SYSTEMY LICZBOWE
obliczenia

Systemy liczbowe

System liczenia – sposób tworzenia liczb ze znaków cyfrowych oraz zbiór reguł umożliwiających wykonywanie operacji arytmetycznych na liczbach.

Systemy liczbowe

Systemy pozycyjne

Systemy addytywne (niepozycyjne)

Liczbę tworzy się, sumując poszczególne wartości jej znaków cyfrowych (system rzymski, hieroglificzny, alfabetyczny)

Np. system rzymski:

I (1)	V (5)	X (10)	L (50)
C (100)	D (500)	M (1000)	

np.
XVI = 10+5+1 = 16
XIV = 10-1+5=14 (gdy przed większą jest mniejsza)

Pozycyjne systemy liczbowe

Pozycyjny system liczbowy (ang. *positional number system*) – to sposób zapisywania liczb za pomocą skończonego zbioru znaków (cyfry arabskie, litery alfabetu), w którym wartość liczbowa cyfry zależy od jej umiejscowienia (pozycji) względem pozostałych znaków.

Podstawa systemu pozycyjnego – cecha charakterystyczna systemu pozycyjnego, to liczba która jednocześnie określa liczbę używanych cyfr (znaków). Liczby są zapisywane za pomocą cyfr, które ustawia się na określonych pozycjach – każda z nich ma swoją wagę, równą podstawie podniesionej do potęgi o wartości numeru pozycji.

Wartość liczby uzyskuje się po zsumowaniu poszczególnych iloczynów wag i cyfr pozycji.
Zakładając, że p oznacza podstawę systemu pozycyjnego, to dowolną liczbę l_p, n-cyfrową można zapisać w postaci wielomianu:

$$l_p = \sum_{i=0}^{n-1} a_i * p^i$$

$$a_{n-1}a_{n-2} \ \dots \ a_2a_1a_0 = \ a_{n-1} * p^{n-1} + a_{n-2} * p^{n-2} + \ \dots + \ a_2 * p^2 + a_1 * p^1 + a_0 * p^0$$

a_i – cyfry należące do zbioru {0,1 , …, p-1}
p_i – waga
i – numer pozycji cyfry w ciągu liczbowym
n – liczba cyfr w ciągu

Systemy liczbowe

System **dziesiętny** / decymalny
(oznaczenie liczb 99_{10} /99_D)

System **dwójkowy**/ binarny
(oznaczenie liczb 0101_2 /0101_B)

System **szesnastkowy**/ heksadecymalny
(oznaczenie liczb FF_{16} /FF_H)

System **ósemkowy**/ oktalny
(oznaczenie liczb 77_8 /77_O)

System dziesiętny (decymalny)

$$x = L(A) = \sum_{i=0}^{n-1} a_i 10^i$$

Podstawę stanowi liczba **10**; do zapisu używa się dziesięciu cyfr arabskich – **0,1,2,3,4,5,6,7,8,9**

$$543_D = 5 * 100 + 4 * 10 + 3 * 1$$

setki dziesiątki jedności waga

$$5_2 4_1 3_0 = 5 * 10^2 + 4 * 10^1 + 3 * 10^0$$

cyfra podstawa

Każda cyfra w ciągu została ponumerowana, począwszy od prawej strony. Pozycji jedynek przyporządkowano 0, dziesiątek 1, a setek 2. Następnie każda cyfra z ciągu została pomnożona przez wagę, którą stanowi podstawa 10 podniesiona do potęgi równej pozycji.

System dwójkowy (binarny)

Podstawę stanowi liczba **2**; do zapisu używa się dwóch cyfr arabskich – **0, 1**

$$L(A) = \sum_{i=0}^{n-1} a_i 2^i, \quad a_i \{0,1\}$$

Takie przyporządkowanie nazywa **się naturalnym kodem binarnym NKB**
(ang. Natural Binary Code NBC)

Długość słowa – liczba cyfr w słowie

System szesnastkowy (heksadecymalny)

Podstawę stanowi liczba **16**; do zapisu używa się szesnaście cyfr, dziesięć to arabskie – **0,1,2,3,4,5,6,7,8,9**, pozostałe 6 to pierwsze litery alfabetu łacińskiego: **A (10), B (11), C(12), D(13), E(14), F(15)**.

Konwersja liczby **szesnastkowej** → **dziesiętną**

$$4C5_H = 4_2C_15_0 = 4*16^2 + C*16^1 + 5*16^0 = 4*256 + 12*16 + 5*1 = 1024 + 192 + 5 = 1221_D$$

Kolejne cyfry w liczbie heksadecymalnej należy ponumerować, począwszy od pierwszej (0) z prawej strony. Następnie każdą cyfrę mnoży się przez wagę otrzymaną z podstawy (16) podniesionej do potęgi równej pozycji. Po przemnożeniu cyfr przez wagi (litery należy zamienić na odpowiedniki dziesiętne) wykonujemy sumowanie.

Konwersja liczby **dziesiętnej** → **heksadecymalną** (cykliczne dzielenie z resztą)

$$\begin{array}{rl} 1221:16 = 76 & r = 5 \\ 76:16 = 4 & r = 12(C) \\ 4:16 = 0 & r = 4 \end{array} \Big\uparrow$$

$$1221_D = 4C5_H$$

dr inż. Andrzej Dulbiński

System ósemkowy (oktalny)

Podstawę stanowi liczba **8**; do zapisu używa się dziesięciu cyfr arabskich – **0,1,2,3,4,5,6,7**

Konwersja liczby **ósemkowej** → **dziesiętną**

$$355_O = 3_2 5_1 5_0 = 3 * 8^2 + 5 * 8^1 + 5 * 8^0 = 3 * 64 + 5 * 8 + 5 =$$
$$192 + 40 + 5 = 237_D$$

Konwersja liczby **dziesiętnej** → **ósemkową** (cykliczne dzielenie z resztą)

$$237 : 8 = 29 \qquad r = 5$$
$$29 : 8 = 3 \qquad r = 5$$
$$3 : 8 = 0 \qquad r = 3$$

$$237_D = 355_O$$

Konwersja liczby dziesiętnej na binarną i odwrotnie cz.1

$$10101_B = 1_4 0_3 1_2 0_1 1_0 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 =$$
$$1 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 1 * 1 = 21_D$$

LSB
(Least Significant Bit)

(cykliczne dzielenie z resztą)

$$\begin{array}{ll} 25:2 = 12 & r = 1 \\ 12:2 = 6 & r = 0 \\ 6:2 = 3 & r = 0 \\ 3:2 = 1 & r = 1 \\ 1:2 = 0 & r = 1 \end{array}$$

$$25_D = 11001_B$$

MSB
(Most Significant Bit)

Konwersja liczby dziesiętnej na binarną i odwrotnie cz.2

Konwersja liczby **binarnej → dziesiętną**

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
						1	1	0	0	1
						16	8	-	-	1

wartości odpowiadające jedynkom należy zsumować

$$\mathbf{11001} \Rightarrow np. 16 + 8 + 1 = 25$$

Konwersja liczby **dziesiętnej → binarną** (inna metoda)

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	1	0	1	1	0	1	0	0	0	0
	512		128	64		16				
$512 + 128 + 64 + 16 = 720$ $(720)_{10} = 01011010000_2$										

Sumowanie zaczynamy od liczby, która jest mniejsza od liczby przekształconej. Jeśli wybierzemy daną wartość potęgi 2 to stawiamy 1, jeśli ją pomijamy to stawiamy 0.

Działania na liczbach binarnych

dodawanie

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=0$ i 1 dalej

odejmowanie

$0-0=0$
$1-0=1$
$1-1=0$
$0-1=1$ i pożyczka

mnożenie

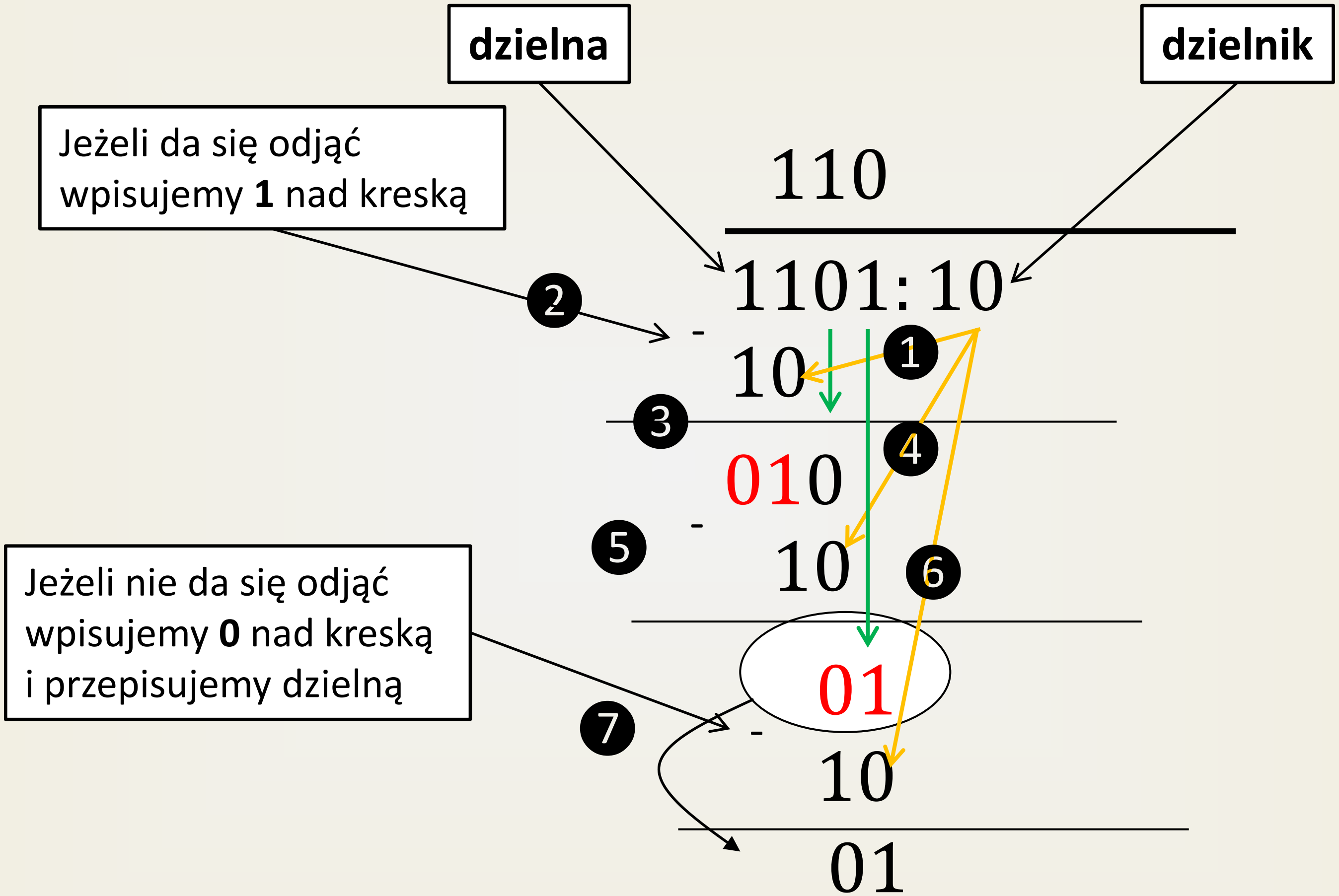
$0*0=0$
$1*0=0$
$0*1=0$
$1*1=1$

$$\begin{array}{r} 1101 \\ +\ 1011 \\ \hline 11000 \end{array}$$

$$\begin{array}{r} 1000 \\ -\ 0011 \\ \hline 0101 \end{array}$$

$$\begin{array}{r} 1010 \\ *\ 0110 \\ \hline 0000 \\ 1010 \\ +\ 1010 \\ 0000 \\ \hline 111100 \end{array}$$

Dzielenie liczb binarnych



Konwersja - system szesnastkowy a binarny

cyfra HEX	cyfra binarna	cyfra HEX	cyfra binarna
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Jeżeli ostatni fragment liczby nie jest pełną czwórką, możemy ją dopełnić do czwórki zerami

7cd5$_H$ = 0111 | 1100 | 1101 | 0101$_B$ = 11111010101$_B$

*00***10** | **1111** | **0101** | **1101** | **0000** | **0101**$_{\mathbf{B}}$ = 2F 5D05$_H$

Konwersja - system ósemkowy a binarny

Aby zamienić liczbę binarną na ósemkową należy podzielić ja na 3 bity, zaczynając od prawej strony. Każde otrzymane 3 cyfry zamieniamy na odpowiadająca mu jedną cyfrę systemu ósemkowego.

System 2	000	001	010	011	100	101	110	111
System 8	0	1	2	3	4	5	6	7

$(1\ 010\ 101\ 000\ 111)_2 = (12507)_8$

1 2 5 0 7

Kod dwójkowo-dziesiętny BCD (Binary Code Decimal)

Kodowanie BCD polega na tym, ze każda cyfra zapisana w systemie dziesiętnym jest przedstawiana za pomocą grupy czterech cyfr binarnych tzw. **tetrad**. Każdej cyfrze przyporządkowuje się na stałe określoną liczbę binarną.

Np. liczbę **89** można przedstawić za pomocą 2 tetrad **1000 1001**

Liczba dziesiętna	Kod NKB	Kod BCD
127	111 1111	001 0010 011

Kodowanie – przypisanie symbolom informacji.

Reprezentacja znak moduł ZM cz.1

Kodowanie znaku jest realizowane poprzez najstarszą cyfrę w liczbie binarnej.
Najstarszą liczbę określa się jako znak $\boldsymbol{a_{n-1}}$, podczas gdy pozostałe cyfry są modułem reprezentującym daną liczbę binarną.

W celu obliczenia wartości naturalnej z liczby binarnej posługujemy się wzorem:

$$\boldsymbol{a_{n-1}}a_{n-2}\ldots a_2a_1a_0 = (1 - 2 * a_{n-1}) \cdot \sum_{i=0}^{n-2} a_i 2^i$$

Jeżeli najstarsza cyfra jest jedynką, to wynik wyrażenia będzie -1; jeżeli zerem, to będzie 1

Liczba dziesiętna	Znak moduł
9	01001
-9	11001

Reprezentacja znak moduł ZM cz.2

$$\boldsymbol{a_{n-1}} a_{n-2} \dots a_2 a_1 a_0 = (1 - 2 * a_{n-1}) * \sum_{i=0}^{n-2} a_i 2^i$$

Konwersja liczby dziesiętnej ZM → binarną

Aby uzyskać liczbę binarną ze znakiem na podstawie liczby dziesiętnej, należy obliczyć moduł metodą dzielenia przez podstawę 2, a następnie dołączyć 0 jeżeli chcemy mieć liczbę dodatnią, lub 1 – ujemną.

Konwersja liczby binarnej → dziesiętna ZM:

$$0111_{ZM} = 0\ 1_2 1_1 1_0 = (1 - 2 * 0) * (1 * 2^2 + 1 * 2^1 + 1 * 2^0) = 1 * (4 + 2 + 1) = 7_D$$
$$1111_{ZM} = 1\ 1_2 1_1 1_0 = (1 - 2 * 1) * (1 * 2^2 + 1 * 2^1 + 1 * 2^0) = -1 * (4 + 2 + 1) = -7_D$$

Zapis **ZM** nie pozwala na zastąpienie odejmowania dodawaniem. Odjemnik > odjemnej → błędne wyniki.
Występują dwa zapisy liczby zero.

+0=00000000
-0=10000000 (ZM)

Reprezentacja uzupełnień do 1 - U1 cz.1

Zapis liczby dodatniej tak jak w ZM
Zapis liczby ujemnej uzyskuje się negując każdy bit reprezentacji binarnej modułu zapisanego w kodzie naturalnym.

Liczba dziesiętna	ZM (znak moduł)	Zapis U1
9	01001	01001
-9	11001	10110

Działania na liczbach w kodzie U1 cz.2

5	0.0101	5	0.0101	9	0.1001
-9	-0.1001	+(-9)	+1.0110	+(-4)	+1.1011
-4	(1)1.1100	-4	1.1011	5	(1)0.0100
korekcja	-1				+1
	1.1011				0.0101
	(100_B)				

Jeżeli w wyniku działania przed bitem znaku pojawi się (1) musimy przeprowadzić korekcję wyniku. Korekcja polega na przesunięciu tej jedynki na najmniej znacząca pozycję i wykonaniu jeszcze raz tego samego działania.

Reprezentacja uzupełnień do 2 - U2 cz.1

Cyfra określająca znak jest zintegrowana z liczbą binarną, co pozwala na działania arytmetyczne.

W celu obliczenia wartości naturalnej z liczby binarnej z wykorzystaniem metody U2 posługujemy się wzorem:

$$a_{n-1}a_{n-2}\dots a_2a_1a_0 = a_{n-1} * (-2^{n-1}) + \sum_{i=0}^{n-2} a_i 2^i$$

Zapis liczby dodatniej tak jak w ZM czy U1
Zapis liczby ujemnej to dopełnienie modułu tej liczby do wartości 2^n , gdzie n jest pozycją bitu określenia znaku. Praktycznie zapis ujemny uzyskuje się negując każdy bit reprezentacji binarnej modułu zapisanego w kodzie naturalnym, a następnie dodając liczbę 1.

Liczba dziesiętna	ZM (znak moduł)	Zapis U1	Zapis U2
9	01001	01001	01001
-9	11001	10110	10111

Działania na liczbach U2 cz.2

5	0.0101	5	0.0101	9	0.1001
-9	-0.1001	+(-9)	+1.0111	+(-4)	+1.1100
-4	1.1100	-4	1.1100	5	0.0101

Wszystkie wyniki otrzymujemy także w kodzie U2.

Zamiana liczby binarnej → dziesiętną

$$0111_B = 0_3 1_2 1_1 1_0 = 0 * (-2^3) + 1 * (2^2) + 1 * (2^1) + 1 * (2^0) = 4 + 2 + 1 = 7_D$$
$$1111_B = 1_3 1_2 1_1 1_0 = 1 * (-2^3) + 1 * (2^2) + 1 * (2^1) + 1 * (2^0) = -8 + 4 + 2 + 1 = -1_D$$

moduł

Bit znaku + negacja modułu

+ 1

$$(-7)_D \rightarrow -(7)_D \xrightarrow{D \rightarrow B} -(0111)_B \xrightarrow{B \rightarrow U1} (1.1000)_{U1} \xrightarrow{B \rightarrow U2} (1.1001)_{U2}$$

Kody liczbowe

W komputerach wykorzystuje się 2 rodzaje kodów

Kody stałopozycyjne

mają ustalone miejsc rozdziału części całkowitej i ułamkowej (przecinka)

Kody zmiennopozycyjne

dokładność reprezentacji zależy od wartości wykładnika

Liczby stałoprzecinkowe

Przekształcenie liczby dziesiętnej → postać binarną

1. Zamiana liczby całkowitej na postać binarną za pomocą cyklicznego dzielenia przez 2.

$\mathbf{10,225}$

$$
\begin{array}{ll}
10:2=5 & r=0 \\
5:2=2 & r=1 \\
2:2=1 & r=0 \\
1:2=0 & r=1
\end{array}\ \uparrow
$$

$$10_D = 1010_B$$

2. Zamiana części ułamkowej na postać binarną za pomocą cyklicznego mnożenia przez 2. Jeżeli wynik jest ≥ 1, to wyznaczony bit części ułamkowej jest także równy 1. Do dalszych obliczeń wykorzystujemy część ułamkową wyniku.

1. $0{,}225 * 2 = 0{,}45$
2. $0{,}45 * 2 = 0{,}9$
3. $0{,}9 * 2 = 1{,}8$
4. $0{,}8 * 2 = 1{,}6$
5. $0{,}6 * 2 = 1{,}2$
6. $0{,}2 * 2 = 0{,}4$
7. $0{,}4 * 2 = 0{,}8$
8. $0{,}8 * 2 = 1{,}6$
9. $0{,}6 * 2 = 1{,}2$
10. $0{,}2 * 2 = 0{,}4$

$$
\begin{array}{l}
\textit{część całkowita } 0 \\
\textit{część całkowita } 0 \\
\textit{część całkowita } 1 \\
\textit{część całkowita } 1 \\
\textit{część całkowita } 1 \\
\textit{część całkowita } 0 \\
\textit{część całkowita } 0 \\
\textit{część całkowita } 0 \\
\textit{część całkowita } 1 \\
\textit{część całkowita } 1 \\
\textit{część całkowita } 0
\end{array}\ \downarrow
$$

$$0{,}225_D = 0{,}0011100110_B$$

$$10{,}225_D = 1010{,}0011100110_B$$

Kody zmiennopozycyjne (zmiennoprzecinkowe) cz.1

Liczby zmiennoprzecinkowe umożliwiają obsługę większego zakresu liczb (bardzo małych lub bardzo dużych), jednak kosztem wolniejszego przetwarzania i mniejszej dokładności.

Termin „zmiennoprzecinkowe” oznacza, iż nie istnieje stała liczba cyfr przed przecinkiem i po przecinku.

Liczba zmiennoprzecinkowa składa się z dwóch części :

- ❑ liczby stałoprzecinkowej – **mantysy, $\boldsymbol{m}$** oraz
- ❑ podstawy **base, $\boldsymbol{b}$** podniesionej do potęgi zwanej cechą lub wykładnikiem (ang. **exponent, $\boldsymbol{e}$**).

$$l_{PF} = m * b^e$$

Zamiana zmiennoprzecinkowej liczby binarnej → postać dziesiętną.

Należy ze słowa kodu wydobyć cyfry cechy i mantysy (np. 1101 1010) - cztery cyfry cechy i cztery mantysy.

cecha				**mantysa**			
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Liczby zmiennoprzecinkowe (zmiennopozycyjne) cz.1

Z zastosowaniem metody uzupełnień do 2 oblicza się wartość cechy

$$e = b_7(-2^3) + b_6 2^2 + b_5 2^1 + b_4 2^0 = (-8)b_7 + 4b_6 + 2b_5 + b_4$$

mantysa jest liczbą stałoprzecinkową z przedziału [1,2], obliczana ze wzoru:

$$m = b_3 b_2, b_1 b_0 = b_3(-2^1) + b_2 2^0 + b_1 2^{-1} + b_0 2^{-2} = -2b_3 + b_2 + \frac{1}{2} b_1 + \frac{1}{4} b_0$$

Otrzymane wartości podstawia się do wzoru: $l_{PF} = m * 2^e$

$$\mathbf{1111\ 1001_{FP}}$$

$$e = 1111_{U2}$$
$$1111_{U2} = -8 + 4 + 2 + 1 = -1_D$$

$$m = 10{,}01_{U2}$$
$$10{,}01_{U2} = -2 + \frac{1}{4} = -1{,}75$$

$$l_{FP} = m * 2^e = -1\frac{3}{4} * 2^{-1} = -\frac{7}{4} * \frac{1}{2} = -0{,}875$$
$$1111\ 1001_{FP} = -0{,}875$$

Liczby zmiennoprzecinkowe (zmiennopozycyjne) cz.2

Konwersje liczby dziesiętnej na postać binarną można dokonać stosując metodę dla liczb stałoprzecinkowych

$\mathbf{13,7}_{D}$

$$\begin{aligned}
13:2 &= 6 \quad r = 1 \\
6:2 &= 3 \quad r = 0 \\
3:2 &= 1 \quad r = 1 \\
1:2 &= 0 \quad r = 1
\end{aligned} \uparrow$$

$13_D = 1101$

$$\begin{aligned}
0{,}7 * 2 &= 1{,}4 \ \ część\ całkowita\ 1 \\
0{,}4 * 2 &= 0{,}8 \ \ część\ całkowita\ 0 \\
0{,}8 * 2 &= 1{,}6\ część\ całkowita\ 1 \\
0{,}6 * 2 &= 1{,}2 \ \ część\ całkowita\ 1
\end{aligned} \downarrow$$

$13{,}7_D = 1101{,}1011_B$

Algebra George`a Boole`a

Algebra Boole`a operuje zmiennymi dwuwartościowymi o wartościach 0,1.

Suma logiczna (alternatywa, dysjunkcja) – jest równa 1, gdy którykolwiek ze składników jest równy 1.

$(a + b \quad a \cup b)$

Iloczyn logiczny (koniunkcja) – jest równy 1, gdy wszystkie czynniki są równe 1

$(a \cdot b; \quad a \cap b)$

Negacja (dopełnienie) – działanie jednoargumentowe, jest równa 1, gdy argument ma wartość 0

(a), a`, ~a, -a, a#, /a

$$0 + 0 = 0 \qquad 0 \cdot 0 = 0$$
$$0 + 1 = 1 \qquad 0 \cdot 1 = 0 \qquad \overline{0} = 1$$
$$1 + 0 = 1 \qquad 1 \cdot 0 = 0 \qquad \overline{1} = 0$$
$$1 + 1 = 1 \qquad 1 \cdot 1 = 1$$

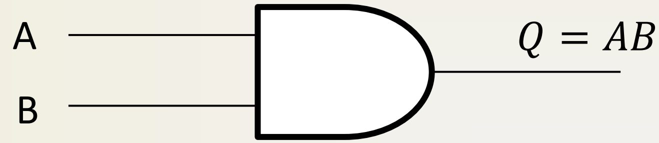
a	b	$a+b$	ab	$\overline{a}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Tożsamości algebry Boole`a

Własności operacji sumy i iloczynu		
Przemienność	$A + B = B + A$	$A \cdot B = B \cdot A$
Łączność	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Rozdzielczość	$A + (B \cdot C) = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$
Tożsamość	$A + 0 = A$	$A \cdot 0 = 0$
	$A + 1 = 1$	$A \cdot 1 = A$
	$A + A = A$	$A \cdot A = A$
Komplementarność	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
Prawo de Morgana	$\overline{A + B} = \overline{A} \cdot \overline{B}$	$\overline{A \cdot B} = \overline{A} + \overline{B}$
Prawo sklejania	$A \cdot \overline{B} + A \cdot B = A$	$\left(A + \overline{B}\right) \cdot (A + B) = A$
Prawo pochłaniania	$A \cdot \overline{B} + B = A + B$	

Bramki logiczne (funktory)

AND (I)

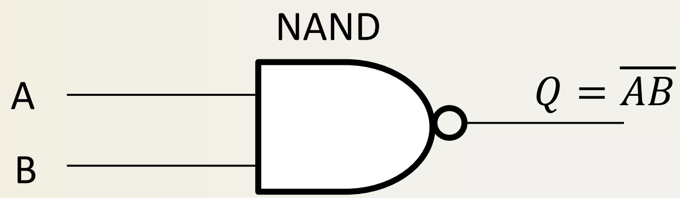


A	B	Q
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

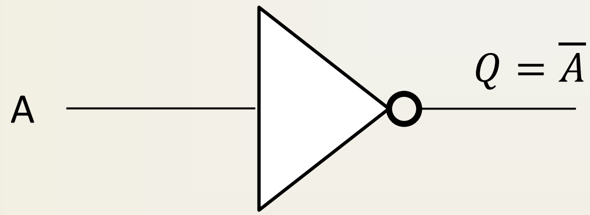
OR (LUB)

A	B	Q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Bramki logiczne (funktory)

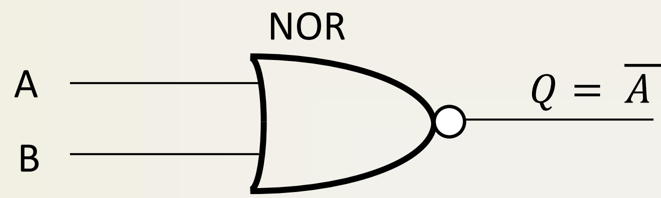


A	B	Q
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**



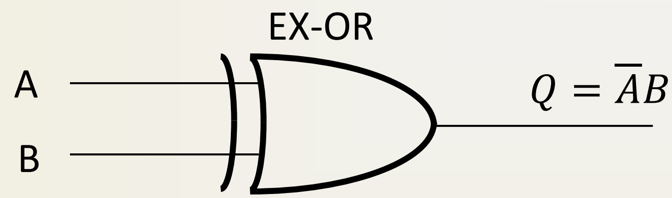
A	Q
0	**1**
1	**0**

Bramki logiczne (funktory)



$$Q = \overline{A + B}$$

A	B	Q
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**



$$Q = \overline{A}B + A\overline{B}$$

A	B	Q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Dziękuję