# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 1 Brojni sistemi

## 1.1 Nepozicioni i pozicioni sistemi

Svi brojni sistemi se mogu podeliti na nepozicione i pozicione. Kod nepozicionih brojnih sistema svaka cifra ukazuje na istu vrednost bez obzira na kom se mestu nalazi. Primeri nepozicionih brojnih sistema su egipatski i rimski brojni sistem. Cifre koje figurišu kod rimskog sistema su I, V, X, L, C, D i M, koje redom imaju vrednosti 1, 5, 10, 50, 100, 500 i 1000. Njihova vrednost će uvek biti ista, bez obzira na kojoj poziciji u broju se nalaze. Na primer, broj MDCCXXXII, zapisan u rimskom sistemu, ima vrednost 1732 u dekadnom. Oba pojavljivanja cifre C u tom broju označavaju vrednost 100, a sva tri pojavljivanja cifre X u tom broju označavaju vrednost 10.

Kod pozicionih brojnih sistema, ista cifra može imati različite vrednosti u zavisnosti od toga na kom mestu se nalazi. Primer pozicionog brojnog sistema je dekadni sistem. Na primer, kod broja 2322, sve tri dvojke imaju različito značenje. Jedna označava cifru jedinica, druga cifru desetica, a treća cifru hiljada, i shodno tome sve imaju različite vrednosti.

Pozicioni brojni sistemi mogu imati fiksnu ili promenljivu osnovu. Sistemi sa fiksnom osnovom su, na primer, dekadni ili binarni sistem. U prvom slučaju, ona iznosi 10, a u drugom 2. Primer pozicionog sistema sa promenljivom osnovom je sistem za računanje vremena, zapisan u obliku sat:minut:sekunda. Osnovne sistema, gledajući zdesna ulevo su redom 60, 60 i 24, budući da ima 60 sekundi u minutu, 60 minuta u satu i 24 sata u danu. Primer zapisa broja u tom sistemu je 13:25:37.

## 1.2 Pozicioni sistemi sa fiksnom osnovom

Neka je dat proizvoljan ceo broj $x$ u osnovi $n$, zapisan sa $k$ cifara. On se može zapisati u obliku $(x)_n = x_{k-1}x_{k-2}\ldots x_1x_0$. Mesto cifre u zapisu broja se naziva pozicija cifre, a ovde nulta pozicija odgovara cifri najmanje težine, a $(k-1)$-va pozicija cifri najveće težine. Broj cifara $k$ predstavlja ujedno i dužinu broja. Oznaka $(x)_n$ označava da je broj $x$ zapisan u osnovi $n$. Na primer, broj $(3125)_{10}$ je broj zapisan u dekadnom sistemu i dužine je 4. Cifra najmanje težine je 5, a cifra najveće težine 3.

Da bi se definisao brojni sistem sa fiksnom osnovom, dovoljno je navesti vrednosti njegovih cifara, kao i vrednost osnove. U nastavku će biti navedeno nekoliko takvih brojnih sistema.

- Kod dekadnog sistema, osnova je 10, a cifre su 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Primeri zapisa brojeva su 3125 i 4329. Kao i kod svakog sistema iz ove grupe, prva cifra ne sme biti 0, a svaka druga može biti bilo koja cifra iz skupa.

- Kod binarnog sistema, osnova je 2, a cifre su 0 i 1. Primeri zapisa brojeva su 101 i 11111.

- Kod oktalnog sistema, osnova je 8, a cifre su 0, 1, 2, 3, 4, 5, 6 i 7. Primeri zapisa brojeva su 7013 i 41236.

- Kod troičnog sistema, osnova je 3, a cifre su 0, 1 i 2. Primeri zapisa brojeva su 2012 i 111102. Važno je napomenuti da brojevi poput 301 ili 4302 ne mogu pripadati ovom sistemu, jer cifre 3 i 4 ne pripadaju skupu cifara. Za sve cifre važi da je njihova vrednost manja od vrednosti osnove.

- Kod heksadekadnog sistema, osnova je 16, a cifre su 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F. Ovo je primer sistema kod koga se koristi nekoliko prvih slova abecede za zapis, pored standardnih dekadnih cifara. Primeri zapisa brojeva su A3EE5 i 114F.

- Kog negabinarnog sistema, osnova je $-2$, a cifre su 0 i 1. Kao i kod binarnog sistema, primeri zapisa brojeva mogu biti 101 i 11111, ali pri prevođenju u dekadni sistem, oni ne bi imali istu vrednost kao pri prevođenju u dekadni iz binarnog sistema. Ovo je primer sistema gde je osnova negativan ceo broj.

- Drugi primer sistema čija je osnova negativan ceo broj može biti negadekadni sistem, čija je osnova $-10$, a cifre 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Primeri zapisa brojeva mogu biti 3125 i 4329, ali njihove vrednosti nisu identične i u dekadnom brojnom sistemu.

- Osnova sistema može biti i racionalan ili realan broj. Na primer, postoji sistem sa osnovom 0.5 čije su cifre 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Primeri zapisa brojeva mogu biti 3125 i 4329, ali njihove vrednosti nisu identične i u dekadnom brojnom sistemu.

- Kod balansiranog troičnog sistema, osnova je 3 kao kod troičnog, a cifre su 0, 1 i $-1$. Primeri zapisa brojeva su 101101 i $1(-1)0(-1)11$.

## 1.3 Zapis mešovitih brojeva

Broj je mešovit ukoliko ima cifara sa obe strane decimalne tačke. Mešovit broj $x$ se u osnovi $n$ može zapisati kao $(x)_n = x_{k-1}x_{k-2}\ldots x_1x_0.x_{-1}x_{-2}\ldots x_{-l}$. Njegova dužina, odnosno ukupan broj cifara, iznosi $k+l$. Ako je broj oblika $x_{k-1}x_{k-2}\ldots x_1x_0$, radi se o celom broju, a ako je oblika $0.x_{-1}x_{-2}\ldots x_{-l}$, reč je o razlomljenom broju. Ovakav zapis se naziva zapis mešovitih brojeva u običajenom obliku. Na primer, $(1326.43278)_{10}$ i $(1011.011)_2$ su primeri zapisa dva mešovita broja u dekadnom i binarnom sistemu.

Drugi način zapisa mešovitih brojeva je zapis u fiksnom zarezu. Ideja je da se svaki mešovit broj zapisuje pomoću $m$ cifara, od kojih $l$ ($l \leq m$) cifara predstavlja razlomljeni deo, a preostalih $m-l$ cifara celobrojni deo broja. Takav zapis u fiksnom zarezu se označava sa $m.l$. Ukoliko je broj cifara u celobrojnom delu veći od $m-l$, broj ne može da se zapiše. Ukoliko je broj cifara u razlomljenom delu veći od $l$, može se izvršiti odsecanje ili zaokruživanje. Ukoliko je broj cifara u razlomljenom delu manji od $l$, ostatak zapisa se dopunjuje nulama. Da bismo ovo objasnili primerom, pretpostavimo zbog jednostavnosti

da je reč o dekadnom sistemu, jer je slična situacija i u drugim sistemima. U nastavku je prikazan zapis dekadnog mešovitog broja 135.43914 za različite vrednosti $m$ i $l$. Pritom zvezdice označavaju da broj ne može biti zapisan, a donja crta prazninu.

```
8.5 |  135.43914
9.6 | 135.439140
8.6 |   ********
6.3 |    135.439
8.3 |  __135.439
```

Treći način zapisa mešovitih brojeva je zapis u pokretnom zarezu. Ideja je da se svaki broj može napisati u obliku $f \cdot n^e$, gde je $n$ osnova sistema, $f$ frakcija, a $e$ eksponent. Na primer, dekadni broj 123.456 se može napisati u obliku $12.3456 \cdot 10^1$ ili u obliku $123456 \cdot 10^{-3}$. Zapravo, postoji beskonačno mnogo zapisa ovog broja u datom obliku. Zapis oblika $f_0.f_{-1}f_{-2}\dots f_{-l}$, kod koga je $f_0 \neq 0$ se naziva normalizovan. Ovde je sa $f_0.f_{-1}f_{-2}\dots f_{-l}$ označena frakcija. Normalizovan zapis dekadnog broja 123.456 je $1.23456 \cdot 10^2$. Kako se osnova $n$ najčešće podrazumeva, broj se u pokretnom zarezu najčešće piše u obliku $(f, e)$. Tako su neki od zapisa pomenutog dekadnog broja oblika $(12.3456, 1)$, $(123456, -3)$ i $(1.23456, 2)$. Poslednji zapis je i normalizovan.

## 1.4 Prevođenje u dekadni sistem

Neka je broj $x$ u osnovi $n$ zapisan u obliku $x_{k-1}x_{k-2}\dots x_1x_0$. Vrednost broja $x$ u osnovi 10 je jednaka $x_{k-1}n^{k-1} + x_{k-2}n^{k-2} + \dots + x_1n^1 + x_0n^0 = \sum_{i=0}^{k-1} x_in^i$. Ako je broj $x$ razlomljen i oblika $0.x_{-1}x_{-2}\dots x_{-l}$, za njegovo prevođenje u osnovu 10, razmatraju se negativni stepeni osnove $n$. Vrednost broja $x$ u dekadnoj osnova je tada $x_{-1}n^{-1} + x_{-2}n^{-2} + \dots + x_{-l}n^{-l} = \sum_{i=-l}^{0} x_in^i$. Prevođenje mešovitog broja $x$ iz osnove $n$ u osnovu 10 se dobija uniranjem prethodne dve formule i dato je sa

$$\begin{aligned}(x)_n &= (x_{k-1}x_{k-2}\dots x_1x_0.x_{-1}x_{-2}\dots x_{-l})_n \\ &= x_{k-1}n^{k-1} + x_{k-2}n^{k-2} + \dots + x_1n^1 + x_0n^0 + x_{-1}n^{-1} + x_{-2}n^{-2} + \dots + x_{-l}n^{-l} \\ &= \sum_{i=-l}^{k-1} x_in^i.\end{aligned}$$

U prva dva izraza, broj $x$ i njegove cifre u zapisu su dati u osnovi $n$. U formuli gde se vrši sabiranje, i cifre i osnova $n$ su u dekadnoj osnovi. Zbog takvih situacija, na primer, heksadekadne cifre A, B, C, D, E i F, kada figurišu u sumi pri prevođenju, treba tretirati kao dekadne brojeve 10, 11, 12, 13, 14 i 15, respektivno.

U nastavku su dati neki primeri prevođenja iz raznih sistema u sistem sa osnovom 10:

- $(10110)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 16 + 4 + 2 = (22)_{10}$.
- $(3127)_8 = 2 \cdot 8^3 + 1 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 = 3 \cdot 512 + 1 \cdot 64 + 2 \cdot 8 + 7 \cdot 1 = (1623)_{10}$.
- $(1A3)_{16} = 1 \cdot 16^2 + 10 \cdot 16^1 + 3 \cdot 16^0 = 256 + 160 + 3 = (419)_{10}$. Primetimo da je ovde cifri A dodeljena dekadna vrednost 10.
- $(212001)_3 = 2 \cdot 3^5 + 1 \cdot 3^4 + 2 \cdot 3^3 + 1 \cdot 3^0 = (622)_{10}$.

- $(10(-1)(-1)00)_{bt} = 1 \cdot 3^5 + 0 \cdot 3^4 + (-1) \cdot 3^3 + (-1) \cdot 3^2 + 0 \cdot 3^1 + 0 \cdot 3^0 = (207)_{10}$. Ovde je sa *bt* u indeksu označen balansirani troični sistem.
- $(101101)_{-2} = 1 \cdot (-2)^5 + 0 \cdot (-2)^4 + 1 \cdot (-2)^3 + 1 \cdot (-2)^2 + 0 \cdot (-2)^1 + 1 \cdot (-2)^0 = (-35)_{10}$.
- $(145)_{0.5} = 1 \cdot 0.5^2 + 4 \cdot 0.5^1 + 5 \cdot 0.5^0 = 0.25 + 2 + 5 = (7.25)_{10}$.
- $(1101.101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 8 + 4 + 1 + 0.5 + 0.125 = (13.625)_{10}$.
- $(233.12)_8 = 2 \cdot 8^2 + 3 \cdot 8^1 + 3 \cdot 8^0 + 1 \cdot 8^{-1} + 2 \cdot 8^{-2} = 128 + 24 + 3 + 0.125 + 0.03125 = (155.15625)_{10}$.
- $(0.42)_5 = 4 \cdot 5^{-1} + 2 \cdot 2^{-2} = 0.8 + 0.08 = (0.88)_{10}$.

## 1.5 Prevođenje iz dekadnog sistema

Neka ceo broj $x$ treba prevesti iz osnove 10 u osnovu $n$. Neka je pritom zapis broja $x$ u osnovi $n$ dat sa $(x)_n = x_{k-1}x_{k-2}\ldots x_1x_0$. Važi da je $(x)_{10} = (x)_n$, odakle je

$$(x)_{10} = x_{k-1}n^{k-1} + x_{k-2}n^{k-2} + \cdots + x_1n^1 + x_0n^0.$$

Deljenjem obe strane sa $n$ se dobija

$$\frac{(x)_{10}}{n} = x_{k-1}n^{k-2} + x_{k-2}n^{k-3} + \cdots + x_1n^0 + x_0n^{-1}.$$

Ako označimo $(x')_{10} = x_{k-1}n^{k-2} + x_{k-2}n^{k-3} + \cdots + x_1n^0$, dobija se

$$\frac{(x)_{10}}{n} = (x')_{10} + \frac{x_0}{n}.$$

Očigledno je da je $(x')_{10}$ ceo broj. Kako su sve cifre sistema manje od osnove sistema, važi da je $x_0 \leq n$, pa je vrednost $\frac{x_0}{n}$ razlomljeni broj.

Na osnovu prethodnog sledi da se deljenjem polaznog dekadnog broja $(x)_{10}$ osnovom $n$ može dobiti cifra $x_0$ na najnižoj poziciji, jer ona predstavlja ostatak pri tom deljenju. Količnik $(x')_{10}$ predstavlja međurezultat, na koji se može ponoviti prethodni postupak (broj $(x')_{10}$ se sada posmatra kao početna vrednost $(x)_{10}$ koja se deli sa $n$ u cilju dobijanja naredne cifre).

Dakle, algoritam za prevođenje celog dekadnog broja u osnovu $n$ se sastoji od niza uzastopnih deljenja brojem $n$, pri čemu se sva deljenja vrše u dekadnoj osnovi. Ostaci koji se redom dobijaju pri tim deljenjima predstavljaju cifre rezultata u osnovi $n$, a količnici su međurezultati nad kojim se primenjuje uzastopno deljenje. Postupak se ponavlja sve dok se ne dostigne vrednost 0.

U nastavku su dati neki primeri prevođenja celih brojeva iz dekadne osnove u sistem sa osnovom $n$. Pritom se razmatraju samo celobrojne osnove takve da je $n \geq 2$.

- $(3129)_{10} = (300321)_4$:

| 3129 | 782 | 195 | 48 | 12 | 3 |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 0 | 0 | 3 |

- $(3129)_{10} = (C39)_{16}$:

| 3129 | 195 | 12 |
|---|---|---|
| 9 | 3 | C |

- $(23)_{10} = (10111)_2$:

| 23 | 11 | 5 | 2 | 1 |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 |

- $(76)_{10} = (2211)_3$:

| 76 | 25 | 8 | 2 |
|---|---|---|---|
| 1 | 1 | 2 | 2 |

- $(146222)_{10} = (23B2E)_{16}$:

| 146222 | 9138 | 571 | 35 | 2 |
|---|---|---|---|---|
| E | 2 | B | 3 | 2 |

Neka sada razlomljeni broj $x$ treba prevesti iz osnove 10 u osnovu $n$. Neka je pritom zapis broja $x$ u osnovi $n$ dat sa $(x)_n = 0.x_{-1}x_{-2}\ldots x_{-l}$. Važi da je $(x)_{10} = (x)_n$, odakle je

$$(x)_{10} = x_{-1}n^{-1} + x_{-2}n^{-2} + \cdots + x_{-l}n^{-l}.$$

Množenjem obe strane sa $n$ se dobija

$$n(x)_{10} = x_{-1}n^{0} + x_{-2}n^{-1} + \cdots + x_{-l}n^{-(l-1)}.$$

Ako označimo $(x')_{10} = x_{-2}n^{-1} + \cdots + x_{-l}n^{-(l-1)}$, dobija se

$$n(x)_{10} = x_{-1} + (x')_{10}.$$

Primetimo da je $(x')_{10}$ razlomljeni broj. Sledi da se množenjem polaznog dekadnog broja $(x)_{10}$ osnovom $n$ može dobiti cifra $x_{-1}$, budući da ona predstavlja celobrojni deo rezultata. Razlomljeni deo $(x')_{10}$ predstavlja međurezultat, na koji se može ponoviti prethodni postupak (broj $(x')_{10}$ se sada posmatra kao početna vrednost $(x)_{10}$ koja se množi sa $n$ u cilju dobijanja naredne cifre).

Dakle, algoritam za prevođenje razlomljenog dekadnog broja u osnovu $n$ se sastoji od niza uzastopnih množenja brojem $n$, pri čemu se sva množenja vrše u dekadnoj osnovi. Celobrojni delovi koji se dobijaju pri množenju predstavljaju redom sleva nadesno cifre rezultata u osnovi $n$, a razlomljeni delovi su međurezultati nad kojim se primenjuje uzastopno množenje. Postupak se ponavlja sve dok se ne dostigne vrednost 0, s tim što se u nekim slučajevima, kada se rezultat ne može napisati konačnim brojem cifara, vrednost 0 ne dostiže. Tada se postupak može zaustaviti nakon određenog broja iteracija. Rezultujući zapis je u tom slučaju periodičan.

U nastavku su dati neki primeri prevođenja razlomljenih brojeva iz dekadne osnove u sistem sa osnovom $n$. Pritom se razmatraju samo celobrojne osnove takve da je $n \geq 2$.

- $(0.84375)_{10} = (0.11011)_2$:

| 0.84375 | 0.6875 | 0.375 | 0.75 | 0.5 | 0 |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 |

- $(0.375)_{10} = (0.011)_2$:

| 0.375 | 0.75 | 0.5 | 0 |
|---|---|---|---|
| 0 | 0 | 1 | 1 |

- $(0.4)_{10} = (0.011001100...)_2 = (0.\overline{0110})_2$:

| 0.4 | 0.8 | 0.6 | 0.2 | 0.4 | 0.8 | 0.6 | 0.2 | 0.4 | 0.8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

  Pri prevođenju dekadnog broja 0.4 u binarni sistem, zapis u binarnom sistemu je periodičan. Deo koji se ponavlja je nadvučen u rezultatu.

Kod prevođenja mešovitih brojeva iz dekadnog u sistem sa osnovom $n$, potrebno je odvojeno izvršiti prevođenje celobrojnog i razlomljenog dela. Na primer, pri prevođenju broja dekadnog broja 9.375 u binarni sistem, budući da je $(9)_{10} = (1001)_2$ i $(0.375)_{10} = (0.011)_2$, prevod je 1001.011.

## 1.6 Specijalni slučaj prevođenja

Nekad se jednostavno može izvršiti prevođenje iz sistema sa osnovom $m$ u sistem sa osnovom $n$, bez međuprevođenja u dekadni sistem. U tom slučaju broj $m$ mora biti stepen broja $n$ ili broj $n$ mora biti stepen broja $m$.

Ako bismo želeli da broj iz osnove $n$ prevedemo u osnovu $m$, pri čemu važi $m = n^k$, potrebno je da, počevši od decimalne tačke i krećući se levo i desno, izdvajamo grupe od po $k$ cifara. Ukoliko odgovarajuća poslednja grupa ima manje od $k$ cifara, potrebno je da ih dopunimo nulama u istom smeru i zatim svaku grupu od po $k$ cifara ponaosob pretvorimo u cifru u sistemu sa osnovom $m$. Na primer,

$$(101101.01)_2 = (10|1101.01)_2 = (0010|1101.0100)_2 = (2D.4)_{16}.$$

Ako bismo želeli da broj iz osnove $m$ prevedemo u osnovu $n$, pri čemu važi $m = n^k$, potrebno je svaku cifru broja iz osnove $m$ da pretvorimo u osnovu $n$ i napišemo na $k$ mesta (ukoliko je za zapis cifre u osnovi $n$ dovoljno manje od $k$ mesta, zapis treba dopuniti nulama), a zatim izbrisati eventualne nule na početku celog i kraju razlomljenog dela broja. Na primer,

$$(2D.4)_{16} = (0010|1101.0100)_2 = (101101.01)_2.$$

U narednim primerima je primenjen opisani specijalni slučaj prevođenja. U svim primerima važi da je jedna osnova stepen druge.

- $(10110001.0101101)_2 = (010|110|001.010|110|100)_2 = (261.264)_8$.
- $(C1.F1F9)_{16} = (30|01.33|01|33|21)_4 = (3001.33013321)_4$.
- $(D2.EA5)_{16} = (1101|0010.1110|1010|0101)_2 = (11010010.111010100101)_2$.

Primetimo da se ne može na ovaj način direktno prevoditi iz osnove 16 u osnovu 8 (ni obratno), budući da 16 nije stepen broja 8. Pritom je moguće izvršiti međuprevođenje u sistem sa osnovom 2.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 2 Zapis označenih brojeva u računaru

## 2.1 Neoznačeni i označeni brojevi

Ukoliko se za zapis broja ne uzima u obzir njegov znak, broj nazivamo neoznačenim. Nasuprot tome, ukoliko se za zapis znak uzima u obzir, broj je označen. Na primer, dekadni brojevi 425 i 12345 su neoznačeni, a brojevi +425, −12345 i 2267 su označeni. Iako je pozitivan znak izostavljen, u kontekstu označenih brojeva, broj 2267 se smatra pozitivnim brojem, a znak + se podrazumeva.

Neoznačeni celi brojevi u proizvoljnoj osnovi se u računaru zapisuju kao pri njihovom standardnom zapisu. Neka je $(x)_n = x_{k-1}x_{k-2}\ldots x_1x_0$ proizvoljan broj u osnovi $n$, zapisan u računaru kao neoznačen ceo broj sa $k$ cifara. Kako za svaku cifru imamo $n$ mogućnosti za zapis (svaka cifra može uzeti neku od vrednosti $0, 1, \ldots, n-1$), ukupan broj različitih brojeva koji može da se zapiše na ovaj način je $n^k$. Specijalno, ako je $n = 2$, taj broj iznosi $2^k$, a interval u kome se brojevi mogu napisati na ovaj način je $[0, 2^k - 1]$. Ako je $(x)_2 = x_{k-1}x_{k-2}\ldots x_1x_0$ neoznačen binarni broj sa $k$ cifara, njegova dekadna vrednost se može dobiti standardnim prevođenjem iz binarnog u dekadni sistem i iznosi

$$2^0x_0 + 2^1x_1 + \cdots + 2^{k-1}x_{k-1} = \sum_{i=0}^{k-1} 2^i x_i.$$

Nasuprot neoznačenim brojeva, kod označenih brojeva, pored samog broja, treba voditi računa o znaku. Na jedan način treba zapisivati pozitivne, a na drugi negativne brojeve. U računaru bi znakovi + i − trebalo takođe da budu predstavljeni brojevima. Zbog toga se izdvaja mesto za dodatnu cifru. U zavisnosti od toga na koji način se celi brojevi zapisuju u računaru kao označeni brojevi, razlikujemo sledeća četiri zapisa:

- znak i apsolutna vrednost,
- neptpuni komplement,
- potpuni komplement,
- višak $k$.

Za svaki od ovih zapisa će biti posebno reči u nastavku. Svi razmatrani brojevi će biti celi brojevi čija je osnova ceo broj $n$ takav da je $n \geq 2$. Biće razmotrene i osnovne operacije u tim zapisima, među kojima su:

- konverzija između zapisa različitih dužina,

- promena znaka,
- sabiranje,
- oduzimanje.

Pri računskim operacijama, razmatraćemo samo binarne brojeve, a opisani postupci se mogu uopštiti i na proizvoljnu osnovu $n$.

## 2.2 Znak i apsolutna vrednost

Neka je dat proizvoljan pozitivan broj $(x)_n = x_{k-2} \dots x_1 x_0$ u osnovi $n$ sa $(k-1)$-om cifrom. On se sa $k$ cifara u zapisu znak i apsolutna vrednost piše kao $0x_{k-2} \dots x_1 x_0$. Drugim rečima, pozitivni brojevi se u ovom zapisu pišu tako što im se na mesto najveće težine doda cifra 0 za znak, a apsolutna vrednost broja se dopiše u nastavku. Neka je sa $n' = n - 1$ označena najveća cifra sistema sa osnovom $n$. Negativan broj $(x)_n = -x_{k-2} \dots x_1 x_0$ se u zapisu znak i apsolutna vrednost piše kao $n' x_{k-2} \dots x_1 x_0$. Drugim rečima, cifra najveće težine koja označava negativan znak broja je najveća cifra sistema, a nakon nje se dopisuje apsolutna vrednost broja. Tako se, na primer, dekadni broj 345 zapisuje kao 0345, a dekadni broj $-345$ kao 9345. Heksadekadni broj 6577A bi se zapisivao kao 06577A, a njegova negativna varijanta $-6577A$ ima zapis F6577A, budući da je F najveća cifra heksadekadnog sistema. Uopšte uzev, ceo broj $(x)_n = \pm x_{k-2} \dots x_1 x_0$ se u zapisu znak i apsolutna vrednost zapisuje u obliku $x_{k-1} x_{k-2} \dots x_1 x_0$, gde je $x_{k-1} = 0$ ako je broj pozitivan, a $x_{k-1} = n'$, ako je broj negativan.

Ako ceo broj ima $k-1$ cifru, najmanji broj cifara na koji može biti zapisan u ovom zapisu iznosi $k$, budući da je potrebna jedna cifra za zapis znaka. Isti broj može biti zapisan i na više mesta od $k$, samo je potrebno odgovarajuća cifarska mesta između cifre za znak i apsolutne vrednosti dopuniti nulama. Tako bi se taj broj pisao u obliku $x_{k-1} 0 \dots 0 x_{k-2} \dots x_1 x_0$, gde je između cifre za znak $x_{k-1}$ i apsolutne vrednosti broja $x_{k-2} \dots x_1 x_0$ dopisan odgovarajući broj nula. Tako se, na primer, dekadni broj 345 može napisati na 6 mesta u obliku 000345, a dekadni broj $-345$ na 6 mesta kao 900345.

Jedna od osnovnih mana ovog zapisa je dvoznačan zapis nule, što otežava izvođenje računskih operacija. Na primer, u dekadnom sistemu na tri mesta nula se može zapisati kao 000 ili 900, a u binarnom sistemu na četiri mesta, nula se može zapisati kao 0000 ili 1000. Dodatno, ostale računske operacije su nešto komplikovanije, jer posebno treba razmatrati cifru za znak.

Pretpostavimo u nastavku da je $n = 2$, odnosno da je reč o binarnim brojevima. Tada je cifra za znak $x_{k-1} = 0$ za pozitivne, a $x_{k-1} = 1$ za negativne brojeve. Ako je sa $x_{k-1} x_{k-2} \dots x_1 x_0$ zapisan pozitivan broj u znaku i apsolutnoj vrednosti, njegova vrednost je $\sum_{i=0}^{k-2} 2^i x_i$, a ako je zapisan negativan, vrednost je $-\sum_{i=0}^{k-2} 2^i x_i$. U opštem slučaju, dekadna vrednost se može dobiti na osnovu zapisa broja prema formuli

$$(-1)^{x_{k-1}} \sum_{i=0}^{k-2} 2^i x_i.$$

U binarnom sistemu, interval brojeva koji se mogu zapisati u znaku i apsolutnoj vrednosti na $k$ mesta iznosi $[-2^{k-1}+1, 2^{k-1}-1]$. Promena znaka je jednostavna, budući da pritom apsolutna vrednost ostaje ista, potrebno je samo promeniti cifru za znak. Ako se vrši prelazak iz pozitivnog u negativan broj, potrebno je cifru 0 promeniti u 1, i

obratno. Ako je ceo broj sa $k$ cifara potrebno zapisati kao ceo broj sa $l$ cifara u znaku i apsolutnoj vrednosti, potrebno je dopuniti ili, ukoliko je to moguće, izbaciti odgovarajući broj 0. Ako je $k < l$, broj se jednostavno može proširiti sa $l - k$ nula između cifre za znak i ostalih cifara. Ako je $k > l$, potrebno je oduzeti $k - l$ nula između cifre za znak i ostalih cifara, ukoliko je to moguće.

U sledećoj tabeli su dati primeri zapisa nekih celih brojeva u znaku i apsolutnoj vrednosti u raznim osnovama:

| Ceo broj | Zapis sa 3 cifre | Zapis sa 4 cifre | Zapis sa 6 cifara |
|---|---|---|---|
| $(345)_{10}$ | Ne može da se zapiše | $(0345)^4_{10}$ | $(000345)^6_{10}$ |
| $(-110)_2$ | Ne može da se zapiše | $(1110)^4_2$ | $(100110)^6_2$ |
| $(-E3)_{16}$ | $(FE3)^3_{16}$ | $(F0E3)^4_{16}$ | $(F000E3)^6_{16}$ |

**Sabiranje i oduzimanje u znaku i apsolutnoj vrednosti.** Sabiranje binarnih brojeva zapisanih u znaku i apsolutnoj vrednosti je slično sabiranju običnih binarnih brojeva, s tim što ovde treba voditi računa o znaku. Razlikujemo sledeće slučajeve:

- Ukoliko se sabiraju dva pozitivna broja, znak rezultata je pozitivan, a apsolutna vrednost rezultata je jednaka zbiru apsolutnih vrednosti sabiraka. Na primer, pri sabiranju $(0001101)^7_2$ i $(0010011)^7_2$, rezultat je pozitivan. Sabiranjem apsolutnih vrednosti u binarnoj osnovi se dobija

```
  001101
+ 010011
--------
  100000
```

  Rešenje je $(0100000)^7_2$.

- Ukoliko se sabiraju dva negativna broja, znak rezultata je negativan, a apsolutna vrednost rezultata je jednaka zbiru apsolutnih vrednosti sabiraka. Na primer, pri sabiranju $(1001101)^7_2$ i $(1010011)^7_2$, rezultat je negativan. Sabiranjem u apsolutnih vrednosti u binarnoj osnovi se dobija

```
  001101
+ 010011
--------
  100000
```

  Rešenje je $(1100000)^7_2$.

- Ukoliko se sabiraju dva broja različitog znaka, znak rezultata je jednak znaku rezultata koji ima veću apsolutnu vrednost. Apsolutna vrednost rezultata se dobija oduzimanjem apsolutnih vrednosti brojeva, pri čemu se oduzima manja apsolutna vrednost od veće. Na primer, kod sabiranja brojeva $(10011)^5_2$ i $(01000)^5_2$, rezultat je pozitivan, a apsolutna vrednost rezultata je

```
  1000
- 0011
------
  0101
```

Rešenje je $(00101)_2^5$.

Oduzimanje se svodi na sabiranje, pri čemu se vrši promena znaka umanjiocu. Ako su $x$ i $y$ dva broja zapisana u znaku i apsolutnoj vrednosti, važi da je $x - y = x + y'$, gde je $y'$ broj koji se dobija promenom znaka broja $y$. Na taj način se i oduzimanje može svesti na jedan od pomenuta tri slučaja sabiranja.

Pri sabiranju dva broja istog znaka može doći do pojave da se rezultat ne može zapisati na isti broj cifara kao sabirci. Tada dolazi do prekoračenja. Na primer, pri sabiranju dva pozitivan broja 0100 i 0111 (ili dva negativna broja 1100 i 1111), rezultat ne može biti zapisan na četiri cifre, već na pet. Tada dolazi do prekoračenja, jer rezultat inače treba da bude zapisan na isti broj cifara kao sabirci. Primetimo da kod sabiranja dva broja različitog znaka (što se svodi na oduzimanje apsolutnih vrednosti) ne može doći do prekoračenja, jer rezultat uvek ostaje u odgovarajućem opsegu unutar kojeg brojevi mogu biti zapisani.

## 2.3 Nepotpuni komplement

Neka je dat proizvoljan pozitivan broj $(x)_n = x_{k-2} \dots x_1 x_0$ u osnovi $n$ sa $(k-1)$-om cifrom. On se sa $k$ cifara u nepotpunom komplementu piše kao $0x_{k-2} \dots x_1 x_0$, odnosno na isti način kao i u znaku i apsolutnoj vrednosti. Neka je sa $n' = n-1$ označena najveća cifra sistema sa osnovom $n$ i neka je $x_i' = n' - x_i$. Negativan broj $(x)_n = -x_{k-2} \dots x_1 x_0$ se u nepotpunom komplementu piše kao $n' x_{k-2}' \dots x_1' x_0'$. Drugim rečima, cifra najveće težine koja označava negativan znak broja je najveća cifra sistema, a sve ostale cifre se dobijaju tako što se oduzmu od najveće cifre sistema (ovaj postupak se naziva komplementiranje). Tako se negativan broj može najpre napisati kao odgovarajući pozitivan, a u drugoj fazi se svaka cifra komplementira oduzimanjem od $n'$. Tako se, na primer, dekadni broj 345 zapisuje kao 0345, a dekadni broj $-345$ kao 9654. Heksadekadni broj 6577A bi se zapisivao kao 06577A, a njegova negativna varijanta $-6577A$ ima zapis F9A885.

Nepotpuni komplement pripada grupi zapisa koji koriste tzv. komplementacionu konstantu. Ideja komplementacione konstante je da se pozitivni brojevi zapisuju kao u znaku i apsolutnoj vrednosti, a negativni oduzimanjem od nje. U slučaju nepotpunog komplementa, za osnovu $n$ i zapis od $k$ cifara, komplementaciona konstanta iznosi $n^k - 1$. Na primer, za dekadne brojeve na 4 mesta vrednost joj je 9999, a za binarne brojeve na 6 mesta vrednost joj je 111111. Oduzimanje od komplementacione konstante se poklapa sa opisanim algoritmom za zapis negativnih brojeva u nepotpunom komplementu.

Ako ceo broj ima $k-1$ cifru, najmanji broj cifara na koji može biti zapisan u ovom zapisu iznosi $k$, budući da je potrebna jedna cifra za zapis znaka. Isti broj može biti zapisan i na više mesta od $k$, samo je potrebno odgovarajuća mesta sleva dopuniti ciframa koje su jednake cifri za znak. Tako bi se taj broj pisao u obliku $x_{k-1} \dots x_{k-1} x_{k-2} \dots x_1 x_0$, gde je levo od cifre za znak $x_{k-1}$ dopisan odgovarajući broj istih cifara. Tako se, na primer, dekadni broj 345 može napisati na 6 mesta u obliku 000345, a dekadni broj $-345$ na 6 mesta kao 999654.

Jedna od osnovnih mana ovog zapisa je dvoznačan zapis nule, što otežava izvođenje računskih operacija. Na primer, u dekadnom sistemu na tri mesta nula se može zapisati kao 000 ili 999, a u binarnom sistemu na četiri mesta, nula se može zapisati kao 0000 ili 1111. Ostale računske operacije se izvode nešto lakše nego u zapisu znak i apsolutna vrednost.

Pretpostavimo u nastavku da je $n = 2$, odnosno da je reč o binarnim brojevima. U

binarnom sistemu, interval brojeva koji se mogu zapisati u nepotpunom komplementu na $k$ mesta iznosi $[-2^{k-1}+1, 2^{k-1}-1]$.

Kod promene znaka, potrebno je izvršiti komplementiranje svake cifre, odnosno svaku cifru oduzeti od najveće cifre sistema, bez obzira da li se radi o prelasku iz pozitivnog u negativan broj ili obratno. Na primer, zapis negativnog broja $(9654)_{10}^4$ menjanjem znaka postaje $(0345)_{10}^4$, i obratno.

Ako je ceo broj sa $k$ cifara potrebno zapisati kao ceo broj sa $l$ cifara u nepotpunom komplementu, potrebno je dopuniti ili, ukoliko je to moguće, izbaciti odgovarajući broj istih cifara za znak sleva. Ako je $k < l$, broj se jednostavno može proširiti sa $l-k$ cifara sleva čija je vrednost jednaka ciframa za znak broja. Ako je $k > l$, potrebno je oduzeti $k-l$ cifara za znak sleva, ukoliko je to moguće.

U sledećoj tabeli su dati primeri zapisa nekih celih brojeva u nepotpunom komplementu u raznim osnovama:

| Ceo broj | Zapis sa 3 cifre | Zapis sa 4 cifre | Zapis sa 6 cifara |
|---|---|---|---|
| $(345)_{10}$ | Ne može da se zapiše | $(0345)_{10}^4$ | $(000345)_{10}^6$ |
| $(-110)_2$ | Ne može da se zapiše | $(1001)_2^4$ | $(111001)_2^6$ |
| $(-E3)_{16}$ | $(F1C)_{16}^3$ | $(FF1C)_{16}^4$ | $(FFFF1C)_{16}^6$ |

**Sabiranje i oduzimanje u nepotpunom komplementu.** Sabiranje u binarnom sistemu u nepotpunom komplementu se obavlja nešto jednostavnije nego u znaku i apsolutnoj vrednosti, budući da ne treba posebno voditi računa o znaku rezultata. Štaviše, cifre za znak se sabiraju ravnopravno kao i ostale cifre. Sabiranje se obavlja u dve faze. U prvoj fazi se vrši klasično sabiranje dva broja u binarnom sistemu, a u drugoj fazi se eventualni prenos na poziciji najveće težine dodaje na rezultat. Preciznije, pri sabiranju brojeva $x_{k-1}x_{k-2}\ldots x_1x_0$ i $y_{k-1}y_{k-2}\ldots y_1y_0$, u prvoj fazi se dobija

$$\begin{array}{r} x_{k-1}x_{k-2}\ldots x_1x_0 \\ +y_{k-1}y_{k-2}\ldots y_1y_0 \\ \hline z'_kz'_{k-1}z'_{k-2}\ldots z'_1z'_0 \end{array}$$

U drugoj fazi se poslednji prenos $z'_k$ (koji može biti 0 ili 1) dodaje na vrednost $z'_{k-1}z'_{k-2}\ldots z'_1z'_0$, gde se dobija rezultat $z_{k-1}z_{k-2}\ldots z_1z_0$. Celokupan postupak sabiranja izgleda ovako:

$$\begin{array}{r} x_{k-1}x_{k-2}\ldots x_1x_0 \\ +y_{k-1}y_{k-2}\ldots y_1y_0 \\ \hline z'_{k-1}z'_{k-2}\ldots z'_1z'_0 \\ +\qquad\qquad z'_k \\ \hline z_{k-1}z_{k-2}\ldots z_1z_0 \end{array}$$

Na primer, pri sabiranju brojeva 00110 i 00111 u nepotpunom komplementu se u prvoj fazi dobija

```
  00110
+ 00111
-------
  01101
```

Kako je poslednji prenos 0, to je ujedno i rezultat i drugu fazu nije potrebno izvršavati. Pri sabiranju brojeva 11001 i 01110 se dobija

```
  11001
+ 01110
-------
1|00111
```

Prenos 1 je odvojen uspravnom crtom od ostatka. Kako je prenos 1, potrebno je izvršiti drugu fazu, radi dobijanja konačnog rezultata:

```
  00111
+     1
-------
  01000
```

Oduzimanje se svodi na sabiranje, pri čemu se vrši promena znaka umanjiocu na način kako se to i radi u nepotpunom komplementu. Ako su $x$ i $y$ dva broja zapisana u nepotpunom komplementu, važi da je $x - y = x + y'$, gde je $y'$ broj koji se dobija promenom znaka broja $y$.

Pri sabiranju dva broja istog znaka može doći do pojave prekoračenja. Prekoračenje u nepotpunom komplementu se javlja kada se pri sabiranju dva broja istog znaka ne dobija rezultat tog znaka. Preciznije, prekoračenje se javlja ako zbir pozitivnih brojeva nije pozitivan ili ako zbir negativnih brojeva pri sabiranju nije negativan. Ovo znači da rezultat može biti drugog znaka ili da se dobija nekorektan zapis, što znači da prva cifra rezultata nije ni najmanja ni najveća cifra sistema (u binarnom sistemu će uvek rezultat svakako uvek biti pozitivan ili negativan). Prekoračenje se jedino može javiti kod sabiranja dva broja istog znaka, a nikad se ne javlja ako su sabirci različitog znaka.

Prekoračenje ne treba mešati sa prenosom na mestu cifre najveće težine. Nekada može doći do prekoračenja, a da nema prenosa, i obratno. Najpre treba izvršiti obe faze sabiranja, a onda razmotriti znak rezultata i time ustanoviti da li je ili nije došlo do prekoračenja.

Na primer, pri sabiranju brojeva 0100 i 0111 se dobija

```
  0100
+ 0111
------
  1011
```

Kako nema prenosa na cifri najveće težine, rezultat je 1011. Međutim, sabiranjem dva pozitivna broja se nije dobio pozitivan zbir, pa je došlo do prekoračenja. Nasuprot tome, kod sabiranja brojeva 10110 i 10111 u se u prvoj fazi dobija

```
  10110
+ 11111
-------
1|10101
```

U drugoj fazi, pri sabiranju se jedinicom sledi

```
  10101
+     1
-------
  10110
```

Sabiranjem dva negativna broja se dobio negativan rezultat, pa ovde ne dolazi do prekoračenja.

## 2.4 Potpuni komplement

Neka je dat proizvoljan pozitivan broj $(x)_n = x_{k-2} \dots x_1 x_0$ u osnovi $n$ sa $(k-1)$-om cifrom. On se sa $k$ cifara u potpunom komplementu piše kao $0x_{k-2} \dots x_1 x_0$, odnosno na isti način kao i u znaku i apsolutnoj vrednosti i nepotpunom komplementu. Neka je sa $n' = n - 1$ označena najveća cifra sistema sa osnovom $n$ i neka je $x'_i = n' - x_i$. Negativan broj $(x)_n = -x_{k-2} \dots x_1 x_0$ se u potpunom komplementu piše kao $n' x'_{k-2} \dots x'_1 x'_0 + 1$. Drugim rečima, cifra najveće težine koja označava negativan znak broja je najveća cifra sistema, a sve ostale cifre se dobijaju tako što se oduzmu od najveće cifre sistema (ovaj postupak se naziva komplementiranje), a zatim se na dobijeni broj doda vrednost 1. Tako se negativan broj može najpre napisati kao odgovarajući pozitivan, zatim se u drugoj fazi svaka cifra komplementira oduzimanjem od $n'$, a u trećoj fazi se vrši dodavanje cifre 1 na dobijeni broj. Na primer, u potpunom komplementu se dekadni broj 345 zapisuje kao 0345, a dekadni broj $-345$ kao 9655. Heksadekadni broj 6577A bi se zapisivao kao 06577A, a njegova negativna varijanta $-6577A$ ima zapis F9A886.

Potpuni komplement takođe pripada grupi zapisa koji koriste komplementacionu konstantu za zapis brojeva. Ideja komplementacione konstante i ovde je da se pozitivni brojevi zapisuju kao u znaku i apsolutnoj vrednosti, a negativni oduzimanjem od nje. U slučaju potpunog komplementa, za osnovu $n$ i zapis od $k$ cifara, komplementaciona konstanta iznosi $n^k$. Na primer, za dekadne brojeve na 4 mesta vrednost joj je 10000, a za binarne brojeve na 6 mesta vrednost joj je 1000000. Oduzimanje od takve komplementacione konstante se poklapa sa opisanim algoritmom za zapis negativnih brojeva u potpunom komplementu.

Ako ceo broj ima $k - 1$ cifru, najmanji broj cifara na koji može biti zapisan u ovom zapisu iznosi $k$, budući da je potrebna jedna cifra za zapis znaka. Isti broj može biti zapisan i na više mesta od $k$, samo je potrebno odgovarajuća mesta sleva dopuniti ciframa koje su jednake cifri za znak. Tako bi se taj broj pisao u obliku $x_{k-1} \dots x_{k-1} x_{k-2} \dots x_1 x_0$, gde je levo od cifre znak $x_{k-1}$ dopisan odgovarajući broj istih cifara. Tako se, na primer, dekadni broj 345 može napisati na 6 mesta u obliku 000345, a dekadni broj $-345$ na 6 mesta kao 999655.

Za razliku od znaka i apsolutne vrednosti i nepotpunog komplementa, ovde se nula piše na jedinstven način. Na primer, u dekadnom sistemu na tri mesta nula piše kao 000, a u binarnom sistemu na četiri mesta nula se piše kao 0000. Ovaj zapis označava i pozitivnu i negativnu nulu. Time su i računske operacije jednostavnije nego u prethodnim zapisima.

Pretpostavimo u nastavku da je $n = 2$, odnosno da je reč o binarnim brojevima. Tada je cifra za znak $x_{k-1} = 0$ za pozitivne, a $x_{k-1} = 1$ za negativne brojeve. Ako je sa $x_{k-1} x_{k-2} \dots x_1 x_0$ zapisan pozitivan broj u potpunom komplementu, njegova vrednost je $\sum_{i=0}^{k-2} 2^i x_i$, a ako je zapisan negativan, vrednost je $-2^{k-1} + \sum_{i=0}^{k-2} 2^i x_i$. U opštem slučaju, dekadna vrednost se može dobiti na osnovu zapisa broja prema formuli

$$-2^{k-1} x_{k-1} + \sum_{i=0}^{k-2} 2^i x_i.$$

U binarnom sistemu, interval brojeva koji se mogu zapisati u potpunom komplementu na $k$ mesta iznosi $[-2^{k-1}, 2^{k-1} - 1]$. Ovo je za jedan broj više nego u prethodna dva zapisa, budući da se 0 može zapisati na jedinstven način.

Kod promene znaka, potrebno je u prvoj fazi izvršiti komplementiranje svake cifre, odnosno svaku cifru oduzeti od najveće cifre sistema. U drugoj fazi se vrši dodavanje

jedinice na dobijeni rezultat. Na primer, zapis negativnog broja $(9655)_{10}^4$ menjanjem znaka postaje $(0345)_{10}^4$, i obratno.

Ako je ceo broj sa $k$ cifara potrebno zapisati kao ceo broj sa $l$ cifara u potpunom komplementu, potrebno je dopuniti ili, ukoliko je to moguće, izbaciti odgovarajući broj istih cifara za znak sleva. Ako je $k < l$, broj se jednostavno može proširiti sa $l - k$ cifara sleva čija je vrednost jednaka ciframa za znak broja. Ako je $k > l$, potrebno je oduzeti $k - l$ cifara za znak sleva, ukoliko je to moguće.

Dokažimo da se u potpunom komplementu konverzijom u zapis veće dužine ne menja vrednost broja, tj. da je takva konverzija validna. Ako je pozitivan broj sa $k$ cifara potrebno zapisati na $l$ cifara ($l > k$), potrebno je dodati $l-k$ nula, pa se pritom očigledno vrednost broja ne menja. Ako je broj negativan, tada je prvi zapis oblika $1x_{k-2} \dots x_1x_0$, a drugi $1 \dots 1x_{k-2} \dots x_1x_0$, pri čemu drugi zapis ima $l - k$ jedinica više. Kako su oba zapisa binarna i u potpunom komplementu, vrednost prvog broja je

$$\sum_{i=0}^{k-2} 2^i x_i - 2^{k-1},$$

a drugog

$$\sum_{i=0}^{k-2} 2^i x_i + 2^{k-1} + \dots + 2^{l-1} - 2^l.$$

Kako su prve sume u oba zapisa identične, a pritom važi identitet

$$-2^{k-1} = 2^{k-1} + \dots + 2^{l-1} - 2^l,$$

sledi da su vrednosti oba zapisa jednake.

U sledećoj tabeli su dati primeri zapisa nekih celih brojeva u potpunom komplementu u raznim osnovama:

| Ceo broj | Zapis sa 3 cifre | Zapis sa 4 cifre | Zapis sa 6 cifara |
|---|---|---|---|
| $(345)_{10}$ | Ne može da se zapiše | $(0345)_{10}^4$ | $(000345)_{10}^6$ |
| $(-110)_2$ | Ne može da se zapiše | $(1010)_2^4$ | $(111010)_2^6$ |
| $(-E3)_{16}$ | $(F1D)_{16}^3$ | $(FF1D)_{16}^4$ | $(FFFF1D)_{16}^6$ |

**Sabiranje i oduzimanje u potpunom komplementu.** Sabiranje u binarnom sistemu u potpunom komplementu se obavlja jednostavnije nego u znaku i apsolutnoj vrednosti i nepotpunom komplementu. Zbog toga su zapisi označenih brojeva, kao i odgovarajuće operacije, na današnjim računarima uglavnom implementirane u potpunom komplementu.

Sabiranje u potpunom komplementu se vrši kao klasično sabiranje dva broja u binarnom sistemu, a eventualni prenos se ignoriše. Preciznije, pri sabiranju $x_{k-1}x_{k-2} \dots x_1x_0$ i $y_{k-1}y_{k-2} \dots y_1y_0$, se dobija

$$\begin{array}{r} x_{k-1}x_{k-2} \dots x_1x_0 \\ +y_{k-1}y_{k-2} \dots y_1y_0 \\ \hline z_kz_{k-1}z_{k-2} \dots z_1z_0 \end{array}$$

Rezultat je uvek $z_{k-1}z_{k-2} \dots z_1z_0$, bez obzira na vrednost poslednjeg prenosa $z_k$ (on se briše). Na primer, pri sabiranju brojeva 00110 i 00111 u potpunom komplementu se dobija

```
  00110
+ 00111
-------
  01101
```

Pri sabiranju brojeva 11001 i 01110 se dobija

```
  11001
+ 01110
-------
  00111
```

Prenos 1 na poziciji najveće težine se ingoriše.

Oduzimanje se svodi na sabiranje, pri čemu se vrši promena znaka umanjiocu na način kako se to i radi u potpunom komplementu. Ako su $x$ i $y$ dva broja zapisana u potpunom komplementu, važi da je $x - y = x + y'$, gde je $y'$ broj koji se dobija promenom znaka broja $y$.

Pri sabiranju dva broja istog znaka može doći do pojave prekoračenja. Prekoračenje u potpunom komplementu se javlja kada se pri sabiranju dva broja istog znaka ne dobija rezultat tog znaka. Preciznije, prekoračenje se javlja ako zbir pozitivnih brojeva nije pozitivan ili ako zbir negativnih brojeva pri sabiranju nije negativan. Ovo znači da rezultat može biti drugog znaka ili da se dobija nekorektan zapis, što znači da prva cifra rezultata nije ni najmanja ni najveća cifra sistema (u binarnom sistemu će uvek rezultat svakako uvek biti pozitivan ili negativan). Prekoračenje se jedino može javiti kod sabiranja dva broja istog znaka, a nikad se ne javlja ako su sabirci različitog znaka.

Prekoračenje ne treba mešati sa prenosom na mestu cifre najveće težine. Nekada može doći do prekoračenja, a da nema prenosa, i obratno. Najpre treba izvršiti sabiranje, a onda razmotriti znak rezultata i time ustanoviti da li je ili nije došlo do prekoračenja.

Na primer, pri sabiranju brojeva 0100 i 0111 se dobija

```
  0100
+ 0111
------
  1011
```

Sabiranjem dva pozitivna broja se nije dobio pozitivan zbir, pa je došlo do prekoračenja. Kod sabiranja brojeva 10110 i 10111 u se dobija

```
  10110
+ 11111
-------
  10101
```

Poslednji prenos 1 se ignoriše. Sabiranjem dva negativna broja se dobio negativan rezultat, pa ovde ne dolazi do prekoračenja.

## 2.5 Višak $k$

U ovom zapisu, brojevi se zapisuju tako što se na njihov zapis u potpunom komplementu doda vrednost $k$. Zbog toga su operacije koje se obavljaju sa njima slične operacijama u potpunom komplementu. Na primer, zapis dekadnog broja 345 u višku 4 na 5 mesta bi bio 00349, budući da je 00345 odgovarajući zapis u potpunom komplementu. Zapis negativnog dekadnog broja $-345$ u istom zapisu bi bio 99659.

Zapis višak $k$ se često koristi kada želimo da imamo sortirani poredak brojeva. Na primer, ako je dozvoljeni interval brojeva za zapis $[-2^i, 2^i - 1]$, nekad je zgodno te brojeve zapisati u višku $2^i$, kako bi najmanji broj bio predstavljen svim nulama, a svi brojevi zapisa bili pozitivni, čime se olakšava njihovo poređenje.

Operacije sabiranja i oduzimanja se u višku $k$ vrše slično kao u potpunom komplementu, s tim što je potrebno voditi računa da je i rezulat sabiranja i oduzimanja takođe zapisan u višku $k$. Odnosno, nakon operacije sabiranja, od rezultata treba oduzeti $k$, a nakon operacije oduzimanja rezultatu treba dodati $k$. Preciznije, neka su $x$ i $y$ zapisa dva broja u potpunom komplementu, a $x' = x + k$ i $y' = x + k$ odgovarajući zapisi u višku $k$. Pri sabiranju se dobija

$$x' + y' = (x + k) + (y + k) = (x + y) + 2k,$$

pa je takav rezultat zapisan u višku $2k$. Ovo se ispravlja oduzimanjem jednog $k$. Slično, pri oduzimanju se dobija

$$x' - y' = (x + k) + (y + k) = x - y,$$

pa razlika nije zapisana u višku $k$. Ovo se ispravlja dodavanjem $k$ na rezultat.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 3 Množenje i deljenje

## 3.1 Množenje neoznačenih brojeva

Množenje neoznačenih binarnih brojeva je slično množenju dekadnih brojeva, na način kako bi se to radilo pomoću papira i olovke. Posmatrajmo, na primer, množenje dekadnih brojeva 14 i 9. Njihovi binarni zapisi su redom 1110 i 1001. Izračunavanje njihovog proizvoda se odvija na sledeći način:

```
1110 x 1001
-----------
     1110
    0000
   0000
  1110
-----------
  1111110
```

Binarni zapis 1111110 u dekadnom sistemu predstavlja broj 126. Mada je operacija množenja komutativna, u računaru se prvi i drugi činilac posmatraju odvojeno. Nazovimo prvi činilac množenikom, a drugi množiocem. U svakom koraku se množenik množi ciframa množioca. Ako je cifra množioca 1, tada se potpiše sâm množenik, a ako je cifra množioca 0, tada se dopiše onoliko nula koliko množenik ima cifara. Možemo primetiti i da se u svakom koraku međurezultat uvlači za pojedno mesto ulevo. Pretpostavimo, radi lakšeg snalaženja, da množenik i množilac imaju jednak broj cifara. Ako je taj broj cifara jednak $k$, tada proizvod može imati najviše $2k$ cifara. U našem primeru, činioci imaju po 4 cifre, a proizvod 7. Da je postojao prenos na poslednjoj poziciji, proizvod bi imao 8 cifara. Na osnovu svih pomenutih razmatranja, može se konstruisati hardverski algoritam za množenje dva neoznačena cela binarna broja.

Neka su data dva neoznačena cela binarna broja sa po $k$ cifara i neka je potrebno izračunati njihov proizvod. Za zapis njihovog proizvoda će biti dovoljno $2k$ bitova. Prvi činilac nazovimo množenikom i označimo ga sa $M$, a drugi množiocem i označimo ga sa $P$. $M$ i $P$ možemo smatrati binarnim registrima koji su dužine $k$. Registri zapravo predstavljaju niske od $k$ bitova (nula ili jedinica). Za algoritam su nam potrebni još registar $A$ dužine $k$ i jednobitni registar $C$.

Na početku algoritma u registru $M$ je upisan množenik, a u registru $P$ množilac, kao neoznačeni binarni brojevi sa $k$ cifara. U registre $A$ i $C$ se upisuje onoliko nula kolika je odgovarajuća dužina registra. Neka je poslednji bit registra $P$ označen sa $P_0$. Algoritam se izvršava u $k$ koraka od kojih se svaki sastoji iz dva dela:

- U prvom delu, ako je $P_0 = 0$, ne vrši se nikakva akcija. Ako je $P_0 = 1$, vrši se sabiranje brojeva koji su u registrima $A$ i $M$, a dobijeni rezultat postaje nova vrednost registra $A$. Eventualni prenos koji se može pojaviti prilikom sabiranja se upisuje u registar $C$.
- U drugom delu se registri $C$, $A$ i $P$ posmatraju kao jedinstven registar od $2k + 1$ bitova, tako što se nadovežu jedan na drugi. Pritom se vrši logičko pomeranje pomeranje registra $CAP$ udesno. Logičko pomeranje znači da će se svi bitovi osim poslednjeg pomeriti za jedno mesto udesno, poslednji će se izgubiti, a na mesto prvog bita sleva novodobijenog broja se dodaje 0. Na primer, logičko pomeranje udesno niza od šest bitova 100101 daje kao rezultat nisku 010010.

Vrednost proizvoda je smešten u registar $AP$, koji posmatramo kao jedinstven registar od $2k$ bitova, dobijen nadovezivanjem registara $A$ i $P$.

**Primer.** Dekadne brojeve 112 i 9 zapisati kao neoznačene cele brojeve u binarnom sistemu na 8 mesta, pa izvršiti njihovo množenje hardverskim algoritmom. Dobijeni rezultat prevesti u dekadni sistem.

Prevođenjem brojeva 112 i 9 u binarni sistem i njihovim zapisivanjem na 8 mesta dobijaju se vrednosti $(112)_{10} = (0111000)_2^8$ i $(9)_{10} = (00001001)_2^8$. Prema tome, početne vrednosti registara koji označavaju množenik i množilac su redom $M = 0111000$ i $P = 00001001$. Početne vrednosti ostalih registara se inicijalizuju nulama, odnosno iznose $A = 00000000$ i $C = 0$. Vrednost registra $M$ ostaje nepromenljiva, a vrednosti ostalih registara se menjaju kroz narednih 8 koraka, što je prikazano tabelom ispod.

| Korak | $C$ | $A$ | $P$ | Komentar |
|---|---|---|---|---|
| 0 | 0 | 00000000 | 00001001 | Vrši se inicijalizacija. |
| 1 | 0 | 01110000 | 00001001 | $P_0 = 1$, vrši se sabiranje $A = A + M$. |
| | 0 | 00111000 | 00000100 | Registar $CAP$ se logički pomera udesno. |
| 2 | 0 | 00111000 | 00000100 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00011100 | 00000010 | Registar $CAP$ se logički pomera udesno. |
| 3 | 0 | 00011100 | 00000010 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00001110 | 00000001 | Registar $CAP$ se logički pomera udesno. |
| 4 | 0 | 01111110 | 00000001 | $P_0 = 1$, vrši se sabiranje $A = A + M$. |
| | 0 | 00111111 | 00000000 | Registar $CAP$ se logički pomera udesno. |
| 5 | 0 | 00111111 | 00000000 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00011111 | 10000000 | Registar $CAP$ se logički pomera udesno. |
| 6 | 0 | 00011111 | 10000000 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00001111 | 11000000 | Registar $CAP$ se logički pomera udesno. |
| 7 | 0 | 00001111 | 11000000 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00000111 | 11100000 | Registar $CAP$ se logički pomera udesno. |
| 8 | 0 | 00000111 | 11100000 | $P_0 = 0$, ne vrši se nikakva akcija. |
| | 0 | 00000011 | 11110000 | Registar $CAP$ se logički pomera udesno. |

Rezultat množenja je sadržan u registru $AP$. Prevođenjem u dekadni sistem se dobija

$$(1111110000)_2 = (1008)_{10}.$$

## 3.2 Množenje označenih brojeva (Butov algoritam)

Ovde će biti predstavljen hardverski algoritam za množenje označenih celih binarnih brojeva zapisanih u potpunom komplementu, koji nosi naziv Butov algoritam. Neka su data dva cela binarna broja zapisana sa po $k$ cifara u potpunom komplementu i neka je potrebno izračunati njihov proizvod. Za zapis njihovog proizvoda će biti dovoljno $2k$ bitova. Prvi činilac nazovimo množenikom i označimo ga sa $M$, a drugi množiocem i označimo ga sa $P$. $M$ i $P$ možemo smatrati binarnim registrima koji su dužine $k$. Od pomoćnih registara se koriste $A$ i $P_{-1}$, od kojih prvi ima $k$ bitova, a drugi 1 bit. Oba pomoćna registra se na početku inicijalizuju nulama. Neka je poslednji bit registra $P$ označen sa $P_0$. Algoritam se izvršava u $k$ koraka od kojih se svaki sastoji iz dva dela:

- U prvom delu se posmatra par registara $P_0P_{-1}$. Ako je $P_0P_{-1} = 00$ ili $P_0P_{-1} = 11$, ne vrši se nikakva akcija. Ako je $P_0P_{-1} = 01$, vrši se sabiranje brojeva koji su u registrima $A$ i $M$, a dobijeni rezultat postaje nova vrednost registra $A$. Ako je $P_0P_{-1} = 10$, vrši se oduzimanje brojeva koji su u registrima $A$ i $M$. Kako se operacije sabiranja i oduzimanja vrše u potpunom komplementu, nije potrebno razmatrati poslednji prenos. Pritom se oduzimanje može izvršiti direktno, ili se svesti na sabiranje, gde se $A - M$ svodi na $A + M'$, pri čemu je $M'$ registar čija je vrednost promenjenog znaka broja koji je u registru $M$.

- U drugom delu se registri $A$, $P$ i $P_{-1}$ posmatraju kao jedinstven registar od $2k + 1$ bitova, tako što se nadovežu jedan na drugi. Pritom se vrši aritmetičko pomeranje pomeranje registra $APP_{-1}$ udesno. Aritmetičko pomeranje znači da će se svi bitovi osim poslednjeg pomeriti za jedno mesto udesno, poslednji će se izgubiti, a na mesto prvog bita sleva novodobijenog broja se dodaje cifra koja je prethodno bila prva. Na primer, aritmetičko pomeranje udesno niza od šest bitova 100101 daje kao rezultat nisku 110010.

Vrednost proizvoda je smešten u registar $AP$, koji se posmatra kao jedinstven registar nadovezanih registara $A$ i $P$.

**Primer.** Dekadne brojeve 103 i $-13$ zapisati kao označene cele brojeve u binarnom sistemu u potpunom komplementu na 8 mesta, pa izvršiti njihovo množenje Butovim algoritmom. Dobijeni rezultat prevesti iz potpunog komplementa u dekadni sistem.

Prevođenjem brojeva 103 i $-13$ u binarni sistem u potpuni komplement na 8 mesta dobija se $(103)_{10} = (01100111)^8_2$ i $(-13)_{10} = (11110011)^8_2$. Prema tome, početne vrednosti registara koji označavaju množenik i množilac su redom $M = 01100111$ i $P = 11110011$. Početne vrednosti ostalih registara se inicijalizuju nulama, odnosno iznose $A = 00000000$ i $P_{-1} = 0$. Vrednost registra $M$ ostaje nepromenljiva, a vrednosti ostalih registara se menjaju kroz narednih 8 koraka, što je prikazano tabelom ispod.

| Korak | $A$ | $P$ | $P_{-1}$ | Komentar |
|---|---|---|---|---|
| 0 | 00000000 | 11110011 | 0 | Vrši se inicijalizacija. |
| 1 | 10011001 | 11110011 | 0 | $P_0P_{-1} = 10$, pa se vrši oduzimanje $A = A - M$. |
| | 11001100 | 11111001 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 2 | 11001100 | 11111001 | 1 | Kako je $P_0P_{-1} = 11$, ne vrši se nikakva akcija. |
| | 11100110 | 01111100 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 3 | 01001101 | 01111100 | 1 | $P_0P_{-1} = 01$, pa se vrši sabiranje $A = A + M$. |
| | 00100110 | 10111110 | 0 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 4 | 00100110 | 10111110 | 0 | Kako je $P_0P_{-1} = 00$, ne vrši se nikakva akcija. |
| | 00010011 | 01011111 | 0 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 5 | 10101100 | 01011111 | 0 | $P_0P_{-1} = 10$, pa se vrši oduzimanje $A = A - M$. |
| | 11010110 | 00101111 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 6 | 11010110 | 00101111 | 1 | Kako je $P_0P_{-1} = 11$, ne vrši se nikakva akcija. |
| | 11101011 | 00010111 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 7 | 11101011 | 00010111 | 1 | Kako je $P_0P_{-1} = 11$, ne vrši se nikakva akcija. |
| | 11110101 | 10001011 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |
| 8 | 11110101 | 10001011 | 1 | Kako je $P_0P_{-1} = 11$, ne vrši se nikakva akcija. |
| | 11111010 | 11000101 | 1 | Registar $APP_{-1}$ se aritmetički pomera udesno. |

Rezultat je $AP = 1111101011000101$. Prevođenjem iz potpunog komplementa u dekadni sistem se dobija

$$(1111101011000101)_2^{16} = (-1339)_{10}.$$

## 3.3 Modifikovani Butov algoritam

Modifikovani Butov algoritam najčešće omogućava efikasnije množenje dva binarna broja u potpunom komplementu. Postupak ćemo objasniti kroz primer množenja dekadnih brojeva $-28$ i 111, zapisanih u potpunom komplementu. Ako se za množilac koristi $k$ bitova, za množenik je potrebno iskoristiti $2k$ bitova. U našem primeru, množenik će biti zapisan na 16, a množilac na 8 bitova.

Najpre se prevođenjem dekadnih brojeva $-28$ i 111 u binarni sistem u potpuni komplement na odgovarajući broj mesta dobijaju zapisi $(-28)_{10} = (1111111111100100)_2^{16}$ i $(111)_{10} = (01101111)_2^8$. Zapis 01101111 se naziva Butov množilac, a od njega se dobija Butov kodirani množilac na sledeći način:

```
 0  1  1  0  1  1  1  1
----------------------
+1  0 -1 +1  0  0  0 -1
```

Za prebacivanje množioca u modifikovani oblik ideja je uočiti uzastopne nizove jedinica krećući se zdesna nalevo. Početak serije jedinica treba obeležiti sa $-1$, a prvu pojavu nule nakon serije jedinica sa $+1$. Na svim ostalim pozicijama treba upisati nule.

Neka je Butov kodirani množilac označen sa $a_7a_6a_5a_4a_3a_2a_1a_0$. U nastavku algoritma, parove izdvajamo zdesna nalevo počevši od pozicije najmanje težine: $(a_1, a_0)$, $(a_3, a_2)$, $(a_5, a_4)$ i $(a_7, a_6)$. Svakom paru $(a_i, a_{i-1})$ treba pridružiti odgovarajuću vrednost sa $(a_{2i+1}, a_{2i}) \to 2a_{2i+1} + a_{2i}$. Moguće vrednosti parova koje se na ovaj način mogu dobiti su $-2$, $-1$, 0, 1 i 2. Za posmatrani primer se dobijaju sledeće vrednosti parova (parovi su numerisani redom sa $i = 0, 1, 2, 3$):

| $i$ | Par cifara | Vrednost |
|---|---|---|
| 0 | $(a_1, a_0) = (0, -1)$ | $2 \cdot 0 + (-1) = -1$ |
| 1 | $(a_3, a_2) = (0, 0)$ | $2 \cdot 0 + 0 = 0$ |
| 2 | $(a_5, a_4) = (-1, +1)$ | $2 \cdot (-1) + 1 = -1$ |
| 3 | $(a_7, a_6) = (+1, 0)$ | $2 \cdot 1 + 0 = 2$ |

Dalje, za svaku vrednost broja $i$ ($i = 0, 1, 2, 3$) prvo treba pomeriti množenik za $2i$ bita ulevo, a zatim tako dobijeni binaran broj treba pomnožiti vrednošću $i$-tog para. Pritom važi:

- Ako je vrednost para 2, množenje se svodi na pomeranje množenika za jednu poziciju ulevo.
- Ako je vrednost para 1, množenik se ne menja.
- Ako je vrednost para 0, rezultat je 0.
- Ako je vrednost para $-1$, množenje se svodi na menjanje znaka množeniku (komplementiranje i dodavanje jedinice na poziciju najmanje težine).
- Ako je vrednost para $-2$, množenje se svodi na menjanje znaka množeniku, a zatim na pomeranje tako dobijenog broja za jednu poziciju ulevo (isti efekat će se postići i ako se promeni redosled promene znaka i pomeranja ulevo).

Konačan proizvod se dobija sabiranjem svih međuproizvoda. Sabiranje se izvodi po pravilima koja važe za sabiranje brojeva u potpunom komplementu. Na taj način se u primeru rezultat dobija sabiranjem međuproizvoda u poslednjoj koloni sledeće tabele:

| $i$ | Vrednost para | Pomeren množenik | Međuproizvod |
|---|---|---|---|
| 0 | $-1$ | 11111111 11100100 | 00000000 00011100 |
| 1 | 0 | 11111111 10010000 | 00000000 00000000 |
| 2 | $-1$ | 11111110 01000000 | 00000001 11000000 |
| 3 | 2 | 11111001 00000000 | 11110010 00000000 |

Sabiranjem se dobija:

```
  00000000 00011100
  00000000 00000000
  00000001 11000000
+ 11110010 00000000
-------------------
  11110011 11011100
```

Prevođenjem rezultata 1111001111011100 iz potpunog komplementa u dekadni sistem se dobija

$$(1111001111011100)_2^{16} = (-3108)_{10}.$$

## 3.4 Deljenje neoznačenih brojeva

Deljenje neoznačenih binarnih brojeva je slično deljenju dekadnih brojeva, na način kako bi se to radilo pomoću papira i olovke. Ovde ćemo govoriti o celobrojnom deljenju. Posmatrajmo, na primer, deljenje dekadnih brojeva 53 i 5. Njihovi binarni zapisi su redom 110101 i 101. Izračunavanje njihovog količnika i ostatka se odvija na sledeći način:

```
110101 : 101 = 1010
101
---
  110
  101
  ---
    11
```

Količnik je $(1010)_2 = (10)_{10}$, a ostatak $(11)_2 = (3)_{10}$. Ukoliko se delilac sadrži u trenutnim ciframa deljenika koje se razmatraju, količniku se dopisuje cifra 1, a delilac se potpisuje ispod razmatranih cifara i vrši se oduzimanje. Inače, ukoliko se ne sadrži, količniku se dopisuje cifra 0. U oba slučaja se na trenutni ostatak dodaje naredna cifra deljenika. Postupak se ponavlja dok god se ne iskoriste sve cifre deljenika. Na osnovu prethodnog opisa, može se konstruisati hardverski algoritam za deljenje dva neoznačena binarna broja.

Neka su data dva neoznačena cela binarna broja sa po $k$ cifara i neka je potrebno izračunati njihov količnik i ostatak celobrojnim deljenjem. Označimo deljenik sa $P$, a delilac sa $M$. $P$ i $M$ možemo smatrati binarnim registrima koji su dužine $k$. Za algoritam deljenja nam je potreban i registar $A$ dužine $k$.

Na početku algoritma u registru $P$ je upisan deljenik, a u registru $M$ delilac, kao neoznačeni binarni brojevi sa po $k$ cifara. U registar $A$ se upisuje $k$ nula. Algoritam se izvršava u $k$ koraka od kojih se svaki sastoji iz dva dela:

- U prvom delu se registri $A$ i $P$ posmatraju kao jedinstven registar od $2k$ bitova, tako što se nadovežu jedan na drugi. Pritom se vrši pomeranje registra $AP$ ulevo. Svi bitovi osim prvog će se pomeriti za jedno mesto ulevo, prvi će se izgubiti, a na mesto poslednjeg bita novodobijenog broja se dodaje 0.
- U drugom delu, ako je $A < M$, ne vrši se nikakva akcija. Inače, ako je $A \geq M$, vrši se oduzimanje $A = A - M$, a u najniži bit registra $P$ se upisuje cifra 1.

Vrednost količnika je na kraju smeštena u registar $P$, a ostatka u registar $A$.

**Primer.** Dekadne brojeve 24 i 9 zapisati kao neoznačene cele brojeve u binarnom sistemu na 8 mesta, pa izvršiti njihovo deljenje hardverskim algoritmom. Dobijeni količnik i ostatak prevesti u dekadni sistem.

Prevođenjem brojeva 24 i 9 u binarni sistem i njihovim zapisivanjem na 8 mesta dobijaju se vrednosti $(24)_{10} = (00011000)_2^8$ i $(9)_{10} = (00001001)_2^8$. Prema tome, početne vrednosti registara koji označavaju deljenik i delilac su redom $P = 00011000$ i $M = 00001001$. Početna vrednost pomoćnog registara je $A = 00000000$. Vrednost registra $M$ ostaje nepromenljiva, a vrednosti ostalih registara se menjaju kroz narednih 8 koraka, što je prikazano tabelom ispod.

| Korak | $A$ | $P$ | Komentar |
|---|---|---|---|
| 0 | 00000000 | 00011000 | Vrši se inicijalizacija. |
| 1 | 00000000 | 00110000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 00110000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 2 | 00000000 | 01100000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 01100000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 3 | 00000000 | 11000000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 11000000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 4 | 00000001 | 10000000 | Registar $AP$ se pomera ulevo. |
| | 00000001 | 10000000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 5 | 00000011 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000011 | 00000000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 6 | 00000110 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000110 | 00000000 | Kako je $A < M$, ne vrši se nikakva akcija. |
| 7 | 00001100 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000011 | 00000001 | Kako $A \geq M$, važi $A = A - M$ i $P_0 = 1$. |
| 8 | 00000110 | 00000010 | Registar $AP$ se pomera ulevo. |
| | 00000110 | 00000010 | Kako je $A < M$, ne vrši se nikakva akcija. |

## 3.5 Deljenje označenih brojeva

Ovde će biti predstavljen hardverski algoritam za deljenje označenih celih binarnih brojeva zapisanih u potpunom komplementu. Neka su data dva cela binarna broja zapisana u potpunom komplementu i neka je potrebno izračunati njihov količnik i ostatak. Neka je deljenik zapisan sa $2k$, a delilac sa $k$ bitova. Deljenik se na početku zapisuje u jedinstven registar $AP$ koji ima $2k$ bitova, koji se dobija nadovezivanjem registara $A$ i $P$, od kojih svaki ima po $k$ bitova. Pritom treba voditi računa da tada, ako je deljenik pozitivan, registar $A$ će počinjati nulama, a ako je negativan, registar $A$ će počinjati jedinicama. Algoritam se izvršava u $k$ koraka od kojih se svaki sastoji iz dva dela:

- U prvom delu se registri $A$ i $P$ posmatraju kao jedinstven registar od $2k$ bitova, tako što se nadovežu jedan na drugi. Pritom se vrši pomeranje registra $AP$ ulevo. Svi bitovi osim prvog će se pomeriti za jedno mesto ulevo, prvi će se izgubiti, a na mesto poslednjeg bita novodobijenog broja se dodaje 0.
- U drugom delu se proverava da li su zadovoljeni odgovarajući uslovi da se izvrši sledeća akcija:
  - Ako su brojevi $A$ i $M$ različitog znaka, vrši se sabiranje $A = A+M$, a u najniži bit registra $P$ se upisuje cifra 1.
  - Ako su brojevi $A$ i $M$ istog znaka, vrši se oduzimanje $A = A - M$, a u najniži bit registra $P$ se upisuje cifra 1.

  Za izvršavanje pomenute akcije, dovoljno je da je ispunjen jedan od sledeća dva uslova:

  - Nakon izvršene akcije sabiranja ili oduzimanja registar $A$ neće promeniti znak.
  - Ako se stiglo do poslednjeg koraka, nakon izvršene akcije sabiranja ili oduzimanja registar $A$ ima vrednost 0.

Inače se opisana akcija neće izvršiti.

Vrednost ostatka je na kraju smeštena u registar $A$. Ukoliko su deljenik i delilac istog znaka, vrednost količnika je smeštena u registar $P$, a ako su razlčitog, za količnik treba uzeti vrednost registra $P$ sa promenjenim znakom.

**Primer.** Dekadne brojeve 24 i $-9$ zapisati kao označene cele brojeve u binarnom sistemu u potpunom komplementu na 16 i 8 mesta, redom, pa izvršiti njihovo deljenje hardverskim algoritmom. Dobijeni količnik i ostatak prevesti iz potpunog komplementa u dekadni sistem.

Prevođenjem brojeva 24 i $-9$ u binarni sistem u potpuni komplement na 16 i 8 mesta dobija se $(24)_{10} = (0000000000011000)_2^{16}$ i $(-9)_{10} = (11110111)_2^8$. Prema tome, početne vrednosti registara koji označavaju deljenik i delilac su redom $AP = 0000000000011000$ i $M = 11110111$. Vrednost registra $M$ ostaje nepromenljiva, a vrednosti ostalih registara se menjaju kroz narednih 8 koraka, što je prikazano tabelom ispod.

| Korak | $A$ | $P$ | Komentar |
|---|---|---|---|
| 0 | 00000000 | 00011000 | Vrši se inicijalizacija. |
| 1 | 00000000 | 00110000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 00110000 | Ne vrši se nikakva akcija. |
| 2 | 00000000 | 01100000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 01100000 | Ne vrši se nikakva akcija. |
| 3 | 00000000 | 11000000 | Registar $AP$ se pomera ulevo. |
| | 00000000 | 11000000 | Ne vrši se nikakva akcija. |
| 4 | 00000001 | 10000000 | Registar $AP$ se pomera ulevo. |
| | 00000001 | 10000000 | Ne vrši se nikakva akcija. |
| 5 | 00000011 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000011 | 00000000 | Ne vrši se nikakva akcija. |
| 6 | 00000110 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000110 | 00000000 | Ne vrši se nikakva akcija. |
| 7 | 00001100 | 00000000 | Registar $AP$ se pomera ulevo. |
| | 00000011 | 00000001 | Vrši se oduzimanje $A = A - M$ i $P_0 = 1$. |
| 8 | 00000110 | 00000010 | Registar $AP$ se pomera ulevo. |
| | 00000110 | 00000010 | Ne vrši se nikakva akcija. |

Ostatak deljenja je sadržan u registru $A$ i iznosi 6. Vrednost registra $P$ je 2. Kako je znak deljenika i delioca različit, prema algoritmu se za vrednost količnika uzima vrednost registra $P$ sa promenjenim znakom, pa je količnik jednak $-2$.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 4 Binarno kodirani dekadni brojevi

## 4.1 Osnovna svojstva

Budući da se nekad pri prevođenju racionalnih brojeva iz dekadne u binarni sistem može dogoditi da se rezultat ne može zapisati na konačan broj cifara, u računaru se nekad sâm dekadni broj ne može zapisati sa stoprocentnom tačnošću. Ukoliko je taj uslov neophodan, pribegava se principu zapisa kojim se svaka dekadna cifra kodira kao određen binarni broj. Na taj način se dobijaju takozvani binarni kodovi dekadnih cifara, skraćeno BCD kod.

BCD kod predstavlja bilo koju funkciju koja svaku dekadnu cifru preslikava u neki niz binarnih cifara. U opštem slučaju kod može biti i različitih dužina za različite brojeve, a može i imati istu vrednost za različite dekadne cifre. Ovde ćemo se koncentrisati na one kodove koji imaju istu dužinu binarne vrednosti za svaku dekadnu cifru. Budući da imamo 10 različitih dekadnih cifara, najmanja moguća dužina koda je 4, što će ovde i biti slučaj. Dodatno, funkcija kodiranja će biti 1-1, odnosno kod će biti jednoznačan, čime će različitim dekadnim ciframa odgovarati različiti binarni kodovi.

Pored jednoznačnosti, poželjno je da BCD kod poseduje i neke od sledećih svojstava:

- Parnost. Ako je kod paran, parnim ciframa odgovaraju parni kodovi, a neparnim ciframa odgovaraju neparni kodovi.
- Komplementarnost. Neka su $a$ i $b$ cifre takve da je $a + b = 9$. Kod je komplementaran ako su bitovi koda za $a$ i bitovi koda za $b$ na odgovarajućim pozicijama komplementarne, u smislu da zbirovi cifara na istim pozicijama tih kodova daju rezultat 1.
- Da bude težinski. Ako je $x$ dekadna cifra koja se kodira, a $x_3x_2x_1x_0$ njen BCD kod, kod je težinski ukoliko postoje brojevi $c_3$, $c_2$, $c_1$ i $c_0$ takvi da je $x = c_0x_0 + c_1x_1 + c_2x_2 + c_3x_3$. Brojeve $c_3$, $c_2$, $c_1$ i $c_0$ nazivamo težinama.
- Najvećoj dekadnoj cifri se pridružuje kod koji, posmatran kao binarni broj, ima najveću vrednost.

Neki primeri BCD kodova su:

- 8421. Ovaj zapis se dobija tako što se dekadna cifra prevede u binarni broj, a zatim eventualno dodaju vodeće nule do dužine zapisa (dužina ovog zapisa, kao i ostalih koji će biti pomenuti je 4).

- Višak 3. Ovaj zapis se dobija tako što se na svaku dekadnu cifru doda vrednost 3, a zatim dobijeni broj prevede u binarni sistem i dopišu odgovarajuće nule do dužine 4.

- Ciklički kod. Ovaj zapis podrazumeva da se zapis za svake dve susedne cifre razlikuje za tačno jedan bit. Pritom se i 0 i 9 smatraju susednim ciframa. Bilo koji kod koji zadovoljava ovo svojstvo se može nazvati cikličkim.

U narednoj tabeli su prikazani zapisi 8421, višak 3 i ciklički kod. Za ciklički kod prikazana je jedna od mogućih varijanti, za koju se zapis susednih cifara razlikuje za jedan bit. Iz tabele se može videti da su svi kodovi jednoznačni. Kod 8421 je paran i težinski, a nije komplementaran. Kod višak 3 je komplementaran, a nije paran i težinski. Za oba koda važi da cifra 9 ima najveći binarni broj za zapis. Navedeni ciklički kod ne zadovoljava nijednu od pomenutih dodatnih osobina.

| Cifra | 8421 | Višak 3 | Ciklički kod |
|---|---|---|---|
| 0 | 0000 | 0011 | 0001 |
| 1 | 0001 | 0100 | 0101 |
| 2 | 0010 | 0101 | 0111 |
| 3 | 0011 | 0110 | 1111 |
| 4 | 0100 | 0111 | 1110 |
| 5 | 0101 | 1000 | 1100 |
| 6 | 0110 | 1001 | 1000 |
| 7 | 0111 | 1010 | 1001 |
| 8 | 1000 | 1011 | 1011 |
| 9 | 1001 | 1100 | 0011 |

## 4.2 Grejov kod

Grejov kod $g(n)$ je bilo koja funkcija koja vrši preslikavanje celog broja $n$ iz intervala $[0, 2^k - 1]$ u binarnu nisku dužine $k$, čije se vrednosti za svaka dva susedna broja razlikuju tačno za jedan bit. Primetimo da ovakva funkcija nije jedinstvena. Grejov kod je takođe i 1-1 funkcija, odnosno različitim celim brojevima odgovaraju različite vrednosti koda.

Primer jednog Grejovog koda dužine $k$ je

$$g(n) = (n)_2^k \oplus \left(\left\lfloor \frac{n}{2} \right\rfloor\right)_2^k.$$

Ovde $(n)_2^k$ označava da je ceo broj $n$ zapisan u binarnom sistemu na $k$ cifara, $\oplus$ označava ekskluzivnu disjunkciju, a $\lfloor n \rfloor$ celobrojni deo broja. Ekskluzivna disjunkcija je logička funkcija koja za dva bita vraća vrednost 1 ako su različiti, a vrednost 0 ako su jednaki. Ovde se na dva binarna broja primenjuje na svaki par cifara ponaosob. Napišimo nekoliko prvih vrednosti ovako definisane funkcije $g(n)$ za $k = 4$:

- $g(0) = (0)_2^4 \oplus \left(\left\lfloor \frac{0}{2} \right\rfloor\right)_2^4 = (0)_2^4 \oplus (0)_2^4 = 0000 \oplus 0000 = 0000.$

- $g(1) = (1)_2^4 \oplus \left(\left\lfloor \frac{1}{2} \right\rfloor\right)_2^4 = (1)_2^4 \oplus (0)_2^4 = 0001 \oplus 0000 = 0001.$

- $g(2) = (2)_2^4 \oplus \left(\left\lfloor \frac{2}{2} \right\rfloor\right)_2^4 = (2)_2^4 \oplus (1)_2^4 = 0010 \oplus 0001 = 0011.$

- $g(3) = (3)_2^4 \oplus \left(\left\lfloor \frac{3}{2} \right\rfloor\right)_2^4 = (3)_2^4 \oplus (1)_2^4 = 0011 \oplus 0001 = 0010.$

Možemo primetiti da se svaki susedni par vrednosti funkcije razlikuje za tačno 1 bit.

Prethodna funkcija se za binaran broj može jednostavnije definisati. Naime, neka je $a = a_{k-1} \dots a_1 a_0$ binaran broj sa $k$ cifara, a $g = g_{k-1} \dots g_1 g_0$ odgovarajući Grejov kod. U slučaju konverzije broja $a$ u kod $g$ važi

$$g_i = \begin{cases} a_i & i = k-1, \\ a_i \oplus a_{i+1} & i < k-1, \end{cases}$$

dok u slučaju konverzije koda $g$ u binarni broj $a$ važi

$$a_i = \begin{cases} g_i & i = k-1, \\ g_i \oplus a_{i+1} & i < k-1. \end{cases}$$

**Primer.** Grejov kod binarnog broja $a = 100011$ je $g = 110010$, budući da je

$$\begin{aligned} g_5 &= a_5 = 1, \\ g_4 &= a_4 \oplus a_5 = 0 \oplus 1 = 1, \\ g_3 &= a_3 \oplus a_4 = 0 \oplus 0 = 0, \\ g_2 &= a_2 \oplus a_3 = 0 \oplus 0 = 0, \\ g_1 &= a_1 \oplus a_2 = 1 \oplus 0 = 1, \\ g_0 &= a_0 \oplus a_1 = 1 \oplus 1 = 0. \end{aligned}$$

Obratno, za Grejov kod $g = 110010$, odgovarajući binarni broj je $a = 100011$, jer važi sledeće:

$$\begin{aligned} a_5 &= g_5 = 1, \\ a_4 &= g_4 \oplus a_5 = 1 \oplus 1 = 0, \\ a_3 &= g_3 \oplus a_4 = 0 \oplus 0 = 0, \\ a_2 &= g_2 \oplus a_3 = 0 \oplus 0 = 0, \\ a_1 &= g_1 \oplus a_2 = 1 \oplus 0 = 1, \\ a_0 &= g_0 \oplus a_1 = 0 \oplus 1 = 1. \end{aligned}$$

## 4.3 Nepakovan i pakovan zapis

Dekadni brojevi u BCD kodu mogu biti zapisani u nepakovanom i pakovanom zapisu. Kod nepakovanog zapisa se svaka cifra zapisuje u zaseban bajt. Kako je potrebno 4 bita za zapis jedne cifre, preostala 4 bita služe da se označi da je u datom bajtu u pitanju cifra. Oznaka se smešta u levi polubajt, a sama cifra u desni polubajt. Kod pakovanog zapisa se dve cifre smeštaju u jedan bajt i svaka zauzima tačno po jedan polubajt. Nepakovani zapis se češće koristi kada je potrebno posebno istaći da je reč o cifri. Ako se podrazumeva da su u pitanju brojevi, pakovani zapis je kompaktniji način zapisa.

Nepakovani zapis može biti zapisan u ASCII ili EBCDIC kodu. Kod EBCDIC koda u levi polubajt se smešta cifra F (oznaka da je u pitanju cifra), a u desni sama cifra broja.

Kod ASCII koda u levi polubajt se smešta cifra 3 (oznaka da je u pitanju cifra), a u desni takođe cifra broja. Na primer, neoznačen dekadni broj 1579 se u nepakovanom zapisu u EBCDIC kodu zapisuje kao F1 F5 F7 F9, a u ASCII kodu kao 31 35 37 39. Par F1 predstavlja heksadekadni zapis i njime je označen jedan bajt. Svakom heksadekadnom cifrom su predstavljena 4 bita, pa par čini ceo bajt. Kod označenih brojeva se na mesto poslednje cifre stavlja kod za znak broja, umesto znaka F ili 3. Pozitivni brojevi se označavaju sa C, a negativni sa D. Tako se, na primer, pozitivni dekadni označeni broj 1579 u EBCDIC kodu zapisuje kao F1 F5 F7 C9, a negativna varijanta $-1579$ u ASCII kodu kao 31 35 37 D9.

Kod pakovanog zapisa se cifra F ili 3, koja označava da je reč o cifri broja, izostavlja. Zbog toga je njihov zapis isti i u ASCII i u EBCDIC kodu. Tako se na primer neoznačen broj 1579 zapisuje kao 15 79, a neoznačen broj 15799 kao 01 57 99. Poslednji bajt se, ukoliko je potrebno, nulama dopunjuje do cele vrednosti. Označeni brojevi se zapisuju tako što se cifra za znak zapisuje u poslednji polubajt. Tako se pozitivan broj 1579 zapisuje u obliku 01 57 9C, a negativan broj $-15799$ kao 15 79 9D.

Promena znaka pri zapisu označenih brojeva u pakovanom ili nepakovanom zapisu je jednostavna. Ukoliko se vrši prelazak iz pozitivnog u negativan broj, cifra C se na odgovarajućem mestu pretvara u cifru D, i obratno. Na primer, za nepakovani zapis u EBCDIC kodu F1 F3 F7 C3, pri promeni znaka, dobija se zapis F1 F3 F7 D3. Pakovani zapis 01 35 6D pri promeni znaka postaje 01 35 6C.

## 4.4 Čen-Ho kodiranje

Primetimo da se pri BCD kodovima dekadnih cifara troši dosta nepotrebne memorije, budući da 4 bita dozvoljavaju zapis 16 različitih karaktera, a ima 10 različitih dekadnih cifara. Čen-Ho kodiranje kodira trojku dekadnih cifara sa 10 bitova. Kako bi standardni BCD kod 3 dekadne cifre kodirao sa 12 bitova, ovde je postignuto 20% uštede. U nastavku će biti prikazana tabela za kodiranje, pomoću koje se od tri dekadne cifre dobija odgovarajući kod od 10 bitova (ova tabela, uključujući i njenu varijantu za dekodiranje ne mora da se uči napamet, već će biti data na ispitu):

| aei | p qrs tuv wxy |
|---|---|
| 000 | 0 bcd fgh jkl |
| 100 | 1 00d fgh jkl |
| 010 | 1 01d bch jkl |
| 001 | 1 10d fgh bcl |
| 011 | 1 11d 00h bcl |
| 101 | 1 11d 01h fgl |
| 110 | 1 11d 10h jkl |
| 111 | 1 11d 11h 00l |

Neka je sada data uređena trojka celih dekadnih cifara $c_2c_1c_0$. Svaku od pomenutih cifara treba zapisati u BCD kodu u zapisu 8421, gde se dobija zapis $abcdefghijkl$. Na osnovu vrednosti bitova $a$, $e$ i $i$ treba se pozicionirati u odgovarajući red tabele, na osnovu čega se jednosavno može izračunati vrednost $pqrstuvwxy$, što je 10 bitova koji predstavljaju odgovarajući kod.

Na primer, trocifren broj 918 se u zapisu 8421 piše kao

$$abcdefghijkl = 100100011000.$$

Kako je $aei = 101$, sledi da se treba pozicionirati u peti red tabele. Iz njega se vidi da je

$$pqrstuvwxy = 111d01hfgl = 1111011000.$$

Ukoliko je dat broj sa $3k$ cifara, dovoljno je po istom principu $k$ puta trojku cifara kodirati sa po 10 bitova, a zatim dobijene kodove sjediniti. Ukoliko broj cifara broja nije deljiv sa 3, treba po potrebi dodati jednu ili dve vodeće nule.

Obratno, pomoću tabele za dekodiranje se uređena desetorka bitova može dekodirati u 3 dekadne cifre. Ukoliko ima $10k$ bitova, oni se primenom algoritma $k$ puta mogu dekodirati u dekadni broj sa $3k$ cifara. Dekodiranje se vrši na osnovu sledeće tabele:

| pqrtu | abcd efgh ijkl |
|---|---|
| 0.... | 0qrs 0tuv 0wxy |
| 100.. | 100s 0tuv 0wxy |
| 101.. | 0tus 100v 0wxy |
| 110.. | 0wxs 0tuv 100y |
| 11100 | 0wxs 100v 100y |
| 11101 | 100s 0wxv 100y |
| 11110 | 100s 100v 0wxy |
| 11111 | 100s 100v 100y |

Neka je, na primer, dat niz od 10 bitova

$$pqrstuvwxy = 1111011000.$$

Kako je $pqrtu = 11101$, treba se pozicionirati u šesti red tabele. Dobija se da je

$$abcdefghijkl = 100s0wxv100y = 100100011000,$$

pa je traženi broj 918.

## 4.5 DPD kodiranje

DPD kodiranje kodira trojku dekadnih cifara sa 10 bitova. Kako bi standardni BCD kod 3 dekadne cifre kodirao sa 12 bitova, ovde je postignuto 20% uštede. DPD kodiranje je slično Čen-Ho kodiranju, ali je nešto efikasnije. U nastavku će biti prikazana tabela za kodiranje, pomoću koje se od tri dekadne cifre dobija odgovarajući kod od 10 bitova (ova tabela, uključujući i njenu varijantu za dekodiranje ne mora da se uči napamet, već će biti data na ispitu):

| aei | pqr stu v wxy |
|---|---|
| 000 | bcd fgh 0 jkl |
| 001 | bcd fgh 1 00l |
| 010 | bcd jkh 1 01l |
| 100 | jkd fgh 1 10l |
| 110 | jkd 00h 1 11l |
| 101 | fgd 01h 1 11l |
| 011 | bcd 10h 1 11l |
| 111 | 00d 11h 1 11l |

Neka je sada data uređena trojka celih dekadnih cifara $c_2c_1c_0$. Svaku od pomenutih cifara treba zapisati u BCD kodu u zapisu 8421, gde se dobija zapis $abcdefghijkl$. Na osnovu vrednosti bitova $a$, $e$ i $i$ treba se pozicionirati u odgovarajući red tabele, na osnovu čega se jednosavno može izračunati vrednost $pqrstuvwxy$, što je 10 bitova koji predstavlja odgovarajući kod.

Na primer, trocifren broj 918 se u zapisu 8421 piše kao

$$abcdefghijkl = 100100011000.$$

Kako je $aei = 101$, sledi da se treba pozicionirati u peti red tabele. Iz njega se vidi da je

$$pqrstuvwxy = fgd01h111l = 0010111110.$$

Ukoliko je dat broj sa $3k$ cifara, dovoljno je po istom principu $k$ puta trojku cifara kodirati sa po 10 bitova, a zatim dobijene kodove sjediniti. Ukoliko broj cifara broja nije deljiv sa 3, treba po potrebi dodati jednu ili dve vodeće nule.

Obratno, pomoću tabele za dekodiranje se uređena desetorka bitova može dekodirati u 3 dekadne cifre. Ukoliko ima $10k$ bitova, oni se primenom algoritma $k$ puta mogu dekodirati u dekadni broj sa $3k$ cifara. Dekodiranje se vrši na osnovu sledeće tabele:

| vwxst | abcd efgh ijkl |
|---|---|
| 0.... | 0pqr 0stu 0wxy |
| 100.. | 0pqr 0stu 100y |
| 101.. | 0pqr 100u 0sty |
| 110.. | 100r 0stu 0pqy |
| 11100 | 100r 100u 0pqy |
| 11101 | 100r 0pqu 100y |
| 11110 | 0pqr 100u 100y |
| 11111 | 100r 100u 100y |

Neka je, na primer, dat niz od 10 bitova

$$pqrstuvwxy = 0010111110.$$

Kako je $vwxst = 11101$, treba se pozicionirati u peti red tabele. Dobija se da je

$$abcdefghijkl = 100r0pqu100y = 100100011000,$$

pa je traženi broj 918.

## 4.6 Sabiranje i oduzimanje u BCD kodu

Pretpostavimo da su dva dekadna broja zapisana u nekom od zapisa u BCD kodu i da je potrebno izračunati njihov zbir ili razliku. Ako se radi o označenim brojevima, oni se mogu, slično pravilima za sabiranje i oduzimanje u znaku i apsolutnoj vrednosti, uvek svesti na sabiranje dva pozitivna broja ili oduzimanje manjeg pozitivnog broja od većeg, pri čemu na kraju jedino treba voditi računa o znaku rezultata. Na primer, ako se sabiraju dva broja istog znaka, dovoljno je sabrati dva odgovarajuća pozitivna broja, a rezultat je onog znaka kao sabirci. Sa druge strane, ako se vrši sabiranje dva broja različitog znaka, sabiranje se može svesti na oduzimanje, gde se oduzima broj sa manjom apsolutnom vrednošću od broja sa većom. Slična situacija je i kod oduzimanja dva označena broja.

Zbog toga možemo u nastavku pretpostaviti da vršimo sabiranje dva neoznačena broja ili oduzimanje dva neoznačena broja, od kojih je prvi veći po apsolutnoj vrednosti.

Ako su dva $n$-tocifrena neoznačena cela dekadna broja $A = a_{n-1}a_{n-2}\dots a_1a_0$ i $B = b_{n-1}b_{n-2}\dots b_1b_0$ zapisana u BCD kodu, za njihovo kodiranje bilo kojim od standardnih pomenutih zapisa (poput, na primer, 8421 ili višak 3) je potrebno $4n$ bitova. Neka je $\alpha$ funkcija koja predstavlja preslikavanje dekadne cifre u konkretan BCD zapis. Kodovi $\alpha(A) = \alpha(a_{n-1})\alpha(a_{n-2})\dots\alpha(a_1)\alpha(a_0)$ i $\alpha(B) = \alpha(b_{n-1})\alpha(b_{n-2})\dots\alpha(b_1)\alpha(b_0)$ su nizovi od $4n$ bitova i predstavljaju zapise brojeva $A$ i $B$, dok $\alpha(a_i)$ i $\alpha(b_i)$ predstavlja četvorku bitova koja kodira jednu dekadnu cifru.

Sabiranje $\alpha(A)+\alpha(B)$ se vrši u dve faze. U prvoj fazi se vrši sabiranje dva neoznačena binarna broja, pri čemu se dobija međurezultat:

$$\begin{array}{r} \alpha(a_{n-1})\alpha(a_{n-2})\dots\alpha(a_1)\alpha(a_0) \\ +\ \alpha(b_{n-1})\alpha(b_{n-2})\dots\alpha(b_1)\alpha(b_0) \\ \hline \alpha(c'_{n-1})\alpha(c'_{n-2})\dots\alpha(c'_1)\alpha(c'_0) \end{array}$$

Dobijeni međurezultat je potrebno korigovati, budući da dobijene četvorke bitova $\alpha(c'_i)$ ne predstavljaju tačne zapise zbira. Zbog toga se u drugoj fazi vrši dodavanje korekcije $\alpha(k_i)$ za svaku grupu bitova $\alpha(c'_i)$. Konačan rezultat se dobija u drugoj fazi, sabiranjem međurezultata i broja koji se dobija nadovezivanjem korekcija:

$$\begin{array}{r} \alpha(c'_{n-1})\alpha(c'_{n-2})\dots\alpha(c'_1)\alpha(c'_0) \\ +\ \alpha(k_{n-1})\alpha(k_{n-2})\dots\alpha(k_1)\alpha(k_0) \\ \hline \alpha(c_{n-1})\alpha(c_{n-2})\dots\alpha(c_1)\alpha(c_0) \end{array}$$

Prva faza je identična za sve zapise, a u drugoj fazi način određivanja korekcije zavisi od vrste zapisa. Oduzimanje se može raditi na analogan način kao sabiranje, s tim što se u obe faze umesto sabiranja binarnih brojeva vrši njihovo oduzimanje. Drugi način da se oduzmu dva broja je da se zapišu u potpunom komplementu, a zatim svedu na sabiranje promenom znaka umanjiocu. U nastavku će biti opisani konkretni algoritmi sabiranja i oduzimanja u zapisima 8421 i višak 3.

**Sabiranje u zapisu 8421.** Neka su data dva neoznačena cela broja sa $n$ cifara $A = a_{n-1}a_{n-2}\dots a_1a_0$ i $B = b_{n-1}b_{n-2}\dots b_1b_0$, čiji su BCD kodovi u zapisu 8421 $\alpha(A) = \alpha(a_{n-1})\alpha(a_{n-2})\dots\alpha(a_1)\alpha(a_0)$ i $\alpha(B) = \alpha(b_{n-1})\alpha(b_{n-2})\dots\alpha(b_1)\alpha(b_0)$. Potrebno je izračunati zbir $\alpha(A) + \alpha(B)$, gde funkcija $\alpha$ svakoj cifri dodeljuje četvorobitni kod u skladu sa zapisom 8421.

Prva faza sabiranja je identična kao opisana prva faza u opštem slučaju, gde se vrši sabiranje dva neoznačena binarna broja od $4n$ cifara. Neka $p'_i \in \{0, 1\}$ označava prenos sa $i$-te četvorke bitova na narednu četvorku bitova. Na primer, $p'_1$ označava eventualni prenos sa četvrtog na peti par pitova gledajući zdesna, a $p'_2$ označava eventualni prenos sa osmog na deveti par pitova gledajući zdesna. Na početku je uvek $p'_0 = 0$.

Druga faza sabiranja je specifična samo za zapis 8421. Konkretno, specifičan je način na koji se određuju korekcije pri sabiranju. Najpre se određuje korekcija za četvorku bitova $\alpha(c'_0)$. Ako je $\alpha(c'_0) \geq (1010)_2$ ili je $p'_1 = 1$, korekcija je $\alpha(k_0) = (0110)_2$. Inače, korekcija je $\alpha(k_0) = (0000)_2$. Izvršimo sada sabiranje $\alpha(c'_0) + \alpha(k_0)$ i neka je $p''_1 \in \{0, 1\}$ odgovarajući prenos. Za određivanje korekcije za četvorku bitova $\alpha(c'_1)$, dovoljno je proveriti da li je $\alpha(c'_1) + p''_1 \geq (1010)_2$ ili je $p'_2 = 1$. Ukoliko je bilo koji od ova dva uslova ispunjen, korekcija je $\alpha(k_1) = (0110)_2$, a inače je $\alpha(k_1) = (0000)_2$. Ovaj

postupak se ponavlja, gde se redom dobijaju korekcije $\alpha(k_2), \alpha(k_3), \ldots, \alpha(k_{n-1})$ i prenosi $p''_3, p''_4, \ldots, p''_n$. Bitovi $p''_{i+1}$ predstavljaju dakle prenos pri sabiranju četvorki bitova $\alpha(c'_i)$ i $\alpha(k_i)$. Uzima se da je $p''_0 = 0$.

Na osnovu prethodno opisanog, može se izvesti pravilo za određivanje korekcije $\alpha(k_i)$ za četvorku bitova međurezultata $\alpha(c'_i)$:

- Ako je $\alpha(c'_i) + p''_i \geq (1010)_2$, korekcija je $\alpha(k_i) = (0110)_2$.
- Ako je $p'_{i+1} = 1$, korekcija je $\alpha(k_i) = (0110)_2$.
- Inače, ako nijedan od prethodna dva uslova nije ispunjen, korekcija je $\alpha(k_i) = (0000)_2$.

Sabiranjem međurezultata i korekcije, dobija se rezulat $\alpha(c_{n-1})\alpha(c_{n-2})\ldots\alpha(c_1)\alpha(c_0)$. Prekoračenje pri sabiranju se javlja ako je $p'_n = 1$ ili $p''_n = 1$. Sabiranje u 8421 se šematski može prikazati na sledeći način:

| | | | | | |
|---|---|---|---|---|---|
| | $\alpha(a_{n-1})$ | $\alpha(a_{n-2})$ | $\ldots$ | $\alpha(a_1)$ | $\alpha(a_0)$ |
| $+$ | $\alpha(b_{n-1})$ | $\alpha(b_{n-2})$ | $\ldots$ | $\alpha(b_1)$ | $\alpha(b_0)$ |
| $p'_n$ | $p'_{n-1}$ | $p'_{n-2}$ | $\ldots$ | $p'_1$ | $p'_0$ |
| | $\alpha(c'_{n-1})$ | $\alpha(c'_{n-2})$ | $\ldots$ | $\alpha(c'_1)$ | $\alpha(c'_0)$ |
| $p''_n$ | $p''_{n-1}$ | $p''_{n-2}$ | $\ldots$ | $p''_1$ | $p''_0$ |
| $+$ | $\alpha(k_{n-1})$ | $\alpha(k_{n-2})$ | $\ldots$ | $\alpha(k_1)$ | $\alpha(k_0)$ |
| | $\alpha(c_{n-1})$ | $\alpha(c_{n-2})$ | $\ldots$ | $\alpha(c_1)$ | $\alpha(c_0)$ |

Svaka kolona, odvojena dvostrukom uspravnom crtom predstavlja četvorku bitova, pri čemu su prenosi $p'_i$ i $p''_i$ desno poravnati. U prvoj fazi se dobija međurezultat

$$\alpha(c'_{n-1})\alpha(c'_{n-2})\ldots\alpha(c'_1)\alpha(c'_0)$$

i prenosi $p'_i$. Druga faza se izvodi u $n$ koraka. Svaki korak se sastoji od određivanja korekcije $\alpha(k_i)$, sabiranja $\alpha(c'_i) + \alpha(k_i) = \alpha(c_i)$ i određivanje prenosa $p''_i$.

Na primer, ako je dekadne brojeve 23492 i 5189 potrebno sabrati u BCD kodu u zapisu 8421 na 5 mesta, ceo postupak izgleda ovako (broju 5189 se dodaju vodeće nule do 5 mesta):

```
  0010 0011 0100 1001 0010
+ 0000 0101 0001 1000 1001
-------------------------
0    0    0    1    0    0
  0010 1000 0110 0001 1011
0    0    0    0    1    0
+ 0000 0000 0000 0110 0110
-------------------------
  0010 1000 0110 1000 0001
```

Rešenje je 28681. Kako su poslednji prenosi u obe faze jednaki 0, nije došlo do prekoračenja. Radi lakšeg razumevanja prethodnog primera, prikažimo kako se on radi postupno. Najpre se u prvoj fazi odradi sabiranje prva dva binarna broja, dobije rezultat i upišu odgovarajući prvi prenosi:

```
  0010 0011 0100 1001 0010
+ 0000 0101 0001 1000 1001
-------------------------
0    0    0    1    0    0
  0010 1000 0110 0001 1011
```

Zatim se druga faza radi korak po korak. Nulti prenos u drugoj fazi se postavi na nulu, odredi se korekcija, a zatim se sabiraju poslednja četiri bita međurezultata sa korekcijom i pritom dobijaju prva četiri bita rezultata i prvi prenos u drugoj fazi:

```
  0010 0011 0100 1001 0010
+ 0000 0101 0001 1000 1001
-------------------------
0    0    0    1    0    0
  0010 1000 0110 0001 1011
                     1    0
+                      0110
                     ------
                       0001
```

Nakon toga se određuje druga korekcija, pa vrši sabiranje odgovarajuće druge četvorke bitova, nakon čega se određuju naredna četiri bita rezultata i drugi prenos u drugoj fazi:

```
  0010 0011 0100 1001 0010
+ 0000 0101 0001 1000 1001
-------------------------
0    0    0    1    0    0
  0010 1000 0110 0001 1011
                0    1    0
+                 0110 0110
                -----------
                  1000 0001
```

Isti postupak se ponavlja naredna tri koraka, nakon čega se završava druga faza i dobija konačan rezultat.

**Oduzimanje u zapisu 8421.** Neka su uvedene iste oznake kao kod sabiranja u zapisu 8421. Oduzimanje se radi na sličan način kao sabiranje, a osnovna razlika je što se umesto sabiranja u obe faze vrši oduzimanje:

| | | | | | |
|---|---|---|---|---|---|
| | $\alpha(a_{n-1})$ | $\alpha(a_{n-2})$ | $\dots$ | $\alpha(a_1)$ | $\alpha(a_0)$ |
| $-$ | $\alpha(b_{n-1})$ | $\alpha(b_{n-2})$ | $\dots$ | $\alpha(b_1)$ | $\alpha(b_0)$ |
| $p'_n$ | $p'_{n-1}$ | $p'_{n-2}$ | $\dots$ | $p'_1$ | $p'_0$ |
| | $\alpha(c'_{n-1})$ | $\alpha(c'_{n-2})$ | $\dots$ | $\alpha(c'_1)$ | $\alpha(c'_0)$ |
| $-$ | $\alpha(k_{n-1})$ | $\alpha(k_{n-2})$ | $\dots$ | $\alpha(k_1)$ | $\alpha(k_0)$ |
| | $\alpha(c_{n-1})$ | $\alpha(c_{n-2})$ | $\dots$ | $\alpha(c_1)$ | $\alpha(c_0)$ |

Ovde nisu od značaja prenosi u drugoj fazi, pa će biti izostavljeni. Korekcija $\alpha(k_i)$ za $\alpha(c'_i)$ se određuje jedino na osnovu vrednosti pozajmice $p'_{i+1}$. Ako je $p'_{i+1} = 1$, tada je

$\alpha(k_i) = (0110)_2$, a ako je $p'_{i+1} = 0$, tada je $\alpha(k_i) = (0000)_2$. Slično kao i kod sabiranja, na početku se uvek postavlja $p'_0 = 0$. Budući da se uvek oduzimaju neoznačeni celi brojevi, takvi da je umanjilac manji od umanjenika, kod oduzimanja nikad ne dolazi do prekoračenja.

Na primer, neka je potrebno izvršiti oduzimanje brojeva 52629 i 2634, ako su oni zapisani u BCD kodu u zapisu 8421 na 5 mesta. Nakon što se umanjilac dopuni jednom vodećom nulom, oduzimanje se može prikazati na sledeći način:

```
  0101 0010 0110 0010 1001
- 0000 0010 0110 0011 0100
--------------------------
0    1    1    1    0    0
  0100 1111 1111 1111 0101
- 0000 0110 0110 0110 0000
--------------------------
  0100 1001 1001 1001 0101
```

Rešenje je 49995. Primetimo da prenosi u drugoj fazi nisu ni zapisivani, jer, za razliku od sabiranja u zapisu 8421, ovde nisu od značaja. Redosled izvođenja operacija je takođe pravolinijski – najpre se oduzmu dva broja u prvoj fazi, a zatim odrede prenosi. Nakon toga se na osnovu prenosa odrede vrednosti korekcija, i na kraju se u drugoj fazi može odjednom izvršiti oduzimanje međurezultata i broja dobijenog na osnovu korekcija.

Oduzimanje u zapisu 8421 se može svesti i na sabiranje, ali je potrebno da se oba broja zapišu u potpunom komplementu. Tada se umanjilac komplementira, nakon čega se izvrši sabiranje u kodu 8421 umanjenika i komplementiranog umanjioca. Na primer, neka je potrebno izvršiti oduzimanje brojeva 9463 i 6423, ako su oni zapisani na 5 mesta. Oduzimanje se može vršiti opisanim algoritmom (s tim što treba kod oba broja dodati vodeću nulu, jer je u zadatku naglašeno da je zapis na 5 mesta), a može se vršiti i svođenjem na potpuni komplement. Vrednosti brojeva u potpunom komplementu su 09463 i 06423. Komplementirani umanjilac je 93577. Nakon toga je potrebno primeniti algoritam za sabiranje brojeva 09463 i 93577 u zapisu 8421. Kada se četvorke bitova rezultata prevedu u dekadne cifre, on je u potpunom komplementu i treba ga nazad prevesti u dekadni broj (budući da će rezultat uvek biti pozitivan, ovo će se svoditi na brisanje vodeće nule za znak broja). Ukoliko se oduzimanje na ovaj način svodi na sabiranje, nikad ne može doći do prekoračenja. Ovo je posledica činjenice da su brojevi zapisani u potpunom komplementu, u kome se kod sabiranja poslednji prenosi uvek ignorišu. Primetimo još i da je potrebno da u zadatku bude naglašeno da su brojevi zapisani na bar jedno mesto više od njihovog broja cifara u dekadnom zapisu, da bi oduzimanje moglo da se izvrši. Inače, ne može se izvršiti svođenjem na potpuni komplement, jer nema mesta za zapis cifre za znak.

**Sabiranje u zapisu višak 3.** Neka su uvedene iste oznake kao kod sabiranja u zapisu 8421. Sabiranje u višku 3 se radi na sličan način kao sabiranje u zapisu 8421:

| | | | | | |
|---|---|---|---|---|---|
| | $\alpha(a_{n-1})$ | $\alpha(a_{n-2})$ | $\ldots$ | $\alpha(a_1)$ | $\alpha(a_0)$ |
| $+$ | $\alpha(b_{n-1})$ | $\alpha(b_{n-2})$ | $\ldots$ | $\alpha(b_1)$ | $\alpha(b_0)$ |
| $p'_n$ | $p'_{n-1}$ | $p'_{n-2}$ | $\ldots$ | $p'_1$ | $p'_0$ |
| | $\alpha(c'_{n-1})$ | $\alpha(c'_{n-2})$ | $\ldots$ | $\alpha(c'_1)$ | $\alpha(c'_0)$ |
| $+$ | $\alpha(k_{n-1})$ | $\alpha(k_{n-2})$ | $\ldots$ | $\alpha(k_1)$ | $\alpha(k_0)$ |
| | $\alpha(c_{n-1})$ | $\alpha(c_{n-2})$ | $\ldots$ | $\alpha(c_1)$ | $\alpha(c_0)$ |

Prva faza je identična kao kod sabiranja u 8421, a druga faza je karakteristična za ovaj zapis. Korekcija $\alpha(k_i)$ za deo međurezultata $\alpha(c'_i)$ se određuje na osnovu vrednosti prenosa $p'_{i+1}$. Ako je $p'_{i+1} = 1$, tada je $\alpha(k_i) = (0011)_2$, a ako je $p'_{i+1} = 0$, tada je $\alpha(k_i) = (1101)_2$. Na početku se uvek postavlja $p'_0 = 0$. U drugoj fazi, kada se vrši sabiranje jedne četvorke međurezultata sa korekcijom, ignoriše se prenos na narednu četvorku (kao kada sabiramo dva binarna broja u potpunom komplementu, zapisana na 4 mesta). Zbog toga prenosi u drugoj fazi i ne postoje. Prekoračenje kod sabiranja u višku 3 se javlja ako je $p'_n = 1$.

Na primer, neka je potrebno izvršiti sabiranje u višku 3 brojeva 28367 i 2847, ako su oni zapisani na 5 mesta. Nakon određivanja njihovih kodova u višku 3, algoritam izgleda ovako:

```
  0101 1011 0110 1001 1010
+ 0011 0101 1011 0111 1010
--------------------------
0    1    1    1    1    0
  1001 0001 0010 0001 0100
+ 1101 0011 0011 0011 0011
--------------------------
  0110 0100 0101 0100 0111
```

Rešenje je 31214. Do prekoračenja nije došlo, budući da je poslednji prenos jednak 0.

**Oduzimanje u zapisu višak 3.** Oduzimanje u zapisu višak 3 se svodi na sabiranje. Pritom je potrebno da se oba broja zapišu u potpunom komplementu. Tada se umanjilac komplementira, nakon čega se izvrši sabiranje u višku 3 umanjenika i komplementiranog umanjioca. Na primer, neka je potrebno izvršiti oduzimanje brojeva 9463 i 6423, ako su oni zapisani na 5 mesta. Vrednosti brojeva u potpunom komplementu su 09463 i 06423. Komplementirani umanjilac je 93577. Nakon toga je potrebno primeniti algoritam za sabiranje brojeva 09463 i 93577 u zapisu višak 3. Kada se četvorke bitova rezultata prevedu u dekadne cifre, on je u potpunom komplementu i treba ga nazad prevesti u dekadni broj (budući da će rezultat uvek biti pozitivan, ovo će se svoditi na brisanje vodeće nule za znak broja). Pritom u ovakvom sabiranju, na koje se svodi oduzimanje, nikad ne može doći do prekoračenja. Ovo je posledica činjenice da su brojevi zapisani u potpunom komplementu, u kome se kod sabiranja poslednji prenosi uvek ignorišu. Primetimo još i da je potrebno da u zadatku bude naglašeno da su brojevi zapisani na bar jedno mesto više od njihovog broja cifara u dekadnom zapisu. Inače, oduzimanje ne bi moglo da se izvrši, jer nema mesta za zapis cifre za znak.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 5 Zapis realnih brojeva

## 5.1 Zapis u nepokretnom i pokretnom zarezu

Za zapis realnih brojeva u računaru, bez obzira koliko memorije je odvojeno za zapis broja, uvek može da se zapiše konačno mnogo brojeva. Zbog toga razlomljeni deo broja, ukoliko postoji, uvek mora biti zapisan sa konačnim brojem cifara. Svaki zapis određuje opseg brojeva koji se može zapisati, kao i broj dozvoljenih cifara u razlomljenom delu. Realni brojevi se mogu zapisati u nepokretnom (fiksnom) i pokretnom zarezu.

Kod zapisa u nepokretnom zarezu je unapred određen broj cifara za zapis broja i broj cifara za zapis razlomljenog dela. Nekad sam programer mora voditi računa o mestu zapisa decimalne tačke, zbog čega u praksi može doći do greške. Na primer, neka je dozvoljena dužina zapisa za razlomljeni deo 11 i neka je potrebno zapisati heksadekadni broj $(0.919)_{16}$. Za zapis razlomljenog dela ovog broja je potrebno 12 cifara, a na ovaj način greškom poslednji bit zapisa može biti odbačen. Tada se greškom može dobiti heksadekadni broj $(0.48C)_{16}$, što je ilustrovano sledećom šemom:

```
Ispravan zapis:            9       1       9
                      0|1 0 0 1|0 0 0 1|1 0 0 1
-----------------------------------------------
Zapis u memoriji:     0 1 0 0 1 0 0 0 1 1 0 0 1
-----------------------------------------------
                      0 1 0 0|1 0 0 0|1 1 0 0|1
Pogrešan zapis:          4       8       C
```

Pri izvođenju aritmetičkih operacija može doći do greške, ukoliko se ne vodi računa o broju cifara u razlomljenom delu. Na primer, pri sabiranju dekadnih brojeva $(0.63)_{10}$ i $(0.2)_{10}$, umesto da se brojevi ispravno potpišu pri sabiranju i dobije korektan rezultat $(0.83)_{10}$, može se zanemariti dodavanje nule kod drugog sabirka i time dobiti pogrešan zbir:

```
  0 6 3
+ 0 0 2
-------
  0 6 5
```

Kod brojeva u pokretnom zarezu, ove greške se ne mogu pojaviti, jer se uvek automatski vodi računa o mestu decimalne tačke. Zapis u pokretnom zarezu je znatno

praktičniji od zapisa u nepokretnom, zbog čega se i danas najčešće koristi. Svaki realan broj se time može napisati u obliku $\pm f \cdot n^e$, gde $\pm$ označava da se mogu zapisati i pozitivni i negativni brojevi, $f$ frakciju (značajni deo broja), $n$ osnovu, a $e$ eksponent. Zapis kod koga je $f = f_0.f_1f_2 \dots f_{p-1}$ i $f_0 \neq 0$ se naziva normalizovan. Broj cifara frakcije (ukupan broj cifara u celobrojnom i razlomljenom delu) se naziva preciznost, u oznaci $p$.

Na primer, ako je broj $(0.4)_{10}$ potrebno zapisati u osnovi $n = 10$ i sa preciznošću $p = 4$, zapis je $4.000 \cdot 10^{-1}$, pa je normalizovana frakcija $f = 4.000$, a eksponent $e = -1$. Ukoliko se ne zahteva da frakcija bude normalizovana, isti broj se može napisati na više načina, na primer, u obliku $0.040 \cdot 10^1$ ($f = 0.040$ i $e = 1$) ili $0.004 \cdot 10^2$ ($f = 0.004$ i $e = 2$). Ako je broj $(0.4)_{10}$ potrebno zapisati u osnovi $n = 2$ i sa preciznošću $p = 6$ u normalizovanom obliku, najpre je potrebno izvršiti prevođenje $(0.4)_{10} = (0.011001100110...)_2$. Da bi broj mogao biti zapisan sa 6 značajnih cifara, mora se izvršiti odsecanje ili zaokruživanje razlomljenog dela (ovde će biti izvršeno odsecanje). Tako je traženi zapis $(1.10011)_2 \cdot 2^{-2}$. Primetimo da je frakcija uvek zapisana u osnovi $n$, a eksponent u dekadnoj osnovi. Osnova $n$ se uvek podrazumeva, tako da se pri kodiranju njen zapis izostavlja, o čemu će više biti reči kada se bude govorilo o konkretnim načinima zapisa.

Za razliku od brojeva zapisanih u nepokretnom zarezu, ovde se mogu zapisati i veoma veliki brojevi, ukoliko nemaju veliki broj značajnih cifara. Na primer, broj $53 \cdot 10^{100}$, ako je $n = 10$ i $p = 5$, može se zapisati u obliku $5.3000 \cdot 10^{101}$. Realni brojevi u pokretnom zarezu se u računaru najčešće zapisuju u jednostrukoj (32 bita), dvostrukoj (64 bita) i četvorostrukoj tačnosti (128 bitova).

## 5.2 Zapis sa binarnom osnovom na starijim računarima

Ilustracije radi, ovde će biti prikazano kako su se realni brojevi zapisivali na nekim od starijih verzija računara. Brojevi su bili zapisivani u binarnoj osnovi i podržavali su jednostruku i dvostruku tačnost. Frakcija proizvoljnog dekadnog broja se u ovom konktekstu piše u obliku $0.f_0f_1 \dots f_{p-1}$, pri čemu mora važiti $f_0 \neq 0$. Kako se ovde radi o binarnom sistemu, mora biti $f_0 = 1$, pa se ova cifra, budući da je podrazumevano uvek jednaka 1, izostavlja. Eksponent se piše na 8 mesta u zapisu višak 128. To znači da se eksponent, nakon dodavanja 128 u dekadnom sistemu, pretvara u binarni sistem i zapisuje na 8 mesta (sa eventualnim vodećim nulama, ukoliko je potrebno). Osnova 2 se podrazumeva i ona se ne kodira. Takođe, znak broja se kodira pomoću jednog bita. Ako je broj pozitivan, odgovarajući bit je 0, a ako je negativan, odgovarajući bit je 1.

U zapisu u jednostrukoj tačnosti, na raspolaganju su 32 bita. Sleva nadesno, oni su podeljeni u tri grupe. Prvi bit označava znak, narednih 8 eksponent, a preostala 23 bita frakciju. Kod dvostruke tačnosti se znak i eksponent zapisuju na isti način, s tim što su dodata još 32 bita za frakciju. Ovim se povećava preciznost. Specijalno, 0 se piše sa svim nulama u zapisu, bilo da se radi o jednostrukoj ili dvostrukoj tačnosti.

Primetimo da se na ovaj način može zapisati samo konačan interval brojeva. Ukoliko je broj veliki po apsolutnoj vrednosti tako da ne može da se zapiše, dolazi do prekoračenja. Nasuprot tome, broj može biti po apsolutnoj vrednosti veoma blizak nuli i nedovoljno veliki za zapis, kada dolazi do potkoračenja.

U nastavku će biti navedeno nekoliko primera kodiranja i dekodiranja brojeva u ovom zapisu:

- Broj 0 se piše kao sve nule u zapisu. Njegov zapis u jednostrukoj tačnosti je 0 00000000 $\underbrace{0 \dots 0}_{23 \text{ nule}}$.

- Za zapis dekadnog broja 15 u jednostrukoj tačnosti, najpre je potrebno izvršiti prevođenje $(15)_{10} = (1111)_2$. Pogodan oblik broja za zapis mora biti takav da se frakcija sastoji samo od razlomljenih cifara, pri čemu je prva cifra jednaka 1. Pogodan zapis je $(0.1111)_2 \cdot 2^4$. Eksponent 4 je potrebno napisati sa uvećanjem 128. Kako je $4 + 128 = 132$, zapis eksponenta je $(132)_{10} = (10000100)_2$. Kao što je već navedeno, kod frakcije 0.1111 se deo 0.1 izostavlja, pa je njen zapis 111, sa odgovarajućim brojem nula u nastavku frakcije. Broj je pozitivan, pa je bit za znak jednak 0. Konačno, zapis je 0 10000100 $111\underbrace{0\ldots0}_{20 \text{ nula}}$.

- Za zapis dekadnog broja $-15$ u dvostrukoj tačnosti se dobija $(-15)_{10} = (-1111)_2 = (-0.1111)_2 \cdot 2^4$. Kod eksponenta je $(10000100)_2$, kao u prethodnom primeru. Bit za znak je 1, jer se radi o negativnom broju, a frakcija se sastoji od 111, nakon čega treba nadovezati odgovarajući broj nula. Zapis broja je 1 10000100 $111\underbrace{0\ldots0}_{52 \text{ nule}}$.

- Neka je dat zapis 0 01111011 $\underbrace{0\ldots0}_{23 \text{ nule}}$, koji je potrebno prevesti u dekadni sistem. Vidimo da je vrednost eksponenta $(1111011)_2 - 128 = 123 - 128 = -5$. Kako se frakcija sastoji samo od nula, njena binarna vrednost je 0.1. Broj je pozitivan, pa je njegova dekadna vrednost $(0.1)_2 \cdot 2^{-5} = 2^{-1} \cdot 2^{-5} = 2^{-6} = 1/64$.

## 5.3 IEEE 754 standard

U početku nije postojao jedinstven standard za zapis realnih brojeva, pa je bilo moguće da različite arhitekture implementiraju zapis na različite načine, zbog čega može doći do grešaka i nepodudaranja pri prenosu podataka. Zbog potrebe sa unifikacijom načina zapisa realnih brojeva, nastao je IEEE 754 standard. Njime je precizno propisan način zapisa brojeva, kao i algoritmi za različite operacije nad njima, ponašanje u graničnim situacijama, i slično. Kako danas sve mašine podržavaju ovaj standard, smanjena je verovatnoća greške pri prenosu podataka. IEEE 754 standard podržava zapis brojeva sa dekadnom i binarnom osnovom, a mogući su zapisi sa jednostrukom, dvostrukom i četvorostrukom tačnošću. Oni redom zauzimaju 32, 64 i 128 bitova.

**Zaokruživanje.** U računaru uvek može da se zapiše konačno mnogo realnih brojeva, zbog čega se neki brojevi ne mogu precizno zapisati, već se pamti njihova približna vrednost, pri čemu se vrši zaokruživanje. Postoji nekoliko načina na koji se može izvršiti zaokruživanje, među kojima su:

- Zaokruživanje na parnu cifru – broj se zaokružuje na najbližu pretpostavljenu vrednost, a ako je na sredini intervala, onda se zaokruživanje vrši na parnu cifru. Tako se, na primer, brojevi 1.212, 1.218, 1.225 i 1.235 zaokružuju na dve decimale redom na vrednosti 1.21, 1.22, 1.22 i 1.24.

- Zaokruživanje ka $+\infty$ – broj se zaokružuje tako da mu je zaokružena vrednost uvek veća ili jednaka od stvarne. Tako se, na primer, pozitivni brojevi 1.21 i 1.29 na jednu decimalu zaokružuju redom na 1.3 i 1.3, a negativni brojevi $-1.21$ i $-1.29$ redom na $-1.2$ i $-1.2$.

- Zaokruživanje ka $-\infty$ – broj se zaokružuje tako da mu je zaokružena vrednost uvek manja ili jednaka od stvarne. Tako se, na primer, pozitivni brojevi 1.21 i 1.29 na

jednu decimalu zaokružuju redom na 1.2 i 1.2, a negativni brojevi $-1.21$ i $-1.29$ redom na $-1.3$ i $-1.3$.

- Zaokruživanje ka nuli – broj se zaokružuje tako da mu je zaokružena vrednost uvek po apsolutnoj vrednosti manja ili jednaka od stvarne. Tako se, na primer, pozitivni brojevi 1.21 i 1.29 na jednu decimalu zaokružuju redom na 1.2 i 1.2, a negativni brojevi $-1.21$ i $-1.29$ redom na $-1.2$ i $-1.2$.

Neka 0.0574367 predstavlja matematički tačnu vrednost, a neka približan zapis broja nakon zaokruživanja za preciznost $p = 3$ iznosi $5.74 \cdot 10^{-2}$. Možemo primetiti da je razlika stvarne i zaokružene vrednosti 0.0000367, a ako to posmatramo u jedinicama koje su u odnosu na poslednje mesto zaokruženog zapisa, greška iznosi 0.367. Veličina koja predstavlja broj jedinica na poslednjem mestu se naziva ULP. Drugi način merenja greške je relativna greška. Ona predstavlja apsolutnu vrednost razlike stvarne i zakoružene vrednosti broja, podeljenu sa apsolutnom vrednošću broja. U našem primeru ona iznosi $|0.0574367 - 0.0574|/|0.0574367| \approx 0.0006$. Osnovna razlika između ULP-a i relativne greške se ogleda u tome šta oni mere. Dok se pomoću ULP meri stvarna greška, pomoću relativne greške se meri odgovarajući odnos, zbog čega se ona najčešće koristi kod analize greške koja se javlja pri različitim izračunavanjima. Time, ako se broj poveća nekoliko puta, toliko puta će se povećati i ULP, dok relativna greška ostaje ista.

**Cifre čuvari.** Cifre čuvari predstavljaju dodavanje još jedne cifre na zapis broja pri primeni računskih operacija, gde se povećava preciznost za 1 i time smanjuje greška. Neka je potrebno oduzeti brojeve 10.1 i 9.93, ako su zapisani u obliku $1.01 \cdot 10^1$ i $0.99 \cdot 10^1$. Njihovim oduzimanjem se dobija rezultat $0.02 \cdot 10^1$, a stvarna vrednost bi trebalo da bude 0.17. Preciznija razlika se može dobiti uvođenjem dodatne cifre čuvara. Tada se oduzimanjem zapisa $1.010 \cdot 10^1$ i $0.993 \cdot 10^1$ dobija vrednost $0.017 \cdot 10^1$.

**Tačno zaokružene operacije.** Kada se koriste cifre čuvari, u računaru se preciznost uvećava za 1, pa dobijeni rezultat, iako precizniji od slučaja kada se one ne koriste, nije naročito precizniji. U idealnom slučaju, ukoliko to resursi dozvoljavaju, operacije se mogu izvršiti nad stvarnim vrednostima brojeva, a kasnije zaokružiti rezultat. Za takve operacije kažemo da su tačno zaokružene.

## 5.4 IEEE 754 standard sa dekadnom osnovom

Kod IEEE 754 standarda sa dekadnom osnovom, osnova je uvek 10, a frakcija može biti dekadna ili binarna. Ako je frakcija dekadna, radi se o DPD kodiranju, a ako je binarna, o BID kodiranju. Za obe vrste kodiranja će biti opisan zapis u jednostrukoj tačnosti, kada se brojevi kodiraju sa 32 bita.

**DPD kodiranje.** Ukoliko su i frakcija i osnova dekadne, IEEE 754 standard koristi DPD kodiranje. Prethodno smo se upoznali kako se mogu iskoristiti tabele za DPD kodiranje i dekodiranje, pri čemu se parovi od 3 dekadne cifre mogu napisati sa 10 binarnih cifara. Podsetimo se kako one izgledaju (iznad je navedena tabela za kodiranje, a ispod tabela za dekodiranje):

| aei | pqr stu v wxy |
|---|---|
| 000 | bcd fgh 0 jkl |
| 001 | bcd fgh 1 00l |
| 010 | bcd jkh 1 01l |
| 100 | jkd fgh 1 10l |
| 110 | jkd 00h 1 11l |
| 101 | fgd 01h 1 11l |
| 011 | bcd 10h 1 11l |
| 111 | 00d 11h 1 11l |

| vwxst | abcd efgh ijkl |
|---|---|
| 0.... | 0pqr 0stu 0wxy |
| 100.. | 0pqr 0stu 100y |
| 101.. | 0pqr 100u 0sty |
| 110.. | 100r 0stu 0pqy |
| 11100 | 100r 100u 0pqy |
| 11101 | 100r 0pqu 100y |
| 11110 | 0pqr 100u 100y |
| 11111 | 100r 100u 100y |

U nastavku ćemo kroz primere opisati kako se vrši kodiranje iz dekadne vrednosti u zapis i dekodiranje iz zapisa u dekadnu vrednost. Neka je, na primer, potrebno kodirati broj 123.4 po IEEE 754 standardu u dekadnoj osnovi u jednostrukoj tačnosti koristeći DPD kodiranje. Broj najpre treba napisati kao proizvod celobrojne frakcije i stepenovane osnove. Ovde je $123.4 = 1234 \cdot 10^{-1}$. Celobrojnu frakciju je uvek potrebno zapisati na 7 mesta. Ako ima manje od 7 značajnih cifara, dopunjuje se nulama, a ako ima više, vrši se zaokruživanje. Ovde je prilagođena frakcija oblika 0001234. Ona se sada deli na tri dela, gde prvi deo sadrži jednu cifru, a ostala dva dela po 3 cifre. U našem primeru frakcija se deli na 0, 001 i 234. Poslednja dva dela frakcija se kodiraju sa po 10 bitova koristeći tabelu za kodiranje. Kodirajmo najpre trojku 001. Zapisi 8421 dekadnih cifara 0, 0 i 1 su redom 0000, 0000 i 0001. Ako označimo da je $abcd = 0000$, $efgh = 0000$ i $ijkl = 0001$, budući da je $aei = 000$, iz tabele za kodiranje sledi da je $pqr = bcd$, $stu = fgh$, $v = 0$ i $wxy = jkl$, pa je traženi kod $pqrstuvwxy = 0000000001$. Na sličan način se dobija da se dekadna trojka 234 kodira sa 0100110100. Preostala je još prva cifra frakcije. Ukoliko je ona između 0 i 7, kodira se sa tri binarne cifre, tako što se cifra pretvori u binarni zapis na tri mesta, uz eventualno dodavanje vodećih nula (na primer, 0 se kodira kao 000, 5 kao 101 i 7 kao 111). Ukoliko je cifra frakcije 8 ili 9, ona se kodira sa jednom binarnom cifrom. Osmica se kodira sa 0, a devetka sa 1. U našem primeru je prva cifra frakcije 0, pa se ona kodira kao 000. Ostalo je još da se kodira eksponent. Eksponent $-1$ je potrebno zapisati u binarnom sistemu na 8 mesta sa uvećanjem 101. Dakle, zapis eksponenta je oblika $-1 + 101 = 100 = (01100100)_2$. Na osnovu koda prve cifre frakcije (3 bita) i eksponenta (8 bitova), formira se kombinacija koja se sastoji od 11 bitova. (Ako je prva cifra 8 ili 9, ona se kodira samo sa jednom binarnom cifrom, pa se kombinacija od 11 bitova dobija na nešto drugačiji način, što će biti objašnjeno u narednom primeru.) Kombinacija se dobija tako što se eksponent razdvoji na dva dela, levi koji sadrži prva dva i desni koji sadrži poslednjih šest bitova, a između njih se ubaci trobitni kod početka frakcije. U našem slučaju, kombinacija je 01000100100. Broj je pozitivan, pa je bit za znak 0. Konačno, prvi bit zapisa je za znak, narednih 11 bitova čini kombinacija, a preostalih 20 ostatak

frakcije. Zapis traženog broja je 0 01000100100 0000000001 0100110100.

Pretpostavimo da je potrebno kodirati na isti način broj $-982.5294 \cdot 10^{42}$. Pogodan zapis broja za kodiranje je $-9825294 \cdot 10^{38}$. Kod cifre 9 je 1, a desetobitni kodovi trojki 825 i 294 su redom 1000101101 i 0101011010. Ovi kodovi su dobijeni koristeći tabelu za kodiranje, slično kao u prethodnom primeru. Zapis eksponenta je $38 + 101 = 139 = (10001011)_2$. Kombinacija od 11 bitova se i ovde dobija od ekponenta i koda prve cifre frakcije, ali je ovde problem sto zapis početka frakcije ima samo 1 bit. U slučaju kada je prva cifra frakcije 8 ili 9 (kada se ona ustvari kodira jednim bitom), na početku kombinacije se dodaje 11, a ostatak se dobija analogno, razdvajanjem eksponenta i ubacivanjem cifre frakcije. Dakle, u ovom slučaju kombinacija se sastoji od 11, početka eksponenta 10, početka frakcije 1 i nastavka eksponenta 001011, pa je ona oblika 11101001011. Uzimajući u obzir i da je broj negativan, njegov zapis je 1 11011001011 1000101101 0101011010.

Objasnimo sada obratni proces, kada se vrši prevođenje iz zapisa u dekadni broj na primeru 32-bitnog zapisa 0 01111011110 0000000000 0000110101. Najpre treba posmatrati kombinaciju, jer tu mogu postojati dva slučaja – kada je prva cifra frakcije između 0 i 7, odnosno kada je 8 ili 9. Ukoliko ona počinje sa 11, radi se o početnoj cifri 8 ili 9, a inače, početna cifra je između 0 i 7. U našem primeru prve dve cifre i poslednjih 6 cifara kombinacije su cifre eksponenta, a preostala trojka 111 kodira prvu cifru frakcije čija je vrednost 7. Eksponent iznosi $(01011110)_2 - 101 = 94 - 101 = -7$. Na osnovu tabele za dekodiranje (videti lekciju gde se prvi put pominje DPD), dobija se da kodovi 0000000000 i 000011010 redom predstavljaju trojke dekadnih cifara 000 i 035. Kako je broj pozitivan, konačno se dobija da je tražena vrednost $7000035 \cdot 10^{-7} = 0.7000035$.

Pored zapisa konačnih brojeva, IEEE 754 standard sa dekadnom osnovom (i kod DPD i kod BID kodiranja) podržava zapis specijalnih vrednosti. Kada se vrši dekodiranje, potrebno je prvo razmotriti da li se radi o nekoj specijalnoj vrednosti, pa ako se ustanovi da je u pitanju konačan broj, treba pribeći samom algoritmu. Specijalne vrednosti koje imaju poseban zapis u ovom standardu su:

- Nula se može pojaviti kao konačna vrednost ili kao neka veoma mala vrednost koja se ne može zapisati, kada dolazi do potkoračenja, gde se ta vrednost zapisuje da je približno jednaka nuli. Može postojati pozitivna ili negativna nula, pa joj bit za znak može biti 0 ili 1. Na svim bitovima kojim je označena frakcija je potrebno da stoje nule, a na bitovima gde je eksponent može biti proizvoljna vrednost. To znači da su treći, četvrti i peti bit kombinacije, kao i poslednjih 20 bitova zapisa jednaki 0.

- Beskonačno se može pojaviti kada se neki veoma veliki broj ne može zapisati ili kao rezultat operacija poput $5 - \infty = -\infty$, $\infty + \infty = \infty$, $-5/0 = -\infty$, i slično. U zavisnosti od toga da li se radi o pozitivnoj ili negativnoj beskonačnosti, bit za znak može biti 0 ili 1. Za beskonačno važi da je početak kombinacije oblika 11110.

- NaN vrednosti se javljaju u izuzetnim situacijama. Postoji signalni NaN (sNaN) i tihi NaN (qNaN). Signalni NaN se javlja pri greškama u inicijalizaciji ili konverziji, a tihi NaN pri greškama u aritmetičkim operacijama. Primeri za pojavu qNaN vrednosti su nedefinisane vrednosti poput $\infty - \infty$, $0 \cdot \infty$, $0/0$, itd. Ukoliko je argument aritmetičke operacije sNaN ili qNaN, rezultat je qNaN. Kod sNaN vrednosti

kombinacija počinje sa 111111, a kod qNaN vrednosti sa 111110. Ostale cifre zapisa mogu biti proizvoljne.

**BID kodiranje.** Ukoliko je osnova dekadna, a frakcija binarna, IEEE 754 standard koristi BID kodiranje. U nastavku će biti navedeni primeri kodiranja i dekodiranja u jednostrukoj tačnosti. Specijalne vrednosti se zapisuju na isti način kao u slučaju DPD kodiranja.

Neka je, na primer, potrebno odrediti zapis broja $-14.37$. Pogodan zapis broja za kodiranje je

$$-14.37 = -1437 \cdot 10^{-2} = (-10110011101)_2 \cdot 10^{-2}.$$

Dakle, broj treba zapisati tako da mu je frakcija celobrojna, a zatim tu frakciju prevesti u binarni sistem. Osnova ostaje sve vreme dekadna. Ukoliko frakcija sadrži manje od 23 bita, potrebno je dopisati vodeće nule, kako bi bila dopunjena do tog broja bitova. U ovom slučaju, zapis frakcije bi bio 00000000000010110011101. Eksponent se i ovde piše na 8 mesta sa uvećanjem 101 i iznosi $-2+101 = 99 = (01100011)_2$. Zapis broja se sastoji iz tri dela, jednobitnog znaka, kombinacije od 11 bitova koja se formira nadovezivanjem cifara eksponenta i prve tri cifre frakcije i nastavka frakcije od 20 bitova. Kako je u pitanju negativan broj, zapis je 1 01100011000 00000000010110011101.

Neka je potrebno izvršiti prevođenje broja 9000001. Pogodan zapis za kodiranje je

$$9000001 \cdot 10^0 = (10001001010101000100000 1)_2 \cdot 10^0.$$

Kada frakcija ima 24 bita, što je ovde slučaj, njena prva tri bita se ingorišu (podrazumevano su uvek jednaki 100). Kombinacija se dobija dodavanjem para bitova 11 na početak, zatim 8 cifara eksponenta i četvrtog bita frakcije. Ostalih 20 bitova se dodaju u nastavku frakcije. Eksponent je $0 + 101 = 101 = (01100101)_2$, pa je kombinacija 11011001010. Broj je pozitivan, pa je njegov zapis 0 11011001010 10010101010001000001.

Izvršimo sada dekodiranje na primeru zapisa 1 01100110000 00000000000000001011. Nakon što se utvrdi da se ne radi ni o kakvoj specijalnoj vrednosti, treba proveriti da li kombinacija počinje sa 11. Ako počinje, razmatra se slučaj dekodiranja gde frakcija ima 24 cifre, a inače slučaj kada frakcija ima 23 cifre. Ovde prvih 8 bitova kombinacije predstavlja eksponent, a poslednja tri početak frakcije. Eksponent je $(01100110)_2 - 101 = 102 - 101 = 1$, a frakcija $(-1011)_2 = -11$. U pitanju je broj $-11 \cdot 10^1 = -110$.

## 5.5 IEEE 754 standard sa binarnom osnovom

Kod IEEE 754 standarda sa binarnom osnovom, frakcija je binarna, a osnova 2. Zapis se sastoji iz tri dela – zapisa znaka, eksponenta i frakcije. Ovde će biti razmotrena jednostruka i dvostruka tačnost. Kod jednostruke tačnosti broj se kodira sa 32 bita, a kod dvostruke sa 64 bita. IEEE 754 standard sa binarnom osnovom propisuje zapis normalnih i subnormalnih brojeva, kao i specijalnih vrednosti. I u jednostrukoj i u dvostrukoj tačnosti, podržane specijalne vrednosti su 0, beskonačno i NaN vrednost. Nula se piše sa svim nulama u eksponentu i frakciji, a bit za znak može biti 0 ili 1, u zavisnosti od znaka nule. Beskonačno se piše sa svim jedinicama u eksponentu i svim nulama u frakciji. Znak za beskonačno takođe može biti 0 ili 1. Obe NaN vrednosti se pišu sa svim jedinicama u eksponentu. Kod sNaN vrednosti, prvi bit frakcije je 0, a kod ostatka frakcije nisu svi

bitovi jednaki nuli, jer bi se tada poklapao zapis sa $\infty$, a ostale kombinacije su dozvoljene. Kod qNaN vrednosti, prvi bit frakcije je 1, a ostali bitovi mogu biti proizvoljni.

**Jednostruka tačnost.** U jednostrukoj tačnosti u zapisu se znak kodira pomoću jednog bita, eksponent pomoću 8, a frakcija pomoću 23. Prikažimo algoritam prevođenja na primeru broja 13.25. Pogodan zapis za prevod je $13.25 = (1101.01)_2 = (1.10101)_2 \cdot 2^3$. Dekadni broj je najpre potrebno prevesti u binarni, nakon čega frakciju treba dovesti na oblik $1.f$ ($f$ je ostatak frakcije iza decimalne tačke) i ažurirati po potrebi eksponent. Ekponent se zapisuje na 8 mesta sa uvećanjem 127 i iznosi $3 + 127 = 130 = (10000010)_2$. Pri zapisu frakcije, ceo deo (prva jedinica) se ignoriše, budući da je uvek ista i nema potrebe je pamtiti. Kako je broj pozitivan, njegov zapis je 0 10000010 10101000000000000000000. Primetimo da je ostatak frakcije dopunjen nulama.

Obratno, pretpostavimo da je dat zapis 1 10000010 10101000000000000000000 i neka je potrebno odrediti njegovu dekadnu vrednost. Za vrednost eksponenta broja se dobija $(10000010)_2 - 127 = 130 - 127 = 3$. Dodavanjem podrazumevane jedinice, i imajući u vidu da je broj negativan, sledi da je frakcija $(-1.10101)_2$. Dekadna vrednost broja je $(-1.10101)_2 \cdot 2^3 = (-1101.01)_2 = -13.25$.

IEEE 754 standard sa binarnom osnovom podržava i zapis subnormalnih brojeva. Subnormalni brojevi su oni brojevi koji su veoma bliski nuli. Prepoznaju se tako što imaju sve nule u eksponentu. Kada se iščitava, njihova frakcija je, za razliku od normalnih brojeva, oblika $0.f$. Za jednostruku tačnost, eksponent iznosi $-126$. Na primer, na osnovu zapisa 0 00000000 00010000000000000000000, zaključujemo da se radi o subnormalnom broju. Na osnovu početka zapisa frakcije 0001 (ostatak su nule, pa nisu od značaja), zaključujemo da je frakcija oblika 0.0001. Broj je pozitivan, pa je vrednost datog denormalizovanog broja $(0.0001)_2 \cdot 2^{-126} = 2^{-4} \cdot 2^{-126} = 2^{-130}$.

**Dvostruka tačnost.** U dvostrukoj tačnosti u zapisu se znak kodira pomoću jednog bita, eksponent pomoću 11, a frakcija pomoću 52. Prikažimo algoritam prevođenja na primeru broja 13.25. Pogodan zapis za prevod je $13.25 = (1101.01)_2 = (1.10101)_2 \cdot 2^3$. Dekadni broj je najpre potrebno prevesti u binarni, nakon čega frakciju treba dovesti na oblik $1.f$ ($f$ je ostatak frakcije iza decimalne tačke) i ažurirati po potrebi eksponent. Ekponent se zapisuje na 11 mesta sa uvećanjem 1023 i iznosi $3 + 1023 = 1026 = (10000000010)_2$. Pri zapisu frakcije, ceo deo (prva jedinica) se ignoriše, budući da je uvek ista i nema potrebe je pamtiti. Kako je broj pozitivan, njegov zapis je

0 10000000010 1010100000000000000000000000000000000000000000000000.

Obratno, pretpostavimo da je dat zapis

1 10000000010 1010100000000000000000000000000000000000000000000000

i neka je potrebno odrediti njegovu dekadnu vrednost. Vrednost eksponenta broja iznosi $(10000000010)_2 - 1023 = 1026 - 1023 = 3$. Dodavanjem podrazumevane jedinice, i imajući u vidu da je broj negativan, sledi da je frakcija $(-1.10101)_2$. Dekadna vrednost broja je $(-1.10101)_2 \cdot 2^3 = (-1101.01)_2 = -13.25$.

I u dvostrukoj tačnosti je podržan zapis subnormalnih brojeva. Za dvostruku tačnost, eksponent iznosi $-1022$. Na primer, na osnovu zapisa

0 00000000000 0001000000000000000000000000000000000000000000000000,

zaključujemo da se radi o denormalizovanom broju. Na osnovu početka zapisa frakcije 0001 (ostatak su nule, pa nisu od značaja), zaključujemo da je frakcija oblika 0.0001. Broj je pozitivan, pa je vrednost datog denormalizovanog broja $(0.0001)_2 \cdot 2^{-1022} = 2^{-4} \cdot 2^{-1022} = 2^{-1026}$.

## 5.6 Aritmetičke operacije

U nastavku će biti opisane operacije sabiranja, oduzimanja, množenja i deljenja u IEEE 754 standardu na primeru zapisa sa binarnom osnovom u jednostrukoj tačnosti. Pre izvođenja bilo koje operacije, treba ustanoviti da li je rezultat eventualno neka specijalna vrednost. Ukoliko nije, operacija se može izvršiti odgovarajućih algoritmom. Na kraju izvršene operacije, eventualno može doći do prekoračenja ili potkoračenja, kada se rezultat redom proglašava za beskonačno ili nulu.

**Sabiranje.** Neka su data dva broja u IEEE 754 standardu u binarnoj osnovi u obliku $f_1 \cdot 2^{e_1}$ i $f_2 \cdot 2^{e_2}$, gde su $f_1$ i $f_2$ frakcije, a $e_1$ i $e_2$ eksponenti. Za sabiranje je brojeve potrebno dovesti na isti eksponent, a u ovom algoritmu se uzima onaj koji je veći. Pretpostavimo da je $e_1 \geq e_2$ i da je nakon svođenja na eksponent $e_1$ drugi sabirak oblika $f_2' \cdot 2^{e_1}$. Sada se sabiranje svodi na

$$f_1 \cdot 2^{e_1} + f_2 \cdot 2^{e_2} = f_1 \cdot 2^{e_1} + f_2' \cdot 2^{e_1} = (f_1 + f_2') \cdot 2^{e_1}.$$

Na primer, neka je potrebno izvršiti sabiranje 0 10000010 00110000000000000000000 i 0 10000001 01100000000000000000000. Bit za znak rezultata je 0, budući da se vrši sabiranje dva pozitivna broja. Eksponent rezultata je jednak većem od eksponenata sabiraka i iznosi 10000010. Sada je potrebno sabrati odgovarajuće frakcije. Kako je eksponent rezultata jednak eksponentu prvog sabirka, frakcija prvog sabirka ostaje nepromenjena i iznosi 1.10011. Eksponent drugog sabirka treba povećati za 1, pa je potrebno izvršiti modifikaciju njegove frakcije 1.1011 pomeranjem decimalne tačke za jedno mesto ulevo. Dakle, njegova frakcija je oblika 0.11011, pa je frakcija zbira jednaka 1.10011+0.11011 = 10.0111. Kada se ne dobije normalizovana frakcija, potrebno je izvršiti njenu normalizaciju i u skladu sa tim promeniti vrednost eksponenta. Važi da je $(10.0111)_2 = (1.00111)_2 \cdot 2^1$, pa eksponent rezultata treba uvećati za 1. On sada iznosi 10000011. Zapis zbira je 0 10000011 00111000000000000000000.

**Oduzimanje.** Neka su data dva broja u obliku $f_1 \cdot 2^{e_1}$ i $f_2 \cdot 2^{e_2}$. Kod oduzimanja je potrebno eksponent umanjioca dovesti na eksponent umanjenika. Pretpostavimo da je nakon svođenja na eksponent $e_1$ umanjilac oblika $f_2' \cdot 2^{e_1}$. Sada se oduzimanje svodi na

$$f_1 \cdot 2^{e_1} - f_2 \cdot 2^{e_2} = f_1 \cdot 2^{e_1} - f_2' \cdot 2^{e_1} = (f_1 - f_2') \cdot 2^{e_1}.$$

Na primer, neka je potrebno izvršiti oduzimanje 0 10000001 01011000000000000000000 i 0 01111110 00100000000000000000000. Bit za znak razlike je 0, budući da se vrši oduzimanje dva pozitivna broja, gde se oduzima manji broj od većeg. Eksponent razlike je

jednak eksponentu umanjenika. Frakciju umanjioca treba pomeriti za odgovarajući broj mesta ulevo kako bi se eksponent izjednačio sa eksponentom umanjenika. U ovom slučaju, frakcija umanjioca 1.001 postaje 0.001001. Decimalnu tačku je potrebno pomeriti tri mesta ulevo zbog toga što se njegov eksponent povećava za 3, budući da je $(10000001)_2 - (01111110)_2 = 3$. Oduzimanjem frakcija se dobija $1.010110 - 0.001001 = 1.001101$. Frakcija razlike je normalizovana, pa je rezultat 0 10000001 00110100000000000000000.

**Množenje.** Neka su data dva broja u obliku $f_1 \cdot 2^{e_1}$ i $f_2 \cdot 2^{e_2}$. Njihovim množenjem se dobija

$$(f_1 \cdot 2^{e_1}) \cdot (f_2 \cdot 2^{e_2}) = (f_1 f_2) \cdot 2^{e_1+e_2}.$$

Iz poslednjeg sledi da je frakcija proizvoda jednaka proizvodu frakcija, a eksponent proizvoda zbiru eksponenata. Pri sabiranju eksponenata treba voditi računa da su oni u višku 127, koji nakon sabiranja treba anulirati.

Na primer, neka je potrebno izvršiti množenje 0 10000011 00100000000000000000000 i 0 10000010 00110000000000000000000. Kako se množe dva pozitivna broja, rezultat je takođe pozitivan. Kako su eksponenti oba činioca zapisani u višku 127, a eksponent proizvoda je potrebno takođe napisati u višku 127, pri sabiranju eksponenata činioca, da bi se dobio zapis eksponenta rezultata, potrebno je od dobijenog zbira oduzeti 127. Dakle, dobija se:

```
   10000011
+  10000010
-----------
  100000101
-  01111111
-----------
   10000110
```

Zatim se množenjem frakcija brojeva (postupak množenja je sličan množenju dva dekadna broja) dobija $1.001 \cdot 1.0011 = 1.0101011$. Dobijenu frakciju nije potrebno normalizovati, jer je već svedena na oblik $1.f$. Proizvod je 0 10000110 01010110000000000000000.

**Deljenje.** Neka su data dva broja u obliku $f_1 \cdot 2^{e_1}$ i $f_2 \cdot 2^{e_2}$. Njihovim deljenjem se dobija

$$(f_1 \cdot 2^{e_1})/(f_2 \cdot 2^{e_2}) = (f_1/f_2) \cdot 2^{e_1-e_2}.$$

Iz poslednjeg sledi da je frakcija količnika jednaka količniku frakcija, a eksponent količnika razlici eksponenata. Pri oduzimanju eksponenata treba voditi računa da su oni u višku 127, koji nakon oduzimanja treba dodati.

Na primer, neka je potrebno izvršiti deljenje 0 10000100 11101110000000000000000 i 0 10000001 10100000000000000000000. Kako se dele dva pozitivna broja, rezultat je takođe pozitivan broj. Kako su eksponenti i deljenika i delioca zapisani u višku 127, a eksponent količnika je potrebno takođe napisati u višku 127, pri oduzimanju eksponenata deljenika i delioca, da bi se dobio zapis eksponenta rezultata, potrebno je dobijenoj razlici dodati 127. Dakle, dobija se:

```
   10000100
-  10000001
-----------
   00000011
+  01111111
-----------
   10000010
```

Zatim se deljenjem frakcija brojeva (postupak deljenja je analogan deljenju dva dekadna broja) dobija $1.1110111/1.101 = 1.0011$. Dobijenu frakciju nije potrebno normalizovati, jer je već svedena na oblik $1.f$. Dobijeni količnik je

$$0\ 10000010\ 00110000000000000000000.$$

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

## 6 Zapis brojeva pomoću ostatka

Podsetimo se najpre oznaka vezanih za modularnu aritmetiku. Ako je $r = a \bmod m$, kažemo da je $r$ ostatak pri celobrojnom deljenju broja $a$ brojem $m$. Na primer, za $11 \bmod 3 = 2$ je $a = 11$, $m = 3$ i $r = 2$. Ako je $a \equiv b \pmod m$, tada brojevi $a$ i $b$ daju isti ostatak pri deljenju brojem $m$. Na primer, važi da je $11 \equiv 2 \pmod 3$ ili $11 \equiv 20 \pmod 3$, dok je $11 \not\equiv 4 \pmod 3$.

Neka je dat skup celih brojeva $m_1, m_2, \ldots, m_n$ koji su veći od 1, za koje je $m_1 < m_2 < \cdots < m_n$. Zapis RBS$(m_n|m_{n-1}|\ldots|m_2|m_1)$ označava brojčani sistem sa ostacima $m_n, m_{n-1}, \ldots, m_2, m_1$. Ceo dekadni broj $a$ se tako u navedenom sistemu zapisuje pomoću $n$ cifara sa

$$(a_n|a_{n-1}|\ldots|a_2|a_1)_{\text{RBS}(m_n|m_{n-1}|\ldots|m_2|m_1)},$$

pri čemu je $a_i = a \bmod m_i$ za $1 \leq i \leq n$. Jasno je da mora važiti da je $0 \leq a_i < m_i$.

Brojevi $m_i$ se mogu izabrati proizvoljno, a zapis može biti proizvoljne dužine. Na primer, za RBS dužine 4, pri čemu su moduli redom jednaki 8, 7, 5 i 3, dekadni broj 62 se zapisuje kao

$$(6|6|2|2)_{\text{RBS}(8|7|5|3)},$$

budući da je redom $62 \bmod 8 = 6$, $62 \bmod 7 = 6$, $62 \bmod 5 = 2$ i $62 \bmod 3 = 2$. Isti dekadni broj se može zapisati i u, na primer, zapisu RBS(9|8|7). Kako je $62 \bmod 9 = 8$, $62 \bmod 8 = 6$ i $62 \bmod 7 = 6$, to je

$$(62)_{10} = (8|6|6)_{\text{RBS}(9|8|7)}.$$

Da bi se izbegla višeznačnost zapisa, odnosno pojava da dva različita dekadna broja imaju isti RBS zapis, moduli brojeva moraju da budu uzajamno prosti. Drugim rečima, treba da važi $\text{NZD}(m_i, m_j) = 1$ za sve $i \neq j$. Neka je

$$m = \prod_{i=1}^{n} m_i = m_1 \cdot m_2 \cdot \cdots \cdot m_n.$$

Ako su parovi modula uzajamno prosti, na ovaj način se može predstaviti bilo koji interval od $m$ uzastopnih brojeva. Na primer, ukoliko treba predstaviti neoznačene brojeve, može se uzeti interval $[0, m-1]$, a za označene interval $[-m/2, m/2-1]$. Ukoliko izlazimo van nekog intervala od $m$ uzastopnih brojeva, zapis takođe ne može biti jednoznačan. Tako će, na primer, svi brojevi iz intervala $[0, m-1]$, $[m, 2m-1]$ i $[2m, 3m-1]$ imati isti zapis.

## 6.1 Modularna aritmetika

**Zapis negativnih brojeva.** Za zapis negativnog dekadnog broja, prvo se vrši prevođenje odgovarajućeg pozitivnog broja, nakon čega se svakoj od dobijenih cifara u RBS zapisu promeni znak. Nakon toga se rezultat dobija tako što se svaka cifra iz RBS zapisa zameni onom koja joj je jednaka po modulu $m_i$, a koja se nalazi u intervalu $[0, m_i - 1]$.

Na primer, neka je potrebno zapisati dekadni broj $-538$ u sistemu RBS(9|7|4). Najpre ćemo odgovarajući pozitivan broj 538 zapisati u tom sistemu. Važi da je $538 \bmod 9 = 7$, $538 \bmod 7 = 6$ i $538 \bmod 4 = 2$, pa je zapis pozitivnog broja $(538)_{10} = (7|6|2)_{\text{RBS}(9|7|4)}$. Za zapis negativnog broja, najpre je potrebno promeniti znak svakoj od cifara 7, 6 i 2. Dobija se da je $(-538)_{10} = (-7|-6|-2)_{\text{RBS}(9|7|4)}$. Nenegativan ceo broj manji od 9 koji pri deljenju sa 9 daje isti ostatak kao $-7$ je 2. Slično je i $-6 \equiv 1 \pmod 7$ i $-2 \equiv 2 \pmod 4$, pa je konačno $(-538)_{10} = (2|1|2)_{\text{RBS}(9|7|4)}$.

**Sabiranje.** Za bilo koju od operacija sabiranja, oduzimanja, množenja i deljenja, potrebno je da dva broja budu u RBS zapisu sa istim modulima. Ako su dati brojevi $(a_n|a_{n-1}|\ldots|a_2|a_1)$ i $(b_n|b_{n-1}|\ldots|b_2|b_1)$ u zapisu RBS$(m_n|m_{n-1}|\ldots|m_2|m_1)$, vrednost njihovog zbira je broj $(c_n|c_{n-1}|\ldots|c_2|c_1)$, gde je $c_i = (a_i + b_i) \bmod m_i$, $1 \leq i \leq n$. Na primer, važi da je $(3|2|2)_{\text{RBS}(7|5|3)} + (4|1|1)_{\text{RBS}(7|5|3)} = (3+4|2+1|2+1)_{\text{RBS}(7|5|3)} = (7|3|3)_{\text{RBS}(7|5|3)} = (0|3|0)_{\text{RBS}(7|5|3)}$.

**Oduzimanje.** Ako su dati brojevi $(a_n|a_{n-1}|\ldots|a_2|a_1)$ i $(b_n|b_{n-1}|\ldots|b_2|b_1)$ u brojčanom sistemu RBS$(m_n|m_{n-1}|\ldots|m_2|m_1)$, vrednost njihove razlike je $(c_n|c_{n-1}|\ldots|c_2|c_1)$, gde je $c_i = (a_i - b_i) \bmod m_i$, $1 \leq i \leq n$. Ako se pri oduzimanju $a_i - b_i$ dobije negativna cifra, ona se dopunjuje do pozitivne, slično kao u zapisu negativnog broja. Na primer, važi da je $(3|2|2)_{\text{RBS}(7|5|3)} - (4|1|1)_{\text{RBS}(7|5|3)} = (3-4|2-1|2-1)_{\text{RBS}(7|5|3)} = (-1|1|1)_{\text{RBS}(7|5|3)} = (6|1|1)_{\text{RBS}(7|5|3)}$.

**Množenje.** Ako su brojevi $(a_n|a_{n-1}|\ldots|a_2|a_1)$ i $(b_n|b_{n-1}|\ldots|b_2|b_1)$ zapisani u brojčanom sistemu RBS$(m_n|m_{n-1}|\ldots|m_2|m_1)$, njihov proizvod je broj $(c_n|c_{n-1}|\ldots|c_2|c_1)$, pri čemu je $c_i = (a_i \cdot b_i) \bmod m_i$, $1 \leq i \leq n$. Na primer, važi da je $(3|2|2)_{\text{RBS}(7|5|3)} \cdot (4|1|1)_{\text{RBS}(7|5|3)} = (3 \cdot 4|2 \cdot 1|2 \cdot 1)_{\text{RBS}(7|5|3)} = (12|2|2)_{\text{RBS}(7|5|3)} = (5|2|2)_{\text{RBS}(7|5|3)}$.

**Deljenje.** Sa operacijom deljenja u RBS zapisu treba biti obazriv. Ona se ovde izvršava znatno sporije od sabiranja, oduzimanja i množenja, a često ne mora biti ni definisana. Da bi bila definisana, potrebno je da odgovarajući par brojeva bude deljiv u smislu koji će u nastavku biti opisan. Ako su dati brojevi $(a_n|a_{n-1}|\ldots|a_2|a_1)$ i $(b_n|b_{n-1}|\ldots|b_2|b_1)$ u zapisu RBS$(m_n|m_{n-1}|\ldots|m_2|m_1)$, vrednost njihovog količnika je $(c_n|c_{n-1}|\ldots|c_2|c_1)$, gde je $c_i = (a_i/b_i) \bmod m_i$, $1 \leq i \leq n$. Zapis $c_i = (a_i/b_i) \bmod m_i$ znači zapravo da je $c_i$, ukoliko postoji, rešenje jednačine $a_i \equiv b_i c_i \pmod{m_i}$. Ukoliko za sve $1 \leq i \leq n$ takvo $c_i$ postoji, može se izvršiti deljenje. Najjednostavniji način za nalaženje takve vrednosti $c_i$ je da se redom pokušava sa nenegativnim celim brojevima, počev od 0 pa naviše.

Na primer, neka je potrebno izvršiti deljenje $(9|5|2|0)_{\text{RBS}(11|7|5|2)}$ i $(4|6|3|1)_{\text{RBS}(11|7|5|2)}$. Za vrednost prve cifre sleva rezultata, treba pronaći cifru $x$ takvu da je $9 \equiv 4x \pmod{11}$. Neposrednom proverom se proverava da $x = 0$, $x = 1$, $x = 2$, $x = 3$ i $x = 4$ nisu rešenja jednačine, a da $x = 5$ jeste, budući da je $9 \equiv 20 \pmod{11}$. Na sličan način se rešavanjem jednačina $5 \equiv 6x \pmod 7$, $2 \equiv 3x \pmod 5$ i $0 \equiv 1x \pmod 2$ dobijaju redom rešenja 2, 4 i 0, pa je količnik jednak $(5|2|4|0)_{\text{RBS}(11|7|5|2)}$.

**Aditivni i multiplikativni inverz.** Usko sa operacijama sabiranja i oduzimanja se vezuje pojam aditivnog inverza, a usko sa operacijama množenja i deljenja pojam multiplikativnog inverza. U skupu realnih brojeva, aditivni inverz je broj koji ima suprotan znak. U ovom kontekstu, aditivni inverz broja $a$ po modulu $m$ je broj $\bar{a}$, takav da je $a \equiv -\bar{a} \pmod{m}$ i $0 \leq \bar{a} < m$. Na primer, aditivni inverz broja 6 po modulu 8 je 2, jer je $6 \equiv -2 \pmod{8}$ i $0 \leq 2 < 8$. Multiplikativni inverz nekog broja u skupu realnih brojeva je njegova recipročna vrednost. Kod RBS zapisa, multiplikativni inverz broja $a$ po modulu $m$ je broj $a^{-1}$ takav da je $aa^{-1} \equiv 1 \pmod{m}$. Broj $a^{-1}$ se bira slično kao količnik kod deljenja – isprobavaju se svi celi brojevi počev od 0 pa naviše. Da bi multiplikativni inverz bio definisan, potrebno je da brojevi $a$ i $m$ budu uzajamno prosti.

## 6.2 Prevođenje brojeva

Prevođenje brojeva iz dekadnog sistema u RBS zapis je već opisano. U nastavku će biti opisano direktno prevođenje iz binarnog sistema u RBS zapis, kao i prevođenje iz RBS zapisa u dekadni sistem.

**Prevođenje iz binarnog sistema u RBS.** Ako je dat binarni broj

$$(a)_2 = (a_{n-1} \dots a_1 a_0)_2,$$

njegova dekadna vrednost iznosi

$$(a)_{10} = 2^{n-1}a_{n-1} + \cdots + 2^1 a_1 + 2^0 a_0.$$

Primetimo da, ukoliko želimo da izračunamo vrednost $(a)_{10} \bmod m$, dovoljno je za svaki sabirak izračunati $2^i a_i \bmod m$, a zatim odrediti zbir takvih sabiraka po modulu $m$. Formulom bi se ovo moglo zapisati u obliku

$$(a)_{10} \bmod m = \left( \sum_{i=0}^{n-1} (2^i a_i \bmod m) \right) \bmod m.$$

Odavde sledi da je u binarnom zapisu $(a_{n-1} \dots a_1 a_0)_2$ dovoljno posmatrati stepene $2^i$ uz koje je $a_i = 1$. Sabiranjem njihovih ostataka po modulu $m$ se dobija i vrednost ostatka deljenja broja $a$ po modulu $m$.

Na primer, neka je potrebno prevesti broj $a = (10111010)_2$ u zapis RBS(8|7|5|3). Najpre primetimo da se $a \bmod 8$ može jednostavno izračunati, budući da je dovoljno posmatrati njegove poslednje tri cifre u binarnom sistemu. Kako je $(010)_2 = (2)_{10}$, to je $a \bmod 8 = 2$. Ovo je zapravo ekvivalentno činjenici da se u dekadnom sistemu ostatak pri deljenju sa 1000 određuje tako što se posmatraju poslednje tri cifre, budući da je dekadni broj 8 u binarnom sistemu jednak 1000. Za određivanje ostataka pri deljenju sa 7, 5 i 3, potrebno je najpre primetiti da je $a = 2^7 + 2^5 + 2^4 + 2^3 + 2^1$. U nastavku ćemo kreirati tabelu vrednosti stepena $2^i$, $0 \leq i \leq 7$ pri deljenju sa 7, 5 i 3:

| $i$ | $2^i$ | $2^i \bmod 7$ | $2^i \bmod 5$ | $2^i \bmod 3$ |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 2 | 2 | 2 | 2 |
| 2 | 4 | 4 | 4 | 1 |
| 3 | 8 | 1 | 3 | 2 |
| 4 | 16 | 2 | 1 | 1 |
| 5 | 32 | 4 | 2 | 2 |
| 6 | 64 | 1 | 4 | 1 |
| 7 | 128 | 2 | 3 | 2 |

Iz tabele sada jednostavno sledi:

- $a \bmod 7 = (2 + 4 + 2 + 1 + 2) \bmod 7 = 4$,
- $a \bmod 5 = (3 + 2 + 1 + 3 + 2) \bmod 5 = 1$,
- $a \bmod 3 = (2 + 2 + 1 + 2 + 2) \bmod 3 = 0$,

pa je traženi zapis $(2|4|1|0)_{\mathrm{RBS}(8|7|5|3)}$.

**Prevođenje iz RBS u dekadni sistem.** Za određivanje dekadne vrednosti zapisa

$$(a_n|a_{n-1}|\ldots|a_2|a_1)_{\mathrm{RBS}(m_n|m_{n-1}|\ldots|m_2|m_1)}$$

potrebno je odrediti vrednosti težina $t_n, t_{n-1}, \ldots, t_2, t_1$. Težina $t_i$ je dekadni broj čiji je zapis

$$(0|\ldots|0|1|0|\ldots|0)_{\mathrm{RBS}(m_n|m_{n-1}|\ldots|m_2|m_1)}.$$

Zapis je takav da se na $i$-toj poziciji nalazi vrednost 1, a na svim ostalim vrednost 0. Odavde sledi da je $t_i \bmod m_i = 1$ i $t_j \bmod m_j = 0$ za $j \neq i$. Nakon što se pronađu vrednosti težina, dekadna vrednost broja iznosi

$$\left(\sum_{i=1}^{n} a_i t_i\right) \bmod m, \text{ gde je } m = \prod_{i=1}^{n} m_i.$$

Neka je, na primer, potrebno odrediti dekadnu vrednost broja $(3|2|2)_{\mathrm{RBS}(7|5|3)}$. Za težinu $t_3$ važi $t_3 = (1|0|0)_{\mathrm{RBS}(7|5|3)}$, odakle sledi da je $t_3 \equiv 1 \pmod 7$, $t_3 \equiv 0 \pmod 5$ i $t_3 \equiv 0 \pmod 3$. Kako je $t_3$ deljivo sa 3 i sa 5, a 3 i 5 su uzajamno prosti brojevi, zaključujemo da je $t_3$ oblika $15k$. Zamenom u prvu jednačinu sledi $15k \equiv 1 \pmod 7$, pa je $k = 1$ (redom se probaju vrednosti za $k$ počev od 0 pa naviše) i $t_3 = 15 \cdot 1 = 15$. Na sličan način se dobija da je $t_2 = 21$ i $t_1 = 70$. Konačno je

$$(3|2|2)_{\mathrm{RBS}(7|5|3)} = (3 \cdot 15 + 2 \cdot 21 + 2 \cdot 70) \bmod (7 \cdot 5 \cdot 3) = 227 \bmod 105 = 17.$$

Primetimo i da je rešenje bilo koji broj koji je oblika $17 + 105k$, $k \in \mathbb{Z}$.

## 6.3 Izbor modula

Pretpostavimo da je potrebno predstaviti cele dekadne brojeve iz intervala $[0, 1000000]$. Budući da je vrednost $2^{20}$ nešto veća od 1000000, sledi da binarni zapis dekadnog broja 1000000 ima 20 cifara, što znači da je toliko bitova potrebno za predstavljanje brojeva iz navedenog intervala na klasičan način.

U nastavku će biti dato nekoliko pristupa za što efikasnije predstavljanje brojeva iz navedenog intervala u računaru pomoću RBS zapisa. Da bi se navedeni brojevi mogli predstaviti, potrebno je da proizvod modula bude veći od 1000000, a zbog jednoznačnosti, da su moduli uzajamno prosti brojevi. Pokušajmo sa nekoliko pristupa izbora modula koji zadovoljavaju ovaj uslov.

- Pokušajmo sa izborom prvih nekoliko prostih brojeva dok njihov proizvod ne pređe 1000000. Ispostavlja se da je $2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 9699690$ prvi takav izbor koji se nameće. Međutim, primetimo da se na ovaj način može predstaviti više od 9 puta više brojeva nego što ih ima u intervalu. Zbog toga je najpogodnije iz izbora modula izostaviti broj 7, a ostaviti sve ostale proste brojeve. Dobija se zapis RBS$(19, 17, 13, 11, 5, 3, 2)$, čiji je proizvod modula 1385670. Imajući u vidu koliko svaki od modula zauzima bitova, ukupan broj bitova potrebnih za zapis na ovakav način iznosi $5 + 5 + 4 + 4 + 3 + 2 + 1 = 24$.

- Pokušajmo sada sa izborom prvih nekoliko brojeva koji su prosti ili stepeni prostih brojeva. Za prvi takav pogodan izbor važi $5 \cdot 7 \cdot 3^2 \cdot 11 \cdot 13 \cdot 2^4 \cdot 17 = 12252240$. Kako je proizvod takvih brojeva nešto više od 12 puta veći od navedenog intervala, najpogodnije je iz izbora modula izbaciti broj 11. Na taj način se dobija zapis RBS$(17, 16, 13, 9, 7, 5)$ čiji je proizvod modula 1113840. Ovaj zapis zauzima 23 bita.

- U računaru je pogodno birati module oblika $2^i$ ili $2^i - 1$, kako zbog bolje iskorišćenosti memorije, tako i zbog efikasnijeg izvršavanja operacija. Podsetimo se i da su ove vrednosti i komplementacione konstante kod potpunog, odnosno nepotpunog komplementa. Potrebno je voditi računa da moduli treba da budu uzajamno prosti brojevi. Može se primetiti da će $2^i$ ili $2^i - 1$ (za isti stepen $i$) uvek biti uzajamno prosti. Dodatno, važi da ako su brojevi $i$ i $j$ uzajamno prosti, takvi su i brojevi $2^i - 1$ i $2^j - 1$. Na ovaj način se mogu pogodno izabrati uzajamno prosti brojevi $a_{n-1} > \cdots > a_1 > a_0$ i konstruisati zapis RBS$(2^{a_{n-1}} | 2^{a_{n-1}} - 1 | \ldots | 2^{a_1} - 1 | 2^{a_0} - 1)$. Tako se, na primer, za uzajamno proste brojeve 7, 5 i 2 dobija zapis RBS$(2^7 | 2^7 - 1 | 2^5 - 1 | 2^2 - 1)$, čiji je proizvod modula 1511808. Ovaj zapis zauzima 21 bit.

## 6.4 Prednosti, mane i primene

Osnovne prednosti ovog zapisa se ogledaju u efikasnom izvršavanju operacija sabiranja, oduzimanja i množenja. Čak i za velike brojeve, cifre su veoma male, a prenos kod sabiranja i množenja ne postoji. Osnovna mana je što su operacije poput deljenja izuzetno spore i složenije za implementaciju. Nedostatak je i to što je u odnosu na standardni način zapisa zauzeće memorije veće. Zbog toga je i primena ovog zapisa najčešće tamo gde se vrši samo sabiranje, oduzimanje i množenje, a gde nema deljenja. Neke od takvih primena su u obradi digitalnih signala i u oblasti telekomunikacija.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 7 Zapis teksta u računaru

## 7.1 Azbuka

Konačna azbuka $V$ je konačni neprazni skup simbola. Elementi skupa $V$ se nazivaju simboli ili slova. Konačne niske slova iz $V$ se nazivaju reči. Prazna reč je reč koja ne sadrži nijedno slovo i najčešće se označava sa $\lambda$. Skup svih reči nad azbukom $V$ se označava sa $V^*$, a skup svih nepraznih reči sa $V^+$. Važi da je $V^+ = V^* \setminus \{\lambda\}$. Jezik $L$ je proizvoljan skup reči iz $V^*$, odnosno proizvoljan podskup tog skupa. Dužina reči $p$, u oznaci $|p|$, predstavlja broj simbola u reči $p$. Važi da je $|\lambda| = 0$. Oznaka $p^i$ predstavlja $i$ puta dopisanu reč $p$. Na primer, $p^3 = ppp$. Posebno je $p^0 = \lambda$. Ako azbuka ima $n$ znakova, broj reči u njoj koji imaju dužinu $d$ iznosi $n^d$.

Primeri nekih azbuka i jezika nad njima:

- Za azbuku $V = \{a, b\}$ jezik koji sadrži sve reči do dužine 1 je $L_1 = \{\lambda, a, b\}$, a jezik koji sadrži sve reči do dužine 2 je $L_2 = L_1 \cup \{aa, ab, ba, bb\}$.
- Za azbuku $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ jezik $L = V^+$ svih nepraznih reči te azbuke predstavlja cele brojeve u dekadnom sistemu sa eventualnim vodećim nulama.
- Za azbuku $V = \{0, 1\}$ jezik $L = V^+$ svih nepraznih reči te azbuke predstavlja cele brojeve u binarnom sistemu sa eventualnim vodećim nulama.

## 7.2 Kodovi

Neka su date dve azbuke $V_1$ i $V_2$ i dva jezika nad njima $L_1$ i $L_2$. Funkcija kodiranja je svaka funkcija $f : L_1 \to L_2$ koja slika jezik $L_1$ u jezik $L_2$. Ukoliko postoji inverz $f^{-1}$, može se definisati i funkcija dekodiranja $g = f^{-1}$. Funkcija dekodiranja $g : L_2 \to L_1$, nasuprot funkciji $f$, slika jezik $L_2$ u jezik $L_1$. Kodiranje predstavlja izračunavanje vrednosti $f(p)$ za neku reč $p \in L_1$, a dekodiranje izračunavanje vrednosti $g(q)$ za neku reč $q \in L_2$. Kod jezika $L_1$ je skup svih takvih vrednosti $f(p)$, $p \in L_1$, a može se označiti i za $f(L_1)$. Važi da je $f(L_1) \subseteq L_2$, ali ta dva skupa ne moraju da se poklapaju (ne moraju sve reči iz jezika $L_2$ da budu slike reči iz $L_1$). Kod može imati sledeće osobine:

- Kod je jednoznačan ako je funkcija $f$ 1-1. Inače, kod je višeznačan.
- Kod je ravnomeran ako je dužina svih njegovih reči ista.
- Kod je potpun ako obuhvata sve reči određene dužine u jeziku $L_2$.

Primeri nekih kodova:

- Neka su definisane azbuke $V_1 = \{+, -, *, /\}$ i $V_2 = \{0, 1\}$ i jezici nad njima $L_1 = V_1 = \{+, -, *, /\}$ i $L_2 = \{00, 01, 10, 11\}$. Vidimo da se jezik $L_1$ poklapa sa azbukom $V_1$, a da je jezik $L_2$ skup svih reči dužine 2 iz azbuke $V_2$. U narednoj tabeli su prikazani primeri nekih funkcija kodiranja $f : L_1 \to L_2$. Svi dobijeni kodovi su jednoznačni, ravnomerni i potpuni.

| Reč | Kod 1 | Kod 2 | Kod 3 |
|---|---|---|---|
| $+$ | 00 | 10 | 11 |
| $-$ | 01 | 11 | 00 |
| $*$ | 10 | 01 | 10 |
| $/$ | 11 | 00 | 01 |

- Neka su definisane azbuke $V_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ i $V_2 = \{0, 1\}$. Neka je $L_1 = V_1$, $L_2$ skup svih reči dužine 4 iz $V_2$ i neka je funkcijom kodiranja $f : L_1 \to L_2$ definisan kod koji odgovara 8421 zapisu (BCD kodiranje). Taj kod je jednoznačan i ravnomeran, ali nije potpun. Iako su sve vrednosti funkcije jednake dužine, nisu obuhvaćene sve reči dužine 4.

## 7.3 Kodne strane

Tekst se obično zamišlja kao dvodimenzionalni objekat, ali se u računaru predstavlja kao jednodimenzionalni niz karaktera. Pritom je potrebno uvesti i specijalne karaktere koji označavaju prelazak u novi red, tabulator, kraj teksta, itd. Ti specijalni karakteri se u računaru tretiraju na isti način kao slova, cifre i ostali karakteri. Osnovna ideja kod zapisivanja teksta u računaru je da se svakom karakteru pridruži odgovarajući ceo broj na unapred definisan način. Uređena lista karaktera predstavljena svojim kodovima se naziva kodna strana.

Primeri kodnih strana (jednobajtni strandard):

- ASCII. U ASCII kodu se može zapisati 128 različitih karaktera. Svakom karakteru se dodeljuje odgovarajuća sedmobitna niska. ASCII je jednobajtni standard, pa se vodeća cifra zapisa svakog karaktera ne koristi. Između ostalog, tu su predstavljeni kontrolni karakteri, velika slova engleske abecede, mala slova engleske abecede, dekadne cifre i neki interpunkcijski znakovi.

- ISO 8859-1 (Latin 1). Pomoću ove kodne strane se može zapisati 256 različitih karaktera i svakom karakteru se dodeljuje osmobitna niska. Na prvih 128 mesta se poklapa sa ASCII kodom. Pomoću nje se zapisuju svi znakovi zapadnoevropskih latinica.

- ISO 8859-2 (Latin 2). Ova kodna strana ima slične karakteristike kao Latin 1, s tim što se u ovom slučaju mogu zapisati svi znakovi istočnoevropskih latinica, uključujući srpsku.

- ISO 8859-5. Ova kodna strana ima slične karakteristike kao Latin 1, s tim što se u ovom slučaju mogu zapisati svi znakovi istočnoevropskih ćirilica, uključujući srpsku.

Primeri kodnih strana (višebajtni strandard):

- Unicode UCS-2. Ovaj standard svakom karakteru dodeljuje dvobajtni kod. Prvih 256 karaktera se poklapa sa Latin 1 standardom, a na preostalim mestima su, izmedu ostalih, obuhvaćeni karakteri današnjih jezika. U njemu mogu da se zapišu srpska latinična i ćirilična slova.
- Unicode UTF-8. UTF-8 algoritmom se svakom karakteru dodeljuje 1 ili više bajtova (do najviše 4 bajta), s tim što su kodovi definisani tako da je onim karatkerima koji se češće javljaju dodeljeno manje bajtova, a onim koji se ređe javljaju više bajtova. Na primer, svi ASCII karakteri se zapisuju pomoću jednog bajta. U njemu takođe mogu biti zapisana sva slova srpske latinice i ćirilice.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 8 Otkrivanje i korekcija grešaka

U računaru prilikom prenosa podataka, bilo da se radi o dva udaljena uređaja ili unutar jednog računarskog sistema, može doći do pojave grešaka. Greške se mogu pojaviti na prenosnom putu ili na lokacijama gde se nalaze otpremni i prijemni uređaji. One se manifestuju tako što se jedan bit ili niz susednih bitova poremeti (umesto 0 na određenom mestu stoji 1, ili obratno). Postoje dva pristupa za otkrivanje grešaka:

- Kontrola greške unazad, kod koje se samo može ustanoviti da li je došlo do greške ili ne. Ukoliko jeste došlo do greške, poruka se šalje ponovo. Ovde se uz bitove poruke šalju i određeni dodatni bitovi, koji omogućavaju primaocu da otkrije prisustvo greške. Sa povećanjem poruke, oni i dalje ne zauzimaju mnogo memorijskog prostora. Zbog toga su najčešće pogodni pri komunikaciji između dva udaljena uređaja.
- Kontrola greške unapred, kod koje se može ustanoviti da li je došlo do greške i izvršiti korekcija. Ukoliko jeste došlo do greške, korekcija se vrši na određenom bitu ili bloku bitova. I ovde se uz bitove poruke šalju dodatni bitovi, koji omogućavaju primaocu ne samo da otkrije prisustvo greške, već i mesto na kome se ona nalazi. Sa povećanjem poruke memorijski prostor koji oni zauzimaju brzo povećava. Zbog toga su najčešće pogodni pri komunikaciji unutar jednog računarskog sistema.

Izbor algoritma za otkrivanje grešaka zavisi od tipa greške i broja bitova sa greškom. Tip greške može biti pojedinačan ili proširen. U prvom slučaju se radi o grešci na jednom bitu, a u drugom o grešci na nizu uzastopnih bitova. Broj bitova sa greškom (eng. bits error rate – BER) predstavlja verovatnoću da jedan bit u definisanom vremenskom intervalu ima grešku. Na primer, ako je BER = $10^{-6}$, to znači da u proseku 1 od $10^6$ bitova ima grešku u datom intervalu vremena.

U nastavku će biti pomenuta tri pristupa za otkrivanje grešaka – kontrola parnosti, kontrola zbira bloka i ciklička provera redundanci. Biće opisan i jedan metod za otkrivanje i korekciju grešaka – Hamingov SEC kod.

## 8.1 Kontrola parnosti

Kod kontrole parnosti se uz poruku šalje i dodatan bit koji ima vrednost 1 ako je broj jedinica u poruci neparan, a vrednost 0 ako je broj jedinica paran. Na primer, pri slanju poruke 101101 pošiljalac će nadovezati i bit 0, budući da je broj jedinica paran. Poruka koja se šalje je oblika 1011010. Ako nije došlo do greške, primalac će tu poruku i primiti.

Neka je, na primer, došlo do greške na prvom bitu. Poruka koju dobija primalac je tada 0011010. Na osnovu dela 001101 primalac zaključuje da ima neparan broj jedinica, a kodiran bit je 0, čime konstatuje da je došlo do greške. Na ovaj način se može ustanoviti da li je došlo do greške ako je promenjen neparan broj bitova. Ali ako dva bita promene vrednosti, parnost će ostati ista, pa se u tim slučajevima ne može ustanoviti greška.

## 8.2 Kontrola zbira bloka

Kod kontrole zbira bloka se uz poruku šalje i dodatan broj (zapisan u binarnom sistemu na odgovarajući broj bitova) koji koji predstavlja broj jedinica u poruci. Na primer, pri slanju poruke 101101 pošiljalac će nadovezati i broj 4, budući da u njoj postoje 4 jedinice. Poruka koja se šalje je oblika 101101|4. Ako nije došlo do greške, primalac će tu poruku i primiti. Neka je, na primer, došlo do greške na trećem i četvrtom bitu. Poruka koju dobija primalac je tada 100001|4. Na osnovu dela 100001 primalac zaključuje da postoje dve jedinice, a kodiran broj je 4, čime konstatuje da je došlo do greške. Na ovaj način se može ustanoviti da li je došlo do greške ako je promenjen broj bitova tako da broj jedinica ne ostaje isti. Ali ako dva bita 0 i 1 promene vrednosti ili se izvrši zamena dva susdna bita, broj jedinica će ostati isti, pa se u takvim situacijama ne može ustanoviti greška.

## 8.3 Ciklička provera redundanci

Kod cikličke provere redundanci (alternativni naziv je CRC metoda) pošiljalac primaocu šalje kodiranu poruku i polinom generator (koji će u nastavku biti objašnjen). Primalac na osnovu kodirane poruke i oblika polinoma generatora može ustanoviti da li je došlo do greške. Ukoliko nije došlo do greške, može se iz kodirane poruke izdvojiti početna poruka, a ako je došlo do greške, primalac javlja pošiljaocu da poruku ponovo pošalje.

Neka je poruka koju pošiljalac treba da pošalje primaocu 11100110. Pošiljalac sam bira polinom generator $G(x)$, pri čemu je to bilo koji polinom čiji koeficijenti mogu da budu samo 0 ili 1. Neka je odabran polinom $G(x) = x^4 + x^3 + 1$. Budući da su koeficijenti ovakvog polinoma binarne cifre, svaki polinom generator može da ima svoj binarni zapis koji se dobija izlistavanjem koeficijenata počev od najvišeg stepena. U našem slučaju se dobija da je

$$G(x) = x^4 + x^3 + 1 = 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0,$$

pa je binarni zapis polinoma $G(x)$ dat sa 11001. Na poruku koja se šalje je potrebno dodati onoliko nula koliki je stepen polinoma generatora (u ovom slučaju 4), a zatim izvršiti deljenje tog broja i binarnog ekvivalenta. Pritom je deljenje ta dva binarna broja nešto jednostavnije od klasičnog deljenja, budući da se umesto oduzimanja u svakom koraku vrši ekskluzivna disjunkcija, i samim tim nema ni pozajmica. U našem slučaju se na početnu poruku 11100110 nadovezuje niska 0000, a dobijeni binarni zapis se deli sa 11001. U osnovi, ovde se početna poruka takođe može zameniti polinomom, a nadovezivanje $k$ nula (gde je $k$ stepen polinoma $G(x)$) predstavlja množenje te poruke sa $x^k$. Deljenje sa binarnim zapisom polinoma je zapravo deljenje sa polinomom $G(x)$. Međutim, u našem slučaju će sve biti zapisano binarnim brojevima, jer je tako zapis znatno jednostavniji. Kod ovog deljenja nije od značaja količnik, već samo ostatak. Postupak deljenja je sledeći:

```
111001100000 / 11001
11001
  10111
  11001
   11100
   11001
     10100
     11001
      11010
      11001
        110
```

Poruka koja se šalje primaocu je početna poruka na koju je nadovezan ostatak. Ostatak je potrebno dopuniti vodećim nulama, tako da bude zapisan na onoliko mesta koliki je stepen polinoma. Dakle, ostatak postaje 0110, a poruka koja se šalje je 111001100110. Uz takvu kodiranu poruku treba poslati i polinom generator $G(x) = x^4 + x^3 + 1$.

Razmotrimo sada kroz primer i obrnut scenario. Neka je do primaoca stigla kodirana poruka 1100101101 i polinom generator $G(x) = x^2 + 1$ koji je izabrao pošiljalac. Primalac najpre ustanovljava da je binarni zapis polinoma $G(x)$ dat sa 101, budući da je $G(x) = 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$. Primalac sada vrši deljenje kodirane poruke i binarnog zapisa polinoma, na analogan način kao u prethodnom slučaju. U osnovi tog deljenja se vrši deljenje dva polinoma – kodirane poruke koja se može predstaviti polinomom sa koeficijentima 0 i 1 i polinoma generatora $G(x)$. Postupak deljenja izgleda ovako:

```
1100101101 / 101
101
 110
 101
  111
  101
   100
   101
     111
     101
      100
      101
        11
```

Ako je ostatak 0, poruka je uspešno primljena, a inače to nije slučaj. Ovde je ostatak 11, pa poruka nije uspešno primljena. Primalac tada sugeriše pošiljaocu da poruku ponovo pošalje. Da poruka jeste uspešno primljena, bilo bi potrebno odrediti njen polazni oblik. Ako je stepen polinoma generatora $k$, polazni oblik poruke se dobija odbacivanjem njenih poslednjih $k$ bitova.

## 8.4 Hamingov SEC kod

Hamingov SEC kod (SEC kod je skraćenica od engleskog termina single error detection code), za razliku od prethodnih algoritama, može da otkrije postojanje greške i da izvrši korekciju na bitu na kome se greška pojavila. Algoritam će biti objašnjen na primeru

konkretne poruke koja ima 12 bitova. Svaka poruka koja će biti razmatrana će u našem slučaju imati isto toliko bitova.

Pretpostavimo da je kodirana poruka koju je pošiljalac poslao primaocu 101001100110. Ovo znači da prvih 8 bitova predstavlja bitove poruke, a poslednja 4 kontrolne bitove. Primalac će sada na osnovu bitova poruke generisati svoje kontrolne bitove i uporediti ih sa kontrolnim bitovima koju mu je prosledio pošiljalac. Na osnovu toga će zaključiti da li je došlo do greške na nekom bitu, i ako jeste odraditi potrebne korekcije. Neka su bitovi poruke označeni sa $m_8, \ldots, m_1$, a kontrolni bitovi sa $c_4, \ldots, c_1$ na sledeći način:

| $m_8$ | $m_7$ | $m_6$ | $m_5$ | $m_4$ | $m_3$ | $m_2$ | $m_1$ | $c_4$ | $c_3$ | $c_2$ | $c_1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

Sada je potrebno formirati tablicu Hamingovih SEC kodova. U prvoj koloni su dekadni brojevi od 12 do 1, u drugoj koloni odgovarajući binarni brojevi zapisani pomoću 4 bita, a u trećoj se svakom broju (od 12 do 1) dodeljuje neki od bitova $m_8, \ldots, m_1, c_4, \ldots, c_1$. Bitovi $c_4, \ldots, c_1$ se dodeljuju onim brojevima koji su stepeni dvojke (brojevima 1, 2, 4 i 8), a na preostala mesta se dodaju bitovi $m_8, \ldots, m_1$ na način prikazan u sledećoj tabeli:

| | | |
|---|---|---|
| 12 | 1100 | $m_8$ |
| 11 | 1011 | $m_7$ |
| 10 | 1010 | $m_6$ |
| 9 | 1001 | $m_5$ |
| 8 | 1000 | $c_4$ |
| 7 | 0111 | $m_4$ |
| 6 | 0110 | $m_3$ |
| 5 | 0101 | $m_2$ |
| 4 | 0100 | $c_3$ |
| 3 | 0011 | $m_1$ |
| 2 | 0010 | $c_2$ |
| 1 | 0001 | $c_1$ |

Zatim se izračunavaju kontrolni bitovi $c'_4, \ldots, c'_1$ preko bitova $m_8, \ldots, m_1$ tako što se redom u četvrtoj, trećoj, drugoj i prvoj koloni svih binarnih zapisa iz tabele u odgovarajuću formulu ubace oni bitovi $m_i$ čija je vrednost 1 i izvrši se njihova ekskluzivna disjunkcija. Drugim rečima, važi:

$$
\begin{aligned}
c'_4 &= m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 0 \oplus 1 = 0 \\
c'_3 &= m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 1 \oplus 1 \oplus 0 \oplus 1 = 1 \\
c'_2 &= m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 0 \\
c'_1 &= m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1
\end{aligned}
$$

Konačno, izračunava se vrednost $c_4c_3c_2c_1 \oplus c'_4c'_3c'_2c'_1 = 0110 \oplus 0101 = 0011$. U tabeli, u redu gde je broj 0011, nalazi se bit $m_1$. To znači da je došlo do greške baš na tom bitu i da je tu vrednost bita potrebno invertovati. Zbog toga je ispravna poruka 10100111. Ako je u pravcu odgovarajućeg broja u tabeli neka od vrednosti $c_4, \ldots, c_1$ ili ako je rezultat poslednje ekskluzivne disjunkcije broj koji se ne nalazi u tabeli (dekadna vrednost mu je manja od 1 ili veća od 12), ne dolazi do greške.

# Uvod u organizaciju i arhitekturu računara 1

Stefan Mišković

2020/2021.

# 9 Istorijat računskih sistema

## 9.1 Kontinualna i diskretna računska sredstva

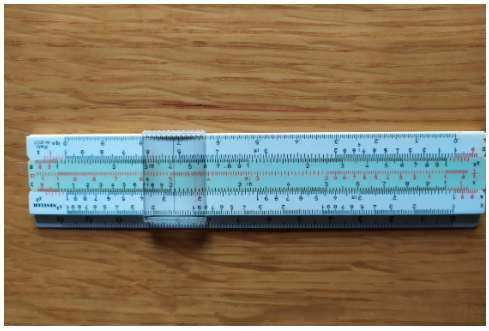
Računsko sredstvo je svako pomagalo koje je izrađeno u cilju izvršavanja računskih operacija. Prema načinu izvršavanja operacije, računsko sredstvo može biti:

- ručno (npr. lenjir) – kada ga čovek ručno pokreće i ručno očitava rezultat;
- poluautomatsko (npr. klasični kalkulator) – kada ulazne podatke zadaje čovek, a pojedinačna operacija se automatski izvršava, pri čemu čovek određuje narednu operaciju nakon što se prethodna izvrši;
- automatsko (npr. računar) – kada se automatski izvršava niz operacija koje su unapred zadate programom.

U zavisnosti od fizičkih principa na kojima se izvodi operacija, računska sredstva mogu biti kontinualna i diskretna.

**Kontinualna računska sredstva.** Kontinualna računska sredstva se izrađuju tako da njihov matematički model bude ekvivalentan matematičkom modelu problema koji rešavaju. Svi podaci se predstavljaju preko neprekidnih fizičkih veličina, zbog čega i tačnost rezultata zavisi od preciznosti izrade. Ova sredstva se izrađuju uvek za neki specifičan problem, odnosno nisu u mogućnosti da rešavaju opšte probleme. Tipični primeri ove grupe računskih sredstava su šiber (klizajući lenjir) i analogni računari (slika 1). Analogni računari su bili tipični za šezdesete godine prošlog veka. Kod njih su se vrednosti ulaznih promenljivih izražavale preko voltaže, a bili su u mogućnosti da obavljaju osnovne matematičke operacije.

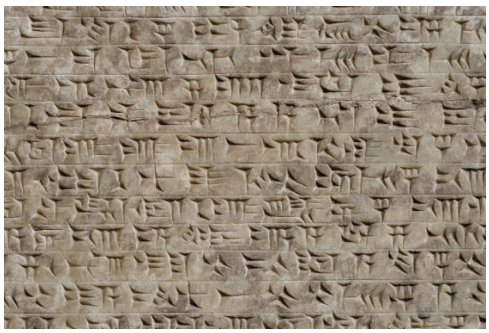
**Diskretna računska sredstva.** Diskretna računska sredstva obavljaju operacije sa diskretnim podacima, odnosno sve veličine kojima se barata se predstavljaju u obliku brojeva. U svakom trenutku se nalaze u nekom diskretnom stanju, a prelazak u naredno stanje se vrši kao rezultat spoljašnjeg uticaja. Ovde tačnost rezultata ne zavisi od preciznosti izrade. Diskretna računska sredstva mogu da rešavaju opšte probleme, a brzina izračunavanja zavisi od problema koji se rešava. Tipični primeri ove grupe su abakus i savremeni računari.



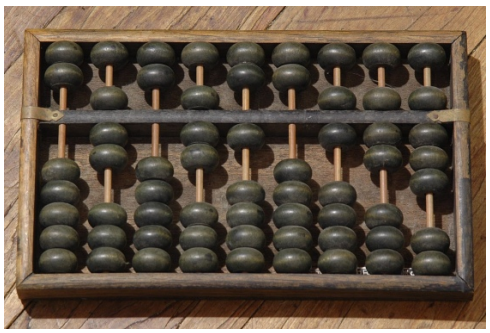
(a) šiber

(b) analogni računar

Slika 1: Primeri kontinualnih računskih sredstava



Slika 2: Klinasto pismo



Slika 3: Abakus

## 9.2 Periodi u razvoju informacionih tehnologija

U zavisnosti od nivoa tehnologije koja se koristi za rešavanje problema, razvoj informacionih tehnologija se može podeliti u četiri perioda:

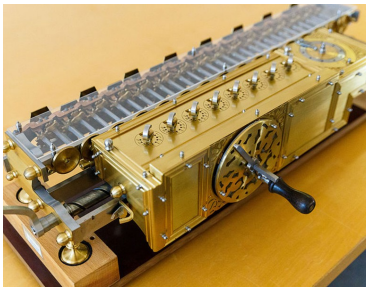
- premehanički period (od oko 3000. pre n. e. do 1450. godine n. e.),
- mehanički period (od 1450. do 1840. godine),
- elektromehanički period (od 1840. do 1939. godine),
- elektronski period (od 1939. godine do danas).

**Premehanički period.** Oko 3000. godine pre n. e. Sumeri su u Mesopotamiji kreirali klinasto pismo, koje još uvek nije dešifrovano (slika 2). Na glinenim pločicama su urezivali znake koji su odgovarali rečima u govornom jeziku. Feničani su oko 2000. godine pre n. e. pojednostavili proces pisanja, tako što su raskinuli vezu između reči i znakova i uveli pojedinačne slogove i suglasnike, i time kreirali prvi alfabet. Grci su kasnije modifikovali feničanski alfabet i dodali mu samoglasnike. Rimljani su kasnije modifikovali grčki alfabet dodelivši slovima latinska imena, koji je uticao na razvoj velikog broja današnjih pisama.

Oko 2600. godine pre n. e. Egipćani su razvili pisanje na listu biljke papirus. Oko 100. godine n. e. Kinezi su pronašli način za proizvodnju papira, koji se, neznatno modifikovan, koristi i danas. U ovom periodu se javljaju i prve biblioteke. U Mesopotamiji su postojale lične biblioteke koje su sadržale veliki broj glinenih pločica, smeštenih u specijalno označene sanduke. Egipćani su papirus umotavali u rolne oko štapova od drveta. Prve prave javne biblioteke su se pojavile u Grčkoj oko 500. godine pre n. e.

(a) Šikardova mašina

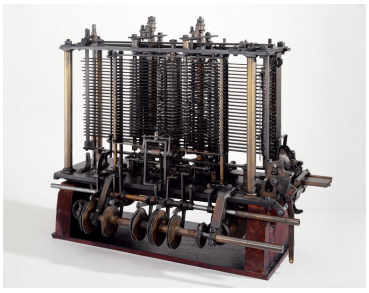
(b) Paskalova mašina



(c) Lajbnicova mašina

(d) aritmometar

(e) komptometar



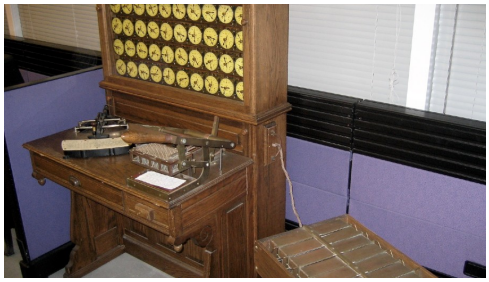
(f) analitička mašina

Slika 4: Mašine mehaničkog perioda

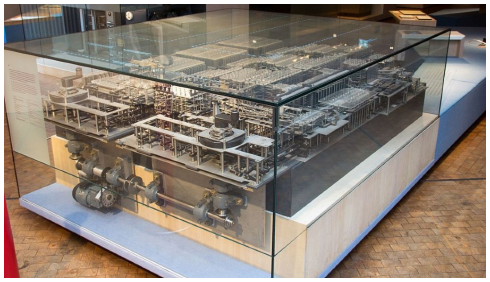
U ovom periodu su se najpre pojavili nepozicioni brojni sistemi, od kojih su najpoznatiji egipatski i rimski brojni sistem. Rimski brojni sistem se u ograničenoj meri i danas koristi. Tek su Indusi u II veku n. e. uveli devetocifreni brojni sistem, kome su Arapi u IX veku dodali nulu, kada je nastao desetocifreni brojni sistem. Najznačajnije sredstvo za računanje iz ovog perioda je abakus (slika 3).

**Mehanički period.** Početak mehaničkog perioda se vezuje za godinu kada je Johan Gutenberg otkrio štamparsku presu. Za ovaj period su značajna sledeća računska sredstva (navedena su hronološki po godini nastanka, a većina se može pronaći na slici 4):

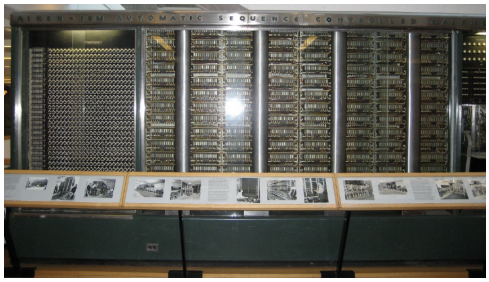
- šiber (klizajući lenjir) – ručno su se mogle obavljati jednostavne računske operacije;
- Šikardova mašina – mogla je da sabira i oduzima šestocifrene brojeve, a imala je i zvono za pojavu prekoračenja;
- Paskalova mašina – mogla je da sabira i oduzima osmocifrene brojeve;
- Lajbnicova mašina – mogla je da izvršava operacije sabiranja, oduzimanja, množenja i deljenja;
- aritmometar – na njemu su se mogle izvršavati operacije sabiranja, oduzimanja, množenja i deljenja, a to je prva računska mašina koja je ušla u serijsku proizvodnju;
- komptometar – prva računska mašina kod koje su se brojevi unosili pritiskanjem tastera;
- analitička mašina – ova mašina, koju je projektovao Bebidž, nikada nije u potpunosti napravljena zbog nedostatka sredstava, a, načinom na koji je projektovana, mogla je da izvršava razne računske operacije i sadržala je jednostavnu memoriju, zbog čega se smatra pretečom današnjih računara.



(a) Holeritov tabulator



(b) Z1



(c) MARK I

(d) Enigma

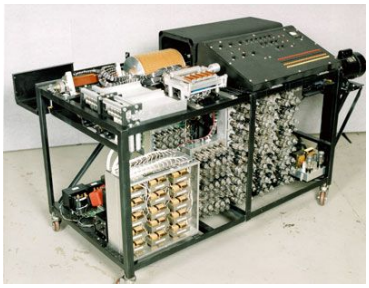
Slika 5: Mašine elektromehaničkog perioda

**Elektromehanički period.** Početak elektromehaničkog perioda se vezuje za otkriće načina iskorišćavanja elektriciteta, gde su informacije mogle da se konvertuju u električne impulse. Otkrićem telegrafa dolazi do naglog razvoja telekomunikacija. Nešto kasnije je otkrivena i Morzeova azbuka, a podmorskim telegrafskim kablom su spojene Evropa i Amerika. Kasnije je oktriven i telefon (Aleksander Grejam Bel) i radio (Markoni). Za ovaj period su karakteristične mašine koje se zasnivaju na elektromehaničkom izračunavanju, među kojima su (slika 5):

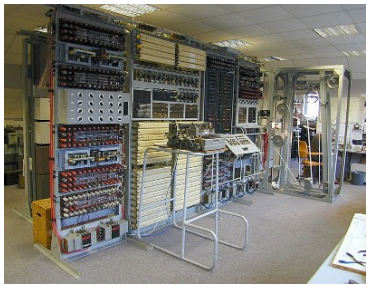
- Holeritov tabulator – mašina koja je za kratak vremenski period mogla da obradi veliku količinu podataka sa popisa stanovništva u Americi krajem XIX veka;
- Z1 – prvi elektromehanički kalkulator;
- MARK I – prvi elektromehanički programabilni kalkulator, kod koga su realni brojevi zapisivani u fiksnom zarezu;
- Enigma – mašina specijalizovane namene za šifrovanje i dešifrovanje poruka proizvedena u Nemačkoj.

**Elektronski period.** Elektronski računari su počeli da se konstruišu neposredno pred Drugi svetski rat. Ne postoji opšta saglasnost ko je prvi sagradio savremeni elektronski računar. Tehnologija za razvoj elektronskih računara se razvijala od samog početka XX veka. Nikola Tesla je najpre 1903. patentirao elektronsko logičko kolo, nakon čega su usledile konstrukcije elektronskih vakuumskih cevi i flip-flop elektronskog kola. Nešto kasnije je otkrivena televizija i ekran sa katodnom cevi. Neke od mašina koje se vezuju za ovaj period su (slika 6):

- ABC – kalkulator za rešavanje sistema linearnih jednačina, zasnovan na vakuumskim cevima;



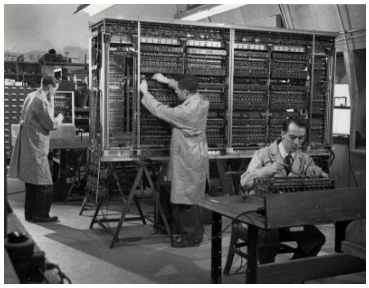
(a) ABC



(b) Kolos


(c) ENIAC

(d) EDSAC



(e) BINAC

(f) Vihor

Slika 6: Mašine elektronskog perioda

- Kolos – mašina za dešifrovanje nemačkih poruka tokom Drugog svetskog rata, zasnovana na tehnologiji sa vakuumskim cevima;
- ENIAC – ogromna mašina koja je težila do 30 tona, koja je radila u dekadnom sistemu i mogla je da obavlja osnovne računske operacije;
- EDSAC – prvi operativni računar koji je mogao da čuva program u memoriji;
- BINAC – prvi računar sa dualnim procesorom, gde je drugi procesor služio da preuzme rad u slučaju otkaza prvog;
- Vihor – prvi računar koji je namenjen radu u realnom vremenu, koji je mogao da obavi i do 500 000 sabiranja i 50 000 množenja u sekundi, što je znatno više od njegovih prethodnika.

## 9.3 Generacije elektronskih računara

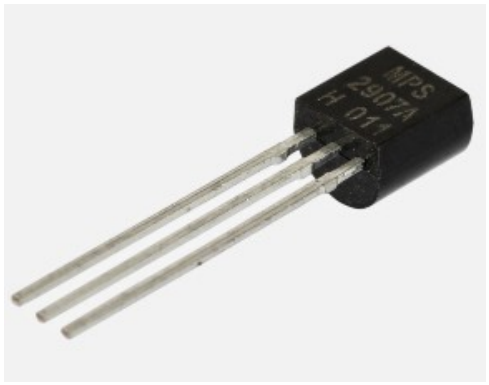
Tokom elektronskog perioda korišćene su različite tehnologije za izradu računara, na osnovu čega se mogu izdvojiti četiri generacije:

- prva generacija (od kraja tridesetih do kraja pedesetih godina XX veka),
- druga generacija (od kraja pedesetih do sredine šezdesetih),
- treća generacija (od sredine šezdesetih do početka sedamdesetih),
- četvrta generacija (od početka sedamdesetih do danas).

**Prva generacija računara.** Kao logički elementi u ovoj generaciji su se koristile vakuumske cevi (slika 7). One nisu bile dovoljno pouzdane, trošile su mnogo energije,

Slika 7: vakuumske cevi

Slika 8: UNIVAC I



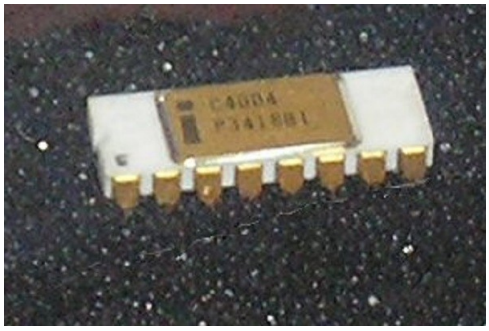

Slika 9: tranzistor

Slika 10: računar druge generacije

jako su se zagrevale i bile su velike po gabaritu. Lako su se i kvarile, tako da su im troškovi održavanja bili visoki. Unutrašnju memoriju su predstavljale magnetne trake i magnetni doboši, a za ulazno-izlazne uređaje ovi računari su, između ostalog, imali bušene kartice, pisaće mašine i štampače. Za programiranje je najpre korišćen čist mašinski jezik, a kasnije je razvijen i asembler. Krajem perioda se pojavio i Fortran, prvi viši programski jezik. Procenjuje se da je krajem perioda na tržištu bilo oko 2500 različitih računara koji pripadaju prvoj generaciji. Tipičan predstavnik ove generacije je računar UNIVAC I (slika 8), koji je sredinom XX veka koštao nešto preko milion dolara.

**Druga generacija računara.** Kao logički elementi u drugoj generaciji računara su se koristili tranzistori (slika 9). Oni su dosta manji po gabaritu od vakuumskih cevi, manje se zagrevaju i kvare, brži su i troše manje električne energije, pa je i cena njihove izrade manja. U početku su se pravili od germanijuma, a kasnije od silicijuma. Korišćenje silicijuma je znatno smanjilo cenu njihove izrade, pa je tako omogućena i masovnija proizvodnja računara. Za memorisanje podataka su se koristila magnetna jezgra, koja su mogla da čuvaju informacije i po prestanku električnog napajanja. Javljaju se i prvi operativni sistemi i razvijaju novi programski jezici, među kojima su Cobol i Lisp. Procenjuje se da je krajem perioda na tržištu bilo oko 18 000 različitih računara koji pripadaju drugoj generaciji. Na slici 10 je prikazan tipičan predstavnik računara druge generacije, gde se može videti koliko je manjeg gabarita u odnosu na računare prve generacije.

**Treća generacija računara.** Ovu generaciju računara karakteriše spajanje tranzistora u integrisano kolo (slika 11). Time je povećana brzina i pouzdanost računara, inte-



Slika 11: integrisano kolo


Slika 12: PDP-8 računar

Slika 13: IBM S/360 serija

Slika 14: MITS Altair 8800

grisana kola troše manje struje i proizvode manje toplote, a računari postaju manji, čime je omogućena njihova upotreba u različitim okruženjima. U ovom periodu dolazi do daljeg napretka na polju telekomunikacija, a lansiranjem prvih telekomunikacionih satelita počinje nov period u bežičnoj komunikaciji. Dolazi i do daljeg razvoja operativnih sistema i novih programskih jezika, među kojima su Pascal i C. Tipični predstavnici ove generacije su računar PDP-8 (slika 12) i računari iz serije S/360 kompanije IBM (slika 13).

PDP-8 je bio prvi miniračunar na tržištu. Zbog njegove niže cene i manje veličine, bio je dosta pristupačniji za obavljanje različitih poslova. U tom periodu je prodato oko 50 000 računara iz ove familije, što je kompaniju koja je proizvela PDP-8 dovelo na drugo mesto na tržištu, nakon kompanije IBM. Pored različitih poslova koje je mogao da obavlja u laboratoriji, neki proizvođači su ga ugrađivali i u svoje proizvode za dalju prodaju (ciljna funkcija tih proizvoda nije nužno bila vezana za računarstvo).

IBM S/360 serija je bila prva unapred planirana familija računara, u čiji razvoj je IBM uložio oko 5 milijardi dolara. Svi računari u seriji su imali isti ili sličan operativni sistem i skup instrukcija, pa su skuplji računari iz serije bili kompatibilni sa jeftinijim. Za veću cenu jači model je nudio veću brzinu, veću količinu unutrašnje memorije, kao i veći broj kanala na koje su mogle da se priključe ulazno-izlazne jedinice. U slučaju potrebe, korisnik je tako mogao da nadogradi postojeću mašinu sa više memorije ili jačim procesorom.

**Četvrta generacija računara.** U četvrtoj generaciji dolazi do dalje miniturizacije integrisanih kola. Početkom ovog perioda, broj komponenti na jednom delu integrisanog kola iznosio je oko 10 000, a danas taj broj premašuje nekoliko miliona. Javlja se poluprovodnička memorija, što je dovelo do pojave manjih i bržih računara, koji su imali znatno više memorije od mašina iz prethodnog perioda. Tako sredinom sedamdesetih godina dolazi do pojave personalnih računara, a prvi takav model je bio MITS Altair 8800 (slika 14). Do današnjeg dana, konstantno dolazi do napretka hardvera, u smislu

daljeg povećanja kapaciteta memorije i procesora, tako da su današnji personalni računari daleko boljih performansi od prvih modela. U isto vreme, dolazi i do pojave superračunara, mašina specifične namene, koji su po svojim karakteristikama znatno jači od ostalih računara iz perioda u kom su napravljeni. Daljim razvojem operativnih sistema i novih programskih jezika, dolazi do velikog napretka i u softverskoj podršci. Dolazi do razvoja globalnih telekomunikacionih mreža, što ima veliki uticaj na kvalitet života ljudi. Danas većina poslova zahteva u manjoj ili većoj meri poznavanje informacionih tehnologija.