2.3.6. Reprezentacja danych liczbowych w komputerze

Binarna reprezentacja liczb ujemnych

Dane w komputerze zapisywane są w postaci liczb binarnych. Wynika to stąd, że najmniejsza jednostka informacji, czyli bit, służy do zapisu jednej cyfry systemu dwójkowego: 0 lub 1. Dotychczas poznaliśmy reprezentację binarną liczb całkowitych nieujemnych. Wartości ujemne zapisuje się, **używając kodu uzupełnieniowego do dwóch**, zwanego **kodem U2**. Ogólny zapis liczby w kodzie U2 można przedstawić za pomocą następującego wzoru:

$$y = \begin{cases} x & dla \quad x \geq 0 \\ 2^n + x & dla \quad x < 0 \end{cases} \tag{2.17}$$

gdzie:

x — liczba, którą chcemy zapisać w kodzie U2;

n — liczba bitów przeznaczonych do zapisania kodowanej liczby;

y — liczba x zapisana za pomocą kodu U2.

Liczba y po wykonaniu obliczeń przedstawiana jest w postaci binarnej.

Zakres wartości liczby x, którą konwertujemy za pomocą kodu U2, zależy od liczby bitów przeznaczonych do zapisania tej liczby. Mając do dyspozycji n bitów, pierwszy bit rezerwujemy do oznaczenia znaku liczby (1 — liczba ujemna, 0 — liczba nieujemna), pozostałe $n-1$ bitów do zapisania liczby. Zauważmy, że rolę pierwszego bitu można rozumieć na dwa równoważne sposoby: reprezentuje on znak liczby x w kodzie U2, a zarazem jest pierwszym (najbardziej znaczącym) bitem nieujemnej liczby y w zwykłym układzie dwójkowym. Wartość kodowanej liczby x zawiera się więc w przedziale $[-2^{n-1}, 2^{n-1})$.

Przykład 2.18.

Załóżmy, że dysponujemy 1 bajtem (czyli 8 bitami) przeznaczonym do zapisania liczby. Stąd $n = 8$, a wartość kodowanej liczby x musi zawierać się w przedziale $[-2^{n-1}, 2^{n-1}) = [-2^7, 2^7) = [-128_{10}, 128_{10})$. Korzystając z wzoru, wyznaczmy wartość liczby $x = -56$ w kodzie U2. Musimy wykonać następujące obliczenia:

$$y = 2^8 + (-56) = 256 - 56 = 200_{10}.$$

Następnie konwertujemy uzyskaną wartość z systemu dziesiętnego na dwójkowy:

$$200_{10} = 11001000_2.$$

Uzyskana liczba, $y = 11001000_2$, to zakodowana wartość liczby $x = -56_{10}$.

Zadanie 2.27. Zapisz podane liczby ujemne dla określonej wartości n za pomocą kodu U2.

a) -108_{10} dla $n = 8$ bitów,

b) -99_{10} dla $n = 8$ bitów,

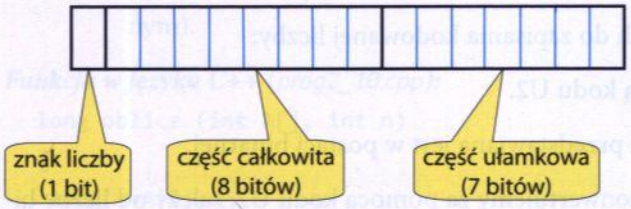
c) -241_{10} dla $n = 16$ bitów,

d) -189_{10} dla $n = 16$ bitów.

Stałopozycyjna reprezentacja liczb

Stałopozycyjna reprezentacja liczb charakteryzuje się stałym położeniem przecinka, który oddziela część całkowitą od części ułamkowej zapisywanej liczby. Powoduje to, że taki zapis liczby jest dokładny tylko wtedy, gdy dana liczba nie wykracza poza zakres miejsca, jakie zostało przeznaczone do jej zapisu.

Załóżmy, że mamy do dyspozycji 2 bajty (czyli 16 bitów) do zapisania liczby w reprezentacji stałopozycyjnej. Wówczas podział na część całkowitą i ułamkową może przedstawiać się jak na rysunku 2.7.



Rysunek 2.7. Przykładowy podział na część całkowitą i ułamkową w stałopozycyjnej reprezentacji liczb

W reprezentacji stałopozycyjnej liczba zapisywana jest w kodzie uzupełniającym do dwóch.

Potrafimy już wykonywać konwersję liczby całkowitej pomiędzy systemami binarnym i decymalnym. Jednak aby przedstawić liczbę rzeczywistą z wykorzystaniem reprezentacji stałopozycyjnej, trzeba uwzględnić również część ułamkową.

Konwertując liczby z systemu binarnego na decymalny, mnożymy kolejne cyfry tej liczby przez potęgi dwójki. W części ułamkowej mnożnikiem są ujemne potęgi liczby 2.

Przykład 2.19.

Zapiszmy liczbę rzeczywistą $101111{,}01101_2$ w systemie dziesiętnym. Zaczynamy od dopasowania potęg liczby 2 do kolejnych cyfr podanej wartości:

1	0	1	1	1	1	,	0	1	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}

Następnie obliczamy wartość konwertowanej liczby w systemie dziesiętnym:

$$101111{,}01101_2 = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} + 0\cdot 2^{-4} + 1\cdot 2^{-5} = 47\frac{13}{32_{10}}.$$

Zamiana liczby ułamkowej z systemu dziesiętnego na dwójkowy wykonywana jest przez mnożenie części ułamkowej przez 2 tak długo, aż w części ułamkowej uzyskamy zero lub zauważymy, że wynikiem jest ułamek nieskończony. Rozwiązaniem jest liczba utworzona z całkowitych części wyników uzyskiwanych podczas mnożenia liczby przez 2.

Przykład 2.20.

Przekonwertujmy liczbę ułamkową $0{,}1875_{10}$ na system binarny.

W tym celu należy wykonać mnożenie części ułamkowej tej liczby przez 2:

0,1875	0,1875 · 2	=	0,375
0,375	0,375 · 2	=	0,75
0,75	0,75 · 2	=	1,5
1,5	0,5 · 2	=	1,0
1,0			

Rozwiązaniem jest ułamek skończony. Widać to po wartości ostatniego wyniku 1,0, w którym część ułamkowa wynosi zero. Uzyskaliśmy więc następujące rozwiązanie:

$$0{,}1875_{10} = 0{,}\mathbf{0011}_2.$$

Poniżej pokazany został czytelniejszy zapis konwersji liczb ułamkowych z systemu decymalnego na binarny:

0	1875
0	375
0	75
1	5
1	0

Przykład 2.21.

Przekonwertujmy liczbę $0{,}2_{10}$ na system binarny. Rozwiązaniem będzie ułamek nieskończony.

0	2
0	4
0	8
1	6
1	2
0	4
0	8
1	6
1	2
0	4

...

Łatwo zauważyć, że sekwencja liczb „0011" będzie się powtarzać. Wynikiem jest więc ułamek okresowy $0{,}(0011)_2$.

Zadanie 2.28. Przekonwertuj podane liczby rzeczywiste na system dziesiętny:

a) $10100{,}11101_2$,

b) $0{,}0111011_2$,

c) $11{,}110001_2$,

d) $10110011{,}11100101_2$,

e) $11011100{,}10010101_2$.

Zadanie 2.29. Przekonwertuj podane liczby rzeczywiste na system binarny:

a) $852{,}6875_{10}$,

b) $620{,}09375_{10}$,

c) $612{,}03125_{10}$,

d) $1536{,}9921875_{10}$,

e) $2707{,}7734375_{10}$.

Zadanie 2.30. Wykonaj następujące operacje arytmetyczne w systemie binarnym:

a) $1110{,}011_2 \cdot 10001{,}00111_2$,

b) $1111{,}001_2 + 100000{,}11011_2$,

c) $10011{,}01011_2 - 100{,}1011_2$.

System U2

System U2 inaczej **kod uzupełnień do dwóch** jest przeznaczony do przechowywania liczb całkowitych dodatnich i ujemnych. W języku C++ zmienne typu **int**, **long long,** czy **char** używają tego systemu do przechowywania wartości. Żeby można było wykonywać operacje w tym systemie, należy określić na ilu bitach będziemy operować.

Zakres wartości dla **n** bitów mieści się w przedziale:

$$[-2^{n-1}; 2^{n-1}-1]$$

,

a więc na ośmiu bitach możemy przechować liczby z zakresu $[-128; 127]$.

Zamiana liczb dodatnich

Dla przykładu przedstawimy liczbę 50

na ośmiu bitach w **U2**.

Najpierw zamieniamy ją na system dwójkowy:

$$50 = (110010)_2$$

Następnie z lewej strony dopełniamy zerami tak, aby w sumie otrzymać osiem bitów i w rezultacie otrzymujemy:

$$50 = (00110010)_{U2}$$

Zamiana liczb ujemnych

Teraz zamieńmy liczbę -50

na system **U2**. Tu algorytm jest bardziej skomplikowany.

W pierwszym kroku wyznaczamy wartość bezwzględną z tej liczby:

$$|-50|=50$$

W drugim kroku otrzymaną liczbę zamieniamy na postać binarną:

$$50 = (110010)_2$$

W trzecim kroku przedstawiamy ją na ośmiu bitach:

$$50 = 00110010$$

W czwartym kroku negujemy wszystkie bity (każdy bit zamieniamy na przeciwny: zero na jedynkę, jedynkę na zero):

$$\sim(00110010) = 11001101$$

Na końcu zwiększamy otrzymaną postać o **1**:

$$11001101 + 1 = 11001110$$

W ten sposób otrzymaliśmy liczbę -50

zapisaną na **ośmiu bitach** w systemie **U2:**

$$-50 = (11001110)_{U2}$$

.

Zamiana z U2 na system dziesiętny

Najbardziej znaczący bit (ten który stoi po lewej stronie) określa znak liczby. Jeśli jest to **jedynka,** to liczba jest **ujemna**, w przeciwnym razie jest ona **dodatnia.** Dla przykładu posłużmy się otrzymaną wyżej postacią:

$$(11001110)_{U2}$$

Teraz odliczamy kolejne potęgi dwójki począwszy od strony prawej. Przy pierwszym bicie stoi 2_0

, przy następnym 2_1, ..., natomiast przy ostatnim -2_7

. Teraz dodajemy tylko te potęgi liczby dwa, które stoją nad cyfrą **1**:

$$-2_7+2_6+2_3+2_2+2_1=-128+64+8+4+2=-50$$