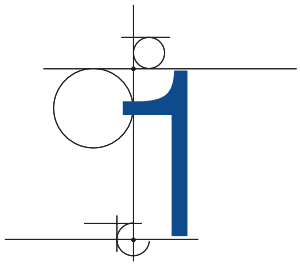


Systemy liczbowe używane w technice komputerowej

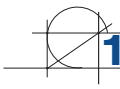
Systemem liczenia nazywa się sposób tworzenia liczb ze znaków cyfrowych oraz zbiór reguł umożliwiających wykonywanie operacji arytmetycznych na liczbach. Podstawą systemów liczenia są systemy liczbowe dzielące się na pozycyjne i addytywne.

W **systemach addytywnych** liczbę tworzy się, sumując poszczególne wartości jej znaków cyfrowych. Do systemów addytywnych zaliczamy systemy: rzymski, hieroglificzny i alfabetyczny.

UWAGA

Cyfry systemu rzymskiego to: I (1), V (5), X (10), L (50), C (100), D (500), M (1000). Liczby są tworzone przez dodawanie poszczególnych cyfr w ciągu, np. XVI = 10+5+1 = 16. Jeżeli przed większą cyfrą pojawia się mniejsza, to przyjmuje ona wartość ujemną, np. XIV = 10–1+5 = 14.

W niniejszym rozdziale zostaną omówione następujące zagadnienia: pozycyjne systemy liczbowe, arytmetyka liczb binarnych, sposoby zapisu liczb binarnych ze znakiem oraz zapis liczb binarnych stało- i zmiennopozycyjnych.



1.1. Pozycyjne systemy liczbowe

Pozycyjny system liczbowy (ang. *positional numeral 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 sąsiednich znaków. System pozycyjny charakteryzuje liczba zwana **podstawą systemu pozycyjnego**, która jednocześnie określa liczbę używanych cyfr (znaków). Liczby są zapisywane za

pomocą cyfr, które ustawia się na określonych pozycjach. Każda pozycja ma swoją wagę równą podstawie podniesionej do potęgi o wartości numeru pozycji. Wartość liczby uzyskujemy po zsumowaniu poszczególnych iloczynów wag i cyfr pozycji.

Załóżmy, że p oznacza podstawę systemu pozycyjnego. Dowolną liczbę l_p n-cyfrową można wówczas zapisać w następującej postaci (wielomianowy zapis liczby):

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

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

gdzie: a_i to cyfry należące do zbioru {0, 1, …, p–1}, p_i — waga, i — numer pozycji cyfry w ciągu liczbowym, n — liczba cyfr w ciągu, * — iloczyn.

Do najpopularniejszych pozycyjnych systemów liczbowych należą:

- system dziesiętny/decymalny (sposób oznaczenia liczb: $99_{10}/99_D$),
- system dwójkowy/binarny (sposób oznaczenia liczb: $0101_2/0101_B$),
- system szesnastkowy/heksadecymalny (sposób oznaczenia liczb: FF_{16}/FF_H),
- system ósemkowy/oktalny (sposób oznaczenia liczb: $77_8/77_O$).

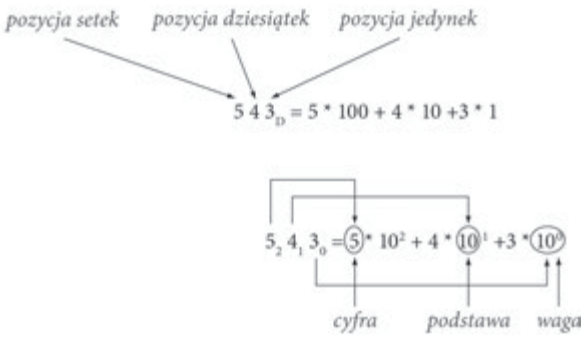
UWAGA

Liczby w poszczególnych systemach są oznaczane za pomocą indeksu dolnego w postaci podstawy lub pierwszej litery nazwy angielskiej.

1.1.1. System dziesiętny (decymalny)

Ludzie posługują się najczęściej pozycyjnym **systemem dziesiętnym** (ang. *decimal* — decymalny), w którym podstawę stanowi liczba 10, a do zapisu liczb używa się dziesięciu cyfr arabskich: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Jeśli spróbujemy rozpisać dowolną liczbę dziesiętną z wykorzystaniem podanego przed chwilą wzoru, uzyskamy zapis wielomianowy:

$p = 10, a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$,



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.

1.1.2. System dwójkowy (binarny)

Cyfrowe urządzenia elektroniczne wykorzystują **dwójkowy** (ang. *binary* — binarny) pozycyjny system liczbowy, w którym podstawą jest liczba 2, a liczby zapisuje się za pomocą dwóch cyfr arabskich: 0, 1. Zapis liczby dwójkowej jest dłuższy niż dziesiętnej, jednak stosowanie tylko dwóch cyfr ułatwia budowanie układów półprzewodnikowych, w których w uproszczeniu np. 1 oznacza przepływ prądu, a 0 — brak przepływu. Trudno jest natomiast zbudować układ elektroniczny, który wydajnie i stabilnie reprezentowałby dziesięć stanów odpowiadających cyfrom: 0, 1, 2, …, 9.

UWAGA

Przykładem praktycznego zastosowania systemu binarnego może być proces wyznaczania adresu sieci lub maski podsieci na podstawie adresu IP w notacji dwójkowej.

Liczba naturalna $\mathbf{l}_B$ w systemie dwójkowym ma postać $a_i \ldots a_1\ a_0$, gdzie a przyjmuje wartość 1 lub 0, np. 1100_B (jeden jeden zero zero, nie tysiąc sto!).

Aby dokonać konwersji liczby dwójkowej na postać dziesiętną, należy użyć zapisu wielomianowego:

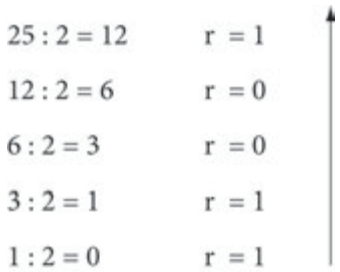
$p = 2,\ a_i \in \{0, 1\},$

$10101_B = 1_4\ 0_3\ 1_2\ 0_1\ 1_0 = 1*2^4+0*2^3+1*2^2+0*2^1+1*20 =$

$= 1*16+0*8+1*4+0*2+1*1 = 16+4+1 = 21_D$

Kolejne cyfry w liczbie binarnej należy ponumerować, począwszy od pierwszej (0) z prawej strony. Następnie każdą cyfrę mnoży się przez wagę otrzymaną z podstawy podniesionej do potęgi równej pozycji. Po przemnożeniu cyfr przez wagi należy je zsumować. Otrzymana liczba dziesiętna jest odpowiednikiem liczby binarnej. Liczba zapisana w systemie dwójkowym jako 10101_B odpowiada 21_D w systemie dziesiętnym.

Aby dokonać zamiany liczby dziesiętnej na postać binarną, należy wykonać cykliczne dzielenie z resztą. Dzielną jest liczba dziesiętna, a dzielnikiem — podstawa systemu binarnego, czyli 2. Wynik z pierwszego dzielenia ponownie jest dzielony przez 2, i tak aż do uzyskania 0. Liczba binarna powstaje na bazie reszt zapisanych w odwrotnej kolejności:



$$25_D = 11001_B$$

Po przekształceniu dziesiętnej liczby 25_D uzyskujemy odpowiednik binarny 11001_B.

W celu szybkiego przekształcania liczb binarnych na postać dziesiętną dobrze jest zapamiętać krotności poszczególnych wag systemu binarnego zamieszczone poniżej.

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**

Dzięki temu w prosty sposób możemy przekształcić liczbę binarną, sumując odpowiedniki dziesiętne wszędzie tam, gdzie w ciągu dwójkowym występują jedynki:

64	32	16	8	4	2	1
1	**1**	**0**	**0**	**1**	**0**	$\mathbf{1_B}$
64	32	–	–	4	–	1

$$64 + 32 + 4 + 1 = 101_D$$

1.1.3. System szesnastkowy (heksadecymalny)

System szesnastkowy (ang. *hexadecimal* — heksadecymalny) najczęściej jest wykorzystywany do uproszczonego zapisu długich liczb binarnych.

UWAGA

Ethernetowe karty sieciowe mają 48-bitowy unikatowy adres sprzętowy zapisany w postaci szesnastkowej, np. 00:50:56:C0:00:08.

Podstawę systemu heksadecymalnego stanowi 16 cyfr. Pierwsze 10 to arabskie cyfry: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, pozostałe 6 to pierwsze litery alfabetu łacińskiego: A, B, C, D, E, F, oznaczające kolejno dziesiętne: 10, 11, 12, 13, 14, 15.

UWAGA

Oprogramowanie do wyszukiwania błędów w skompilowanych plikach binarnych przekształca pierwotny zapis danych dwójkowych na krótszy, szesnastkowy, ułatwiając w ten sposób analizę kodu. W systemie binarnym odpowiednik dziesiętnej liczby 15_D ma aż cztery cyfry: 1111_B, natomiast w szesnastkowym — tylko jedną: F_H.

Liczba naturalna l_H w systemie szesnastkowym ma postać: $a_i \ldots a_1 a_0$, gdzie a przyjmuje wartość 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, np. $1BF_H$.

Chcąc dokonać konwersji liczby szesnastkowej na postać dziesiętną, powinniśmy użyć zapisu wielomianowego:

$p = 16$, $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$,

$4C5_H = 4_2\ C_1\ 5_0 = 4*16^2 + C*16^1 + 5*16^0 = 4*256+12(C)*16+5*1 = 1221_D$

Kolejne cyfry w liczbie heksadecymalnej należy ponumerować, począwszy od pierwszej (0) z prawej strony. Następnie każdą cyfrę mnożymy przez wagę otrzymaną z podstawy (16) podniesionej do potęgi równej pozycji. Po przemnożeniu cyfr przez wagi (litery należy zamienić na odpowiedniki dziesiętne) wykonujemy sumowanie. Otrzymana liczba dziesiętna jest odpowiednikiem liczby szesnastkowej. Liczba zapisana w systemie szesnastkowym jako $4C5_H$ odpowiada 1221_D w systemie dziesiętnym.

Aby dokonać zamiany liczby dziesiętnej na postać szesnastkową, należy wykonać cykliczne dzielenie z resztą. Dzielną jest liczba dziesiętna, natomiast dzielnikiem — podstawa systemu heksadecymalnego, czyli 16. Wynik uzyskany z pierwszego dzielenia ponownie jest dzielony przez 16, i tak aż do uzyskania 0. Liczba szesnastkowa powstaje na bazie reszt zapisanych w odwrotnej kolejności. Wartości powyżej 9 koduje się za pomocą odpowiednich cyfr-liter, np. A:

1221 : 16 = 76	r = 5
76 : 16 = 4	r = 12 (C)
4 : 16 = 0	r = 4

$$1221_D = 4C5_H$$

UWAGA

W celu szybkiego obliczenia reszty z dzielenia, np. 1221:16 = 76,3125, należy pomnożyć część całkowitą wyniku, czyli 76, przez dzielnik 16. Wynik 1216 należy odjąć od dzielnej: 1221–1216, co da resztę 5.

Przy konwersji liczb szesnastkowych na postać binarną i odwrotnie najprościej posłużyć się poniższą tabelą.

Cyfra szesnastkowa	Cyfra dwójkowa	Cyfra szesnastkowa	Cyfra dwójkowa
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

Konwersję liczby binarnej na postać szesnastkową należy rozpocząć od pogrupowania ciągu po cztery cyfry. Grupowanie rozpoczynamy od prawej strony i kontynuujemy aż do uzyskania końca liczby. Jeżeli ostatnie cyfry w pogrupowanej liczbie mają mniej niż cztery znaki, należy uzupełnić puste pozycje zerami:

$$10111101011101000001010_B = 10 \mid 1111 \mid 0101 \mid 1101 \mid 0000 \mid 0101_B =$$

$$= 0010 \mid 1111 \mid 0101 \mid 1101 \mid 0000 \mid 0101_B$$

Następnie, posługując się tabelą, należy wszystkie pogrupowane znaki zamienić na odpowiadające im cyfry heksadecymalne:

0010	1111	0101	1101	0000	0101
2	F	5	D	0	5

Po dokonaniu zamiany powstaje liczba szesnastkowa $2F5D05_H$ — prawda, że proste?

Konwersja z liczby szesnastkowej na binarną jest jeszcze prostsza. Wystarczy na podstawie tabeli zamienić cyfry heksadecymalne na czterocyfrowe ciągi binarne i połączyć je w jedną liczbę (np. dla $A4B9F0_H$):

1010	0100	1011	1001	1111	0000

Otrzymujemy liczbę binarną $101001001011100111110000_B$.

1.1.4. System ósemkowy (oktalny)

System ósemkowy (ang. *octal* — oktalny) jest pozycyjnym systemem liczbowym, w którym podstawą jest liczba 8, a liczby zapisuje się za pomocą ośmiu kolejnych cyfr arabskich: 0, 1, 2, 3, 4, 5, 6, 7. System ten jest rzadko stosowany; zastosowanie można zobaczyć w uniksowym poleceniu `chmod` (*służącym do zmiany uprawnień dostępu do plików i katalogów*).

Liczba naturalna l_O w systemie ósemkowym ma postać: $a_i \ldots a_1\ a_0$, gdzie a przyjmuje wartość 0, 1, 2, 3, 4, 5, 6, 7, np. 212_O.

Konwersję liczb ósemkowych na postać dziesiętną i odwrotnie wykonuje się analogicznie jak w przykładach poświęconych systemom binarnemu i szesnastkowemu.

1.2. Działania na liczbach binarnych

Liczby binarne umożliwiają wykonywanie operacji arytmetycznych (ang. *arithmetic operations on binary numbers*), takich jak suma, różnica, iloczyn i iloraz. Arytmetyką liczb binarnych rządzą pewne zasady, tzw. tabliczki: dodawania, odejmowania, mnożenia i dzielenia.

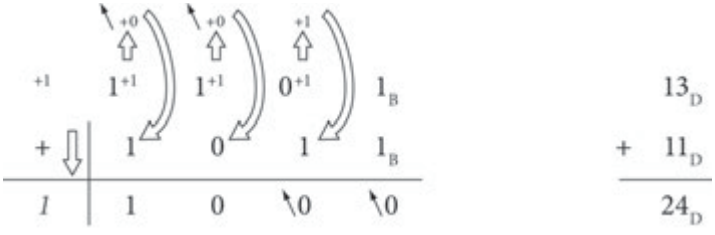
1.2.1. Dodawanie liczb binarnych

Dodawanie liczb binarnych (ang. *addition of binary numbers*) opiera się na prostej tabliczce dodawania, w której reprezentowane są cztery sumy cząstkowe:

0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 0 i ***1 dalej***

Trzy pierwsze sumy nie wymagają komentarza. Czwarta suma, 1+1, daje wynik 0 w bieżącej kolumnie oraz przeniesienie (ang. *carry*) jedynki do następnej kolumny (w lewo), gdzie jest ona dodawana do stojącej tam liczby.

W celu przybliżenia szczegółów dodawania liczb binarnych rozpatrzmy przykład, w którym dodamy liczby binarne 1101_B i 1011_B.



Czarne strzałki oznaczają przeniesienie jedynki do kolumny sąsiedniej, górne strzałki wskazują wyniki sumowania cyfr liczby binarnej oraz przeniesionych jedynek. Strzałki półokrągłe wskazują, że wynik z wcześniejszego obliczenia należy dodać do drugiej liczby w danej kolumnie. Strzałka skierowana w dół oznacza, że jedynka z przeniesienia, która wyszła poza zakres sumowanych liczb, zostaje przepisana do wyniku.

Łatwo zauważyć, że sumowane liczby zawierały po cztery cyfry, wynik natomiast zawiera jedną jedynkę więcej. Tego typu sytuację określamy jako **przepełnienie** (ang. *overflow*).

Przykłady

	0			
+1	1	1	0	1
+	1	0	1	0
1	0	1	1	1

	0	0	0	
+1	1^{+1}	0	1	0
	0	1^{+1}	1	1
+	1	0	1	0
1	1	0	1	1

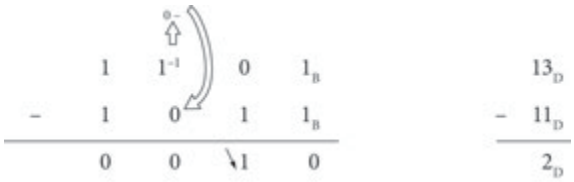
1.2.2. Odejmowanie liczb binarnych

Odejmowanie liczb binarnych (ang. *subtraction of binary numbers*) opiera się na tabliczce odejmowania, w której reprezentowane są cztery różnice cząstkowe:

0–0 = 0
1–0 = 1
1–1 = 0
0–1 = 1 i ***pożyczka***

Ostatnia różnica, 0–1, daje jedynkę oraz wymusza **pożyczkę** (ang. *borrow*) z następnej kolumny.

W celu przybliżenia szczegółów odejmowania liczb binarnych rozpatrzmy przykład, w którym od liczby 1101_B odejmiemy liczbę 1011_B.



Czarna strzałka oznacza pożyczkę jedynki z następnej kolumny. Górna strzałka wskazuje wynik odejmowania pożyczki od cyfry liczby binarnej. Strzałka półokrągła wskazuje, że od wyniku z wcześniejszego obliczenia należy odjąć drugą liczbę w danej kolumnie.

Przykłady

		0	1		
	1	1^{-1}	0^{-1}	0	
–	0	0	1	1	
	1	0	0	1	

	0	1	1		
	1^{-1}	0^{-1}	0^{-1}	0	
–	0	0	1	1	
	0	1	0	1	

Podczas odejmowania naturalnych liczb binarnych może wystąpić zjawisko **niedomiaru** (ang. *underflow*), gdy pożyczka pojawia się poza dostępnym zakresem cyfr. Zjawisko zachodzi, gdy liczba odjemna jest mniejsza niż odjemnik:

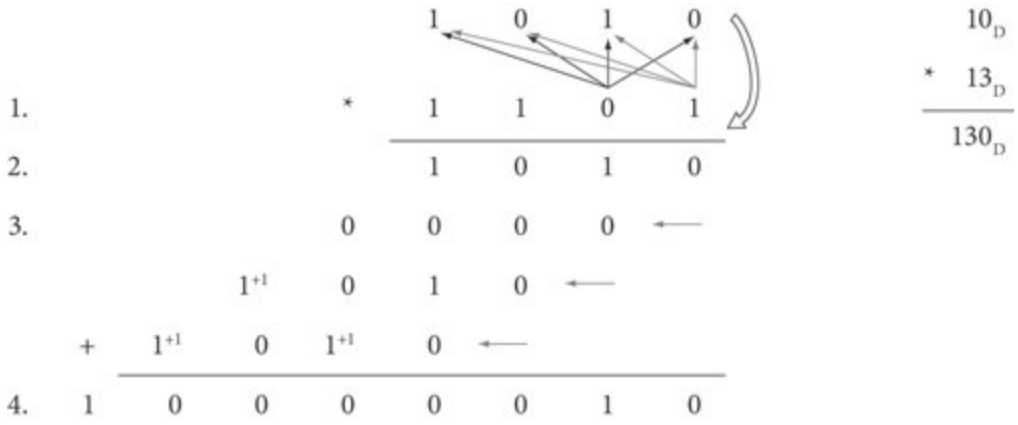
	0	0	0	0	
	0^{-1}	0^{-1}	1^{-1}	1^{-1}	0_B
	–	1	1	1	1_B
...	*1*	1	1	1	1_B

1.2.3. Mnożenie liczb binarnych

Mnożenie liczb binarnych (ang. *multiplication of binary numbers*) opiera się na bardzo prostej tabliczce mnożenia, w której znajdują się cztery iloczyny cząstkowe:

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

Oto przykład, w którym zostały pomnożone dwie liczby binarne: 1010_B i 1101_B.



1. Mnożną mnoży się przez wszystkie kolejne cyfry mnożnika, a uzyskane wyniki wprowadza się, począwszy od aktualnie używanej cyfry mnożnika.
2. Powstaje słupek, w którym każdy kolejny wiersz jest przesunięty o jedną cyfrę w lewo.
3. Zero w mnożniku oznacza, że wszystkie iloczyny również będą miały wynik zerowy, można więc pominąć taki wiersz w późniejszych obliczeniach.
4. Ostatecznie wiersze (powstałe przy przemnażaniu mnożnej przez mnożnik) sumujemy i otrzymujemy wynik.

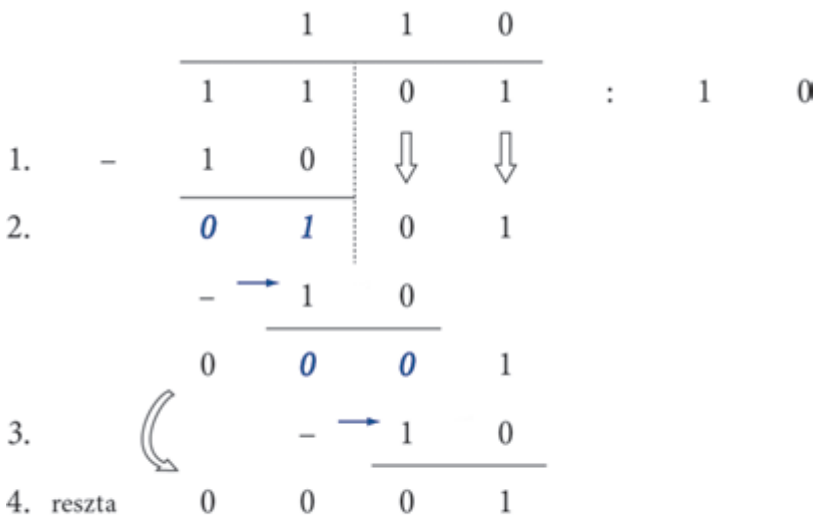
Przykłady

				1	0	1	0
			×	0	1	1	0
				0	0	0	0
			1	0	1	0	
		1	0	1	0		
+	0	0	0	0			
		1	1	1	1	0	0

			1	1	1	1
		×	0	1	0	1
			1+1	1	1	1
+	1+1	1+1	1	1		
1	0	0	1	0	1	1

1.2.4. Dzielenie liczb binarnych

Dzielenie liczb binarnych (ang. *division of binary numbers*) jest teoretycznie najtrudniejszą operacją na tych liczbach. Jedną z metod wykonywania ilorazu liczb binarnych jest cykliczne odejmowanie odpowiednio przesuwanego dzielnika od dzielnej:



$$13_D : 2_D = 6_D \text{ i reszty } 1_D$$

1. Dzielenie zaczyna się od podstawienia dzielnika pod dzielną, począwszy od jej najstarszej cyfry (lewa strona).

   Następnie sprawdza się, czy dzielnik można odjąć od fragmentu dzielnej. Jeżeli tak, to w wyniku wprowadza się jedynkę — w kolumnie nad najmłodszą cyfrą dzielnika (prawa strona).
2. Następnie odejmuje się cyfry i uzupełnia brakujące znaki w powstałej dzielnej cyframi przepisanymi z dzielnej oryginalnej.
3. Jeżeli dzielnika nie da się odjąć od fragmentu dzielnej, w wyniku wprowadza się zero, a dzielną przepisuje się bez zmian.

   Cały proces powtarza się aż do momentu uzyskania ostatniej cyfry w wyniku.
4. Jeżeli ostatnie odejmowanie nie może być wykonane lub z ostatniej różnicy nie wychodzą zera, przepisana dzielna lub liczba powstała z różnicy stanowi resztę z dzielenia.

Przykłady

			1	0	1	1	0	0				
	1	1^{-1}	0	1	1	1	0	1	:	1	0	1
–	1	0	1									
	0	*0*	*1*	1	1	1	0	1				
	–	1	0	1								
		0	1	1	1	1	0	1				
		–	1	0	1							
			0	*1*	*0*	1	0	1				
			–	1	0	1						
				0	*0*	*0*	0	1				
				–	1	0	1					
					0	0	0	1				
					–	1	0	1				
reszta	0	0	0	0	0	0	0	1				

		1	0	1			
	1	0	1	0	:	1	0
–	1	0					
	0	0	1	0			
	–	1	0				
		0	1	0			
		–	1	0			
			=	=			

1.3. Zapis liczb binarnych ze znakiem

W systemie dziesiętnym liczby ujemne są opatrzone specjalnym znakiem graficznym — minusem „–”, np. –6, –22 itd., a liczby dodatnie w niektórych przypadkach plusem „+”, np. +5, +20. W systemie binarnym opartym wyłącznie na zerach i jedynkach brakuje dodatkowego znaku, który wskazywałby na ujemny lub dodatni charakter określonej liczby.

Opracowano kilka metod zapisu liczb binarnych ze znakiem, które charakteryzują się różnym stopniem przydatności, m.in.:

- metodę znak-moduł (ZM),
- metodę uzupełnień do 1 (U1),
- metodę uzupełnień do 2 (U2).

1.3.1. Metoda znak-moduł (ZM)

W metodzie znak-moduł zastosowano prosty zabieg kodowania znaku za pomocą najstarszej cyfry w liczbie binarnej. Najstarszą cyfrę określa się jako znak, podczas gdy pozostałe cyfry są modułem reprezentującym daną liczbę binarną.

znak	moduł			
a_{n-1}	a_{n-2}	…	a_1	a_0

W celu obliczenia wartości naturalnej liczby binarnej ze znakiem należy się posłużyć następującym wzorem:

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

Stosując powyższy (zmodyfikowany) zapis wielomianowy, zauważymy, że znak otrzymanej po jego wyliczeniu liczby jest zależny od wyrażenia: 1–2*najstarsza cyfra liczby. Jeżeli najstarsza cyfra jest jedynką, to wynikiem wyrażenia będzie –1; jeżeli zerem, otrzymamy 1. Obliczony moduł należy przemnożyć przez wyrażenie znakowe, dzięki czemu otrzymujemy dodatnią lub ujemną liczbę dziesiętną będącą odpowiednikiem danej liczby binarnej.

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 — dla liczby ujemnej.

Przykłady

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

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

Jedną z wad metody ZM jest brak możliwości prostego wykonywania operacji arytmetycznych, co znacznie ogranicza jej powszechne stosowanie:

$$\begin{array}{cccccc} & 1 & 1 & 1 & 0_{ZM} \\ + & 0 & 1 & 1 & 0_{ZM} \\ \hline 1 & 0 & 1 & 0 & 0 \end{array} \qquad \begin{array}{cc} & -6_D \\ + & 6_D \\ \hline & 20_D \end{array}$$

Kolejną niedogodnością związaną z systemem znak-moduł jest to, że zero może zostać zapisane na dwa sposoby: ze znakami plus i minus. To przykład nieefektywności tej metody, w której tracony jest jeden wyraz kodowy.

1.3.2. Metoda uzupełnień do 2 (U2)

Niedoskonałości systemu ZM spowodowały, że konieczne było opracowanie bardziej naturalnej metody zapisu liczb binarnych ze znakiem. Powstała **metoda uzupełnień do 2** (U2), w której cyfra określająca znak jest zintegrowana z liczbą binarną, co pozwala na wykonywanie obliczeń arytmetycznych.

W celu obliczenia wartości liczby binarnej z wykorzystaniem metody U2 należy zastosować poniższy wzór:

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

W metodzie U2 wyrażenie znaku jest tak skonstruowane, że uczestniczy w ustalaniu wartości liczby tak jak pozostałe pozycje. Wartość podstawy w wadze najstarszej liczby określającej znak jest ujemna.

Przykłady

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

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

Jak widać w przykładach, liczby binarne dodatnie i ujemne U2 wyglądają po przekształceniu na dziesiętne inaczej niż w przypadku metody ZM.

Przekształcenie ujemnej liczby dziesiętnej na postać binarną jest bardziej pracochłonne niż w metodzie ZM.

1. Na początku obliczamy postać binarną z wartości bezwzględnej dziesiętnej liczby ujemnej:

$$5 : 2 = 2 \quad r = 1$$
$$2 : 2 = 1 \quad r = 0$$
$$1 : 2 = 0 \quad r = 1$$

$$-5_D = |-5_D| = 5_D = 101_B$$

2. Powstałą liczbę binarną należy uzupełnić zerami do liczby cyfr będącej krotnością dwójki. W tym przypadku, gdy liczba binarna ma trzy cyfry, dopełniamy do czterech. Jeżeli byłoby siedem cyfr, należałoby uzupełnić do ośmiu itd.

$$0101_B$$

3. Następnie należy zamienić wszystkie cyfry w liczbie binarnej na przeciwne, czyli jedynki na zera i odwrotnie:

1010

4. W ostatnim etapie do powstałej liczby dodajemy binarną jedynkę, a wynik jest ujemną liczbą binarną:

$$\begin{array}{cccc} 1 & 0 & 1 & 0 \\ + \quad 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 \end{array}$$

$$1_3\,0_2\,1_1\,1_0 = 1(-2^3) + 0(2^2) + 1(2^1) + 1(2^0) = -8 + 3 = -5$$

1.4. Liczby binarne stało- i zmiennoprzecinkowe

Podobnie jak w systemie dziesiętnym, liczby binarne mogą być zapisane w postaci ułamkowej. Zapis binarnych liczb pozycyjnych z przecinkiem może przyjąć postać stało- lub zmiennoprzecinkową.

1.4.1. Liczby stałoprzecinkowe (stałopozycyjne)

Liczby stałoprzecinkowe (ang. *fixed-point numbers*) umożliwiają zapis liczb w postaci ułamkowej, tak że pozycja przecinka jest ustalana arbitralnie w zależności od wymaganej dokładności.

Binarną liczbę stałoprzecinkową można potraktować jako złożenie dwóch części — liczby całkowitej oraz ułamkowej rozdzielonych przecinkiem:

część całkowita	część ułamkowa
10110011,	0101

W celu przekształcenia binarnej liczby stałoprzecinkowej na postać dziesiętną należy się posłużyć poniższym wzorem:

$$a_{n-1} \ldots a_1\, a_0, a_{-1} \ldots a_{-m} = a_{n-1}*2^{n-1}+\ldots+a_1*2^1+a_0*2^0+a_{-1}*2^{-1}+\ldots+a_{-m}*2^{-m}$$

Wartości wag części ułamkowej przyjmują postać ułamków, w których dokładność jest określona przez wagę najmłodszej cyfry.

Przykłady

$1101{,}11_B = 1_3\,1_2\,0_1\,1_0, 1_{-1}\,1_{-2} = 1*2^3+1*2^2+0*2^1+1*2^0+1*2^{-1}+1*2^{-2} = 8+4+1+1/2+1/4 = 13{,}75_D$

$11100101{,}1011 = 2^7+2^6+2^5+2^2+2^0+2^{-1}+2^{-3}+2^{-4} = 128+64+32+4+1+1/2+1/8+1/16 = 229{,}6875_D$