

Informatyka 1 (EZ1F1002)

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

Wykład nr 2 (18.10.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 2

- Język C
 - wyrażenia arytmetyczne, funkcje matematyczne (**math.h**)
 - funkcje printf() i scanf()

- Systemy liczbowe
 - liczby i cyfry
 - systemy pozycyjne i niepozycyjne
 - konwersje między systemami liczbowymi

- Jednostki informacji cyfrowej
 - bit, bajt słowo, FLOPS

- Kodowanie znaków
 - ASCII, ISO/IEC 646, ISO 8859
 - EBCDIC, Windows-1250, Unicode

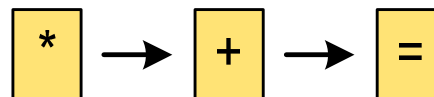
Język C - wyrażenia arytmetyczne

- Wyrażenia arytmetyczne mogą zawierać:
 - stałe liczbowe, zmienne, stałe
 - operatory: $+$ $-$ $*$ $/$ $\%$ $=$ $()$ i inne
 - wywołania funkcji (plik nagłówkowy **math.h**)
- Kolejność wykonywania operacji wynika z priorytetu operatorów

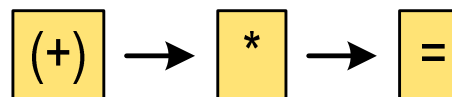
```
w = a + b;
```



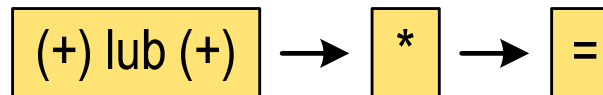
```
w = a + b * c;
```



```
w = (a + b) * c;
```



```
w = (a + b) * (c + d);
```



Język C - wyrażenia arytmetyczne

- Kolejność wykonywania operacji

$$\boxed{w = a + b + c;} \rightarrow \boxed{w = ((a + b) + c);}$$

$$\boxed{w = x = y = a + b;} \rightarrow \boxed{w = (x = (y = (a + b)))};$$

- Zapis wyrażeń arytmetycznych

$$w = \frac{a+b}{c+d}$$

$$\boxed{w = a + b / c + d;} \quad \text{ŹLE}$$

$$\boxed{w = (a + b) / (c + d);} \quad \text{DOBRZE}$$

$$w = \frac{a+b}{c \cdot d}$$

$$\boxed{w = (a + b) / c * d;} \quad \text{ŹLE}$$

$$\boxed{w = (a + b) / (c * d);} \quad \text{DOBRZE}$$

Język C - wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa

$$w = \frac{5}{4}$$

```
5 / 4 = 1
```

```
5.0 / 4 = 1.25
```

```
5 / 4.0 = 1.25
```

```
5.0 / 4.0 = 1.25
```

```
5.0f / 4 = 1.25
```

```
5. / 4 = 1.25
```

```
(float) 5 / 4 = 1.25
```

Rzutowanie: (typ)

Język C - funkcje matematyczne (math.h)

- Plik nagłówkowy **math.h** zawiera definicje wybranych stałych

Nazwa	Wartość	Znaczenie
M_PI	3.14159265358979323846	liczba pi
M_E	2.71828182845904523536	e - liczba Eulera
M_LN2	0.693147180559945309417	ln 2
M_SQRT2	1.41421356237309504880	$\sqrt{2}$

- W środowisku Visual Studio 2008 użycie stałych wymaga definicji odpowiedniej stałej (przed **#include <math.h>**)

```
#define _USE_MATH_DEFINES  
#include <math.h>
```

Język C - funkcje matematyczne (math.h)

- Wybrane funkcje matematyczne:

Nazwa	Nagłówek	Znaczenie
abs	<code>int abs(int x);</code>	moduł x (x - całkowite)
fabs	<code>double fabs(double x);</code>	moduł x (x - rzeczywiste)
sqrt	<code>double sqrt(double x);</code>	pierwiastek kwadratowy x
pow	<code>double pow(double x, double y);</code>	x^y - x do potęgi y
sin	<code>double sin(double x);</code>	sinus argumentu x w radianach
atan	<code>double atan(double x);</code>	arcus tangens argumentu x
atan2	<code>double atan2(double y, double x);</code>	arcus tangens ilorazu y/x

- Wszystkie funkcje mają po trzy wersje - dla argumentów typu: **float**, **double** i **long double**

Przykład: częstotliwość rezonansowa

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>

int main(void)
{
    double L, C, fr;

    printf("Podaj L [H]: "); scanf("%lf", &L);
    printf("Podaj C [F]: "); scanf("%lf", &C);

    fr = 1 / (2 * M_PI * sqrt(L * C));

    printf("-----\n");
    printf("fr [Hz]: %.3f\n", fr);

    return 0;
}
```

```
Podaj L [H]: 0.01
Podaj C [F]: 1e-6
-----
fr [Hz]: 1591.549
```

$$f_r = \frac{1}{2\pi\sqrt{LC}}$$

Język C - Funkcja printf()

- Ogólna składnia funkcji `printf()`

```
printf("łańcuch_sterujący", arg1, arg2, ...);
```

- W najprostszej postaci `printf()` wyświetla tylko tekst

```
printf("Witaj świecie");
```

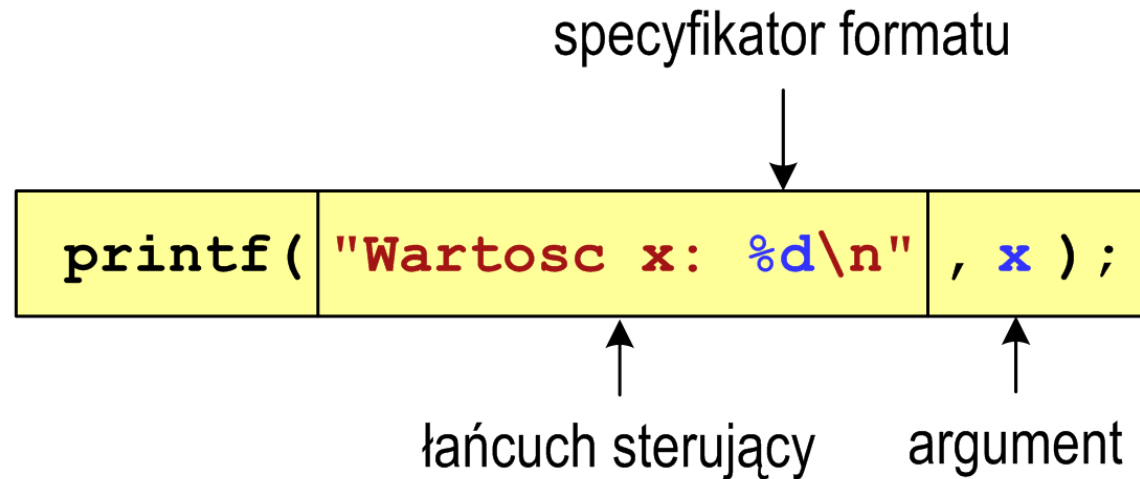
```
Witaj świecie
```

- Do wyświetlenia wartości zmiennych konieczne jest zastosowanie **specyfikatorów formatu**, określających typ oraz sposób wyświetlania argumentów

```
%[znacznik][szerokość][.precyzja][modyfikator]typ
```

Język C - Funkcja printf()

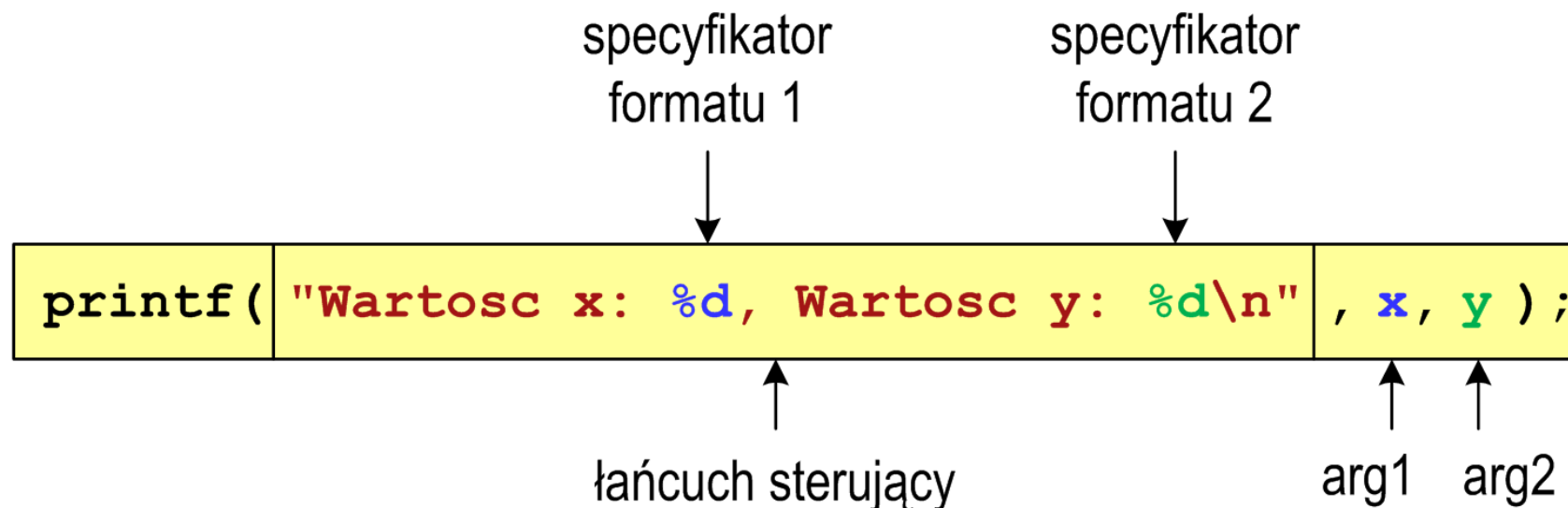
```
int x = 10;  
printf("Wartosc x: %d\n", x);
```



```
Wartosc x: 10
```

Język C - Funkcja printf()

```
int x = 10, y = 20;  
printf("Wartosc x: %d, Wartosc y: %d\n", x, y);
```



```
Wartosc x: 10, Wartosc y: 20
```

Język C - Specyfikatory formatu (printf)

Typ w C	Specyfikator	Uwagi
char	%c	pojedynczy znak
	%d	kod ASCII znaku, liczba całkowita
char *	%s	łańcuch znaków, napis
int	%d %i	liczba całkowita, dziesiętna
	%o %O	liczba całkowita, ósemkowa
	%x %X	liczba całkowita, szesnastkowa
float double	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)

Język C - Funkcja printf()

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%d], y = [%f]\n", x, y);
```

```
x = [123], y = [1.234568]
```

```
printf("x = [], y = []\n", x, y);
```

```
x = [], y = []
```

```
printf("x = [%d], y = [%d]\n", x, y);
```

```
x = [123], y = [-536870912]
```

Język C - Funkcja printf()

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%6d], y = [%12f]\n", x, y);
```

```
x = [ 123], y = [ 1.234568]
```

```
printf("x = [%6d], y = [%12.3f]\n", x, y);
```

```
x = [ 123], y = [ 1.235]
```

```
printf("x = [%6d], y = [%.3f]\n", x, y);
```

```
x = [ 123], y = [1.235]
```

Język C - Funkcja printf()

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%+6d], y = [%+12f]\n", x, y);
```

```
x = [ +123], y = [ +1.234568]
```

```
printf("x = [%-6d], y = [%-12f]\n", x, y);
```

```
x = [123   ], y = [1.234568   ]
```

```
printf("x = [%06d], y = [%012f]\n", x, y);
```

```
x = [000123], y = [00001.234568]
```

Język C - Funkcja printf()

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%d], y = [%f]\n", x+321, y*25.5f);
```

```
x = [444], y = [31.481482]
```

```
printf("x = [%d], y = [%f]\n", 123, 2.0f*sqrt(y));
```

```
x = [123], y = [2.222222]
```


Język C - Funkcja scanf()

- Ogólna składnia funkcji `scanf()`

```
scanf ("specyfikator", adresy_argumentów) ;
```

- Składnia `specyfikatora formatu`

```
% [szerokość] [modyfikator] typ
```

- Argumenty są adresami obszarów pamięci, dlatego muszą być poprzedzone znakiem `&`

```
int x;  
scanf ("%d", &x) ;
```

Język C - Funkcja scanf()

- **Specyfikatory formatu** w większości przypadków są takie same jak w przypadku funkcji **printf()**
- Największa różnica dotyczy typów **float i double**

Typ w C	Specyfikator	Uwagi
float	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)
double	%lf	liczba rzeczywista
	%le %LE	liczba rzeczywista, format naukowy
	%lg %LG	liczba rzeczywista (%f lub %e)

Język C - Funkcja scanf()

```
int a, b, c;  
scanf("%d %d %d", &a, &b, &c);
```

- Wczytywane argumenty mogą być oddzielone od siebie dowolną liczbą białych (niedrukowalnych) znaków: **spacja, tabulacja, enter**

```
15 20 -30
```

```
15 20 -30<enter>
```

```
15    20    -30
```

```
15    20    -30<enter>
```

```
15  
20  
-30
```

```
15<enter>  
20<enter>  
-30<enter>
```

Liczby i cyfry

- **Liczba** - pojęcie abstrakcyjne, abstrakcyjny wynik obliczeń, wartość
 - umożliwia wyrażenie wyniku liczenia przedmiotów oraz mierzenia wielkości
- **Cyfra** - umowny znak (symbol) stosowany do zapisu liczby
 - liczba znaków służących do zapisu jest zależna od **systemu liczbowego** i przyjętego sposobu zapisu
 - system dziesiętny - 10 znaków
 - system szesnastkowy - 16 znaków
 - system rzymski - 7 znaków
- Cyfry rzymskie

I	V	X	L	C	D	M
<i>1</i>	<i>5</i>	<i>10</i>	<i>50</i>	<i>100</i>	<i>500</i>	<i>1000</i>

Liczby i cyfry

- Cyfry arabskie (pochodzą z Indii)
 - arabskie, standardowe europejskie

1	2	3	4	5	6	7	8	9	0
---	---	---	---	---	---	---	---	---	---

- indyjsko-arabskie

१	२	३	४	५	६	७	८	९	०
1	2	3	4	5	6	7	8	9	0

- wschodnio-indyjsko-arabskie

١	٢	٣	٤	٥	٦	٧	٨	٩	٠
1	2	3	4	5	6	7	8	9	0

- W niektórych systemach jako cyfry stosowane są litery, np.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:

cyfry etruskie

┆	∧	X	XX	∧XX	↑	*	(C)	⊙	⊙
1	5	10	20	25	50	100		1000	

cyfry grecko-jońskie

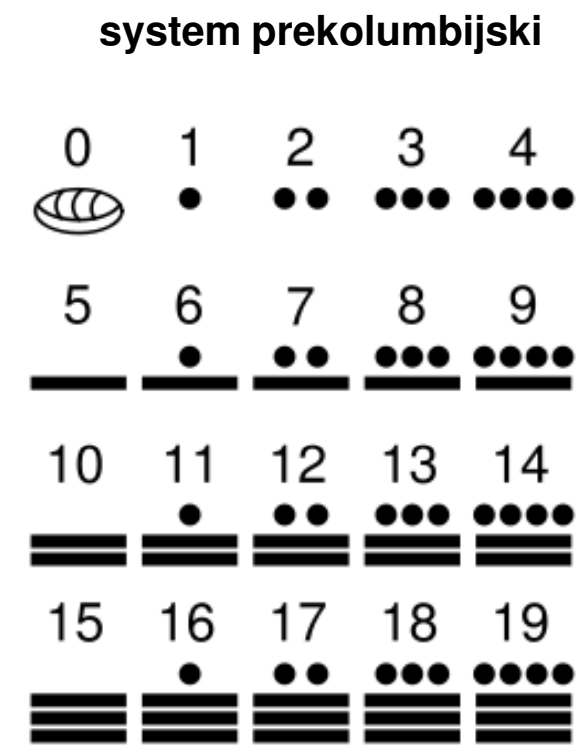
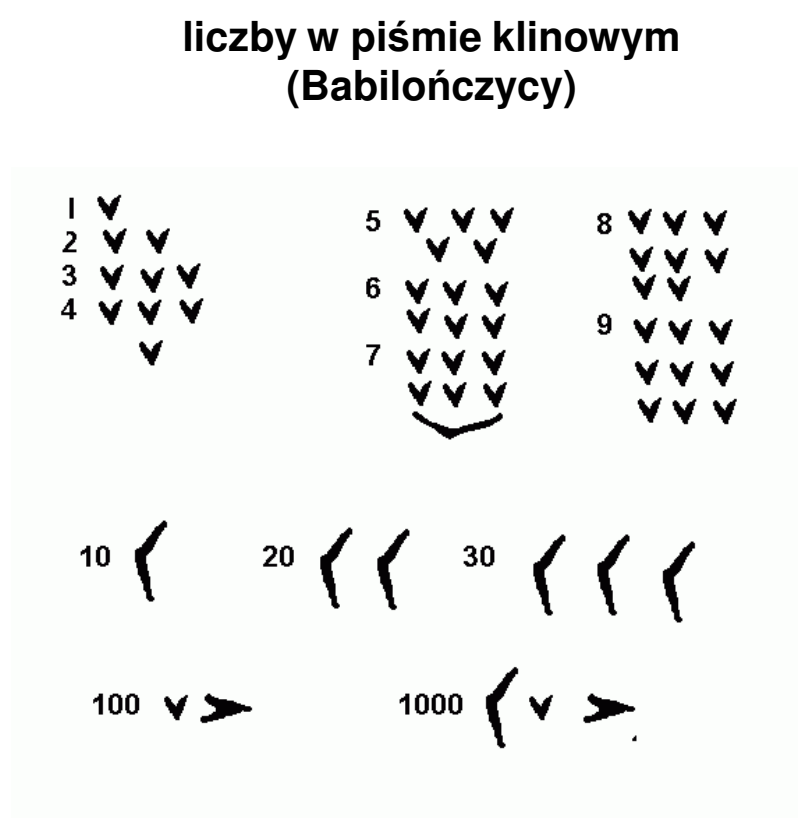
α	β	γ	δ	ε	ς	ζ	η	θ
1	2	3	4	5	6	7	8	9
ι	κ	λ	μ	ν	ξ	ο	π	ρ
10	20	30	40	50	60	70	80	90
σ	ϖ	τ	υ	φ	χ	ψ	ω	Ͱ
100	200	300	400	500	600	700	800	900
α'	β'	γ'	δ'	ε'	ς'	ζ'	η'	θ'
1000	2000	3000	4000	5000	6000	7000	8000	9000
α'Ͱε'β'								

cyfry w pisowni chińskiej

jeden	一	sześć	六
dwa	二	siedem	七
trzy	三	osiem	八
cztery	四	dziewięć	九
pięć	五	dziesięć	十
zero	另		

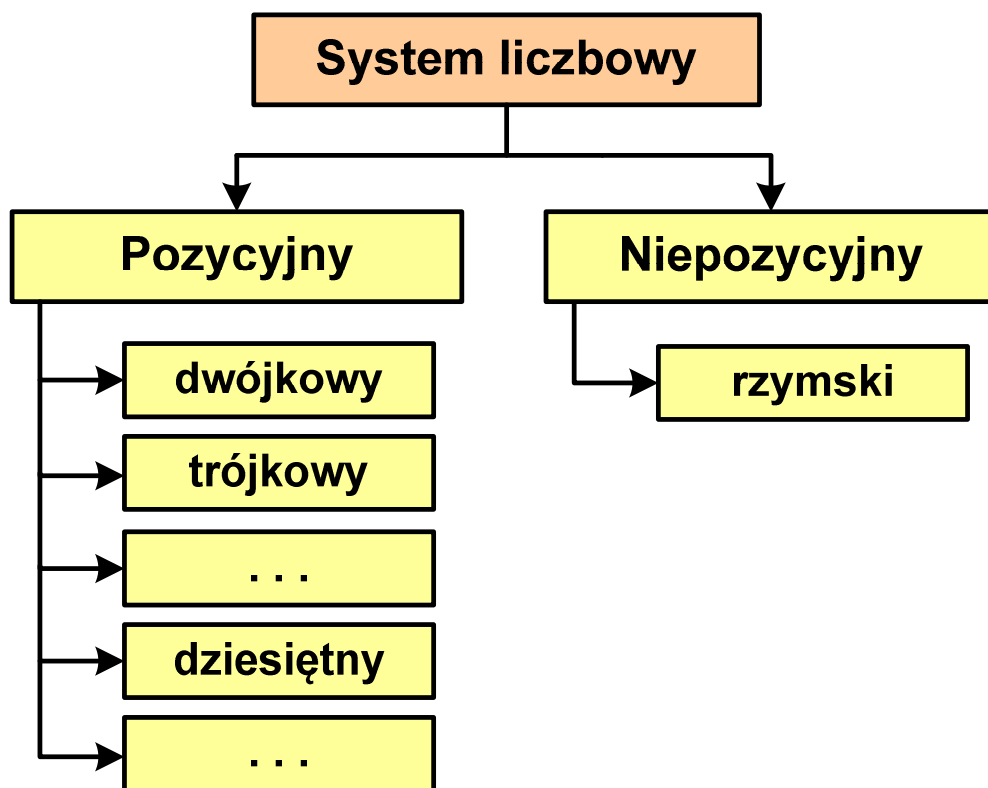
Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:



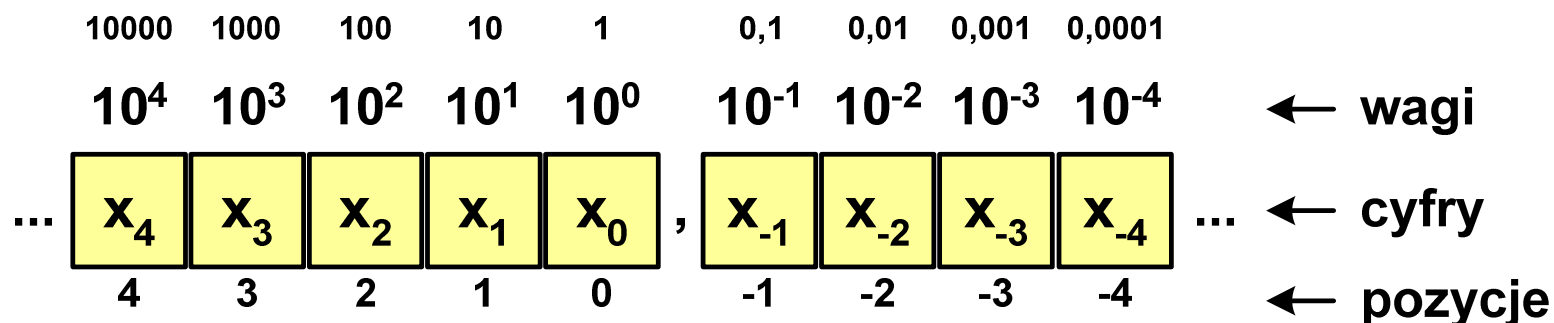
Systemy liczbowe

- **System liczbowy** - zbiór zasad umożliwiających zapis liczb za pomocą cyfr oraz wykonywanie działań na tych liczbach



- **Pozycyjny** - znaczenie cyfry jest zależne od miejsca (pozycji), które zajmuje ona w liczbie
 - system dziesiętny - liczba **111** (każda cyfra ma inne znaczenie)
- **Niepozycyjny** - znaczenie cyfry jest niezależne od miejsca położenia w liczbie
 - system rzymski - liczba **III**

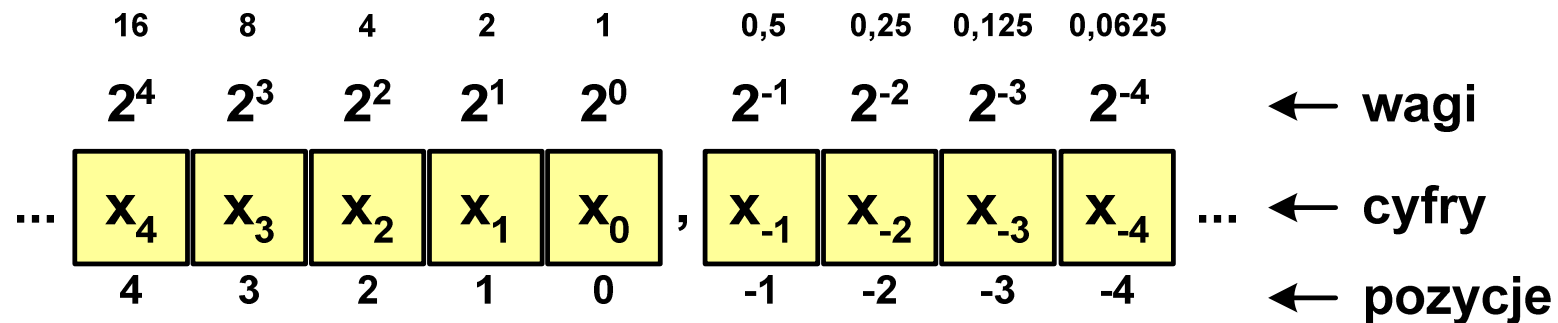
System dziesiętny (ang. decimal)



- p - podstawa systemu pozycyjnego, D - zbiór dozwolonych cyfr
- $p = 10$, $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$\begin{array}{r}
 \begin{array}{cccccc}
 10^3 & 10^2 & 10^1 & 10^0 & 10^{-1} & 10^{-2} \\
 \boxed{1} & \boxed{4} & \boxed{0} & \boxed{8} & \boxed{2} & \boxed{5}
 \end{array} \\
 \begin{array}{l}
 \diagup \quad \diagdown \quad \diagup \quad \diagdown \quad \diagup \quad \diagdown \\
 \end{array} \\
 1408,25_{(10)} = \\
 = \boxed{1 \cdot 10^3} + \boxed{4 \cdot 10^2} + \boxed{0 \cdot 10^1} + \boxed{8 \cdot 10^0} + \boxed{2 \cdot 10^{-1}} + \boxed{5 \cdot 10^{-2}} \\
 = 1000 + 400 + 0 + 8 + 0,2 + 0,05
 \end{array}$$

System dwójkowy (ang. binary)



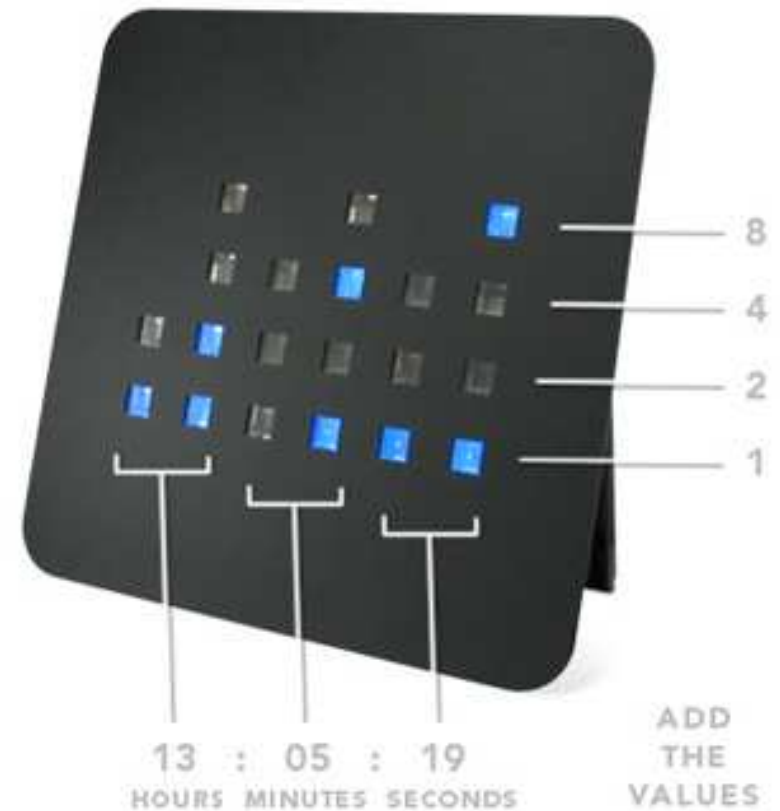
- w systemie dwójkowym: $p = 2, D = \{0, 1\}$

The diagram shows the conversion of the binary number $1101,101_{(2)}$ to decimal. The binary digits are arranged in a row: $1, 1, 0, 1, 1, 0, 1$. Above the digits are their weights: $2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, 2^{-3}$. Lines connect each digit to its corresponding weight. Below the digits, the expansion is shown: $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$. This is then simplified to $8 + 4 + 0 + 1 + 0,5 + 0 + 0,125$, and finally to $13,625_{(10)}$.

$$\begin{aligned} 1101,101_{(2)} &= \\ &= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= 8 + 4 + 0 + 1 + 0,5 + 0 + 0,125 \\ &= 13,625_{(10)} \end{aligned}$$

System dwójkowy - zastosowania

- Powszechnie używany w informatyce, technice cyfrowej



System szesnastkowy (ang. hexadecimal)

- System heksadecymalny
- $p = 16$, $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- Powszechnie używany w informatyce - jeden bajt można zapisać za pomocą tylko dwóch cyfr szesnastkowych

$$3A5D_{(16)} = 3 \cdot 16^3 + 10 \cdot 16^2 + 5 \cdot 16^1 + 13 \cdot 16^0 = 14941_{(10)}$$

- Sposoby zapisu liczb w systemie szesnastkowym:

3A5Dh

0x3A5D

#3A5D

$3A5D_{(16)}$

$3A5D_{16}$

$3A5D_{\text{hex}}$

$(3A5D)_{\text{hex}}$

$(3A5D)_{16}$

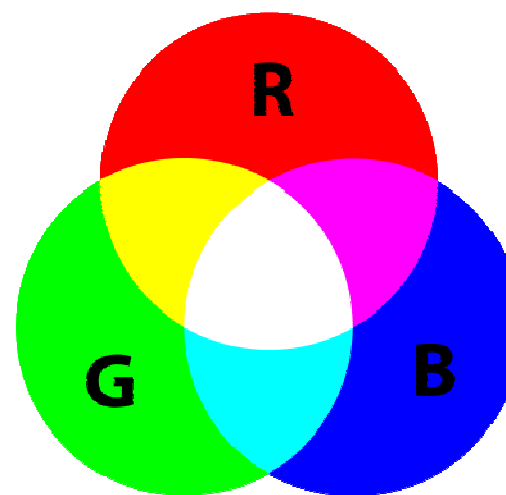
\$3A5D

System szesnastkowy - zastosowania

- Zapis 24-bitowego koloru RGB (Red-Green-Blue), 16 mln kolorów
- Każda barwa przyjmuje wartość z zakresu: $0..255_{(10)}$, $00..FF_{(16)}$



#FF48B8



System szesnastkowy - zastosowania

- Zapis 24-bitowego koloru RGB (**Red-Green-Blue**), 16 mln kolorów
- Kolory w dokumentach HTML:

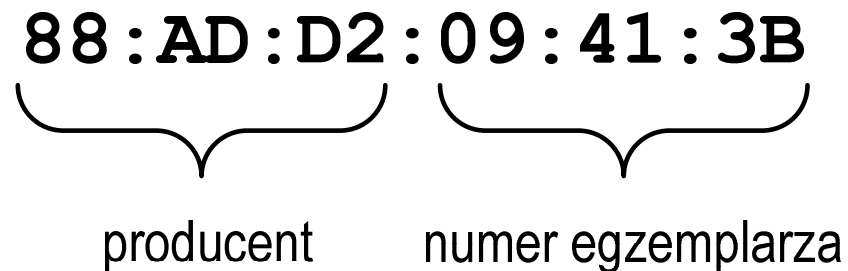
```
<BODY bgcolor="#336699" text="#000000" link="#FFFF00"  
vlink="#33FFFF" alink="#FF0000">
```

The screenshot shows a website interface with a dark blue background. On the left, there is a vertical sidebar with two buttons: 'ARCHIWUM' and 'ENGLISH'. The main content area is titled 'Studia stacjonarne:' and lists a schedule for 'Poniedziałek:' and 'Wtorek:'. The schedule includes times and links to course pages.

ARCHIWUM	Studia stacjonarne:
ENGLISH	Poniedziałek:
	12:15 - 14:00 Informatyka 1 - wykład , sem. 2 ED, WE-Aula II
	Wtorek:
	08:30 - 10:00 Informatyka 1 - prac. , sem. 2 ED, gr. PS3, WE-110
	12:15 - 13:45 Informatyka 1 - prac. , sem. 2 ED, gr. PS2, WE-110
	14:15 - 15:45 Informatyka 1 - prac. , sem. 2 ED, gr. PS1, WE-110

System szesnastkowy - zastosowania

- 48-bitowy adres fizyczny urządzenia (MAC - Media Access Control)



- <http://hwaddress.com>

OUI	MAC range	Company
88-AD-D2	88-AD-D2-00-00-00 - 88-AD-D2-FF-FF-FF	Samsung Electronics Co.,Ltd

Przykład systemu niepozycyjnego - system rzymski

- W systemie rzymskim posługujemy się siedmioma znakami:
 $I - 1$ $V - 5$ $X - 10$ $L - 50$ $C - 100$ $D - 500$ $M - 1000$
- Za pomocą dostępnych symboli można określić liczby od **1** do **3999**
- System **addytywny** - wartość liczby określa się na podstawie sumy wartości cyfr, np.
 - II ($1 + 1 = 2$), XXX ($10 + 10 + 10 = 30$)
 - CLX ($100 + 50 + 10 = 160$), $MMXII$ ($1000 + 1000 + 10 + 1 + 1 = 2012$)
- Wyjątkiem od powyższej zasady są liczby do opisu których używa się odejmowania, np.
 - IV ($5 - 1 = 4$), IX ($10 - 1 = 9$), XL ($50 - 10 = 40$), XC ($100 - 10 = 90$)
- Stosowany w łacińskiej części Europy do końca Średniowiecza
- Niewygodny w prowadzeniu nawet prostych działań arytmetycznych, brak ułamków

Konwersja na system dziesiętny

- $p = 4, D = \{0, 1, 2, 3\}$

4^4	4^3	4^2	4^1	4^0
2	1	3	0	2

$$21302_{(4)} = ?_{(10)}$$

$$21302_{(4)} = 2 \cdot 4^0 + 0 \cdot 4^1 + 3 \cdot 4^2 + 1 \cdot 4^3 + 2 \cdot 4^4$$

$$21302_{(4)} = 2 \cdot 1 + 0 \cdot 4 + 3 \cdot 16 + 1 \cdot 64 + 2 \cdot 256$$

$$21302_{(4)} = 2 + 0 + 48 + 64 + 512 = 626_{(10)}$$

- $p = 17, D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G\}$

17^3	17^2	17^1	17^0
A	C	2	4

$$AC24_{(17)} = ?_{(10)}$$

$$AC24_{(17)} = 4 \cdot 17^0 + 2 \cdot 17^1 + 12 \cdot 17^2 + 10 \cdot 17^3$$

$$AC24_{(17)} = 4 \cdot 1 + 2 \cdot 17 + 12 \cdot 289 + 10 \cdot 4913$$

$$AC24_{(17)} = 4 + 34 + 3468 + 49130 = 52636_{(10)}$$

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu $p = 10$ na system $p = 2$

$$626_{(10)} = ?_{(2)}$$

$$626_{(10)} = 1001110010_{(2)}$$

$626/2 = 313$	<i>reszta</i>	0
$313/2 = 156$	<i>reszta</i>	1
$156/2 = 78$	<i>reszta</i>	0
$78/2 = 39$	<i>reszta</i>	0
$39/2 = 19$	<i>reszta</i>	1
$19/2 = 9$	<i>reszta</i>	1
$9/2 = 4$	<i>reszta</i>	1
$4/2 = 2$	<i>reszta</i>	0
$2/2 = 1$	<i>reszta</i>	0
$1/2 = 0$	<i>reszta</i>	1

kolejność odczytywania
cyfr liczby w systemie
dwójkowym

kończymy, gdy liczba dziesiętna ma wartość 0

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu $p = 10$ na system $p = 7$

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

$626/7 = 89$	<i>reszta</i>	3	↑
$89/7 = 12$	<i>reszta</i>	5	
$12/7 = 1$	<i>reszta</i>	5	
$1/7 = 0$	<i>reszta</i>	1	

- zamiana liczby z systemu $p = 10$ na system $p = 14$

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

$626/14 = 44$	<i>reszta</i>	10	↑	→ A
$44/14 = 3$	<i>reszta</i>	2		
$3/14 = 0$	<i>reszta</i>	3		

Szybkie konwersje: $2 \rightarrow 4, 8, 16$ $4, 8, 16 \rightarrow 2$

$2 \rightarrow 4$

$$110110011_{(2)} = ?_{(4)}$$
$$\underbrace{01}_1 \mid \underbrace{10}_2 \mid \underbrace{11}_3 \mid \underbrace{00}_0 \mid \underbrace{11}_3$$
$$110110011_{(2)} = 12303_{(4)}$$

$2 \rightarrow 8$

$$10110011_{(2)} = ?_{(8)}$$
$$\underbrace{010}_2 \mid \underbrace{110}_6 \mid \underbrace{011}_3$$
$$10110011_{(2)} = 263_{(8)}$$

$2 \rightarrow 16$

$$1011010_{(2)} = ?_{(16)}$$
$$\underbrace{0101}_5 \mid \underbrace{1010}_A$$
$$1011010_{(2)} = 5A_{(16)}$$

$4 \rightarrow 2$

$$12303_{(4)} = ?_{(2)}$$
$$\underbrace{01}_1 \mid \underbrace{10}_2 \mid \underbrace{11}_3 \mid \underbrace{00}_0 \mid \underbrace{11}_3$$
$$12303_{(4)} = 110110011_{(2)}$$

$8 \rightarrow 2$

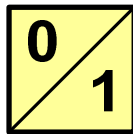
$$263_{(8)} = ?_{(2)}$$
$$\underbrace{010}_2 \mid \underbrace{110}_6 \mid \underbrace{011}_3$$
$$263_{(8)} = 10110011_{(2)}$$

$16 \rightarrow 2$

$$5A_{(16)} = ?_{(2)}$$
$$\underbrace{0101}_5 \mid \underbrace{1010}_A$$
$$5A_{(16)} = 1011010_{(2)}$$

Jednostki informacji - bit

- **Bit** (ang. **binary digit**) - podstawowa jednostka informacji stosowana w informatyce i telekomunikacji
- Określa najmniejszą ilość informacji potrzebną do stwierdzenia, który z dwóch możliwych stanów przyjął układ
- Bit przyjmuje jedną z dwóch wartości:
 - 0 (zero)
 - 1 (jeden)
- Bit jest tożsamy z cyfrą w systemie dwójkowym
- Oznaczenia bitów:
 - standard IEEE 1541 (2002) - mała litera „b”
 - standard IEC 60027 - „bit”



Jednostki informacji - bit

■ Wielokrotności bitów:

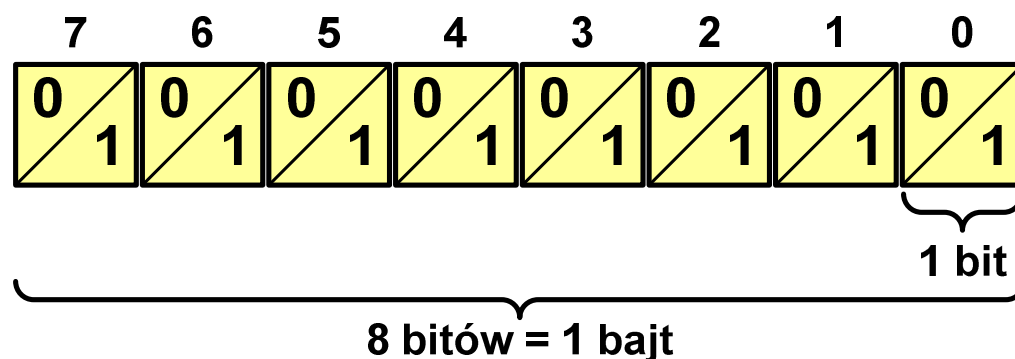
Przedrostki dziesiętne (układ SI)		
Nazwa	Symbol	Mnożnik
bit	b	---
kilobit	kb	$10^3 = 1000^1$
megabit	Mb	$10^6 = 1000^2$
gigabit	Gb	$10^9 = 1000^3$
terabit	Tb	$10^{12} = 1000^4$
petabit	Pb	$10^{15} = 1000^5$
eksabit	Eb	$10^{18} = 1000^6$
zettabit	Zb	$10^{21} = 1000^7$
jottabit	Yb	$10^{24} = 1000^8$

Przedrostki binarne (IEC 60027-2)		
Nazwa	Symbol	Mnożnik
bit	b	---
kibibit	Kib	$2^{10} = 1024^1$
mebibit	Mib	$2^{20} = 1024^2$
gibibit	Gib	$2^{30} = 1024^3$
tebibit	Tib	$2^{40} = 1024^4$
pebibit	Pib	$2^{50} = 1024^5$
eksbibit	Eib	$2^{60} = 1024^6$
zebibit	Zib	$2^{70} = 1024^7$
jobibit	Yib	$2^{80} = 1024^8$

- **Przedrostki binarne** - wprowadzone w 1999 roku w celu odróżnienia przedrostków o mnożniku 1000 (10^3) od przedrostków o mnożniku 1024 (2^{10})

Jednostki informacji - bajt

- **Bajt** (ang. byte) - najmniejsza adresowalna jednostka informacji pamięci komputerowej składająca się z bitów
- W praktyce przyjmuje się, że jeden bajt to 8 bitów



- Za pomocą jednego bajtu można zapisać $2^8 = 256$ różnych wartości:

0000 0000	→	0
0000 0001	→	1	1111 1101	→ 253
0000 0010	→	2	1111 1110	→ 254
...		...	1111 1111	→ 255

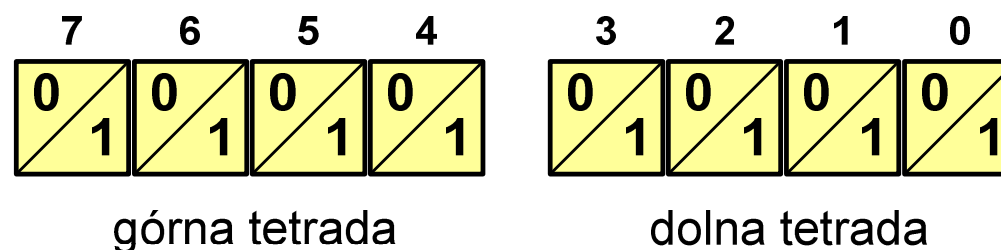
Jednostki informacji - bajt

- W pierwszych komputerach bajt mógł mieć inną liczbę bitów: 4, 6, 7, 9, 12
- 8-bitowy bajt:
 - koniec 1956 r. - pierwsze zastosowanie
 - 1964 r. - uznanie za standard (IBM System/360)
- Inna nazwa 8-bitowego bajtu - **oktet**
- Najczęściej stosowanym skrótem dla bajtu jest wielka litera „**B**”
 - „**B**” używane jest także do oznaczania **bela** - jednostki miary wielkości ilorazowych
 - zamiast bela częściej używa się jednostki podwielokrotnej - **decybela (dB)** więc nie ma problemu z rozróżnieniem obu jednostek

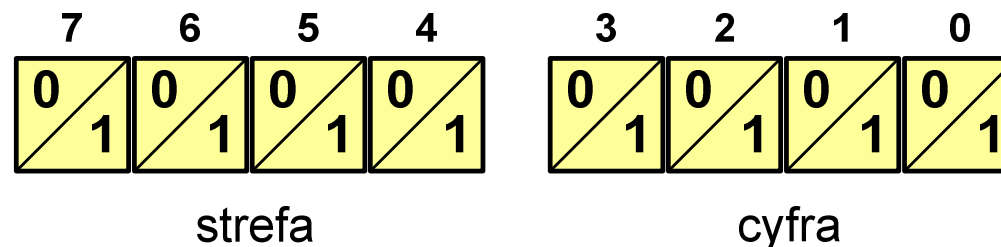


Jednostki informacji - tetrada

- Bajt 8-bitowy można podzielić na dwie połówki 4-bitowe nazywane **tetradami** (ang. nibbles)
- Rozróżniamy bardziej znaczącą (górną) i mniej znaczącą (dolną) tetradę



- Spotyka się też określenie **strefa** i **cyfra**



Jednostki informacji - bajt

■ Wielokrotności bajtów:

Przedrostki dziesiętne (układ SI)		
Nazwa	Symbol	Mnożnik
bajt	B	---
kilobajt	kB	$10^3 = 1000^1$
megabajt	MB	$10^6 = 1000^2$
gigabajt	GB	$10^9 = 1000^3$
terabajt	TB	$10^{12} = 1000^4$
petabajt	PB	$10^{15} = 1000^5$
eksabajt	EB	$10^{18} = 1000^6$
zettabajt	ZB	$10^{21} = 1000^7$
jottabajt	YB	$10^{24} = 1000^8$

Przedrostki binarne (IEC 60027-2)		
Nazwa	Symbol	Mnożnik
bajt	B	---
kibibajt	KiB	$2^{10} = 1024^1$
mebibajt	MiB	$2^{20} = 1024^2$
gibibajt	GiB	$2^{30} = 1024^3$
tebibajt	TiB	$2^{40} = 1024^4$
pebibajt	PiB	$2^{50} = 1024^5$
eksbibajt	EiB	$2^{60} = 1024^6$
zebibajt	ZiB	$2^{70} = 1024^7$
jobibajt	YiB	$2^{80} = 1024^8$

Jednostki informacji - bajt

- Przedrostki binarne (dwójkowe) nie zostały przyjęte przez wszystkie środowiska zajmujące się informatyką
- Producenci nośników pamięci korzystają z przedrostków dziesiętnych

Prefiks	Nazwa	System SI	System binarny	Różnica
k	kilo	$10^3 = 1000$	$2^{10} = 1024$	2,40%
M	mega	$10^6 = 1000000$	$2^{20} = 1048576$	4,86%
G	giga	$10^9 = 1000000000$	$2^{30} = 1073741824$	7,37%
T	tera	$10^{12} = 1000000000000$	$2^{40} = 1099511627776$	9,95%

- Z ulotki „Dysk Desktop HDD - zestawienie danych”, Seagate:
 - w przypadku oznaczania pojemności dysków, jeden gigabajt (oznaczany także jako „GB”) jest równy jednemu miliardowi bajtów, a jeden terabajt (oznaczany także jako „TB”) jest równy jednemu bilionowi bajtów

Jednostki informacji - bajt

- Seagate ST1000DM003 (1 TB)
- Drive specification:
 - formatted capacity: 1000 GB (1 TB)
 - guaranteed sectors: 1,953,525,168
 - bytes per sector: 4096
(4K physical emulated at 512-byte sectors)
- Pojemność dysku:
 - $1.953.525.168 \times 512 = 1.000.204.886.016$ bajtów
 - $1.000.204.886.016 / (1024) = 976.762.584$ kB
 - $1.000.204.886.016 / (1024 \times 1024) = 953.870$ MB
 - $1.000.204.886.016 / (1024 \times 1024 \times 1024) = 931,5$ GB



Słowo maszynowe (słowo)

- **Słowo maszynowe** (**słowo** - ang. word) - jednostka danych używana przez określony komputer (określoną architekturę)
- Słowo składa się odgórnie określonej liczby bitów, nazywanej **długością** lub **szerokością słowa** (najczęściej jest to potęga 2, np. 8, 16, 32, 64 bity)
- Zazwyczaj wielkość słowa określa:
 - rozmiar rejestrów procesora
 - rozmiar szyny danych i szyny adresowej
- Architektury:
 - 8-bitowa: Intel 8080, Z80, Motorola 6800, Intel 8051
 - 16-bitowa: Intel 8086, Intel 80286
 - 32-bitowa: Intel od 80386 do i7, AMD od 5x86 do Athlona, ARM
 - 64-bitowa: Intel Itanium, Pentium 4/EM64T, Core 2, Core i7
AMD Opteron, Athlon 64, Athlon II

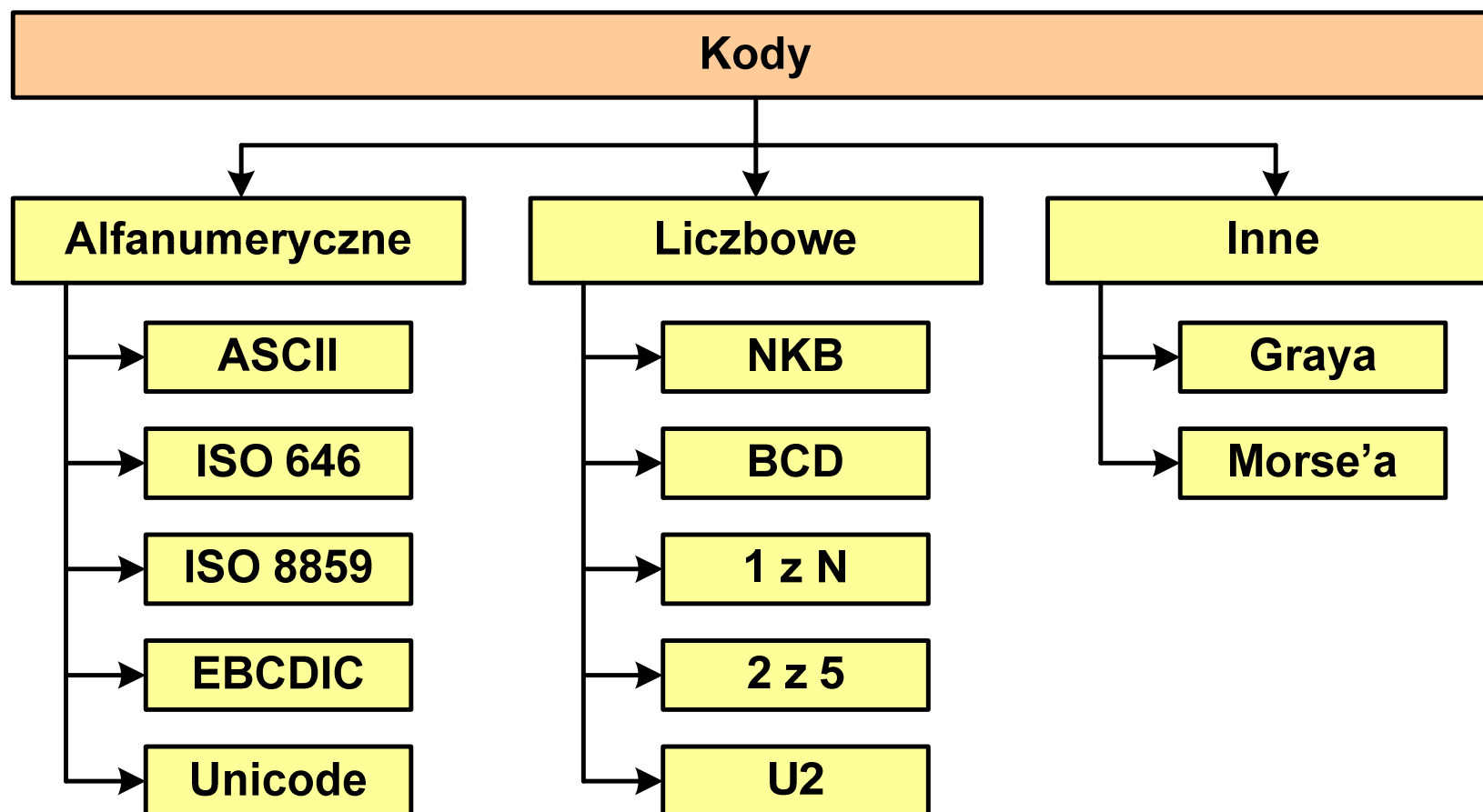
FLOPS

- **FLOPS (FL**oating point **O**perations **P**er **S**econd)
 - liczba operacji zmiennoprzecinkowych na sekundę
 - jednostka wydajności układów zmiennoprzecinkowych
- Przykłady wydajności procesorów (teoretyczne):
 - Intel Core i7 975 3,46 GHz - 55,36 GFlops
 - Intel Core2 Quad Q9650 3,00 GHz - 48 GFlops
 - Intel Core2 Duo E8400 3,00 GHz - 24 GFlops
 - najszybszy system równoległy na świecie:
Frontier (USA), HPE+AMD - 1.206 PFlops
DOE/SC/Oak Ridge National Laboratory
processors: AMD Optimized 3rd
Generation EPYC 64C 2 GHz,
cores: 8.699.904, HPE Cray OS
US\$600M, power: 21 MW
www.top500.org



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 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 8859-2

- 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	Æ	Ç	Ð	Ñ	Ò	Ó	Ô	Õ	Ö
B0	°	à	á	â	ã	ä	å	æ	ç	ð	ñ	ò	ó	ô	õ	ö
C0																
D0	Đ	Ñ	Ń	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ü	Ý	Ť	ß
E0	đ	ñ	ń	ó	ô	õ	ö	÷	ř	ů	ú	ů	ü	ý	ť	ß
F0	đ	ñ	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																
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	Ă	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	ă	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	·



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-8
- **UTF** - UCS Transformation Format, **UCS** - Universal Character Set
- Metody kodowania różnią się liczbą bajtów przeznaczonych do opisanego kodu znaku
- Wszystkie metody obejmują wszystkie kodowane znaki w Unicode.



Unicode - kodowanie UTF-32

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

A	Ω	語	卍	UTF-32
00000041	000003A9	00008A9E	00010384	

- 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

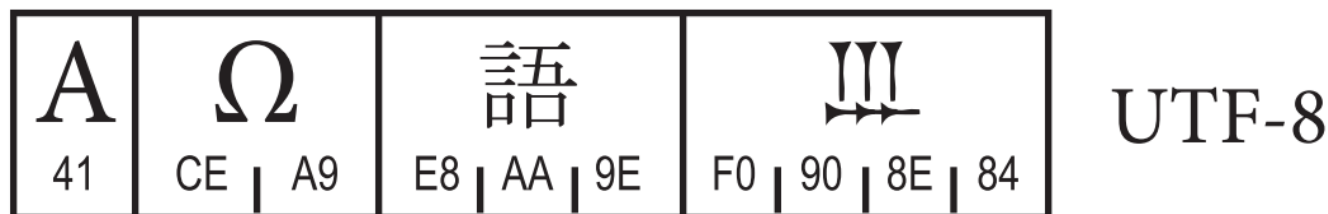


- Dla znaków z przedziału od **U+0000** do **U+FFFF** używane jest jedno słowo, którego wartość jest jednocześnie kodem znaku w Unicode
- Dla znaków z wyższych pozycji używa się dwóch słów:
 - pierwsze słowo należy do przedziału: **U+D800 - U+DBFF**
 - drugie słowo należy do przedziału: **U+DC00 - U+DFFF**.



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 Ǻ LATIN CAPITAL LETTER A WITH BREVE
≡ 0041 A 0306 ̆
- 0103 ǻ 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

Koniec wykładu nr 2

Dziękuję za uwagę!