1.2 Brojevni sustavi

Paralelno s razvojem pisma, razvijali su se i znakovi za prikaz brojeva. Potreba stvaranja naziva i znakova za veće brojeve bila je prva okolnost koja je prisilila čovjeka na traženje sustavnih postupaka. Na primjer, brojevi 1, 2, 3, 4 mogli bi se označavati s I, II, III, IIII, ali je ovakav sustav nemoguće zadržati za velike brojeve. Zbog toga razvijeni brojevni sustavi, tj. načini označavanja brojeva nizovima znakova - znamenki.

Postoje različiti sustavi, a danas je u upotrebi tzv. aditivno-multiplikativni sustav koji su u Evropu prenijeli Arapi, a razvijen je u Indiji. U tom sustavu možemo po volji veliki broj napisati pomoću svega nekoliko različitih znamenki (najmanje dvije). Svaka znamenka toga sustava ima svoju brojevnu i mjesnu vrijednost. Takav sustav se zato naziva i težinski ili položajni. Krajnje lijeva znamenka ima najveću težinu, a krajnje desna znamenka najmanju. Zbog toga se krajnje lijeva znamenka zove najznačajnijom znamenkom, a krajnje desna znamenka najmanje značajnom znamenkom. Broj upotrijebljenih znamenki određuje osnovu (bazu) sustava. Opći prikaz broja **R** u težinskom sustavu je:

$$\mathbf{R = d_n d_{n-1} ... d_2 d_1 d_0 . d_{-1} d_{-2} ... d_{-(m-1)} d_{-m} =}$$

$$\mathbf{= d_n B^n + d_{n-1} B^{n-1} + ... + d_2 B^2 + d_1 B^1 + d_0 B^0 + d_{-1} B^{-1} + d_{-2} B^{-2} + ... + d_{-(m-1)} B^{-(m-1)} d_{-m} B^{-m}}$$

gdje je $\mathbf{d_i}$ odgovarajuća znamenka ($\mathbf{d_i \leq (B-1)}$), a **B** osnova sustava.

Danas je uobičajen težinski sustav s osnovom 10. Razlog je anatomske prirode: čovjek ima deset prstiju koje je koristio kao pomoćno sredstvo prilikom računanja. Zapravo, sustav s osnovom 12 bio bi praktičniji (djeljivost bez ostatka s 2, 3, 4, 6), ali bi prijelaz na njega uzrokovao velike probleme. Zanimljivo je i to da su Babilonci upotrebljavali sustav s osnovom 60, čije tragove nalazimo kod mjera za kut i vrijeme. Računala koriste binarni brojevni sustav, tj. sustav s osnovom 2. Takav sustav je najjednostavniji jer zahtijeva svega dvije znamenke (0 i 1), a to znači i jednostavne elektroničke sklopove za prikaz tih znamenki. U računarstvu se upotrebljavaju i sustavi s osnovom 8 i 16, prvenstveno zbog lagane pretvorbe između njih i binarnog sustava, pa se katkada koriste za skraćeni prikaz binarnih brojeva.

Dekadski sustav

Dekadski sustav ima osnovu 10 i koristi slijedeće znamenke: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Svaka znamenka dekadskog broja ima svoju težinu koja je potencija broja 10 (10^i). Pritom je eksponent (i) cijeli broj, a njegova vrijednost određena je položajem znamenke u broju.

Primjer 1.2.1

$43 = 4*10^1 + 3*10^0$
$444 = 4*10^2 + 4*10^1 + 4*10^0$
$72056 = 7*10^4 + 2*10^3 + 0*10^2 + 5*10^1 + 6*10^0$
$12.5 = 1*10^1 + 2*10^0 + 5*10^{-1}$

Može nas zanimati koliko različitih brojeva možemo prikazati brojem koji ima n znamenki (npr. kod kalkulatora i računala n je ograničen). Tada govorimo o **kapacitetu (K) broja s n znamenki**: $\mathbf{K = B^n}$, gdje je **B** osnova brojevnog sustava. Dakle, kapacitet je broj koji nam kaže koliko različitih brojeva možemo prikazati s n znamenki, ako je zadana osnova sustava. Najveći broj **M** koji možemo prikazati s n znamenki je za jedan manji od kapaciteta, tj.: $\mathbf{M = B^n - 1 = K - 1}$.

Primjer 1.2.2

S 4 znamenke u dekadskom sustavu možemo prikazati $10^4 = 10000$ različitih brojeva, a najveći je $10000 - 1 = 9999$.

Binarni sustav

Znamenke binarnog sustava su **0** i **1**, a njegova osnova **B = 2**. Binarna znamenka zove se **bit** (skraćeno od engleskog izraza **Bi**nary digi**t**). Ukupni kapacitet **K** binarnog broja s n bita je $\mathbf{K = 2^n}$, a najveći broj M koji možemo prikazati je $\mathbf{M = 2^n - 1 = K - 1}$.

Primjer 1.2.3
S 8 bita možemo prikazati 2^8=256 različitih brojeva, najveći je 255 (11111111_2), a najmanji je 0 (00000000_2).

Pretvorba binarnog broja u dekadski

Kao i kod dekadskog sustava radi se o težinskom sustavu, dakle primjerice vrijedi:

$101101_2 = 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 32 + 0 + 8 + 4 + 0 + 1 = 45_{10}$

Na taj način možemo bilo koji binarni broj pretvoriti u dekadski. Kod dekadskog broja obično ne označavamo osnovu sustava, ali, ako se radi o nekoj drugoj osnovi, moramo je označiti kao u prethodnom primjeru.

Primjer 1.2.4

$11001_2 = 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 25_{10}$

Primjer 1.2.5

$1.111_2 = 1*2^0 + 1*2^{-1} + 1*2^{-2} + 1*2^{-3} = 1 + 0.5 + 0.25 + 0.125 = 1.875_{10}$

Primjer 1.2.6

Može li broj 1020 pripadati binarnom sustavu? Ne može. Zašto? Za svaku znamenku d mora vrijediti $d \leq (B-1)$. Budući da je B=2, za znamenku 2 broja 1020 ne vrijedi $2 \leq 1$.

Pretvorba dekadskog broja u binarni

Pretvorba prirodnog dekadskog broja u binarni može se opisati slijedećim postupkom:

1. Podijeliti dekadski broj s 2.
2. Zapisati ostatak dijeljenja (0 ili 1).
3. Dobiveni kvocijent (cjelobrojni dio) podijeliti s 2.
4. Zapisati ostatak dijeljenja.
5. Ako kvocijent nije 0 vratiti se na točku 3.

Ostaci dijeljenja koje smo zapisivali predstavljaju traženi binarni broj koji treba čitati obrnuto, tj. zadnja dobivena znamenka je najznačajnija znamenka, a prva dobivena znamenka je najmanje značajna znamenka.

Primjer 1.2.7
Pretvoriti dekadski broj 43 u binarni.

```
43 : 2 = 21  -- ostatak 1
21 : 2 = 10  -- ostatak 1
10 : 2 = 5   -- ostatak 0
5  : 2 = 2   -- ostatak 1
2  : 2 = 1   -- ostatak 0
1  : 2 = 0   -- ostatak 1
```

Prema tome, dobije se $43_{10} = 101011_2$

Primjer 1.2.8
Načiniti tablicu dekadskih brojeva od 0 do 18 i njihovih binarnih ekvivalenta. Uočite kako se broji u binarnom sustavu.

Dekadski	Binarno
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010

Dekadski brojevi manji od 1 pretvaraju se u binarne brojeve upotrebom slijedećeg postupka:

1. Pomnožiti dekadski broj s 2.
2. Ako je dobiveni broj veći od 1 iza točke u binarnom broju napiše se 1.
3. Ako je dobiveni broj manji od 1 iza točke u binarnom broju napiše se 0. Postupak se ponavlja s dijelom umnoška iza decimalne točke s time da se 0 ili 1 dopisuje već napisanim brojevima (s desne strane).

Primjer 1.2.9
Pretvoriti dekadski broj 0.625 u binarni.

0,625*2 = 1.250 -- bilježimo 1
0.250*2 = 0.500 -- bilježimo 0
0.500*2 = 1.000 -- bilježimo 1

$0.625_{10} = 0.101_2$

Ispravnost pretvorbe može se provjeriti tako da se dobiveni binarni broj ponovo pretvori u dekadski:

$0.101_2 = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = 0.625_{10}$

Ako imamo realni dekadski broj veći od 1, možemo ga pretvoriti u binarni broj tako da pretvorimo posebno cjelobrojni dio, a posebno dio iza decimalne točke, a dobivene binarne brojeve zbrojimo.

Primjer 1.2.10
Pretvoriti 43.625 u binarni broj.

Od prije imamo: $43_{10} = 101011_2$ i $0.625_{10} = 0.101_2$. Dakle, $43.625_{10} = 101011.101_2$.

Zbrajanje binarnih brojeva
Zbrajanje binarnih brojeva može se naučiti imajući u vidu slijedeće pravila za zbrajanje dva bita:

0+0=0
0+1=1
1+0=1
1+1=0 i prijenos 1

Prijenos se prenosi u slijedeći stupac.

Primjer 1.2.11

```
 001101          kontrola:  13
+100101                    +37
 ------                     --
 110010                     50
```

Primjer 1.2.12

```
 1011011                    91
+1011010                  + 90
--------                  ----
10110101                   181
```

Oktalni sustav

Oktalni sustav ima osnovu **8** i koristi slijedeće znamenke: **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**. Kapacitet (**K**) **n** oktalnih znamenki je $\mathbf{K = 8^n}$, a najveći broj (**M**) koji možemo prikazati s **n** znamenki je $\mathbf{M = 8^n - 1 = K - 1}$. U informatici se oktalni sustav koristi za skraćeni prikaz binarnih brojeva.

Primjer 1.2.13
S dvije oktalne znamenke možemo prikazati $8^2 = 64$ različita broja, a najveći je $8^2 - 1 = 63$ (77_8).

Pretvorba oktalnog broja u dekadski

Pretvorba se vrši jednako kao i u slučaju binarnog broja, što pokazuje slijedeći primjer.

Primjer 1.2.14
Pretvoriti oktalne brojeve 37, 142 i 364 u dekadske.

$37_8 = 3*8^1 + 7*8^0 = 24 + 7 = 31_{10}$
$142_8 = 1*8^2 + 4*8^1 + 2*8^0 = 64 + 32 + 2 = 98_{10}$
$364_8 = 3*8^2 + 6*8^1 + 4*8^0 = 192 + 48 + 4 = 244_{10}$

Pretvorba oktalnog broja u binarni i binarnog u oktalni

Ovo je vrlo jednostavna pretvorba i zbog toga se oktalni sustav koristi za skraćeni prikaz binarnih brojeva. Svaku oktalnu znamenku treba prikazati s tri bita i obrnuto.

Primjer 1.2.15
Pretvoriti oktalni broj 76543 u binarni.

```
 7   6   5   4   3
111 110 101 100 011
```

Dakle, $76543_8 = 111110101100011_2$.

Primjer 1.2.16
Pretvoriti binarni broj 1101101111 oktalni.

Potrebno je rastaviti binarni broj u grupe po tri bita počevši s desne strane. Ako na kraju nedostaju znamenke, treba dodati jednu ili dvije nule s lijeve strane. Svaku grupu od tri bita treba zamijeniti jednom oktalnom znamenkom.

$1101101111_2 = 001\ 101\ 101\ 111_2 = 1557_8$

Pretvorba dekadskog broja u oktalni

Primjenjuje se jednaki algoritam kao i u slučaju dekadsko - binarne pretvorbe, s razlikom da se dijeli s 8.

Primjer 1.2.17
Pretvoriti dekadski broj 127 u oktalni.

```
127 : 8 = 15  -- ostaje 7
15  : 8 = 1   -- ostaje 7
1   : 8 = 0   -- ostaje 1
```

Dakle, $127_{10} = 177_8$.

Heksadecimalni sustav

Heksadecimalni sustav ima osnovu **16** i koristi znamenke **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A**, **B**, **C**, **D**, **E**, **F**. Vidimo da heksadecimalni sustav koristi slova A - F za dekadske ekvivalente 10 - 15. S **n** heksadecimalnih znamenki možemo prikazati $\mathbf{K = 16^n}$ različitih brojeva, a najveći je $\mathbf{M = 16^n - 1 = K - 1}$. U informatici se heksadecimalnim sustavom služimo za skraćeni prikaz binarnih brojeva.

Pretvorba heksadecimalnog broja u dekadski

Pretvorba se vrši kao i kod binarnog i oktalnog sustava, što ilustrira slijedeći primjer.

Primjer 1.2.18

Pretvoriti heksadecimalne brojeve 23, 3B i 1AF u dekadske.

$23_{16} = 2*16^1 + 3*16^0 = 32 + 3 = 35_{10}$

$3B_{16} = 3*16^1 + B*16^0 = 3*16^1 + 11*16^0 = 48 + 11 = 59_{10}$

$1AF_{16} = 1*16^2 + A*16^1 + F*16^0 = 1*16^2 + 10*16^1 + 15*16^0 = 256 + 160 + 15 = 431_{10}$

Pretvorba dekadskog broja u heksadecimalni

Pretvorba cijelog dekadskog broja u heksadecimalni vrši se dijeljenjem s 16, slično kao i pretvorbe u binarni i oktalni sustav.

Primjer 1.2.19

Pretvoriti dekadski broj 127 u heksadecimalni.

```
127 : 16 = 7  -- ostaje 15 (F)
7   : 16 = 0  -- ostaje  7
```

Dakle, $127_{10} = 7F_{16}$.

Pretvorba heksadecimalnog broja u binarni i obratno

Pretvorba je jednaka kao u slučaju oktalnog broja, ali se radi s grupom od četiri bita. Svakoj heksadecimalnoj znamenki odgovaraju četiri binarne znamenke (bita).

Primjer 1.2.20

Načiniti tablicu dekadskih brojeva od 0 do 16 i njihovih binarnih, oktalnih i heksadecimalnih ekvivalenta.

Dekadski	Binarni	Oktalni	Heksadecimalni
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Primjer 1.2.21
Pretvoriti heksadecimalni broj AF3 u binarni

$A_{16} = 10_{10} = 1010_2$ $F_{16} = 15_{10} = 1111_2$ $3_{16} = 0011_2$

$AF3_{16} = 101011110011_2$

Primjer 1.2.22
Pretvoriti binarni broj 1110110011 u heksadecimalni.

Prvo treba podijeliti binarni broj u grupe po četiri bita, počevši s desne strane. Kako zadnja grupa sadrži samo dva bita, treba je nadopuniti na četiri bita dodavanjem dvije nule s lijeve strane. Svaku grupu od četiri bita treba prikazati jednom heksadecimalnom znamenkom.

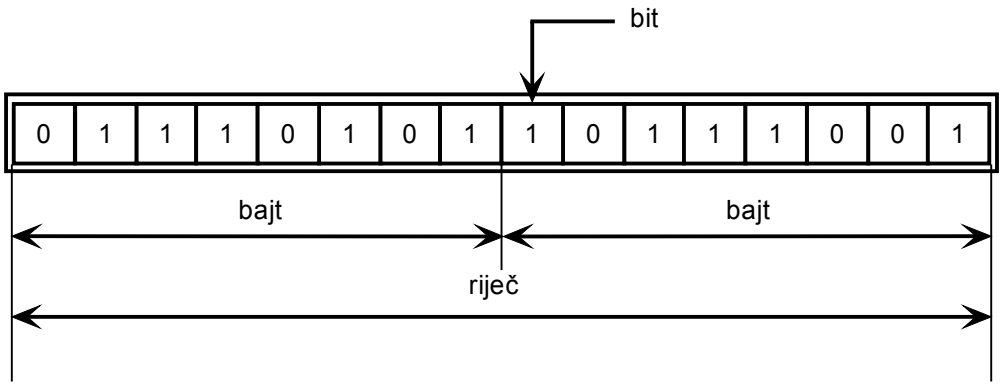
$1110110011_2 = 0011\ 1011\ 0011 = 3B3_{16}$

Prikaz brojeva i znakova u računalu

Za pohranu brojeva u računalu služi elektronički sklop koji se naziva **bistabil**. Naziv bistabil dolazi otuda što takav sklop ima dva stabilna stanja. Jedno stanje (npr. niskog napona) odgovara znaku 0, a drugo stanje (npr. visokog napona) odgovara znaku 1. Prema tome bistabil je sklop koji može zapamtiti znamenku 1 ili 0 (jedan bit). Kako se binarni broj sastoji od više znamenki (bitova) za prikaz broja moramo upotrijebiti nekoliko bistabila. Takva grupa bistabila čini **registar**. Registri su sastavni dio svih dijelova računala. Broj bistabila u registru nekog računala određuje njegovu **duljinu**. Duljina većine registara u nekom računalu je određena duljinom riječi računala. **Riječ** je količina informacija koju računalo može obraditi u jednoj operaciji, pohraniti u memoriju, odnosno dobaviti iz memorije. Najčešće duljine riječi (pa prema tome i registara) su 8, 16, 32 i 64 bita, a kod osobnih računala danas je uobičajena duljina riječi od 32 bita.

Grupa od 8 bitova obično se naziva **bajt** (*byte*). Jedan bajt se sastoji od osam bitova. Bit se skraćeno označava s **b**, a bajt s **B**, pa vrijedi **1B = 8b**. Dakle, osobna računala imaju duljinu riječi od 4B ili 32b.

Slijedeća slika prikazuje registar duljine 16 bita (2 bajta). Svaki bistabil simbolički je prikazan kao jedan kvadratić u koji je zapisan jedan bit. Informacija pohranjena u registru čini jednu riječ. Riječ može predstavljati broj, znak, kod neke instrukcije i sl. Kako je sadržaj jednog bistabila 0 ili 1, a registar se u ovom slučaju sastoji od 16 bistabila, možemo jednom riječju predstaviti 2^{16} različitih objekata, na primjer 2^{16} različitih brojeva ($2^{16} = 65536$).



Slika 1.2.1

Za pohranu podataka i programa u računalu služi memorija računala koja se može predočiti kao skup registara o čemu će više riječi biti u narednom poglavlju. Za sada nas zanima u kom obliku se podaci

zapisuju u memoriju računala. Osnovni tipovi podataka koji se upisuju u memoriju računala su brojevi (prirodni, cijeli i realni) i znakovi (slova, znamenke, znakovi interpunkcije i sl.).

Prikaz prirodnih brojeva

Prirodni brojevi se zapisuju u memoriju računala slično kao što bi ih zapisivali na papir. Najvažnija razlika je u tome što je broj bitova u računalu koji imamo na raspolaganju za prikaz broja ograničen. Broj bita za prikaz broja nije proizvoljan i može biti jednak duljini riječi, ali i duplo manji (polu riječ) ili duplo veći (dvostruka riječ). Što ćemo odabrati, ovisi o našoj procjeni veličina brojeva koji će se u našim proračunima pojaviti.

Primjer 1.2.23

Na raspolaganju za prikaz broja imamo jedan bajt. Kako će u memoriji računala biti prikazan dekadski broj 8?

$8_{10} = 1000_{2.}$

U memoriji računala biti će zapisano `00001000`. Važno je uočiti da smo napisali i nule s lijeve strane što je uobičajeno kada se prikazuje sadržaj nekog dijela memorije. Na taj način se vidi koliko je bita određeno za prikaz broja, te iznos svakog bita.

Primjer 1.2.24

Na raspolaganju za prikaz broja imamo dva bajta. Kako će u memoriji računala biti prikazan binarni broj 11011? Koji je najveći, a koji najmanji dekadski broj koji možemo prikazati s dva bajta?

Broj 11011_2 će s dva bajta biti prikazan kao `0000000000011011.`

Najmanji broj je `0000000000000000` ($=0_{10}$), a najveći `1111111111111111` ($=2^{16}-1=65535_{10}$)

Primjer 1.2.25

Na raspolaganju za prikaz prirodnog broja imamo dva bajta. Kako će u memoriji računala biti prikazan broj $1F4B_{16}$?

Odgovor glasi: `0001111101001011`. Uočite praktičnost prikaza stanja dva bajta pomoću heksadecimalnog sustava.

Ovakav način prikaza prirodnih brojeva u memoriji računala naziva se **prirodni binarni kod**.

Jedna od posljedica ograničenog broja bitova za prikaz brojeva u računalu je i pojava **preljeva** (carry) kod aritmetičkih operacija. Naime, rezultat neke aritmetičke operacije može zauzimati više bita nego što imamo na raspolaganju. Ako se to dogodi rezultat aritmetičke operacije nije točan (jer nedostaju bitovi najveće težine) i tada treba registrirati pojavu greške prilikom izvođenja aritmetičke operacije.

Primjer 1.2.26

Za prikaz brojeva u računalu na raspolaganju je jedan bajt. Zbrojiti binarne brojeve 10101010 i 1000000.

```
 10101010
+10000000
----------
100101010
```

Vidimo da rezultat zauzima 9 bita. Deveti bit biti će "odsječen" i izgledati će da je rezultat `00101010`, što je pogrešno. Zato u sklopu za zbrajanje postoji i deveti bit (ako se radi o sklopu koji može zbrajati 8-bitne brojeve), koji služi za kontrolu ispravnosti dobivenog rezultata. Ako je deveti bit nula, rezultat je ispravan, a, ako je jednak jedinici rezultat nije ispravan jer je došlo do preljeva. U našem slučaju deveti bit je jednak jedinici, što znači da rezultat `00101010` nije točan.

Prikaz cijelih brojeva

Negativne brojeve prikazujemo dodajući znak minus (-) ispred apsolutne vrijednosti broja. Međutim, računalo upotrebljava binarni sustav zato jer je građeno od elektroničkih sklopova koji imaju samo dva stanja (npr. niskog i visokog napona), koja predstavljaju znak 0 ili znak 1. Prema tome, umjesto znakova plus i minus moramo koristiti znakove 0 ili 1.

U računalu za prikaz nekog broja imamo na raspolaganju određen broj znamenki (bitova). Na primjer, za prikaz nekog broja možemo imati na raspolaganju memorijska lokaciju dužine 4 bita. Ako želimo prikazati i cijele brojeve moramo jedan **bit** odvojiti **za predznak**. Za predznak se odvaja krajnji lijevi bit. Ako je on 0, to znači da je broj pozitivan, a ako je on 1, to znači da se radi o negativnom broju. Na primjer, 0001_2 bi bio broj $+1_{10}$, a 1001_2 bi bio broj -1_{10}. Takav način prikazivanja negativnih brojeva je vrlo jednostavan, ali je pritom postupak zbrajanja i oduzimanja relativno kompliciran. Osim toga postoje dvije nule (+0 i -0).

Zbog navedenih razloga primjenjuje se često tehnika **dvojnog komplementa**. Bit za predznak u tehnici dvojnog komplementa interpretira se kao binarno mjesto s odgovarajućim težinskim faktorom, ali s negativnim predznakom.

Primjer 1.2.27
Broj 1011_2 prikazan u registru od 4 bita tehnikom dvojnog komplementa shvaćamo ovako:
$1011_2 = -1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = -8 + 0 + 2 + 1 = -5_{10}$.

Dvojni komplement nekog binarnog broja dobiva se tako da se slijedi slijedeći postupak:

1. Nadopuniti broj čiji dvojni komplement tražimo na broj bita koji imamao na raspolaganju za prikaz broja dodajući nule s lijeve strane.
2. U dobivenom broju zamijenimo nule s jedinicama i obratno.
3. Dodati 1.
4. Ako se pojavi prijenos koji bi zahtijevao dodatni bit on se zanemaruje.

Primjer 1.2.28
Na raspolaganju za prikaz broja je 8 bita. Prikazati dekadski broj -5 tehnikom dvojnog komplementa.
Broj 101_2 (5_{10}) nadopunimo s nulama i dobijemo: `00000101`.
Zamijenimo nule i jedinice i dobijemo: `11111010`.
Dodamo 1:

```
 11111010
+00000001
----------
 11111011     Dakle, -5 = 11111011₂.
```

Primjer 1.2.29
Naći dvojni komplement binarnog broja 00000000.
Provođenjem gornjeg postupka dobivamo (1)00000000. Jedinica nastala prijenosom prilikom zbrajanja bila bi deveti bit i nju odbacujemo. Dvojni komplement broja 00000000 je 00000000.

Možemo postaviti slijedeće pitanje: Kako znamo da li je u registru upisan negativan broj ili pozitivan broj koji počinje s jedinicom? Radi se o dogovoru. Moramo uvijek naglasiti da radimo s cijelim brojevima u tehnici dvojnog komplementa ili da radimo samo s prirodnim brojevima. Drugim riječima, moramo znati što sadrži neka memorijska lokacija (registar): prirodni broj, cijeli broj, realni broj, znakove u ASCII kodu itd..

Binarno	Dekadski
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Primjer 1.2.30
Imamo na raspolaganju 4 bita za prikaz broja. Koliko možemo prikazati brojeva ako prikazujemo samo pozitivne (prirodne) brojeve, a koliko brojeva možemo prikazati ako prikazujemo i negativne brojeve tehnikom dvojnog komplementa?

Ako prikazujemo samo pozitivne brojeve možemo prikazati 16 (2^4) različitih brojeva (od 0000_2 do 1111_2). Ako odvojimo jedan bit za predznak možemo prikazati 8 pozitivnih i 8 negativnih brojeva, dakle ukupno također 16 brojeva što pokazuje tablica.

Prednost upotrebe dvojnog komplementa za zapis cijelih brojeva je u činjenici da se **oduzimanje binarnih brojeva** svodi se na pribrajanje vrijednosti dvojnog komplementa broja koji je trebao biti oduzet. Pritom treba zanemariti eventualni dodatni bit koji nastaje prijenosom kod zbrajanja.

Primjer 1.2.31

$0101_2 - 0010_2 = ?$ ($5_{10} - 2_{10} = ?$)

Prvo se nađe dvojni komplement broja 0010. To je 1101+1=1110.

```
 0101
+1110
-----
10011
```

Peti bit nastao prijenosom treba zanemariti. Rezultat je 0011_2 (3_{10}).

Prikaz realnih brojeva

Realni brojevi prikazuju se u dekadskom sustavu tako da točka odjeljuje cijeli dio od razlomljenog dijela. Na primjer 12.343, 0.000233, 112000.1 su realni dekadski brojevi. Ovakav način prikaza (ponekad nazivan **zapis s nepokretnom točkom**) nepraktičan je za jako velike ili jako male brojeve. U tom slučaju koristi se **eksponencijalni prikaz** realnog broja (nazvan i **zapis s pokretnom točkom**). Na primjer, masa elektrona vrlo je mala i iznosi $9.109*10^{-31}$ kg, a njegov naboj $1.602*10^{-19}$ C. Brzinu svjetla također je praktičnije prikazati u eksponencijalnom prikazu jer se radi o velikom broju ($3*10^8$ m/s). Zapis tog oblika sastoji se od mantise, osnove i eksponenta. Na primjer, broj 15.825 mogli bi u zapisu s pokretnom točkom zapisati ovako:

	mantisa	**eksponenet**	**osnova**
$15.825 = 15.825 * 10^0$	15.825	0	10
$= 0.15825 * 10^2$	0.15825	2	10
$= 15825 * 10^{-3}$	15825	-3	10

Vidi se da se na ovaj način broj može zapisati na mnogo načina. Ako se postavi ograničenje na mantisu takvo da se ona uvijek nalazi u području:

$$\mathbf{B^{-1} \le |mantisa| < 1},$$

gdje je B osnova brojevnog sustava, govorimo o **normiranom prikazu**. Dakle, u normiranom prikazu točka se postavlja ispred najznačajnijeg broja koji nije nula, što pokazuju slijedeći primjeri:

decimalni broj	**mantisa**	**eksponent**
15.825	0.15825	2
0.054	0.54	-1
1234	-0.1234	4
0.0	0.0	0
0.00343	-0.343	-2

U dekadskom brojevnom sustavu mantisa se nalazi između 0.1 i 1. Naravno, iznimka je broj nula. U binarnom brojevnom sustavu (B = 2, $2^{-1} = 0.5$) za mantisu vrijedi: $0.5 \le |\text{mantisa}| < 1$.

U računalu se realni brojevi prikazuju u zapisu s pokretnom točkom. Takav zapis može zauzimati jednu ili dvije riječi (4 ili 8B). Od toga jedan dio (manji) zauzima eksponent, a drugi dio mantisa. Nula kojom započinje mantisa u normiranom prikazu se ne zapisuje. Mantisa i eksponent mogu biti u istom kodu (npr. dvojni komplement), ali ne moraju. Detaljan opis ovakvog zapisa daje npr. standard ANSI/IEEE Std 754-1985 čiji opis prelazi okvire ovog teksta.

Prikaz nebrojevnih veličina u računalu

Osim s brojevima, računala moraju raditi i sa slovima i drugim znakovima. Njih u memoriju računala ne možemo zapisati u izvornom obliku, već samo koristeći unaprijed dogovorenu kombinaciju binarnih znamenki za svaki znak. Takva kombinacija bitova naziva se **kod** određenog znaka.

Da bi se omogućila razmjena podataka između računala potrebno je imati standardiziran kod koji će svi upotrebljavati i razumjeti. Danas je u širokoj upotrebi **ASCII** (*American Standards Code for Information Interchange*). To je osam-bitni kod (kod čija je duljina 8 bita), koji omogućuje prikaz velikih i malih slova, specijalnih znakova (npr. *, +, =, ?, $, %, itd.), te upravljačkih znakova (npr. početak poruke, kraj poruke, novi red, itd.). Ukupno je s osam bita moguće prikazati 256 (2^8=256) različitih znakova. Međutim, prvih 128 znakova je zaista standardizirano, a preostalih 128 nije jedinstveno standardizirano. Razlog tome je što je originalni ASCII koristio 7 bita. Dodatnih 128 kodova za novih 128 znakova dobiveno je dodavanjem jednog bita, te je dobiven tzv. prošireni skup znakova. Tvrtka IBM koristi neke od dodatnih 128 kodova za prikaz slova koja su specifična za različite evropske zemlje. Naime, ne treba zaboraviti da je riječ o američkom standardu, koji ne vodi računa o specifičnostima drugih zemalja. Slijedeća slika prikazuje ASCII kod s tzv. kodnom stranicom IBM 852 (Latin II), kod koje se među gornjih 128 kodova nalaze kodovi za slova slavenskih jezika, pa i hrvatskog.

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ń
4	♦	36	$	68	D	100	d	132	ä	164	Ą	196	─	228	ń
5	♣	37	%	69	E	101	e	133	ů	165	ą	197	┼	229	ň
6	♠	38	&	70	F	102	f	134	ć	166	Ž	198	Ă	230	Š
7	•	39	'	71	G	103	g	135	ç	167	ž	199	ă	231	š
8	◘	40	(	72	H	104	h	136	ł	168	Ę	200	╚	232	Ŕ
9	○	41	)	73	I	105	i	137	ë	169	ę	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	Ő	170		202	╩	234	ŕ
11	♂	43	+	75	K	107	k	139	ő	171	ź	203	╦	235	Ű
12	♀	44	,	76	L	108	l	140	î	172	Č	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	Ź	173	ş	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ţ
15	☼	47	/	79	O	111	o	143	Ć	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	đ	240	˝
17	◄	49	1	81	Q	113	q	145	Ĺ	177	▒	209	Đ	241	˛
18	↕	50	2	82	R	114	r	146	í	178	▓	210	Ď	242	ˇ
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	˘
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	ď	244	˙
21	§	53	5	85	U	117	u	149	Ľ	181	Á	213	Ň	245	§
22	▬	54	6	86	V	118	v	150	ľ	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	Ś	183	Ě	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ś	184	Ş	216	ě	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	˙
27	←	59	;	91	[	123	{	155	Ť	187	╗	219	█	251	ű
28	∟	60	<	92	\	124	¦	156	ť	188	╝	220	▄	252	Ř
29	↔	61	=	93	]	125	}	157	Ł	189	Ż	221	Ţ	253	ř
30	▲	62	>	94	^	126	~	158	×	190	ż	222	Ů	254	■
31	▼	63	?	95	_	127	⌂	159	č	191	┐	223	▀	255	

852 Slavic (Latin II)

Slika 1.2.2 Kodna stranica Latin II (IBM 852)

U gornjoj tablici prikazani su dekadski ekvivalenti binarnih kodova. Primjerice, kod slova A zapravo je `01000001`, što je zgodnije prikazati skraćeno heksadecimalno kao 41, ili dekadski 65.

Postoji i drugi način za prikazivanje naših slova (popularno zvani CROSCII kod), koji je stariji od kodne stranice 852, a naše znakove smješta unutar prvih 128 znakova, žrtvujući pri tome neke važne znakove. Rješenje prikazuje slijedeća tablica. Ovaj kod se danas rijetko koristi. U MS DOS operativnom sustavu prevladava IBM 852 kod.

U tablici su prikazani i kodovi koje za prikaz naših znakova koristi tvrtka Microsoft u operativnom sustavu Windows, čime se dodatno komplicira problem naših slova.

Hrvatski znak	**CROSCII**	**IBM 852**	**Microsoft 1250**
Č	94_{10} (^)	172_{10}	200_{10}
Ć	93_{10} ( ])	143_{10}	198_{10}
Đ	92_{10} (\ )	209_{10}	208_{10}
Š	91_{10} ([ )	230_{10}	138_{10}
Ž	64_{10} (@)	166_{10}	142_{10}
č	126_{10} (~)	159_{10}	232_{10}
ć	125_{10} (})	134_{10}	230_{10}
đ	124_{10} (|)	208_{10}	240_{10}
š	123_{10} {)	231_{10}	154_{10}
ž	96_{10} (`)	167_{10}	158_{10}

Poteškoće s našim znakovima mogu nastupiti prilikom sortiranja riječi po abecedi. Naime, kod sortiranja se koristi činjenica da numeričke vrijednosti kodova (dakle kodovi shvaćeni kao binarni brojevi) odgovaraju redoslijedu slova u abecedi. Primjerice, kod slova a manji je od koda slova b, a taj je opet manji od koda slova c, itd. Međutim, to vrijedi samo za slova američke abecede koja zauzimaju kodove 65 - 90 (slova A - Z) i 97 - 122 (slova a - z). Zbog toga program koji ne vodi računa o specifičnostima naših slova neće dobro sortirati npr. prezimena s našim slovima, iako će sortiranje engleskih prezimena raditi bez greške.

Pitanja:

1. *Zbog čega se brojevni sustav koji upotrebljavamo zove težinski?*
2. *Zbog čega doprinos broju znamenki 5 u broju 5005 nije jednak?*
3. *Saznajte kakav su brojevni sustav koristili Rimljani. Zbog čega je bio nepraktičan? Koji su ovo brojevi: XI, IX, XC, CX, VI, IV?*
4. *Koje znamenke upotrebljavaju binarni, oktalni, dekadski i heksadecimalni brojevni sustavi?*
5. *Što je to kapacitet n znamenki?*
6. *Koliko različitih brojeva možemo prikazati s 6 znamenki u dekadskom sustavu? Koji broj je pri tome najveći broj?*
7. *Koliko različitih brojeva možemo prikazati s 6 brojeva u binarnom sustavu? Koji je pri tome najveći broj (izražen dekadski)? Napišite taj najveći broj u binarnom sustavu.*
8. *Pretvorite slijedeće binarne brojeve u dekadski sustav: 1010, 11011, 101011, 1111.*
9. *Pretvorite slijedeće binarne brojeve u dekadske: 101, 1, 10, 10101, 11, 101011, 1101.*
10. *Pretvorite slijedeće dekadske brojeve u binarne: 12, 23, 120, 17, 101.*
11. *Brojite u binarnom sustavu od 0 do 20.*
12. *Može li broj 11102 pripadati binarnom sustavu? Zašto?*
13. *Pretvorite slijedeće binarne brojeve u dekadske: 11.11, 1011.0001, 111,111, 10000.101010.*
14. *Pretvorite slijedeće dekadske brojeve u binarne: 0.25, 0.125, 12.5, 50.375, 5.4, 22,1. U slučaju potrebe zaokružite binarni broj na 8 mjesta (desno od točke).*
15. *Zbrojite slijedeće binarne brojeve: 10+11, 101+11, 1010+1100, 1010111+110101. Rezultate provjerite zbrajanjem u dekadskom sustavu.*
16. *Pretvorite slijedeće oktalne brojeve u dekadske: 72, 27, 101, 432.*
17. *Pretvorite slijedeće dekadske brojeve u oktalne: 72, 27, 100, 500, 321.*

18. *Pretvorite slijedeće binarne brojeve u oktalne: 101010101, 1101, 1101111, 110011, 1011000011110001.*
19. *Pretvorite slijedeće oktalne brojeve u binarne: 101, 777, 527, 12345.*
20. *Pretvorite slijedeće heksadecimalne brojeve u dekadske: 11, AA, 1F1, FCF.*
21. *Pretvorite slijedeće dekadske brojeve u heksadecimalne: 10, 100, 220, 321.*
22. *Pretvorite slijedeće heksadecimalne brojeve u binarne: F5A, 111, ABCDEF.*
23. *Pretvorite slijedeće binarne brojeve u heksadecimlne: 110011, 101010101, 111100001111.*
24. *Koliko različitih brojeva možemo prikazati s 3 znamenke u binarnom, oktalnom i heksadecimalnom sustavu? Koji je najveći broj koji možemo prikazati s 3 znamenke u navedenim sustavima?*
25. *Brojite u binarnom, oktalnom i heksadecimalnom sustavu od 0 do 20*
26. *Koji je algoritam (postupak) za dobivanje dvojnog komplementa binarnog broja?*
27. *Uz pretpostavku da brojeve bilježimo s 8 bita, oduzmite slijedeće binarne brojeve: 1001-11, 1100-1111, 100-1000, 1-101. Koristite pribrajanje dvojnog komplementa.*
28. *Napišite u zapisu s pokretnom točkom (normiran prikaz) slijedeće dekadske brojeve: 1.124, 1234.1231, 234.43, 3333, 0.000234, 1000100.*
29. *Koji je najveći i najmanji dekadski broj koji se može prikazati s 6 bita u računalu u slučaju da prikazujemo prirodne brojeve, a koji u slučaju da prikazujemo cijele brojeva (dvojnim komplementom)?*
30. *Čemu služi ASCII kod? Koja je dužina toga koda? Koliko različitih znakova možemo prikazati s ACCII kodom?*
31. *Koliko mjesta u memoriji računala zauzima rečenica koja se sastoji 80 znakova?*
32. *Na čemu se temelji sortiranje?*
33. *U memorijskoj lokaciji dužine 8 bita zapisano je 11110010. Što sve može predstavljati takav zapis?*
34. *Kojim brojevnim sustavima bi mogli pripadati slijedeći brojevi: 0, 10, 12, 34, 99, A1, 553, 1011?*