2. Predstavljanje brojeva

2.1 Brojni sistemi

Uobičajen način predstavljanja brojeva koji koristimo svakodnevno je dekadni brojni sistem u kojem se brojevi zapisuju ciframa iz skupa 0,1,2,...,9.

Ovaj način prikazivana pripada klasi pozicionih brojnih sistema u kojima, ako koristimo cifre iz skupa $\{c_1, c_2, \ldots, c_n\}$, tada se neki broj $x \in N$ predstavlja u obliku $c_k c_{k-1} \ldots c_1 c_0$ tako da je:

$$x = i_k \cdot n^k + i_{k-1} \cdot n^{k-1} + \ldots + i_1 \cdot n + i_0 \qquad (1.1)$$

Broj cifara koje se koriste za pisanje broja nazivamo osnovom brojnog sistema.

Tako za broj 1997 zapisan u dekadnom sistemu (osnove 10) imamo:

$$1997 = 1 \cdot 10^3 + 9 \cdot 10^2 + 9 \cdot 10 + 7.$$

Moguće je brojeve zapisivati sa različitim skupovima cifara $\{c_1, c_2, \ldots, c_n\}$, pa čak i sa skupom od jedne cifre. Razvoj računarske tehnike, baziran na Bulovoj algebri, uslovio je da se pored dekadnog sistema, koji je pogodan za ljudsku upotrebu, intezivno koriste binarni, oktalni i heksadecimalni brojni sistemi.

Kao što se može naslutiti iz njihovih latinskih naziva, radi se sa sistemima osnove 2, 8 i 16, respektivno.

Kod **binarnog** brojnog sistema koriste se cifre iz skupa 0,1, pa se tako neki broj $x \in N$ zapisuje kao $b_k b_{k-1} \ldots b_1 b_0$, gdje su $b_i \in \{0,1\}$ odabrani tako da je:

$$x = b_k \cdot 2^k + b_{k-1} \cdot 2^{k-1} + \ldots + b_1 \cdot 2 + b_0. \qquad (1.2)$$

Tako se recimo broj 135 zapisuje u binarnom obliku kao 10001010 jer je:

$$136 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2 + 0$$

Binarne cifre nazivamo $b_i \in \{0,1\}$ nazivamo i bitovima (skraćenica od engleskog binary digit).

Oktalni brojni sistem koristi 8 cifara iz skupa 0,1,2,...,7 pa se, u tom sistemu, broj 136 zapisuje kao 210 jer je:

$$136 = 2 \cdot 8^2 + 1 \cdot 8 + 0.$$

Heksadecimalni brojni sistem koristi 16 cifara iz skupa 0,1,2,...,9,A,B,C,D,E,F, pa se u njemu broj 136 zapisuje kao 88 jer je:

$$136 = 8 \cdot 16 + 8.$$

Konverzija iz binarnog sistem u dekadni sistem je jednostavna i vrši se primjenom jednakosti (1.2).

Obrnuta konverzija, iz dekadnog u binarni oblik, je nešto složenija i vrši se poznatim algoritmom uzastopnog dijeljenja dekadnog zapisa sa 2. Pri dijeljenju se zapisuju redom ostaci (koji mogu biti 0 ili 1), a na kraju se od ostataka formira binarni zapis (u obrnutom redosljedu).

Zato je konverzija iz binarnog u oktalni i heksadecimalni oblik, kao i obrnute konverzije izuzetno jednostavna.

Iz oktalnog (heksadecimalnog) zapisa, konverzija se vrši tako što se svaka oktalna (heksadecimalna) cifra "proširi" u binarni prikaz.

Na primjer oktalni broj 765 se konvertuje u binarni broj kako slijedi:

7 6 5

1011 1100 101

to jest $(765)_8=(10111100101)_2$.

Ili na primjer heksadecimalni broj 5FA se konvertuje u:

5 F A

101 1111 1010

te je $(5FA)_{16}=(10111111010)_2$.

Obrnute konverzije iz binarnog u oktalni (heksadecimalni) zapis se vrše tako što se binarni broj rasporedi u razrede (grupe od po 3 binarne cifre za oktalni, ili od 4 cifre za heksadecimalni sistem), pa se svakom razredu dodijeli odgovarajuća oktalna (heksadecimalna cifra).

Na primjer binarni broj 1101110010101 se raspore|ivanjem:

1 101 110 010 101

1 5 6 2 5

konvertuje u $(15625)_8$, rapore|ivanjem:

1 1011 1001 0101

1 B 9 5

konvertuje u $(1B95)_{16}$.

2.2. Nenegativni cijeli brojevi

U računarskoj tehnici je uobičajeno da se brojevi zapisuju u binarnom obliku, onako kako se zapravo smještaju u računarsku memoriju. Heksadecimalni (oktalni) oblik se koristi jedino sa ciljem skraćenog pisanja, jer je dužina binarnog zapisa često velika, pa nije pogodna za čovječju upotrebu.

S druge strane, obično se kod računara koriste fiksne dužine riječi za memorisanje binarnih zapisa. Tako se svaki broj zapisuje sa recimo 32 binarne pozicije, kao na primjer.

$(00000000000000000000000000000011)_2=(3)_{10}$

Za slučaj 32-bitne dužine binarnog zapisa imamo 2^{32} =4,294,967,296 različitih zapisa kako slijedi:

Binarni zapis	Dekadno
31 0	
000000000000000000000000000000	0
000000000000000000000000000001	1
000000000000000000000000000011	2
...	
111111111111111111111111111101	4,294,967,293
111111111111111111111111111110	4,294,967,294
111111111111111111111111111111	4,294,967,295

Cifru na poziciji 0 nazivamo cifrom najmanjeg značaja (least significant), a cifru napoziciji 31 cifrom največeg znažaja (most significant).

I tako, nenegativni brojevi mogu biti zapisani na pomenuti način, gdje za svaki broj x u 32-bitnoj prezentaciji važi:

$$x = \sum_{i=0}^{31} b_i 2^i \qquad (1.3)$$

gdje je $b_i \in \{0,1\}$.

2.3 Negativni brojevi

Problem nastaje kada želimo da pored nenegativnih brojeva koristimo i negativne brojeve. Kako ih zapisivati? Ako, recimo, usvojimo da cifra na poziciji 31 igra ulogu znaka, tako da su svi brojevi sa cifrom 0 na 31-ov poziciji nenegativni, a sa cifrom 1 negativni brojevi, možemo naizgled dobiti razumno rješenje za postavljeni problem.

Međutim, u tom slučaju imamo dva zapisa za broj nula to jest zapis:

00000000000000000000000000000000, "pozitivna nula",

kao i zapis

10000000000000000000000000000000, "negativna nula".

Ovo bi moglo da nam predstavlja problem pri radu.

Zato je opšte usvojen jedan drugi način zapisivanja brojeva koji otklanja ovaj nedostatak, a ima i druge prednosti.

Nenegativni i negativni brojevi se zapisuju na način kako slijedi:

Binarni zapis	Dekadno
31 0	
000000000000000000000000000000	0
000000000000000000000000000001	1
000000000000000000000000000011	2
...	
011111111111111111111111111101	2,147,483,645
011111111111111111111111111110	2,147,483,646
011111111111111111111111111111	2,147,483,647
100000000000000000000000000000	-2,147,483,648
100000000000000000000000000001	-2,147,483,647
100000000000000000000000000010	-2,147,483,646
...	
111111111111111111111111111101	-3
111111111111111111111111111110	-2
111111111111111111111111111111	-1

Ovako predstavljanje zapravo odgovara sljedećoj jednakosti za broj x (nenegativan ili negativan, svejedno):

$$x = -b_{31} \cdot 2^{31} + \sum_{i=0}^{30} b_i 2^i .$$

Ovakva prezentacija brojeva naziva se "drugim komplementom". Kao što se možemo lako uvjeriti ona obezbje|uje i da je x+(-x)=0.

Naziv "drugi komplement" nastao je otuda što se ovakvo predstavljanje negativnog broja dobija kad se njegova apsolutna vrijednost predtavi kao binarni broj u standardnom obliku, a zatim sve binarne cifre tog broja invertuju (0 u 1, a 1 u 0) i na kraju doda 1.

Tako na primjer broj -151, čija apsolutna vrijednost 151 ima binarni oblik 10010111, poslije operacija invertovanja i dodavanja 1:

010010111	apsolutna vrijednost broja
101101000 +	invertovanje
1	dodavanje jedinice

101101001	drugi komplement

što znači da će broj -151 biti predstavljen kao 101101001.

2.4 Realni brojevi

Pored cijelih brojeva čije smo predstavljanje pokazali u poglavljima 2 i 3, potrebno je usvojiti i neki pogodan načIn za predstavljanje realnih brojeva.

Evo, najprije, nekoliko primjera uobičajenog dekadnog zapisivanja realnih brojeva:

3.14159265... - broj pi

2.71828... - broj e

0.000000001 ili 1.0×10^{9} - broj nanosekundi u 1 sec.

3,155,760,000 ili 3.15576×10^{9} - broj sekundi u vijeku.

Kada je broj zapisan u obliku $c_1.c_2c_3\ldots 10^N$ pri čemu je $c_1 \neq 0$, kažemo da je tada broj prikazan u **normalizovanoj formi**.

Lako je uočiti da se realni broj može predstaviti i u normalizovanoj binarnoj formi: $1.bbb\ldots 2^N$, gdje su b binarne cifre (0 i 1). Ponovo ćemo uzeti da nam na raspolaganju stoji 32 bita za zapisivanje realnog broja. Tada možemo usvojiti sljedeći format zapisa realnog broja:

31	30	23	22	0
s	eksponent		mantisa	
1bit	8 bitova		23 bita	

Bit 31 predstavlja znak broja (0 pozitivan, 1 negativan), eksponent je je eksponent n (uključujući znak) u standardnoj binarnoj prezentaciji (nije drugi komplement), a mantisa predstavlja dio broja iza decimalnog zareza u binarnom obliku. U ovakvom načinu predstavljanja moguće je zapisivati realne brojeve na segmentu:

$$[-2.0x10^{-38}, 2.0x10^{38}].$$

I pored tako velikog opsega, ponekad je potrebno zapisivati i brojeve van ovog opsega. U tu svrhu, definiše se i takozvana prezentacija realnih brojeva sa dvostrukom preciznošću kako slijedi:

31	30	20	19	0
s	eksponent		mantisa	
1bit	11 bitova		20 bitova	

mantisa (nastavak)
32 bita