1. Brojni sistemi, prevođenje brojeva i predstavljanje podataka u računaru

1.1. Brojni sistemi

Brojni sistemi (numeracije) predstavljaju skupove znakova (simbola) kao i pravila njihovog korišćenja za predstavljanje brojeva. Možemo reći da brojni sistemi predstavljaju notaciju za predstavljanje brojeva odnosno definisani način izražavanja i označavanja. Znaci (simboli) koji se koriste za prikazivanje brojeva zovu se brojke ili cifre.

Danas postoji veliki broj različitih brojnih sistema koji su u upotrebi. U svakodnevnom životu najčešće se koristi dekadni brojni sistem[1]. U ovom sistemu svaki broj A se predstavlja kao niz cifara $A = \overline{a_n a_{n-1} a_{n-2} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}}$ gde su $a_{-m}, a_{-m+1}, \dots, a_n \in \{0, 1, \dots, 9\}$ i važi:

$$A = \sum_{i=-m}^{n} a_i \cdot 10^i = a_{-m} \cdot 10^{-m} + \dots + a_{-1} \cdot 10^{-1} + a_0 + a_1 \cdot 10 + \dots + a_n \cdot 10^n$$

Deo broja ispred decimalne tačke $\overline{a_n a_{n-1} a_{n-2} \dots a_1 a_0}$ naziva se ceo deo broja dok je deo iza decimalne tačke razlomljeni deo broja A. Ovaj brojni sistem je primer pozicionog brojnog sistema. Osnovna karakteristika pozicionih brojnih sistema je da vrednost (udeo) sa kojom svaka cifra učestvuje u ukupnoj vrednosti broja zavisi od pozicije na kojoj se cifra nalazi. Ukoliko to nije slučaj, brojni sistem je nepozicioni. Kao primer nepozicionog brojnog sistema možemo navesti pisanje brojeva pomoću rimskih cifara.

U opštem slučaju, kod pozicionih brojnih sistema, za svaku poziciju k u zapisu broja treba zadati težinu t_k sa kojom ta cifra učestvuje u ukupnoj vrednosti broja. Vrednost broja se tada računa na sledeći način:

$$A = \overline{a_n a_{n-1} a_{n-2} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}}, \qquad A = \sum_{i=-m}^{n} a_i \cdot t_i$$

Najčešće su u upotrebi brojni sistemi kod kojih je $t_k = r^k$ gde je r dati broj. Ovakav sistem se naziva prirodni brojni sistem ili sistem sa osnovom r. Dekadni sistem je specijalan slučaj ovog sistema i dobija se za $r = 10$. Osim dekadnog u računarstvu su u upotrebi i sledeći sistemi:

1. Binarni, $r = 2$
2. Oktalni, $r = 8$
3. Heksadekadni, $r = 16$

Kod heksadekadnog sistema, za označavanje cifara koje imaju vrednost 10, 11, 12, 13, 14 i 15 koristićemo redom slova A, B, C, D, E i F. Brojeve predstavljene u sistemu sa osnovom r pisaćemo na sledeći način:

$$A = (a_n a_{n-1} a_{n-2} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m})_r$$

Ovim načinom pisanja eksplicitno naglašavamo o kojoj se osnovi radi. Za brojeve u dekadnom brojnom sistemu nećemo pisati osnovu i predstavljaćemo ih na uobičajen način.

U nastavku ćemo detaljnije proučavati isključivo prirodne brojne sisteme. Pritom ćemo najčešće razmatrati binarni, oktalni, heksadekadni i naravno dekadni sistem. Isključivo ovi brojni sistemi se danas koriste u računarstvu (a i ranije su se isključivo oni koristili)[2].

[1] Koristimo ga iz istorijskih razloga, a u upotrebu je ušao veoma davno, najverovatnije zbog deset prstiju na rukama. Postoje zabeleške i o brojnim sistemima sa osnovom 20, a na primer, recimo u Mezopotamiji su ljudi koristili sistem sa osnovom 60.

Primer. Broj 1234567 u heksadekadnom sistemu predstavlja se kao $(12D687)_{16}$, u oktalnom $(4553207)_8$ a u binarnom $(100101101011010000111)_2$.

Pravila za sabiranje, množenje i deljenje brojeva mogu se lako generalisati iz dekadnog u bilo koji prirodni brojni sistem. Naravno, odgovarajuće tablice sabiranja i množenja moraju se posebno generisati za svaki sistem. Tablice množenja i sabiranja za binarni, oktalni i heksadekadni sistem dati su u prilogu na kraju ovog odeljka.

Zadatak 1. Izračunati:

1. $(10110)_2 + (10111)_2$
2. $(10110)_2 - (10011)_2$
3. $(762)_8 \cdot (45)_8$
4. $(1A)_{16} \cdot (2B)_{16}$

Rešenje:

1.
```
Prenos:101100
        10110
     +  10111
       101101
```

2.
```
  Zajam:101100
        10110
     -  10011
        00011
```

3.
```
762 * 45 = 4672
         3710
        43772
```

4.
```
1A * 2B  = 11E
        34
       45E
```

1.2. Prevođenje brojeva

Razmotrićemo tri načina za prevodjenje broja iz sistema sa osnovom r_1 u sistem sa osnovom r_2. Kod prvog se operacije izvršavaju u sistemu sa osnovom a kod drugog u sistemu sa osnovom . Treći način je specifičan, ali je primenljiv samo kada važi $r_2 = r_1^k$ ili obrnuto $r_1 = r_2^k$ za neki ceo broj k.

Kod prvog metoda, najpre se sve cifre broja A u sistemu sa osnovom r_1 kao i vrednost same osnove r_1 prevedu u sistem sa osnovom r_2. Reprezentacija broja A u sistemu sa osnovom r_2 dobija se kao vrednost sledećeg izraza:

$$A = \sum_{i=-m}^{n} a_i r_1^i = \sum_{i=-m}^{n} (a_i)_{r_2} \ (r_1)_{r_2}^i$$

Pošto smo navikli da radimo u dekadnom sistemu, ovaj metod se primenjuje kada se broj iz nekog drugog prevodi u dekadni sistem.

Kod drugog metoda na različite načine prevodimo ceo i razlomljeni deo broja A. Razmotrimo najpre prevodnjenje celog dela:

$$\lfloor A \rfloor = a_0 + a_1 r_1 + \cdots + a_n r_1^n = b_0 + b_1 r_2 + \cdots + b_N r_2^N$$

Ukoliko broj $\lfloor A \rfloor$ podelimo sa r_2, dobijamo količnik:

$$q_1 = b_1 + b_2 r_2 + \cdots + b_{N-1} r_2^{N-1}$$

i ostatak b_0. Ako sada novodobijeni količnik ponovo podelimo sa r_2 dobijamo količnik:

[2] Prvi elektronski računar, ENIAC, (koji je proradio na Univerzitetu Ilinois, SAD, u julu 1946.) je koristio dekadni brojni sistem. Za pamćenje jedne cifre bilo mu je potrebno 10 elektronskih cevi, od kojih je samo jedna radila u jednom trenutku.

$$q_2 = b_2 + b_3 r_2 + \cdots + b_{N-2} r_2^{N-2}$$

i ostatak b_1. Nastavljajući ovaj postupak dobijamo sve cifre u reprezentaciji broja $\lfloor A \rfloor$ u sistemu sa osnovom r_2. Naravno, sve navedene operacije deljenja vršimo u sistemu sa osnovom r_1. Na početku je potrebno prevesti osnovu r_2 u sistem sa osnovom r_1 a na kraju dobijene vrednosti cifara b_i u sistem sa osnovom r_2. Razlomljeni deo prevodimo vrlo slično. Imamo da je:

$$\{A\} = a_{-1} r_1^{-1} + a_{-2} r_1^{-2} + \cdots + a_{-m} r_1^{-m} = b_{-1} r_2^{-1} + b_{-2} r_2^{-2} + \cdots + b_{-m} r_2^{-m}$$

Ako sada pomnožimo $\{A\}$ sa r_2 dobijamo:

$$r_2\{A\} = b_{-1} + b_{-2} r_2^{-1} + \cdots + b_{-m} r_2^{-m+1}$$

Vidimo da je ceo deo predhodnog izraza upravo jednak $b_{-1} = \lfloor r_2\{A\} \rfloor$ dok je razlomljeni deo:

$$\{r_2\{A\}\} = b_{-2} r_2^{-1} + b_{-3} r_2^{-2} + \cdots + b_{-m} r_2^{-m+1}$$

Nastavljajući dalje ovaj postupak redom dobijamo ostale cifre b_{-i}. Napomenimo da cifre b_i celog dela broja A dobijamo u obrnutom redosledu dok cifre b_{-i} dobijamo u pravom redosledu. Ovaj metod je pogodan kada iz dekadnog sistema treba prevesti broj u neki drugi sistem jer se operaciju deljenja čovek mnogo lakše izvodi u dekadnom nego u nekom drugom brojnom sistemu.

Ukoliko je $r_2 = r_1^k$ ili obrnuto $r_1 = r_2^k$ za neki ceo broj k tada se prevodjenje može znatno uprostiti primenom trećeg metoda. Predpostavimo da je $r_2 = r_1^k$. Tada se grupisanjem po k uzastopnih cifara broja A u sistemu sa osnovom r_1 dobija jedna cifra u sistemu sa osnovom r_2, odnosno važi:

$$\lfloor A \rfloor = a_0 + a_1 r_1 + \cdots + a_n r_1^n = \left(a_0 + a_1 r_1 + \cdots + a_{k-1} r_1^{k-1}\right) + \left(a_k + a_{k+1} r_1 + \cdots + a_{2k} r_1^{k-1}\right) r_2 + \cdots$$

Zadatak 2. Primenom metoda za prevodjenje brojeva kod koga se operacije izvršavaju u dekadnom sistemu, prevesti broj $(ABC)_{16}$ iz heksadekadnog u dekadni sistem.

Rešenje:

Najpre se sve cifre datog broja, kao i osnova prevedu u dekadni sistem. Onda se zamenom i direktnim izračunavanjem u dekadnom brojnom sistemu dobija tražena reprezentacija broja:

$$A = (ABC)_{16} = 10 \cdot 16^2 + 11 \cdot 16 + 12 = 2748$$

Zadatak 3. Prevesti broj 193 iz dekadnog u oktalni sistem izvodeći pritom operacije u oktalnom sistemu.

Rešenje:

$$A = 193 = 1 \cdot 10^2 + 9 \cdot 10 + 3 = (1)_8 \cdot (12)_8^2 + (11)_8 \cdot (12)_8 + (3)_8$$
$$= (144)_8 + (132)_8 + (3)_8 = (301)_8$$

Zadatak 4. Prevesti broj $A = 42$ u binarni brojni sistem. Pritom izvoditi operacije u dekadnom brojnom sistemu.

Rešenje:

$42 : 2 = 21 \quad (0)$
$21 : 2 = 10 \quad (1)$
$10 : 2 = 5 \quad (0)$
$5 : 2 = 2 \quad (1)$
$2 : 2 = 1 \quad (0)$
$1 : 2 = 0 \quad (1)$

Prema tome, $42 = (101010)_2$.

Zadatak 5. Prevesti broj $A = 0.312$ u oktalni brojni sistem. Pritom izvoditi operacije u dekadnom brojnom sistemu. Naći prvih 7 cifara u reprezentaciji broja A u oktalnom sistemu

Rešenje:

$0.312 * 8 = 2.496$ (2)
$0.496 * 8 = 3.968$ (3)
$0.968 * 8 = 7.744$ (7)
$0.744 * 8 = 5.952$ (5)
$0.952 * 8 = 7.616$ (7)
$0.616 * 8 = 4.928$ (4)
$0.928 * 8 = 7.424$ (7)

$A = (0.2375747 \dots)_8$

Zadatak 6. Broj $(534)_8$ najpre u binarni a zatim i u heksadekadni brojni sistem.

Rešenje:

Svaku oktalnu cifru datog broja prevedemo u binarni sistem i dobijene binarne brojeve jednostavno nadovežemo jedan na drugi. Da bi preveli broj u heksadekadni sistem, binarne cifre grupišemo u grupe od po 4 cifre. Svakoj grupi odgovara po jedna heksa cifra.

$(5)_8 = (101)_2, (3)_8 = (011)_2, (4)_8 = (100)_2$
$(534)_8 = (101\ 011\ 100)_2 = (101011100)_2 = (0001\ 0101\ 1100)_2 = (15A)_{16}$

1.3. Predstavljanje celih brojeva u računaru

Za predstavljanje celih brojeva u računaru koristimo binarni sistem. Pomoću n bita[3] (binarnih cifara), možemo predstaviti sve cele brojeve od 0 do $2^n - 1$ (najveći broj koji možemo predstaviti pomoću n bita je $(11 \dots 11)_2$). Ovaj način koristimo za predstavljanje pozitivnih (neoznačenih) brojeva.

Ukoliko želimo da predstavimo i negativne (označene) brojeve, prvi bit u reprezentaciji se koristi za kodiranje znaka broja. Postoje ukupno 3 načina za predstavljanje označenih brojeva.

1.3.1. Direktno kodiranje bita znaka

Kod ovog načina se na poziciji bita znaka upisuje 0 ukoliko je broj pozitivan a 1 ukoliko je negativan. Ostalih $n - 1$ bitova se koristi za predstavljanje apsolutne vrednosti broja u binarnom sistemu. Ovaj način je sigurno najjednostavniji za predstavljanje, ali se osnovne aritmetičke operacije nad ovako predstavljenim brojevima teško izvršavaju.

1.3.2. Nepotpuni komplement

Cilj je izvršiti takvo predstavljanje označenih brojeva da se aritmetičke operacije mogu izvršavati na sličan način kao i kod neoznačenih brojeva. Neka je dat broj A čija se apsolutna vrednost može predstaviti u binarnom sistemu pomoću n cifara (bitova). Kao i kod direktnog kodiranja i ovde će biti potrebno ukupno

[3] Fraza "binarna cifra" se u računarstvu veoma često koristi. Zato je izmišljena sada već standardna skraćenica *bit*. Skraćenica je nastala od engleskog *binary digit*. Inače, *bit* je reč književnog engleskog jezika i znači *delić, parčence.*

$n+1$ binarnih cifara. Za predstavljanje bita znaka se i ovde koristi isti način kao i kod direktnog kodiranja. Zatim se u binarnoj reprezentaciji apsolutne vrednosti broja $|A|$ zamene cifre 0 i 1 na svakoj poziciji. Tako dobijamo preostalih n bita. O predstavlja se (u binarnom sistemu) sledeći broj:

$$\tilde{A} = \begin{cases} A, & A \geq 0 \\ 2^{n+1} - |A| - 1, & A < 0 \end{cases}$$

Praktično se $\tilde{A}$ dobija tako što se u binarnoj reprezentaciji broja $|A|$ zamene cifre 0 i 1 na svakoj poziciji. Pomoću ukupno $n+1$ bita, na ovaj način je moguće predstaviti sve brojeve u intervalu $[-2^n + 1, 2^n - 1]$.

Sabiranje ovako predstavljenih označenih brojeva se obavlja na potpuno isti način kao da su u pitanju neoznačeni brojevi (pritom i bit znaka učestvuje ravnopravno u sabiranju) pri čemu se eventualni prenos na najvišem bitu dodaje zbiru.

Definicija nepotpunog komplementa može se proširiti na bilo koju osnovu r.

Primer: Izvršimo sabiranje brojeva $(10011)_2$ i $-(10001)_2$ (u dekadnom sistemu su to brojevi 19 i -17). Najpre predstavljamo ove brojeve u nepotpunom komplementu a zatim sabiramo ovako dobijene binarne brojeve:

$(10011)_2 = 010011$
$-(10001)_2 = 101110$

```
  010011
  101110
1 000001
       1
  000010
```

Vodeći bit rezultata je 0 pa zaključujemo da je rezultat pozitivan i jednak 2.

Da se u predhodnom primeru dobio bit znaka 1, bilo bi potrebno pronaći apsolutnu vrednost rezultata, tj. dekomplementirati rezultat. Ukoliko je poznat nepotpuni komplement $\tilde{A}$ broja A, apsolutna vrednost $|A|$ se dobija ponovnom primenom operacije komplementiranja na broj $\tilde{A}$, odnosno važi $\tilde{\tilde{A}} = A$.

Zadatak 7. Izračunati vrednost izraza $323 - 14$. Najpre oba broja predstaviti u binarnom sistemu, zatim drugi broj komplementirati i tako dobijene brojeve sabrati.

Rešenje:

$323 = (101000011)_2$
$14 = (1110)_2$

Primetimo da prvi broj ima 9 cifara u binarnom sistemu a drugi 4. Najpre moramo oba broja dovesti na isti broj cifara, što znači da drugom broju dodajemo još 5 nula plus još jednu za bit znaka, dok prvom broju samo nulu za bit znaka. Prema tome, oba broja sad imaju po 10 binarnih cifara:

```
  0 101000011
  0 000001110
```

Sada vršimo komplementiranje drugog broja i sabiranje:

```
  0 101000011
+ 1 111110001
  0 100110100
+           1
  0 100110101
```

Prema tome rezultat je pozitivan i jednak $(100110101)_2 = 309$.

Zadatak 8. Odabrati proizvoljan broj x, predstaviti ga u binarnom sistemu i naći vrednost $x - x$ u nepotpunom komplementu. Prokomentarisati rezultat.

Rešenje:

Neka je $x = 11001$. Reprezentacija ovog broja u nepotpunom komplementu je 0 11001. Broj $-x$ dobijamo tako što invertujemo cifre u reprezentaciji broja x. Tako dobijamo 1 00110. Sada imamo da je:

```
  011001
  100110
  111111
```

Dakle dobili smo rezultat -0. Činjenica da se nula može predstaviti na dva načina u nepotpunom komplementu je glavni razlog zašto ovaj način predstavljanja označenih brojeva nije praktično primenljiv u računarstvu.

1.3.3. Potpuni komplement

Glavni nedostatak nepotpunog komplementa je taj što je moguće nulu predstaviti na 2 načina. Ovaj nedostatak rešava potpuni komplement (komplement osnove). Potpuni komplement se dobija kada se nepotpunom komplementu broja A doda 1:

$$\bar{A} = \tilde{A} + 1$$

Dakle, najpre se invertuju binarne cifre a zatim se na novodobijeni broj doda 1. U ovom komplementu nula može da se prikaže na jedinstven način (svi bitovi jednaki 0), a opseg brojeva koje je moguće na ovaj način predstaviti je $[-2^n, 2^n - 1]$. Ovim se postiže ravnomerna distribucija pozitivnih i negativnih brojeva.

Kod sabiranja u potpunom komplementu, eventualni prenos na najvišem bitu se zanemaruje. Napomenimo samo da se i ovaj komplement može generalisati za bilo koju osnovu r i da se vrednost potpunog komplementa broja A može definisati ekvivalentno na sledeći način:

$$\bar{A} = \begin{cases} A, & A \geq 0 \\ 2^{n+1} - |A|, & A < 0 \end{cases}$$

Primer. Broj -10110 predstaviti u potpunom komplementu:

Imamo da je 10110 u potpunom komplementu 010110 (dodali smo bit znaka koji je jednak nuli). Sada invertujemo sve cifre i dobijamo 101001. Ovim smo dobili broj u nepotpunom komplementu. Zatim dodajemo 1 i dobijamo broj u potpunom komplementu 101010.

Zadatak 9. Predstaviti sledeće brojeve u potpunom komplementu (u binarnom sistemu) pomoću $n = 10$ binarnih cifara (bitova): 1, -1, 0, -5, 1023, -1023, -1024.

Rešenje:

```
   1  =  0000000001          1023  =  0111111111
  -1  =  1111111111         -1023  =  1000000001
   0  =  0000000000         -1024  =  1000000000
  -5  =  1111110111
```

Zadatak 10. Izračunati vrednosti sledećih izraza predstavljajući brojeve u potpunom komplementu u binarnom sistemu:

a) $912 + 32 - 1120.$
b) $654 + 332 - 286.$
c) $-23 + 322 - 120.$

Rešenje:

a) Prevedimo najpre sve brojeve u binarni sistem.

```
 912 = 1110010000
  32 = 100000
1120 = 10001100000
```

Pošto 1120 ima ukupno 11 binarnih cifara a ostali brojevi manje radićemo sa ukupno 12 bita. Sada predstavljamo brojeve u potpunom komplementu i vršimo sabiranje.

```
  001110010000
  000000100000
+ 101110100000
  111101010000
```

Dobili smo negativan broj pa vršimo dekomplementiranje. Apsolutna vrednost rezultata je 000010110000 odnosno 176, pa je rezultat -176.

Zadatak 11. Pokazati da se primenom operacije komplementiranja na broj A prikazan u potpunom komplementu dobija broj $-A$ (takodje u potpunom komplementu), kao i korektnost zbira dva broja predstavljena na ovaj način.

Rešenje:

Neka je $\bar{A} = \overline{a_0 a_1 \dots a_n}$ (uključujući i bit znaka). Invertovanje cifara a_i zapravo predstavlja oduzimanje broja $\bar{A}$ od broja čija je binarna reprezentacija $111 \dots 11$ ($n+1$ jedinica). Ovaj broj je upravo $2^{n+1} - 1$. Prema tome operacija inverovanja daje rezultat $2^{n+1} - 1 - \bar{A}$. Dodavanjem jedinice dobija se konačni rezultat operacije komplementiranja $2^{n+1} - \bar{A}$. Odavde trivijalno sledi da se u oba slučaja $A \geq 0$ i $A < 0$ komplementiranjem dobija broj $-A$ u potpunom komplementu.

Dokažimo sada da se nad brojevima predstavljenim u potpunom komplementu operacija sabiranja izvršava na predhodno opisani način. Pošto radimo sa ukupno n binarnih cifara a prenos na poslednjoj cifri ignorišemo, mi zapravo izvršavamo operaciju sabiranja po modulu 2^{n+1}. Naravno, ovo sabiranje je korektno ukoliko je zbir u opsegu brojeva koji se mogu predstaviti u potpunom komplementu pomoću n bitova. Dakle posmatramo $(\bar{A}+\bar{B}) \bmod 2^n$. Ako su $A, B > 0$ tada je $A = \bar{A}$ i $B = \bar{B}$ kao i $A + B = \overline{A + B}$ pa je rezultat tačan. Ako je $A \geq 0$ i $B < 0$ onda je

$$(\bar{A} + \bar{B}) \bmod 2^{n+1} = (A + 2^n - |B|) \bmod 2^{n+1}.$$

Ako uz to i $A \geq |B|$ onda je predhodni izraz jednak $A - |B|$. Takodje je i $\overline{A + B} = \overline{A - |B|} = A - |B|$. Prema tome rezultat sabiranja je korektan. Ako je $A \leq |B|$ onda je rezultat sabiranja $2^{n+1} - (|B| - A)$. Sada je $\overline{A + B} = \overline{A - |B|} = 2^{n+1} - (|B| - A)$. Opet dobijamo korektnost rezultata sabiranja. Slučaj $A < 0$ i $B \geq 0$ se razmatra analogno. Neka je sada $A < 0$ i $B < 0$. Imamo da je

$$(\bar{A} + \bar{B}) \bmod 2^{n+1} = (2^{n+1} - |A| + 2^{n+1} - |B|) \bmod 2^{n+1} = 2^{n+1} - |A| - |B|$$

Takodje je i $\overline{A+B} = \overline{-|A|-|B|} = 2^n - |A| - |B|$. Prema tome i u ovom poslednjem slučaju zaključujemo da se dobija ispravna vrednost zbira brojeva A i B.

Zadatak 12. Izračunati vrednosti sledeća dva izraza koristeći potpun komplement i reprezentaciju brojeva pomoću $n = 10$ bita:

a) $1023 + 1$ b) $-1024 - 1$

Prokomentarisati dobijene rezultate.

Rešenje:

a)

```
1023+1 =  0111111111
          _         1
           1000000000
```

Primećujemo da smo dobili rezultat -1024. Ovo je posledica činjenice da pravi rezultat izlazi iz opsega brojeva koje je moguće predstaviti u potpunom komplementu pomoću 10 bita.

b)

```
-1024-1 =  1000000000
           1111111111
           0111111111
```

Dakle dobili smo rezultat 1023. I ovde pravi rezultat -1025 izlazi iz opsega.

1.4. BCD aritmetika

Binarni brojni sistem je najprirodniji za predstavljanje brojeva u računaru. Sa druge strane, u svakodnevnom životu se korisi isključivo dekadni sistem. Jedan način za prevazilaženje ove razlike je da se dekadni brojevi najpre konvertuju u binarne, zatim izvrše aritmetičke operacije i na kraju rezultati konvertuju nazad u dekadni sistem. Pošto kompjuter prihvata samo binarne vrednosti, potrebno je dekadne cifre kodirati binarnim vrednostima i onda ih smestiti u memoriju. Kao što ćemo videti, moguće je nad ovako kodiranim brojevima direktno izvršavati aritmetičke operacije. Ovaj način predstavljanja brojeva se naziva BCD kod (Binary Coded Decimals, binarno kodirane dekadni sistem ili binarno-dekadni kod).

Da bismo binarno kodirali svaku dekadnu cifru potrebno je bar 4 binarne pozicije jer je $2^3 < 10 \leq 2^4$. Zbog toga će najveći broj binarnih kodova biti dužine 4. Medjutim za neke kodove se ovaj broj bitova povećava da bi kod imao neke druge zahtevane osobine. U sledećoj tabeli dato je nekoliko BCD kodova.

Dekadna cifra	8421	2421	XS3	2 od 5	9876543210	Hafmenov kod	Grejov kod	51111
0	0000	0000	0011	00011	0000000001	0000000	0000	00000
1	0001	0001	0100	00101	0000000010	1101001	0001	00001
2	0010	0010	0101	00110	0000000100	0101010	0011	00011
3	0011	0011	0110	01001	0000001000	1000011	0010	00111
4	0100	0100	0111	01010	0000010000	1001100	0110	01111
5	0101	1011	1000	01100	0000100000	0100101	0111	10000
6	0110	1100	1001	10001	0001000000	1100110	0101	11000
7	0111	1101	1010	10010	0010000000	0001111	0100	11100
8	1000	1110	1011	10100	0100000000	1110000	1100	11110
9	1001	1111	1100	11000	1000000000	0011001	1000	11111

Kod 8421 se često naziva *prirodni BCD kod* jer se svaka dekadna cifra kodira svojom binarnom reprezentacijom. Ovaj kod je najčešće u upotrebi zato što se aritmetičke operacije nad ovako kodiranim brojevima najjednostavnije obavljaju.

Kod 2421 ili *Ajkenov kod* je takodje težinski pri čemu je težina prve cifre takodje 2. Ovaj kod je komplementarni, tj ima osobinu da su kodovi cifara x i $9-x$ komplementarni (dobijaju se jedan iz drugog invertovanjem cifara).

Kod XS3 ili *višak 3* takodje ima osobinu komplementarnosti. Cifra x se kodira tako što se u binarnom sistemu predstavi broj $x+3$.

U kodu 2 od 5 se za kodiranje koristi 5 binarnih cifara. Kodovi koji koriste više od minimalnog broja cifara za predstavljanje nazivaju se redundantni kodovi. Takvi su svi ostali kodovi osim Grejovog koda.

Hafmenov kod spada u grupu kodova kod kojih je moguće otkriti i ispraviti greške nastale prilikom prenosa. Naime, ako se pri prenosu neke cifre nadje greška na tačno jednoj binarnoj poziciji, uredjaj je može automatski detektovati i ispraviti. Ako se greška javi na dve ili tri binarne pozicije, grešku je moguće detektovati ali ne i jednoznačno ispraviti. U tom slučaju uredjaj samo signalizira nastanak greške. Ako bitove Hafmenovog koda označimo sa $ABc_8Cc_4c_2c_1$ onda su kontrolni bitovi A,B i C redom jednaki

$$A = c_8 \oplus c_4 \oplus c_1, \quad B = c_8 \oplus c_2 \oplus c_1, \quad C = c_4 \oplus c_2 \oplus c_1.$$

Grejov kod je netežinski kod sa osobinom da se kodni nizovi dve susedne cifre razlikuju u tačno jednoj binarnoj poziciji. Često se naziva i *Grejov ciklički kod*. Ovaj kod se može definisati za sve brojeve od 0 do 15 a može se proširiti i na više od 4 bita.

Kodovi koji su najčešće u upotrebi su 8421, 2421, XS3 kao i Grejov ciklički kod. U nastavku ćemo proučiti način na koji se izvode aritmetičke operacije u 8421 kodu, i pod pojmom BCD kod podrazumevaćemo upravo ovaj kod.

Ukratko ćemo opisati postupak sabiranja brojeva predstavljenih u BCD kodu. Najpre se brojevi saberu kao binarni brojevi. Poželjno je u zapisu odvojiti tetrade bitova koje odgovaraju različitim ciframa. Zatim se neke od cifara koriguju tako što se na mestu te cifre doda korekciona tetrada 0110 na osnovu sledećih pravila:

1. Ukoliko je dobijena cifra veća od 10.
2. Ukoliko je dobijena cifra manja od 10, a prilikom sabiranja bilo je prenosa u sledeću tetradu
3. Ukoliko je dobijena cifra 9 a tetrada sa desne strane je korigovana na osnovu pravila 1 ili 3 (ovim se izbegava višestruka korekcija.

Posle dodavanja korekcionog niza (eventualni prenosi pri ovom dodavanju se uzimaju u obzir) dobija se tačan broj u BCD formatu.

Postoji još jedan način za sabiranje brojeva predstavljenih u BCD kodu. Algoritam se sastoji u sledećem:

1. Najpre se sve tetrade prvog sabirka koriguju dodavanjem 0110. Prilikom ovog dodavanja nema prenosa.
2. Zatim se na tako korigovan prvi sabirak doda drugi.
3. Na kraju se tetrade iz kojih nije bilo prenosa koriguju dodavanjem 1010 i pritom se prenos u naredne tetrade ignoriše.

Zadatak 13. Predstaviti broj 9785 u BCD kodovima 8421, 2421, Grejov kod, XS3.

Rešenje:

8421:
1001 0111 1000 0101
2421:
1111 1101 1110 1011
Grejov kod:
1000 0100 1100 0111
XS3:
1000 0100 1100 0111

Zadatak 14. Predstaviti sledeće brojeve u BCD kodu i izvršiti odgovarajuća sabiranja (u delovima a) i b) koristiti prvi a u delu c) drugi način za sabiranje).

a) 9785 + 1432
b) 80499 + 19899
c) 257 + 28563

Rešenje:

a)

```
     1001 0111 1000 0101
+    0001 0100 0011 0010
-------------------------
     1010 1011 1011 0111
+    0110 0110 0110_____
-------------------------
   1 0001 0010 0001 0111
```

Vidimo da druga, treća i četvrta tetrada (cifra) u rezultatu su veće od 10, pa prema tome na tim mestima vršimo korekciju. Korekciju sabiramo sa rezultatom da bi dobili korektnu vrednost zbira.

b)

```
     1000 0000 0100 1001 1001
+    0001 1001 1000 1001 1001
-----------------------------
     1001 1001 1101 0011 0010
+    0110 0110 0110 0110 0110
-----------------------------
   1 0000 0000 0011 1001 1000
```

Korekcija se ovde vrši na prvoj i drugoj tetradi zato što postoji prenos u narednu tetradu, na trećoj zato što je vrednost veća od 9 kao i na četvrtoj zato što se na predhodnoj vrši korekcija na osnovu 1. pravila.

c)

```
     0000 0000 0010 0101 0111
+    0110 0110 0110 0110 0110
-----------------------------
     0110 0110 1000 1011 1101
+    0010 1000 0101 0110 0011
-----------------------------
     1000 1110 1110 0010 0000
+    1010 1010 1010      ____
-----------------------------
     0010 1000 1000 0010 0000
```

Zadatak 15. Broj X u Ajkenovom kodu ima zapis $0011\ 1111\ 1100$ a broj Y u Grejovom kodu $0110\ 0111$. Naći $X + Y$ i rezultat prevesti u dekadni sistem.

1.5. Predstavljanje razlomljenih brojeva u računaru

Postoje dva načina za predstavljanje razlomljenih brojeva u računaru:

- Predstavljanje brojeva u fiksnom zarezu
- Predstavljanje brojeva u pokretnom zarezu

1.5.1 Predstavljanje brojeva u fiksnom zarezu

Kod ovog načina se m bita koristi za predstavljanje celog dela broja a n bitova za predstavljanje razlomljenog dela. Jedan bit (najveće težine) koristi se za predstavljanje znaka broja. Format ovog predstavljanja je prikazan na sledećoj slici.

Aritmetičke operacije u ovom formatu se jednostavno realizuju. Najmanji broj koji se može predstaviti u ovom formatu je 2^{-m}. Takodje apsolutna greška predstavljanja brojeva je konstantna i ne zavisi od broja. To znači da je relativna greška veća za manje a manja za veće brojeve. Opseg vrednosti koje se na ovaj način mogu predstaviti je mali. Zbog ovih nedostataka, ovaj način se veoma retko koristi.

Zadatak 16. Predstaviti u fiksnom zarezu sledeće brojeve:

a) 189.931293
b) 530.8421983
c) -81.123312

Pritom za pamćenje celog dela i znaka broja na raspolaganju je ukupno $n = 22$ bita a za pamćenje razlomljenog dela $m = 10$ bita.

Rešenje:

a) Dat broj ima sledeću reprezentaciju u binarnom sistemu:

```
10111101.11101110011010010011
```

Broj je pozitivan, pa je bit znaka jednak 0. Ceo deo broja ima 8 cifara. Prema tome potrebno je dodati još 13 nula na početku broja. Reprezentacija u fiksnom zarezu ima sledeći oblik:

```
0000000000000010111101111110111001
```

b)

```
0000000000001000010010111011100l
```

c)

```
1000000000000000101000100011111l0
```

Zadatak 17.

a) Naći vrednost najvećeg i najmanjeg broja (po apsolutnoj vrednosti, različitog od nule) koji može da se predstavi u fiksnom zarezu pri čemu je za predstavljanje celog dela i znaka broja ukupno na raspolaganju $n+1$ bita a za predstavljanje razlomljenog dela m bita.

b) Definisati potpuni komplement za razlomljene brojeve u fiksnom zarezu uopštavajući odgovarajuću definiciju koja se odnosi na cele brojeve. Opisati način kako se mogu realizovati operacije sabiranja i oduzimanja brojeva u fiksnom zarezu.

Rešenje:

a) Najveći broj po apsolutnoj vrednosti se dobija kada su svi bitovi osim bita znaka jednaki 1. Tada je vrednost broja jednaka

$$R = \sum_{i=-m}^{n-1} 2^i = \sum_{j=0}^{m+n-1} 2^{j-m} = 2^{-m}(2^{m+n}-1) = 2^n - 2^{-m}.$$

Očigledno, najmanji broj se dobija kada je samo bit najmanje težine jednak 1. Vrednost broja je tada

$$R = 2^{-m}.$$

b) Sabiranje i oduzimanje mogu se obavljati na potpuno isti način kao i kod celih brojeva. Pritom mora se voditi računa da broj bitova za predstavljanje kako celog tako i razlomljenog dela oba sabiraka (odnosno umanjenika i umanjioca) budu jednaki. Ukoliko to nije slučaj potrebno je reprezentacije brojeva proširiti (u opštem slučaju na obe strane). Kada se postigne da su traženi brojevi bitova jednaki (odnosno da se tačke koje razdvajaju ceo od razlomljenog dela broja poklope) dalje se sa brojevima operiše kao da su to u pitanju celi brojevi. Tako da bez problema može da se definiše potpuni (a i nepotpuni) komplement i da se na taj način realizuje oduzimanje.

Primer. Predstaviti brojeve 2.5 i 3.375 u fiksnom zarezu i naći $2.5 - 3.375$.

U binarnom sistemu brojevi 2.5 i 3.25 redom se predstavljaju kao 10.1 i 11.011. Pošto drugi broj ima 3 cifre u razlomljenom delu a prvi jednu, prvi broj proširujemo sa desne strane za dve cifre. Tako dobijamo reprezentacije ovih brojeva u fiksnom zarezu:

```
010100
011011
```

Drugi broj komplementiramo i dobijamo 100101. Sabiranjem dobijamo sledeći rezultat:

```
   010100
 + 100101
 --------
   111001
```

Dekomplementiranjem dobijamo da je rezultat -000111. Pritom ovo treba shvatiti kao broj u fiksnom zarezu sa 3 cifre u razlomljenom delu. Prema tome rezultat je -0.111 u binarnom odnosno -0.875 u dekadnom sistemu.

1.5.2. Predstavljanje brojeva u pokretnom zarezu

U mnogim oblastima javljaju se i vrlo velike i vrlo male vrednosti brojeva sa kojima se vrše izračunavanja. Prema tome, potrebno je kreirati format koji će biti u stanju da pamti kako velike tako i male

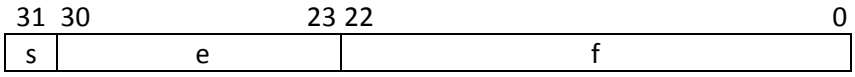
brojeve pri čemu će relativna greška za sve brojeve biti jednaka. Upravo te osobine ima format u pokretnom zarezu. Svaki razlomljeni broj R moguće je predstaviti u sledećem eksponencijalnom obliku ili obliku sa pokretnim zarezom

$$R = mb^E.$$

Veličina m se naziva mantisa broja R, E eksponent a b osnova. Pošto sve brojeve pamtimo u binarnom sistemu smatraćemo da je $b = 2$. Svaki broj je moguće predstaviti na više načina u eksponencijalnom obliku. Zbog toga je usvojen standardni oblik koji se uvek koristi. U računarstvu se brojevi u pokretnom zarezu predstavljaju najčešće pomoću 32 ili 64 bita. Razmotrićemo slučaj kada koristimo 32 bita.

Ukoliko se za predstavljanje brojeva koristi ukupno 32 bita, odgovarajući format se naziva *jednostruka tačnost*. Mantisa ima format $m = \overline{1.f}$ gde su sa f označene preostale binarne cifre. Vrednosti eksponenta E mogu biti kako pozitivne tako i negativne. Za predstavljanje eksponenta koristi se 8 bita i pamti se broj $e = E + 127$. Jedan bit se koristi za znak (bit s). Preostali bitovi (ukupno 23) se koriste za pamćenje vrednosti f.

Napomenimo da se svaki broj R na jedinstven način može opisati trojkom (s, e, f). Predstavljanje se obavlja kao na sledećoj slici.



Pritom vrednosti broja e mogu biti od 1 do 254 i tada je vrednost broja $R = (-1)^s \cdot 1.f \cdot 2^{e-127}$. Vrednosti 0 i 255 koriste se za opisivanje sledećih ekstremnih slučaja (R je vrednost broja koji se predstavlja).

a) $e = 255$ i $f \neq 0$. Onda je $R = NaN$.
b) $e = 255$ i $f = 0$. Onda je $R = (-1)^s \cdot \infty$.
c) $e = 0$ i $f = 0$. Onda je $R = (-1)^s \cdot 0$.
d) $e = 0$ i $f \neq 0$. Onda je $R = (-1)^s \cdot 0.f \cdot 2^{-126}$.

Ako je $e = 255$ a $f \neq 0$ tada je rezultat *nebroj* ili *NaN* (*Not a Number*). Ovaj ekstremni slučaj se koristi ukoliko je prilikom izračunavanja došlo do greške. Vrednost broja f je tada kod greške koja je nastala. NaN se koristi prilikom otkrivanja sledećih neregularnosti pri izvršavanju operacija sa realnim brojevima: (1) neispravna operacija, (2) deljenje sa nulom i (3) prekoračenje. Pod neispravnim operacijama podrazumevaju se sledeći nedefinisani izrazi $0/0$, $0 \cdot \infty$, $\sqrt{r}\ r < 0$ itd.

Ako je $e = 0$ i $f \neq 0$ tada se radi o denormalizovanim brojevima. Kod ovih brojeva je eksponent jednak minimalnoj vrednosti a vodeći bit mantise jednak 0. Ovo su po apsolutnoj vrednosti najmanji brojevi koje je moguće predstaviti u formatu pokretnog zareza.

Napomenimo samo još da u ovom formatu postoje dve reprezentacije za nulu (ako je $e = 0$ i $f \neq 0$). Medjutim ova činjenica ne predstavlja problem (kao što je to bio slučaj kod nepotpunog komplementa) pošto su operacije koje se izvršavaju nad brojevima u pokretnom zarezu ionako dovoljno složene.

Ukoliko je dužina celog broja 64 bita, pamćenje se obavlja na isti način pri čemu je sad $e = E + 1023$ i predstavlja se pomoću 11 bita a f se predstavlja pomoću 52 bita.

Zadatak 18. Predstaviti u pokretnom zarezu pomoću 32 bita sledeće vrednosti:
a) 129.3123, b) 2^{100}, c) $34 \cdot 2^{-50}$ d) 2^{-130}

Rešenje:

a) Kada broj predstavimo u binarnom sistemu dobijamo:

$$(1.0000001010011111111)_2 \cdot 2^7$$

Prema tome imamo da je $e = 7 + 127 = 134 = (1000110)_2$ dok je $f = 00000010100111111111000$. Prema tome traženo predstavljanje broja je

`00100011000000010100111111110000`

b) Na sličan način kao u delu a) dobijamo da je $e = 100 + 127 = 227 = (11100011)_2$ i $f = 0$ pa je traženo predstavljanje

`01110001100000000000000000000000`

c) $34 = (100010)_2 \cdot 2^{-50} = (1.0001)_2 \cdot 2^{-44}$,
$e = -44 + 127 = (1010011)_2$, $f = 00010000000000000000$,

`00101001100010000000000000000000`

d) Pošto je eksponent ovog broja manji od -126, radi se o denormalizovanom broju. Prema tome $e = 0$ a $f = 00010000000000000000000$ odnosno reprezentacija broja je:

`00000000000010000000000000000000`

Zadatak 19. Predstaviti u pokretnom zarezu, u standardnom formatu jednostruke tačnosti:

a) najveći pozitivan broj
b) najmanji pozitivan broj
c) najveći negativan normalizovan broj
d) najmanji negativan denormalizovan broj

Rešenje:

a) Najveća moguća vrednost broja e je 254 a u delu f svih 23 bita su jednaki 1. Prema tome to je broj:

`01111111011111111111111111111111`

b) Najmanji pozitivan broj se dobija kada je $e = 0$ i kada je samo poslednji bit broja f jednak 1.

`00000000000000000000000000000001`

c) Najveći negativan normalizovan broj se dobija ua $e = 1$ i $f = 0$ i ima predstavljanje:

`10000000100000000000000000000000`

d) Najmanji negativan denormalizovan broj dobija se za $e = 0$ i kada su svi bitovi broja f jednaki 1.

`10000000011111111111111111111111`

Zadatak 20. Naći vrednosti sledećih brojeva predstavljenih u formatu jednostruke tačnosti:

a) 10110101001000101011100000000000
b) 11111111100000000000000000000000
c) 11111111100000000001000000000000
d) 00000000000001000000000000000000

Rešenje:

a) $e = 01101010$, $f = 010001010111$, pa je $E = -21$ odnosno $m = 1.f = 1.54248$. Vrednost broja je $R = 1.54248 \cdot 2^{-21}$.

b) pošto je $e = 255$, $f = 0$ i $s = 1$ zaključujemo da je $R = -\infty$.

c) pošto je $e = 255, f \neq 0$ zaključujemo da je $R = NaN$.

d) pošto je $e = 0$, broj je denormalizovan. $f = 00001000 \ldots$ pa zaključujemo da je $R = 2^{-131}$.

1.6. Predstavljanje znakovnih podataka

Znakovni ili alfanumerički podaci su podaci koji se pojavljuju u formi čitljivoj za ljude. Ovi podaci su sastavljeni od nizova reči nad usvojenim skupom znakova koji obično sadrži velika, mala slova engleske abecede, decimalne cifre, znake aritmetičkih operacija, znake interpunkcije, specijalne znake itd. Nad ovakvim podacima primenjuju se operacije kao što su spajanje niza, izdvajanje dela niza, traženje zadatog podniza i druge. Uvodjenje znakovnih podataka omogućilo je pamćenje, obradu i štampanje u tekstualnom obliku podataka o licima i objektima i njihovim osobinama, obradu teksta razne namene kao što je izdavačka delatnost ili obrada teksta u kancelarijama. Takodje i u nekim naučnim i korisničkim aplikacijama postoji potreba označavanja odredjenih izračunatih vrednosti, da bi se znalo šta te vrednosti zapravo predstavljaju.

Za predstavljanje znakovnih podataka koriste se standardni binarni kodovi odnosno binarni kodovi usvojeni od nacionalnih i medjunarodnih organizacija za standardizaciju. Najpoznatiji kod je sigurno ISO7 kod ili ASCII - American Standard Code for Information Interchange. Svaki karakter se u ovom kodu predstavlja pomoću 7-bitne kodne reči. U sledećoj tabeli dati su kodovi svih karaktera koji se mogu kodirati pomoću ASCII koda. Vrste tabele su indeksirane pomoću 3 najstarija bita ($b_7 b_6 b_5$) a kolone pomoću ostala 4 bita ($b_4 b_3 b_2 b_1$).

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Prilikom smeštanja u memoriju i pri prenosu podataka za svaki znak se koristi 8 bita (1 bajt). Dodatni bit se koristi za proveru parnosti. Ovaj bit se takodje može koristiti za proširenje ISO 7 koda kako bi se predstavili neki simboli specifični za pojedine jezike. Na taj način svaka zemlja formira svoje proširenje ISO 7 koda. Prema tome tekst napisan npr. na ćirilici (kodna tablica proširenja ASCII koda za ćirilicu data je standardom JUS I.B1.015) neće biti korektno prikazan na računaru koji npr. koristi hebrejsku ekstenziju ASCII koda. Ovo je ozbiljan nedostatak ASCII koda i zbog toga se pristupilo konstrukciji novih kodova koji ne pate od ovog problema.

Na taj način se došlo do novog standarda Unicode za predstavljanje alfanumeričkih podataka. Sada se svaki karakter reprezentuje pomoću 16 bita. Kodna reč bilo kog simbola u kodu Unicode može se predstaviti pomoću četiri heksadekadne cifre. Na primer, kodovi svih karaktera koji se kodiraju u kodu ASCII dobijaju se tako što se na odgovarajući ASCII kod dodaju dve nule (tj $(00)_{16}$). Ovaj kod može da kodira ukupno 65536 karaktera što je dovoljno da se kodira veliki broj slova i simbola karakterističnih za neki jezik. Svaka kodna reč u ovom kodu može se predstaviti pomoću dva bajta. Postoje dva načina za smeštanje ova dva bajta u memoriji. Označimo sa 0 niži a sa 1 viši bajt. Ukoliko se u memoriji bajtovi smeštaju u redosledu 10 onda se to smeštanje naziva little-endian a u suprotnom big-endian. Koji će se od ova dva načina koristiti zavisi od vrednosti prva dva bajta u poruci. Ako su oni redom jednaki FE i FF onda se koristi little-endian a ako su jednaki FF i FE redom onda se koristi big-endian.

Danas su u upotrebi i kodovi kod kojih dužina kodnih reči nije konstantna i zavisi od karaktera koji se kodira. Tako se najčešće ASCII karakteri predstavljaju samo pomoću jednog bajta (izmedju ostalog da bi se ostvarila potpuna kompatibilnost sa ASCII standardom) dok se za neke specijalne karaktere koristi 2, 3 ili više bajtova. Primer takvog koda je UTF-8 koji predstavlja ekstenziju Unicode koda (on se alternativno naziva UTF-16).

Zadatak 21. Koja se reč predstavlja sledećim Unicode stringovima:

a) FEFF0043006F00640065
b) FFFE43006F0064006500

Rešenje:

U delu pod a) zaključujemo da se koristi format little-endian. Prema tome u memoriji se smešta najpre viši pa onda niži bajt. Prema tome ako pročitamo naredna 2 bajta dobijamo da je 00 viši a 43 niži. Pošto je viši bajt 00 radi se o ASCII karakteru čiji je kod 43. Vidimo da se radi o karakteru ‘C’. Ako ovaj postupak nastavimo dalje dobićemo da je tražena reč “Code”.

U delu pod b) na sličan način zaključujemo da se radi o big-endian formatu i da je tražena reč takodje “Code”.

Zadatak 22. Ukoliko se najviši bit u ASCII kodu odredjuje kao bit parnosti, utvrditi da li je ispravno preneta kodna sekvenca i dekodirati je: `01000100 11100101 01110100 11100101 11010010 01011111 01101111 01100110 01011111 00101110 01101001 11110011`.

Rešenje:

Najpre proveravamo da li je bit parnosti ispravan, sabiranjem ostalih bitova po modulu 2 i uporedjivanjem sa bitom parnosti. Pošto smo utvrdili da je bit parnosti ispravan u svakom prenesenom bajtu, ovaj bit izbacujemo i dekodiramo sekvencu. Znakovni niz koji na ovaj način dobijamo je “DeXteR_of_Nis”.

1.7. Zadaci za vežbu

Zadatak 1. Izvršiti odgovarajuće operacije bez prevodjenja brojeva u dekadni sistem.

a) $(715)_8 + (367)_8$ b) $(15F)_{16} \cdot (A7)_{16}$ c) $(110101)_2 - (110110)_2$

Zadatak 2. Odrediti sve brojeve $b \geq 3$ takve da je broj $(121)_b$ kvadrat nekog broja.

Zadatak 3. Odrediti najmanji prirodan broj $b \geq 10$ takav da je broj $(792)_b$ deljiv brojem $(297)_b$.

Zadatak 4. Konvertovati sledeće brojeve u odgovarajuće osnove

a) $(3517.63126)_8$ u heksadekadni sistem.
b) (F921.ABC)_16 u oktalni sistem.
c) (AB.CD) u dekadni sistem.
d) 2132 u binarni sistem.

Zadatak 5. Izvršiti sledeće operacije u nepotpunom i potpunom komplementu:

a) $31231 - 4238 + 9884$ b) $12.3829 + 90.98 - 1.7$ c) $877.213 + 32.21$

Prilikom konverzije u binarni sistem razlomljene brojeve zaokružiti na 9 decimala.

Zadatak 6. Predstaviti brojeve 831 i 793 u BCD kodu 8421 a zatim naći njihovu sumu (korišćenjem najpre prvog a zatim i drugog načina). Rezultat predstaviti u Hafmenovom kodu, u Grejovom kodu i konvertovati ga nazad u dekadni sistem.

Zadatak 7. Broj -482.1329031 predstaviti najpre u formatu sa fiksnim zarezom pri čemu se ceo deo broja predstavlja pomoću 15 a razlomljeni pomoću 17 bita, zatim u formatu sa pokretnim zarezom i to najpre u jednostrukoj a onda u dvostrukoj tačnosti. Broj cifara dela f u oba slučaja ograničiti na 10.

Zadatak 8. Opisati ekstremne slučajeve u formatu dvostruke tačnosti. Naći najveći i najmanji pozitivan broj koji se može predstaviti u ovom formatu.

Zadatak 9. Znakovne podatke “Marko” i “DeXteR” kodirati standardnim ASCII kodom i predstaviti njihov izgled u binarnom kodu u obliku u kome se smeštaju u memoriji pri čemu se osmi bit odredjuje kao bit parnosti.