UNIVERZITET U BEOGRADU
TEHNIČKI FAKULTET
BOR

INFORMATIKA 1

Profesor dr Ilija Mladenović

2008

SADRŽAJ

1 Glava - TEORIJSKE OSNOVE RAČUNARSTVA

TEORIJSKE OSNOVE RAČUNARA

1. BROJNI SISTEMI I PREVOĐENJE BROJEVA

Za zapisivanje brojeva koriste se brojni sistemi – skupovi znakova (simbola) i pravila njihovog korišćenja prilikom predstavljanja brojeva. Znaci koji se koriste za predstavljanje brojeva nazivaju se cifre ili brojke.

Digitalni računar operiše samo sa brojevima. Način na koji mašina manipuliše ovim brojevima zavisi od toga šta ovi brojevi predstavljaju (sami sebe, druge brojeve ili alfa-numeričke znakove) i u kojem su obliku oni predstavljeni. Dizajn centralne procesorske jedinice, tj. dela računara koji manipuliše svim aritmetičkim i logičkim operacijama, se ne može realizovati bez poznavanja forme u kojoj se brojevi predstavljaju od strane mašine. Šta više, ova forma može da bude drastično različita od oblika u kome se ti brojevi prezentiraju operatoru, pa je zbog toga neophodno obaviti neku vrstu konverzije u ulazno/izlaznom delu računara.

U ovoj glavi ćemo ukazati na različite načine prezentacije brojeva i drugih veličina u računaru, kao i konverziju između različitih brojnih sistema.

1.1. Suština brojnog sistema

Neka je $C = \{c_0, c_1, \ldots c_{r-1}\}$ konačan skup znakova koji služe kao cifre nekog brojnog sistema. Brojevi u ovom brojnom sistemu predstavljaju se kao nizovi konačne dužine nad azbukom C, oblika

$$c_{n-1}c_{n-2}\ldots c_1 c_0 . c_{-1}c_{-2}\ldots c_{-m} \tag{1.1}$$

Indeks koji je pridružen nekoj cifri naziva pozicija te cifre u zapisu (1.1).

Svakom takvom nizu stavlja se jednoznačno u korespodenciju količina (brojna vrednost) prema unapred usvojenim pravilima:

Svakoj cifri se posebno dodeljuje neka jednoznačna brojna vrednost.

Ukupna brojna vrednost koju označava ovakav zapis izračunava se na osnovu vrednosti cifara u svakoj poziciji, prema nekoj određenoj funkciji. Ova funkcija je najčešće sumiranje, ali se mogu sresti i neki drugi slučajevi.

Zavisno od načina na koji se određuju brojne vrednosti cifara razlikuju se dva slučaja:

- Vrednost koja se stavlja u korespodenciju svakoj cifri u zapisu broja ne zavisi od pozicije u kojoj se ta cifra nalazi. Takvi brojni sistemi se nazivaju nepozicioni brojni sistemi.
- Vrednost koja se stavlja u korespodenciju svakoj cifri u zapisu broja zavisi od pozicije u kojoj se ta cifra nalazi. Takvi brojni sistemi se nazivaju pozicioni ili

težinski brojni sistemi. To znači da u pozicionim brojnim sistemima ista cifra označava različite brojne vrednosti, u zavisnosti od njene pozicije u zapisu broja.

Kao primer nepozicionog brojnog sistema navodimo rimski brojni sistem. U broju *XXX*, koji je zapisan u rimskom brojnom sistemu, svaka cifra označava istu vrednost 10, nezavisno od njene pozicije. Osim toga, funkcija za izračunavanje ukupne vrednosti u ovom brojnom sistemu nije samo sabiranje, već to može biti i sabiranje i oduzimanje. Oduzimanje se primenjuje ako se u zapisu broja, gledano sleva udesno, cifra sa manjom vrednošću nalazi ispred cifre sa većom vrednošću. Na primer, prilikom izračunavanja vrednosti koja odgovara broju *CXLIV*, vrednosti cifara *C*, *L* i *V* se sabiraju dok se vrednosti cifara *X* i *I* oduzimaju, tako da je ukupna vrednost pokazana ovim zapisom jednaka 144.

Kod pozicionih brojnih sistema, pored vrednosti cifara potrebno je zadati i vrednosti težina t_i u svakoj poziciji $i = (0, \pm1, \pm2, \ldots)$. Broju koji je zapisan u obliku (1.1) odgovara brojna vrednost $A = \sum_{i=-m}^{n-1} c_i t_i$.

Kako je zadavanje težina za svaku poziciju nemoguće, u praksi se koriste brojni sistemi u kojima se vrednost težina zadaje kao funkcija rednog broja pozicije. Neka je dat brojni sistem sa r različitih cifara. Broj različitih cifara r kod ovakvih brojnih sistema naziva se *osnova* ili *baza brojnog sistema*. Ako se težine t_i u svim pozicijama i brojnog sistema sa osnovom r određuju prema pravilu

$$t_i = r^i, i = 0, \pm1, \pm2, \ldots$$

takav brojni sistem se naziva *prirodni brojni sistem*.

Brojni sistem određuju sledeći elementi:

1. *Skup* (numeričkih) *vrednosti cifara*. Sa D_i obeležavamo skup vrednosti koje uzima cifra c_i, a sa $|D_i|$ broj elemenata skupa D_i. U decimalnom brojnom sistemu važi $D_i = \{0,1,2,3,4,5,6,7,8,9\}$, $|D_i| = 10$, $i = -m, \ldots, n - 1$.

2. *Pravilo interpretacije.* Reč je o pravilu preslikavanja skupa vrednosti vektora cifara u skup brojeva.

Skup brojeva predstavljenih vektorom cifara u obliku (1.1) je *konačan skup* sa najviše

$$K = \prod_{i=-m}^{n-1} |D_i|$$

elemenata, jer je ovo najveći broj različitih vrednosti vektora cifara.

(Primer: za $D_i = D$, $i = -m, \ldots, n - 1$ važi $K = |D|^{m+n}$)

U opštem slučaju, razlikujemo dve vrste brojnih sistema:

Neredundantan brojni sistem: u nemu svaki vektor cifara predstavlja različit broj (preslikavanje 1:1).

Redundantan brojni sistem: u njemu može postojati više različitih vektora cifara koji predstavljaju isti broj.

Klasifikacija brojnih sistema sa osnovom

U brojnim sistemima sa osnovom (eng. *radix*) vektor težina

$T = \{t_{n-1}, ..., t_0, t_{-1}, ..., t_{-m}\}$ se nalazi u sledećem odnosu sa vektorom osnova

$R = \{R_{n-1}, ..., R_0, R_{-1}, ..., R_{-m}\}$:

$$t_0 = 1, t_i = t_{i-1} R_{i-1},$$

što se svodi na

$$t_0 = 1, \quad t_i = \prod_{j=0}^{i-1} R_j, \ i = 1,...,n-1, \quad t_i = \prod_{j=0}^{i-1} R_j^{-1}, \ i = -1,...,-m$$

Prema elementima u *vektoru osnova*, brojni sistemi se dele na brojne sisteme sa fiksnom i sa mešovitom osnovom.

1. *Fiksna osnova:* svi elementi vektora *osnova R* imaju istu vrednost *r*. Vektor težina *T* je tada jednak $T = \{r^{n-1}, r^{n-2}, ..., r^2, r^1, 1, r^{-1}, ..., r^{-m}\}$, a pravilo preslikavanja:

$$A = \sum_{i=-m}^{n-1} c_i t_i$$

2. *Mešovita osnova:* elementi vektora osnova su različiti. Primer za ovakav brojni sistem može biti predstavljanje vremena vektorom cifara

(<sat>, <minut>, <sekunda>).

Vrednosti za vektor osnova i vektor težina su $R = (24,60,60)$; $T = (3600,60,1)$.

1.1.1. Pozicioni brojni sistemi

Najčešće se koriste sledeći pozicioni brojni sistemi:

$r = 10$, $D = \{1,2,3,4,5,6,7,8,9\}$: decimalni (eng. *decimal*, skraćeno *dec.*)

$r = 2$, $D = \{0,1\}$: binarni (eng. *binary*, skraćeno *bin.*)

$r = 8$, $D = \{0,1,2,3,4,5,6,7\}$: oktalni (eng. *octal*, skraćeno *oct.*)

$r = 16$, $D = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ heksadekadni (eng. *hexadecimal*, skraćeno *hex.*)

Za predstavljanje brojeva u računaru koristi se binarni brojni sistem, a oktalni i heksadekadni su pogodni za "skraćeno" zapisivanje binarno predstavljenih brojeva.

Ljudi koriste pozicioni dekadni (decimalni) brojni sistem. To znači da se za osnovu računanja koristi broj 10, a kao cifre 0; 1; 2; 3; 4; 5; 6; 7; 8; 9. Niz cifara 12573 predstavlja broj

$$1*10^4 + 2*10^3 + 5*10^2 + 7*10^1 + 3*10^0.$$

Decimalni brojni sistem nije ni bolji ni lošiji od nekog drugog pozicionog brojnog sistema. Koristimo ga iz istorijskih razloga, a u upotrebu je ušao veoma

davno, najverovatnije zbog deset prstiju na rukama. Postoje zabeleške i o brojnim sistemima sa osnovom 20. Kako su se, recimo u Mezopotamiji, dosetili da koriste sistem sa osnovom 60 teško je pretpostaviti.

Uopšteno, ako radimo sa brojnim sistemom sa osnovom b, onda se kao "cifre" koriste brojevi iz skupa $\{0, 1, ..., b\text{-}1\}$, a nizom "cifara" $a_k a_{k-1} a_{k-2} \ldots a_1 a_0$ sistema sa osnovom b predstavljen je sledeći broj:

$$a_k * b^k + a_{k-1} * b^{k-1} + : : : + a_1 * b^1 + a_0 * b^0 :$$

Kod *pozicionih brojnih sistema*, razlomljeni brojevi se predstavljaju vektorom od $m+n$ cifara, pri čemu je svakoj cifri pridružena *težina* saglasna njenoj poziciji u vektoru.

Svaki pozicioni brojni sistem karakteriše se osnovom brojnog sistema r (ona se označava i sa N ili B). Cifre brojnog sistema sa osnovom r uzimaju vrednosti iz skupa $\{0, 1, ..., r\text{-}1\}$. Neka je neki broj A predstavljen sledećim nizom od n cifara:

$$A = c_{n-1}\, c_{n-2} \ldots c_1\, c_0, \tag{1.2}$$

gde je $c_i \in \{0, 1, ..., r - 1\}$, $0 \le i \le n\text{-}1$, tada se vrednost broja A može odrediti kao

$$|A| = c_{n-1} r^{n-1} + c_{n-2} r^{n-2} + \ldots + c_1 r^1 + c_0 r^0, \tag{1.3}$$

odnosno

$$|A| = \sum_{i=0}^{n-1} c_i r^i.$$

Osim za cele brojeve, ovo se može primeniti i za razlomljene brojeve. Vrednost broja

$$A = c_{n-1} c_{n-2} \ldots c_1 c_0 . c_{-1} c_{-2} \ldots c_{-m} \tag{1.4}$$

jednaka je

$$|A| = c_{n-1} r^{n-1} + c_{n-2} r^{n-2} + \ldots + c_1 r^1 + c_0 r^0 + c_{-1} r^{-1} + c_{-2} r^{-2} + \ldots + c_{-m} r^{-m}$$

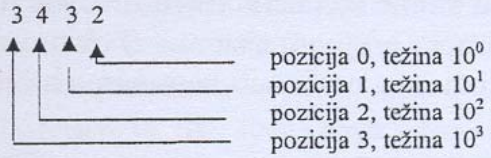
$$= \sum_{i=-m}^{n-1} c_i r^i. \tag{1.5}$$

Ilustracije radi, broj 473.9 se interpretira kao numerička vrednost

$$4*10^2 + 7*10^1 + 3*10^0 + 9*10^{-1} = 473.9.$$

Različiti stepeni broja 10 koji se koriste kod ove interpretacije, ukazuju na pretpostavku da je broj 413.8 napisan kao decimalni broj, ili broj napisan u brojnom sistemu sa *bazom* 10. Baza brojnog sistema poznata je i kao *osnova* tog sistema. *Osnova* (base ili radix) brojnog sistema je broj simbola u sistemu.

Primer 1.1. Decimalni brojni sistem ima deset simbola 0, 1, 2,..., 9, tj. ima 10 cifara. Svaka cifra decimalnog sistema je 10 puta značajnija od prethodne pozicije. Na primer posmatramo decimalni broj 3432:



Uočimo da decimalna cifra 3 na poziciji 3 ima različitu vrednost od 3 na poziciji 1. Radi se o vrednostima $3*10^3$ i $3*10^1$, respektivno.

Vrednost dekadnog broja određuje se množenjem vrednosti svake cifre u zapisu broja vrednošću pozicije na kojoj se javlja cifra, a nakon toga vrši se sabiranje proizvoda. Shodno tome, broj 3432 se interpretira kao

$$3*10^3 + 4*10^2 + 3*10^1 + 2*10^0 = 3432$$

U konkretnom slučaju, krajnja desna cifra je cifra najmanje težine (LSD - *least significant digit*), a krajnja leva cifra je cifra najveće težine (MSD - *most significant digit*). Za decimalni razlomak N=0.6341 dobija se sledeća numerička vrednost:

$$N = 6*10^{-1} + 3*10^{-2} + 4*10^{-3} + 1*10^{-4}.$$

Osim dekadnog, postoje i drugi brojni sistemi koji se mogu praktično koristiti.

U računarskim sistemima se, osim decimalnog brojnog sistema, najčešće koristi *binarni* brojni sistem ($r = 2$), *oktalni* ($r = 8$) ili *heksadecimalni* ($r = 16$) brojni sistem.

U principu, osnova brojnog sistema može biti bilo koja: 7, 11, -3, ili čak iracionalan broj kakav je π ili e. Ipak, obično se za brojnu osnovu sistema uzima pozitivna celobrojna vrednost. Kada se broj napiše u sistemu sa brojnom osnovom različitom od 10, praksa je da se posebno naglasi o kojoj se brojnoj osnovi radi. Standardno se to označava tako što se broj stavlja u zagrade, a nakon desne zagrade se napiše indeks koji ukazuje na bazu. Na primer

$$(1230)_4 = 1*4^3 + 2*4^2 + 3*4^1 + 0*4^0,$$

što je ekvivalentno broju 108 u dekadnom brojnom sistemu. Analogno je u sistemu sa osnovom 7.

$$(364.213)_7 = 3*7^2 + 6*7^1 + 4*7^0 + 2*7^{-1} + 1*7^{-2} + 3*7^{-3}.$$

Heksadekadni brojni sistem je sistem sa osnovom 16. Kao cifre u ovom sistemu se, prema tome, koriste brojevi iz skupa {0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10, 11, 12, 13, 14, 15}. Postojanje "dvocifrenih cifara" stvara mali tehnički problem, jer se, na primer, iz zapisa $(911)_{16}$ ne vidi da li se radi o dvocifrenom broju $(9\ \ 11)_{16}$ ili o trocifrenom broju $(9\ \ 1\ \ 1)_{16}$. Zato je uvedena konvencija da se cifre 10, 11, 12, 13, 14 i 15 označe slovima A, B, C, D, E i F, redom. Na taj način, skup cifara heksadekadnog brojnog sistema je {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.

1.2. Prevođenje brojeva iz jednog brojnog sistema u drugi

Zadat je broj X u sistemu sa osnovi r_1. Potrebno je naći predstavljanje tog broja u sistemu sa osnovom r_2.

Brojni sistem koji čovek najčešće koristi je *decimalni sistem*. Decimalni brojni sistem nije pogodan za korišćenje od strane računara. Računari rade mnogo efikasnije sa podacima koji su zapisani u binarnom brojnom sistemu. S obzirom da za računare nisu pogodni decimalni brojevi, a ljudi nisu naviknuti na binarne brojeve, potrebno je

ostvariti neki vid konverzije između ovih brojnih sistema koja će predstavljati interfejs između čoveka i mašine.

Generalno, razlikujemo dva postupka prevođenja broja iz brojnog sistema sa osnovom r_1 u brojni sistem sa osnovom r_2. Jedan je kada se operacije izvršavaju u brojnom sistemu sa osnovom r_2 (tj. ciljnom brojnom sistemu), a drugi je kada se operacije izvršavaju u brojnom sistemu sa osnovom r_1 (polaznom brojnom sistemu).

1.2.1. Prevođenje brojeva kod koga se operacije izvršavaju u brojnom sistemu sa osnovom r_2

Kako smo navikli da radimo u dekadnom brojnom sistemu, jasno je da se ovaj postupak koristi kada se broj iz nekog drugog brojnog sistema prevodi u dekadni brojni sistem.

Svaka cifra u težinskom brojnom sistemu ima svoju težinu, koja je zapravo neki stepen osnove. Recimo, za broj $(23014)_6$ težine cifara su:

6^4	6^3	6^2	6^1	6^0
2	3	0	1	$4_{(6)}$

Zbog toga je

$$(23014)_6 = 2\cdot 6^4 + 3\cdot 6^3 + 0\cdot 6^2 + 1\cdot 6^1 + 4\cdot 6^0$$
$$= 2592 + 648 + 0 + 6 + 4$$
$$= 3250.$$

Odatle dobijamo $(23014)_6 = (3250)_{10}$.

U opštem slučaju, prilikom konverzije broja $(c_{n-1}c_{n-2}\ldots c_1c_0.c_{-1}c_{-2}\ldots c_{-m})_{r_1}$ potrebno je da se izračuna vrednost izraza

$$X = \sum_{i=-m}^{n-1} c_i r_1^i$$

u brojnom sistemu sa osnovom r_2.

Primer 1.2. $X = (DA03)_{16}$
$= ((13*16 + 10)*16 + 0)*16 + 3$
$= ((208 + 10) * 16 + 0)*16 + 3$
$= 3488 *16 + 3 = (55811)_{10}$

Primer 1.3.

a) Broj $(1230)_4$ može se prevesti u dekadni na sledeći način:

$$(1230)_4 = 1*4^3 + 2*4^2 + 3*4^1 + 0*4^0 = (108)_{10}.$$

b) Broj $(364.213)_7$ može se prevesti u dekadni na sledeći način:

$$(364.213)_7 = 3*7^2 + 6*7^1 + 4*7^0 + 2*7^{-1} + 1*7^{-2} + 3*7^{-3}$$
$$= (193.3148690\ldots)_{10}.$$

c) Prevesti dekadne brojeve 10, 100 i 1000 u brojni sistem sa osnovom 5.

$10 = 2*5^1 = (20)_5$ (2 na poziciji 1).

$100 = 4*5^2 = (400)_5$.

$1000 = 1*5^4 + 3*5^3 = (1300)_5$.

1.2.2. Prevođenje brojeva kod koga se operacije izvršavaju u brojnom sistemu sa osnovom r_1

Ovde je situacija komplikovanija, jer se na različite načine vrši prevođenje celih i razlomljenih brojeva.

Prevođenje celih brojeva

Neka je ceo broj X u brojnom sistemu sa osnovom r_1 predstavljen na sledeći način:

$$(X)_{r_1} = x_n\, x_{n-1} \ldots x_1 x_0, \tag{1.6}$$

Neka se taj isti broj u brojnom sistemu sa osnovom r_2 predstavlja u obliku:

$$(X)_{r_2} = y_p\, y_{p-1} \ldots y_1 y_0 = y_p r_2^p + y_{p-1} r_2^{p-1} + \cdots + y_1 r_2 + y_0\,, \tag{1.7}$$

Ako poslednji izraz podelimo osnovom r_2 dobijamo:

$$\frac{X}{r_2} = y_p\, r_2^{\,p-1} + y_{p-1}\, r_2^{\,p-2} + \ldots + y_1\, r_2^{\,0} + \frac{y_0}{r_2}\,, \tag{1.8}$$

Primetimo da su svi sabirci sa leve strane izraza celobrojni, osim poslednjeg koji je sigurno razlomljen, jer je svaka cifra brojnog sistema manja od osnove brojnog sistema. Drugim rečima, cifra najmanje težine (y_0) u prezentaciji broja X u sistemu sa osnovom r_2 pojavljuje se kao ostatak pri ovakvom deljenju. Ostale cifre se dobijaju iterativnim ponavljanjem postupka nad celobrojnim delovima količnika.

Izraz

$$y_p\, r_2^{\,p-1} + y_{p-1}\, r_2^{\,p-2} + \ldots + y_1\, r_2^{\,0} \tag{1.9}$$

predstavlja celobrojni deo količnika $\frac{X}{r_2}$, dok y_0 predstavlja ostatak tog deljenja.

Koristimo sledeće oznake:

$$y_p\, r_2^{\,p-1} + y_{p-1}\, r_2^{\,p-2} + \ldots + y_1 = Int\left(\frac{X}{r_2}\right), \tag{1.10}$$

$$y_0 = Rem\left(\frac{X}{r_2}\right), \tag{1.11}$$

Na taj način je

$$\frac{X}{r_2} = Int\left(\frac{X}{r_2}\right) + Rem\left(\frac{X}{r_2}\right), \tag{1.12}$$

Ako se ovaj proces sada ponovi počev sa $Int\left(\frac{X}{r_2}\right)$, naredni ostatak biće y_1 a naredni celobrojni deo biće $y_p\, r_2^{\,p-2} + y_{p-1}\, r_2^{\,p-3} + \ldots + y_2$.

Algoritam se završava kada celobrojni deo $Int\left(\frac{X}{r_2}\right)$ postane jednak nuli.

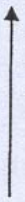
U opštem slučaju, konverzija nekog broja X iz decimalnog sistema u sistem sa osnovom b se odvija prema sledećem algoritmu: broj X delimo sa b sve dok se ne dobije količnik jednak nuli, i beležimo ostatke. Ostaci se posle deljenja ispišu od kraja ka početku. Na primer, odredimo zapis broja 9012 u sistemu sa osnovom 7:

celobrojni deo	količnik	ostatak
9012	9012:7	
1287	1287:7	3
183	183:7	6
26	26:7	1
3	3:7	5
0		3

Na taj način je $9012 = (35163)_7$.

Primer 1.4. *a*) $X=(19)_{10}=(?)_2$.

$X = 19/2 = 9\ [1$
$9/2 = 4\ [1$
$4/2 = 2\ [0$
$2/2 = 1\ [0$
$1/2 = 0\ [1$



$X=(10011)_2$

b) $X=(55811)_{10}=(?)_{16}$.

$X = 55811/16 = 3488\ [3 = 3$
$3488/16 = 218\ [0 = 0$
$218/16 = 13\ [10 = A$
$13/16 = 0\ [13 = D$

$X=(DA03)_{16}$

Primer 1.5. a) Potrebno odrediti broj u bazi 3 ekvivalentan broju (278)10. Proces konverzije je sledeći:

celobrojni deo	količnik	ostatak
	278:3	
92	92:3	$2 = a_0$
30	30:3	$2 = a_1$
10	10:3	$0 = a_2$
3	3:3	$1 = a_3$
1	1:3	$0 = a_4$
0		$1 = a_5$

Prema tome,

$(278)_{10}=(101022)_3$

Da bi proverili da li smo izvršili konverziju korektno, izvršimo konverziju broja $(101022)_3$ u decimalni broj, koristeći aritmetičke operacije u dekadnom brojnom sistemu.

$(101022)_3 = 1*3^5 + 0*3^4 + 1*3^3 + 0*3^2 + 2*3^1 + 2*3^0 = (278)_{10}$

Prevođenje razlomljenih brojeva

Neka je sada razlomljeni broj X u sitemu sa brojnom osnovom r_1 predstavljen na sledeći način:

$$(X)_{r_1} = 0.x_{-1}x_{-2}\ldots x_{-m+1}x_{-m} = \sum_{i=1}^{m} x_{-i}r_1^{-i}, \tag{1.13}$$

dok je isti broj u sistemu sa osnovom r_2 predstavljen izrazom:

$$(X)_{r_2} = 0.y_{-1}y_{-2}\ldots y_{-q+1}y_{-q} = \sum_{i=1}^{q} y_{-i}r_2^{-i}., \tag{1.14}$$

Ako ovaj izraz pomnožimo osnovom brojnog sistema r_2, dobijamo

$$X * r_2 = y_{-1} + y_{-2}\, r_2^{-1} + y_{-3}\, r_2^{-2} + \ldots + y_{-q}\, r_2^{-q+1}. \tag{1.15}$$

Prvi sabirak na desnoj strani u (1.15) je sigurno celobrojni deo proizvoda (to je cifra brojnog sistema) dok ostali sabirci predstavljaju razlomljeni deo (cifre podeljene stepenima osnove brojnog sistema). U stvari, celobrojni deo proizvoda predstavlja prvu cifru posle tačke osnove. Ako postupak nastavimo sa razlomljenim delom proizvoda dobićemo i ostale cifre. Kraj algoritma je postignut kada razlomljeni deo proizvoda postane jednak 0, ili kad broj dobijenih cifara obezbedi potrebnu tačnost.

Trebalo bi napomenuti da se ovde ne dobija uvek apsolutna tačnost, jer neki racionalni brojevi prilikom prevođenja u drugi brojni sistem postaju iracionalni. Tačnost prevođenja najčešće se zadaje kao stepen broja r_2 (osnove u koju se vrši prevođenje).

S obzirom da se kod ovog postupka prevođenja, kako celih tako i razlomljenih brojeva, operacije izvršavaju u brojnom sistemu sa osnovom r_1 (tj. u sistemu iz koga se prevodi), pogodno je da se ovaj postupak primeni kada se brojevi prevode iz dekadnog u neki drugi brojni sistem.

Primer 1.6. Izvršiti konverziju (0.27)10 = (?)4.

Proces konverzije se obavlja na sledeći način

celobrojna vrednost	razlomljeni deo	proizvod
	0.27	0.27*4 = 1.08
$a_{-1} = 1$	0.08	0.08*4 = 0.32
$a_{-2} = 0$	0.32	0.32*4 = 1.28
$a_{-3} = 1$	0.28	0.28*4 = 1.12
$a_{-4} = 1$	0.012	0.012*4

Prema tome $(0.27)_{10} = (0.1011\ldots)_4$.

Provera se može izvršiti na sledeći način:

$(0.1011\ldots)_4 = 1*4^{-1} + 0*4^{-2} + 1*4^{-3} + 1*4^{-4} + \ldots = (0.2695\ldots)_{10}$

Na osnovu dobijenog rezultata vidimo da se procesom konverzije ponekad generiše ekvivalent koji nije identičan polaznom broju.

Konverzija brojeva koji imaju celobrojni i razlomljeni deo se može izvesti tako što se posebno vrši konverzija svakog dela a zatim kombinuju rezultati.

Primer 1.7. Konverzija (123.56)10 = (?)7 se vrši na sledeći način.

Prvo se vrši konverzija celobrojne vrednosti

celobrojni deo	količnik	ostatak
	123:7	
17	17:7	$4 = a_0$
2	2:7	$3 = a_1$
0		$2 = a_2$

Nakon toga se vrši konverzija razlomljenog dela

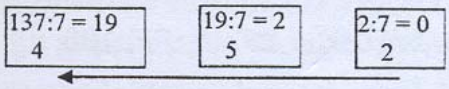
celobrojna vrednost	razlomljeni deo	proizvod
	0.56	0.56*7 = 3.92
$a_{-1} = 3$	0.92	0.92*7 = 6.44
$a_{-2} = 6$	0.44	0.44*7 = 3.08
$a_{-3} = 3$	0.08	0.08*7 = 0.56
$a_{-4} = 0$	0.56	0.56
...	...	...

Za rezultat dobijamo $(123.56)_{10} = (234.3630...)_7$

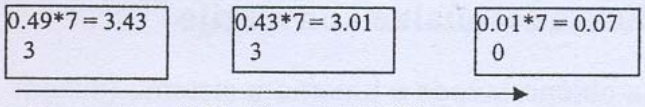
pri čemu '...' ukazuje da je proces konverzije izvršen sa određenom tačnošću..

Primer 1.8. Broj $(137.49)_{10}$ prevesti u brojni sistem sa osnovom 7 tako da greška prevođenja bude manja od 7^{-3}.

Prvo prevodimo celobrojni deo:



Zatim prevodimo razlomljeni deo, pri čemu iz uslova tačnosti prevođenja sleduje da treba odrediti 3 cifre.



Prema tome, $(137.49)_{10} = (254.330)_7$.

Konverzija između dva nedecimalna sistema se najlakše sprovodi ako se kao međukorak koristi decimalni sistem.

Primer 1.9. Konverzija (1354.24)6 = (?)4 sprovodi se najpre konverzijom iz baze 6 u bazu 10, a zatim konverzijom iz baze 10 u bazu 4:

$(1354.24)_6 = (358.4444...)_{10} = (11212.1301...)_4$

1.3. Brojanje u sistemu sa osnovom *r*

Numeričke vrednosti koje cifre u sistemu sa osnovom r mogu da uzimaju nalaze u granicama od 0 do r-1. Osim toga, broj 10 u sistemu sa osnovom r ima sledeću vrednost u dekadnom brojnom sistemu:

$$(10)_r = 1*r^1 + 0*r^0 = r_{10} \qquad (1.16)$$

Na osnovu ovih sagledavanja, možemo zaključiti da brojanje u osnovi r generiše sekvencu 0, 1, 2, ..., (r-1), 10, 11, 12, ... , 1(r-1), ...

Na slici 1.3.1. prikazana je brojačka sekvenca za različite osnove.

$r=10$	$r=2$	$r=3$	$r=8$	$r=16$
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	10	3	3
4	100	11	4	4
5	101	12	5	5
6	110	20	6	6
7	111	21	7	7
8	1000	22	10	8
9	1001	100	11	9
10	1010	101	12	*A*
11	1011	102	13	*B*
12	1100	110	14	*C*
13	1101	111	15	*D*
14	1110	112	16	*E*
15	1111	120	17	*F*
16	10000	121	20	10
17	10001	122	21	11
...	...	...	...	...

Slika 1.3.1. Brojanje u sistemima sa različitim osnovama r

Kada je $r > 10$ javlja se problem kod prezentacije onih cifara x koje se nalaze u opsegu $9<x<r$, s obzirom da ne postoje standardni simboli iz dekadnog brojnog sistema za ove brojeve. Dogovorno se za predstavljanje ovih cifara koriste velika slova engleske azbuke. Tako, na primer, za r = 16 (heksadecimalni brojni sistem) brojačka sekvenca će biti

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F*, 10, 11, ...,1*F*, 20,... ,2*F*, ..., *FF*, 100, ...

gde je $(A)_{16} = (10)_{10}$, $(B)_{16} = (11)_{10}$, itd.

1.4. Binarne, oktalne, decimalne i heksadecimalne konverzije

U samom računaru se izračunavanja obično izvode u binarnom sistemu (sistem osnove 2). Razlog je jednostavan: digitalna kola koja izvode operacije nad brojevima koriste dva stabilna stanja. Ova kola mogu raditi i sa više od dva stanja, ali je tada njihov rad nepouzdan što je neprihvatljivo sa aspekta rada sistema. On i Off elektronsko stanje predstavljaju se bitom kao on-bit i off-bit stanje. U binarnom brojnom sistemu, on bit stanje predstavlja binarnu cifru 1, dok off-bit stanje predstavlja binarnu cifru 0. Fizički, ova dva stanja se ostvaruju na različite načine. U primarnoj memoriji, ova dva stanja se predstavljaju mrežom strujnog toka. Strujno kolo može da bude uključeno ili isključeno. U sekundarnim memorijskim jedinicama, ova dva elektronska stanja su realizovana magnetizacijom fero-oksida na trakama i diskovima.

Osnova binarnog brojnog sistema je dva. Za r = 2 potrebne su samo dve cifre, a one su 0 i 1. Binarna cifra, 0 ili 1, zove se *bit*. Pozicija svakog bita odgovara nekom stepenu broja 2 (kod decimalnog je to bio stepen broja 10). Broj napisan u binarnom brojnom sistemu naziva se binarni broj.

Ljudima je svojstvena manipulacija decimalnim brojevima, pa zbog toga digitalni sistemi treba da obezbede konverziju između decimalnih i binarnih brojeva. Osim toga, oktalni i heksadecimalni brojni sistem u računarstvu omogućavaju lakše zapisivanje niza binarnih cifara.

Konverzija između binarnog i dekadnog sistema

Binarni broj sa n celih i m razlomljenih mesta piše se u obliku

$$N = b_{n-1}b_{n-2}\ldots b_1 b_0 . b_{-1}b_{-2}\ldots b_{-m}, \quad b_i \in \{0,1\}. \tag{1.17}$$

Decimalna vrednost binarnog broja formira se množenjem svakog stepena dvojke sa 0 ili 1, i sabiranjem svih vrednosti:

$$N = \sum_{i=-m}^{n} b_i 2^i \tag{1.18}$$

Na primer, decimalni ekvivalent binarnog broja $(101010)_2$ je

$$\begin{aligned} N &= (101010)_2 \\ &= 1*2^5+0*2^4+1*2^4+0*2^2+1*2^1+0*2^0 \\ &= 32+0+8+0+2+0 \\ &= 42 \end{aligned}$$

Ovaj način konverzije je pogodan za čoveka, ali ne i za mašinsku implementaciju, jer zahteva relativno složenu (sa aspekta ugrađenog hardvera ili vremena izračunavanja) operaciju stepenovanja da bi se izračunao svaki stepen dvojke.

Jasno je da se broj N u (1.18) može izraziti u obliku zbira celobrojnog i razlomljenog dela:

$$N = N_c + N_r,$$

gde je

$$N_c = \sum_{i=0}^{n} b_i 2^i, \quad N_r = \sum_{i=-1}^{-m} b_i 2^i$$

Stepenovanje pri konverziji N_c se može izbeći korišćenjem višestrukog množenja sa dva. Tada se vrednost N_c izračunava na sledeći način:

$$\begin{aligned} N_c &= \sum_{i=0}^{n-1} b_i 2^i \\ &= 2(\sum_{i=1}^{n-1} b_i 2^{i-1}) + b_0 \\ &= 2(2(\sum_{i=2}^{n-1} b_i 2^{i-2}) + b_1) + b_0 \\ &= 2(2\ldots(2\,(2b_{n-1} + b_{n-2})\ldots) + b_1) + b_0 \end{aligned}$$

Primer 1.10. Korišćenjem prethodne relacije dobija se

a) $(101010)_2 = 2*(2*(2*(2*(2*1+0)+1)+0)+1)+0$
$= 2*(2*(2*(2*2+1)+0)+1)+0$
$= 2*(2*(2*5+0)+1)+0$
$= 2*(2*10+1)+0$
$= 2*21+0 = 42$

b) $(10011)_2 = ((((1 \cdot 2) + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1$
$= (((2 + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1$
$= ((4 \cdot 2) + 1) \cdot 2 + 1$
$= 9 \cdot 2 + 1 = (19)_{10}$

U ovom algoritmu, konverzija binarnog broja sa n cifara u dekadnu vrednost svodi se na sekvencu od n-1 množenja brojem 2 i n-1 sabiranja. Proces konverzije se može izraziti sledećom poluformalnom procedurom ili algoritmom koji je poznat pod imenom *BINDECi* (BINary to DECimal integer).

Algoritam BINDECi

Ulaz: binarna celobrojna vrednost $(N)_2 = b_{n-1}\, b_{n-2} \ldots b_0$;

Izlaz: Dekadna vrednost $(N)_{10}$ koja odgovara zadatom binarnom broju.

Korak 1. Postavi N_{10} na inicijalnu vrednost 0.

Korak 2. Analizirati zadati binarni broj $(N)_2$ sa desne strane ulevo. Za svaki bit b_i izračunati novu vrednost $2*(N)_{10} + b_i$ a zatim dodeli ovu vrednost promenljivoj $(N)_{10}$. Konačna vrednost $(N)_{10}$, koja se dobija nakon n koraka, predstavlja željeni rezultat.

Primer 1.11. Prema algoritmu *BINDECi*, proces konverzije 8-cifrene binarne celobrojne vrednosti $(01100101)_2$ u decimalni oblik može se opisati na sledeći način.

Najpre ćemo izraziti $(N)2$ u obliku niza od 8 binarnih cifara:

$(N)2 = 011001012 = b7\, b6\, b5\, b4\, b3\, b2\, b1\, b0.$

Saglasno algoritmu, najpre postavljamo $(N)10$ na nulu. Proces konverzije koji sledi nakon inicijalizacije čini sledećih osam koraka.

$i = 7$ $(N)10 = 2*0 + b7 = 0$
$i = 6$ $(N)10 = 2*0 + b6 = 1$
$i = 5$ $(N)10 = 2*1 + b5 = 3$
$i = 4$ $(N)10 = 2*3 + b4 = 6$
$i = 3$ $(N)10 = 2*6 + b3 = 12$
$i = 2$ $(N)10 = 2*12 + b2 = 25$
$i = 1$ $(N)10 = 2*25 + b1 = 50$
$i = 0$ $(N)10 = 2*50 + b_0 = 101$

Odatle se dobija $(N)10 = 10110$.

Konverzija decimalnog broja u binarni zasniva se na sukcesivnom deljenju decimalnog broja brojnom osnovom 2. Ostaci deljenja, napisani u obrnutom redosledu, daju binarni ekvivalent decimalnog broja.

Primer 1.12. Proces konverzije broja 35310 u binarni je sledeći:

celobrojni deo	količnik	ostatak
	353:2=176	
176	176:2=88	$1 = a_0$
88	88:2=44	$0 = a_1$
44	44:2=22	$0 = a_2$
22	22:2=11	$0 = a_3$
11	11:2=5	$0 = a_4$
5	5:2=2	$1 = a_5$
2	2:2=1	$1 = a_6$
1	1:2=0	$0 = a_7$
0		$1 = a_8$

Shodno prethodnom, $353_{10} = 101100001_2$.

Efikasna konverzija iz dekadnog broja u binarni se izvodi na sledeći način. Decimalni broj se razlaže na dva dela - jedan deo odgovara maksimalnom stepenu broja 2 koji nije veća od datog dekadnog broja, a drugi deo ostatku koji je jednak razlici datog broja i maksimalnog stepena broja 2. Nakon toga, ponovo se ostatak razlaže na dva dela: na maksimalnu potenciju broja dva koja nije veća od ostatka i na novi ostatak. Proces se ponavlja sve dok se ne dobije ostatak koji je jednak 0. Binarna vrednost se dobija zapisivanjem 1 na bit pozicijama čije težine odgovaraju potencijama dvojke dobijenim tokom konverzije.

Na primer, analizirajmo konverziju decimalnog broja 426 u binarni.

$$426 = 256 + 170$$
$$= 256 + 128 + 42$$
$$= 256 + 128 + 32 + 10$$
$$= 256 + 128 + 32 + 8 + 2$$
$$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow$$
$$2^8 \quad 2^7 \quad 2^5 \quad 2^3 \quad 2^1$$

Prema tome, $(426)_{10} = (110101010)_2$.

Razlomljeni dekadni broj se konvertuje u binarni sukcesivnim množenjem sa 2. Ceo deo svakog proizvoda, 0 ili 1, se pamti i na taj način se formira razlomljeni broj.

Primer 1.13. Binarni ekvivalent decimalnog razlomka 0.203125 se dobija sukcesivnim množenjem razlomka sa dva, na sledeći način:

celobrojna vrednost	razlomljeni deo	proizvod
	0.203125	0. 203125*2 = 0.40625
$a_{-1} = 0$	0.40625	0.40625*2 = 0.8125
$a_{-2} = 0$	0.8125	0.8125*2 = 1.625
$a_{-3} = 1$	0.625	0.625*2 = 1.25
$a_{-4} = 1$	0.25	0.25*2=0.5
$a_{-5} = 0$	0.5	0.5*2=1
$a_{-6} = 1$	0	

Binarni ekvivalent broja $(0.203125)_{10}$ je $(0.001101)_2$.

Alternativno, konverzija razlomljenog dekadnog broja N_r u dekadni brojni sistem se vrši na sličan način.

$$N_r = \sum_{i=-1}^{-m} b_i 2^i = \sum_{i=1}^{m} b_{-i} 2^{-i}$$

$$= \tfrac{1}{2}\left(\sum_{i=1}^{n-1} \frac{b_{-i}}{2^{i-1}} + b_{-1}\right)$$

$$= \tfrac{1}{2}\left(\tfrac{1}{2}\left(\sum_{i=1}^{n-1} \frac{b_{-i}}{2^{i-2}} + b_{-2}\right) + b_{-1}\right)$$

$$= \tfrac{1}{2}\left(\tfrac{1}{2}\left(\ldots \left(\tfrac{1}{2}\ \left(\tfrac{1}{2} b_{-m} + b_{-m+1}\right) \ldots \right) + b_{-2}\right) + b_{-1}\right).$$

Primer 1.14. Posmatrajmo konverziju binarnog broja 0.101011 u decimalni.

$N = (0.101011)_2 = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 0*2^{-4} + *2^{-5} + 1*2^{-6}$.

U ovom slučaju je: $a\text{-}1 = 1$, $a\text{-}2 = 0$, $a\text{-}3 = 1$, $a\text{-}4 = 0$, $a\text{-}5 = 1$ i $a_{-6} = 1$.

Prema tome,

$$N = 0.101011 = 1/2 + 1/8 + 1/32 + 1/64 = 0.671875$$

Napomenimo da, kao i uopštem slučaju konverzije, decimalnom broju sa konačnim brojem cifara iza decimalne tačke može da odgovara binarni broj sa beskonačno mnogo binarnih cifara u razlomljenom broju. Prema tome, u procesu konverzije razlomljenog dela dekadnog broja množenje brojem 2 se produžuje sve dok se ne dobije proizvod jednak nuli ili dok se ne postigne željena tačnost.

Konverzija između binarnog i oktalnog sistema

U principu, rad sa binarnim brojevima je zaista zametan, zbog toga što se zahteva veliki broj bitova, čak i za prezentaciju malih decimalnih vrednosti. Iz ovog razloga se za prezentaciju binarnih brojeva češće koriste *oktalni* i *heksadecimalni* brojevi. Da bi ukazali na odnos između binarnih, oktalnih i heksadecimalnih brojeva razmotrićemo sledeći binarni broj.

$$(110101011)_2 = 1*2^8 + 1*2^7 + 0*2^6 + 1*2^3 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$$

$$= (1*2^2 + 1*2^1 + 0*2^0)*2^6 + (1*2^2 + 0*2^1 + 1*2^0)*2^3 + (0*2^2 + 1*2^1 + 1*2^0)*2^0$$

$$= 6*(2^3)^2 + 5*(2^3)^1 + 3*(2^3)^0$$

$$= 6*8^2 + 5*8^1 + 3*8^0 = (653)_8$$

Ovim primerom je na jedan ekstremni način prikazan postupak konverzije broja iz binarnog u oktalni sistem. Konverzija se u praksi izvodi jednostavnijim algoritmom. Vrši se grupisanje bitova u grupe od po tri bita, i svakoj grupi se dodeljuje decimalna vrednost. Za prethdodni primer je

$$(\underline{110}\ \underline{101}\ \underline{011})_2 = (6\ 5\ 3)_8$$

Na osnovu gornjeg izlaganja mogu se formulisati pravila za prevođenje brojeva iz binarnog u oktalni brojni sistem, i obrnuto. Za prelaz iz predstavljanja binarnog broja u predstavljanje u oktalnom brojnom sistemu potrebno je levo i desno od binarne tačke sve cifre zadatog broja podeliti u grupe od po tri binarne cifre. Ove

grupe binarnih cifara se nazivaju *triade*. Ukoliko na krajevima nema potpunih trijada, dodaje se potreban broj beznačajnih nula. Zatim se svaka trijada zamenjuje odgovarajućom oktalnom cifrom, prema sledećoj tabeli:

Oktalna cifra	Trijada	Oktalna cifra	Trijada
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Prelaz iz reprezentacije broja u oktalnom brojnom sistemu na predstavljanje istog broja u binarnom brojnom sistemu vrši se zamenom svake oktalne cifre odgovarajućom trijadom.

Konverzija između binarnog i heksadekadnog sistema

Analognim postupkom se vrši konverzija binarnog broja u heksadecimalni. Prilikom konverzije binarnog broja u heksadekadni, cifre binarnog broja podelimo u grupe od po 4 (tetrade), počevši sdesna, i onda svaku grupu pročitamo kao heksadecimalnu cifru. Ukoliko krajnja leva grupa binarnih cifara nema 4 elementa, dopisuju se nule na početku.

$$(\underline{1001}\ \ \underline{1110}\ \underline{0011}\ \underline{1000})_2 = (9\ E\ 3\ 8)_{16}.$$

Direktna konverzija iz heksadecimalnog u binarni sistem se takođe može obaviti brzo i efikasno: svaku cifru heksadecimalnog broja zapišemo pomoću četiri binarne cifre i pročitamo kao jedan binarni broj. Pri tome se početne nule ignorišu. Konverzija heksadekadnih cifara u binarne tetrade je data u sledećoj tabeli:

Heksadekadna cifra	Tetrada	Heksadekadna cifra	Tetrada	Heksadekadna cifra	Tetrada	Heksadekadna cifra	Tetrada
0	0000	4	0100	8	1000	*C*	1100
1	0001	5	0101	9	1001	*D*	1101
2	0010	6	0110	*A*	1010	*E*	1110
3	0011	7	0111	*B*	1011	*F*	1111

Primer 1.15. U sledećoj tabeli prikazan je odnos između decimalnog, binarnog, oktalnog i heksadekadnog brojnog sistema.

decimalni	binarni	oktalni	heksadecimalni
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Konverzija binarnog brojnog sistema u heksadecimalni sastoji se u deljenju binarnog broja u grupe od po 4 bita, i predstavljanju svake grupe heksadecimalnim ekvivalentom.

Na primer, binarni broj 1101 1000 1100 1001 se generiše kao

1101	1000	1100	1001
↓	↓	↓	↓
D	8	C	9

što je ekvivalentno sa $(D8C9)_{16}$.

Konverzija iz heksadecimalnog u binarni je direktna. Tako na primer, $(3C2E)_{16}$ se konvertuje u binarni kao

3	C	2	E	16
↓	↓	↓	↓	
0011	1100	0010	1110	2

Ponekad je potrebno konvertovati heksadecimalni broj u decimalni. Svaka cifra kod heksadecimalnog broja je 16 puta značajnija od prethodne pozicije. Prema tome, decimalni ekvivalent $2B8C_{16}$ je

$$2*16^3 + B*16^2 + 8*16^1 + C*16^0 = 2*16^3 + 11*16^2 + 8*16^1 + 12*16^0$$
$$= 8192 + 2816 + 128 + 12 = 11148.$$

Heksadecimalni brojevi se često koriste za opis podataka koji se smeštaju u memoriju. Najčešće se kapacitet memorije izražava u bajtovima. Bajt je 8-bitni podatak. S obzirom da grupa od 4 binarne cifre predstavlja jednu heksadecimalnu cifru, u jednom bajtu se mogu zapamtiti ga dve heksadecimalne cifre, odnosno jedna heksadecimalna cifra se može smestiti u pola bajta. Pola bajta se naziva nibla (*nibble*).

Primer 1.16. Konvertovati binarni broj 11010111011111 u heksadekadni.

$$11010111011111 \rightarrow 11|0101|1101|1111 \rightarrow \underset{3}{0011}|\underset{5}{0101}|\underset{D}{1101}|\underset{F}{1111} \rightarrow 35DF.$$

Primer 1.17. Konvertovati heksadekadni broj 2*CAD*3 u binarni.

	2	*C*	*A*	*D*	3
2*cad*3 →	0010	1100	1010	1101	0011

$\rightarrow (10110010101101 0011)_2$.

Konverzija između oktalnog i heksadekadnog sistema

Ako je neophodno vršiti konverziju broja iz heksadecimalne u oktalnu brojnu prezentaciju, ili obratno, lakše je koristiti binarnu decimalnu prezentaciju kao međukorak.

Na primer, konverzija $(1A8E)_{16} = (?)_8$ može se predstaviti na sledeći način:

$(1A8E)_{16} = (0001\ 1010\ 1000\ 1110)_2$

$= (001\ 101\ 010\ 001\ 110)_2 = (1\ 5\ 2\ 1\ 6)_8$

Kao što se može uočiti, rezultat se dobija prevođenjem zadatog heksadecimalnog broja u binarni ekvivalent, a zatim ponovnim grupisanjem bitova sa ciljem da se formira oktalni rezultat. Prilikom prevođenja iz heksadecimalnog

brojnog sistema u binarni svaka heksadecimalna cifra se prevodi u uređenu četvorku binarnih cifara, dok se prilikom prevođenja iz binarnog brojnog sistema u oktalni, binarne cifre grupišu u trijade koje se potom zamenjuju oktalnim ciframa. Ukoliko je potrebno, sa leve i sa desne strane binarnog broja dopisuju se nevažeće nule.

Primeri

Primer 1.18. Oktalni broj (375)8 ima vrednost

$$3*8^2 + 7*8^1 + 5*8^0 = 192 + 56 + 5$$

$$= (253)_{10}$$

Decimalni ekvivalent oktalnog broja $(14.3)_8$ je

$$(14.3)_8 \quad - \quad 1*8 + 4*8^0 + 3*8^{-1}$$

$$= \quad 8 + 4 + 0.375$$

$$= \quad (12.375)_{10}$$

Metod konverzije decimalnog broja u oktalni je isti kao i metod konverzije decimalnog broja u binarni, sa izuzetkom što se decimalni broj deli sa 8 umesto sa dva.

Primer 1.19. Oktalni ekvivalent decimalnog broja 278 je

278 : 8 = 34, ostatak 6

34 : 8 = 4, ostatak 2

4 : 8 = 0, ostatak 4

Prema tome $(278)_{10} = 426_8$.

Decimalni razlomljeni broj se konvertuje u oktalni uzastopnim množenjem razlomljenih delova sa 8. Pri ovom postupku celobrojni deo svakog proizvoda se pamti kao oktalni razlomak.

Primer 1.20. Dekadni broj (0.651)10 se konvertuje u oktalni na sledeći način:

Celobrojni deo	Razlomljeni deo	Proizvod
		0.651*8
5	0.208	0.208*8
1	0.664	0.664*8
5	0.312	0.312*8
2	0.496	0.496*8
3	0.968	0.968*8
. . .	. . .	. . .

Prema tome $0.65110 = (0.51523)_8$. Veći broj oktalnih cifara rezultiraće većom tačnošću.

Broj bitova potreban da se predstavi oktalna cifra je tri. To znači da je, ako svaku oktalnu cifru napišemo kao grupu od po tri bita, konverzija iz oktalnog u binarni sistem direktna.

Primer 1.21. Broj (270)8 se konvertuje u binarni na sledeći način

2	7	0
↓	↓	↓
010	111	000

Konverzija iz binarnog u oktalni brojni sistem je opet direktna. Počevši od LS cifre (cifre najveće težine), binarni broj se deli na grupe od po tri cifre. Svaka grupa se zatim zamenjuje odgovarajućom vrednošću.

Primer 1.22. Konverzija binarnog broja 101110010111 u oktalni se vrši na sledeći način:

101	110	010	111
↓	↓	↓	↓
5	6	2	7

Za slučaj da krajnja leva grupa nema dovoljan broj cifara tada se dopisuju nule, kao u slučaju konverzije binarnog broja 1100001011 u oktalni ekvivalent:

001	100	001	011
↓	↓	↓	↓
1	4	1	3

Kada je binarni broj razlomljen, a broj bitova se ne može grupisati u segmente od po 3 bita dodaju se nule krajnjoj desnoj i krajnjoj levoj grupi.

Primer 1.23. Binarni broj 1101111.1011 se konvertuje u oktalni na sledeći način:

001	101	111	.	101	100
↓	↓	↓		↓	↓
1	5	7	.	5	4

Konverzija decimamog broja u binarni može da se vrši prvo konverzijom u oktalni a zatim direktno iz oktalnog u binarni.

Primer 1.24. Broj 52310 se prvo konvertuje u oktalni

523 : 8 = 65, ostatak 3

65 : 8 = 8, ostatak 1

8 : 8 = 1, ostatak 0

1 : 8 = 0, ostatak 1

a zatim direktno, cifru po cifru, iz oktalnog u binarni kao

523_{10}	1	0	1	3	8
	↓	↓	↓	↓	
	001	000	001	011	2

1.5. Aritmetika u sistemu sa osnovom $r \geq 2$

Pravila za aritmetičke operacije koje važe u dekadnom brojnom sistemu mogu se primeniti i u bilo kom drugom prirodnom brojnom sistemu sa osnovom $r \geq 2$. Razlika je u tome što se koriste tablice sabiranja i množenja jednocifrenih brojeva za taj brojni sistem.

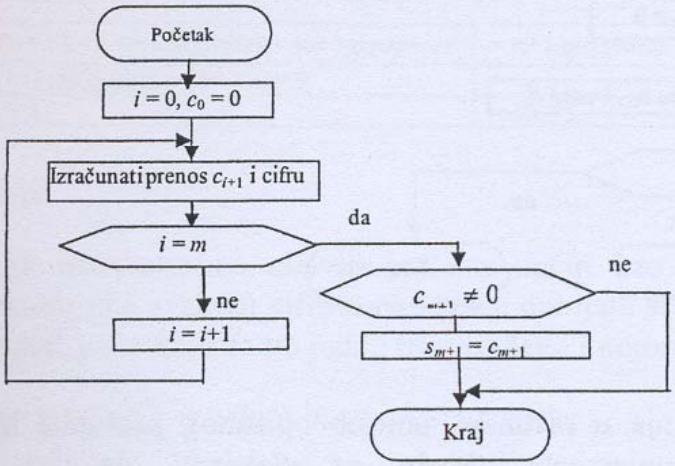
Data su dva broja, $x = x_{m-1} \ldots x_0$ i $y = y_{m-1} \ldots y_0$ u sistemu sa osnovom r.

1.5.1. Sabiranje

Sabiranje dva višecifrena broja u pozicionom sistemu sa osnovom r vrši se prema sledećem algoritmu: Prvo se sabiraju LS cifre x_0 i y_0 koristeći početni prenos c_0

= 0, i generiše se prenos c_1 i suma s_0. Sabrati dve cifre sledeće više pozicije i cifru prenosa iz prethodne niže pozicije. Novi prenos zapamtiti za sledeću višu poziciju. Ako ima preostalih pozicija u kojima nije izvršeno sabiranje preći na korak 2. Ako je izvršeno sabiranje cifara u svim pozicijama i ne postoji prenos, postupak sabiranja je završen. Ako postoji prenos, upisati 1 u sledeću višu poziciju, čime je postupak sabiranja završen.

Na sledećoj slici je prikazana procedura za sabiranje brojeva u sistemu sa osnovom r.



Slede tablice sabiranja u nekim brojnim sistemima.

$r = 2$

+	0	1
0	0	1
1	1	10

$r =$

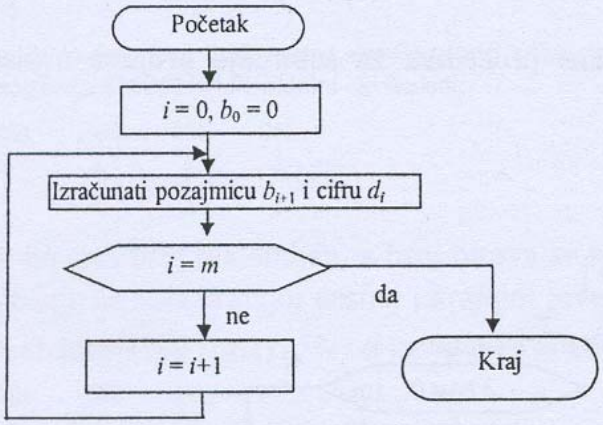
+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

$r = 16$

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

1.5.2. Oduzimanje

Oduzimanje se obavlja na sličan način kao u dekadnom brojnom sistemu. Pri tome se kod rada sa razlomljenim brojevima tačke razlomka pišu u istoj poziciji po vertikali. U datom trenutku oduzima se jedan par bitova i generiše u svakom koraku bit pozajmice, počev od najniže pozicije.



Za realizaciju oduzimanja u računaru umesto opisanog postupka koristi se predstavljanje pomoću komplementa. Može se dokazati da se pomoću komplementarnog predstavljanja operacija oduzimanja zamenjuje operacijom sabiranja.

Primer 1.25. Operacije sabiranja i oduzimanja kod oktalnih brojeva slične su onima koje se koriste kod decimalnog sistema. Naime, generiše se prenos ako suma premaši 710, kao na primeru

154_8
$+ \quad 426_8$
602_8

$6+4 = 10_{10}$	= 2+1 prenos	prva kolona ←
5+2+1 prenos	= 0+1 prenos	druga kolona ←
1+4+1 prenos	= 6	treća kolona ←

Operaciju oduzimanja analizirajmo kroz sledeći primer

670_8
-125_8
543_8

0-5 = (8-5+1 pozajmica)	= 3+1 pozajmica	prva kolona ←
7-(2+1 pozajmica)	= 7-3 = 4	druga kolona ←
6-1	= 5	treća kolona ←

Primer 1.26. Heksadecimalno sabiranje i oduzimanje se obavlja identično kao i kod bilo kog pozicionog brojnog sistema. Na primer, suma $688_{16} + 679_{16}$ se određuje kao u sledećoj tabeli:

688_{16}
$+679_{16}$
$D01_{16}$

Detaljno, ovaj postupak se može opisati na sledeći način:

$8+9=17_{10}$	$=1+1$ prenos	prva kolona ←
$8+7+1$ prenos	$=16_{10} = 0+1$ prenos	druga kolona ←
$6+6+1$ prenos	$=13_{10} = D$	treća kolona ←

Heksadecimalno oduzimanje analizirajmo na primeru oduzimanja brojeva $2A5_{16}$ i $11B_{16}$.

$2A5_{16}$
$-11B_{16}$
$18A_{16}$

(5-B) = (21-11+1 pozajmica) = 10+1 pozajmica	= A+1 pozajmica	prva kolona ←
A-(1+1 pozajmica) = $(10-2)_{10} = 8$		druga kolona ←
2-1 = 1		treća kolona ←

1.5.3. Množenje

Operacija množenja se obavlja na isti način kao i decimalno množenje. Množenjem množenika svakom cifrom množioca dobijaju se parcijalni proizvodi koji se potpisuju ispod, pomeraju za po jedno mesto ulevo i potom sabiraju.

Slede tablice množenja u nekim brojnim sistemima.

$r = 2$

×	0	1
0	0	1
1	0	1

$r = 8$

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	2	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	30
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

$r = 16$

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	15	68	70
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Primer 1.27. Množenje oktalnih brojeva 27 i 63 je oblika:

decimalno	binarno	
27	27	množenik
63	63	množilac
1701	105	prvi parcijalni proizvod
	172	drugi parcijalni proizvod
	2025	konačni proizvod

1.5.4. Deljenje

Proces deljenja sličan je standardnom decimalnom deljenju, ali bi trebalo da se koriste tablice sabiranja i množenja za taj brojni sistem.

Deljenje se u računarima vrši modifikovanim postupkom koji se sastoji u sukscesivnom oduzimanju delioca od deljenika.

1.6. Binarna aritmetika

Za binarnu aritmetiku neophodne su tabele sabiranja i množenja, prikazane u nastavku.

$0 + 0 = 0$	$0*0 = 0$
$1 + 0 = 0 + 1 = 1$	$1*0 = 0*1 = 0$
$1 + 1 = 10$	$1*1 = 1$
a)	b)

Tablice sabiranja (a) i množenja (b) binarnih brojeva.

Analiziraćemo sada realizaciju operacija sabiranja, oduzimanja, množenja i deljenja nad binarnim brojevima.

1.6.1. Binarno sabiranje

Kod sabiranja dva binarna broja, $x = x_{m-1} \ldots x_0$ i $y = y_{m-1} \ldots y_0$ prvo se sabiraju LS cifre x_0 i y_0 koristeći početni prenos $c_0 = 0$, i generiše se prenos c_1 i suma s_0. Nakon toga se sabiraju cifre x_1 i y_1 sa prenosom c_1, itd. Pri sabiranju dve binarne jedinice, rezultat je 0, a javlja se prenos u naredni razred. U sledećoj tabeli je prikazana procedura za sabiranje dve binarne cifre u poziciji i, generisanja odgovarajuće cifre i prenosa.

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tablica za sabiranje binarnih brojeva

Primer 1.28. Konkretnije, ako su $x = 11110000$, $y = 11001100$, tada ćemo kao rezultat sabiranja imati

	256	128	64	32	16	8	4	2	1
x		1	1	1	1	0	0	0	0
y		1	1	0	0	1	1	0	0
c_i	1	1	0	0	0	0	0	0	0
$x+y$	1	1	0	1	1	1	1	0	0
	s_8	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

Primer 1.29. Posmatramo još neke primere sabiranja binarnih brojeva

```
  110111.11
+   1011.10
-----------
 1000011.01
```

1.6.2. Binarno oduzimanje

Binarno oduzimanje se obavlja na sličan način. U datom trenutku oduzima se jedan par bitova i generiše u svakom koraku bit pozajmice (borrow) umesto bita prenosa, kao i bit razlike umesto bita sume. Kod operacije oduzimanja koristi se sledeća tablica binarnog oduzimanja.

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1 sa pozajmicom od naredne bit pozicije veće težine

U sledećoj tabeli je data tablica za generisanje tekuće cifre i pozajmice prilikom oduzimanja binarnih brojeva.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Primer 1.30. Od broja $x = 1111011011$ (=98110) oduzimamo broj $y = 1111011$ (= 12310).

	512	256	128	64	32	16	8	4	2	1
x	1	1	1	1	0	1	1	0	1	1
y	0	0	0	1	1	1	1	0	1	1
b_i	0	0	1	1	0	0	0	0	0	0
$x-y$	1	1	0	1	1	0	0	0	0	0
	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

Procedura startuje od cifre nakmanje težine, pri čemu se generiše bit pozajmice b_1 i razlika d_0. Zatim se postupak nastavlja do krajnje leve cifre. U konkretnom primeru prvo se nulama popunjavaju ostale bit pozicije broja y (uokvireni deo). Nakon toga se vrši oduzimanje y_0 od x_0 *i* generiše razlika d_0 i bit pozajmice b_1. Oduzimanjem y_1 i b_1 od x_1 generiše se $d_1 = 0$ i $b_2 = 0$. Proces produžava sve do generisanja d_9.

Primer 1.31. Još neki tipični primeri oduzimanja binarnih brojeva:

10000 - 101 = 1011

1010 -1 = 1001

110010 -101 = 101101

1101 - 100101 = -11000

1.6.3. Binarno množenje

Kod operacije množenja koriste se i tabela sabiranja i tabela množenja. Operacija binarnog množenja se obavlja na isti način kao i decimalno množenje. Množenjem množenika svakom cifrom množioca dobijaju se parcijalni proizvodi koji se pomeraju za po jedno mesto ulevo i potom sabiraju.

Primer 1.32. Binarno množenje decimalnih brojeva 67 sa 13 je oblika:

decimalno		binarno	
	67	1000011	- množenik
*	13	1101	- množilac
	871	1000011	- prvi parcijalni proizvod
		0000000	- drugi parcijalni proizvod
		1000011	- treći parcijalni proizvod
		1000011	- četvrti parcijalni proizvod
		1101100111	- konačni proizvod

Primer 1.33. Na sličan način, množenjem 13.5 sa 3.25 dobija se

decimalno		binarno	
	13.5	1101.10	- množenik
*	3.25	11.01	- množilac
	43.875	110110	- prvi parcijalni proizvod
		000000	- drugi parcijalni proizvod
		110110	- treći parcijalni proizvod
		110110	- četvrti parcijalni proizvod
		101011.1110	- konačni proizvod

1.6.4. Binarno deljenje

Proces deljenja sličan je standardnom decimalnom deljenju. U sustini, binarno deljenje je jednostavnije jer prilikom provere koliko puta se delilac sadrži u deljeniku postoje samo dve mogućnosti: 0 ili 1.

Primer 1.34. Analizirajmo deljenje broja $(101110)_2$ $(=46_{10})$ sa $(111)_2$ $(=7_{10})$.

```
101110 : 111  = 0001
 111
 ---
 100
```

S obzirom da je delilac 111 veći od prva tri bita deljenika, prva tri bita količnika su jednaki 0. Delilac je manji od prva četiri bita deljenika. Tada je deljenje moguće, pa je četvrti bit količnika 1. Razlika je manja od delioca tako da uzimamo i naredni bit deljenika.

```
101110  :  111 =  00011
 111
 ---
 1001
  111
 ---
   10
```

Razlika je manja od delioca, pa ponovo uzimamo naredni bit deljenika.

```
101110  :  111 =  000110
 111
 ---
 1001
  111
 ---
  100          ostatak
```

Sada je deljenik manji od delioca tako da je naredni bit količnika jednak 0 pa je deljenje završeno. Decimalna konverzija celobrojnog dela i ostatka pri deljenju daje daje $46/7 = 000110_2 = 6_{10}$ a ostatak je $100_2 = 4_{10}$.

1.7. Zadaci

1. Koji decimalni ekvivalent odgovara najvećem binarnom broju koji se može izraziti sa:

   a) 8 bitova; b) 16 bitova; c) 32 bita.

2. Izvršiti konverziju sledećih binarnih brojeva u decimalne:

   a) 111010; b) 10101111.101; c) 110110110.

3. Izvršiti konverziju sledećih dekadnih brojeva u binarne:

   a) 1946; b) 2005; c) 138; d) 1998.

4. Izvrši konverziju sledećih brojeva:
   a) $(764.7)_8$ u heksadecimalni;
   b) $(F6D.C)_{16}$ u oktalni;
   c) $(147.5)_8$ u sistem sa osnovom 4.

5. Izvršiti konverziju sledećih brojeva iz naznačene baze u decimalne:
   a) $(12021)_3$,
   b) $(4321)_5$,
   c) $(A98)_{12}$.

6. Prevesti iz jedne osnove u drugu:
   6.1. $(101022.12)_3 \rightarrow (?)_{10}$
   6.2. $(1110110101.110)_2 \rightarrow (?)_8$
   6.3. $(1F4.7)_{16} \rightarrow (?)_2$
   6.4. $(9761.3)_{10} \rightarrow (?)_5$
   6.5. $(1986.95)_{10} \rightarrow (?)_5$
   6.6. $(1F4.7)_{16} \rightarrow (?)_8$
   6.7. $(576.2)_8 \rightarrow (?)_2$
   6.8. $(136.7)_8 \rightarrow (?)_2 \rightarrow (?)_{10}$
   6.9. $(1A8E)_{16} = (?)_8$

7. Koji decimalni ekvivalent odgovara najvećem binarnom broju koji se može izraziti sa:
   a) 8 bitova; b) 16 bitova; c) 32 bita.

8. Izvršiti konverziju sledećih binarnih brojeva u decimalne:
   a) 111010; b) 10101111.101; c) 110110110.

9. Izvršiti konverziju sledećih dekadnih brojeva u binarne:
   a) 1946; b) 2005; c) 138; d) 1998.

10. Izvrši konverziju sledećih brojeva:
    a) $(764.7)_8$ u heksadecimalni;
    b) $(F6D.C)_{16}$ u oktalni;
    c) $(147.5)_8$ u sistem sa osnovom 4.

11. Konvertovati sledeće binarne brojeve u heksadecimalni sistem:
    1101001010111, 1000100010001000, 10010101110

12. Konvertovati sledeće heksadecimalne brojeve u binarni sistem:
    $fff, fadffc1, 19a0c, 101010.$

13. Izvršiti konverziju sledećih brojeva iz naznačene baze u decimalne:
    a) $(12021)_3$,
    b) $(4321)_5$,
    c) $(A98)_{12}$.

14. Izračunati u sistemu sa osnovom 2
    $(10000)_2$
    $-(101)_2$

15. Izračunati u sistemu sa osnovom 8
    $(142.5)_8$
    $-(54.7)_8$

16. Izračunati u naznačenim osnovama
    a. $(F688)_{16}$
       $+ \ (679)_{16}$
    b. $(2A5)_{16}$
       $- \ (11B)_{16}$

17. Izračunati u sistemu sa osnovom 2
    1110100010.111
    * 101.01

18. Izračunati u sistemu sa osnovom 7
    $(532.14)_7$
    $* \ (35.5)_7$

19. Podeliti u binarnom brojnom sistemu
    101110 : 111

20. Za broj zapisan u decimalnom sistemu znamo da je paran ako i samo ako se završava cifrom 0, 2, 4, 6, ili 8. Da li postoji slično pravilo koje se odnosi na brojeve zapisane u binarnom sistemu? Da li postoji pravilo koje nam pomaže da brzo utvrdimo da li je broj zapisan u binarnom sistemu deljiv sa 4?

2. PREDSTAVLJANJE PODATAKA U RAČUNARU

Podaci se u računaru predstavljaju u različitim formatima, zavisno od tipova podataka koji se obrađuju kao i od tipova obrade koji se nad njima primenjuju. Zajedničko za sve formate je da predstavljaju podatke pomoću niza binarnih cifara, čije se generisanje zasniva na binarnom brojnom sistemu i binarnim kodovima.

Sve matematičke funkcije se mogu izraziti preko četiri osnovne aritmetičke operacije: sabiranje, oduzimanje, množenje i deljenje. Ove operacije se mogu izvršavati na tri načina, kojima odgovaraju tri načina predstavljanja brojeva: fiksni zarez (fixed point), pokretni zarez (floating point) i binarno kodirani dekadni brojevi (BCD - Binary Coded Decimals).

Aritmetika u fisknom zarezu se koristi kod problema kod kojih se podaci predstavljaju sa fiksiranom pozicijom tačke osnove. Operacije u fiksnom zarezu se mogu podeliti na celobrojne (kada je tačka osnove desno od broja) i operacije sa razlomcima (tačka osnove je levo od broja). Aritmetika u pokretnom zarezu se koristi za različita naučno-tehnička izračunavanja. Aritmetičke operacije u pokretnom zarezu se mogu podeliti na *normalizovane* i *nenormalizovane.*

2.1. Osnovne organizacione jedinice podataka

Bit (Binary Digit) je najmanja jedinica podataka. Jedan bit informacije daje odgovor na pitanje: tačno ili netačno; 1 ili 0.

Bajt (Binary term-Bytes) predstavlja niz od 8 binarnih jedinica i nula:

1 byte = 8 bit = 2^3 bit.

Bajt prestavlja najmanju količinu podataka u računarskom sistemu koja se može adresirati i kojoj se može pristupati. Jedan bajt može da predstavlja dve cifre dkadnog brojnog sistema ili jedan znak (karakteri). Znakovi mogu biti: alfabetski (slova engleske abecede A do Z), numerički (cifre 0 do 9) i specijalni (<,>,!,?,#,$,%,...).

Kilobajt (kB) predstavlja količinu informacija od 1024 bita :

1kB = 1024 byte = 2^{10} byte.

Primetimo da 1kB nema 1000, već 1024 bajta. Razlog za to leži u činjenici da se za obradu podataka u računaru koristi binarni brojni sistem (brojni sistem sa osnovom 2), a ne dekadni brojni sistem (brojni sistem sa osnovom 10). Zbog toga se za merenje količine informacija koriste stepeni broja 2.

Megabajt (MB) predstavlja količinu informacija od 1024 kB:

1MB = 1024 kB = 2^{10} kB.

Gigabajt (GB) predstavlja količinu informacija od 1024 MB.

Terabajt (TB) predstavlja količinu informacija od 1024 GB.

2.2. Formati predstavljanja podataka

Svi podaci se prilikom registrovanja u memoriju računara svode na neki od formata standardne dužine. Za standardne dužine formata koriste se neki uobičajeni nazivi:

Naziv	*Dužina u bitovima*
Bajt	8
Polureč	16
Reč	32
Dvostruka reč	64

Osnovni tipovi podataka mogu biti: celi brojevi (integer), realni brojevi (real ili float), znakovni tip podataka (character) ili logičke vrednosti (logical, boolean).

2.3. Predstavljanje celih brojeva

Celi brojevi se obično deklarišu kao podaci tipa integer. Svaki ceo broj se predstavlja kao niz cifara sa znakom ili bez znaka. Ako za predstavljanje pozitivnih celih brojeva koristimo n bitova, onda na taj način možemo predstaviti brojeve iz opsega $[0,2^n-1]$. Međutim, potrebno je predstavljati i negativne brojeve. Pri tome treba obezbediti sledeće:

- podjednaku distribuciju negativnih i pozitivnih brojeva,
- jednostavnu detekciju znaka,
- jedinstven prikaz nule, i
- jednostavnu implementaciju aritmetičkih operacija.

Obično se za kodiranje znaka broja koristi bit najveće težine. Ako se broj A u sistemu sa brojnom osnovom r predstavlja kao

$$A = (a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_r \tag{2.1}$$

tada cifra znaka a_{n-1} ima sledeću vrednost:

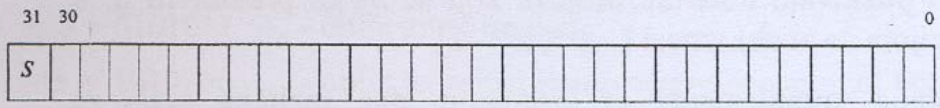
$$a_{n-1} = \begin{cases} 0, & |A| \geq 0 \\ r-1, & |A| < 0 \end{cases} \tag{2.2}$$

Kod binarnog brojnog sistema ($r = 2$) to znači

$$a_{n-1} = \begin{cases} 0, & |A| \geq 0 \\ 1, & |A| < 0 \end{cases} \tag{2.3}$$

Ostale cifre predstavljaju ili pravu vrednost broja ili njen komplement. Postoje tri načina za predstavljanje celih brojeva: označena vrednost broja ili znak-moduo, nepotpuni komplement i potpuni komplement.

U memoriji računara se za predstavljanje celih brojeva obično koristi standardni format jedne reči od 32 bita.



Krajnji levi bit u binarnom prikazu predstavlja bit znaka (sign bit), i označen je slovom *s* na slici. Ukoliko bit znaka ima vrednost 0, broj je pozitivan, a ukoliko ima vrednost 1 broj je negativan.

2.3.1. Prezentacija znak-moduo

Kod ove prezentacije se pozitivni i negativni brojevi razlikuju samo u bitu znaka (kod binarnog brojnog sistema 0 za pozitivne i 1 za negativne brojeve). Međutim, ovde se javlja problem zbog postojanja dva ravnopravna načina za predstavljanje broja nula. Praktično je moguće nulu predstaviti kao +0 i kao -0. Takođe se javlja problem i kod sabiranja brojeva različitog znaka, jer se najpre mora izvršiti poređenje apsolutnih vrednosti sabiraka da bi se odredio znak rezultata.

Primer 2.1. Binarni broj –100110 predstavljen sa 16 pozicija pri direktnom kodiranju predznakom ima oblik 1000000000100110.

2.3.2. Nepotpuni komplement

Nepotpuni komplement se još naziva i *komplement najveće cifre* pa se kod binarnog brojnog sistema još naziva i *jedinični komplement.*

Predstavljanje brojeva pomoću nepotpunog komplementa sastoji se u transformaciji prema sledećoj relaciji

$$A \to A_{NK} = \begin{cases} A, & A \geq 0 \\ r^n - 1 - |A|, & A < 0 \end{cases}$$

gde je n ukupan broj pozicija koji je predviđen za predstavljanje broja, uključujući i poziciju znaka, a r je osnova brojnog sistema.

Primer 2.2. Binarni broj –1001101011 predstavljen u 16 pozicija pomoću jediničnog komplementa ima oblik 1111110110010100.

Nenegativni celi brojevi (u opsegu od 0 do 2^{n-1} -1, ako je za predstavljanje upotrebljeno n bitova) se i dalje predstavljaju kao u binarnom pozicionom sistemu, tj.

$$A = 0\ a_{n-2}\, a_{n-3} \ldots\ a_1\, a_0 . \tag{2.4}$$

Negativni brojevi se komplementiraju do najveće cifre (tj. do jedinice kod binarnog brojnog sistema). Tako je negativni broj koji ima istu apsolutnu vrednost kao gornji broj A predstavljen u binarnom brojnom sistemu kao:

$$\overline{A} = 1\, \overline{a}_{n-2} \overline{a}_{n-3} \ldots \overline{a}_1 \overline{a}_0 , \tag{2.5}$$

ili, generalno,

$$\overline{A} = [(r-1)\, \overline{a}_{n-2} \overline{a}_{n-3} \ldots \overline{a}_1 \overline{a}_0\ ]_r , \tag{2.6}$$

gde je $\overline{a}_i = (r-1) - a_i$.

Opseg negativnih i pozitivnih binarnih brojeva koji se mogu predstaviti u n pozicija (zajedno sa pozicijom za znak) iznosi $[-2^{n-1}, 2^{n-1}-1]$

Nedostatak jediničnog komplementa je postojanje dva različita koda za vrednost 0 u binarnom sistemu, a to su: 00...0 (ako se 0 posmatra kao +0) i 11...1 (ako se 0 posmatra kao -0) .

Primer 2.3. a) $(-959)_{10}=(...99040)_{NK}$.

b) Ako se koristi 16 pozicija, nepotpuni komplement binarnog broja (-11001110) jednak je $(1111111100110001)_{NK}$.

2.3.3. Potpuni komplement

Potpuni komplement se još naziva i *komplement osnove,* pa je u binarnom brojnom sistemu njegov naziv i *dvojični komplement.*

Ako se pretpostavi da je broj cifara svakog broja D uvek jednak n, tada se svaka celobrojna vrednost D se može predstaviti u obliku sume

$$D = \sum_{i=0}^{n-1} d_i r^i.$$

Ako se nakon aritmetičke operacije generiše rezultat koji ima veći broj cifara od n tada se zadržava samo n najznačajnijih cifara (LS cifara). U tom slučaju dolazi do prekoračenja opsega.

Kod sistema *komplement brojne osnove* (*radix complement system*), komplement $\overline{D}$, n-tocifrenog broja D, se dobija oduzimanjem tog broja od r^n:

$$\overline{D} = r^n - D.$$

Na primer,

desetični komplement od 5	10-5=5	($r = 10$, $m = 1$)
desetični komplement od 27	100-27=73	($r = 10$, $m = 2$)
desetični komplement od 146	1000-146=854	($r = 10$, $m = 3$)

U opštem slučaju, ako se D nalazi između 1 i $r^n - 1$, kao rezultat se dobija drugi broj koji se nalazi u opsegu od 1 do $r^n - 1$. Ako je $D = 0$, kao rezultat se dobija r^n što odgovara broju koji ima $n+1$ cifru od kojih je MS cifra (cifra na poziciji n) jednaka 1, a iza nje sledi n nula. S obzirom da zadržavamo samo n LS cifara, to znači da se nula kao broj ekvivalentno predstavlja nizom koji sadrži n nula.

Kod binarnog brojnog sistema komplement brojne osnove se zove dvojični komplement (*two's complement*). Prezentacija pozitivnih brojeva kod dvojičnog komplementa je ista kao i kod prezentacije znak-moduo, dok se negativni broj dobija komplementiranjem prezentacije znak-moduo odgovarajućeg pozitivnog broja i sabiranjem sa 1 na mestu LS pozicije.

Na primer, dvojični komplement od -43 se određuje kao

+43 = 0101011 - oblik znak-moduo

-43 = 1010100 - oblik jedinični komplement (nepotpuni komplement)

-43 = 1010101 - oblik dvojični komplement (potpuni komplement).

Kod dvojičnog komplementa, 0 je pozitivan broj i ima jedinstvenu prezentaciju. Opseg brojeva koji se može predstaviti nalazi se u granicama -2^{n-1} do $+(2^{n-1}-1)$. Često je u toku izvođenja aritmetičkih operacija potrebno konvertovati *m*-tobitni broj u *n*-tobitni. Kod ovakvih situacija potrebno je prvo odrediti da li je $n>m$. Ako je *n* veće od *m* dodaje se $(n-m)$ nula posle bita za znak kod pozitivnih brojeva. Kada je broj negativan, pridružuje se $(n-m)$ jedinica nakon cifre znaka. S obzirom da smo dodali bitove koji su jednaki bitu znaka, ovo proširenje zovemo *znakovno proširenje* (sign-extension). Alternativno, kada je $n<m$, mi obavljamo operaciju *znakovno odsecanje* (sign-truncation), izbacivanjem $m-n$ bitova koji slede iza bita znaka. Operacija odsecanja bitova je važeća pod uslovom da su svi odsečeni bitovi identični kao i bit znaka.

Predstavljanje brojeva pomoću potpunog komplementa sastoji se u sledećoj transformaciji brojeva:

$$A \to A_{PK} = \begin{cases} A, & A \geq 0 \\ r^n - |A|, & A < 0 \end{cases}$$

gde je *n* ukupan broj pozicija koji je predviđen za predstavljanje broja, uključujući i poziciju znaka, a *r* je osnova brojnog sistema.

Nenegativni brojevi se predstavljaju na isti način kao i ranije, dok se prezentacija negativnih brojeva dobija tako što se najpre dobije nepotpuni komplement a onda na mestu najmanje težine doda 1.

Primer 2.4. Binarni broj –1001101011 predstavljen u 16 pozicija pomoću jediničnog komplementa ima oblik 1111110110010101.

Na ovaj način se sa *n* bitova mogu predstaviti brojevi u opsegu od -2^{n-1} do $+2^{n-1}-1$. Pri tome imamo jedinstven prikaz nule, a i izvođenje aritmetičkih operacija je pojednostavljeno.

U sledećoj tabeli je prikazano kodiranje brojeva kod prezentacije dvojičnog komplementa i znak moduo.

decimalni	dvojični komplement	znak-moduo
-8	1000	-
-7	1001	1111
-6	1010	1110
-5	1011	1101
-4	1100	1100
-3	1101	1011
-2	1110	1010
-1	1111	1001
0	0000	1000 ili 0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Primer 2.5. Predstaviti $-4 \leq A \leq 3$ u sistemima komplementa dvojke i jedinice.

Rešenje:

S obzirom na opseg brojeva koji se predstavljaju, dovoljno je uzeti $n = 3$ pozicije.

	A	3	2	1	0	-0	-1	-2	-3	-4
Komplement dvojke	A_c	3	2	1	0	0	7	6	5	4
	X	011	010	001	000	000	111	110	101	100
Komplement jedinice	A_c	3	2	1	0	7	6	5	4	3
	X	011	010	001	000	111	110	101	100	-

Osobine sistema komplementa dvojke i jedinice (iz primera):

1. Reprezentacija 0:
   1.1. u sistemu komplementa dvojke – jedinstvena
   1.2. u sistemu komplementa jedinice – dve reprezentacije 0.
2. Opseg brojeva:
   2.1. u sistemu komplementa dvojke nije simetričan: $[-2^{n-1}, 2^{n-1} - 1]$.
   2.2. u sistemu komplementa jedinice je simetričan: $[-(2^{n-1}-1), 2^{n-1} - 1]$.

2.3.4. Komplement aritmetika

Već smo ukazali da se operacija oduzimanja izvodi korišćenjem operacije sabiranja ako (kada) se umanjilac "korektno" kodira. Posledica ovog pristupa u odnosu na složenost ugrađenog hardvera koji obavlja ovu operaciju u računaru je da se jedinstvena hardverska celina, nazvana sabirač, može koristiti da izvrši sve aritmetičke operacije. Prilog ovome je svakako i to da se operacija binarnog množenja izvodi isključivo sabiranjem, a operacija binarnog deljenja isključivo oduzimanjem. Na ovaj način pojednostavljuje se, pre svega, proces projektovanja bloka koji vrši aritmetičke operacije, a zatim i hardver koji izvodi aritmetičke operacije. Sagledajmo sada neke detalje koji se tiču prezentacije brojeva a shodno tome i implementacije osnovnih aritmetičkih operacija.

Aritmetika u prezentaciji znak-moduo

Negativni brojevi se mogu predstaviti na više načina. U svakodnevnom životu koristimo sistem *znak-moduo* (sign-magnitude). Sa ciljem da se pojednostave aritmetička kola kod računara se koristi sistem *komplement broja,* Nezavisno od prethodne konstatacije, ukažimo prvo na složenost izvođenja operacija sabiranja i oduzimanja kod brojeva u prezentaciji znak-moduo.

Kod sistema znak-moduo broj čine dva dela, moduo i znak. Znak može biti + ili -, što ukazuje na pozitivan ili negativan znak modula. Ako ispred broja nema znaka obično to znači da je broj pozitivan. Postoje dve mogućnosti prezentacije nule "+0" i "-0" koje imaju istu vrednost.

Kada se sistem znak-moduo koristi za predstavljanje binarnih brojeva, znak se predstavlja jednim dodatnim bitom, kome je u okviru prezentacije broja dodeljeno mesto MS bit pozicije. Kada je bit znaka jednak 0, broj je pozitivan, a kada je jednak 1, broj je negativan. Ostali bitovi u okviru prezentacije broja dodeljuju se modulu. Shodno prethodnom, brojevi +123 i -123 razlikuju se samo na MS bitu na sledeći način

$01111011_2 = +123$

$11111011_2 = -123$

Sistem znak-moduo sadrži isti broj pozitivnih i negativnih celobrojnih vrednosti, koji se kod n-tobitnog broja nalaze u opsegu $-(2^{n-1}-1)$ do $+(2^{n-1}-1)$.

Kod aritmetike znak-moduo vrši se poređenje znakova i modula operanada. Za prezentaciju oba operanda i rezultata se koristi notacija $D_1 = <s_1, m_1>$, $D_2 = <s_2, m_2>$ i $D_r = <s_r, m_r>$, gde su D_1 i D_2 prvi i drugi operand a D_r rezultat, pri čemu su s_1, s_2 i s_r, kao i m_1, m_2 i m_r, pripadajući znakovi i moduli, respektivno. Kod oduzimanja brojeva istog znaka ili sabiranja brojeva različitog znaka, znak rezultata se određuje poređenjem apsolutnih vrednosti operanada. Kod operacije množenje (deljenje) vrši se množenje (deljenje) modula brojeva a rezultat je pozitivan ako oba operanda imaju isti znak, a negativan kada su različitog znaka. Uobičajeno se, kada je rezultat jednak nuli, generiše +0.

Aritmetika u prezentaciji potpunog komplementa

Kod sabiranja dva broja u dvojičnom komplementu koristi se binarna aritmetika, a eventualni prenos na mestu za znak se odbaci i tako dobije rezultat u potpunom komplementu. Sve dok opseg brojnog sistema nije premašen, dobijeni rezultat sabiranja je korektan, uključujući i znak.

Na primer, sabiranjem dva pozitivna broja generiše se pozitivan rezultat.

```
  0010     (+2)
+ 0100     (+4)
  0110     (+6)
```

Na sličan način, kada se sabiraju dva negativna broja i ignorišući bit prenosa nakon bita za znak dobije negativna suma, ona je uvek korektna.

```
                1110 (-2)
            +   1100 (-4)
ignorisani prenos = 1   1010 (-6)
```

Ipak postoje situacije kada rezultat premašuje opseg brojeva, generišući uslov poznat kao *prekoračenje* (overflow). Pravilo koje važi je sledeće: kod sabiranja dva broja različitog znaka nikad ne dolazi do prekoračenja. Ali, kada se sabiraju dva broja istog znaka čija je suma veća od najvećeg broja koji se može predstaviti dobija se nekorektni rezultat. Tako na primer, sabiranjem binarnih reprezentacija dekadnih brojeva 4 i 5 u $n = 4$ pozicija dobija se negativan rezultat.

```
   0100     (+4)
+  0101     (+5)
   1001     (-7)
```

Slična situacija se dobija sabiranjem binarnih reprezentacija negativnih dekadnih brojeva -4 i -5 u $n = 4$ pozicija. Dobija se pozitivan rezultat dobija se negativan rezultat.

```
          1100     (-4)
+         1011     (-5)
ignorisani prenos = 1 0111     (+7)
```

Uzrok pojave greške u ova dva slučaja je činjenica da se u $n = 4$ pozicija pomoću potpunog komplementa mogu predstaviti dekadni brojevi iz opsega

$[-2^3, 2^3-1] = [-8,7]$.

Formalno pravilo za detekciju prekoračenja, pogodno za čoveka je sledeće: premašaj kod sabiranja uvek se javlja kada je znak sume različit od znaka oba sabirka. Projektanti računara se za detekciju premašenja koriste sledećim pravilom: ako je bit prenosa koji se prenosi u bit znaka različit od bita prenosa koji se dobija nakon bita znaka, tada se javlja premašaj.

Da bi ukazali na ovaj problem detaljnije analizirajmo sledeće primere:

a) Izračunati razliku dekadnih brojeva 13 i 7 koristeći potpuni komplement i 5 pozicija.

```
      1   <-------------  prenos na mestu bita za znak
      0 1 1 0 1    (+13)
      1 1 0 0 1    (-7)
    1 0 0 1 1 0    (+6)  <-  korektan rezultat
bit prenosa na izlazu
```

b)

```
      0   <-------------  prenos na mestu bita za znak
      1 0 0 0 1    (-15)
      1 1 0 1 0    (-6)
    1 0 1 0 1 1    (+11) <-  nekorektan rezultat
bit prenosa na izlazu
```

c)

```
      0   <-------------  prenos na mestu bita za znak
      1 0 1 1 1    (-9)
      0 0 1 1 0    (+6)
    0 1 1 1 0 1    (-3)  <-  korektan rezultat
bit prenosa na izlazu
```

Najveći broj elektronskih kola za oduzimanje brojeva u dvojičnom komlementu ne obavlja oduzimanje direktno nego prvo komplementira umanjilac i uzima njegov dvojični komplement a zatim ga sabira sa umanjenikom koristeći pravila sabiranja.

Da bi ukazali na razliku između direktnog oduzimanja i sabiranja u dvojičnom komplementu, analizirajmo sledeća dva primera: (i) direktno oduzimanje

```
  0 0 1 0        (+2)
- 0 1 0 0        (-4)
-------------  <------------  pozajmice
  1 1 0 0
ignorisana pozajmica = 1   (-2)
```

(ii) sabiranje u dvojičnom komplementu

```
0 0 1 0  (+2)
1 1 0 0  dvojični komplement od (-4)
0 0 0    <------------  prenosi
1 1 1 0  (-2)
```

U slučaju (ii) koriste se ista pravila za detekciju premašaja kao i kod sabiranja.

Primer 2.6. Izračunati

-101011+111011

pomoću potpunog komplementa, u brojnom sistemu sa osnovom $r = 2$, vršeći aritmetičke operacije u $m = 8$ pozicija.

Rešenje.

Potpuni komplement broja -101011 jednak je (11010101)PK. Slično, potpuni komplement binarnog broja 111011 jednak je (00111011)PK. Sabiranjem potpunih komplemenata

$$\begin{array}{r} (11010101)_{PK} \\ +\ (00111011)_{PK} \\ \hline 1\ 00010000 \end{array}$$

dobija se zbir $(00010000)_{PK}$ (zbog ignorisanja prenosa iz mesta najveće težine). Prevođenjem ove brojne vrednosti iz potpunog komplementa dobija se binarni broj 10000_2.

Aritmetika u prezentaciji nepotpunog komplementa

Sabiranje brojeva u nepotpunom komplementu odvija se tako sto se brojevi saberu, a eventualni prenos na mestu za znak sabere sa cifrom najmanje težine, i tako dobije rezultat koji je takođe u nepotpunom komplementu. Aritmetika u prezentaciji nepotpunog komplementa se razlikuje od aritmetike u prezentaciji potpunog komplementa po tome što se eventualni prenos na mestu za znak sabira sa cifrom najmanje težine.

Primer 2.7. a) Prevesti dekadne brojeve 43_{10} i 59_{10} u binarni brojni sistem.

b) Predstaviti binarne brojeve koji su dobijeni u prvom zadatku u obliku nepotpunog komplementa, i na osnovu toga izračunati

-43 + 59

pomoću nepotpunog komplementa, u brojnom sistemu sa osnovom $r = 2$, vršeći aritmetičke operacije u $n = 16$ pozicija.

a) Proces konverzije broja 43_{10} u binarni je sledeći:

43 : 2 = 21 , ostatak 1

21:2 = 10 , ostatak 1

10 : 2 = 5 , ostatak 0

5 : 2 = 2 , ostatak 1

2 : 2 = 1 , ostatak 0

1:2 = 0 , ostatak 1

Shodno prethodnom,

$43_{10} = 101011_2$

Proces konverzije broja 59_{10} u binarni je sledeći:

59 : 2 = 29 , ostatak 1

29:2= 14 , ostatak 1

14 : 2 = 7 , ostatak 0

7 : 2 = 3 , ostatak 1

3 : 2 = 1 , ostatak 1

1:2 = 0 , ostatak 1

Odatle se dobija

$59_{10} = 111011_2$

b) Nepotpuni komplement broja -43 u 16 pozicija jednak je (1111111111010100)NK. Slično, nepotpuni komplement binarnog broja 59 jednak je (0000000000111011)NK. Sabiranjem nepotpunih komplementa

$$\begin{array}{r} (1111111111010100)_{NK} \\ +\ (0000000000111011)_{NK} \\ \hline 1 \quad 0000000000001111 \end{array}$$

dobija se zbir $(0000000000010000)_{NK}$ (zbog dodavanja prenosa iz mesta najveće težine u mesto najmanje težine). Prevođenjem ove brojne vrednosti iz potpunog komplementa dobija se binarni broj 10000_2. Odgovarajući dekadni broj je 16_{10}.

Primer 2.8. a) Izračunati 25-134 koristeći sistem sa osnovom $r = 10$, koristeći $m = 8$ pozicija i nepotpuni komplement.

b) Izračunati 25-134 koristeći sistem sa osnovom $r = 10$, koristeći $m = 10$ pozicija i potpuni komplement.

a) $25 = (00000025)_{NK}$

$-134 = (99999865)_{NK}$

Sabiranjem dva nepotpuna komplementa dobija se rezultat u nepotpunom komplementu: $(99999890)_{NK}$. Dekodiranjem nepotpunog komplementa za rezultat se dobija dekadni broj –109.

b) $25 = (0000000025)_{PK}$

$-134 = (9999999866)_{PK}$

Sabiranjem dva potpuna komplementa dobija se rezultat u potpunom komplementu: $(9999999891)_{PK}$. Nepotpuni komplement se dobija dodavanjem broja $(999999999)_{PK}$ (umesto oduzimanja broja 1).

$$\begin{array}{r} (9999999891)_{PK} \\ +\ (9999999999)_{PK} \end{array}$$

Posle odbacivanja prenosa iz mesta najveće težine, dobija se nepotpuni komplement $(9999999890)_{NK}$. Dekodiranjem nepotpunog komplementa za rezultat se dobija dekadni broj –109.

2.4. Zadaci

1. Izračunati dekadnu vrednost 25-134, koristeći potpuni komplement, brojni sistem sa osnovom $r = 2$, koristeći minimalan broj pozicija.

a) Prevesti dekadne brojeve 153_{10} i 77_{10} u binarni brojni sistem.

b) Predstaviti binarne brojeve koji su dobijeni u prvom zadatku u obliku nepotpunog komplementa, i na osnovu toga izračunati

-153 + 77

pomoću nepotpunog komplementa, u brojnom sistemu sa osnovom $r = 2$, vršeći aritmetičke operacije u $m = 16$ pozicija, koristeći nepotpuni komplement.

2. Izračunati 25-134 koristeći sistem sa osnovom $r = 10$, koristeći $m = 8$ pozicija i nepotpuni komplement.

3. Izračunati -25-134 koristeći sistem sa osnovom $r = 10$, koristeći minimalan broj pozicija i potpuni komplement.

4. Izračunati dekadnu vrednost -15-6, koristeći binarni brojni sistem, m=6 pozicija i koristeći potpuni komplement.

5. Sledeći brojevi su dati u raznim osnovama. Osnova je naznačena u zagradi u donjem indeksu nakon broja. Prevesti ih u decimalni sistem: $(2102)_3$, $(2102)_4$, $(310)_7$.

6. Dokazati $(23014)_6 = (3250)_{10}$.

7. Odrediti zapis dekadnog broja 9012 u sistemu sa osnovom 7.

8. Konvertovati sledeće brojeve u decimalni sistem: $(563)_{11}$, $(101101)_2$, $(23022)_4$, $(670)_8$.

9. Izvršiti naznačene konverzije:

$$(12312)_{10} = (?)_3 \quad (9999)_{10} = (?)_9 \quad (32768)_{10} = (?)_2.$$

10. Izvršiti naznačene konverzije: $(1209)_{11} = (?)_8$ $(32334)_5 = (?)_6$.

11. Zapisati sledeće decimalne brojeve u binarnom sistemu: 0, 1, 10, 2, 4, 8, 128, 255, 32768, 1129.

12. Zapisati sledeće binarne brojeve u decimalnom sistemu: 110110, 1010, 0, 1, 101010, 10101.

13. Za broj zapisan u decimalnom sistemu znamo da je paran ako i samo ako se završava cifrom 0, 2, 4, 6, ili 8. Da li postoji slično pravilo koje se odnosi na brojeve zapisane u binarnom sistemu? A da li postoji pravilo koje nam pomaže da brzo utvrdimo da li je broj zapisan u binarnom sistemu deljiv sa 4?

14. Koliko je bitova potrebno da bi se zapisao broj 254? A koliko za broj 1129?

(a) Konvertovati broj $(72364723642703243423045)_8$ iz osnove 8 u osnovu 16.

(b) Konvertovati broj $(f36d2a2c652bf45095ff)_{16}$ iz osnove 16 u osnovu 8.

15. Konvertovati u decimalni sistem sledeće heksadecimalne brojeve: 1, 10, 1a, ff, 10d, fad4.

16. Konvertovati u heksadecimalni sistem sledeće decimalne brojeve: 65312, 4832, 33261, 415, 22, 65535.

17. Konvertovati sledeće binarne brojeve u heksadecimalni sistem: 1101001010111, 1000100010001000, 10010101110.

18. Konvertovati sledeće heksadecimalne brojeve u binarni sistem: adffc1, 19a0c, 101010, ffff.

19. Pretvoriti brojeve 16 i –39 u zapis po 2–komplement (potpuni komplement) kodu na 8 bita. Sabrati dobijene binarne brojeve i proveriti da li se dobija tačan rezultat – 23.

20. Brojevi se u računaru predstavljaju u komplementu dvojke, na širini od 8 bita. Neka se u osmobitnim lokacijama *a* i *b* nalaze brojevi *A* i *B*. Odrediti broj *F* koji sadrži lokacija *f* nakon izvršene operacije oduzimanja *F=A-B* i vrednost indikatora u sledećim slučajevima:

a) A=+1011101, B=+0110011;

b) A=+1011101, B=-0110011;

c) A=-1011101, B=+0100010;

d) A=-1011101, B=+1010001;

e) A=-1011101, B=-0100010.

2.5. Osnovni tipovi podataka

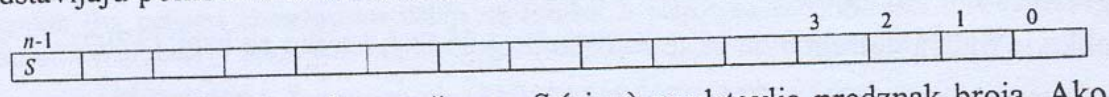
2.5.1. Celobrojni tip podataka (tip INTEGER)

Podaci celobrojnog tipa pripadaju jednom lntervalu celih brojeva koji obuhvata pozitivne i negativne brojeve i koji se obično označava na sledeći način:

$T := \{minint, minint+1, \ldots, -1, 0, 1, \ldots, maxint-1, maxint\}$.

Ovde je najmanji broj označen sa minint, a najveći sa *maxint* (od engl. maximum integer i minimum integer) pri čemu ove veličine nisu fiksne već zavise od implementacije i prema tome variraju od računara do računara.

Od nekoliko načina binarnog kodiranja celih brojeva izdvojićemo metod punog komplementa koji se najčešće sreće u praksi. Kod ovog postupka brojevi se binarno predstavljaju pomoću sledeće *n*-bitne reči:



Bit najstarijeg razreda označen sa *S* (sign) predstavlja predznak broja. Ako je *S*=0 onda je broj nenegativan, a ako je *S*=1 onda je broj negativan. Nula se označava sa nulama u svim bitovima:

0000 0000 0000 0000

Pozitivni brojevi se kodiraju kao standardni binarni brojevi kardinalnog tipa, pa najveći pozitivni broj, uz uslov S=0, ima binarni oblik

0111 1111 1111 1111

t.j. u opštem slučaju n-bitne reči imamo da je $maxint = 2^{n-1} - 1$. Neka je Z dati ceo broj i neka je K binarna kodna reč kojom se predstavlja broj Z. Pozitivni ceo broj +|Z| i negativni ceo broj -|Z| binarno se kodiraju na sledeći način:

+|Z|: $K^+ = (|Z|)_2$

-|Z|: $K^- = (2^n - |Z|)_2 = (2^n - K^+)_2$

Veličina K^- se naziva puni komplement veličine K^+, što važi i u obratnom smeru jer $K^+ + K^- = 2^n$. Na primer, neka je Z=15 (decimalno); u slučaju 16-bitne reči brojevi +15 i -15 se binarno kodiraju na sledeći način:

+15 : K^+ = 0000 0000 0000 1111

-15 : $K^- = 2^{16} - 15 =$ 1 0000 0000 0000 0000

\- 0000 0000 0000 1111

= 1111 1111 1111 0001

Ovaj primer ilustruje sledeći jednostavan postupak kojim se iz K određuje K i obratno:

(1) Odredi se običan komplement polazne veličine tako da se sve jedinice zamene nulama i sve nule jedinicama.

(2) Na tako dobijeni običan komplement doda se 1.

U prethodnom slučaju imamo sledeću konverziju:

Polazni broj (+15)	= 0000 0000 0000 1111
Običan komplement	= 1111 1111 1111 0000
Dodaje se jedan	+1
Potpuni komplement (-15)	= 1111 1111 1111 0001

koja se može primeniti i u obrnutom smeru:

Polazni broj (-15)	= 1111 1111 1111 0001
Običan komplement	= 0000 0000 0000 1110
Dodaje se jedan	+1
Puni komplement (+15)	= 0000 0000 0000 1111

Međutim, prikazani postupak se može i dalje pojednostaviti. Razmotrimo sledeći primer:

Polazni broj (+320)	= 0000 0001 0100 0000
Običan komplement	= 1111 1110 1011 1111

Dodaje se jedan +1

Puni komplement (-320) = 1111 1110 1100 0000

Lako je uočiti da je komplementiranje moguće vršiti neposredno na sledeći način:

(1) U polaznom broju uoči se sa desne strane poslednja jedinica iza koje slede nule i polazni broj se ispred poslednje jedinice podeli na dva dela, levi i desni (pri tome broj desnih nula može da bude i 0!):

Polazni broj =

0000 0001 0	100 0000
levi deo	desni deo

(2) Potpuni komplement dobijamo tako da u levom delu jedinice zamenimo nulama a nule jedinicama, dok desni deo ostavimo neizmenjen:

Potpuni komplement =

1111 1110 1	100 0000
levi deo	desni deo

Ovim postupkom otpada potreba da se bilo šta računa, već se za svaki broj može direktno napisati njegov komplement. Tako na primer imamo sledeće:

+1 = 0000 0000 0000 0001
-1 = 1111 1111 1111 1111
+32767 = 0111 1111 1111 1111 = maxint (za slučaj n =16)
-32767 = 1000 0000 0000 0001 = -maxint

Imajući u vidu poslednji primer postavlja se pitanje koji je to broj koji se kodira sa

1000 0000 0000 0000,

jer očigledno je da ga opisani neposredni algoritam konverzije brojeva preslikava u samoga sebe! Zbog vodeće jedinice ovaj broj je negativan, a iznos mu je -32768 jer ga možemo dobiti oduzimajući 1 od -32767. Komplementiranjem dobijamo

$$1\,0000\,0000\,0000\,0000 - 1000\,0000\,0000\,0000 = 2^{16} - 2^{15} = 2^{15} = 32768\,.$$

Ovo je očito najmanji mogući ceo broj kod 16-bitne mašine (*minint*) jer ako oduzmemo 1 ne dobijamo kao rezultat negativan broj (umesto -32769 rezultat je +32767). Po analogiji sa ovim primerom u opštem slučaju n-bitne mašine važi minint $= -2^{n-1}$, tako da je raspoloživi opseg celih brojeva $-2^{n-1} \le Z \le 2^{n-1} - 1$. Pored toga uočavamo i činjenicu da se zbog nesimetrije veličina minint i maxint postupak neposrednog komplementiranja ne može primeniti u slučaju $Z = minint$.

Komplementiranjem nule dobija se opet nula, t.j. ne postoji (kao kod nekih drugih načina kodiranja) pozitivna i negativna nula, već je nula jedinstvena. Prema tome, u slučaju primene punog komplementa imamo da je ukupna količina n-bitnih binarnih kodnih reči raspoređena na sledeći način:

Ukupan broj pozitivnih brojeva ..... $2^{n-1} - 1$
Jedinstvena nula 1
Ukupan broj negativnih brojeva ..... 2^{n-1}
Ukupan broj n-bitnih kodnih reči ..... 2^n

U nastavku razmotrićemo način obavljanja osnovnih aritmetičkih operacija sabiranja i oduzimanja sa celim brojevima koji su kodirani pomoću punog komplementa. Obavljanje ovih operacija kontroliše se u računaru pomoću dva indikatora: to su indikator prenosa C (carry) i indikator prekoračenja V (overflov). Indikatori C i V se redovno ugrađuju kao dva bita u registru stanja procesora. Ako je rezultat aritmetičke operacije ispravan, onda je V=0, a u slučaju neispravnog rezultata V=1. Prilikom sabiranja dva binarna argumenta generiše se prenos (0 ili 1) iz svakog razreda u naredni razred. Ovo ilustruje sledeći primer sabiranja 16-bitnih brojeva:

Argument 1	1010 0011 1101 0001
Argument 2	0100 1001 1100 1011
Suma	1110 1101 1001 1100
Prenos u naredni razred :	0000 0011 1100 0011

Prenos iz zadnjeg razreda = C ↙ ↘ P = prenos iz predzadnjeg razreda.

Pored opisanih operacija sabiranja i deljenja sa celobrojnim podacima se obavljaju i druge uobičajene aritmetičke operacije. Operacija množenja obavlja se uz sadejstvo C i V indikatora na sličan način kao i sabiranje. Interesantno je primetiti da se množenje binarnih brojeva svodi samo na pomeranje i sabiranje (t.j. nije potrebno poznavati tablicu množenja, jer se množi samo sa ciframa 0 i 1). To ilustruje sledeći primer:

Dekadno množenje:	Binarno množenje:
3*5 = 15	11 * 101 11 00 11 1111

Pored množenja važna je i operacija celobrojnog deljenja koje se vrši sa odsecanjem. Ako su a i b promenljive tipa INTEGER onda ćemo podrazumevati da važi

$$a/b \equiv \lfloor a/b \rfloor$$

što znači da je 3/2 = 1, 8/3 = 2, 999/1000 = 0, itd. Primenom celobrojnog deljenja može se lako realizovati i operacija moduo:

$$a \bmod b = a-(a/b)*b, \quad a-(a/b)*b \geq 0$$
$$= a-(a/b)*b+b, \quad a-(a/b)*b < 0 .$$

Na primer, 8 mod 3 = 2, (-8)mod 3 = -8+6+3 = 9-8 = 1,(-8) mod 6 = 12-8 = 4.

Skoro bez izuzetaka, u svim programskim jezicima postoji jedan ili više tipova podataka za rad sa celim brojevima koji obično u nazivu imaju reč INTEGER.

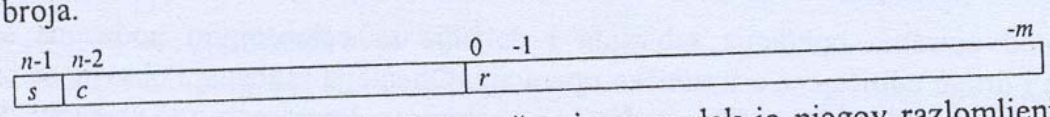
Obično postoji osnovni (standardni) tip INTEGER ili int, koji se zavisno od realizacije odnosi na određeni opseg celih brojeva. Nekad je to opseg koji odgovara formatu jedne polu-reči ili formatu jedne reči. U odnosu na ovaj osnovni celobrojni tip često postoji mogućnost definisanja i drugih celobrojnih tipova koji se odnose na neki kraći ili prošireni format.

2.5.2. Predstavljanje realnih brojeva

Realni brojevi se definišu kao tip podataka real (ponekad se naziva i float). Realni brojevi se mogu predstaviti u obliku fiksne (nepokretne) tačke ili u obliku pokretne tačke (floating point).

Fiksna ili nepokretna tačka

Realni brojevi su podskup skupa realnih brojeva. Za celobrojni i razlomljeni deo realnog broja odvaja se konstantan broj binarnih mesta (bitova). Neka se n bitova koristi za reprezentaciju celobrojnog dela, a m bitova za reprezentaciju razlomljenog dela broja.



Celobrojni deo realnog broja označen je sa c, dok je njegov razlomljeni deo označen sa r. Bit označen sa s predstavlja bit znaka, i ima vrednost 1 za negativne brojeve, a vrednost 0 za pozitivne. Maksimalna vrednost celobrojnog dela realnog broja je $C^{max} = 2^{n-1}$, dok je minimalna vrednost razlomljenog dela jednaka $r_{min} = 2^{-m}$. Brojeve manje od r_{min} računar vidi kao 0, i nazivaju se *mašinska nula*.

Primer 2.9. a) Izračunati 732.029 - 3185.273, koristeći sistem sa osnovom $r = 10$, $m = 8$ celobrojnih mesta, $n = 3$ razlomljenih mesta i potpuni komplement.

$$732.029=(00000732.029)_{PK}$$
$$-3185.273=(99996814.727)_{PK}.$$

Dobija se rezultat u potpunom komplementu $(99997546.756)_{PK}$. Oduzimanje jedinice svodi se na dodavanje njenog potpunog komplementa $(99999999.999)_{PK}$. Dobija se nepotpuni komplement $(99997546.755)_{NK}$. Dekodiranjem nepotpunog komplementa dobija se rezultat –2453.244.

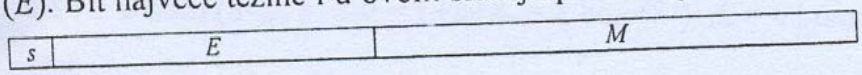
b) Izračunati 732.029 - 3185.273, koristeći sistem sa osnovom $r = 10$, $m = 8$ celobrojnih mesta, $n = 5$ razlomljenih mesta i nepotpuni komplement.

$$732.029=(00000732.02900)_{NK}$$
$$-3185.273=(99996814.72699)_{NK}.$$

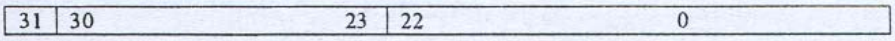
Dobija se rezultat u nepotpunom komplementu $(99997546.75599)_{NK}$. Dekodiranjem nepotpunog komplementa dobija se rezultat –2453.244.

Pokretna tačka

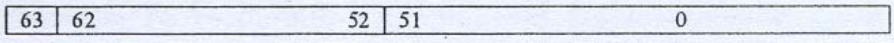
Numeričke vrednosti ovog tipa se predstavljaju pomoću mantise (M) i eksponenta (E). Bit najveće težine i u ovom slučaju predstavlja bit znaka.



Obična tačnost:



Dvostruka tačnost:



Vrednost broja u pokretnom zarezu jednaka je $V = (-1)^s \times M \times R^E$, gde je:

s – znak; M – mantisa, E – eksponent; R – osnova brojnog sistema.

Nedostatak reprezentacije realnih brojeva u obliku pokretne tačke je što se svaki realan broj može predstaviti na više načina. Poznato je nekoliko formata realnih brojeva. Normalizovani zapis ispunjava uslov

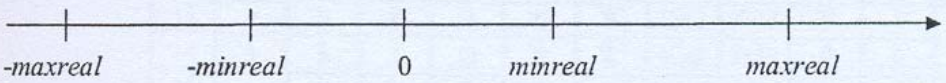
$$1 > M \geq 1/R.$$

Primer 2.10. Broj 137.145 se može predstaviti u obliku 0.00137145×10^3 ili u obliku 13714.5×10^{-2}. Ovi zapisi se nazivaju *nenormalizovani. Normalizovani zapis* ovog realnog broja je 0.137145 E+2 = 0.137145×10^2.

Realni tip (float ili real)

Promenljive ovih tipova uzimaju za svoje vrednosti podskupove skupa realnih brojeva. U programskim jezicima postoji više vrsta podataka realnog tipa, koji se razlikuju po tačnosti predstavljanja podataka.

Realni tip podataka obuhvata jedan konačan podskup racionalnih brojeva ograničene veličine i tačnosti. Naravno, može se odmah postaviti pitanje zbog čega se koriste nazivi "realni tip" i "realni broj" za nešto što u opštem slučaju nije u stanju da obuhvati ni iracionalne brojeve ni beskonačne periodične racionalne brojeve. Ipak, to je terminologija koja je prihvaćena u praksi i opravdava se time što je REAL tip podataka koji se najbliže približava pojrau realnog broja. Sa druge strane, lako je razumeti da su sve memorijske lokacije konačne dužine, pa stoga i brojni podaci koji se u njih smeštaju moraju biti konačne dužine i tako po prirodi stvari otpadaju iracionalni brojevi. Potrebe prakse nisu na ovaj način ni malo ugrožene jer se dovoljna tačnost rezultata može postići i sa veličinama konačne dužine. Tip *REAL* najčešće obuhvata brojne vrednosti iz sledećih podintervala brojne ose:



Ovde *minreal* označava najmanju apsolutnu vrednost, veću od nule, koja se može predstaviti na raćunaru, a *maxreal* predstavlja najveću apsolutnu vrednost.

Realni brojevi iz intervala (*-minreal*/2, *minreal*/2) se zaokružuju i prikazuju kao 0, realni brojevi iz intervala (-∞, *-maxreal*) i (*maxreal*, +∞) ne mogu se predstaviti u memoriji računara, a -∞ i +∞ se kodiraju specijalnim kodovima.

Osnovna ideja računarskog predstavljanja realnih brojeva sastoji se u korišćenju sledeće eksponencijalne predstave: $R = (-1)^S M 2^E$, $s \in \{0,1\}$, $E_{\min} \leq E \leq E_{\max}$

Pri tome se broj R predstavlja pomoću tri vrednosti: s, M i E. Jednobitna vrednost s predstavlja predznak broja, M je mantisa i E je eksponent. Naravno, M i E se kodiraju binarno. Ako mantisa uvek ima standardni oblik $0.1bb. ..b$, gde $b \in \{0,1\}$ i $1/2 \leq M < 1$ ili drugi standardni oblik $1.bb. ..b$ (t.j. $1 \leq M < 2$), onda se kaže da je mantisa normalizovana.

2.6. Predstavljanje nenumeričkih brojeva

Kao nenumerički podaci najpoznatiji su znakovne vrednosti (character) i logičke vrednosti (logical ili boolean).

2.6.1. Predstavljanje znakovnih vrednosti

U praksi se najviše koriste dva standardna koda za predstavljanje znakovnih podataka: EBCDIC (*Extended Binary Coded Decimal Interchange Code*) i ASCII (*American Standard Code for Information Interchange*), koji predstavlja američku verziju međunarodnog koda ISO7. ASCII skup znakova čine:

Velika i mala slova engleske azbuke: {a,b, ..., z,A,B,C, ... Z}

Dekadne cifre: {0,1,2,3,4,5,6,7,8,9}.

Operacioni i specijalni znaci: {+,-*,/,>,=,<,!,blanko,|,",#,$,%,^,&,',@,.,:,,,,;?,{,},[,],~, ↑,↓,→}.

ASCII tabela - American Standard Code for Information Interchange

Ascii tabela				Bits	0	16	32	48	64	80	96	112
				b7	0	0	0	0	0	0	0	
				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
				b4	0	1	0	1	0	1	0	1
	b_3	b_2	b_1	b_0								
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	P
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	V
7	1	1	1	1	BEL	ETB	'	7	G	W	g	W
8	1	0	0	0	BS	CAN	(	8	H	X	h	X
9	1	0	0	1	HT	EM	)	9	1	Y	i	y
10	1	0	1	0	LF	SUB	*		J	z	j	z
11	1	0	1	1	VT	ESC	+	/	K	[	k	{
12	1	1	0	0	FF	FS	,	<	L	\	l	|
13	1	1	0	1	CR	GS	-	=	M	]	m	}
14	1	1	1	0	SO	RS	.	>	N	^	n	~
15					SI	US	/	?	O	-	o	

Kontrolni znaci: DEL (brisanje), STX (start teksta), ETX (kraj teksta), ACK (potvrđivanje), HT (horizontalni tabulator), VT (vertikalni tabulator), LF (kraj reda), CR (prelaz u novi red), NAK (negativna potvrda), SYN (sinhronizacija pri prenosu), ETB (kraj prenosa), FS (separator datoteke), GS (separator grupe) i RS (separator zapisa).

Na primer, znak *A* predstavlja se ASCII kodom $01000001_2 = 65_{10}$.

Današnjom internacionalizacijom upotrebe računara, problem predstavljanja raznih lokalnih znakova (naša ćirilica i latinica, ruska i grčka slova, kao i dalekoistočni simboli) rešen je uvođenjem UNICODE rasporeda koji ima 16 bita i njime je moguće predstaviti $2^{16} = 65536$ karaktera. Trenutno je ovo dovoljno za predstavljanje svih lokalnih karaktera u oblastima na planeti gde se koristi informaciona tehnologija.

U sledećoj tabeli date su izmene koje važe u našoj verziji ISO 7 koda (YUASCII).

YU ASCII Tabela					0	16	32	48	64	80	96	112
					0	0	0	0	0	0	0	0
					0	0	0	0	1	1	1	1
					0	0	1	1	0	0	1	1
					0	1	0	1	0	1	0	1
	b_3	b_2	b_1	b_0								
0	0	0	0	0	NUL	DLE	SP	0	Ž	P	ž	P
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	0C2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	1	1	0	0	EOT	DC4	$	4	D	T	d	t
5	1	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	1	1	1	0	ACK	SYN	&	6	F	V	f	V
7	1	1	1	1	BEL	ETB	'	7	G	w	g	w
8	0	0	0	0	BS	CAN	(	8	H	X	h	X
9	0	0	0	1	HT	EM	)	9	I	Y	i	y
10	0	0	1	0	LF	SUB	*	:	J	z	j	z
11	0	0	1	1	VT	ESC	+	;	K	š	k	š
12	1	1	0	0	FF	FS	,	<	L	Đ	l	đ
13	1	1	0	1	CR	GS	-	=	M	Ć	m	ć
14	1	1	1	0	SO	RS	.	>	N	Č	n	č
15	1	1	1	1	SI	US	/	?	O	-	0	

EBCDIC skup znakova

Kao i u prethodnoj grupi ASCII znakova od 1 do 3, plus kontrolni znaci koji izvršavaju kontrolne funkcije (razlikuju se imena u odnosu na 4 grupu znakova). Postoje skupovi znakova koji se koriste za specijalne aplikacije. Mnogi računarski grafički sistemi koriste skupove znakova za manipulaciju tačkama i linijama na katodnoj cevi. Tako se koriste specijalni znaci za rotaciju, translaciju, povećanje ili smanjenje slike na ekranu. Neki znaci, kao APL, pored skupa EBCDIC i ASCII znakova, uključuju i neka grčka slova i matematičke simbole.

EBCDIC tabela – Extended Binary Coded Decimal Interchange Code

EBCDIC tabela			b_7	0	0	0	0	1	1	1	1	1	1	1	1
			b_6	1	1	1	1	0	0	0	0	1	1	1	1
			b_5	0	0	1	1	0	0	1	1	0	0	1	1
			b_4	0	1	0	1	0	1	0	1	0	1	0	1
b_3	b_2	b_1	b_0												
0	0	0	0												
0	0	0	1					a	j			A	J		1
0	0	1	0					b	k	s		B	K	S	2
0	0	1	1					c	I	t		c	L	T	3
0	1	0	0					d	m	u		D	M	U	4
0	1	0	1					e	n	V		E	N	V	5
0	1	1	0					f	0	w		F	0	W	6
0	1	1	1					g	P	X		G	P	X	7
1	0	0	0					h	q	y		H	Q	Y	8
1	0	0	1					i	r	z		I	R	Z	9
1	0	1	0		!		:								
1	0	1	1	.	$	,	#								
1	1	0	0	<	*	%	@								
1	1	0	1	(	)	-	‘								
1	1	1	0	+	;	>	=								
1	1	1	1		¬	?	“								

2.6.2. Predstavljanje logičkih vrednosti

Logički podatak je struktura podataka koja može uzeti vrednosti istina (true) i laž (false). Ovakvi podaci se obično binarno predstavljaju tako što memorijska reč popunjena jedinicama označava false, a svi drugi sadržaji true. Kao što postoje operacije za aritmetičke podatke (kao sabiranje, oduzimanje, množenje, itd.), postoje operacije za logičke podatke. Tri najčešće logičke operacije su: presek, unija i negacija, i najčešće se prikazuju operatorima $\wedge$, $\vee$, $\sim$, respektivno.

Ako su X i Y logičke promenljive, tada se operatorima $\wedge$, $\vee$, $\neg$ definišu logičke operacije:

$\wedge$ Rezultat $X \wedge Y$ je istinit (*true*) ako i samo ako X i Y imaju vrednost istina (*true*); u suprotnom rezultat je neistinit (*false*).

$\vee$ Rezultat $X \vee Y$ je neistinit (false) ako X i Y imaju vrednost neistinitu (*false*); u suprotnom rezultat je istinit (*true*).

$\neg$ Rezultat $\neg X$ je istinit (true) ako je X neistinito (*false*) i rezultat $\neg X$ je neistinit (*false*) ako X ima vrednost istinitu (*true*).

Izrazi sa logičkim operandima nadgrađuju se izrazima poređenja, korišćenjem relacionih operatora <, <, =, *, >, > u njima. Logičke promenljive koriste se vrlo često da prikažu vrednosti kompleksnih logičkih izraza, npr. $(a < b \wedge c < d) \vee (a < c \wedge d < b)$. Takođe, logičke promenljive mogu se koristiti u zadavanju završnih uslova unutar petlje u programskim jezicima.

Definicija logičkih operatora $\wedge$, $\vee$, $\neg$ prikazana je u sledećoj tabeli:

X	Y	$X \wedge Y$	$X \vee Y$	$\neg X$
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Forma memorijskog prikaza logičkih vrednosti zavisi od prevodioca, tj. kompajlera ili interpretera, i kreće se od jednog bita do pune mašinske reči.

Logički tip podataka

Logički tipovi podataka postoje kao osnovni tipovi podataka u svim novijim jezicima. Obično nose naziv LOGICAL (FORTRAN) ili BOOLEAN (Pascal, Ada). Obuhvataju samo dve vrednosti *true* i *false*, nad kojima su definisane osnovne logičke operacije not, and, or i xor.

Takođe, važi i uređenost skupa vrednosti ovog tipa tako da je *false* < *true*.

Logički tip podataka je prosti tip podataka koji obuhvata samo dve vrednosti: $T:=\{v_1,v_2\}$. Veličina v_1 označava istinitosnu vrednost neistinitog iskaza, dok je v_2 istinitosna vrednost istinitog iskaza. U programskim jezicima se koriste dva načina označavanja istinitosnih vrednosti logičkih iskaza. U prvom slučaju se teži da se onemogući da se sa v_1 i v_2 obavljaju aritmetičke operacije i tada se obično uvode sledeće specijalne simboličke oznake:

$$v_1 = false \text{ (ili } v_1 = F), \quad v_2 = true \text{ (ili } v_2 = T).$$

Drugi način, koji redovno ima za cilj da omogući aritmetičke operacije logičkim veličinama, zasniva se na numeričkim vrednostima

$$v_1 = 0 \ , \quad v_2 = 1.$$

Ako želimo da promenljive *a*, *b*, *c* budu u nekom programu definisane kao promenljive logičkog tipa onda to specificiramo iskazom

DEFINE *a,b,c* : LOGICAL .

Podrazumeva se da se na sve konstante i promenljive logičkog tipa mogu primenjivati osnovne logičke operacije negacija (not), konjunkcija (and) i disjunkcija (or), za koje važi

not (false) = true , not(true) = false

false and false=false, false and true=false, true and false=false, true and true=true

false or false=false, false or true=true, true or false=true, true or true=true.

Pomoću ovih osnovnih operacija lako se realizuju složenije operacije ekskluzivne disjunkcije, ekvivalencije, Pirsova funkcija, Šeferova funkcija i implikacija:

$x \oplus y$ = eor(x,y) = (not(x) and y) or (x and not(y))

$x \sim y$ = equ(x,y) = not(eor(x,y)) = (x and y) or (not(x) and not(y))

$x \downarrow y$ = nor(x,y) = not(x or y)

x | y = nand(x,y)= not(x and y)

x →y = imp(x,y)= not(x and not(y)) = not(x) or y.

Podrazumeva se da primenom relacionih operatora =, ≠, >, <, ≥ i ≤ dobijaju veličine logičkog tipa. Na primer, $(x = y) \in \{true\ false\}$, pri čemu relacija $x = y$ uzima vrednost true ako se u lokacijama x i y nalaze iste vrednosti (x i y mogu biti bilo kog, ali istog tipa), a false ako se ustanovi da se sadržaji lokacija x i y razlikuju. Ponekad se desi da programski jezik ne poseduje logički tip promenljivih, ili da je pogodno da se logičke operacije primenjuju na numeričke podatke. U tom slučaju mogu se negacija, konjunkcija i disjunkcija izraziti pomoću aritmetičkih operacija na sledeći način:

$$\bar{x} = 1 - x, \quad x \wedge y = x \cdot y, \quad x \vee y = x + y - x \cdot y, \quad x, y \in \{0,1\}.$$

U programiranju se često raspolaže samo sa ograničenim skupom simbola, pa je u takvim slučajevima uobičajeno da se relacioni operatori ≠ , ≥ i ≤ označavaju respektivno sa <>, >= i <=.

Izrazi u kojima se primenjuju logičke promenljive i konstante nazivaju se logički izrazi. Ako se logičke konstante označavaju sa false i true onda podrazumevamo da svi izrazi moraju biti sačinjeni striktno od logičkih veličina. Ha primer, izraz z := (x>0) and ((y=1) or (y<0)) je korektan pri čemu se podrazumeva da su x i y celobrojne ili realne veličine, a z je veličina logičkog tipa. Podrazumeva se i da su neispravni mešoviti izrazi u kojima se koriste veličine true i false pomešane sa numeričkim konstantama i aritmetičkim operacijama. Na pr., nije definisan izraz 4*false + 2*true + true, ali izraz 4*(x>0) + 2*(y>0) + (z>0) , koji je takođe besmislen kod logičkih konstanti false i true, u slučaju numeričkog kodiranja T={0,1} redovno ima i smisla i upotrebnu vrednost kao generator veličina 0,1,2,3,4,5,6,7.

3. BULOVA I PREKIDAČKA ALGEBRA, LOGIČKA I PREKIDAČKA KOLA

Razvoj digitalnih sistema, uključujući i računarske sisteme za matematičku osnovu koristi Bulovu (Boolean) algebru. Zbog važnosti prekidačke algebre kod projektovanja ne samo računara, nego i komunikacionih sistema, sistema upravljanja i bilo kojeg drugog sistema koji zahteva ili koristi digitalnu tehnologiju, veoma je važno da se razume značaj ove algebre.

U ovoj glavi definisaćemo osnovne pojmove koji se odnose na Bulovu algebru i njen podskup -prekidačku algebru.

3.1. Osnovni postulati i teoreme

Algebra se definiše skupom iskaza koji se prihvataju kao činjenice. Ove iskaze nazivamo *aksiomima* ili *polsulatima* algebre. Jedan od ciljeva matematičara je da

izvrše redukciju broja potrebnih postulata kojim se definiše algebra na minimalan konzistentan skup.

Šestorka $(B, +, \cdot, \bar{}, 0, 1)$, gde je:

- B skup elemenata ili konstanti algebre,
- $+$ i $\cdot$ dva binarna operatora a simbol $\bar{}$ unarni operator,
- 0 i 1 su različiti elementi skupa B,

naziva se *Bulovom algebrom* ako su ispunjene sledeće aksiome:

1. *Zatvorenost:* za svaki element a i b iz skupa B važi

   (B1) $a + b$ je element B, i
   (B2) $a \cdot b$ je element B.

2. *Postojanje neutralnih elemenata za operacije* $+$ *i* $\cdot$

   (B3) Za svako $a \in B$ važi $0 + a = a + 0 = a$, i

   (B4) Za svako $a \in B$ važi $1 \cdot a = a \cdot 1 = a$.

3. *Komutativnost i asocijativnost:* za sve elemente a, b i c u skupu B važi

   (B5) $a + b = b + a$, (B6) $(a + b) + c = a + (b + c)$
   (B7) $a \cdot b = b \cdot a$. (B8) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

4. *Distributivnost* za sve elemente a, b i c u skupu B važi

   (B9) $a \cdot (b + c) = a \cdot b + a \cdot c$
   (B10) $a + (b \cdot c) = (a + b) \cdot (a + c)$.

5. *Postojanje inverznog elementa:* Za svaki element a iz skupa B postoji u B element $\bar{a}$, takav da važi

   (B11) $a + \bar{a} = 1$
   (B12) $a \cdot \bar{a} = 0$.

6. U skupu B postoje najmanje dva različita elementa, tj. $0 \neq 1$.

Termini *binarni operator* i *unarni operator odnose* se na broj argumenata koji su uključeni u operaciju: dva ili jedan, respektivno.

Bulov izraz se rekurzivno može definisati na sledeći način:

- 0 i 1 jesu Bulovi izrazi
- x_i je Bulov izraz, za svako $x_i \in B$.
- ako su E_1, E_2 Bulovi izrazi, tada su (E_1), $E_1 + E_2$, $E_1 \cdot E_2$, $\overline{E_1}$ takođe Bulovi izrazi.

1. $x_1 + x_2 x_3 + x_4 x_5 \overline{x_2} + \overline{(x_1 + x_2)}$ je validan Bulov izraz.

2. $(((x_1) + (x_2 \overline{x_3}))$ nije validan Bulov izraz, zato što nije isti broj otvorenih i zatvorenih zagrada.

3.2. Neki primeri Bulovih algebri

U ovom poglavlju daćemo nekoliko primera Bulovih algebri.

Primer 3.1. (*Algebra partitivnog skupa*) Neka je X bilo koji skup i $P(X)$ njegov odgovarajući partitivni skup (skup svih podskupova skupa X). Struktura $(P(X), \cup, \cap, \bar{}, \emptyset, X)$ je Bulova algebra. Pri tome operacija $\bar{a}$ predstavlja komplement od a u odnosu na X, tj $\bar{a} = X \setminus \{a\}$, operacija $\cup$ predstavlja uniju dva skupa, operacija $\cap$ predstavlja presek skupova, dok $\emptyset$ predstavlja oznaku za prazan skup. Ovako definisane operacije zadovoljavaju aksiome Bulove algebre.

Primer 3.2. (*Dvoelementna Bulova algebra*) Ako je $X = \{x\}$ jednoelementni skup, skup $P(X)$ se redukuje na skup $\{0, 1\}$, gde je $0 = \emptyset$ i $1 = X$. Ovakva Bulova algebra naziva se dvoelementna Bulova algebra i dogovorno se označava sa 2. Operacije $\cup$ i $\cap$ označavaju se sa $+$ i $\bullet$, respektivno, i date su u sledećoj tabeli:

x	y	$x+y$	$x \bullet y$	$\bar{x}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Ako 0 i 1 identifikujemo kao"netačno" i "tačno" respektivno, tada Bulove operacije mogu biti konjukcija, disjunkcija i negacija.

Primer 3.3. *Prekidačka algebra* je Bulova algebra kod koje je broj elemenata u skupu B jednak 2. Binarni operatori koji se predstavljaju znacima $+$ i $\bullet$ nazivaju se OR (ili $\vee$) i AND (ili $\wedge$), tj. ILI i I, respektivno, dok se unarni operator koji se predstavlja znakom $\bar{}$ naziva NOT (ili $\neg$), tj. NE, ill operator *komplement.* Napomenimo još da se najčešće proizvodi tipa $a \bullet b$ pišu kao ab, izostavljajući ali podrazumevajući operator $\bullet$. Ako je B prekidačka algebra, očigledno je $B = \{0,1\} = \{\perp, T\}$. Operacije definisane su tabelom

x	y	$x \vee y$	$x \wedge y$	$\neg x$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

U sledećoj tabeli je prikazana veza između Bulove algebre, algebre partitivnog skupa i prekidačke algebre.

Bulova algebra	Podskupovi skupa U	Prekidačka algrbra
$x + y$	$X \cup Y$	$x \vee y$
xy	$X \cap Y$	$x \wedge y$
$\bar{x}$	X^c	$\neg x$
0	$\varnothing$	0
1	U	1
=	=	ekvivalencija

To znači da se iz opšte Bulove algebre dobijaju njeni parcijalni slučajevi korišćenjem sledećih smena:

1. u skupovima: $(+ \to \cup), (\bullet \to \cap), (a \to A^c = \bar{A}), (0 \to \{\,\} = \emptyset), (1 \to U)$
2. u matematičkoj logici: $(+ \to \vee), (\bullet \to \wedge), (\bar{a} \to \bar{a}), (0 \to F), (1 \to T)$
3. prekidačka logika: Bulova algebra na skupu B=$\{0, 1\}$

3.3. Osnovna tvrđenja Bulove algebre

Definicija 3.3.1. Ako je A Bulov izraz, pod *dualnim* Bulovim izrazom A^* podrazumevaćemo Bulov izraz koji se dobija kada se u izrazu A operacije + zamene operacijama •, operacije • zamene operacijama +, a konstante zamene njihovim komplementima.

Sledeća teorema, koja se naziva *princip dualnosti*, omogućava da se bilo koje tvrđenje iz Bulove algebre automatski primeni i na njegov dualni izraz.

Teorema 3.3.1. (Princip dualnosti.) Ako iskaz A zadovoljava aksiome Bulove algebre, tada i njegov dualni iskaz A^* takođe zadovoljava aksiome Bulove algebre.

Pokazaćemo nekoliko osnovnih teorema Bulove algebre:

Teorema 3.3.2. Zakon idempotencije (zakon nevaženja stepenovanja)

$\forall a \in B \Rightarrow$ (i) $a + a = a$

(ii) $a \cdot a = a$

Dokaz:

(i) $a+a = (a+a)\cdot 1$ prema (B4)

$= (a+a)\cdot(a+\bar{a})$ prema (B11)

$= a + a\cdot\bar{a}$ prema (B10)

$= a + 0$ prema (B12)

$= a$ prema (B3)

(ii) $a\cdot a = a\cdot a + 0$ prema (B3)

$= a\cdot a + a\cdot\bar{a}$ prema (B12)

$= a\cdot(a+\bar{a})$ prema (B9)

$= a\cdot 1$ prema (B11)

$= a$ prema (B4)

Teorema 3.3.3. U Bulovoj algebri, komplement elementa a (u oznaci $\bar{a}$) je jedinstven.

Dokaz: Pretpostavićemo suprotno: da ipak postoji još jedan element $b \neq \bar{a}$, koji predstavlja inverzni element elementa a i koji, prema tome, zadovoljava $a+b = 1$ i $a\cdot b = 0$.

$b = b\cdot 1$ (B4)

$= b\cdot(a+\bar{a})$ (B11)

$= b\cdot a + b\cdot\bar{a}$ (B9)

$= a\cdot b + \bar{a}\cdot b$ (B7)

$= 0 + \bar{a} \cdot b$ po pretpostavci

$= a \cdot \bar{a} + \bar{a} \cdot b$ (B12)

$= \bar{a} \cdot a + \bar{a} \cdot b$ (B7)

$= \bar{a}\,(a + b)$ (B9)

$= \bar{a} \cdot 1$ po pretpostavci

$= \bar{a}$ (B4)

Dakle, dobijena je protivurečnost, pa je inverzni element jedinstven.

Teorema 3.3.4. Zakon involucije operacije negacije, tj. zakon dvojne negacije.

$\forall a \in B \Rightarrow \bar{\bar{a}} = a.$

Dokaz:

Na osnovu aksioma o inverznom elementu, dobijamo:

$\bar{a} + a = a + \bar{a} = 1$

$\bar{a} \cdot a = a \cdot \bar{a} = 0$

Ako uzmemo da je $x = \bar{a}$, tada je, prema aksiomu o inverznom elementu, $a = \bar{x}$, a otuda dobijamo $a = \bar{\bar{a}}$.

Teorema 3.3.5.

$\forall a \in B \Rightarrow$ (i) $a + 1 = 1$

(ii) $a \cdot 0 = 0$

Dokaz.

(i) $a + 1 = (a+1) \cdot 1$ (B4)

$a + 1 = (a+1) \cdot (a + \bar{a})$ (B11)

$a + 1 = a + 1 \cdot \bar{a}$ (B10)

$a + 1 = a + \bar{a}$ (B4)

$a + 1 = 1$ (B11)

(ii) Prema principu dualnosti, ili:

$a \cdot 0 = (a \cdot 0) + 0$ (B3)

$a \cdot 0 = (a \cdot 0) + (a \cdot \bar{a})$ (B12)

$a \cdot 0 = a \cdot (0 + \bar{a})$ (B9)

$a \cdot 0 = a \cdot \bar{a}$ (B3)

$a \cdot 0 = 0$ (B12)

Teorema 3.3.6. De Morganova Teorema.

$\forall a, b \in B \Rightarrow$ (i) $\overline{a+b} = \bar{a} \cdot \bar{b}$

(ii) $\overline{a \cdot b} = \bar{a} + \bar{b}$

Dokaz:

(i) Pokažimo najpre

$$a+b+\bar{a}\bullet\bar{b}=1 \qquad (3.1)$$

$$(a+b)\bullet\bar{a}\bullet\bar{b}=0 \qquad (3.2)$$

Zaista,

$a+b+\bar{a}\bullet\bar{b}=(a+b+\bar{a})\bullet(a+b+\bar{b})$ (B10)

$=(b+1)\bullet(a+1)$ (B11)

$=1\bullet 1$ (Teorema 3.1.5.)

$=1$

$(a+b)\bullet\bar{a}\bullet\bar{b}=\bar{a}\bullet\bar{b}\bullet a+\bar{a}\bullet\bar{b}\bullet b$ (B7) i (B9)

$=0\bullet\bar{b}+\bar{a}\bullet 0$ (B7) i (B12)

$=0+0=0$ (Teorema 3.1.6.)

Kako važi (3.1) i (3.2), koristeći aksiom o inverznom elementu, jasno je da važi relacija (i).

(ii) Prema principu dualnosti.

Teorema 3.3.7. Opšta De Morganova teorema.

$\forall x_i\in B,\ i=1,\ldots,n\Rightarrow$ (i) $\overline{x_1+x_2+\cdots+x_n}=\bar{x}_1\bullet\bar{x}_2\bullet\ldots\bullet\bar{x}_n$

(ii) $\overline{x_1\cdot x_2\cdot\cdots\cdot x_n}=\bar{x}_1+\bar{x}_2+\ldots+\bar{x}_n$

Dokaz: Tvrđenje se dokazuje principom matematičke indukcije.

(i) Slučaj $n=2$ je dokazan u prethodnoj teoremi.

Pretpostavimo da teorema važi za slučaj $n=m>2$. Dokazaćemo da je tada tačna i za $n=m+1$. Zaista, neka je $z=x_1+x_2+\cdots+x_m$. Prema prethodnoj teoremi, dobijamo $\overline{z+x_{m+1}}=\bar{z}\cdot\bar{x}_{m+1}$.

Zamenom odgovarajuće vrednosti za promenljivu z dobija se željeni rezultat.

(ii) Ovaj deo teoreme se može dokazati metodom indukcije. Za slučaj ($n=2$) može se koristiti dokaz prethodne teoreme ili tabela

x_1	x_2	$x_1\bullet x_2$	$\overline{x_1\bullet x_2}$	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_1+\bar{x}_2$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Znači, za $n=2$, važi

$$\overline{x_1\cdot x_2}=\bar{x}_1+\bar{x}_2$$

Neka teorema važi za $n=m$, tj.

$$\overline{x_1 \cdot x_2 \cdot \ldots \cdot x_m} = \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_m$$

gde je $m>2$. Uočimo izraz

$$z = x_1 \bullet x_2 \bullet \ldots \bullet x_m$$

Zamenom vrednosti z u izrazu

$$\overline{z \cdot x_{m+1}} = \bar{z} + \overline{x_{m+1}}$$

i primenom induktivne hopoteze, dobija se

$$\overline{x_1 \cdot x_2 \cdot \cdots \cdot x_{m+1}} = \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_{m+1}$$

čime je teorema dokazana.

De Morganova teorema igra veoma značajnu ulogu kod projektovanja hardvera računara. Može se upotrebiti u simplifikaciji logičkih izraza.

Primer 3.4. Koristeći De Morganovu teoremu i neke od rezultata pojednostaviti sledeće iskaze:

a) $\overline{\bar{x} + y(\bar{z} + w)}$,

b) $\overline{[\bar{x}(\overline{y+z})](y + w\bar{z})(x+z)}$.

Odgovor:

a) $\overline{\bar{x} + y(\bar{z} + w)} = \bar{\bar{x}}\,\overline{y(\bar{z} + w)}$

$= \bar{\bar{x}}(\bar{y} + \overline{\bar{z} + w})$

$= \bar{\bar{x}}(\bar{y} + \bar{\bar{z}}\,\bar{w})$

$= x(\bar{y} + z\bar{w})$

$= x\bar{y} + xz\bar{w}$

b) $\overline{[\bar{x}(\overline{y+z})](y + w\bar{z})(x+z)} = \overline{\bar{x}(\overline{y+z})} + \overline{y + w\bar{z}} + \overline{x+z}$

$= \bar{\bar{x}} + \overline{\overline{y+z}} + \bar{y}\,\overline{w\bar{z}} + \bar{x}\bar{z}$

$= x + y + z + \bar{y}(\bar{w} + \bar{\bar{z}}) + \bar{x}\bar{z}$

$= x + y + z + \bar{y}(\bar{w} + z) + \bar{x}\bar{z}$

Teorema 3.3.8. Zakon apsorpcije.

$\forall a, b \in B \Rightarrow$ (i) $a + ab = a$

(ii) $a(a + b) = a$

Dokaz:

(i) $a+ab = a \bullet 1 + ab$ (B4)

$= a \bullet (1 + b)$ (B9)

$= a \bullet 1$ ($1+b = b$, Teorema 3.3.5)

$= a$

(ii) Prema principu dualnosti, ili

$a\bullet(a+b)=(a+0)\bullet(a+b)=a+0\bullet b=a+0=a.$

Teorema 3.3.9. Zakon sažimanja.

$\forall a,b\in B\Rightarrow$ (i) $ab+a\bar{b}=a$

(ii) $(a+b)(a+\bar{b})=a$

Dokaz: Dokaz se izvodi direktnom primenom aksiome o distributivnosti i aksiome o inverznom elementu.

(i) $ab+a\bar{b}=a(b+\bar{b})=a\cdot 1=a$

(ii) $(a+b)(a+\bar{b})=a+b\cdot\bar{b}=a+0=a.$

Na drugi način, ovo tvrđenje se može dokazati množenjem:

$(a+b)(a+\bar{b})=a\cdot a+a\cdot\bar{b}+a\cdot b+b\cdot\bar{b}$

$=a+a\cdot\bar{b}+a\cdot b=a+a\cdot(b+\bar{b})=a+a\cdot 1=a+a=a.$

Teorema 3.3.10. Ako je $\bar{x}=\bar{y}$ tada je $x=y$.

Dokaz: $x=x\cdot 1=x\cdot(y+\bar{y})=x\cdot(y+\bar{x})=x\cdot y+x\cdot\bar{x}=x\cdot y+0=x\cdot y.$

$y=x\cdot 1=x\cdot(y+\bar{y})=x\cdot(y+\bar{x})=x\cdot y+x\cdot\bar{x}=x\cdot y+0=x\cdot y.$

Teorema 3.3.11. (i) $\bar{0}=1$

(ii) $\bar{1}=0$.

Dokaz: (i) Prvi način:

Za proizvoljno $x\in B$ važi

$$\bar{0}=\overline{x\cdot\bar{x}}=\bar{\bar{x}}+\bar{x}=x+\bar{x}=1.$$

Drugi način:

Na osnovu Teoreme 3.3.5. dobija se

$$0+1=1,\ 0.1=0.$$

Treći način: Koristeći aksiome (B3) i (B11) dobijamo

$$\bar{0}=\bar{0}+0=1$$

Primer 3.5. Pojednostaviti sledeći izraz:

$(\bar{a}b+ac)(a+\bar{b})(\bar{a}+\bar{c})$.

Odgovor:

$$(\bar{a}b+ac)(a+\bar{b})(\bar{a}+\bar{c})=(\bar{a}ba+\bar{a}b\bar{b}+aca+ac\bar{b})(\bar{a}+\bar{c})$$

$$=aca\bar{a}+aca\bar{c}+ac\bar{b}\bar{a}+acb\bar{c}$$

$$=0+0+0+0$$

$$=0$$

Primer 3.6. Šta se dobija komplementiranjem izraza $a + b(\bar{c} + u\bar{v})$ i kakav se zaključak na osnovu dobijenog rezultata može izvesti?

Odgovor: Kada se izraz $a + b(\bar{c} + u\bar{v})$ komplementira imaćemo

$$\overline{a + b(\bar{c} + u\bar{v})} = \bar{a}\,\overline{b(\bar{c} + u\bar{v})}$$

$$= \bar{a}(\bar{b} + \bar{\bar{c}}\,\overline{u\bar{v}})$$

$$= \bar{a}(\bar{b} + c(\bar{u} + v))$$

Zaključujemo da se komplementarnost izraza dobija zamenom +(OR) sa • (AND) i obrnuto, i zamenom elementarnih izraza njihovim komplimentima.

3.4. Prekidačka algebra

Ako na skupu $B = \{0,1\}$ definišemo operacije $\vee$ (umesto +), $\wedge$ (umesto •) i $^{-}$ na sledeći način:

$\vee$	0	1
0	0	1
1	1	1

disjunkcija

$\wedge$	0	1
0	0	0
1	0	1

konjunkcija

	$^{-}$
0	1
1	0

negacija

tada se može pokazati da skup B zajedno sa navedenim operacijama zadovoljava aksiome Bulove algebre. Bulova algebra na skupu od dva elementa naziva se *prekidačka algebra*. Napomenimo da ćemo nadalje u tekstu podrazumevati da radimo sa prekidačkom algebrom, tj. da je $B = \{0,1\}$.

3.4.1. Prekidačke funkcije i izrazi

Ako na skupu B definišemo preslikavanje

$$f: B^n \to B$$

tada se preslikavanje f naziva prekidačkom funkcijom (još i Bulovom ili logičkom funkcijom). Elementi skupa B^n su uređene n-torke

$$(x_1, x_2, \ldots, x_n),\ x_i \in B$$

koje nazivamo *vektorima* ili *slogovima* ili *tačkama*. Takvih uređenih n-torki ima ukupno 2^n.

Pod *prekidačkim izrazom* podrazumevamo izraz koji se dobija primenom konačnog broja puta operacija $\vee$, $\wedge$, $^{-}$ nad promenljivama i konstantama iz skupa B.

3.4.2. Zadavanje prekidačkih funkcija

Kako prekidačke funkcije imaju konačnu oblast defnisanosti, one se mogu zadavati u obliku liste svih slogova sa odgovarajućim vrednostima funkcije na tim slogovima. Ovakve liste nazivamo *istinitosne* ili *kombinacione tablice*. Opšti oblik jedne takve tablice dat je na slici 3.4.1.

x_1	x_2	...	x_n	f
0	0	...	0	$f(0, 0, ..., 0)$
0	0	...	1	$f(0, 0,..., 1)$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	1	1	1	$f(1, 1,..., 1)$

Slika 3.4.1. Opšti oblik kombinacione tablice

Funkcija može da ima vrednosti 0 ili 1 na pojedinim slogovima, ali se mogu javljati i nedefinisane vrednosti (što se obično označava sa x ili *, a tumači tako da funkcija na tom slogu može biti jednaka 0 ili 1).

Osim pomenutih načina zadavanja funkcije u upotrebi je i analitički oblik zadavanja funkcije. Kao i kod svih ostalih matematičkih funkcija, i ovde se funkcije mogu zadati u *implicitnom* ili u *eksplicitnom* obliku. Jasno je da kod funkcija sa većim brojem promenljivih (n>5) zadavanje tablicom postaje nepregledno, pa se analitičko zadavanje jedino i primenjuje. Naravno, analitičko zadavanje funkcija se često primenjuje i kod funkcija sa manjim brojem promenljivih.

Sledi istinitosna tablica za Bulov izraz $\overline{x \vee \overline{(x \vee y)}}$.

	x	y	$x \vee y$	$\overline{(x \vee y)}$	$x \vee \overline{(x \vee y)}$	$\overline{x \vee \overline{(x \vee y)}}$
	0	0	0	1	1	0
	0	1	1	0	0	1
	1	0	1	0	1	0
	1	1	1	0	1	0
korak	1	1	2	3	4	5

Iz tabele se može se uočiti jednakost $\overline{x \vee \overline{(x \vee y)}} = \bar{x} \wedge y$, jer je poslednja kolona jednaka istinitosnoj vrednosti izraza $\bar{x} \wedge y$. Ova ekvivalencija može biti dokazana pravilima Bulove algebre.

$$\overline{x \vee \overline{(x \vee y)}} = \bar{x} \wedge \overline{\overline{(x \vee y)}} = \bar{x} \wedge (x \vee y) = (\bar{x} \wedge x) \vee (\bar{x} \wedge y) = 1 \vee (\bar{x} \wedge y) = \bar{x} \wedge y.$$

Primer 3.7. Odredi istinitosne tablice za sledeće funkcije:

$$f1\ (x, y, z) = x \wedge y \vee \bar{x} \wedge y \vee \bar{y} \wedge \bar{z}$$
$$f2\ (x, y, z) = \bar{x} \vee y \wedge \bar{z}$$

Odgovor:

x	y	z	$f_1\ (x,y,z)$	$f_2\ (x,y,z)$
0	0	0	1	1
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

Primer 3.8. U tabeli su izlistane sve moguće funkcije od dve promenljive i dodeljena su imena nekim karakterističnim funkcijama.

$xy =$	00	01	10	11	funkcija	ime
	0	0	0	0	0	
	0	0	0	1	$x \wedge y$	AND
	0	0	1	0	$x \wedge \bar{y}$	
	0	0	1	1	x	
	0	1	0	0	$\bar{x} \wedge y$	
	0	1	0	1	y	
	0	1	1	0	$\bar{x} \wedge y \vee x \wedge \bar{y}$	ExOR
	0	1	1	1	$x + y$	OR
	1	0	0	0	$\overline{x \vee y}$	NOR
	1	0	0	1	$\bar{x} \wedge \bar{y} \vee x \wedge y$	ekvivalencija
	1	0	1	0	$\bar{y}$	
	1	0	1	1	$x \vee \bar{y}$	
	1	1	0	0	$\bar{x}$	
	1	1	0	1	$\bar{x} \vee y$	implikacija
	1	1	1	0.	$\overline{x \wedge y}$	NAND
	1	1	1	1	1	

3.4.3. Osobine nekih prekidačkih funkcija

Što se tiče funkcija koje zavise od samo jedne nezavisno promenljive, osim identičke funkcije $f(x) = x$, interesantna je NOT funkcija čija je kombinaciona tablica prikazana na slici 3.4.2.

x	NOT
0	1
1	0

x	y	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Slika 3.4.2. Kombinacione tablice funkcija NE, I i ILI

Od funkcija dve promenljive najpoznatije su funkcije *AND* i *OR*.

Osim ovih funkcija, najčešće se koriste funkcije suma po modulu dva (isključivo ili, ExOR), implikacija, NAND (NI) i NOR (NILI). Kombinacione tablice ovih funkcija prikazane su na slici 3.4.3.

x	y	$\oplus$	$\rightarrow$	NI	NIL
0	0	0	1	1	1
0	1	1	1	1	0
1	0	1	0	1	0
1	1	0	1	0	0

Slika 3.4.3. Kombinacione tablice za funkcije ExOR, implikacija, NAND i NILI.

Analitički oblik funkcije suma po modulu 2 je sledeći:

$$f(x, y) = x \oplus y = \bar{x}\, y + x\, \bar{y}$$

Teorema 3.4.1. Funkcija $\oplus$ poseduje sledeće osobine:

$x \oplus y = y \oplus x$

$x \oplus (y \oplus z) = (x \oplus y) \oplus z = x \oplus y \oplus z$

$x \oplus x = 0$

$x \oplus \bar{x} = 1$

$x \oplus 1 = \bar{x}$

$x \oplus 0 = x$

Dokaz.

1. $x \oplus y = \bar{x}y + x\bar{y} = \bar{y}x + y\bar{x} = y \oplus x$

2. $x \oplus (y \oplus z) = \bar{x}(y \oplus z) + x\overline{(y \oplus z)} = \bar{x}(\bar{y}z + y\bar{z}) + x\overline{(\bar{y}z + y\bar{z})}$

$$= \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \overline{\bar{y}z} \cdot \overline{y\bar{z}}$$
$$= \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot (y + \bar{z})(\bar{y} + z)$$
$$= \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot (y\bar{y} + yz + \bar{z}\bar{y} + \bar{z}z)$$
$$= \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot y \cdot z + x \cdot \bar{y} \cdot \bar{z}$$

$$(x \oplus y) \oplus z = (\bar{x}y + x\bar{y}) \oplus z = \overline{(\bar{x}y + x\bar{y})} \cdot z + (\bar{x}y + x\bar{y})\bar{z}$$
$$= \overline{\bar{x}y} \cdot \overline{x\bar{y}} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$
$$= (x + \bar{y}) \cdot (\bar{x} + y)z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$
$$= (xy + \bar{x}\,\bar{y})z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$
$$= x \cdot y \cdot z + \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$

Funkcija implikácije ima sledeći analitički oblik:

$$f(x, y) = x \to y = \overline{(x \cdot \bar{y})} = \bar{x} + y$$

Teorema 3.4.2. Funkcija implikacije poseduje sledeće osobine:

1. $x \to y \neq y \to x$
2. $x \to (y \to z) \neq (x \to y) \to z$
3. $x \to x = 1$
4. $x \to \bar{x} = \bar{x}$
5. $x \to 0 = \bar{x}$
6. $x \to 1 = 1$
7. $0 \to x = 1$
8. $1 \to x = x$

Analitički oblici funkcija NI i NILI su sledeći:

$\mathrm{NI}(x, y) = \bar{x} + \bar{y} = \overline{xy} = x|y$ $\qquad$ $\mathrm{NILI}(x, y) = x \downarrow y = \overline{x + y} = \bar{x} \cdot \bar{y}$

Teorema 3.4.3. Za funkcije NAND i NOR važi komutativnost, ali asocijativni zakon ne važi. Od ostalih osobina važe sledeće.

1. $x \uparrow x = \bar{x}$ $\quad x \downarrow x = \bar{x}$
2. $x \uparrow \bar{x} = 1$ $\quad x \downarrow x = 0$
3. $x \uparrow 0 = 1$ $\quad x \downarrow 0 = \bar{x}$
4. $x \uparrow 1 = \bar{x}$ $\quad x \downarrow 1 = 0$

3.4.4. Reprezentacija Bulovih operacija.

Osnovne Bulove opracije $\wedge$, $\vee$ i $\neg$ imaju jedno veoma važno svojstvo: svaka druga Bulova operacija se može predstaviti pomoću ove tri specijalne Bulove operacije.

Počnimo jednim primerom. Pokazaćemo kako se operacija koja je data u tabeli desno može predstaviti pomoću tri osnovne Bulove operacije. Recept se sastoji iz tri koraka.

1. Posmatramo samo one redove za koje je vrednost operacije jednaka 1 (redovi označeni strelicom).
2. Za svaki takav red napravimo po jedan Bulov izraz, ovako: ako je vrednost promenljive 0 uzmemo njenu negaciju, a ako je vrednost promenljive 1 uzmemo promenljivu bez izmena. Onda uzmemo konjunkciju tako pripremljenih promenljivih ili njihovih negacija.

x y z	$f(x, y, z)$	
0 0 0	0	
0 0 1	1	←
0 1 0	0	
0 1 1	0	
1 0 0	1	←
1 0 1	1	←
1 1 0	0	
1 1 1	1	←

U navedenom primeru imamo četiri reda u kojima se na kraju javlja jedinica. Odgovarajuće konjunkcije su prikazane u tabeli ispod:

x	y	z	konjunkcija
0	0	1	$\neg x \wedge \neg y \wedge z$
1	0	0	$x \wedge \neg y \wedge \neg z$
1	0	1	$x \wedge \neg y \wedge z$
1	1	1	$x \wedge y \wedge z$

3. Reprezentacija Bulove operacije date tabelom se tada dobija kada se uzme disjunkcija svih ovih konjunkcija. Dakle,

$$f(x, y, z) = (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z).$$

Nećemo dati dokaz da je opisani recept dobar, već ćemo se ubediti na par primera. Pogledajmo, prvo, jedan red iz tabele kome je vrednost operacije 0. Recimo, na osnovu trećeg reda je $f(0, 1, 0) = 0$. Drugim rečima, kada je $x = 0$, $y = 1$ i $z = 0$, onda je $f = 0$. Kada uvrstimo ove vrednosti u izraz na desnoj strani, dobijamo da su svi disjunkti (tj. "sabirci") jednaki 0:

$$\neg x \wedge \neg y \wedge z = \neg 0 \wedge \neg 0 \wedge 0 = 0$$
$$x \wedge \neg y \wedge \neg z = 0 \wedge \neg 0 \wedge \neg 0 = 0$$
$$x \wedge \neg y \wedge z = 0 \wedge \neg 0 \wedge 0 = 0$$
$$x \wedge y \wedge z = 0 \wedge 0 \wedge 0 = 0$$

Odatle sledi da je vrednost izraza $0 \vee 0 \vee 0 \vee 0 = 0$. Ako sada uvrstimo vrednosti promenljivih iz petog reda, $x = 1$, $y = 0$, $z = 0$, dobijamo da je tačno jedan disjunkt jednak 1, dok su ostali jednaki 0:

$$\neg x \wedge \neg y \wedge z = \neg 1 \wedge \neg 0 \wedge 0 = 0$$
$$x \wedge \neg y \wedge \neg z = 1 \wedge \neg 0 \wedge \neg 0 = 1$$
$$x \wedge \neg y \wedge z = 1 \wedge \neg 0 \wedge 0 = 0$$
$$x \wedge y \wedge z = 1 \wedge 0 \wedge 0 = 0$$

pa je u ovom slučaju vrednost izraza $0 \vee 1 \vee 0 \vee 0 = 1$. Primetimo: tačno onaj disjunkt koji je nastao od petog reda je jednak 1, a svi ostali su jednaki 0!

Da zaključimo: svaki disjunkt može da "prepozna" red od koga je nastao i tada da vrati vrednost 1. U svim ostalim slučajevima daje vrednost 0. Ako umesto x, y, z zamenimo vrednosti iz nekog reda kome je na kraju 0, nijedan disjunkt ga ne prepozna, svi daju vrednost 0, pa je i vrednost celog izraza 0. Međutim, ako uvrstimo vrednosti promenljivih iz nekog reda kome je na kraju 1, njegov disjunkt ga prepozna i daje vrednost 1, ostali daju vrednost 0, pa je vrednost celog izraza jednaka $0 \vee \ldots \vee 0 \vee 1 \vee 0 \vee \ldots \neg 0 = 1$. Tako dobijamo da je f predstavljena u obliku $f = D_1 \vee D_2 \vee \ldots \vee D_k$, a svaki D_i radi ovako: "ako je stigao red tabele od koga sam nastao, vraćam 1, inače vraćam 0".

Zadaci.

1. Bulovu operaciju koja je data tabelom pored predstaviti pomoću osnovnih Bulovih operacija.

x	y	z	u	$f(x, y, z, u)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

2. Odaberite proizvoljnu Bulovu operaciju sa 4 promenljive koja u koloni ispod f ima tačno 5 jedinica i predstavite je pomoću osnovnih Bulovih operacija.

3. Uzmite Bulovu operaciju sa dve promenljive kojoj su u koloni ispod f sve nule i predstavite je pomoću osnovnih Bulovih operacija.

(Napomena: ovo je jedini slučaj u kome recept ne radi! Budite lukavi!)

4. Dokažite da je $x \vee y = \neg (\neg x \wedge \neg y)$. (Direktnom proverom! Ima samo četiri mogućnosti za vrednosti promenljivih.) Pitanje: Da li se svaka Bulova operacija zadata tabelom može predstaviti samo pomoću $\wedge$ i $\neg$?

5. Dokažite da je $x \wedge y = \neg (\neg x \vee \neg y)$. (Opet direktnom proverom.) Pitanje: Da li se svaka Bulova operacija zadata tabelom može predstaviti samo pomoću $\vee$ i $\neg$?

6. Posmatrajmo Bulovu operaciju $\uparrow$ koja je definisana sa $x \uparrow y = \neg (x \wedge y)$.

- Dokažite da se $\neg$ i $\wedge$ mogu dobiti samo od $\uparrow$.

(Pogledajte šta je $x \uparrow x$ i primetite da je $x \wedge y = \neg (x \uparrow y)$.)

- Pitanje: Da li se svaka Bulova operacija zadata tabelom može predstaviti samo pomoću $\uparrow$?

7. Posmatrajmo Bulovu operaciju $\downarrow$ koja je definisana sa $x \downarrow y = \neg (x \vee y)$.

- Dokažite da se $\neg$ i $\vee$ mogu dobiti samo od $\downarrow$.

- Pitanje: Da li se svaka Bulova operacija zadata tabelom može predstaviti samo pomoću $\downarrow$?

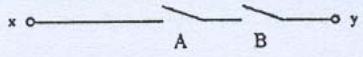
3.5. Prekidačka kola

Algebra prekidačkih kola je specijalan slučaj Bulove algebre, koji nas vodi u svet elektronike. Elementi algebre prekidačkih kola su prekidači, kao analogije binarnim vrednostima. Prekidač može da bude uključen ili isključen.

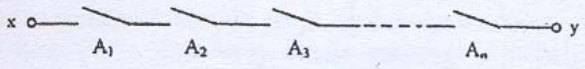
Prekidač je uređaj koji predstavlja tačku u električnom polju koja može imati dva stanja, uključeno ili isključeno (ili kako često kažemo, otvoreno i zatvoreno). Kada je prekidač otvoren, struja ne teče kroz kolo, a u suprotnom, ukoliko je prekidač zatvoren, postoji protok struje u kolu. Prekidač ćemo označiti sa $-\ _A\backslash_-$, gde A označava rečenicu koja je istinita (tačna) ukoliko je prekidač zatvoren, a u suprotnom je neistinita (netačna). Reći ćemo da su dve tačke povezane prekidačkim kolom, ako i samo ako su povezane provodnikom (linijom) sa konačno mnogo prekidača.

Za prekidače povezane u sisteme prekidača ili prekidačka kola važe aksiome bulove algebre.

Na slici 3.5.1. je prikazano prekidačko kolo u kome su prekidači vezani takozvanom rednom vezom. Očigledno struja teče između tačaka x i y kroz ovo prekidačko kolo ako i samo ako su svi prekidači zatvoreni, tj ako i samo ako je tačna formula $A \wedge B$. Ovaj slučaj možemo generalisati za konačno mnogo prekidača koje ćemo vezati rednom vezom, kao na slici 3.5.2. Uslov protoka struje kroz kolo, između tačaka x i y, je da svi prekidači budu zatvoreni odnosno da je tačna formula $A_1 \wedge A_2 \wedge \cdots \wedge A_n$.

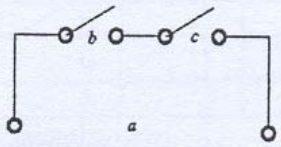


Slika 3.5.1.



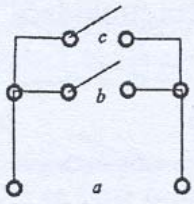
Slika 3.5.2.

Osvrnimo se na primer ispisivanja na štampaču. Štampač ispisuje ako program to zahteva i ako je papir u štampaču. Ispisivanje će aktivirati električna struja koja protiče kroz kolo od štampača. U ovom pojednostavljenom primeru kolo čine dva prekidača. Prvi od njih zavisi od toka izvršavanja programa. Kada on zahteva ispisivanje, uključuje se prekidač *b*. Ali to još uvek nije dovoljan uslov za ispisivanje. Uslov za ispisivanje je zadovoljen kada je uključen i prekidač *c*, koji zavisi od prisutnosti papira.



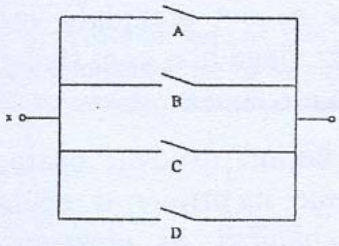
Prekidačkokolo $a = b \cdot c$

Kako predstavljamo *disjunkciju:* $a = b \vee c$? U ovom slučaju je veza *a* spojena ako je uključen bar jedan od prekidača *b* ili *c*. Ovo povezivanje je prikazano na slici. Paralelno povezivanje prekidača predstavlja njihovu disjunkciju.

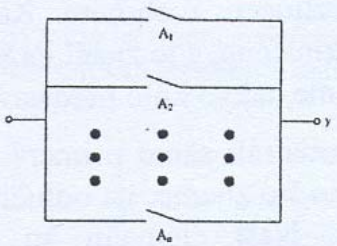


Prekidačkokolo $a = b \vee c$

Na slici 3.5.3. tačke *x* i *y* su povezane prekidačkim kolom; za tako vezane prekidače reći ćemo da su povezani paralelnom vezom. Očigledno da će struja teći u ovom kolu, ukoliko je bilo koji od ovih prekidača zatvoren, tj. ako i samo ako je tačna formula $A \vee B \vee C \vee D$. Možemo napraviti i generalizaciju, da se paralelno može vezati konačno mnogo prekidača (kao na slici 3.5.4).Tada struja teče kroz kolo ako i samo ako je tačna formula $A_1 \vee A_2 \vee .. . \vee A_n$.

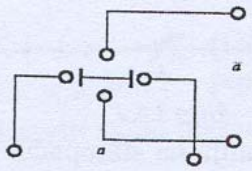


Slika 3.5.3.



Slika 3.5.4.

Pogledajmo još *negaciju*. Prekidač $\bar{a}$ na sledeći način zavisi od prekidača *a*: ako je prekidač *a* uključen, onda je prekidač $\bar{a}$ isključen i obratno. Prekidače između kojih postoji takva zavisnost možemo predstaviti kružnim prekidačem.



Predstavljanje negacije kružnim prekidačem

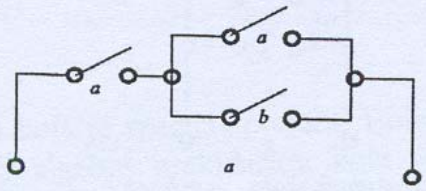
U sledećoj tabeli je prikazana veza između Bulove algebre, algebre partitivnog skupa, prekidačke algebre i prekidačkih kola.

Bulova algebra	Podskupovi skupa U	Prekidačka algrbra	Prekidačka kola
$x + y$	$X \cup Y$	$x \vee y$	paralelna veza
xy	$X \cap Y$	$x \wedge y$	serijska veza
$\bar{x}$	X^c	$\bar{x}$	kružni prekidač
0	$\varnothing$	0	otvoreni prekidač
1	U	1	zatvoreni prekidač
=	=	ekvivalencija	ekvivalentna kola

Sada kada smo razmotrili osnovne operacije, pregledaćemo aksiome Bulove algebre u svetlu algebre prekidačkih kola. Posmatramo aksiomu o apsorpciji:

$$a \wedge (a \vee b) = a.$$

Ta aksioma je predstavljena povezivanjem prekidača sa sledeće slike.



$a = a \wedge (a \vee b)$

Kroz ovo kolo protiče struja ako i samo ako je prekidač a uključen. To znači da kolo ne zavisi od prekidača b. Prema tome, kolo se ponaša kao da sadrži samo prekidač a.

Sada analiziramo sledeću aksiomu: $a \wedge \bar{a} = 0$

Ova aksioma je predstavljen serijskom vezom dva jednaka prekidača, samo što jednom uzimamo stvarni položaj prekidača a drugi put suprotni. Ako je prekidač a uključen, onda je $\bar{a}$ isključen, i obrnuto. Kako su ovi prekidači vezani jedan za drugim, kolo je uvek isključeno, a to znači da kroz njega ni u kom slučaju ne može da protiče struja. Prema tome, takvo kolo predstavlja element 0.

Do sada smo razmatrali samo primere u kojima osnovne operacije povezuju najviše tri elementa. Kao što znamo, na odlučivanje, na primer, u računaru utiče vrlo veliki broj elemenata. Naši elementi su mehanički ili elektronski prekidači realizovani na različite načine. Svaki prekidač možemo da posmatramo kao binarnu promenljivu. Takva promenljiva dobija jednu od dve vrednosti, 1 ili 0. Funkcionalne veze među promenljivim možemo opisati operacijama konjukcije, disjunkcije i negacije. Pomoću obimnih skupova takvih funkcija možemo opisati događaje u digitalnim računarima.

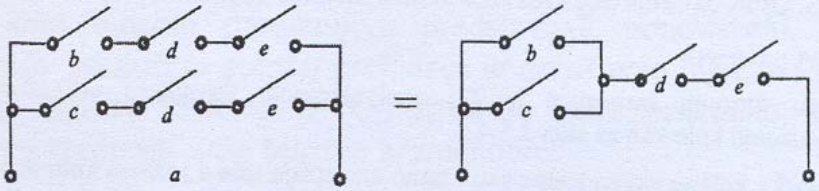
Primer 3.9. Za ilustraciju, pogledajmo primer nešto složenije binarne funkcije

$a = (b \wedge d \wedge e) \vee (c \wedge d \wedge e).$

U ovom slučaju možemo izostaviti zagrade, jer je konjukcija u odnosu na disjunkciju operacija višeg prioriteta. Osim toga, zapis funkcije je duži nego što je potrebno. Uzimajući u obzir komutativnost i distributivnost, zapis možemo da redukujemo na sledeći način:

$$a = (b \wedge d \wedge e) \vee (c \wedge d \wedge e)$$
$$= (d \wedge e) \wedge b \vee (d \wedge e) \wedge c$$
$$= (d \wedge e) \wedge (b \vee c)$$
$$= (b \vee c) \wedge d \wedge e$$

Redukovani zapis sadrži manje promenljivih, pa je samim tim za njegovo predstavljanje potrebno manje prekidača.



Primer 3.10. U sledećem primeru odredimo funkciju F koja utvrđuje jednakost dveju računske reči: ako su reči jednake, neka F bude jednako 1, a inače 0. Tako su, na primer, dve šestobitne reči jednake, ako se na svih šest odgovarajućih mesta nalaze jednaki znaci. Uzmimo dve šestobitne reči A i B. Svaku od njih predstavlja niz od šest binarnih promenljivih:

$$A = a_1\ a_2\ a_3\ a_4\ a_5\ a_6$$
$$B = b_1\ b_2\ b_3\ b_4\ b_5\ b_6$$

Reč A je jednaka reči B ako za svaki par (a_i, b_i) važi $a_i = b_i$, gde je $1 \le i \le 6$. Uvedimo šest funkcija: $f_1, f_2, f_3, f_4, f_5, f_6$.. Neka funkcija f_i dobija vrednost 1, ako je $a_i = b_i$. Napišimo funkciju f_i

$$f_i = a_i \wedge b_i \vee \bar{a}_i \wedge \bar{b}_i$$

Funkcija f_i dobija vrednost 1 u dva slučaja: ako su a_i i b_i istovremeno jednaki 1 ili ako su a_i i b_i istovremeno jednaki 0. Na funkcionalnu vezu f_i se često nailazi, pa je zato dobila i svoje ime. Naziva se *logičkaekvivalencija*.

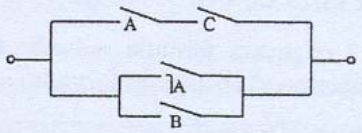
Vrednost funkcije $f_i = 1$ označava tada jednakost dveju reči na i-tom mestu. Funkcija F, koja utvrđuje jednakost reči u celini, dobija vrednost 1, ako su sve funkcije f_i jednake 1. Vidimo da funkciju F dobijamo ako sve funkcije f_i međusobno povežemo operacijom konjukcije. Prema tome je

$$F = f_1 \wedge f_2 \wedge f_3 \wedge f_4 \wedge f_5 \wedge f_6.$$

Uzmimo dve konkretne reči: $A = 100101$ $B = 110101$.

Reči A i B nisu jednake; prema tome, u ovom slučaju je $F = 0$. Kako a_2 nije jednako b_2, to znači da je $f_2 = 0$, pa je samim tim i $F = 0$.

Primer 3.11. U prekidačkom kolu na slici struja može teći samo u slučaju kada je tačna formula $(A \wedge C) \vee (\neg A \vee B)$. Na ovom primeru vidimo da se mogu kombinovati prekidači u rednoj vezi sa prekidačima vezanim paralelno, u takozvanu redno-paralelnu vezu.

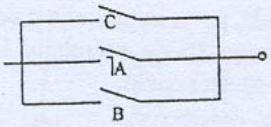


Očigledno bilo koje prekidačko kolo može biti prikazano preko konjukcija, disjunkcija i negacija posebnih prekidača.

Primer 3.12. U prethodnom primeru je pokazano da se uslov protoka struje kroz kolo poklapa sa uslovom tačnosti formule $(A \wedge C) \vee (\neg A \vee B)$. Primenom pravila asocijativnosti dobićemo ekvivalentnu formulu $((A \wedge C) \vee \neg A) \vee B$, ili još jednostavnije

$$((A \wedge C) \vee \neg A) \vee B = ((A \vee \neg A) \wedge (C \vee \neg A)) \vee B = ((1 \wedge (C \vee \neg A)) \vee B = C \vee \neg A \vee B.$$

Kolo predstavljeno ovom formulom prikazano je na slici.



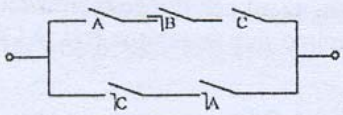
Kolo na toj slici jednostavnije je od kola u prethodnom primeru jer ima manje prekidača. Uopšteno, proces simplifikacije (pojednostavljenja) kola vodi smanjivanju broja prekidača u kolu, što znači da bi konstruisanje datog kola u industriji bilo jeftinije, ekonomičnije, a dato kolo bi brže radilo.

Primer 3.13. Uslov da struja teče kroz kolo na slici 3.5.3. je da je tačna formula

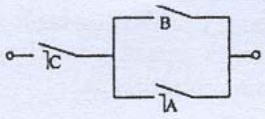
$(A \wedge B \wedge \neg C) \vee (\neg C \wedge \neg A)$.

Ukoliko ovu formulu zamenimo logički ekvivalentnom formulom $\neg C \wedge (B \vee \neg A)$, dobićemo jednostavnije, ekvivalentno kolo kao na slici 3.5.4.

(Dva prekidačka kola su ekvivalentna ako i samo ako struja teče u jednom kolu istovremeno kad i u drugom kolu).



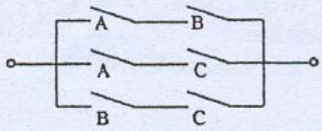
Slika 3.5.3.



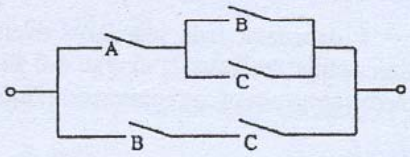
Slika 3.5.4.

Primer 3.14. Tročlana komisija odlučuje većinom glasova, o određenim zakonima. Svaki član pritiska dugme da potvrdi svoj glas "DA". Konstruišimo prekidačko kolo koje će paliti signal "DA", ako i samo ako je većina članova komisije dala glas "DA ".

Označimo sa A-Prvi član daje glas "DA ", sa B-Drugi daje glas "DA ", sa C-Treći daje potvrdan glas. Odgovarajuća formula je $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$. Odgovarajuće kolo je onda prikazano na slici 3.5.5. Jednostavnija formula je $(A \wedge (B \vee C)) \vee (B \wedge C)$, a jednostavnije, ekvivalentno kolo je na slici 3.5.6.



Slika 3.5.5.



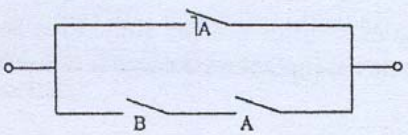
Slika 3.5.6.

Već smo govorili o potrebi da napravimo što jednostavnija kola, čija će izrada i upotreba biti što brža i jeftinija. Time nam se javlja problem *minimizacije* kola, odnosno nalaženje *svih* najjednostavnijih kola, pri čemu kriterijumi jednostavnosti nisu samo broj ulaza $\vee$ i $\wedge$, niti broj formula koje učestvuju u kolu, već vreme potrebno da kolo radi (samim tim kolo će biti jeftinije).

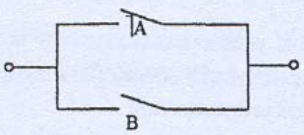
Primer 3.15. Kolu na slici 3.5.7 odgovara formula $\neg A \vee (B \wedge A)$. Očiglednom primenom zakona distributivnosti za logičke promenljive, dobićemo ekvivalentnu formulu

$$\neg A \vee (B \wedge A) = (\neg A \vee B) \wedge (\neg A \wedge A) = (\neg A \vee B) \wedge 1 = \neg A \vee B,$$

čiji je prikaz odgovarajućeg kola dat na slici 3.5.8.



Slika 3.5.7.



Slika 3.5.8.

Ideja za pronalaženje jednostavnijih kola je da ih posmatramo kao istinitosne funkcije, a da zatim za nađene istinitosne funkcije ispitamo koja od njih je "najjeftinija". Naravno, sam proces biva jako komplikovan i praktično neupotrebljiv, već za više od dve promenljive. Zato moramo naći metode koje nam pouzdano daju jednostavnija kola, na brz i operativno praktičan način.

3.6. Logička kola

Do sada smo binarnu promenljivu predstavljali prekidačem, ili tačnije, položajem prekidača. Isključeni položaj prekidača je označavao logičku vrednost 0, a uključeni položaj logičku vrednost 1. Upitajmo se kako praktično, u računaru, postižemo promenu vrednosti neke binarne promenljive?

Mehanički prekidači se uključuju pomeranjem ručice prekidača. U računarima su oni relativno retki. Nalazimo ih uglavnom kod ulazno-izlaznih jedinica. Mnogo su češći elektronski prekidači, kakav je na primer tranzistor. Takve prekidače uključuje struja, odnosno napon. Stanje elektronskog prekidača je određeno naponom na njegovom ulazu. Na primer: napon 0V označava stanje 0, a napon 5V stanje 1. Napon na izlazu elektronskog prekidača možemo da obrađujemo kao binarnu promenljivu.

Predstavljanje binarne promenljive pomoću napona je podesno kada se prihvatamo izrade nekog konkretnog elektronskog logičkog kola. Logičko kolo je povezivanje odgovarajućih komponenata, koje sledi zakonitost neke logičke funkcije. Pre nego što pristupimo izradi neke od ovih logičkih funkcija, pogledajmo kako u slučaju napona kao binarne promenljive predstavljamo osnovne Bulove operacije: *konjukciju, disjunkciju* i *negaciju.*

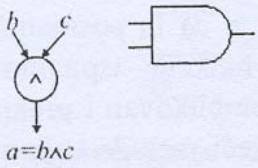
Kao što smo videli, na primer napon 5V može da predstavlja logičku vrednost 1, a napon 0V logičku vrednost 0. Logičko kolo koje predstavlja konjukciju: $a = b \cdot c$ je prikazano na sledećoj slici. Krug sa upisanim znakom konjukcije predstavlja elektronsko kolo koje vrši tu funkciju. Na ulazu u to kolo su naponi b i c, a na izlazu je napon a. Kolo je takvo da je, na primer, napon jednak $a = 5V$, što predstavlja logičku vrednost 1, samo u slučaju ako su oba napona b i c istovremeno jednaka 5. Ista slika prikazuje kolo disjunkcije i kolo negacije.

Logički elementi:

- jesu fizički objekti koji implementiraju neku od funkcija algebre logike,
- predstavljaju osnovne komponente svih elektronskih kola računara

3.6.1. AND logičko kolo

AND logičko kolo na svom izlazu ima 1 ako i samo ako su svi njegovi ulazi jednaki 1. Simbol AND kola sa dva ulaza prikazan je na slici 3.6.1(a), dok je odgovarajuća istinitosna tablica na slici 3.6.1(b).



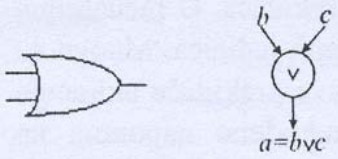
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

a) Različiti logički simboli. b) Istinitosna tablica.

Slika 3.6.1. AND logičko kolo.

3.6.2. OR logičko kolo

OR logičko kolo generiše na izlazu 1 ako je najmanje jedan od njegovih ulaza postavljen na 1. Logički simbol OR kola sa dva ulaza i odgovarajuća istinitosna tablica prikazani su na slici 3.6.2.



x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

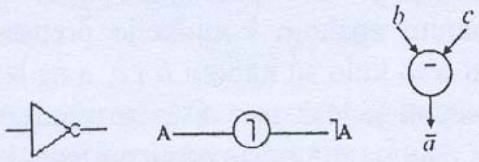
a) Različiti logički simboli. b) Istinitosna tablica.

Slika 3.6.2. OR logičko kolo.

3.6.3. NOT logičko kolo

Inverter (okretač) ¬ je deo logičkog kola, koje umesto ulaznog signala A, daje na izlazu signal $\neg A$; očigledno, ako na ulazu imamo 1, izlazni signal je 0 i obrnuto.

Logički simbol i istinitosna tablica NOT kola prikazane na slici 3.6.3.



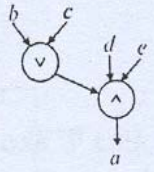
x	z
0	1
1	0

a) Različiti logički simboli. b) istinitosna tablica

Slika 3.6.3. NOT logičko kolo.

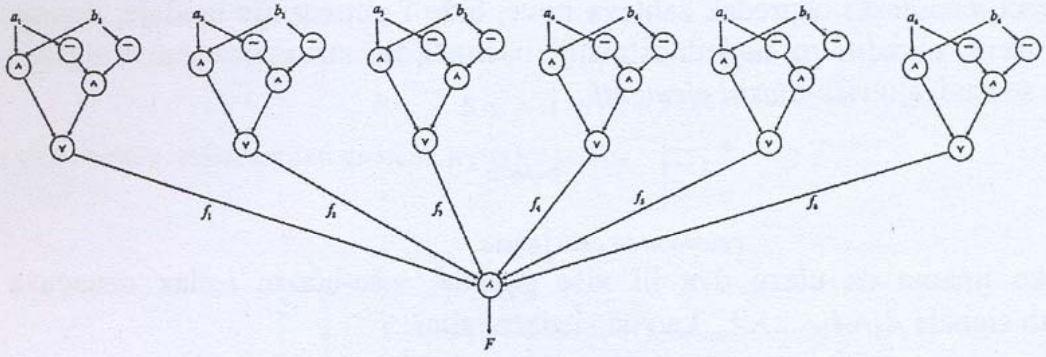
3.6.4. Primeri logičkih kola

Logičko kolo funkcije $a = (b \vee c) \wedge d \wedge e$ dato je narednom slikom



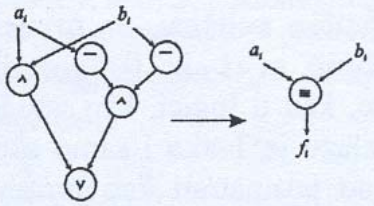
Logičko kolo za funkciju $a = (b \vee c) \wedge d \wedge e$

Razmotrimo logičko kolo za funkciju F koja utvrđuje jednakost dve računarske reči.



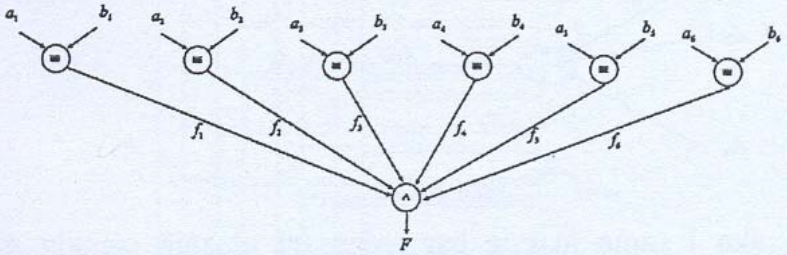
Logičko kolo za funkciju koja utvrđuje jednakost dve 6-bitne reči

Razvoj računara teži elektronskim kolima koja bi u jednom delu objedinjavali što složenije funkcije. Tako bi se, na primer, svaka od funkcija f_i sa prethodne slike realizovala pomoću jednog jedinog elementa: elektronskim kolom koje obavlja funkciju logičke ekvivalencije, koju označavamo simbolom "≡" (videti narednu sliku).



Za funkciju logičke ekvivalencije uvodimo poseban simbol

Na taj način, od daljeg razvoja očekujemo dosta jednostavnije projektovanje i izgradnju računara.



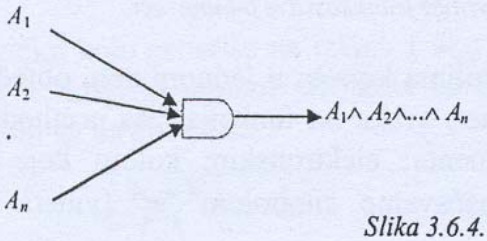
Isto kolo sa nacrtano pomoću simbola logičke ekvivalencije

Zaključujemo da se logički događaji u digitalnim računarima opisuju pomoću funkcija koje iskazuju veze među promenljivima, a koje se opisuju prekidačkim kolima. Ovakva kola realizuju tri osnovne logičke operacije: konjukcije, disjunkcije i negacije, i realizuju se kao logička elektronska kola. Šta smo takvim načinom zapisivanja dobili? Krugovi sa upisanim operacijama $\wedge$, $\vee$ i $^-$, predstavljaju elektronska kola koja vrše te operacije na naponima koji u njih ulaze. Ta kola možemo da kupujemo gotova i da ih po želji međusobno povezujemo. Elektronska kola koja se koriste za realizaciju Bulovih funkcija nazivaju se logička kola. Postoje AND, OR i NOT (invertor) logička kola koji obavljaju AND, OR i NOT operaciju, respektivno.

Sve veći tehnološki napredak zahteva nove, brže i pouzdanije uređaje, kojima će se informacije obraditi na najjednostavniji način. Zato su napravljeni drugačiji uređaji, koji se nazivaju *više-ulazni elementi.*

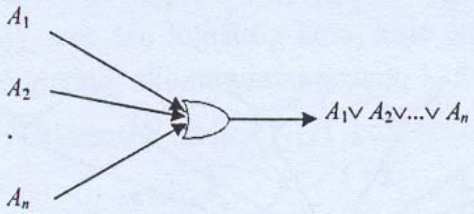
I element sa četiri ulaza

Ukoliko imamo na ulazu dva ili više signala, više-ulazni *i*-ulaz označava konjukciju tih signala $A_1 \wedge A_2 \wedge ... \wedge A_n$, kao na sledećoj slici.



Slika 3.6.4.

Ukoliko su ulazni signali fizičke veličine, na primer napon, tada na izlazu možemo imati dva stanja, označena sa 1 ili 0 (postoji izlazni napon ili ne, respektivno). Stanje 1 označava se, kao u logici, kao istinito, a stanje 0 kao lažno (neistinito). Očigledno stanje na izlazu je 1 ako i samo ako su svi ulazni signali 1 (tačni). Aritmetički, *i*-ulaz možemo posmatrati kao proizvod ulaznih signala. Na sličan način možemo definisati sledeći element logičkog kola, više-ulazni ili-ulaz, kao disjunkciju ulaznih signala $A_1 \vee A_2 \vee .... \vee A_n$, kao na sledećoj slici.



Slika 3.6.5.

Izlazni signal je 1 ako i samo ako je bar jedan od ulaznih signala A_i jednak 1. Aritmetički, izlazni signal možemo posmatrati kao maksimum ulaznih signala.

Logičko kolo je definisano kao kolo sa različitim promenljivama na ulazu, koje su vezane i-ulazima, ili-ulazima i okretačima i eventualno još nekim uređajima za formiranje istinitosnih funkcija.

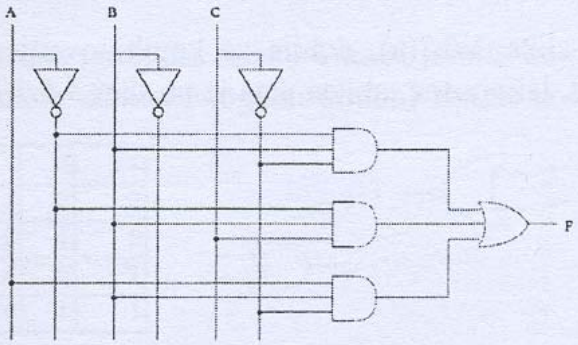
Primer 3.16. Neka je funkcija F zadata preko tabele istinitosnih vrednosti predstavljene u tabeli.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Funkcija F se može izraziti u analitičkom obliku

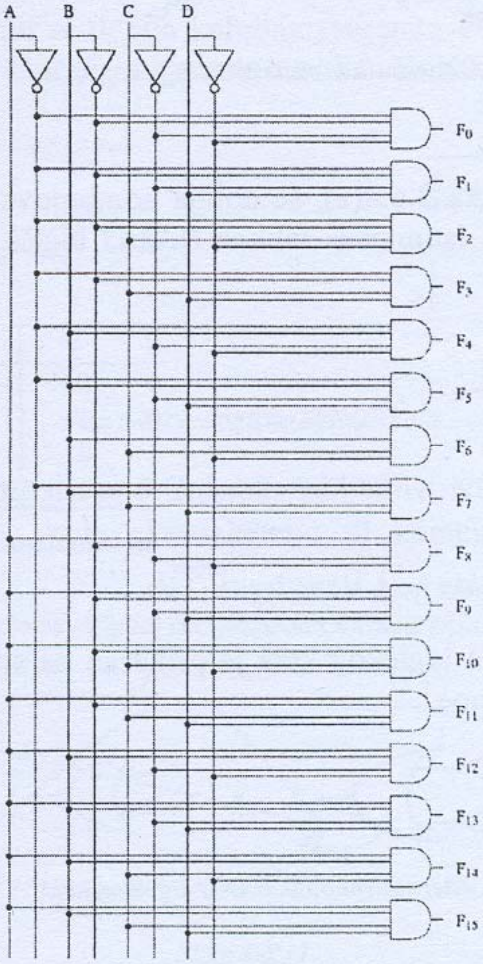
$$F = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C}$$

Ovaj izraz se može implementirati sledećim logičkim kolom.



Primer 3.17. Želimo da konstruišemo dekoder koji koji vrši dekodiranje (prevođenje) grupe od četiri binarne cifre iz koda u heksadekadnu vrednost. Dekoder će imati 4 ulaza i 16 izlaza.

Označimo ulaz u dekoder slovima A, B, C, D. Dekadna vrednost je jednaka izrazu ABCD. Vrednost A=1 označava da je prva cifra jednaka 1, dok vrednost A=0 označava da je prva cifra jednaka 0. Analogno pravilo važi za promenljive B, C i D. Tada, naprimer, $F_0 = \overline{ABCD}$ označava heksadekadnu cifru 0, dok $F_{15} = ABCD$ označava heksadekadnu cifru 15.

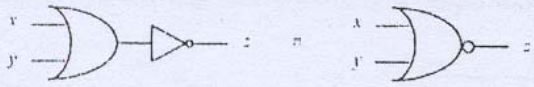


3.6.5. Izvedena logička kola

Od tri elementarna logička kola je moguće izvesti još dva, NOR logičko kolo i NAND logičko kolo.

A. NOR logičko kolo

NOR logičko kolo, slika 3.6.7(a), dobija se kombinovanjem NOT logičkog kola sa OR logičkim kolom. Islinitosna tablica data je na slici 3.6.7(b).



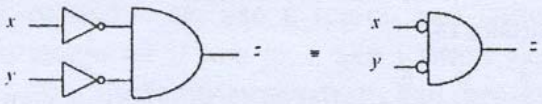
a) Logički simbol

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

b) Istinitosna tablica

Slika 3.6.6. NOR logičko kolo.

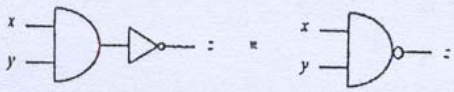
Alternativni oblik NOR logičkog kola je prikazan na slici 3.6.7, i sledi neposredno iz De Morganove teoreme.



Slika 3.6.7. Alternalivni oblik NOR logičkog kola.

B. NAND logičko kolo

NAND logičko kolo, slika 3.6.8(a), se dobija kombinovanjem NOT logičkog kola sa AND logičkim kolom. Istinitosna tablica NAND logičkog kola prikazana je na slici 3.6.8(b).



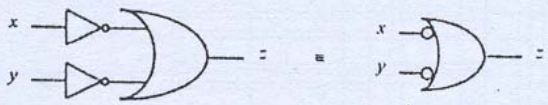
a) Logičiki simbol.

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

b) Istinitosna tablica.

Slika 3.6.8. NAND logičko kolo.

Alternativni oblik NAND logičkog kola je prikazan na slici 3.6.9, i dobija se primenom De Morganove teoreme.



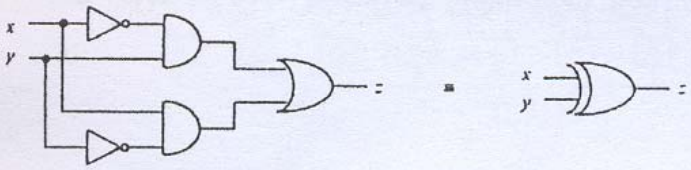
Slika 3.6.9. Alternativni oblik NAND logičkog kola.

3.6.6. Ostali tipovi logičkih kola

Dva dodatna tipa logičkih kola koji se često koriste kod digitalnih kola su ExOR (Exclusive OR) i ExNOR. (Exclusive NOR) kola.

A. ExOR kola

ExOR kolo generiše na svom izlazu 1 kada je bilo koji od njegovih ulaza, ali ne i oba, na 1. Istinitosna tablica i logički simbol ExOR kola prikazani su na slici 3.6.10.



x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

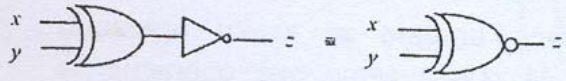
a) Logički simbol. b) Istinitosna tablica

Slika 3.6.10. Logičko ExOR kolo.

ExOR kolo je poznato i kao isključivo ILI kolo. Bulova funkcija koja odgovara istinitosnoj tablici ExOR kola je oblika $f(x, y) = \bar{x}\,y + x\,\bar{y} = x \oplus y$

B. ExNOR kolo

ExNOR logičko kolo se dobija kombinovanjem NOT logičkog kola sa ExOR logičkim kolom. Logički simbol i istinitosna tablica ExNOR logičkog kola prikazani su na slici 3.6.11.



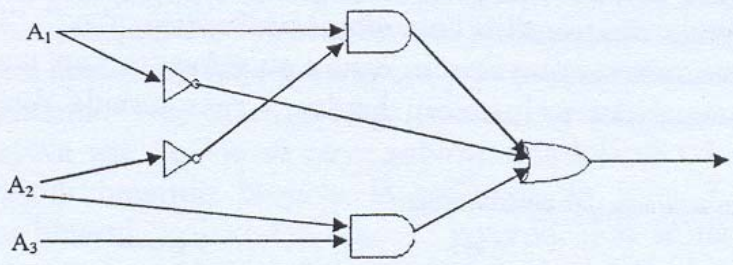
x	y	z
0	0	1
0	1	0
1	0	0
1	1	1

a) Logički simbol. b) Istinitosna tablica

Slika 3.6.11. Logičko ExNOR kolo

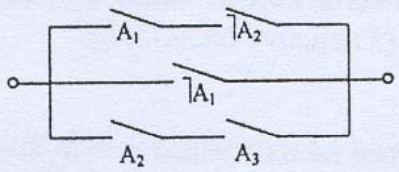
ExNOR kolo je poznato pod imenom isključivo NOR kolo. ExNOR logičko kolo se naziva i logičko kolo ekvivalencije ill koincidencije, a predstavlja se simbolom ⊙.

Primer 3.18. Logičko kolo na sledećoj slici odgovara formuli $(A_1 \wedge \neg A_2) \vee \neg A_1 \vee (A_2 \wedge A_3)$.



Slika 3.6.11.

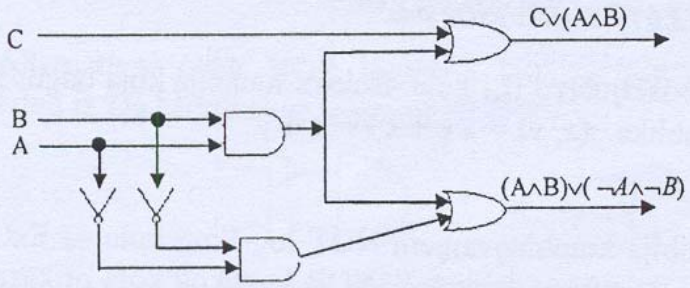
Primetimo da možemo konstruisati kolo sa redno-paralelnom vezom, kao na slici 3.6.12, koje je ekvivalentno traženom logičkom kolu sa slike 3.6.11.



Slika 3.5.12.

Logički zahtevi upućuju nas na potrebu da imamo ponekad više od jednog izlaznog signala.

Primer 3.19. Na slici 3.6.13. prikazano je jedno logičko kolo sa više izlaza.



Slika 3.6.13.

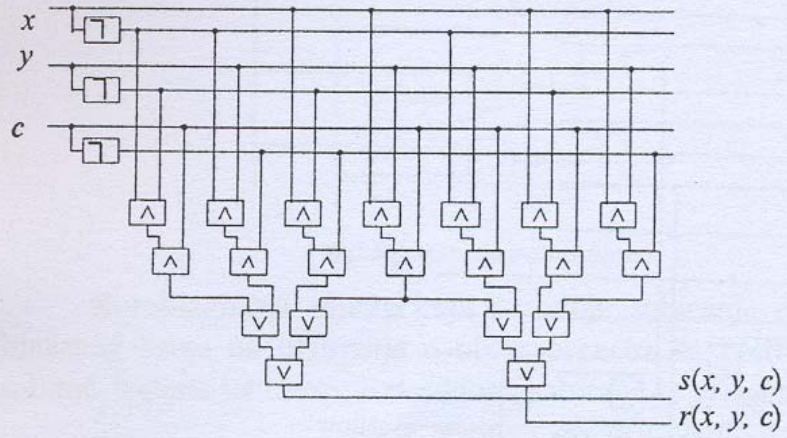
3.6.7. Logičke operacije – osnova hardvera

Prvi elektronski računari su imali posebne uređaje za svaku operaciju koju su umeli da izvrše, i zbog toga su umeli da izvrše samo one operacije za koje su imali odgovarajuće uređaje. Na primer, ENIAC je imao uređaj za numeričku integraciju (štagod to bilo) pa je približne vrednosti integrala računao tako što bi uposlio taj uređaj. Međutim, ako je za rešavanje nekog problema bilo neophodno i numeričko diferenciranje morao mu se dodati novi uređaj. Britanski matematičar Alan Tjuring (Alan Turing) je oštro kritikovao takav pristup građenju računara. On je bio zagovornik koncepcije koja je danas u osnovi računarske tehnologije: *računar ne treba da bude zbirka specijalizovanih uređaja, već univerzalna mašina čije će mogućnosti zavisiti isključivo od sposobnosti programera*. Grubo govoreći, svi, čak i najmoderniji računari koje danas koristimo, su sagrađeni od samo tri vrste elektronskih kola (*logičkih kapija*): "ne", "i" i "ili". Milioni ovih kola organizovanih na pravi način osnov su računara kao univerzalne mašine. Ideju ćemo demonstrirati na primeru hardvera za sabiranje dva binarna broja.

Primer 3.20. Posmatramo sabiranje dva binarna broja:

```
   111        ← prenos
  1010
 +11111
 ------
101001
```

Pretpostavimo sada da su nam data dva binarna broja sa istim brojem cifara (ukoliko polazni brojevi nemaju isti broj cifara, onom sa manje cifara možemo dopisati nule na početak). Prilikom sabiranja "peške" osnovni korak se sastoji u tome da saberemo dva bita i prenos iz prethodnog koraka. Kao rezultat dobijamo jedan bit rezultata i jedan bit prenosa.



Slika 3.6.13: Polusabirač

Evo slikovitog prikaza ove priče, kao i tablice sabiranja sa prenosom:

i-ti korak

prenos →	c_{i+1}	c_i	c_{i-1}	..	c_1
	x_n ... x_{i+1}	x_i	x_{i-1}	..	x_1
+	y_n ... y_{i+1}	y_i	y_{i-1}	..	y_1
	z_n ... z_{i+1}	z_i	z_{i-1}	..	$z1$

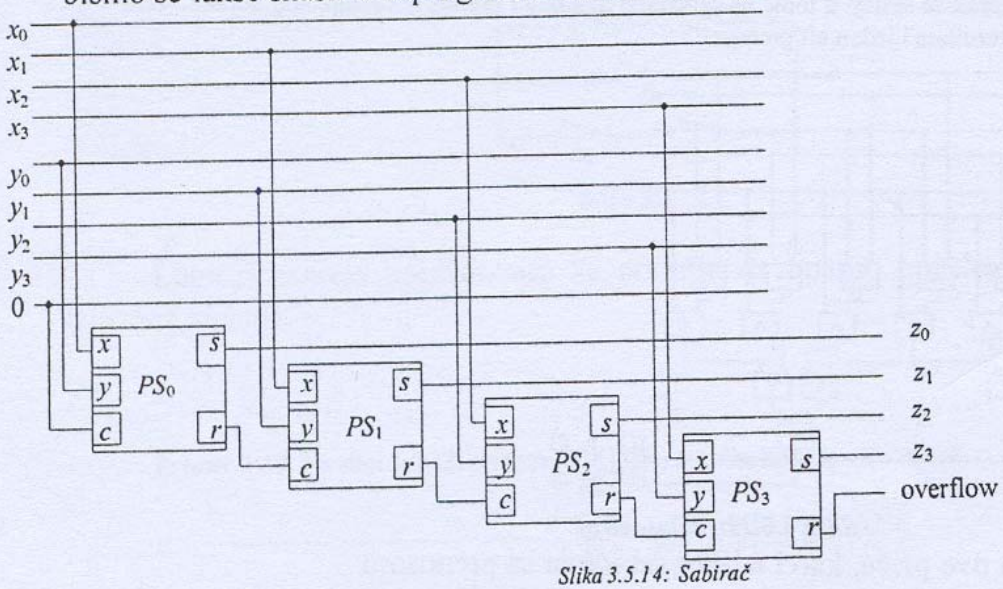
			zbir	novi prenos
x	y	c	$s(x, y, c)$	$r(x, y, c)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Prema priči o predstavljanju Bulovih operacija osnovnim Bulovim operacijama, dobijamo da se Bulove operacije $s(x, y, c)$ i $r(x, y, c)$ koje su date gornjom tabelom mogu predstaviti ovako:

$$s(x, y, c) = (\neg x \wedge \neg y \wedge c) \vee (\neg x \wedge y \wedge \neg c) \vee (x \wedge \neg y \wedge \neg c) \vee (x \wedge y \wedge c)$$
$$r(x, y, c) = (\neg x \wedge y \wedge c) \vee (x \wedge \neg y \wedge c) \vee (x \wedge y \wedge \neg c) \vee (x \wedge y \wedge c).$$

Elektronsko kolo kojim se ove dve Bulove operacije realizuju odednom se zove *polusabirač* i predstavljeno je na Slici 1. Ulaz u polusabirač su tri bita (x, y i c), a izlaz čine dva bita (zbir s i novi prenos r). Napomenimo da je na Slici 1 prikazana veoma naivna verzija polusabirača. Stvarni polusabirači u stvari izgledaju drugačije, zato što se pre implementacije u harvderu vrše razne optimizacije kako bi se uštedelo na komponentama. No, osnovna ideja je ista. Hardver kojim se realizuje sabiranje binarnih brojeva iste dužine se zove *sabirač* i sastoji se od nekoliko polusabirača. Sabirač 4-bitnih binarnih brojeva je prikazan na Slici 3.5.14. Ulaz u sabirač predstavljaju binarni brojevi $x_3x_2x_1x_0$ i $y_3y_2y_1y_0$, dok se na izlazu dobija rezultat $z_3z_2z_1z_0$, i informacija o tome da li je rezultat duži od četiri bita (linija *overflow*) nije ništa drugo do prenos poslednjeg sabiranja.

Polusabirači PS_0, PS_1, PS_2 i PS_3 su identični. Označeni su brojevima samo zato da bismo se lakše snašli u nepreglednim šumama linija.



Slika 3.5.14: Sabirač

Problem. Napraviti polusabirač polazeći samo od kola za operacije $\wedge$, $\vee$ i $\oplus$, gde je poslednja operacija definisana tabelom

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

(Uputstvo: $s(x, y, c)$ se realizuje veoma lako, dok je
$r(x, y, c) = (x \wedge y) \vee ((x \oplus y) \wedge c)$.)

Sabirači

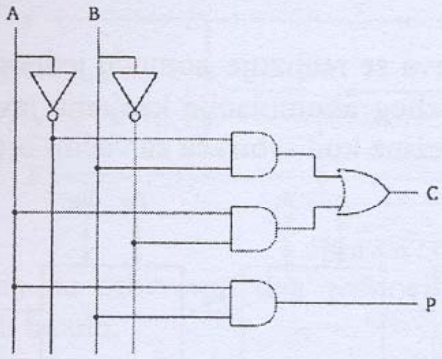
Sabirači su kombinatorne mreže koje se koriste pri realizaciji sabiranja. Sabirači mogu biti binarni i dekadni. Binarni polusabirač je kombinatorna mreža koja vrši sabiranje jednocifrenih hinarnih brojeva. Na osnovu tabele se lako formira izraz za funkcije C (koja predstavlja cifru zbira) i P (koja predstavlja prenos u narednu poziciju):

A	B	C	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

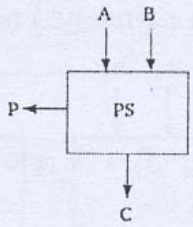
Istinitosna tablica za binarni polusabirač

Odatle se dobijaju analitički izrazi funkcija C i P:

$$C = \bar{A}B + A\bar{B} \qquad P = AB.$$



Implementacija polusabirača



Oznaka za polusabirač

Kombinatorna mreža koja realizuje sabiranje dve binarne cifre višecifrenog binarnog broja uz uzimanje u obzir prenosa sa prethodnog mesta se naziva (pun) sabirač. Tabela istinitosnih vrednosti sabirača je prikazana na slici.

P_p	A	B	C	P_t
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Istinitosna tablica za binarni sabirač

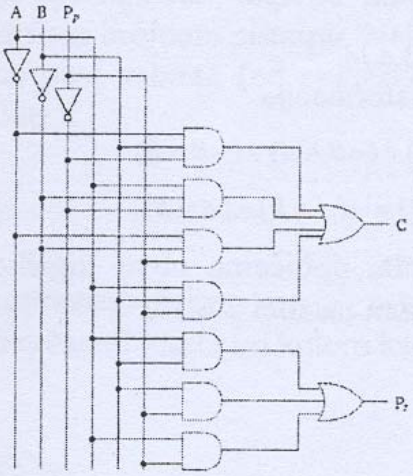
Na osnovu tabele se formira izraz za funkcije C i P_t:

$$C = \overline{A}B\overline{P_p} + A\overline{B}\overline{P_p} + \overline{A}\overline{B}P_p + ABP_p$$

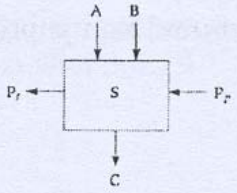
$$P_t = AB\overline{P_p} + \overline{A}BP_p + A\overline{B}P_p + ABP_p$$

Funkcija P_t može da se minimizira jednostavnom algebarskom transformacijom:

$$\begin{aligned} P_t &= AB\overline{P_p} + \overline{A}BP_p + A\overline{B}P_p + ABP_p + ABP_p + ABP_p \\ &= AB(\overline{P_p} + P_p) + BP_p(\overline{A} + A) + AP_p(\overline{B} + B) \\ &= AB + BP_p + AP_p \end{aligned}$$



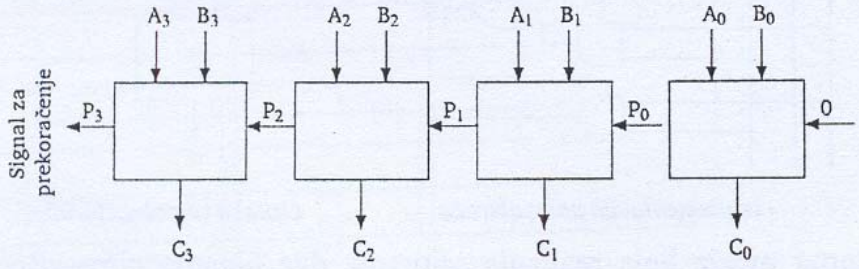
Implementacija sabirača



Oznaka za sabirač

Višecifreni sabirači

Sabiranje višecifrenih binarnih brojeva se realizuje pomoću jednog sabirača za svako binarno mesto. U ovom slučaju se zbog akumulacije kanjenja javlja značajno kašnjenje između bita najmanje i najveće težine kod sabirača sa većim brojem mesta.

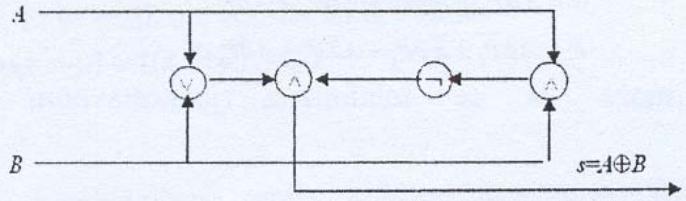


Još jedan oblik sabirača i polusabirača

Poštujući analogiju sa sabiranjem decimalnih brojeva, gde se javlja takozvano "prenošenje cifre" u sledeći decimalni niz, definisaćemo *cifru sume s* i *prenos cifre c.*

A	*B*	*s*	*c*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ovako definisano *s* odgovara iskjučujuce ili (ExOR), (što cemo označiti sa $A \oplus B$), dok za *c* odgovara konjukcija (AND ili $\wedge$). Poštujući ove zaključke, konstruisaćemo odvojena logička kola, koja kao izlazni signal imaju *s* odnosno *c* (prikazana su na slikama 3.6.15. i 3.6.16 respektivno).



Slika 3.6.15.

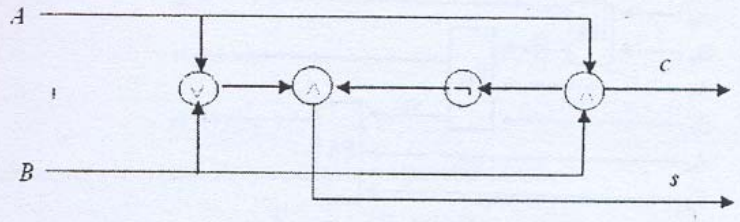
Slika 3.6.16.

U ovom slučaju izraza *s* iskorišćena je transformacija

$$A \oplus B = (\neg A \wedge B) \vee (\neg B \wedge A) = (\neg A \wedge A) \vee (\neg A \wedge B) \vee (\neg B \wedge A) \vee (\neg B \wedge B)$$

$$= (\neg A \vee \neg B) \wedge (A \vee B) = \neg(A \wedge B) \wedge (A \vee B).$$

Ukoliko kombinujemo ova dva kola, dobićemo novo logičko kolo sa dva *izlaza,* takozvani *polusabirač,* koji je prikazan na slici 3.6.17.



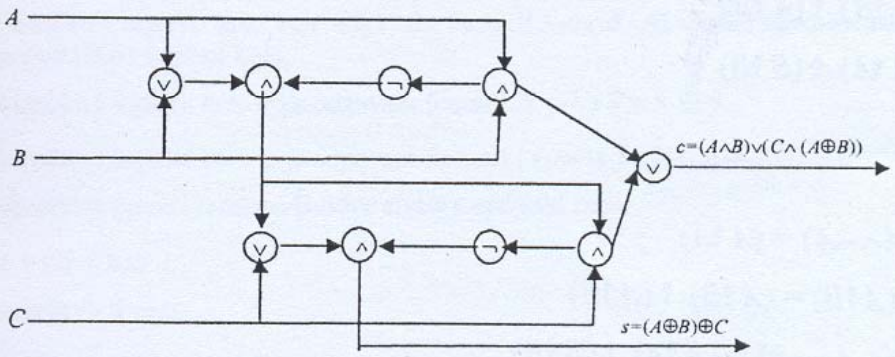
Slika 3.6.17.

Ako želimo da saberemo dva jednocifrena broja A i B, preko prenosa C, dobićemo sledeću tabelu.

A	B	C	s	c
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	1	0
1	1	0	0	1
0	1	0	1	0
1	0	0	1	0
0	0	0	0	0

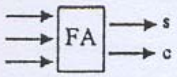
Prema tome, s odgovara formuli $(C\wedge\neg(A\oplus B))\vee((A\oplus B)\ \wedge\neg C)$, čiji je logički ekvivalent $(A\oplus B)\oplus C$.

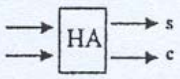
Prenos cifre c odgovara formuli $(A\wedge B)\vee(C\wedge\ (A\oplus B))$. Ukoliko želimo da konstruišemo kolo sa dva *izlaza,* dobićemo kolo kao na slici 3.6.18., takozvani *puni sabirač.*



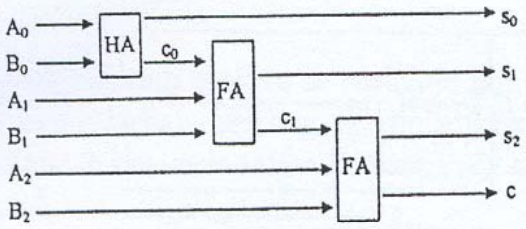
Slika 3.6.18.

Sada možemo konstruisati logičko kolo za sabiranje dva trocifrena broja, predstavljena u binarnom brojnom sistemu. Neka su ti binarni brojevi $A_2\,A_1\,A_0$ i $B_2\,B_1\,B_0$. Sa FA označimo puni sabirač (od engleske reči full-adder), a sa HA označimo polusabirač (half-adder).





Krajnju sumu predstavlja broj $cs_2s_1s_0$, u kome je c prenos iz poslednje pozicije. Ovaj zbir se može izračunati logičkim kolom kao na slici 3.6.19.



Slika 3.6.19.

3.6.8. Pun sistem funkcija

Skup funkcija $F=\{f_1, f_2, ..., f_n\}$ naziva se *pun sistem funkcija* ako se proizvoljna funkcija algebre logike može predstaviti pomoću funkcija iz ovog skupa.

F je minimalan ako skup $G = F\setminus\{f_i\}$ (za neko i) ne predstavlja pun sistem funkcija.

Neki od minimalnih punih sistema funkcija:

1. Negacija i konjunkcija: $F = \{\neg, \wedge\}$.

2. Negacija i disjunkcija: $F = \{\neg, \vee\}$.

3. NAND: $F = \{\uparrow\}$:

- $\neg A = (A \uparrow A)$
- $A \wedge B = (A \uparrow B) \uparrow (A \uparrow B)$
- $A \vee B = (A \uparrow A) \uparrow (B \uparrow B)$

Dokaz:

$\neg A = \neg(A \wedge \neg A) = (A \uparrow A)$

$A \wedge B = \neg(A \uparrow B) = (A \uparrow B) \uparrow (A \uparrow B)$

$A \vee B = \neg(\neg A \wedge \neg B) = (A \uparrow A) \uparrow (B \uparrow B)$

4. NOR: $F=\{\downarrow\}$:

- $\neg A = (A \downarrow A)$
- $A \wedge B = (A \downarrow A) \downarrow (B \downarrow B)$
- $A \vee B = (A \downarrow B) \downarrow (A \downarrow B)$

Problem. Napraviti polusabirač polazeći samo od kola za operaciju $\uparrow$.

(Uputstvo: Pokaže se da je $\neg x = 1 \uparrow x$, $x \wedge y = 1 \uparrow (x \uparrow y)$, i $x \vee y = ((1 \uparrow x) \uparrow (1 \uparrow y))$.)

3.7. Zadaci

1. Primenom aksioma i teorema Bulove algebre uprostiti izraz

$$a\bar{b} + c + (\bar{a} + b)\bar{c}\,.$$

$$a\bar{b} + c + (\bar{a} + b)\bar{c} = \overline{(\bar{a} + b)} + c + (\bar{a} + b)\bar{c} = \overline{(\bar{a} + b)\bar{c}} + (\bar{a} + b)\bar{c} = 1.$$

Proveriti da li važi

$$\bar{a}\,b\,\bar{c} = (\bar{a} + \bar{b}\ \bar{c}\,)(b+c)(a+\bar{c}\,)$$

2. Odrediti funkciju F koja utvrđuje jednakost dve računske reči od šest binarnih promenljivih:

$A = a_1\ a_2\ a_3\ a_4\ a_5\ a_6$

$B = b_1\ b_2\ b_3\ b_4\ b_5\ b_6$

Nacrtati logičko kolo koje je određeno jednakošću ove dve binarne reči.

3. Nacrtati prekidačko i logičko kolo koje odgovara formuli $(A\wedge C)\vee(\neg A\vee B)$. Zatim uprostiti taj izraz i nacrtati odgovarajuće prekidačko i logičko kolo.

4. Nacrtati prekidačko i logičko kolo koje odgovara formuli $(A\wedge B\wedge\neg C)\vee(\neg C\wedge\neg A)$. Zatim uprostiti taj izraz i nacrtati odgovarajuće prekidačko i logičko kolo.

5. Tročlana komisija odlučuje većinom glasova, o određenim zakonima. Svaki član pritiska dugme da potvrdi svoj glas "DA". Konstruišimo prekidačko kolo koje će paliti signal "DA", ako i samo ako je većina članova komisije dala glas "DA ".

6. Nacrtati prekidačko i logičko kolo koje odgovara formuli $\neg A\vee(B\wedge A)$. Zatim uprostiti taj izraz i nacrtati odgovarajuće prekidačko i logičko kolo.

7. Nacrtati prekidačko i logičko kolo koje odgovara formuli $\bar{x}\,y + x\,\bar{y} = x \oplus y$.

8. Nacrtati prekidačko i logičko kolo koje odgovara formuli $(A_1\wedge\neg A_2)\vee\neg A_1\vee(A_2\wedge A_3)$.

9. Primenom aksioma i teorema Bulove algebre uprostiti izraz

$a\bar{b} + c + (\bar{a} + b)\bar{c}\,.$

10. Proveriti da li važi

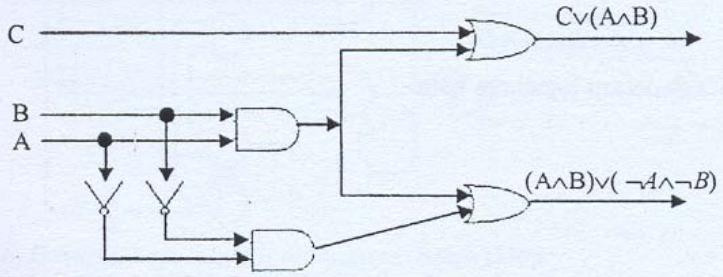
$$\bar{a}\,b\,\bar{c} = (\bar{a} + \bar{b}\ \bar{c}\,)(b+c)(a+\bar{c}\,)$$

11. Nacrtati logičko kolo koje odgovara izrazu $\bar{x}\wedge y\vee x\wedge\bar{y}$.

12. Nacrtati logičko kolo koje odgovara izrazu $(A\wedge(B\vee\overline{C}))\vee(B\wedge C)$.

13. Nacrtati prekidačko kolo koje odgovara izrazu $(A\wedge\overline{C})\vee(\overline{A}\vee B)$.

14. Odrediti izraze koji odgovaraju izlazima u logičkom kolu na slici:



15. Formirati tabelu istinitosti koja odgovara izrazu $(A \wedge (B \vee \overline{C})) \vee (B \wedge C)$.

16. Nacrtati prekidačko kolo koje odgovara ekskluzivnoj disjunkciji $A \oplus B$.

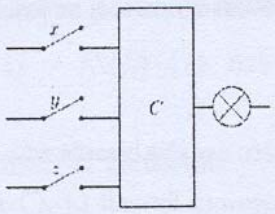
17. Neka je $f(x; y; z) = (x \wedge y) \vee (x \wedge \neg z)$. Izračunati vrednost operacije f za sve mogućnosti argumenata x, y i z. (Napraviti tabelu; koliko različitih redova, osim zaglavlja, treba da ima ova tabela?)

18. Za Bulovu operaciju $f(x; y; z) = (x \wedge y \wedge z) \vee \neg (x \vee y \vee z)$ napraviti odgovarajuću tabelu.

19. Za Bulovu operaciju $f(x; y; z; u) = ((x \vee y) \wedge \neg z) \vee (\neg(x \wedge y) \wedge u)$ napraviti odgovarajuću tabelu. (Za Bulovu operaciju sa četiri argumenta tabela treba da ima 16 redova.)

20. Koliko redova ima tabela za Bulovu operaciju sa n argumenata?

21. Na ulazu u kontrolni uređaj C nalaze se tri prekida·ca x, y i z, a na izlazu sijalica S. Kontrola C pali sijalicu ako su uključena bar dva prekidača (dakle, ili neka dva ili sva tri).



(a) Predstaviti ponašanje kontrolnog uređaja C tabelom (1 = struja protiče; 0 = struja ne protiče);

(b) Dobijenu Bulovu operaciju predstaviti odgovarajućim izrazom.

(c) Skicirati šemu uređaja.

22. Uprostiti algebarske izraze:

A) $a \cdot b + a \cdot b \cdot c + b \cdot c$

B) $a \cdot b + a \cdot \overline{b} \cdot c + b \cdot c$

C) $(a \cdot \overline{b} + c) \cdot (\overline{a} + b) \cdot c$

Rešenje:

A) $a \cdot b + a \cdot b \cdot c + b \cdot c =$ - apsorpcija: $ab + abc = ab \cdot (1 + c) = ab \cdot 1 = ab$

$a \cdot b + b \cdot c =$ - distibutivnost "·" prema "+"

$b \cdot (a + c)$

B) $a \cdot b + a \cdot \overline{b} \cdot c + b \cdot c =$ - distibutivnost "·" prema "+"

$a \cdot (b + \overline{b} \cdot c) + b \cdot c =$ - distibutivnost "+" prema "·"

$a \cdot (b + \overline{b}) \cdot (b + c) + b \cdot c =$ - $b + \overline{b}$ *je 1*

$a \cdot (b + c) + b \cdot c =$

$a \cdot b + a \cdot c + b \cdot c$

C) $(a \cdot \overline{b} + c) \cdot (\overline{a} + b) \cdot \overline{c} =$ - distibutivnost "·" prema "+"

$(a \cdot \overline{b} \cdot \overline{c} + c \cdot c) \cdot (\overline{a} + b) =$ - $c \cdot \overline{c}$ *je 0*

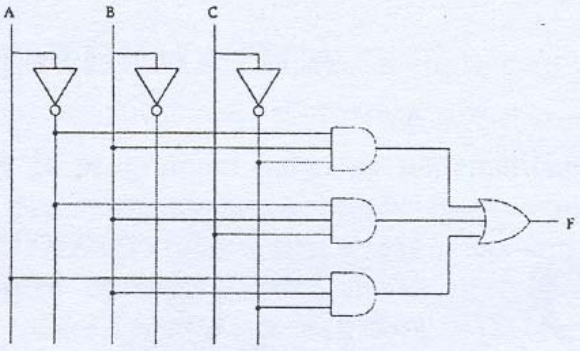
$a \cdot \overline{b} \cdot \overline{c} \cdot (\overline{a} + b) =$ - oba sabirka će imati u sebi proizvod koji je 0

$a \cdot \overline{b} \cdot \overline{c} \cdot \overline{a} + a \cdot \overline{b} \cdot \overline{c} \cdot b = 0$ - prvi proizvod je nula zbog $a \cdot \overline{a}$, a drugi zbog $b \cdot \overline{b}$

23. Pojednostavi sledeću funkciju:

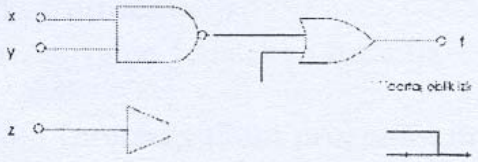
$$F = \overline{\overline{A \wedge B \wedge C \wedge D}}$$

24. Odrediti logičku funkciju koja odgovara sledećem logičkom kolu:



Formirati tabelu istinitosti za tu logičku funkciju.

25. Dato je logičko kolo prikazano na donjoj slici.

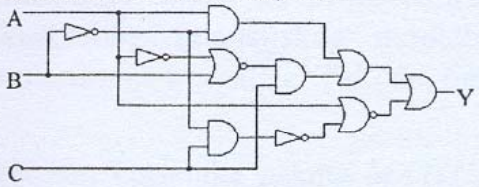


a) Napiši logičku funkciju koje ono realizuje.

b) Nađi vrednosti funkcije f koja odgovara logičkom kolu sa slike u tabeli

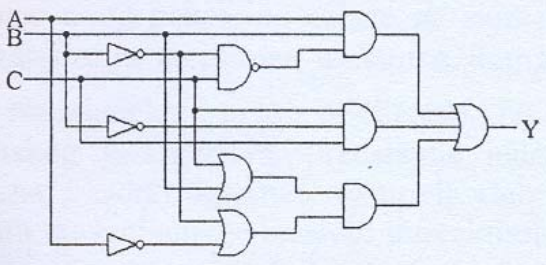
x	y	z	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

26. Dat je logički sklop kao na sledećoj slici:



a) Odrediti koju logičku funkciju realizuje sklop.
b) Pojednostaviti dobijenu funkciju.
c) Na osnovu pojednostavljene funkcije realizirati sklop iste funkcionalnosti kao i početni sklop.

27. Dat je logički sklop kao na sledećoj slici:



a) Odrediti koju logičku funkciju realizuje sklop.
b) Pojednostaviti dobijenu funkciju.
c) Na osnovu pojednostavljene funkcije realizovati sklop iste funkcionalnosti kao i početni sklop.