,}21_{(10)}$ z dokładnością do 4 miejsc po przecinku
  - Wyznaczamy część całkowitą liczby $2_{(10)} = 10_{(2)}$
  - Wyznaczamy cyfry części ułamkowej $0{,}21_{(10)}$
    - 0,21 x 2 = 0,42 - cyfra 0
    - 0,42 x 2 = 0,84 - cyfra 0
    - 0,84 x 2 = 1,68 - cyfra 1
    - 0,68 x 2 = 1,36 - cyfra 1 - koniec, mamy 4 cyfry ułamkowe

Uwaga: pierwsza wyznaczona cyfra części ułamkowej to bit najbliższy przecinkowi

$2{,}21_{(10)} = 10{,}0011_{(2)}$

Po osiągnięciu 4 cyfr, część ułamkowa wciąż jest ≠ 0, tzn. otrzymane rozw. jest przybliżone (z dokładnością do 4 cyfr ułamkowych, czyli błąd < 2^{-4})

$10{,}0011_{(2)} = 2^1+2^{-3}+2^{-4} = 2+{}^1/_8+{}^1/_{16} = {}^{2^3}/_{16} = 2{,}1875_{(10)} \neq 2{,}21_{(10)}$

Konwersja z systemu dziesiętnego

Zadania:

Wyznaczyć stałoprzecinkową reprezentację bez znaku następujących liczb (z dokładnością do ośmiu miejsc po przecinku):

1. $0{,}1_{(10)} = ???_{(2)}$
2. $120{,}125_{(10)} = ???_{(8)}$

Zapis stałoprzecinkowy ze znakiem

- Do reprezentacji znaku w zapisie stałoprzecinkowym możemy wykorzystać DOWOLNĄ reprezentację liczby ze znakiem, przykładowo kod znak-moduł czy kod U2.
- Liczba stałoprzecinkowa *L* w przykładowym kodzie znak-moduł : $L = 11101010,11101001_{(ZM)}$

bit znaku	moduł liczby
1	1101010,11101001

Zapis stałoprzecinkowy ze znakiem

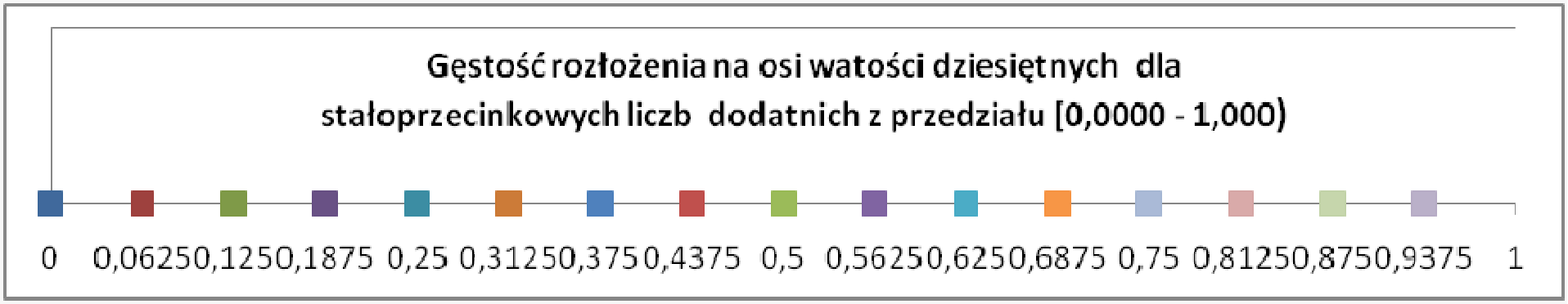
- Liczba stałoprzecinkowa *L* w przykładowym kodzie znak-moduł : $L = 11101010,11101001_{(ZM)}$

bit znaku	moduł liczby
1	1101010,11101001

- Najstarszy (najbardziej z lewej strony) bit to bit znaku. Jedynka na najstarszym bicie mówi nam, że liczba jest ujemna; zero, że liczba jest dodatnia.
- Wartość modułu $1101010,11101001_{(2)} = 106,9101563_{(10)}$
- Wartość liczby ze znakiem $11101010,11101001_{(2)} = -106,9101563_{(10)}$

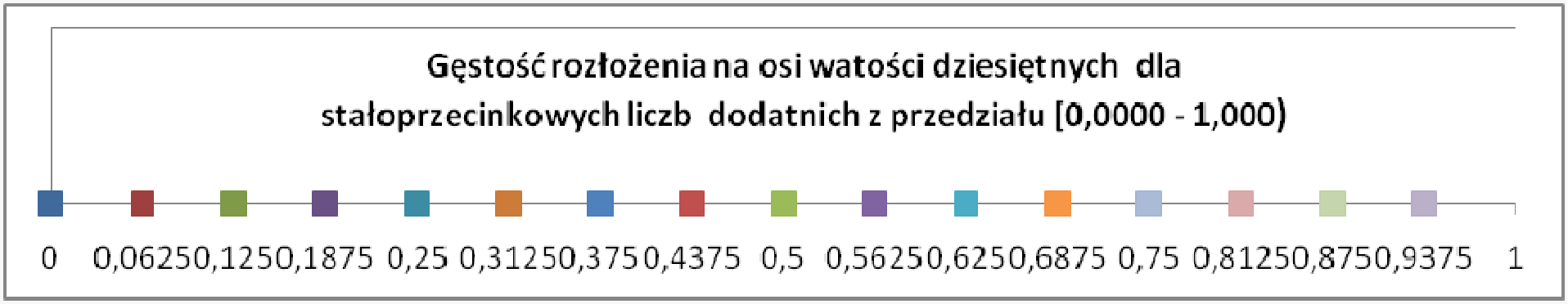
Dokładność reprezentacji stałoprzecinkowej

- Liczby wymierne dokładnie zapisywalne w komputerze to tzw. liczby maszynowe
- Pozostałe liczby wymierne są wyrażane z pewnym przybliżeniem poprzez wykorzystanie liczb maszynowych.
- Wartości liczb stałoprzecinkowych równomiernie rozkładają się na osi (przykład dla reprezentacji z 4 bitami ułamkowymi):



Dokładność reprezentacji stałoprzecinkowej

- Dla reprezentacji z 4 bitami części ułamkowej, wartości pojawiają się na osi w stałym odstępie równym 2^{-4}.
- Liczby nie obecne na osi, nie będę reprezentowane dokładnie, będą reprezentowane jako przybliżenie do najbliższej liczby maszynowej, a więc z błędem mniejszym niż 2^{-4}.



ZMIENNOPRZECINKOWY ZAPIS LICZB RZECZYWISTYCH

Wady zapisu stałoprzecinkowego

- Zapis bardzo dużych lub bardzo małych liczb w notacji stałoprzecinkowej jest niewygodny ponieważ wymaga dużej liczby znaków
- Jeśli chcemy rozszerzyć zakres czy precyzję (liczbę miejsc po przecinku) musimy użyć większej liczby znaków

  - dwanaście bilionów: 12 000 000 000 000
  - trzydzieści trylionów: 30 000 000 000 000 000 000
  - jedna bilionowa: 0,000 000 000 001

Zapis zmiennoprzecinkowy

- Zapis zmiennoprzecinkowy (ang. floating point numbers) zwany również postacią wykładniczą, notacją naukowa składa się z następujących elementów:
  - *m* – mantysy, czyli liczby stałoprzecinkowej
  - *p* – podstawy systemu
  - *c* – cechy, czyli wykładnika potęgowego, do którego podnosimy podstawę systemu
- Wartość liczby zmiennoprzecinkowej *L* obliczamy według wzoru:

$$L_{(FP)} = m * p^{c}$$

Zapis zmiennoprzecinkowy

- Zapis zmiennoprzecinkowy wymaga znacznie mniejszej liczby znaków niż stałoprzecinkowy przy zapisie bardzo dużych lub bardzo małych liczb
- Przykład dla systemu dziesiętnego:
  - 12 000 000 000 000 = $1{,}2 \times 10^{13}$
  - 30 000 000 000 000 000 000 = $3{,}0 \times 10^{19}$
  - 0,000 000 000 001 = $1{,}0 \times 10^{-12}$

Zapis zmiennoprzecinkowy

- Położenie przecinka w mantysie nie jest ustalone i może się dowolnie zmieniać
- Poniższe zapisy oznaczają tą samą liczbę:
  $325 \times 10^{20} = 32{,}5 \times 10^{21} = 3{,}25 \times 10^{22} = 0{,}0325 \times 10^{24}$, itd.
- Zmiana położenia przecinka w mantysie wpływa na wartość cechy liczby:
  - przesunięcie przecinka o 1 pozycję w lewo wymaga zwiększenia cechy o 1
  - przesunięcie przecinka o 1 pozycję w prawo wymaga zmniejszenia cechy o 1

Znormalizowany zapis

- Znormalizowana liczba zmiennoprzecinkowa to taka, w której mantysa spełnia nierówność: $p > |m| \geq 1$

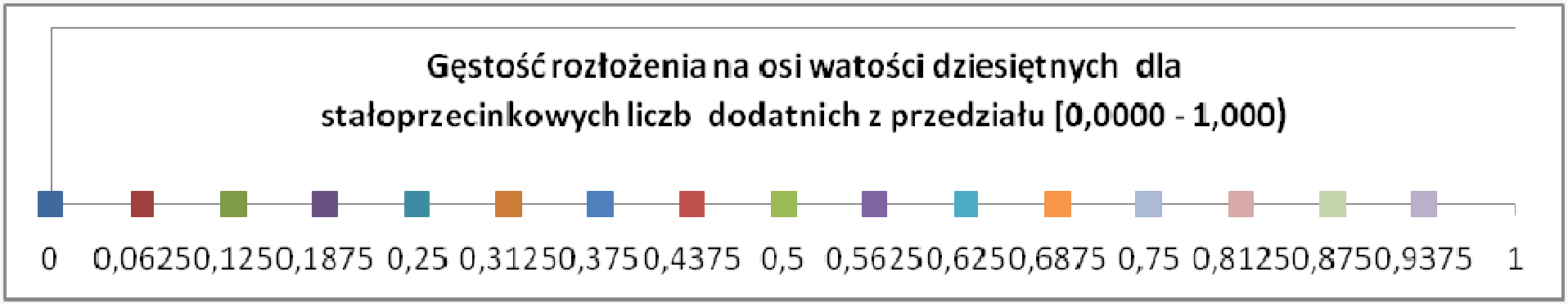
- Według tej definicji postacią znormalizowaną dla zapisów:
  $325 \times 10^{20} = 32{,}5 \times 10^{21} = 3{,}25 \times 10^{22} = 0{,}0325 \times 10^{24}$
  jest jedynie zapis $3{,}25 \times 10^{22}$

Binarny zapis zmiennoprzecinkowy

- W binarnym systemie zmiennoprzecinkowym *m*, *p* oraz *c* zapisane są dwójkowo, a podstawa *p* jest zawsze równa 2
- Z racji, że podstawa jest zawsze znana, do zapisania dwójkowej liczby zmiennoprzecinkowej wystarczy podanie wartości *m*, *c* oraz sposobu ich kodowania

Dokładność reprezentacji zmiennoprzecinkowej

- Określ, który element zapisu zmiennoprzecinkowego (cecha czy mantysa) odpowiada za zakres reprezentowanych liczb, a który za ich precyzję? Odpowiedź uzasadnij.
- Liczb maszynowych w reprezentacji zmiennoprzecinkowej w okolicy zera jest stosunkowo wiele, a im dalej od zera tym rzadziej się one pojawiają (inaczej niż przy reprezentacji stałoprzecinkowej, gdzie poszczególne liczby maszynowe były od siebie równo oddalone).



Nadmiar i podmiar

- W reprezentacji zmiennoprzecinkowej nadmiarem nazywamy sytuację, gdy liczba jest tak duża (co do modułu), że nie zawiera się w przedziale liczb reprezentowalnych
- Podmiarem sytuację, gdy liczba jest tak mała (co do modułu), że musi być reprezentowana przez zero

STANDARDY REPREZENTACJI ZMIENNOPRZECINKOWEJ

Format IEEE 754

- Aby ujednolicić wyniki obliczeń numerycznych wykonywanych na różnych platformach sprzętowych, wprowadzono ściśle określony standard zapisu zmiennoprzecinkowego IEEE 754
- Pełna nazwa standardu to:
  IEEE Standard for Binary Floating-Point Arithmetic
- Pierwsza wersja standardu obowiązywała od 1985r
- Obecnie praktycznie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych oparte są na tym standardzie

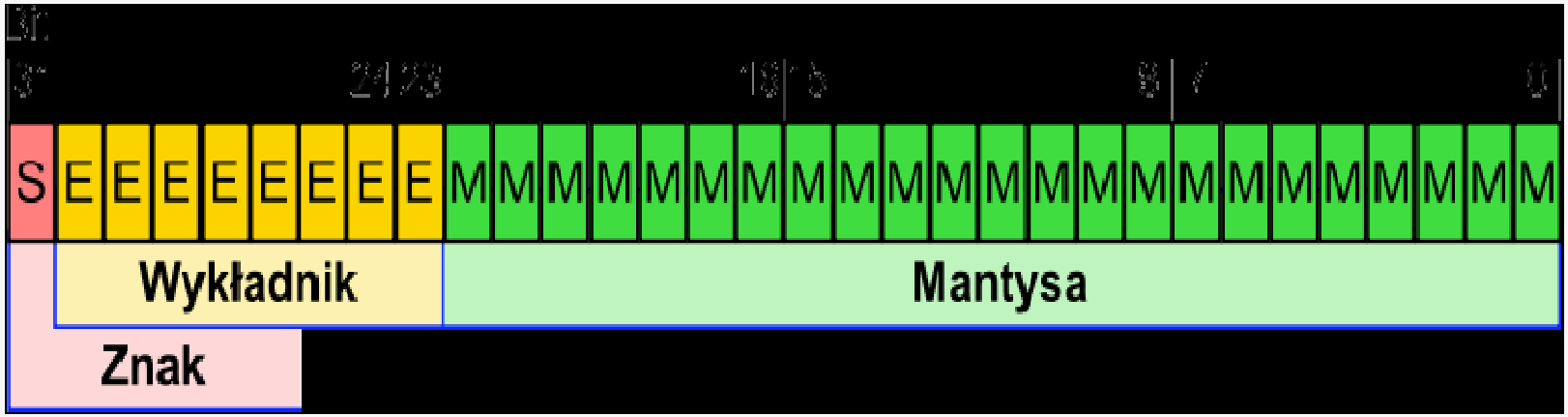
Format IEEE 754

- Standard IEEE 754 definiuje dwie podstawowe klasy binarnych liczb zmiennoprzecinkowych:
  - binary 32 - pojedynczej precyzji (ang. single-precision)
  - binary 64 - podwójnej precyzji (ang. double-precision)

Format	Bit znaku	Bity cechy	Bity mantysy
32 bity - pojedyncza precyzja	1 bit	8 bitów	23 bity
64 bity - podwójna precyzja	1 bit	11 bitów	52 bity

Format IEEE 754

- Reprezentacja zmiennoprzecinkowa IEEE 754 pojedynczej precyzji



  - bit znaku: 0 oznacza liczbę dodatnią, 1 ujemną
  - cecha zapisywana jest w kodzie z nadmiarem (dla 8-mio bitowego zapisu nadmiar wynosi 127, zatem w polu cechy można zapisać wartości od -127 do 128)
  - mantysa zapisywana jest w stałoprzecinkowym kodzie U1

Format IEEE 754

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