1. MASZYNOWA REPREZENTACJA INFORMACJI

1.1. Pozycyjne systemy liczenia

- Przykładem pozycyjnego systemu liczenia (liczbowego) jest system dziesiętny (system o podstawie 10).
- Każda liczba całkowita N ≥ 2 może być . . podstawa . . systemu liczenia (mówimy wówczas o systemie o podstawie N).
- W informatyce najczęściej wykorzystywane systemy liczbowe:
  - dziesiętny (. . decymalny . .),
  - dwójkowy (. . binarny . .),
  - ósemkowy (. . oktalny . .),
  - szesnastkowy (. . heksadecymalny . .).
- Do zapisu liczb wykorzystywane są cyfry:
  - w systemie dwójkowym: 0, 1;
  - w systemie dziesiętnym: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
  - w systemie ósemkowym: 0, 1, 2, 3, 4, 5, 6, 7;
  - w systemie szesnastkowym: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F;
  - w systemie o podstawie N: 0, 1, ..., N - 1
- O tym z jakim systemem liczbowym mamy do czynienia zazwyczaj informuje podstawa systemu podawana w indeksie dolnym liczby lub dodatkowa litera na końcu liczby (często za wyjątkiem systemu dziesiętnego), np.
  - w systemie dwójkowym: $101_{(2)}$ lub . . . 101b,
  - w systemie dziesiętnym: $353_{(10)}$ lub . . 353d . lub 353,
  - w systemie szesnastkowym: $3A_{(16)}$ lub 0x3A lub . . 3Ah . .,
  - w systemie ósemkowym: $71_{(8)}$ lub . . 071 . .
- Liczby w różnych systemach:

System dziesiętny	System dwójkowy	System ósemkowy	System szesnastkowy
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	1 0000	20	10

- Cechą charakterystyczną systemów pozycyjnych jest to, że wartość cyfry uzależniona jest od jej . . . pozycji . .

Przykład

Liczby w systemie dziesiętnym (system o podstawie 10):

$353_{(10)} = 3 * 100 + 5 * 10 + 3 * 1 = 3 * 10^2 + 5 * 10^1 + 3 * 10^0$

$2{,}42_{(10)} = 2 * 1 + 4 * 0{,}1 + 2*0{,}01 = 2 * 10^0 + 4 * 10^{-1} + 2 * 10^{-2}$

$$W = \sum_i c_i \cdot 10^i$$

Przykład

Liczba w systemie o podstawie *N*:

$12AB{,}CD_{(N)} = 1*N^3 + 2*N^2 + A*N^1 + B*N^0 + C*N^{-1} + D*N^{-2}$

$$W = \sum_i c_i \cdot N^i$$

- Zamiana całkowitej liczby dziesiętnej na liczbę w systemie liczbowym o podstawie *N* – w dzieleniu całkowitym należy:
  1. podzielić liczbę dziesiętną przez *N*,
  2. zapisać resztę,
  3. z otrzymanym ilorazem przejść do kroku 1 (podzielić przez *N*).

Operację należy przeprowadzać aż do momentu, kiedy iloraz osiągnie wartość 0. Wtedy odczytane od konca reszty będą liczbą w systemie liczbowym o podstawie *N*.

Przykład

Zamiana liczby 32 na liczbę w systemie o podstawie 3.

liczba lub iloraz z poprzedniego kroku		*N*		*iloraz*	*reszta z dzielenia*
32	:	3	=	10	2
10	:	3	=	3	1
3	:	3	=	1	0
1	:	3	=	0	1

↑ *kierunek odczytania liczby*

$32_{(10)} = 1012_{(3)}$

- Zamiana ułamka dziesiętnego (części ułamkowej liczby) na ułamek w systemie liczbowym o podstawie *N* – należy:
  1. pomnożyć liczbę przez *N*,
  2. zapisać czesc calkowita otrzymanego iloczynu,
  3. z częścią ułamkową otrzymanego iloczynu przejść do kroku 1 (przemnożyć przez *N*).

Operację należy przeprowadzać do momentu, aż iloczyn będzie liczba calkowita (tzn. jego część po przecinku będzie wynosiła 0). Wtedy zapisywane czesci calkowite otrzymanych iloczynów (w kroku 2) przedstawiają ułamek w systemie liczbowym o podstawie *N*.

Przykład

Zamiana liczby dziesiętnej 0,6875 na ułamek w systemie liczbowym o podstawie 2.

ułamek lub część „po przecinku” z poprzedniego kroku		*N*		*iloczyn*	*część całkowita iloczynu*
0,6875	*	2	=	1,375	.1
0,375	*	2	=	0,75	.0
0,75	*	2	=	1,5	.1
0,5	*	2	=	1,0	.1

kierunek odczytania liczby ↓

$0{,}6875_{(10)} = .0{,}1011\ldots_{(2)}$

1.2. System binarny

- Wszelkie informacje przetwarzane, przechowywane oraz przesyłane w systemach komputerowych mają postać . binarną . . , czyli są zapisane w **systemie dwójkowym** (systemie liczbowym **o podstawie 2**).
- System binarny jest . pozycyjny . . . . systemem liczbowym

Przykład

Liczba w systemie dwójkowy (system o podstawie 2):

$101{,}11_{(2)} = 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} = 1*4 + 0*2 + 1*1 + 1*0{,}5 + 1*0{,}25 = 5{,}75$

$$W = \sum_i c_i \cdot 2^i$$

1.2.1. Zamiana liczb dziesiętnych na system binarny

$29_{(10)} = ?_{(2)}$

liczba lub iloraz z poprzedniego kroku		*N*		*iloraz*	*reszta z dzielenia*
29	:	2	=	14	1.
14	:	2	=	7	0.
7	:	2	=	3	1.
3	:	2	=	1	1.
1	:	2	=	0	1.

↑

$29_{(10)} = . . 11011_{(2)}$

$0{,}35_{(10)} = ?_{(2)}$

ułamek lub część „po przecinku” z poprzedniego kroku		*N*		*iloczyn*	*część całkowita iloczynu*
0,35	*	2	=	0,7	.0
0,7	*	2	=	1,4	.1
0,4	*	2	=	0,8	0.
0,8	*	2	=	1,6	1.
0,6	*	2	=	1,2	1.
0,2	*	2	=	0,4	0.
0,4	*	2	=	0,8	0.
0,8	*	2	=	1,6	1.
...	*	2	=	...	...

↓

$0{,}35_{(10)} = . . ?. . . . .{}_{(2)}$

Wniosek: ułamek dziesiętny o skończonej liczbie cyfr może wymagać ułamka binarnego o . . . . . nieskonczonej liczbie cyfr

1.2.2. Operacje arytmetyczne na liczbach binarnych

- Zasad obowiązujące przy wykonywaniu działań arytmetycznych na liczbach binarnych są podobne jak w systemie dziesiętnym.

- Dużym ułatwieniem w biegłym posługiwaniu się systemem dwójkowym oraz w konwersji pomiędzy systemami jest:

  1. znajomość postaci binarnej liczb dziesiętnych od 0 do 15,
  2. znajomość kolejnych potęg liczby 2 od 2^0 do 2^{16},
  3. świadomość, że w systemie binarnym liczby „okrągłe" (tzn. składające się z jedynki i samych zer) mają wartość 2^n, gdzie *n* jest liczba zer w zapisie,
  4. świadomość, że mnożenie liczby binarnej przez kolejne potęgi liczby 2 (10b, 100b, 1000b, czyli 2, 4, 8 itd.) oznacza dopisanie na końcu odpowiedniej liczby zer, np. 101b*100b = 10100b,
  5. świadomość, że dopisanie do liczby binarnej zera na końcu oznacza zwiększenie jej wartości 2. razy, dopisanie dwóch zer – zwiększenie wartość cztery razy itd. (co wynika z wcześniejszego punktu),
  6. świadomość, że cyfra 1 na końcu liczby binarnej oznacza, że jest ona nieparzysta oraz o jeden większa od liczby z zerem na końcu i pozostałymi cyframi takimi samymi.

Przykład

Ile wynosi wartość liczby 101001b ?

Jeżeli wiemy, że 101b ma wartość 5, to:

- liczba 101000b, jest 8 razy większa od liczby 101b (bo ma 3 zera więcej, a 2*2*2) czyli jej wartość wynosi 40.

- do tej liczby należy dodać 1, bo interesująca nas liczba to 101001b.

Czyli szukana wartość to 41.

1.2.2.1. Dodawanie

Elementarne operacje to:

- 0 + 0 = 0,
- 0 + 1 = 1 + 0 = 1,
- 1 + 1 = $10_{(2)}$ – co przy dodawaniu pozycyjnym (pisemnym) oznacza „0 i 1 dalej",
- 1 + 1 + 1 = $11_{(2)}$ – co przy dodawaniu pozycyjnym (pisemnym) oznacza „1 i 1 dalej".

Przykład

Działanie matematyczne: 7 + 10

```
  0111
+ 1010
------
 10001
```

1.2.2.2. Mnożenie

Elementarne operacje to:

- 0 * 0 = 0,
- 0 * 1 = 1*0 = 0,
- 1 * 1 = 1.

Przykład

Działanie matematyczne: 5 * 3

```
  101
*  11
------
  1111
```

1.2.2.3. Odejmowanie

Elementarne operacje to:

- 0 - 0 = 0,
- 1 - 0 = 1,
- 1 – 1 = 0,
- 0 - 1 = 1 – co oznacza, że wystąpiła pozyczka z poprzedniej pozycji. Uwaga: Z poprzedniej pozycji pozyczamy zawsze 1, ale to „1” przechodząc na następną pozycję ma wartość 10b (ponieważ system jest pozycyjny), co oznacza że odejmowanie wygląda tak: „10b – 1 = 1”.

Przykład

Działanie matematyczne: 9 - 3

```
 ,1 10
 1001
-  11
------
  110
```

1.2.2.4. Dzielenie

Przykład

Działanie matematyczne: 14 : 3


```
 100,101...
--------
1110:11
11
---
00100
   11
  ---
    100
     11
    ---
     ...
```

1.3. System szesnastkowy

- Cyfry w systemie szesnastkowym:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Przykładowe – ale często spotykane w zagadnieniach związanych z architekturą komputerów – liczby w systemie szesnastkowym, to:
  - Fh = 15,
  - 10h = 16,
  - 20h = 32,
  - FFh = 255,
  - 100h = 256.
- Zamiana liczby w systemie szesnastkowym na liczbę w systemie dwójkowym:

- dzielimy liczbę binarną na grupy po 4 cyfry - rozpoczynając od pozycji separatora dziesiętnego,
- wartość każdej wydzielonej grupy zapisujemy za pomocą jednej cyfry szesnastkowej.

Przykład

$1111100000_{(2)} = ?_{(16)}$ => 0011 | 1110 | 0000 = $3E0_{(16)}$ => $1111100000_{(2)} = 3E0_{(16)}$

Polecenie: *Przeprowadź ogólny dowód, że taka zamiana liczb (grupowanie po cztery cyfry) pomiędzy systemami jest prawidłowa.*

Uwaga: Do swobodnego posługiwania się systemem binarnym i szesnastkowym konieczna jest dobra znajomość kolejnych potęg liczby 16: od 16^0 do 16^4.

- Przyczyny stosowania systemu szesnastkowego:
  - znacznie większa zwięzłość zapisu w porównaniu z zapisem binarnym (4 cyfry binarne = 1 cyfra szesnastkowa),
  - prosta konwersja pomiędzy systemem binarnym i szesnastkowym,
  - liczba bitów w jednej komórce pamięci jest (zwykle) wielokrotnością liczby 4, co znacznie ułatwia przedstawienie jej zawartości w systemie szesnastkowym.

1.3.1. Operacje arytmetyczne na liczbach szesnastkowych

1.3.1.1. Dodawanie

Przykład

Działanie matematyczne: 17h + 1Ah

```
  17
+ 1A
----
  31
```

1.3.1.2. Mnożenie

Przykład

Działanie matematyczne: 15h * 3h

```
  15
* 31
----
 405
```

Uwaga: Należy zauważyć, że mnożenie liczby przez kolejne potęgi liczby 16 (16, 256, 4096 itd. czyli 10h, 100h, 1000h) można potraktować jako dopisanie na końcu odpowiedniej liczby zer (stosownie do wykładnika potęgi), np. 10*16 = Ah*10h = A0h

1.3.1.3. Odejmowanie

Przykład

Działanie matematyczne: 10Ah - Ch

```
  10A
-   C
------
   FE
```

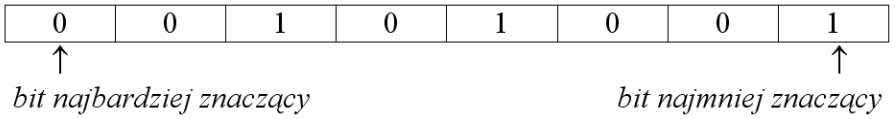
1.4. Reprezentacja liczb całkowitych

1.4.1. Liczby całkowite bez znaku

- Na kolejnych bitach przechowywane są poszczególne cyfry binarne.

Przykład

Liczba $41_{(10)}$ w komórce ośmiobitowej:



1.4.2. Liczby całkowite ze znakiem

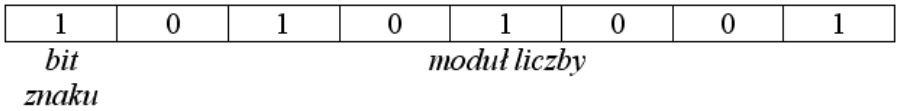
1.4.2.1. Reprezentacja bezpośrednia (kod prosty)

- Jest to prosta reprezentacja liczb całkowitych ze znakiem (reprezentacja typu znak-moduł, reprezentacja bezpośrednia, kod prosty)

- Do kodowanie informacji o znaku służy najbardziej znaczący bit:
  - wartość 0 oznacza liczbę dodatnia,
  - wartość 1 oznacza liczbę ujemna.

Przykład

Liczba $-41_{(10)}$ w komórce ośmiobitowej:



- Wady reprezentacji znak-moduł:
  - możliwe są dwa sposoby reprezentacji zera: 10000000 lub 00000000
  - występują problemy przy realizacji obliczeń na liczbach ujemnych

Przykład

Działanie matematyczne: -2 + 3

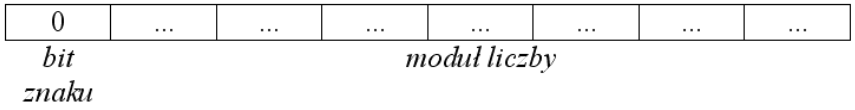
```
  10000010 (-2)
+ 00000011 (+3)
---------------
  10000101 (-5)
```

1.4.2.2. Kod uzupełnieniowy

- Kod uzupełnieniowy służy do reprezentacji liczb całkowitych (także nieujemnych).

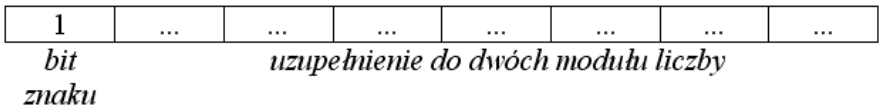
- Liczby **całkowite <u>nieujemne</u>** przyjmują postać (kod prosty):

bit znaku, moduł liczby



- Liczby **całkowite <u>ujemne</u>** przyjmują postać bitu znaku równego 1 i uzupełnia do dwoch modulu liczby:

bit znaku, uzupełnienie do dwóch modułu liczby



- **Uzupełnieniem do dwóch** (U2) dodatniej liczby binarnej *b* zapisanej na *N* bitach jest wartość *u* wyrażona wzorem:

$$u = 2^N - b$$

Przykład

Ile wynosi uzupełnienie do dwóch liczby 3 zapisane na czterech bitach?

$b = 0011_{(2)} = 3_{(10)}$ a $N = 4$.

$2^N = 2^4 = 16_{(10)} =$ 10000

$2^N - b = 16_{(10)} - 3_{(10)} = 13_{(10)} =$ 1101

Uzupełnieniem do dwóch liczby $0011_{(2)}$ jest $1101_{(2)}$.

Przykład

Ile wynosi kod uzupełnieniowy liczby -41$_{(10)}$ zapisany w komórce 8-bitowej?

Należy zapisać bit znaku (1, bo liczba jest ujemna) oraz na 7 bitach uzupełnienia do dwóch liczby 41:

1	...	...	...	...	...	...	...
bit znaku	*zapisane na 7 bitach uzupełnienie do dwóch liczby 41*						

$2^7 = 128$

$2^7 - 41 = 128 - 41 = 87_{(10)} = 1010111_{(2)}$

czyli:

1	1	0	1	0	1	1	1
bit	*zapisane na 7 bitach uzupełnienie do dwóch liczby 41*						

- Zapis **liczby nieujemnej** w kodzie prostym i uzupełnieniowym jest identyczny .

- Konwersja liczby ujemnej z zapisu w kodzie prostym na kod uzupełnieniowy (lub z kodu uzupełnieniowego na prosty):
  - negacja wszystkich bitów z wyjątkiem bitu znaku,
  - do otrzymanej wartości należy dodać (binarnie) liczbę jeden.

Przykład

Reprezentacja w kodzie uzupełnieniowym w komórce 8-bitowej liczby -41$_{(10)}$.

```
10101001(2)  kod prosty
 11010110(2)  negacja wartości (bez bitu
  znaku)

11010110(2)
00000001(2)  dodanie jedynki
---------
 11010111(2)   suma (wartość w kodzie
  uzupełnieniowym)
```

- Zamiana wartości z kodu uzupełnieniowego na system dziesiętny:
  - poszczególnym pozycjom przypisywane są współczynniki będące kolejnymi potęgami liczby 2, przy czym współczynnik odpowiadający pozycji wysuniętej najbardziej w lewo (bit znaku) uwzględniany jest ze znakiem minus,
  - przy konwersji na system dziesiętny sumowane są współczynniki znajdujące się na pozycjach jedynek w rozpatrywanej liczbie binarnej.

Pozycja (i)	7	6	5	4	3	2	1	0
Współczynnik	$-(2^7)$	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	-128	64	32	16	8	4	2	1

Uwaga: Zapisywanie na ośmiu bitach w kodzie uzupełnieniowym liczb ujemnych sprowadza się do znalezienia odpowiedzi na pytanie: *jaką liczbę*

należy dodać do –128, żeby otrzymać wymaganą liczbę ujemną? Ta dodawana liczba, to właśnie uzupełnienie do dwóch zapisane na siedmiu bitach.

Przykład

Ile wynosi wartość w systemie dziesiętnym liczby wyrażonej w kodzie uzupełnieniowym jako 10000011 ?

-128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

Wartość ta wynosi –125, ponieważ -128*1 + 2*1 + 1*1 = -128 + 2 + 1 = -125

Przykład

Ile wynosi wartość w systemie dziesiętnym liczby wyrażonej w kodzie uzupełnieniowym jako 10000000 ?

-128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	0

Jest to –128, ponieważ: -128*1 + 0 = -128

- Kod uzupełnieniowy:
  - trudny do stosowany dla człowieka,
  - ułatwia wykonywanie działań na liczbach całkowitych ze znakiem - w trakcie obliczeń bit znaku jest traktowany w taki sam sposob jak pozostałe bity.

Przykład

Działanie matematyczne: -2 + 3

```
-2
10000010 (kod prosty)
11111101 (negacja, bez bitu znaku)

11111101
00000001 (dodanie jedynki)
--------
11111110 (kod uzupełnieniowy liczby -2)


+3
00000011 (kod prosty)
00000011 (kod uzupełnieniowy)


-2 + 3
  11111110   (-2, kod uzupełnieniowy)
+ 00000011   (+3, kod uzupełnieniowy)
----------
1 00000001   (wynik, kod uzupełnieniowy,
              bit  przeniesienia jest ignorowany)

  00000001   (wynik, kod prosty)
```

1.5. Cechy maszynowej reprezentacji liczb całkowitych

- W systemach komputerowych mogą być reprezentowane wartości całkowite z pewnego zakresu - uzależnionego od:
  - liczby bitow przeznaczonych na przechowywanie jednej wartości numerycznej,
  - przyjętego sposobu reprezentacji.

- Wartości całkowite mieszczące się w dopuszczalnym zakresie przechowywane są w sposób dokładny.
- Wyniki obliczeń realizowanych na liczbach całkowitych są dokładne (pod warunkiem, że mieszczą się na przyjętej liczbie bitów).
- Podstawowym problemem mogącym pojawić się w trakcie obliczeń na liczbach całkowitych jest błąd nadmiaru (overflow) - powstaje wtedy, gdy wynik nie mieści się na przyjętej liczbie bitów.

Przykład

Działanie matematyczne: 87 + 56

```
+87
01010111 (kod prosty)
01010111 (kod uzupełnieniowy)


+57
00111001 (kod prosty)
00111001 (kod uzupełnieniowy)


  01010111 (+87)
+ 00111001 (+57)
----------
  10010000 (wynik?)
           kod uzupełnieniowy liczby: -112,
           kod prosty liczby: -16)
```


1.6. Reprezentacja liczb rzeczywistych

Do reprezentacji wartości rzeczywistych stosuje się:

- reprezentację stałopozycyjna (stałoprzecinkową),
- reprezentację zmiennopozycyjna (zmiennoprzecinkową, półlogarytmiczną).

1.6.1. Reprezentacja stałopozycyjna

W reprezentacji stałopozycyjnej:

- do przechowania jednej wartości rzeczywistej wykorzystuje się N bitów,
- bit najbardziej wysunięty w lewo przechowuje informację o znaku liczby (0 - liczba dodatnia, 1 - liczba ujemna),
- liczba bitów służących do przechowania części całkowitej i części ułamkowej jest stała (stała jest pozycja separatora dziesiętnego - stąd określenie stałopozycyjny).

Przykład

Przykładowa reprezentacja stałopozycyjna do przechowania wartości rzeczywistej na 32 bitach.

znak (1 bit)	*część całkowita (23 bity)*	*część ułamkowa (8 bitów)*

Cechy reprezentacji stałopozycyjnej:

- w systemie komputerowym mogą być przechowywane wartości rzeczywiste z pewnego zakresu,

- komputerowa reprezentacja liczb rzeczywistych ma charakter ...dyskretny... (a nie ...ciagly...), np.

  $0.00000000_{(2)} = 0_{(10)}$

  $0.00000001_{(2)} = 1*2^{-8} = 1/256 = 0{,}00390625$

  $0.00000010_{(2)} = 1 * 2^{-7} = 1/128 = 0{,}0078125$

  $0.00000011_{(2)} = 1 * 2^{-7} + 1 * 2^{-8} = 0{,}01171875$

  .....

  więc np. liczba $0{,}005_{(10)} = \underline{0{,}00000001}01..._{(2)}$

  Wniosek: Liczby rzeczywiste ...nie sa... przechowywane w sposób ...dokladny.....

- pamięć przeznaczona do przechowywania wartości rzeczywistej nie jest wykorzystywana w sposób ...optymalny..., np.

  Wartość $0{,}0000011111_{(2)}$ przechowywana jest jako:

  0 00000000000000000000000 00000111

  Wniosek: Pominięte zostały dwie ostatnie cyfry, a część przeznaczona na przechowywanie części całkowitej nie jest wykorzystana!

- może wystąpić błąd ...nadmiaru..., np.

  Liczba złożona z „1" i dwudziestu pięciu „0" nie może być przechowywana.

1.6.2. Reprezentacja zmiennopozycyjna

W reprezentacji zmiennopozycyjnej (zmiennoprzecinkowa, półlogarytmiczna):

- każda liczba rzeczywista X może być przedstawiona w postaci:

  $$X = \text{...mantysa...} * 2^{\text{cecha}}$$

  gdzie:

  mantysa jest wartością ...rzeczywista... (dodatnią lub ujemną),

  cecha jest wartością ...całkowitą... (dodatnią lub ujemną).

- w systemie komputerowym przechowywana jest mantysa oraz cecha liczby.

Przykład

Przykładowa reprezentacja zmiennopozycyjna zapisana na 16 bitach (w rzeczywistych systemach wykorzystuje się zwykle większą liczbę bitów np. 32 lub 64):

znak mantysy (1 bit)	*mantysa (7 bitów)*	*znak cechy (1 bit)*	*cecha (7 bitów)*

Przykładowo, liczba rzeczywista $7_{(10)}$ ($111_{(2)}$) może być zapisana jako:

$$111_{(2)} = 111 * 2^0 = 11{,}1 * 2^1 = 1{,}11 * 2^{10} = 0{,}111 * 2^{11}$$

Uwaga: w powyższym zapisie mantysa i cecha są zapisane w systemie dwójkowym.

- Zawsze wybiera się taki sposób reprezentacji, w którym mantysa (w zapisie binarnym) zaczyna się od:

  - 0,1… - dzięki temu nie ma potrzeby zapamiętywania znaków „0,".

lub

- 1,... (w tym przypadku w notacji pomija się część całkowitą - pamiętając jednak przy rozkodowywaniu, że ona istnieje), dzięki temu nie ma potrzeby zapamiętywania znaków „1,” oraz oszczędza się jeden bit na zapisywaniu wartości mantysy. Prezentacja spełniająca ten warunek określana jest jako znormalizowana.

Ostatecznie, liczba rzeczywista 7 (mantysa 0,111 oraz cecha 11) przechowywana jest w postaci:

0 1110000 00000011

Cechy prezentacji zmiennopozycyjnej:

- zbiór reprezentowanych wartości jest:
  - ograniczony,
  - dyskretny,
- liczby rzeczywiste nie są przechowywane w sposób dokładny - szczególnie niebezpieczne jest kumulowanie się błędów w trakcie obliczeń,
- reprezentacja zmiennopozycyjna pozwala zwykle na lepsze wykorzystanie pamięci niż reprezentacja stałopozycyjna.

Przykład

W językach C++ / Java mamy typy danych zmiennoprzecinkowe:

- *float* - zapisywany na 32 bitach (cecha 8 bitów, mantysa 23 bity). Zakres wartości od $1.175494351*10^{-38}$ do $3.402823466*10^{38}$. Dokładność 6 - 7 cyfr po przecinku. Nazywany jest typem o pojedynczej precyzji.

- *double* - zapisywany na 64 bitach (cecha 11 bitów, mantysa 52 bity). Zakres wartości od $2.2250738585072014*10^{-308}$ do $1.7976931348623158*10^{308}$, jego dokładność to 15 - 16 cyfr. Nazywany jest typem o podwojnej precyzji.

Wybór pomiędzy reprezentacją stało- i zmiennopozycyjną

- Reprezentacja zmiennopozycyjna:
  - możliwość reprezentowania wartości z zwiekszonego zakresu.
- Reprezentacja stałopozycyjna:
  - wartości reprezentowane są z taką samą dokładnoscia (co jest istotne np. w obliczeniach finansowych).

1.7. Reprezentacja znaków alfanumerycznych

1.7.1. Kod ASCII

- W systemach komputerowych znak alfanumeryczny przechowywany jest w postaci binarnej, jako tzw. kod znaku (będący liczbą całkowitą).
- Najpopularniejszym zestawem kodów jest ASCII - American Standard Code for Information Interchange.
  - w wersji podstawowej - 7 bitowy (128 znaków),
  - w wersji rozszerzonej - 8 bitowy (256 znaków).

- Litery, cyfry znaki interpunkcyjne oraz symbole noszą miano znakow drukowalnych i mają przypisane kody od 32 do 126:

kod (dec)	kod (hex)	znak	kod (dec)	kod (hex)	znak	kod (dec)	kod (hex)	znak	kod (dec)	kod (hex)	znak	kod (dec)	kod (hex)	znak
32	20	spacja	51	33	3	70	46	F	89	59	Y	108	6C	l
33	21	!	52	34	4	71	47	G	90	5A	Z	109	6D	m
34	22	"	53	35	5	72	48	H	91	5B	[	110	6E	n
35	23	#	54	36	6	73	49	I	92	5C	\	111	6F	o
36	24	$	55	37	7	74	4A	J	93	5D	]	112	70	p
37	25	%	56	38	8	75	4B	K	94	5E	^	113	71	q
38	26	&	57	39	9	76	4C	L	95	5F	_	114	72	r
39	27	'	58	3A	:	77	4D	M	96	60	`	115	73	s
40	28	(	59	3B	;	78	4E	N	97	61	a	116	74	t
41	29	)	60	3C	<	79	4F	O	98	62	b	117	75	u
42	2A	*	61	3D	=	80	50	P	99	63	c	118	76	v
43	2B	+	62	3E	>	81	51	Q	100	64	d	119	77	w
44	2C	,	63	3F	?	82	52	R	101	65	e	120	78	x
45	2D	-	64	40	@	83	53	S	102	66	f	121	79	y
46	2E	.	65	41	A	84	54	T	103	67	g	122	7A	z
47	2F	/	66	42	B	85	55	U	104	68	h	123	7B	{
48	30	0	67	43	C	86	56	V	105	69	i	124	7C	|
49	31	1	68	44	D	87	57	W	106	6A	j	125	7D	}
50	32	2	69	45	E	88	58	X	107	6B	k	126	7E	~

- Pozostałe 33 kody (od 0 do 31 i 127) to tzw. kody sterujace, które służą do sterowania urządzeniami.

kod (dec)	kod (hex)	znak	skrót	kod (dec)	kod (hex)	znak	skrót
0	0	Null	NUL	17	11	Device Control 1 (XON)	DC1
1	1	Start Of Heading	SOH	18	12	Device Control 2	DC2
2	2	Start of Text	STX	19	13	Device Control 3 (XOFF)	DC3
3	3	End of Text	ETX	20	14	Device Control 4	DC4
4	4	End of Transmission	EOT	21	15	Negative Acknowledge	NAK
5	5	Enquiry	ENQ	22	16	Synchronous Idle	SYN
6	6	Acknowledge	ACK	23	17	End of Transmission Block	ETB
7	7	Bell	BEL	24	18	Cancel	CAN
8	8	Backspace	BS	25	19	End of Medium	EM
9	9	Horizontal Tab	HT	26	1A	Substitute	SUB
10	0A	Line Feed	LF	27	1B	Escape	ESC
11	0B	Vertical Tab	VT	28	1C	File Separator	FS
12	0C	Form Feed	FF	29	1D	Group Separator	GS
13	0D	Carriage Return	CR	30	1E	Record Separator	RS
14	0E	Shift Out	SO	31	1F	Unit Separator	US
15	0F	Shift In	SI	127	7F	Delete	DEL
16	10	Data Link Escape	DLE				

- Kod ASCII doczekał się wielu rozszerzen (tzw. stron kodowanych) z powodu konieczności przypisania kodów dla występujących w wielu językach liter ze znakami diakrytycznymi (np. ą, ś, ć w polskim alfabecie) oraz dla innych zestawów znaków (np. cyrylicy). Dlatego wykorzystano ósmy bit i w ten sposób uzyskano dodatkowe wartości kodów (od 128 do 255)

- Przykładowo rozszerzenie Windows-1250 (CP-1250) to zestaw kodów używany przez systemy MS Windows do reprezentacji alfabetów języków środkowoeuropejskich, w których stosowany jest alfabet łacińskie (m.in. język chorwacki, czeski, polski, rumuński, słowacki, słoweński, węgierski).

kod (dec)	znak	kod (dec)	znak	kod (dec)	znak	kod (dec)	znak	kod (dec)	znak	kod (dec)	znak	kod (dec)	znak	kod (dec)	znak
128	€	144	NZ	160	NBSP	176	°	192	Ŕ	208	Đ	224	ŕ	240	đ
129	NZ	145	‘	161	ˇ	177	±	193	Á	209	Ń	225	á	241	ń
130	‚	146	’	162	˘	178	˛	194	Â	210	Ň	226	â	242	ň
131	NZ	147	“	163	Ł	179	ł	195	Ă	211	Ő	227	ă	243	ó
132	„	148	”	164	¤	180	´	196	Ä	212	Ô	228	ä	244	ô
133	…	149	•	165	Ą	181	µ	197	Ĺ	213	Ő	229	Í	245	ő
134	†	150	–	166	¦	182	¶	198	Ć	214	Ö	230	ć	246	ö
135	‡	151	–	167	§	183	·	199	Ç	215	×	231	ç	247	÷
136	NZ	152	NZ	168	¨	184	¸	200	Č	216	Ř	232	č	248	ř
137	‰	153	™	169	©	185	ą	201	É	217	Ů	233	é	249	ů
138	Š	154	š	170	Ş	186	ş	202	Ę	218	Ú	234	ę	250	ú
139	‹	155	›	171	«	187	»	203	Ë	219	Ű	235	ë	251	ű
140	Ś	156	ś	172	¬	188	Ľ	204	Ě	220	Ü	236	ě	252	ü
141	Ť	157	ť	173	SHY	189	˝	205	Í	221	Ý	237	í	253	ý
142	Ž	158	ž	174	®	190	ľ	206	Î	222	Ţ	238	î	254	ţ
143	Ź	159	ź	175	Ż	191	ż	207	Ď	223	ß	239	ď	255	˙
133	…	149	•	165	Ą	181	µ	197	Ĺ	213	Ő	229	Í	245	ő

NBSP – spacja niełamliwa (Non-Breaking SPace), SHY (Soft HYphen) – miękki łącznik, NZ – znak niezdefiniowany w kodowaniu.

Kod ASCII – problem polskich znaków

- W przeszłości opracowano różne sposoby kodowania polskich znaków (np. CP-852 /Latin II/, Mazovia, DHN, CSK, Cyfromat) - co utrudniało wymianę dokumentów tekstowych pomiędzy różnymi systemami,
- Obecnie podstawowe sposoby kodowania polskich znaków w kodzie ASCII to:
  - MS Windows CP . 1250 .. . (Windows Latin-2, Windows-. . 1250 . .) - sposób kodowania wprowadzony przez firmę Microsoft wraz z systemem Windows 3.11 PL,

  - ISO Latin-2 (. ISO-8859-2 .. .., Polska Norma PN-93 T-42118) - sposób kodowania określony przez ISO, stosowany powszechnie w Internecie.
- Brak zgodności pomiędzy ww. sposobami kodowania:

kod (dec)	kod (hex)	CP-1250	ISO 8859-2
140	8C	Ś	NZ
156	9C	ś	NZ
161	A1	ˇ	Ą
165	A5	Ą	Ľ
166	A6	¦	Ś
172	AC	¬	Ź
177	B1	±	ą
182	B6	¶	ś
185	B9	ą	š
188	BC	Ľ	ź

1.7.2. Unicode

- Unikod (ang. Unicode lub UCS - . Universal Character Set .) – sposób kodowania znaków uwzględniający większości wykorzystywanych znaków (w zamierzeniu wszystkie znaki pism używanych na świecie).
- Znaki uwzględnione w Unikodzie podzielone zostały na:
  - podstawowy zestaw znaków (określany jako *Basic Multilingual Plane* - BMP lub Plane 0) – dla tych znaków stosowane są kody . 16 . bitowe,
  - dodatkowy zestaw znaków – stosowane są kody . 32 bitowe.

Reprezentacja unikodów (UTF)

UTF - . Unicode Transformation Format . – metody przechowywania unikodów w pamięci komputera, np.:

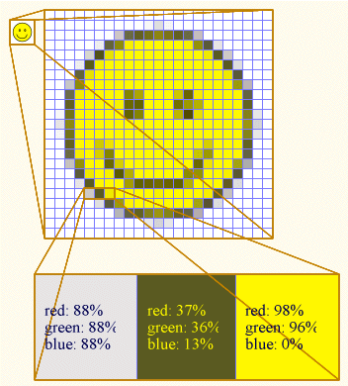
- UTF-8 – kody znaków wchodzących w skład podstawowego zestawu ASCII zapisywane są jako wartości jednobajtowe; pozostałe kody zapisywane są na dwóch, trzech, czterech, pięciu lub sześciu bajtach (znaki o kodach zapisywanych na trzech i większej liczbie bajtów spotykane są we współczesnych językach bardzo rzadko),
- UTF-16 – kody znaków zapisywane są na dwóch, trzech lub czterech bajtach (najczęściej wykorzystywane są znaki o kodach dwubajtowych),
- UTF-32 – kody znaków zapisywane są na 4 bajtach.

1.8. Reprezentacja grafiki

1.8.1. Grafika rastrowa

1.8.1.1. Charakterystyka grafiki rastrowej

Reprezentacja rastrowa (bitmapowa) - zapamiętywane są parametry każdego punktu składającego się na obraz.



Cechy grafiki rastrowej:

- bardzo duże zapotrzebowanie na pamięć (np. obraz formatu A4 zapisany z rozdzielczością 300 punktów na cal i w 24 bitowym kolorze zajmuje ponad 2CMB),
- trudne przekształcanie obrazu (skalowanie, obrót),
- przydatny przy reprezentacji zdjec.

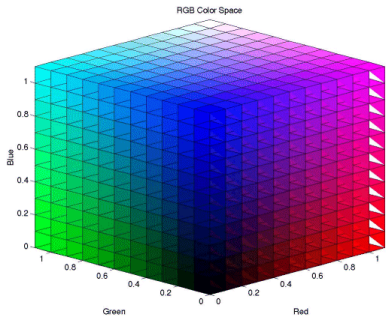
1.8.1.2. Odwzorowanie (głębia) kolorów

- obraz czarno biały - 1 bit,
- 256 kolorów lub 256 odcieni szarości (Grayscale) - 8 bitów,
- High Color czyli 65 tysięcy kolorów - 16 bitow,
- True Color czyli 16 milionów kolorów - 24 bity.

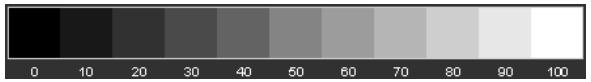
1.8.1.3. Sposób zapisu kolorów

- tryb indeksowy - bitmapa wyposażona jest w tabelę kolorów (palete). Każdy element palety jest kolorem w pewnym formacie (RGB, CMYK itp.). Kolor piksela jest określony indeksem koloru w palecie.

- tryb High Color, True Color - każdy piksel ma przypisany kolor w pewnym formacie (RGB, CMYK itp.),



- tryb Grayscale - każdy piksel ma przypisany odcień szarości.



1.8.1.4. Formaty plików grafiki rastrowej

- formaty niestosujące kompresji (bez kompresji),
- formaty stosujące kompresję bezstratna,
- formaty stosujące kompresję stratna.

1.8.1.4.1. Formaty bez kompresji

- BMP (BitMap):
  - opracowany dla systemu operacyjnego OS/2 w roku 1987,
  - zastosowany jako podstawowy format plików graficznych Windows,
  - 24 bitowa głębia kolorów.

- TIFF (Tagged Image File Format):
  - opracowany w 1986 roku - zapisuje obrazy o dowolnej głębi barw i obsługuje przeźroczystość,
  - wykorzystywany do zapisu różnych obrazów (w faksach, aparaturze medycznej, naświetlarkach),
  - uważany za podstawowy format wymiany plików graficznych w poligrafii,
  - udostępnia wiele rodzajów kompresji (stratnej i bezstratnej), ale najczęściej wykorzystywany jest bez kompresji,
  - pozwala na tworzenie różnorodnych rozszerzeń i zapisywaniu dodatkowych informacji.
- XCF (*eXperimental Computing Facility*):
  - mapa bitowa programu GIMP,
  - może przechowywać wiele warstw.

1.8.1.4.2. Formaty stosujące kompresję bezstratną

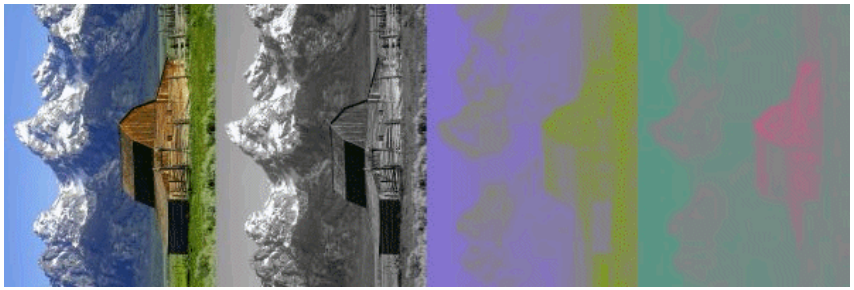
- PCX:
  - opracowany na początku lat 80 (w czasach kart graficznych CGA i Hercules),
  - pozwala na zapis 1, 4, 8 i 24 bitowych obrazów,

- mało wydajny algorytm kompresji stał się powodem wyparcia tego formatu przez format GIF.

- GIF (*Graphics Interchange Format*):
  - opracowany został przez firmę CompuServe w celu rejestrowania grafiki wykorzystującej 256 kolorów,
  - pierwsza wersja - GIF 87 - powstała w 1987 roku, a w 1989 roku GIF 89, który pozwala na zapis przezroczystości, przeplotu i animacji,
  - wykorzystywany jest przy tworzeniu napisów, banerów, rysunków.
- PNG (*Portable Network Graphics*):
  - opracowany w 1996 roku jako alternatywa dla formatu GIF,
  - stale udoskonalany, włączany do nowych wersji edytorów graficznych, ale ciągle mało rozpowszechniony,
  - występuje jako format PNG-8 (indeksowany, o 8-bitowej głębi koloru) i PNG-24 (True Color, o 24-bitowej głębi koloru),
  - jakość obrazów jest lepsza niż takich samych obrazów w formacie GIF,
  - nadaje się do tworzenia jednolitej grafiki na strony WWW (np. banery, przyciski, napisy).
- TIFF - jw.

1.8.1.4.3. Formaty stosujące kompresję stratną

- JPEG (Joint Photographic Expert Group) - rozszerzenia plików: *.jpeg, *.jpg
  - prace nad formatem rozpoczęto w 1986 roku z inicjatywy organizacji ISO oraz CCITT przez zespół ekspertów nazywany Joint Photographic Experts Group. Standard opublikowano w 1991 roku,
  - format przeznaczony głównie do przetwarzania obrazów naturalnych (pejzaży, zdjęć satelitarnych, portretów itp.), czyli takich, które nie mają zbyt wielu ostrych krawędzi i małych detali,
  - algorytm kompresujący oddzielnie zapisuje informacje o jasności (huminancji) i odcieniach barw (chrominancji),



  - stopień kompresji wynosi od 10:1 do 100:1.
- JPEG 2000:
  - opracowany jako uzupełnienie algorytmu kompresji JPEG,
  - lepsza jakość obrazu od JPEG przy tym samym stopniu kompresji,
  - obraz może być również skompresowany bezstratnie (konkurencja dla formatu PNG),

- jego wadą jest duża złożoność obliczeniowa.

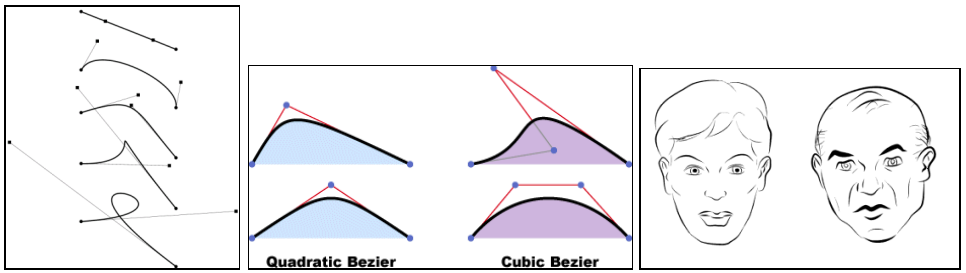
- DjVu - rozszerzenie plików: *.djvu, *.djv
  - format stworzony do przechowywania zeskanowanych dokumentów,
  - opracowany przez naukowców amerykańskiego koncernu AT&T,
  - w porównaniu z formatem JPEG zajmuje od 5 do 10 razy mniej miejsca na dysku dla dokumentów kolorowych oraz od 10 do 20 razy mniej miejsca dla dokumentów czarno-białych; w porównaniu z formatami BMP oraz TIFF zajmuje nawet do 1000 razy mniej miejsca.

1.8.2. Grafika wektorowa

1.8.2.1. Charakterystyka reprezentacji wektorowej

Reprezentacja wektorowa - przechowywany jest matematyczny opis elementów składających się na rysunek. W opisie wykorzystywane są krzywe Beziera lub tzw. prymitywy (proste figury geometryczne, opisane za pomocą odpowiednich parametrów).

Komputer generuje obraz na podstawie takiego opisu (rysując np. koło o określonym promieniu i położeniu). Stąd grafikę wektorową nazywa się również grafiką obiektową, gdyż obraz w tej grafice składa się obiektów o określonych atrybutach.

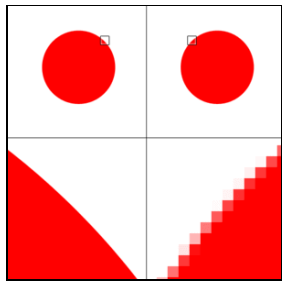


Cechy grafiki wektorowej:

- mniejsze zapotrzebowanie na pamięć w porównaniu z grafiką rastrową,
- łatwiejsze przekształcenie obrazu (skalowanie, obrót).

Możliwa jest zmiana sposobu reprezentacji grafiki poprzez proces:

- rasteryzacji - budowanie mapy bitowej, zwykle na podstawie opisu wektorowego.
- wektoryzacji - przejście do reprezentacji wektorowej.



1.8.2.2. Formaty plików grafiki wektorowej

- SVG (*Scalable Vector Graphics*):
  - stworzony w 1999 r. przez W3C z myślą o zastosowaniu go na stronach WWW,
  - format oparty na języku XML, promowany jako standard grafiki wektorowej nie ograniczany licencjami i patentami,
- Macromedia Flash - rozszerzenia plików: *.swf
  - format tworzenia grafiki wektorowej i animacji,

- działania w oparciu o tzw. metodę klatek kluczowych,
- najpopularniejszy format grafiki wektorowej w Internecie.

- EPS (*Encapsulated PostScript*):
  - format plików, będący podzbiorem języka PostScript,
  - jego głównym przeznaczeniem jest przechowywanie pojedynczych stron (ilustracji),
  - nieformalny standard wymiany obrazów stosowany w DTP.