# Zapis označenih brojeva sa fiksnim brojem bitova

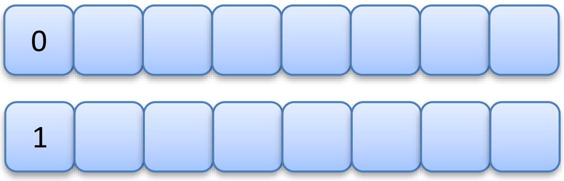
## Znak i apsolutna vrednost (Direktno kodiranje bita znaka)

**OZNAČENI BROJEVI** su u računarskim sistemima **pozitivni, negativni i nula**. U dekadnom sistemu se za označavanje pozitivnih brojeva koristi znak „$+$“, a za negativne „$-$“. Ove oznake se pišu ispred **apsolutne vrednosti** dekadnog broja.

U binarnom brojevnom sistemu ovakav način predstavljanja nije moguć, već se znak predstavlja **bitom $\mathbf{0}$ za $+$ i bitom $\mathbf{1}$ za $-$.**

U programskom jeziku C ili C# je označen ceo broj predstavljen tipom ***int*** ili ***long int***.`

Dok je najmanji neoznačen broj predstavljen sa $\boldsymbol{n\ bitova}$ bio $\mathbf{0}$, a najveći sa $\mathbf{2^n - 1}$, odnosno **interval neoznačenih brojeva sa fiksnim brojem bitova je $\mathbf{0 \div 2^n - 1}$**, sada je **prvi bit rezervisan za predstavljanje znaka, tako da preostaje $\mathbf{2^{n-1}}$ bitova za predstavljanje brojeva**.




Pozitivan $8 - bitni$ označen broj

Negativan $8 - bitni$ označen broj

Pomoću $\boldsymbol{n\ bitova}$ **najmanji broj koji je moguće predstaviti je $\mathbf{-2^{n-1}}$**, a **najveći broj je $\mathbf{2^{n-1} - 1}$**. Dakle, **interval je $\mathbf{-2^{n-1} \div 2^{n-1} - 1}$ (umanjuje se pozitivni deo za 1 jer moramo obeležiti i nulu).**

Broj se u binarnom sistemu piše, dakle, tako što je prvi bit **bit znaka**, a nakon toga se piše **apsolutna binarna vrednost željenog broja**.

U $8 - bitnom$ sistemu:

- najveći broj je $2^{8-1} - 1 = 2^7 - 1 = 127$
- najmanji broj je $-2^{8-1} = -2^7 = -128$

Tip ***int*** se predstavlja sa $32b$ pa je najveći ***int*** broj $2^{32-1} - 1 = 2^{31} - 1$, a najmanji $-2^{32-1} = -2^{31}$.

Na primer:

$$+7_{(10)} \qquad = +|7|_{(10)} \qquad = \boxed{0}111_{(2)}$$
$$-7_{(10)} \qquad = -|7|_{(10)} \qquad = \boxed{1}111_{(2)}$$

Uokvireni bitovi predstavljaju bit znaka

Predstavljanje negativnih brojeva kao u navedenom primeru **nema značajnu primenu**. Glavni razlog je što se nad binarnim brojevima zapisanim pomoću znaka i apsolutne vrednosti **teško obavljaju aritmetičke operacije**.

Još jedan od problema je **dvostruki zapis nule**:

$$+0 \qquad 0000$$
$$-0 \qquad 1000$$

Posledica ove mogućnosti je potreba da se **ispituju dve vrednosti pri proveri jednakosti sa nulom**. Mnogo efikasniji način **predstavljanja <u>označenih brojeva</u>** je pomoću **<u>potpunog (drugog) komplementa</u>**.

## Nepotpuni (prvi) i potpuni (drugi) komplement (NK i PK)

**<u>Potpuni komplement</u>** je najčešći način predstavljanja označenih brojeva. I u ovom slučaju se **znak pozitivnog broja predstavlja sa $\mathbf{0}$**, a **negativan sa $\mathbf{1}$**, ali se **broj $\mathbf{0}$ uvek predstavlja kao pozitivan, odnosno bit znaka je $\mathbf{0}$**.

Posmatraćemo $\mathbf{8-\mathit{bitne}}$ **cele brojeve**:

- **<u>Pozitivni brojevi</u>** se predstavljaju isto: znak i apsolutna vrednost.
  - Npr. $+7_{(10)} = +|7|_{(10)} = \boxed{0}0000111_{(2)}$
- **<u>Negativni brojevi</u>** se dobijaju na sledeći način:
  - **<u>Ispred binarnog zapisa</u>** koji odgovara apsolutnoj vrednosti broja **<u>se dodaju nule</u>**:
    - Npr. $-|7|_{(10)} \rightarrow 00000111_{(2)}$
  - **<u>Sve binarne cifre se invertuju</u>** i tako se dobije **<u>NEPOTPUNI (prvi) KOMPLEMENT (NK)</u>** (invertovanje znači da nule postaju jedinice, a jedinice postaju nule)
    - Npr. $-|7|_{(10)} \rightarrow 11111000_{(2)}\ (\boldsymbol{NK})$
  - **<u>Dobijeni broj se sabere sa 1</u>** i tako se dobije **<u>POTPUNI (drugi) KOMPLEMENT (PK)</u>**
    - Npr.

$$\begin{array}{l} -|7|_{(10)} \rightarrow 11111000_{(2)} \\ +\qquad\qquad\qquad\quad 1 \\ \hline -|7|_{(10)} = 11111001_{(2)}\ (\boldsymbol{PK}) \end{array}$$

**<u>Ukoliko se PK dva puta primeni nad nekim brojem dobija se polazni broj</u>**.

Npr.

$$\begin{array}{rcl} 13 \rightarrow & 00001101 & 8-bitni \text{ binarni zapis} \\ NK \rightarrow & 11110010 & \text{Invertovanje} \\ & +\qquad 1 & \\ \hline PK \rightarrow & 11110011 & \text{Binarni zapis broja } -13 \\ NK \rightarrow & 00001100 & \text{Invertovanje} \\ & +\qquad 1 & \\ \hline PK \rightarrow & 00001101 & \text{Binarni zapis broja } 13 \end{array}$$

Tako je moguće videti vrednost dobijenog negativnog broja u dekadnom zapisu (vršenjem još jednog potpunog komlementa).

*<u>Zadaci za vežbu</u>*

1. U $16-bitnom$ brojevnom sistemu predstaviti brojeve u potpunom komplementu:
   a. $ABC_{(16)}$ b. $-413_{(10)}$ c. $-7654_{(8)}$ d. $-A19_{(16)}$

2. Dobijene vrednosti iz zadatka 1 predstaviti u dekadnom zapisu. NAPOMENA: pozitivni brojevi se odmah konvertuju u dekadni, dok se nad negativnim ponovo izvrši komplementiranje (PK), a onda konvertovanje (na kraju se samo zapiše znak).

*Zadaci za domaći*

1. Date brojeve predstaviti u $16-bitnom$ binarnom zapisu u PK:
   a. $-F2F_{(16)}$  b. $1EE_{(16)}$  c. $-4171_{(8)}$  d. $1234_{(8)}$
2. Binarne brojeve iz 1. zadatka predstaviti u dekadnom zapisu.
3. Napisati najmanji i najveći $16-bitni$ binarni broj. Zapisati te brojeve u heksadekadnom i oktalnom zapisu. NAPOMENA: Ne zapisivati te brojeve sa znacima $+$ ili $-$, već napisani binarni broj samo zapisati heksadekadno ili oktalno.

## Sabiranje i oduzimanje brojeva u nepotpunom komplementu

Pomoću ukupno n+1 bita u nepotpunom komplementu je moguće predstaviti sve brojeve u intervalu od $-2^n+1$ do $2^n-1$. (imamo dve nule, recimo ako je n = 7, možemo da predstavimo brojeve od -127 do 127 sa dve nule: pozitivnom nulom 0 0000000 i negativnom nulom 1 1111111, najveći broj bi bio 0 1111111 odn. 127 a najmanji 1 0000000 odn -127)

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.

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 jer se **<u>oduzimanje može svesti na sabiranje</u>**:

$$X - Y = X + (-Y)$$

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

```
     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 umesto 0, onda bi taj broj bio negativan i da bismo znali koji je bilo bi potrebno pronaći apsolutnu vrednost rezultata, tj. dekomplementirati rezultat. Recimo ako bismo dobili broj 1 01001 to bi u dekadnom sistemu bio broj -22 (konvertovan binarni 0 10110 sa znakom – na koji upućuje prva jedinica u 1 01001)

*Zadatak za vežbu*

1. Izračunati vrednost izraza:
   323 -14

*Zadaci za domaći*

1. Izračunati vrednost izraza:
   a) 580 – 327, b) 703 – 456, c) 128 – 185

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

## Sabiranje i oduzimanje brojeva u potpunom komplementu

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 doda 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 od $2^n$ do $2^n$-1. Ovim se postiže ravnomerna distribucija pozitivnih i negativnih brojeva.

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

**Sabiranje u PK** je lako izvesti, dok se **oduzimanje može svesti na sabiranje**:

$$X - Y = X + (-Y)$$

Jako je bitno **obratiti pažnju na prekoračenja, ukoliko ih ima**. Ukoliko se za bit znaka dobije 0 broj je pozitivan, a ukoliko se dobije 1 broj je negativan.

Kod sabiranja u potpunom komplementu, eventualni prenos na najvišem bitu se zanemaruje.

Primer:

1. Izračunati vrednost sledećeg izraza:
   912 + 32 -1120

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.

*Zadaci za vežbu*

1. U $8 - bitnom$ binarnom zapisu izračunati sledeće brojeve, a potom predstaviti te brojeve u dekadnom zapisu:
   a. $25_{(10)} - 13_{(10)}$
   b. $113_{(10)} - 59_{(10)} - 62_{(10)}$
   c. $177_{(8)} - 62_{(8)}$
   d. $165_{(8)} - 41_{(8)} - 160_{(8)}$
   e. $7D_{(16)} - 2E_{(16)}$
   f. $4B_{(16)} - 65_{(16)} - 3A_{(16)}$

*Zadaci za domaći*

1. Izvršiti oduzimanje u $16 - bitnom$ brojevnom sistemu sledećih brojeva, a potom dobijene brojeve predstaviti u heksadekadnom zapisu:
   a. $657_{(8)} - 732_{(8)} - 276_{(8)}$
   b. $9D8_{(16)} - 3BF_{(16)} - 85C_{(16)}$