Cvetana Krstev

Strukture Informacija – Uvod

Predstavljanje brojeva za potrebe računanja

Jedan način za kodiranje brojeva je njihovo predstavljanje, kao i svih drugih simbola u pisanom tekstu - slova, interpunkcijskih znakova, specijalnih znakova – nekom kodnom šemom. Broj, kakav je, na primer, **1998** može se korišćenjem ASCII kôda kodirati sa 4 bajta na sledeći način:

bajt	binarni kôd	heksadec. kôd	vrednost
1	00110001	31	1
2	00111001	39	9
3	00111001	39	9
4	00111000	38	8

Brojevi se na ovaj način kodiraju kad god su deo nekog teksta u najširem smislu reči, koji će se čuvati na računarskom medijumu i biti podvrgnut nekoj tekstualnoj transformaciji. Primeri pojavljivanja brojeva kodiranih u tekstu cifra po cifra su sledeći izvodi iz dnevnih novina od 2. aprila 1998:

> Vlada Jugoslavije zvanično je procenila da nemačka marka vredi šest, a ne 3,3 dinara, na koliko je zakovana u novembru 1995. godine.
> ... da će građani već od danas u poslovnim bankama moći da kupe po 500 maraka u toku godine ...
> Velikom pakovanju Neskafe nova cena je 70 dinara, a malom 35 dinara.
> Na „crnom tržištu" u Novom Sadu nemačka marka se juče otkupljivala za 6, a prodavala za 6,5 dinara.

Međutim, kada je potrebno da računar izvrši neko izračunavanje, kakvo je, na primer, imajući u vidu gornje informacije, izračunavanje za koliko procenata je dinar devalvirao, za koliko je procenata u Novom Sadu „crni kurs" veći od zvaničnog, koliko je građanima potrebno dinara da bi od poslovnih banaka otkupili 500 maraka i slično, brojevi se u računaru predstavljaju na drugi

način. Za potrebe računanja, brojevi se u računaru predstavljaju na jedan od sledećih načina:

- predstavljanje celih brojeva u binarnom brojnom sistemu;
- predstavljanje realnih brojeva u pokretnom zarezu;
- predstavljanje brojeva binarnim kodiranjem dekadnih cifara.

Prvo ćemo videti kako se mogu kodirati binarne cifre.

Predstavljanje brojeva kodiranjem cifara

Cifre se u rečunaru kodiraju binarnim alfabetom {0,1}. Radi jednostavnosti predstavljanja i manipulisanja za kodiranje se koriste reči iste dužine – u pitanju je, dakle, **kôd fiksne dužine**. Objekata koje treba kodirati ima 10, to su sve cifre u dekadnom brojnom sistemu:

{0,1,2,3,4,5,6,7,8,9}

Odavde zaključujemo da dužina reči za kodiranje binarnim alfabetom ne može da bude 1, jer postoje samo dve takve reči, to su 0 i 1. Dužina ne može da bude ni 2, jer postoje samo 4 takve reči, to su 00, 01, 10, 11. Dužina ne može da bude ni 3, jer postoje samo 8 takvih reči, to su 000, 001, 010, 011, 100, 101, 110, 111. Reči sastavljenih od binarnog alfabeta dužine 4 ima 16 – to su 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111 - pa je ta dužina dovoljna za kodiranje 10 cifara dekadnog brojnog sistema. Jasno je da 6 od svih 16 mogućih reči dužine 4 neće biti iskorišćeno za kodiranje 10 cifara, pa imamo **nepotpun kôd**.

Očigledno je da se 10 kodnih reči dužine 4 u binarnom alfabetu od mogućih 16 može na različite načine izabrati i da se oni mogu na različite načine dodeliti ciframa dekadnog brojnog sistema. To se može zapravo učiniti na nekoliko desetina miliona načina ($16!/6! \approx 2.9*10^{10}$ – u pitanju su kombinacije bez ponavljanja). Ipak na izabrani skup reči se obično postavljaju neka ograničenja:

1. Da bi se obezbedila jednoznačnost kôda sve binarne reči koje ulaze u njega moraju biti međusobom različite (preslikavanje je 1-1).
2. Najvećoj dekadnoj cifri (9) treba pridružiti reč koja je posmatrana kao četvorocifreni binarni broj ima najveću težinu (vrednost).
3. Parnim i neparnim dekadnim ciframa treba pridružiti parne, odnosno neparne, binarne brojeve.
4. Ako dve dekadne cifre ispunjavaju uslov $p + q = 9$ i ako su $y_3y_2y_1y_0$ cifre prvog broja, onda su cifre drugog broja $z_3z_2z_1z_0$ takve da je svaka cifra

z_i komplementarna odgovarajućoj cifri y_i prvog prvog broja. To znači da ako je cifra y_i bila 1 onda će cifra z_i biti 0, i obrnuto ako je cifra y_i bila 0 onda će cifra z_i biti 1. Ovo je **komplementarni kôd**.

5. Ako *i*-toj poziciji u kôdu može da se pridruži broj p_i tako da za svaku dekadnu cifru q, sa pridruženom kodnom reči $y_3y_2y_1y_0$ važi jednakost:

$$q = p_3y_3 + p_2y_2 + p_1y_1 + p_0y_0$$

Ovo je **težinski kôd**. Težine su p_0, p_1, p_2 i p_3.

Poslednja dva uslova, po pravilu, doprinose jednostavnijem izvođenju aritmetičkih operacija.

Dekadna cifra	Binarni kôd			
	8421	2421	višak 3	Ciklični
0	0000	0000	0011	0001
1	0001	0001	0100	0101
2	0010	0010	0101	0111
3	0011	0011	0110	1111
4	0100	0100	0111	1110
5	0101	1011	1000	1100
6	0110	1100	1001	1000
7	0111	1101	1010	1001
8	1000	1110	1011	1011
9	1001	1111	1100	0011
osobine	1,2,3,5	1,2,3,4,5	1,2,4	1

Kôd 8421 je drugi naziv za **prirodni binarni kôd dekadnih cifara NBCD** (engl. natural binary-coded decimal). To znači se svakoj cifri dekadnog brojnog sistema pridružuje binarni kôd koji ima istu vrednost kao dekadna cifra predstavljena u binarnom brojnom sistemu. Na primer, cifra 5 se predstavlja binarno kao: $5 = 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$. Stepeni 2^3, 2^2, 2^1, 2^0 su upravo težine 8, 4, 2, 1, tj. $5 = 0*\mathbf{8} + 1*\mathbf{4} + 0*\mathbf{2} + 1*\mathbf{1}$.

Kôd 2421 (enlg. *Aiken code*) je takođe težinski iako nije prirodan jer težina četvrte cifre nije 8 već opet 2. Tako je, na primer, cifra 5 sada kodirana sa 1011, što u ovom kodu znači: $5 = 1*\mathbf{2} + 0*\mathbf{4} + 1*\mathbf{2} + 1*\mathbf{1}$. Primećujemo da je i u ovom kodu cifra 5 mogla da bude kodirana sa rečju 0101 kao i u

prethodnom primeru, ali je ovim izborom i izborom drugih kodnih reči kôd i komplementaran. Na primer,

4+5 = 9 3+6 = 9
4 = 0 1 0 0 3 = 0 0 1 1
5 = 1 0 1 1 6 = 1 1 0 0

U **kôdu višak 3** (engl. *excess-3*) je težinska suma četiri bita u svakoj kodnoj reči za tri veća od dekadne cifre predstavljene tom reči, i pogodan je za računanje što ćemo videti kada budemo radili BCD aritmetiku, zato se koristio recimo kod ranih kalkulatora iz 70-ih godina prošlog veka. Na primer, kôd cifre 5 je sada 1000 što ima binarnu vrednost $1*2^3 + 0*2^2 + 0*2^1 + 0*2^0 = 8$, a 8 = 5+3. I ovaj kôd je komplementaran. Na primer,

4+5 = 9 3+6 = 9
4 = 0 1 1 1 3 = 0 1 1 0
5 = 1 0 0 0 6 = 1 0 0 1

Ciklički kôd ima svojstvo da se pri prelasku se jedne na sledeću dekadnu cifru po vrednosti menja samo jedna binarna cifra u odgovarajućim kodnim rečima. On nije pogodan za računanje ali ima primene kod nekih drugih procesa u kontroli rada računara.

Kako se kodira broj 1998 kodiranjem njegovih cifara koristeći predstavljene kodove je prikazano u donjoj tabeli.

Broj 1998	1	9	9	8
kod 8421	0001	1001	1001	1000
kod 2421	0001	1111	1111	1110
kod višak 3	0100	1100	1100	1011
ciklični kod	0101	0011	0011	1011

Predstavljanje celih brojeva u binarnom brojnom sistemu

Celi (pozitivni) brojevi se za potrebe računanja najčešće beleže u binarnom brojnom sistemu. Tako bi isti broj 1998, ako bi s njim trebalo računati (na primer, koliko godina ima danas osoba koja je 1998. godine imala 46 godina) bio predstavljen u sledećem obliku:

1998 : 16 ostatak 14 ⇑
124 : 16 ostatak 12 ⇑

7	: 16	ostatak 7	⇑
0	kraj		⇑

U ovom slučaju se broj 1998 u memoriji računara beleži u heksadecimalnom obliku kao $7CE_{16}$ a u binarnom obliku $0111\ 1100\ 1110_2$. Setićemo se da je heksadecimalni brojni sistem pozicioni brojni sistem sa osnovom 16 čije se cifre označavaju sa {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}. Heksadecimalni brojni sistem se u računarstvu često koristi upravo zbog toga što je njegov kôd fiksne dužine sa najkraćom dužinom kodne reči potpun. Naime, sadržaj memorijskih lokacija, bajtova i reči, se često umesto binarnim predstavlja heksadecimalnim ciframa. Kako su i dužina bajta i dužina reči uvek umnožak od d=4, binarni zapis dužine 8, 16, 32,... (što su uobičajene dužine računarskih memorijskih lokacija) treba podeliti u grupe od četiri bita, odnosno binarne cifre, i zameniti ih odgovarajućim heksadecimalnim ciframa.

Druga prednost heksadecimalnog brojnog sistema je lako prevođenje u binarni brojni sistem i iz njega. Podsetićemo se da kada su u pitanju brojni sistemi kod kojih je osnova jednog stepen osnove drugog, onda je prevođenje iz jednog u drugi brojni sistem jednostavno. Takav je, na primer, slučaj heksadecimalnog i binarnog brojnog sistema jer je $16=2^4$. Tada se broj iz heksadecimalnog brojnog sistema prevodi u broj u binarnom brojnom sistemu zamenom svake heksadecimalne cifre njenim prirodnim binarnim kôdom fiksne dužine. Na primer,

$A2F_{16}=1010\ 0010\ 1111_2$

Broj se iz binarnog brojnog sistema prevodi u heksadecimalni brojni sistem tako što se binarne cifre grupišu u grupe od četiri cifre (ovde je četiri stepen osnove 2 koja daje drugu osnovu 16) počev od cifara manje težine ka ciframa veće težine. Cifre veće težine se po potrebi dopunjavaju nulama do potpune grupe od 4 cifre. Na primer,

$1010\ \ 0010\ \ 1111_2 = A2F_{16}$

Da bismo se uverili u ispravnost ovog postpuka prevedimo oba broja u dekadni brojni sistem:

$A2F_{16} = 10\cdot16^2+2\cdot16^1+15\cdot16^0=2560+32+15= 2607_{10}$

$101000101111_2 \quad = 1\cdot2^{11}+1\cdot2^9+1\cdot2^5+1\cdot2^3+1\cdot2^2+1\cdot2^1+1\cdot2^0=$

$$= (2048+512)+(32)+(8+4+2+1)=2607_{10}$$

Za beleženje celih brojeva u memoriji računara koriste se reči koje su najčešće veličine 2 bajta (16 bitova) ili 4 bajta (32 bita). Veličina reči određuje i veličinu celog nenegativnog broja koji se može zabeležiti i sa kojim se može računati. Tako je najveći nenegativan ceo broj koji se može zabeležiti u reči veličine 2 bajta 2^{16}-1=65535 (svi brojevi iz intervala [0,65535]), a u reči veličine 4 bajta je 2^{32}-1=4294967295 (svi brojevi iz intervala [0, 4294967295]). Odatle sledi da je skup celih nenegativnih brojeva sa kojima računar može da računa ograničen, a to ograničenje je uslovljeno veličinom računarske reči.

Negativni celi brojevi za potrebe računanja bi takođe trebalo da budu predstavljeni u binarnom obliku. Jedna mogućnost za kodiranje označenosti broja binarnim alfabetom bila bi kodiranje znaka u bitu najveće težine. Znači da bi se u računarskoj reči od 16 bitova za beleženje apsolutne vrednosti broja koristilo 15 bitova, a jedan bit bi se koristio za znak. Tako bi, na primer, broj 77 u 16-bitnoj reči bio predstavljen binarno

0000 0000 0100 1101

dok bi broj -77 mogao biti predstavljen sa **1000 0000 0100 1101**. Ovim postupkom se simulira uobičajena upotreba znaka plus i minus kod zapisivanja brojeva u dekadnom brojnom sistemu.

Pokazuje se da je ovakvo predstavljanje označenih brojeva nepraktično za računarsku primenu jer se prilikom realizacije, na primer, operacije sabiranja mora voditi računa o predznaku sabiraka. Osim toga broj 0 bi se mogao predstaviti na dva načina: kao 0000 0000 0000 0000 („pozitivna nula“) i kao 1000 0000 0000 0000 („negativna nula“).

Postupak sabiranja se svodi na obavljanje sledećih koraka:

- ako su oba sabirka istog znaka, saberi njihove apsolutne vrednosti, a rezultat je istog znaka kao i sabirci;
- ako su sabirci različitog znaka, oduzmi manjeg po apsolutnoj vrednosti od većeg po apsolutnoj vrednosti, a rezultat ima znak sabirka koji je veći po apsolutnoj vrednosti.

Tako se sabiranje obavlja i u dekadnom brojnom sistemu. Na primer,

(+77) + (+13) = +90	oba broja imaju isti predznak (pozitivan +)
(-77) + (-13) = -90	oba broja imaju isti predznak (negativan -)
(+77) + (-13) = +64	brojevi imaju različiti predznak, od većeg po apsolutnoj vrednosti (77) se oduzima manji po apsolutnoj vrednosti (13), rezultat ima predznak većeg po apsolutnoj vrednosti (pozitivan +)
(-77) + (+13) = -64	brojevi imaju različiti predznak, od većeg po apsolutnoj vrednosti (77) se oduzima manji po apsolutnoj vrednosti (13), rezultat ima predznak većeg po apsolutnoj vrednosti (negativan -)

Da bi se izbegao ovaj relativno složeni postupak, negativni brojevi se u računarskoj reči predstavljaju u obliku **nepotpunog ili potpunog komplementa**.

Nepotpuni komplement:
Nepotpuni komplement broja u brojnom sistemu sa osnovom *N* dobija se ako se svaka cifra broja dopuni do najveće cifre brojčanog sistema, tj. do vrednosti *N*-1.

Na primer, ako je osnova brojnog sistema 10, onda je nepotpuni komplement broja 357 broj 642 (jer je 3+6=9, 5+4=9, 7+2=9). Ako je osnova brojnog sistema 2, onda je nepotpuni komplement broja 0100110 broj 1011001.

Potpuni komplement:
Potpuni komplement broja u pozicionom brojnom sistemu sa osnovom *N* dobija se tako što se najpre odredi nepotpuni komplement broja, a zatim se doda jedinica na poziciji najmanje težine. Na primer, ako je osnova brojnog sistema 10, onda je nepotpuni komplement broja 357 broj 642, a potpuni komplement broj 643. Ako je osnova brojnog sistema 2, onda je nepotpuni komplement broja 0100110 broj 1011001, a potpuni komplement broj 1011010.

Prednost predstavljanja brojeva za potrebe računanja u obliku nepotpunog ili potpunog komplementa u odnosu na predstavljanje brojeva u obliku apsolutne vrednosti i znaka ilustrovaćemo na primeru operacije sabiranja.

Sabiranje brojeva u nepotpunom komplementu:
Sabiranjem brojeva u nepotpunom komplementu dobija se rezultat u nepotpunom komplementu primenom sledeća dva koraka:

1. Saberu se dva broja uključujući mesta za znak broja i odredi se međurezultat.
2. Ukoliko ima prenosa u međurezultatu on se odbacuje iz međurezultata i dodaje na poziciji najmanje težine međurezultata.

Primeri (recimo da se svi brojevi beleže sa 5 dekadnih cifara, peta cifra onda određuje predznak broja – 0 na poziciji 5 znači da je broj pozitivan, 9 na poziciji 5 znači da je broj negativan):
276 + (-372) = -96
(broj -372 je negativan pa se on predstavlja u obliku nepotpunog komplementa)
00276 + 99627 = 99903 (nema prekoračenja)
99903 $\Rightarrow$ -96
(na osnovu pojave najveće cifre brojnog sistema – 9 – na poziciji najveće težine zaključujemo da je to negativan broj u nepotpunom komplementu što bi odgovaralo broju -96, jer je 0+9=9 i 3+6=9)

276 + (-92) = 184
00276 + 99907 = 100183 (ima prekoračenja)
183 + 1 = 184
(na osnovu pojave 0 na poziciji najveće težine – 5 – zaključujemo da je broj s pozitivnim predznakom).

Sabiranje brojeva u potpunom komplementu:
Sabiranjem brojeva u potpunom dobija se rezultat u potpunom komplementu primenom sledeća dva koraka:

3. Saberu se dva broja uključujući mesta za predznak broja i odredi se međurezultat.
4. Ukoliko ima prenosa u međurezultatu on se odbacuje iz međurezultata i tako dobijeni broj predstavlja rezultat sabiranja u potpunom komplementu.

Primeri (recimo da se svi brojevi beleže sa 5 dekadnih cifara):
276 + (-372) = -96
00276 + 99628 = 99904 (nema prekoračenja)
99904 $\Rightarrow$ -(00095 + 1) = -96

(na osnovu pojave najveće cifre brojnog sistema – 9 – na poziciji najveće težine zaključujemo da je to negativan broj u potpunom komplementu što bi odgovaralo broju -96)

276 + (-92) = 184
00276 + 99908 = 100184 (ima prekoračenja)
(1)00184 = 184

Na osnovu ovih primera vidimo da je sabiranje u potpunom komplementu jednostavnije pa se stoga za potrebe računanja celi brojevi u računaru predstavljaju u obliku potpunog komplementa.

Potpuni komplement binarnog broja dobija se zamenom svake cifre u zapisu broja onom drugom cifrom (tj. zamenom svake 0 sa 1 i zamenom svake 1 sa 0) te dodavanjem 1 na tu vrednost. Na primer, potpuni komplement brojeva 77 i 13 bio bi ($77=64+8+4+1=2^6+2^3+2^3+2^0$; $13=8+4+1=2^3+2^3+2^0$):

dec.	**binarno**	**nep. komplement**	**+1 (pot. komplement)**
77	0000 0000 0100 1101	1111 1111 1011 0010	1111 1111 1011 0011
13	0000 0000 0000 1101	1111 1111 1111 0010	1111 1111 1111 0011

Može se videti da je ovakvo predstavljanje negativnih vrednosti pogodno iz više razloga:

- računanje potpunog komplementa je sa stanovišta računara jednostavna operacija;
- Kada se dva puta primeni operacija „potpuni komplement“ na neku binarnu vrednost dobija se ista, početna vrednost. Time je zadovoljen uslov da za svaki ceo broj treba da važi $-(-a) = a$. Na primer, potpuni komplement broja 372 je 99627+1=99628, a ako ponovo primenimo operaciju potpuni komplement dobijamo 00371+1= 00372, tj. početni broj. Isto je i u slučaju binarnih brojeva, npr. za -77, izračunat na osnovu binarnog zapisa **1111 1111 1011 0011** biće **0000 0000 0100 1100 + 1 =0000 0000 0100 1101** a to je početna binarna vrednost;
- Nula se zapisuje na jedinstven način, to jest -0=0. Naime, potpuni komplement broja **0000 0000 0000 0000** je **1111 1111 1111 1111 + 1 = (1) 0000 0000 0000 0000**. Prenos koji se pojavljuje iza bita najveće težine, to jest izvan opsega računarske reči, odbacuje se;
- Predznak broja se može jednostavno utvrditi testiranjem bita najveće težine.

- Sabiranje binarnih vrednosti, bile one pozitivne ili negativne, obavlja se na jedinstven način koristeći jednostavnu operaciju binarnog sabiranja, što se može videti na primeru različitih predznaka brojeva 77 i 13:

77+13=00077+00013=00090
(-77)+(-13)=99923+99987=(1)99910 $\Rightarrow$-(00089+1)=-90
77+(-13)=00077+99987=(1)00064=64
(-77)+13=99923+00013=99936 $\Rightarrow$-(00063+1)=-64

Isto to u binarnom brojnom sistemu je:

	Prvi sabirak	**Drugi sabirak**	**rezultat**	**Dec.**
77+13	0000000001001101	+0000000000001101	0000000001011010	90
			$2^6+2^4+2^3+2^1=64+16+8+2$	
-77+(-13)	1111111110110011	+1111111111110011	(1)1111111110100110	-90
			PK 0000000001011001+1	
77+(-13)	0000000001001101	+1111111111110011	(1)0000000001000000	64
			$2^6=64$	
(-77)+13	1111111110110011	+0000000000001101	(1)1111111111000000	-64
			PK 0000000000111111+1	

I ovde se prenos koji se pojavljuje iza bita najveće težine odbacuje.

Na ovaj način, od ukupno 2^{16}=65536 različitih binarnih vrednosti koje se mogu predstaviti u reči dužine 16 bitova polovina otpada na negativne vrednosti a druga polovina na nenegativne vrednosti pa je opseg celih brojeva koji se mogu predstaviti u reči od 16 bitova [-32768,+32767]. Opseg brojeva koji se mogu predstaviti u reči dužine 32 bita je [-2147483648,+2147483647].

Predstavljanje realnih brojeva u binarnom brojnom sistemu

Način reprezentovanja celih brojeva koji je ovde predstavljen primenjuje se na većini današnjih računara. Što se tiče realnih brojeva, način njihovog reprezentovanja, premda sledi neke osnovne principe, može se znatno razlikovati od jednog do drugog tipa računara. Realni brojevi beleže se najčešće u rečima dužine 32 bita (4 bajta) ili u rečima dužine 64 bita (8 bajtova). Oni se obično registruju u obliku *pokretnog zareza* (engl. floating-point notation) koji omogućava da se vrlo veliki i vrlo mali brojevi podesno izraze. Opšti oblik broja u pokretnim zarezu je

$\pm m \times R^e$

gde je m normalizovana mantisa, R osnova ili baza, a e je eksponent. Za normalizovanu mantisu važi da je $0.1 \leq m < 1$. Na drugi način se to može izraziti na sledeći način:

$m = 0, x_{-1}\, x_{-2}\, x_{-3} \ldots\, x_{-n}$

gde je su x_{-i} cifre iz decimalnog zapisa broja za koje važi da je $x_{-1} \neq 0$.

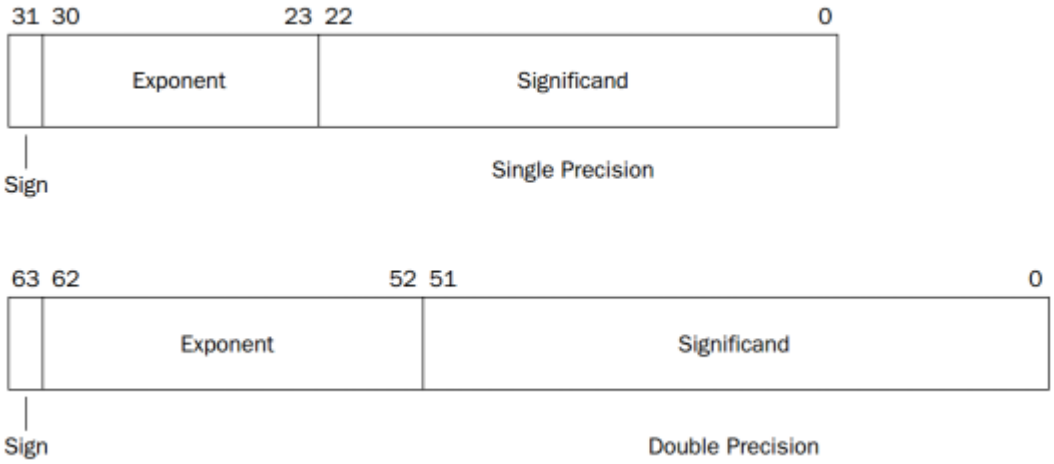
Na primer, u dekadnom brojnom sistemu bi se brojevi predstavili u pokretnom zarezu na sledeći način:

1023.	0.1023×10^4	(0.1023E+4)
0.0025	0.25×10^{-2}	(0.25E-2)
1023.0025	0.10230025×10^4	(0.10230025E+4)

Još neki primeri brojeva zapisanih u nepokretnom i pokretnom zarezu, dekadno i binarno, su:

Osnova	**Broj**	**nepokretni zarez**	**pokretni zarez**
10	12.045	+0012.045000	0.1204E+02
10	-0.000004	-0000.000004	-0.4000E-05
10	0.00000008	+0000.000000	+0.8000E-07
10	32560000	+0000.000000	+0.3256E+08
2	101.1	0101.100000	+0.1011E+11

U ovom primeru se pretpostavlja da se u nepokretnom zarezu uvek koriste 4 cifre za zapis broja ispred decimalne tačke (zareza), a šest cufara za zapis broja iza decimalne tačke (zareza) – otuda naziv „nepokretni zarez“. Treći i četvrti primer u gornjoj tabeli pokazuju da se u nepokretnom zarezu gube, tj. ne mogu se zapisati, sve značajne cifre broja, dok to nije slučaj u pokretnom zarezu.



Tipačan format za predstavljanje brojeva u pokretnom zarezu u računaru koristi bit najveće težine za označavanje predznaka mantise (a to je i predznak celog broja), za njim sledi fiksni broj bitova za predstavljanje eksponenta, za kojim opet sledi fiksni broj bitova koji predstavljaju mantisu. Od jednog do drugog formata razlikuje se broj bitova koji se koriste za predstavljanje eksponenta, odnosno mantise, kao i način njihovog predstavljanja. Na primer, mantisa može predstavljati binarni zapis broja (u tom slučaju je R=2), binarno kodiran dekadni zapis (u tom slučaju je R=10) ili binarno kodirani heksadecimalni zapis (u tom slučaju je R=16). Osim toga, za prestavljanje mantise i eksponenta ne mora se koristiti isti zapis (isti brojni sistem).

Na primer, današnji personalni računari najčešće koriste takozvani IEEE format za predstavljanje realnih brojeva. Radi se o standardu IEEE 754 koji[1], u stvari, propisuje pet različitih formata brojeva u pokretnom zarezu, od koji tri koriste mantisu zapisanu u binarnom brojnom sistemu (u 4, 8 i 16 bajtova), a dva binarno kodiranu decimalnu mantisu (u 4 i 8 bajtova).

Najčešće su u upotrebi zapisi sa binarnom mantisom u 4 i 8 bajtova. U ovom formatu se u 4-bajtnoj reči bit najveće težine koristi za predznak, sledećih 8 bitova za eksponent, a sledeća 23 bita za mantisu. Za zapis u 8-bajtnoj reči bit najveće težine koristi se za predznak, sledećih 11 bitova za eksponent, a 52 bita za mantisu. Realni brojevi koji se na ovaj način mogu predstaviti u 4-bajtnoj reči nalaze se po apsolutnoj vrednosti u opsegu $[0.29 \times 10^{-38}, 0.17 \times 10^{39}]$. U 8-bajtnoj reči mogu se predstaviti realni brojevi iz opsega $[0.55 \times 10^{-308}, 0.2 \times 10^{309}]$.

[1] http://en.wikipedia.org/wiki/IEEE_754-2008

	znak	**eksponent**	**Mantisa**
4 bajta	31. bit	23-30. bit	0-22. bit
8 bajtova	63. bit	52-62. bit	0-51 bit

Kao što se vidi na ovaj način se mogu predstaviti daleko veći brojevi nego što to omogućava predstavljanje celih brojeva u binarnom brojnom sistemu. Naime, najveći ceo broj koji se može predstaviti u 4-bajtnoj reči je reda veličine 10^{10} dok je najveći realni broj koji se može predstaviti reda veličine 10^{38}. Treba, međutim, uočiti da se u reči dužine n bitova može predstaviti **različitih binarnih vrednosti** ukupno 2^n, pa, ako se koristi binarni brojni sistem za predstavljanje celih brojeva u toj reči se može predstaviti ukupno 2^n različitih celih brojeva (od toga pola negativnih, pola nenegativnih), a ako se koristi zapis broja u pokretnom zarezu može se zabeležiti ukupno 2^n različitih realnih brojeva.

Postavlja se onda pitanje kako to da se u reči dužine, recimo, 32 bita mogu predstaviti realni brojevi iz daleko većeg intervala. Odgovor je sledeći: u slučaju celih brojeva, iz intervala $[-2^{31},2^{31}-1]$ predstavljaju se **svi** celi brojevi (njih ukupno 2^{32}) potpuno tačno. U slučaju realnih brojeva, iz navedenog intervala predstavljaju se samo **neki** realni brojevi, ali opet njih ukupno 2^{32}. Svi ostali realni brojevi iz tog intervala, kojih ima beskonačno mnogo, predstavljaju se približno, određenim brojem značajnih cifara. Broj značajnih cifara realnih brojeva koje se beleže u zapisu u pokretnom zarezu zavisi od veličine prostora za zapis mantise i iznosi za 4-bajtni IEEE zapis 6 do 7 cifara a za 8-bajtni IEEE zapis 15 cifara. Na primer, najveći ceo broj koji se može zabeležiti u 4-bajtnoj reči je +2147483647. Taj broj bi se predstavio u pokretnom zarezu u 4-bajtnoj reči na način koji odgovara decimalnom zapisu $+0.214748\times10^{10}$, a to znači da bi svi celi brojevi iz intervala [+2147480000,+2147490000], kao i beskonačno mnogo realnih brojeva iz tog istog intervala, bili u računaru predstavljeni tom jednom jedinom vrednošću.

Prilikom prevođenja realnih brojeva iz dekadnog u binarni zapis, može doći do toga da se dekadni realni broj sa konačnim brojem cifara prevede u binarni broj sa beskonačnim brojem cifara. tj. binarni broj može biti periodičan razlomljeni broj. Na primer, $0.4_{10}\approx 0.01100110..._2$, a to znači da realan broj sa konačnim brojem decimalnih cifara ne mora da bude tačno predstavljen u memoriji računara u formatu pokretnog zareza.

Predstavljanje brojeva binarnim kodiranjem dekadnih cifara

Realni brojevi se u memoriji računara za potrebe računanja predstavljaju približnom vrednošću. Takav način predstavljanja realnih brojeva je pogodan sa stanovišta naučnih i inženjerskih primena jer se računanje sa ovako predstavljenim realnim brojevima može efikasno implementirati, a računanje sa približnim vrednostima je sasvim adekvatno. Na primer, sasvim je dovoljno reći da je Mesec udaljen od Zemlje 384.400km (sa 4 značajne cifre) – podatak da je udaljen 384.437,28km uglavnom nije od značaja pogotovo ako je merenje obavljeno tako da se preciznost do na metar ne može postići.

S druge strane, za potrebe poslovnih primena, kakve su obračun plata, zarada, troškova i slično računanje sa približnim vrednostima nije adekvatno. Na primer, Pajinom ujka Baji nije beznačajno da li na svom kontu u banci ima 384.400$ ili 384.437,28$. Za poslovne primene se stoga realni brojevi predstavljaju u obliku binarno kodiranih dekadnih cifara (engl. *binary-coded decimal* - BCD). Ovakav način omogućava potpuno tačno predstavljanje realnih dekadnih vrednosti sa proizvoljno izabranim brojem cifara što se postiže na uštrb efikasnosti računanja. S druge strane, prevođenje brojeva iz karakterskog zapisa u binarno kodirani zapis i obrnuto, koje se mora obavljati za potrebe unosa podataka i izdavanje rezultata je efikasnije od prevođenja brojeva iz karakterskog u oblik potpunog komplementa ili zapis u pokretnom zarezu. Sve je to bitno za poslovne primene gde se učitava i ispisuje mnogo brojeva koje treba kodirati i dekodirati, ali se nad tim brojevima obavlja neki u principu jednostavan račun (u poređenju sa naučnim primenama).

Samo implementiranje binarno-kodiranih dekadnih cifara je pod kontrolom višeg programskog jezika[2] ili samog programera, ali se svodi uglavnom na to da se sve cifre broja, predznak broja kao i ukupan broj cifara i ukupan broj decimalnih cifara kodiraju dekadno u jednom bajtu (tada imamo *nepakovane binarno kodirane dekadne cifre*) ili u jednom polubajtu (tada imamo *pakovane binarno kodirane dekadne cifre*). Sam kod cifara može se razlikovati i može biti 8421, 2421, višak-3 itd. U narednim primerima koristiće se kod 8421.

Primer:

Nepakovani oblik broja +498,12 je									
z	4	z	9	z	8	z	1	pz+	2
0101	0100	0101	1001	0101	1000	0101	0001	**1010**	0010

[2] Na primer, programski jezici Cobol i PL/1 su podržavali binarno kodiranu aritmetiku.

Nepakovani oblik broja -498,12 je									
z	4	z	9	z	8	z	1	pz-	2
0101	0100	0101	1001	0101	1000	0101	0001	**1011**	0010

0101 – zonski deo broja (Ovaj deo broja može se razlikovati od jedne do druge implementacije ili primene; na primer F_{16} je zonski deo u EBCDIC kodu, dok je 3_{16} zonski deo u ASCII kodu. U slučaju EBCDIC-a zonska vrednost poslednje cifre pozitivnog broja je C_{16}, a negativnog D_{16}).
1010 – predznak +
1011 – predznak –

Pakovani oblik broja +498,12						**Pakovani oblik broja -498,12**					
4	9	8	1	2	pz+	4	9	8	1	2	pz-
0100	1001	1000	0001	0010	**1010**	0100	1001	1000	0001	0010	**1011**

Decimalni zarez se ne registruje u zapisu broja već se taj problem rešava tako što se odvojeno čuva dužina broja i broj decimalnih mesta. Treba uočiti da memorijsko zauzeće brojeva u BCD zapisu nije fiksno već zavisi od ukupnog broja cifara.

Sabiranje (i obavljanje ostalih računske operacije) uopšte nije jednostavno i mnogo je manje efikasno nego kada se koristi zapis u fiksnom zarezu (potpuni komplement) i pokretnom zarezu. Složenost obavljanja ovih operacija ćemo ilustrovati na primeru operacije sabiranja i kodova 8421 i višak 3 za kodiranje pojedinačnih cifara.

Sabiranje brojeva u kodu 8421

broj 389 + 129 = 518
0011 1000 1001
0001 0010 1001

običnо binarno sabiranje

0100 1011 0010 4B2 - međurezultat koji treba doterati
0000 0110 0110 na poziciji najmanje težine smo imali prenos (zbog sabiranja cifara 9+9=18) koji je 16, ali dekadni prenos 10 na toj poziciji treba dodati $6=0110_2$ (da bi nadoknadili „manjak“ koji je nastao jer je heksa prenos samo 2) i ostaviti prenos; na sledećoj poziciji nemamo binarni (heksa) prenos jer se sabira 8+2+**1**=11 ali bi trebalo da imamo dekadni (jer je rezultat veći od $1001_2=9_{10}$) pa opet

dodajemo $6=0110_2$ čime ćemo dobiti 17, odnosno 1 sa prenosom 1, što nam i treba.

0101 0001 1000
5 1 8

Dakle, korekcija se vrši za one cifre gde je postojao binarni (tj. heksa) ili dekadni prenos, što baš i nije lako da se testira.

Sabiranje brojeva u kodu višak-3

Sabiranje je nešto jednostavnije kada se koristi kod višak 3. Pogledajmo prvo kako se sabiraju pojedinačne cifre koje su kodirane u kodu višak 3. Ako rezultat sabiranja cifara nema prenosa u dekadnom sistemu neće ga imati ni u heksadekadnom:

3 + 6 = 9
0110 (kod cifre 3 ima vrednost binarnog 6)
1001 (kod cifre 6 ima vrednost binarnog 9)

1111 (15=9+6 umesto 12, jer „višak“ potiče od oba sabirka)
-0011 (oduzima se 3 viška)

1100 (12=9+3)

Ako rezultat sabiranja cifara ima prenos u dekadnom sistemu imaće ga i u heksadekadnom:
2 + 8 = 10
0101 (kod cifre 2 ima vrednost binarnog 5)
1011 (kod cifre 8 ima vrednost binarnog 11)
---
(1)0000 (prenos 1 i -3 umesto prenos 1 i 0, jer je „višak“ od oba sabirka otišao na popunjavanje praznine od 6 između dekadnog i heksa kodiranja.)
+0011
---
(1)0011 (prenos 1 i 0)

Dakle, postoji korekcija kod svih cifara: tamo gde nema heksa prenosa oduzima se 3 (0011), a tamo gde ima prenosa dodaje se 3 (0011).

broj 389 + 129 = 518

0110	1011	1100	
0100	0101	1100	
			običnо binarno sabiranje
1011	0001	1000	B18 - međurezultat koji treba doterati
-0011	+0011	+0011	
			pogodnost koda višak-3 je da se heksadekadni prenos poklapa sa decimalnim prenosom. Kad god imamo dekadni prenos kod sabiranja u rezultatu nedostaje višak 3 koji treba dodati, a kad nemamo prenos u rezultatu je jedan višak 3 viška i njega treba oduzeti
1000	0100	1011	
5	1	8	

Još jedan primer: 527+658=1185

	1000	0101	1010	
	1001	1000	1011	
0001	0001	1110	0101	međurezultat
+0011	+0011	-0011	+0011	
0100	0100	1011	1000	podešavanje
1	1	8	5	

Rad sa negativnim brojevima u kodu višak 3 se obavlja tako što se broj koji se oduzima predstavlja u potpunom komplementu dekadno, a zatim u kodu višak 3. Dalje obavljanje sabiranja ide na isti način kao i kod sabiranja pozitivnih vrednosti.

389 – 129 = 260

0110	**1011**	**1100**	**00389**	prvi sabirak
0100	*0101*	*1100*	*129*	
1011	1010	0011	99870	
		+1	+1	
1011	**1010**	**0100**	**99871**	drugi sabirak
---	---	---	---	
(1)00100	(1)0110	(1)0000	(1)00260	međurezultat
+11	+11	+11		

0101 1001 0011 260 rezultat

Primer jednog testa

1. (a) Navesti koji su poznati binarni kodovi dekadnih cifara težinski:

   (b) Šta znači da je binarni kod dekadnih cifara komplementaran?

   (c) Navesti jedan komplementaran binarni kod dekadnih cifara?

   (d) Navesti kodove dekadnih cifara u cikličnom kodu:

2. Zapisati brojeve +382 i -72 u obliku:

   (a) nepotpunog komplementa:

   (b) potpunog komplementa:

   (c) pokretnog zareza:

   Napomena: zapis broja i dalje treba da bude dekadan. Za nepotpuni i potpuni komplement koristiti zapis broja sa 6 cifara.

3. Prikazati u obliku pokretnog zareza sa 4 decimalne cifre sledeće brojeve:

(a) 0,0023567 (b) 2345,2123
(c) 2345,2987 (d) 0,00023567
(e) 0,0023569 (f) 0,2345

Da li će neki od ovih brojeva imati isti zapis (koji)?

Rešenja

1. (a) Težinski kodovi su kôd 8421 i kôd 2421.
   (b) To znači da ako su dve cifre komplementarne u dekadnom brojnom sistemu biće komplementarne i njihove binarne reprezentacije, tj. ako su dve dekadne cifre takve da je $X+Y=9$ i ako je $x_3x_2x_1x_0$ binarni kôd cifre X a $y_3y_2y_1y_0$ binarni kôd cifre Y onda i za te binarne cifre važi $x_i+y_i=1$, za $1 \leq i \leq 4$.
   (c) Komplementaran je kôd 2421 i kôd višak 3.
   (d) Cifre u cikličnom kôdu su: 0001, 0101, 0111, 1111, 1110, 1100, 1000, 1001, 1011, 0011.
2. (a) +382 = 000382, -72 = 999927
   (b) +382 = 000382, -72 = 999927 + 1 = 999928
   (c) $+382 = +0.382*10^3$, $-72 = -0.72*10^2$

3. Prikazati u obliku pokretnog zareza sa 4 decimalne cifre sledeće brojeve:
   (a) $0{,}0023567 = 0{,}2357 * 10^{-2}$ (b) $2345{,}2123 = 0{,}2345 * 10^{4}$
   (c) $2345{,}2987 = 0{,}2345 * 10^{4}$ (d) $0{,}00023567 = 0{,}2357 * 10^{-3}$
   (e) $0{,}0023569 = 0{,}2357 * 10^{-2}$ (f) $0{,}2345 = 0{,}2345 * 10^{0}$
   Da li će neki od ovih brojeva imati isti zapis (koji)? Da, brojevi pod (a) i (e) i brojevi pod (b) i (c) jer imaju iste 4 značajne cifre (posle zaokruživanja) i istog su reda veličine.

Primer jednog zadatka sa ispita

Dati su brojevi $\boldsymbol{x} = 101$ i $\boldsymbol{y} = -124$.

a) Koristeći **notaciju potpunog binarnog komplementa**, odrediti $\boldsymbol{x} + \boldsymbol{y}$ i $\boldsymbol{y} - \boldsymbol{x}$;
b) Dobijene rezultate predstaviti u **BCD** notaciji (pakovanom binarno-kodiranom zapisu, i to samo cifre).
c) Broj $(-2048) * \boldsymbol{x} * \boldsymbol{y}$ predstaviti u dekadnom brojnom sistemu u notaciji pokretnog zareza sa **normalizovanom mantisom**.

Rešenje:

a) $[101]_{PK} = 6 * 16 + 5 = [65]_{16} = 0\ 0110\ 0101_2$
$[124]_{PK} = 7 * 16 + 12 = [7C]_{16} = 0\ 0111\ 1100_2$
$[-101]_{PK} = PK([101]_{PK}) = NK([101]_{PK}) + 1 = NK(0\ 0110\ 0101_2) + 1 =$
$= 1\ 1001\ 1010_2 + 1 = 1\ 1001\ 1011_2$
$[-124]_{PK} = PK([124]_{PK}) = NK([124]_{PK}) + 1 = NK(0\ 0111\ 1100_2) + 1 =$
$= 1\ 1000\ 0011_2 + 1 = 1\ 1000\ 0100_2$

$x + y = [101 + -124]_{PK} =$ $[101]_{PK}$ $0\ 0110\ 0101_2$
$[-124]_{PK}$ $1\ 1000\ 0100_2$
$+$
-23 $1\ 1110\ 1001_2$

Provera: $[-23]_{PK} =$
$= PK([23]_{PK}) = NK([23]_{PK}) + 1 = NK(0\ 0001\ 0111_2) + 1 =$
$= 1\ 1110\ 1000_2 + 1 = 1\ 1110\ 1001_2$

$y - x = [-124 - 101]_{PK} =$ $[-124]_{PK}$ $1\ 1000\ 0100_2$
$[-101]_{PK}$ $1\ 1001\ 1011_2$
$+$
-225 $1\ 0001\ 1111_2$

Provera: $[-225]_{PK}$ =
= $PK([225]_{PK}) = NK([225]_{PK}) + 1 = NK(0\ 1110\ 0001_2) + 1 =$
= $1\ 0001\ 1110_2 + 1 = 1\ 0001\ 1111_2$

b) BCD(-23) = 0010 0011
BCD(-225) = 0010 0010 0101
c) Normalizovana mantisa
(-2048)*$\boldsymbol{x}$*$\boldsymbol{y}$ = (-2048) * 101 * (-124) = 25649152 = 0.25649152* 10^8