

2. MATEMATIČKO LOGIČKE OSNOVE RAČUNARA

2.1. BROJNI SISTEMI

2.2. KODIRANJE

2.3. LOGIČKA ALGEBRA

**2.4. PRIMJENA LOGIČKE ALGEBRE U
INFORMATICI**

2.1. BROJNI SISTEMI

Brojni sistem je način označavanja ili izražavanja brojeva, nizova znakova ili naziva. Usporedo s razvojem pisma kroz čovjekovu istoriju razvijali su se i različiti brojni sistemi koji se po strukturi dijele na:

1. aditivne,
2. aditivno-multiplikativne.

Aditivni sistem je niz znakova u kojima je broj jednak zbiru znakova od kojih je sastavljen, npr. kao kod starih Rimljana:

$$\text{XXXVII} = 10 + 10 + 10 + 5 + 2 = 37$$

Ovakvi sistemi nisu omogućavali računске operacije kao što omogućavaju aditivno-multiplikativni brojni sistemi, kod kojih pojedini brojevi (znamenke) predočavaju veličinu pojedinih grupa datog niza s kojom se pomnože i sve grupe saberu:

$$\text{"stotinu četrdeset i pet"} = 1 * 100 + 4 * 10 + 5 * 1 = 145$$

Osnov **aditivno-multiplikativnog brojnog** sistema je **BAZA**, koja ulazi kao multiplikant u komponente oznake ili naziva broja. Danas je u općoj upotrebi **DEKADNI BROJNI SISTEM**, aditivno-multiplikativni brojni sistem s **OSNOVOM (BAZOM)** deset (10).

Općenito se broj "N" u aditivno-multiplikativnom sistemu s osnovom "B" može napisati u obliku:

$$N_B = a * B^n + a * B^{n-1} + \dots + a * B^2 + a * B^1 + a * B^0$$

N je broj brojnog sistema s bazom "B" izražen brojem „a“,

a je bilo koji znamenka brojnog sistema u opsegu od 0 do B-1, a to su u dekadnom sistemu znakovi 0,1,2,3,4,5,6,7,8 i 9 koji predstavljaju brojni raspon unutar baze „B“,

B je baza (osnova) brojnog sistema, koja u dekadnom sistemu iznosi 10 i ukazuje da u sistemu veličine grupe ima 10 različitih stanja na jednom mjestu za jednu znamenku „a“.

Navedenim izrazom izračunava se dekadna vrijednost broja "N" bilo kojeg brojnog sistema.

2.1.1. DEKADNI BROJNI SISTEM

Ljudi broje i računaju po dekadnom brojnom sistemu i vrlo često ne razmišljaju da je nastao na osnovu deset čovjekovih prstiju s kojima se pomagao u računanju.

Koristi se poziciono označavanje brojeva npr. broj 1953 sadrži četiri znamenke od kojih svaka u ovisnosti o mjestu gdje se nalazi označava broj jedinica, desetica, stotica itd. Svakoj znamenki pridružuje se njena **TEŽINA** koja ovisi o njenom mjestu u broju. Najmanju težinu ima znamenka na desnom kraju broja, a najveću težinu ima znamenka na lijevom kraju broja.

Dekadni broj tumači se na slijedeći način:

$$1953_{10} = 1 * 1000 + 9 * 100 + 5 * 10 + 3 * 1 = 1 * 10^3 * 9 * 10^2 + 5 * 10^1 + 3 * 10^0$$

0,1,2,3 su težinske vrijednosti

Osnova sistema je broj **10** a težinska vrijednost znamenke je eksponent osnove u skladu s udaljenosti znamenke od mjesta najmanje težine.

S negativnim eksponentom mogu se prikazati brojevi manji od jedan kao na primjer:

$$0,12 = 1 * 10^{-1} + 2 * 10^{-2}$$

Često se u svakodnevnoj praksi opisuju događaji kojima je osnov brojanja drugačiji, npr. sunca ima ili nema, živ ili mrtav, mokar ili suh i slični. Tim opisima pridružena su **DVA** različita stanja. Elektronika u tom pogledu nije iznimka. Vrlo je složen elektronski sklop koji bi amplitude signala razlikovao u 10 nivoa veličine. Sistem bi bio neotporan na svaku smetnju koja bi izmijenila veličinu amplitude.

Jednostavnije je definisati dvije situacije, impulsa ima (pozitivan impuls) ili ga nema (odsustvo ili negativan impuls). Simbolička oznaka postojanja impulsa je "**I**", a oznaka nepostojanja je "**0**". Sklop koji razlikuje postojanje i nepostojanje impulsa mnogo je jednostavniji, te se stoga računari dizajniraju da računске i logičke operacije vrše s brojnim sistemom koji koristi znamenke "**0**" i "**I**" i ima osnovu "**2**".

Takav sistem naziva se **BINARNI BROJNI SISTEM** u kojemu se na mjestu znamenke mogu pojaviti "0" ili "1", što predstavlja 50% vjerovatan događaj za pojavu jedne od znamenki. To znači da mjesto znamenke sadrži količinu informacije od 1 bit-a.

Nadalje u opisu za brojeve koji nemaju oznaku osnove podrazumijeva se osnova 10, a za brojeve drugih brojnih sistema označit će im se pripadna osnova ili će se na kraju broja dopisati početno slovo sistema kojemu pripada (10B, 16H, ...).

Za slučaj decimalnog broja 1845,34 imamo puni zapis:

$$1845,34 = 1 * 10^3 + 8 * 10^2 + 4 * 10^1 + 5 * 10^0 + 3 * 10^{-1} + 4 * 10^{-2}$$

2.1.2. BINARNI BROJNI SISTEM

Kod dekadnog brojnog sistema brojimo "nula, jedan, dva, tri, četiri, pet, šest, sedam, osam, devet, DESET ", a "deset" je u suštini "0 jedan dalje". Analogno navedenom može se izgraditi binarni sistem brojeva prema primjeru koji slijedi.

| Dekadno | Binarno | |
|----------------|----------------|---------------------------|
| 0 | 0 | |
| 1 | 1 | Slijedi „0 jedan dalje“ |
| 2 | 10 | |
| 3 | 11 | Slijedi „00 jedan dalje“ |
| 4 | 100 | |
| 5 | 101 | |
| 6 | 110 | |
| 7 | 111 | Slijedi „000 jedan dalje“ |
| 8 | 1000 | |

Opći oblik za pretvaranje binarnog broja u dekadni je:

$$N_{10} = a * 2^n + a * 2^{n-1} + \dots + a * 2^2 + a * 2^1 + a * 2^0$$

N je broj brojnog sistema izražen znamenkama „**a**“,

a znamenke sistema: „0“ ili „1“,

2 je baza (osnova) brojnog sistema.

2.1.2.1. Pretvaranje dekadnog broja u binarni broj

Pretvaranje dekadnog broja u binarni broj može se izvršiti na dva načina:

1. *dijeljenjem s 2* ili
2. *pomoću tablica.*

Pretvaranje dijeljenjem sa dva je postupak koji se općenito može primijeniti za pretvaranje dekadnih brojeva, u brojeve bilo kojeg sistema, dijeljenjem sa osnovicom tog sistema.

Pretvaranje dijeljenjem s dva, vrši se sukcesivnim dijeljenjem s 2. Ostatak dijeljenjem predstavljaju brojke 0 ili 1. Kad se dijeljenjem dođe do operacije $1 : 2 = 0$ i 1 ostatak, dijeljenje je završeno. Čitanje rezultata vrši se odozdo prema gore. Na ovaj način se vrši pretvaranje cijelih brojeva dekadnog brojnog sistema u binarni brojni sistem.

Primjer

$(125)_{10} = ()_2$

| | | | | | | |
|-----|---|---|---|----|---------|---|
| 125 | : | 2 | = | 62 | Ostatak | 1 |
| 62 | : | 2 | = | 31 | " | 0 |
| 31 | : | 2 | = | 15 | " | 1 |
| 15 | : | 2 | = | 7 | " | 1 |
| 7 | : | 2 | = | 3 | " | 1 |
| 3 | : | 2 | = | 1 | " | 1 |
| 1 | : | 2 | = | 0 | " | 1 |

↑
Čitanje

$(125)_{10} = (1111101)_2$

Pretvaranje decimalnih brojeva vrši se sukcesivnim množenjem s 2 i upisivanjem dobivenog cjelobrojnog dijela (0 ili 1) kao brojke binarnog broja.

Množenje se nastavlja dok se ne dobije rezultat u decimalnom dijelu = 0 ili dovoljno mala vrijednost.

Primjer:

$(0,6875)_{10} = ()_2$

| | | | | | |
|--------|---|---|---|-------|---|
| 0,6875 | * | 2 | = | 1,375 | 1 |
| 0,375 | * | 2 | = | 0,75 | 0 |
| 0,75 | * | 2 | = | 1,5 | 1 |
| 0,5 | * | 2 | = | 1 | 1 |

↑
Čitanje

Smjer čitanja je odozgo prema dolje.

$(0,6875)_{10} = (0,1011)_2$

Pretvaranje mješovitih brojeva vrši se tako, da se prvo pretvori cjelobrojni dio broja po pravilima za pretvaranje cjelobrojnih brojeva, a zatim decimalni dio broja po pravilima za razlomljene brojeve.

Pretvaranje pomoću tablica vrši se korištenjem tablice koja predstavlja vrijednosti mjesta binarnog brojnog sistema.

Ako se ta vrijednost može prikazati kao 2^n , gdje je n broj iz skupa prirodnih brojeva, onda se uzimanjem prve niže vrijednosti i prikazivanjem te vrijednosti u obliku 2^n ; a zatim pridruživanjem vrijednosti 1 za vrijednosti 2^n koje su upotrijebljene za prikaz tog broja odnosno 0 , ako vrijednosti 2^n nisu upotrijebljene prilikom prikazivanja tog broja, može pretvoriti broj iz dekadnog u binarni oblik.

| Vrijednosti 2^n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-------------------|---|---|---|---|----|----|----|-----|-----|-----|
| 2^n | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |

Primjer:

$$(156)_{10} = ()_2$$

$$\begin{array}{cccccccc} \longrightarrow & \textcircled{1} & \textcircled{0} & \textcircled{0} & \textcircled{1} & \textcircled{1} & \textcircled{1} & \textcircled{0} & \textcircled{0} \\ & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ & 1 \cdot 2^7 & 0 \cdot 2^6 & 0 \cdot 2^5 & 1 \cdot 2^4 & 1 \cdot 2^3 & 1 \cdot 2^2 & 0 \cdot 2^1 & 0 \cdot 2^0 \\ & \text{(156)}_{10} & = & \text{(10011100)}_2 \end{array}$$

2.1.2.2. Pretvaranje binarnog broja u dekadni broj

Pretvaranje binarnih brojeva u dekadne, može se izvršiti na više načina. Jedan od postupaka je i sabiranje mjesnih vrijednosti.

Primjer:

$$\begin{array}{cccccc} (1 & 0 & 1 & 1 & 0 &)_2 & \text{broj} \\ 16 & 8 & 4 & 2 & 1 & & \text{mjesne vrijednosti} \end{array}$$

$$16 \times 1 + 8 \times 0 + 4 \times 1 + 2 \times 1 + 0 \times 1 = 22$$

$$(10110)_2 = (22)_{10}$$

Primjer:


Broj $(101100,11)_2$ u binarnom sistemu ima vrijednost:

$$\begin{aligned} (101100,11)_2 &= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 0x2^0 \\ &+ 1x2^{-1} + 1x2^{-2} = (44,75)_{10} \end{aligned}$$

Druga metoda je takozvani cik - cak postupak, koji je univerzalan za pretvaranje brojeva iz bilo kojeg brojnog sistema u dekadni.

Prava linija u ovom cik - cak postupku odnosno mreži, znači množenje, a kosa sabiranje.

Primjer:

$$(10101000011)_2$$
$$(1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1)_2 = (1347)_{10}$$


2.1.2.3. Aritmetika u binarnom brojnom sistemu

Pravila za osnovne računске operacije su:

a) sabiranje:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ (1 prenos)}$$

Binarno sabiranje obavlja se isto kao i decimalno sabiranje, osim što se prenos na slijedeće značajno mjesto obavlja nakon postignutog zbira $2(1+1)$.

b) oduzimanje:

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$10 - 1 = 1 \text{ (1 prenos)}$$

Binarno oduzimanje obavlja se kao i decimalno oduzimanje, osim što se posuđuje 1 od bita veće težine.

c) množenje:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Kod binarnog množenja djelomičan umnožak pomiče se za jedno mjesto udesno po navedenim pravilima, a zatim se umnošci saberu.

d) dijeljenje:

$$0 / 0 = \text{nedjeljivo}$$

$$1 / 0 = Y$$

$$0 / 1 = 0$$

$$1 / 1 = 1$$

Dijeljenje binarnih brojeva obavlja se na isti način kao i dekadnih.

Praktično se dijeljenje svodi na množenje i oduzimanje.

Primjeri:

1. Sabiranje:

$$\begin{array}{r} \text{a) } 1001011_2 = 75_{10} \\ + 1011110_2 = 94_{10} \\ \hline 10101001_2 = 169_{10} \end{array} \qquad \begin{array}{r} \text{b) } 1111001_2 = 121_{10} \\ + 110100_2 = 52_{10} \\ \hline 10101101_2 = 173_{10} \end{array}$$

2. Množenje:

a) $11001_2 \times 1110_2 = 101011110_2$ ($25 * 14 = 350$)₁₀

$$\begin{array}{r} 11001 \\ 11001 \\ 11001 \\ \hline 00000 \\ \hline 101011110 = 28 + 26 + 24 + 23 + 22 + 31 = 350 \end{array}$$

b) $111_2 \times 1111_2 = 1101001_2$ ($7 * 15 = 105$)₁₀

$$\begin{array}{r} 111 \\ 111 \\ 111 \\ \hline 111 \\ \hline 1101001 = 26 + 25 + 23 + 20 = 105_{10} \end{array}$$

3. Oduzimanje:

$$\begin{array}{r} \text{a) } 1110_2 = 14_{10} \\ - 1101_2 = 13_{10} \\ \hline 0001_2 = 1_{10} \end{array} \qquad \begin{array}{r} \text{b) } 1110_2 = 14_{10} \\ - 100_2 = 4_{10} \\ \hline 1010_2 = 10_{10} \end{array}$$

4. Dijeljenje:

11011 : 11 = 1001 Provjera: 27 : 3 = 9
 - 11
 00011
 - 11
 00

2.1.3. OKTALNI BROJNI SISTEM

Osim binarnog brojnog sistema u računarima se koristi i OKTALNI brojni sistem s bazom **8** i koji koristi osam znamenki dekadnog brojnog sistema i to znamenke **0,1,2,3,4,5,6** i **7**. Brojevi ovog sistema prikazani su u narednoj tabeli:

| | | | | | | | | |
|----------------|-----------|----|----|----|----|----|----|----|
| dekadno | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| oktalno | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| dekadno | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| oktalno | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| dekadno | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| oktalno | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |

Opći oblik za pretvaranje oktalnog broja u dekadni je:

$$N_{10} = a * 8^n + a * 8^{n-1} + \dots + a * 8^2 + a * 8^1 + a * 8^0$$

N je broj brojnog sistema izražen znamenkama „**a**“,

a znamenke sistema: **0, 1, 2, 3, 4, 5, 6** ili **7**,

8 je baza (osnova) brojnog sistema.

2.1.3.1. Pretvaranje dekadnog broja u oktalni broj

Pretvaranje dekadnog broja u oktalni broj može se vršiti univerzalnim postupkom, sukcesivnim dijeljenjem s bazom sistema. U ovom slučaju to je broj **8**:

Primjer:

$$(1016)_{10} = ()_8$$

| | | | | | | | |
|------|---|---|---|-----|---------|---|--------------|
| 1016 | : | 8 | = | 127 | Ostatak | 0 | ↑ Čitanje |
| 127 | : | 8 | = | 15 | “ | 7 | |
| 15 | : | 8 | = | 1 | “ | 7 | |
| 1 | : | 8 | = | 0 | “ | 1 | |

$$(1016)_{10} = (1770)_8$$

2.1.3.2. Aritmetika u oktalnom brojnom sistemu

Operacije u oktalnoj aritmetici izvode se na slijedeći način:

Primjeri zbrajanja:

$$\begin{array}{r|l}
 (1025)_8 & 5+6=11; 11 : 8 = 1 \text{ (prenos)} + 3 \text{ (piše se)} \\
 + (536)_8 & (2+3)+1=6; \text{ (manje od baze, samo se piše)} \\
 \hline
 (1563)_8 & 0+5=5 \\
 & 1+0=1
 \end{array}$$

Sabiranje u oktalnom brojnom sistemu obavlja se sabiranjem znamenki kao i kod dekadskog brojnog sistema. Ukoliko je zbir veći od **8** (baze), dijelimo ga sa bazom (**8**) te rezultat prenosimo za slijedeće sabiranje znamenki, a ostatak predstavlja znamenku rezultata zbrajanja.

Primjeri oduzimanja:

$$\begin{array}{r|l}
 (1025)_8 & 8+5=13; 13-6=7; \text{ (prenos 1)} \\
 - (536)_8 & 1+3=4; 8+2=10; 10-4=6; \text{ (prenos 1)} \\
 \hline
 (267)_8 & 1+5=6; 8+0=8; 8-6=2; \text{ prenos 1} \\
 & 1-1=0
 \end{array}$$

Oduzimanje u oktalnom brojnom sistemu obavlja se kao i u dekadnom brojnom sistemu. Na primjeru je pojašnjena ova operacija. Prvo oduzimamo $5 - 6$. S obzirom da je 5 manja vrijednost dodajemo bazu **8**; $5 + 8 = 13$. Sada oduzimamo $13 - 6 = 7$ i bilježimo 1 prenos. Dodajemo ovaj prenos slijedećoj znamenki 3: $1+3=4$. Sada oduzimamo $2-4$, s obzirom da je 2 manje od 4 posuđujemo bazu **8** i imamo $(8+2)-4 = 6$ i bilježimo prenos. Dodajemo ovaj prenos slijedećoj znamenki 5: $1 + 5 = 6$. Kad posudimo bazu **8** oduzimamo $8 - 6 = 2$ i bilježimo 1 prenos. Dodajemo 1 iz prenosa i oduzimamo $1 - 1 = 0$.

Rezultat predstavljaju znamenke koje su rezultati oduzimanja čitane odozdo prema gore: 267.

Primjeri množenja:

| | |
|---------------------|--|
| $(325)_8 * (167)_8$ | $6 * 5 = 30 : 8 = 3$ (prenos) 6 (ostatak se piše) |
| $(325)_8$ | $6 * 2 = 12 + 3 = 15 : 8 = 1$ (prenos) 7 (ostatak) |
| $(2376)_8$ | $6 * 3 = 18 + 1 = 19 : 8 = 2$ (prenos) 3 (ostatak) |
| $(2723)_8$ | 2376 |
| $(61403)_8$ | $7 * 5 = 35 : 8 = 4$ (prenos) 3 (ostatak) |
| | $7 * 2 = 14 + 4 = 18 : 8 = 2$ (prenos) 2 (ostatak) |
| | $7 * 3 = 21 + 2 = 23 : 8 = 2$ (prenos) 7 (ostatak) |
| | 2723 |

Množenje u oktalanom brojnom sistemu obavlja se množenjem svake znamenke jednog broja sa svim znamenkama drugog broja. Rezultati množenja se potpisuju pomicanjem za jedno mjesto udesno. Pojasnit ćemo na primjeru. Množimo 1×325 i rezultat je 325 potpisujemo za sabiranje. Sada množimo znamenku 6 sa svim znamenkama broja 325. Dobijemo $6 * 5 = 30$. Broj 30 dijelimo s bazom 8 i imamo 6 ostatak i 3 prenos. Množimo znamenku $6 * 2 = 12$ i dodamo 3 iz prenosa $12 + 3 = 15$. Dijelimo s bazom i dobijemo $15 : 8 = 1$ prenos i 7 ostatak. Na kraju množimo $6 * 3 = 18$ i dodamo prenos $1 = 18 + 1 = 19$. Dijelimo $19 : 8 = 2$ prenos i 3 ostatak. Više nema znamenki za množenje te čitamo rezultat uzimajući zadnji prenos i sve ostatke odozdo nagore: 2376. Potpišemo pomicanjem za jedno mjesto udesno. Na isti način množimo znamenku 7 s brojem 325. Rezultat 2723 potpišemo i sve saberemo.

Primjeri dijeljenja:

$$(61406)_8 : (32)_8 = (1717)_8$$

| |
|-------|
| -32 |
| 274 |
| -266 |
| 0060 |
| - 32 |
| 266 |
| - 266 |
| 000 |

Kao u dekadnom brojnom sistemu neophodno je voditi računa o bazi brojnog sistema.

2.1.4. HEKSADEKADNI BROJNI SISTEM

Kod heksadecimalnog brojnog sistema osnova sistema je **16**, te se pored poznatih oblika znamenki **0,1,2,3,4,5,6,7,8** i **9** za preostale znamenke sistema koriste slova **A,B,C,D,E** i **F** kako se za brojeve veće od 9 ne bi koristila dva znaka.

Dakle, znamenke heksadecimalnog sistema su od **0** do **F** po heksadekadnom označavanju, odnosno od **0** do **15** po dekadnom shvatanju njihove vrijednosti. Brojevi heksadecimalnog sistema prikazani su u narednoj tabeli:

| | | | | | | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Dekadno | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Heksadekadno | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Dekadno | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Heksadekadno | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| Dekadno | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| Heksadekadno | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |

Opći oblik za pretvaranje heksadecimalnog broja je:

$$N_{10} = a * 16^n + a * 16^{n-1} + \dots + a * 16^2 + a * 16^1 + a * 16^0$$

N je broj brojnog sistema izražen znamenkama „**a**“,

a znamenke sistema: **0,1,2,3,4,5,6,8,9,A,B,C,D,E** ili **F**,

16 je baza (osnova) brojnog sistema.

2.1.4.1. Pretvaranje dekadnog broja u heksadecimalni broj

Pretvaranje dekadnog broja u heksadecimalni vrši se dijeljenjem s osnovicom **16**.

$$(508)_{10} = ()_{16}$$

| | | | | | | | |
|-----|---|----|---|----|------------|----------|--------------|
| 508 | : | 16 | = | 31 | Ostatak 12 | C | ↑ Čitanje |
| 31 | : | 16 | = | 1 | Ostatak 15 | F | |
| 1 | : | 16 | = | 0 | Ostatak 1 | 1 | |
| | | | | | | | |

$$(508)_{10} = (1FC)_{16}$$

2.1.4.2. Pretvaranje oktalnog zapisa broja u heksadecimalni zapis broja i obrnuto

Dijeljenjem binarnog broja u grupe po 4 znamenke u grupi može se vrlo jednostavno izvršiti njegova pretvaranje u heksdekadni.

$$\begin{array}{ccccccc}
 1101001011111 & = & 0001 & 1010 & 0101 & 1111 & \\
 & & | & | & | & | & \\
 & & 1 & A & 5 & F & = 1A5F_{16}
 \end{array}$$

Prvoj grupi predhode "0" da bi se popunila i bila vjerodostojnija u prikazu, a što je matematički ispravno. Pretvaranje heksadecimalnog broja u binarni vrši se obratnim postupkom:

$$\begin{array}{ccccccc}
 1A5F_{16} & = & 1 & A & 5 & F & \\
 & & | & | & | & | & \\
 & & 0001 & 1010 & 0101 & 1111 & = \\
 & = & 2^{12} & + 2^{11} & + 2^9 & + 2^6 & + 2^4 & + 2^3 & + 2^2 & + 2^1 & + 2^0 = \\
 & = & 1 \times 16^3 & + 10 \times 16^2 & + 5 \times 16^1 & + 15 \times 16^0 & = 6751_{10}
 \end{array}$$

Šesnaest bit-ni binarni broj može se upotrebom heksadecimalnog brojnog sistema vrlo prikladno prikazati. Pretvaranje je dosta jednostavna i omogućava brzo saznanje o očitanim sadržajima u memoriji računara ili nekom njegovom drugom sklopu. To je i razlog o potrebi poznavanja prikazanih brojnih sistema.

Pretvaranje iz oktalnog u heksadekadni sistem i obratno je jednostavno, broj se pretvori u binarni i onda se grupira u grupe od *četiri* ili od *tri* znamenke i pretvara iz jednog u drugi oblik.

$$6154_8 = 110\ 001\ 101\ 100_2 = 1100\ 0110\ 1100_2 = C6C_{16} = 3180_{10}$$

Direktno pretvaranje dekadnog broja u oktalni moguće je po istom principu kao pretvaranje u binarni oblik. No, najjednostavnije je dekadni broj pretvoriti u binarni a onda binarni broj grupiranjem binarnih znamenki pretvoriti u oktalni ili heksadekadni, već prema potrebi.

Pretvaranje oktalnog zapisa u heksadecimalni zapis nekog broja ne može se izvršiti naposredno, već se vrši preko binarnog broja. Jedna tetrad binarnog broja predstavlja heksadecimalni broj. Isto tako, tri znamenke binarnog broja predstavljaju jednu znamenku u oktalnom brojnog sistemu. Prema tome,

oktalni broj se treba prevesti u binarni, a iz binarnog grupiranjem po četiri znamenke izračuna se heksadecimalna vrijednost.

Primjeri zbrajanja:

$$\begin{array}{r|l} (1F4C)_{16} & 3+C=3+12=15=F \\ + (2E83)_{16} & 8+4=12=C \\ \hline (4D83)_{16} & 14+15=29:16=1(\text{prenos}) \quad 13 = D \text{ (ostatak)} \\ & 1+2+1=4 \end{array}$$

Primjeri oduzimanja:

$$\begin{array}{r|l} (21A3)_{16} & 3+16=19-15=4 \text{ ((1) prenos)} \\ - (1FFF)_{16} & 10+16=26-(15+1) = 10 \text{ (A) ((1) prenos)} \\ \hline (1A4)_{16} & 1+16=17-(15+1)=1 \end{array}$$

Primjer množenja:

$$\begin{array}{r} \underline{(A9E4F)_{16} \times (8A7)_{16}} \\ (54F278)_{16} \\ (6A2F16)_{16} \\ \hline 4A5429)_{16} \\ (5BDFBD89)_{16} \end{array}$$

Primjena binarnog brojnog sistema u računarskoj tehnici opravdana je zbog dvije prednosti koje sistem omogućava:

1. Pouzdanost u radu,
2. Ekonomičnost.

Pouzdanost se lako i sigurno ostvaruje jer elektronski sklop treba da zauzme samo dva stanja: ima i nema napona, odnosno "1" ili "0", pojednostavljeno "radi" ili "ne radi". Ako se uz to svakoj binarnoj kombinaciji pridoda određeni broj bit-a na način da se za svaku osigura ukupan paran broj jedinica ili nula (provjera na parnost), ili se izračunava ukupni brojevi iznos kao zbir svih brojki unutar određenog "bloka" podataka - kontrolni zbir (check sum) koji se uz pripadni mu blok prenosi, te slično navedenom, pridonosi se ukupnoj pouzdanosti sistema i kontroli na pojavu greške. Ekonomičnost se očituje u potrebi za najmanjim brojem vodova za prenos signala na daljinu.

No zbog preglednosti i upravljanja radom sistema bolji je za korisnike od drugih brojnih sistemima.

Osnovni i najmanji element u kombinaciji impulsa je mogućnost da na jedno mjesto u kombinaciji impulsa dođe pozitivni ili negativni impuls, odnosno "0" ili "1". Izbor između "0" i "1" predstavlja najmanji mogući izbor i predstavlja mjeru količine informacija nazvanu BIT (BInary digiT=binarna znamenka). Prema ASCII kodu skup od 8b (osam bit-a) predstavlja jedan znak. Prema dogovoru ta je kombinacija nazvana **BAJT (BYTE)**.

Dakle:

$$8 \text{ b (bit-a)} = 1 \text{ B (Bajt)}$$

Veće jedinice za mjerenje količine informacije od navedenih su:

$$1 \text{ kB (kilo Bajt)} = 1\ 024 \text{ B}$$

$$1 \text{ MB (Mega Bajt)} = 1\ 024 \text{ kB} = 1\ 048\ 576 \text{ B}$$

$$1 \text{ GB (Giga Bajt)} = 1\ 024 \text{ MB} = 1\ 048\ 576 \text{ kB} = 1\ 073\ 741\ 824 \text{ B}$$

Ako kažemo da neki memorijski medij ima KAPACITET od 4 MB, pojednostavljeno rečeno to znači da je u njega moguće spremiti 4'194'304 B, odnosno znakova, u veličini od 8b (osam bit-a) svaki.

Multiplikator 1024 rezultat je matematičkog izraza:

$$2^{10} = 1024$$

a to je dekadni iznos binarnog broja: 100'0000'0000 B.

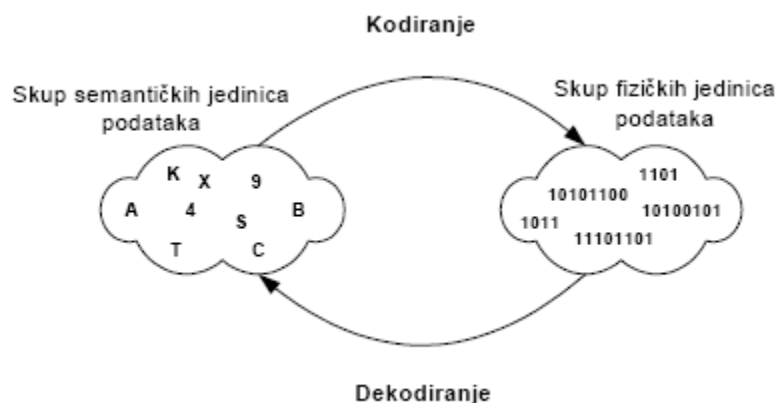
Kako računar (digitalno) koristi isključivo brojeve, to znači da se svi znakovi, instrukcije i podaci moraju pretvoriti u brojeve kako bi računaru bili razumljivi. Znakovi i instrukcije najčešće se unose preko tastature koja šalje računaru odgovarajuće kombinacije impulsa. Uopćeno, tastatura je elektromehanički pretvarač koji znak na tipci po pritisku pretvara u pripadnu mu binarnu kombinaciju. Svakom znaku pripada njemu svojstvena binarna kombinacija. Skup znakova i binarnih kombinacija naziva se KOD, a sam postupak kreiranja binarnih kombinacija naziva se KODIRANJE.

Iz skorije čovjekove historije postupak pridruživanja impulsa, električnih ili svijetlosnih ili sličnih, pojedinim znakovima pisma poznat je pod nazivom "MORZE-ova abeceda". Računar koristi ista načela u naprednijem obliku.

Iz navedenog je jasno da računari ne mogu uspješno razmjenjivati podatke ako ne koriste isti kod, te je od velikog značaja standardizacija koda i njegovo poštivanje, ali i njegovo poznavanje.

2.2. KODIRANJE I POJAM KODIRANJA

Znakovi kojima se ljudi služe u međusobnoj komunikaciji, pa i u komunikaciji s računarom, dakle oni koji se nalaze na tastaturi računara moraju poprimiti binarni oblik prije obrade u elektronskom računaru. Već je spomenuto da elektronski elementi od kojih je građen računar mogu poprimiti samo dva različita stanja od kojih se jedno označava s I , a drugo s 0 . To je veza s binarnim brojevnim sistemom, ali se zapisi, odnosno podatak u računaru ne tumače uvijek kao binarni brojevi, već kao binarni zapis. Kada čovjek zatreba uvid u sadržaj tih zapisa, on ga dobiva u njemu prilagođenom zapisu, skupom semantičkih znakova. Postupci kojima se to omogućava nazivaju se kodiranje odnosno dekodiranje. Na slici 2.1. prikazani su ovi procesi kao preslikavanje skupa semantičkih jedinica podataka u skup fizičkih jedinica podataka i obrnuto.



Slika: 2.1. Proces kodiranja i dekodiranja podataka

Proces kodiranja se provodi na način da se pojedinim semantičkim znakovima pridružuje odgovarajuća kombinacija binarnih nula i jedinica koje tvore fizički zapis semantičke vrijednosti. Skup pravila po kojim se obavlja taj proces, odnosno koja određuju koja se binarna kombinacija pridružuje semantičkoj vrijednosti, naziva se **kod**, a tako se naziva i rezultat kodiranja.

U procesu obrade podataka računara koriste se razni kodovi, ovisno o nosiocu podataka i o zahtjevima izvođenja operacija s kodiranim podacima. Na sreću, korisniku računara uopće nije bitno u kojem su kodu izraženi podaci zato što se procesi kodiranja i dekodiranja provode automatizirano na nivou hardvera. Pritiskom na tipku sa semantičkom oznakom na tastaturi inicira se mikroprogram koji generišu odgovarajuće fizičke znakove.

2.2.1. VRSTE KODOVA

2.2.1.1. Čisti binarni kod

Prvi elektronski računari obrađivali su podatke izražene u čistom binarnom kodu, što znači da su binarni zapisi tumačeni kao binarni brojevi. Nedostatak čistog binarnog koda je potreba za velikim brojem oznaka kojima bi se izrazili veći brojevi.

2.2.1.2. Tetradni kodovi

Za potrebe kodiranja brojčanih podataka koji su u prvom razdoblju korištenja računara bili prevladavajući, razvijeni su tetradni kodovi. U tetradnom kodu postoji $2^4=16$ mogućih tetrada, a budući je za prikazivanje dekadskih znamenaka potrebno 10 tetrada, 6 tetrada ostaje neiskorišteno. Te neiskorištene tetrade nazivaju se pseudotetrade.

BCD kod (Binary Coded Decimal) je najjednostavniji tetradni kod, pri čemu je svaka znamenka kodirana pomoću jedne tetrade, čiji binarni zapis tumačen kao broj odgovara vrijednosti dekadске znamenke.

Uz BCD kod poznatiji tetradni kodovi su Aiken, Exzess3 i Gray kod, koji se razlikuju po pravilima pridruživanja binarnih kombinacija pojedinim dekadskim znamenkama.

2.2.1.3. Osam-bitni kodovi

Upotreba šireg skupa semantičkih znakova, koji osim znamenki sadrži slovne znakove, znakove interpunkcije, matematičke i logičke operatore te druge specijalne znakove uvjetovala je pojavu 8-bitnih kodova. Njihova fizička zaliha znakova je tolika da može poslužiti za uspješno kodiranje praktično cijelog skupa semantičkih znakova, čak razlikujući velika i mala slova. Ta fizička zaliha znakova, odnosno ukupni broj mogućih binarnih varijacija, iznosi $2^8 = 256$. Ovi kodovi se temelje na kodiranju znakova pomoću jednog bajta (byte) pa vrijedi:

$$\mathbf{1 \text{ bajt} \longleftrightarrow 1 \text{ znak}}$$

Najstariji 8-bitni kodovi su prošireni binarno decimalni kod EBCDIC (Extended Binary Coded Decimal Interchange Code), te ASCII kod (American Standard Code for Information Interchange) koji je bio najčešće u upotrebi zbog svoje jednostavnosti. Danas postoji više različitih 8-bitnih kodova prema različitim standardima, a definišu se pomoću kodnih stranica. Kodna stranica sadrži parove semantičkih znakova i pripadajućih kodova iskazanih u heksadecimalnom obliku. Kodne stranice prilagođavaju osnovni kod specifičnim zahtjevima pojedinih govornih jezika, odnosno njihovih pisama.

U tabeli 2.1. prikazana je kodna stranica 1250 koja se kod nas koristi u Windows-ima jer ima definisane kodove za naše posebne znakove.

| HX | G | HX | G | HX | G | HX | G | HX | G | HX | G | HX | G |
|----|----|----|---|----|---|----|-----|----|---|----|---|----|---|
| 20 | | 40 | @ | 60 | ` | | | A0 | | C0 | Ř | E0 | ř |
| 21 | ! | 41 | A | 61 | a | | | A1 | ˘ | C1 | Á | E1 | á |
| 22 | " | 42 | B | 62 | b | 82 | , | A2 | ˘ | C2 | Â | E2 | â |
| 23 | # | 43 | C | 63 | c | | | A3 | Ł | C3 | Ă | E3 | ă |
| 24 | \$ | 44 | D | 64 | d | 84 | „ | A4 | ⌘ | C4 | Ä | E4 | ä |
| 25 | % | 45 | E | 65 | e | 85 | ... | A5 | Ą | C5 | Ĺ | E5 | ĺ |
| 26 | & | 46 | F | 66 | f | 86 | † | A6 | | C6 | Č | E6 | č |
| 27 | ' | 47 | G | 67 | g | 87 | ‡ | A7 | § | C7 | Ç | E7 | ç |
| 28 | (| 48 | H | 68 | h | | | A8 | ˘ | C8 | Ĉ | E8 | ĉ |
| 29 |) | 49 | I | 69 | i | 89 | ‰ | A9 | © | C9 | É | E9 | é |
| 2A | * | 4A | J | 6A | j | 8A | Š | AA | Ş | CA | Ê | EA | ê |
| 2B | + | 4B | K | 6B | k | 8B | ‹ | AB | « | CB | Ë | EB | ë |
| 2C | , | 4C | L | 6C | l | 8C | Ŝ | AC | – | CC | Ĕ | EC | ĕ |
| 2D | - | 4D | M | 6D | m | 8D | Ť | AD | - | CD | Ī | ED | î |
| 2E | . | 4E | N | 6E | n | 8E | Ž | AE | ® | CE | Ï | EE | ï |
| 2F | / | 4F | O | 6F | o | 8F | Ž | AF | Ž | CF | Ď | EF | ď |
| 30 | 0 | 50 | P | 70 | p | | | B0 | ° | D0 | Đ | F0 | đ |
| 31 | 1 | 51 | Q | 71 | q | 91 | ˘ | B1 | ± | D1 | Ñ | F1 | ñ |
| 32 | 2 | 52 | R | 72 | r | 92 | ˘ | B2 | ˘ | D2 | Ň | F2 | ň |
| 33 | 3 | 53 | S | 73 | s | 93 | “ | B3 | ı | D3 | Ó | F3 | ó |
| 34 | 4 | 54 | T | 74 | t | 94 | ” | B4 | ˘ | D4 | Ô | F4 | ô |
| 35 | 5 | 55 | U | 75 | u | 95 | • | B5 | µ | D5 | Õ | F5 | õ |
| 36 | 6 | 56 | V | 76 | v | 96 | – | B6 | ¶ | D6 | Ö | F6 | ö |
| 37 | 7 | 57 | W | 77 | w | 97 | — | B7 | · | D7 | × | F7 | ÷ |
| 38 | 8 | 58 | X | 78 | x | | | B8 | ˘ | D8 | Ř | F8 | ř |
| 39 | 9 | 59 | Y | 79 | y | 99 | ™ | B9 | ą | D9 | Ů | F9 | ů |
| 3A | : | 5A | Z | 7A | z | 9A | š | BA | ş | DA | Ű | FA | ű |
| 3B | ; | 5B | [| 7B | { | 9B | › | BB | » | DB | Û | FB | ü |
| 3C | < | 5C | \ | 7C | | 9C | ś | BC | Y | DC | Ü | FC | ü |
| 3D | = | 5D |] | 7D | } | 9D | ť | BD | ˘ | DD | Ý | FD | ý |
| 3E | > | 5E | ^ | 7E | ~ | 9E | ž | BE | Ł | DE | Ť | FE | ť |
| 3F | ? | 5F | | | | 9F | ž | BF | ž | DF | ß | FF | · |

Tabela: 2.1. Primjer kodne stranice 1250 koja se koristi u Windows-ima

Naša slova Č, Ć, Ž, Š, Đ, č, ć, ž, š, đ ili palatali ranije nisu imala svoj vlastiti kod nego su se koristili ASCII kodovi nekih posebnih znakova koji nisu bili

bitni za korisnike računara. Danas postoje kodne stranice u različitim standardima koje imaju navedene znakove.

Postojanje različitih standarda i kodnih stranica može uzrokovati manje probleme pri razmjeni podataka između sistema koji koriste različite kodne stranice, pri čemu podaci postaju nečitljivi. Dio tog problema se rješava softverski, automatiziranim procedurama prevodenja prilikom prikazivanja semantičkih vrijednosti u pojedinim aplikacijama računarskog sistema.

2.2.1.4.Redundancija

Pri procesu kodiranja semantičke jedinice podataka preslikavaju se u fizičke jedinice podataka. Postupak mora biti jednoznačan da bi se osigurao inverzni postupak dekodiranja bez gubitaka u polaznoj poruci. Za definisanje pojma redundancije potrebno je prethodno definisati pojam zalihe znakova.

Zaliha znakova je broj koji kazuje koliko je različitih kombinacija moguće dobiti korištenjem Z znakova nekog alfabeta na N pozicija. Ukoliko se radi o alfabetu koji sadrži semantičke znakove, govori se o semantičkoj zalihi znakova, odnosno u suprotnom o fizičkoj zalihi znakova. Zaliha znakova se računa izrazom $ZZ = Z^N$.

Ako je fizička zaliha znakova veća od semantičke govori se o redundanciji. To znači da je s upotrijebljenim fizičkim jedinicama podataka moguće kodirati više semantičkih jedinica nego je potrebno, odnosno da je upotrijebljeno više fizičkih jedinica podataka nego je to minimalno potrebno za jednoznačan postupak kodiranja.

Redundancija je proširivanje ili dodavanje znakova uz poruku, a da ta proširenja nisu nužna za razumijevanje poruke. Redundantne poruke imaju isto značenje kao i bez proširenja.

Redundancija se može kvantitativno odrediti kao razlika između fizičkih i semantičkih sadržaja odlučivanja koji su dualni logaritam odgovarajuće zalihe znakova. Jedinica mjere za redundanciju je **bit**.

Moguće je odrediti tri tipa redundancije ovisno o njenoj prirodi i uzrocima, a to su:

- **Tehnološka.**
- **Tehnička.**
- **Organizacijska.**

Tehnološka redundancija određena je odabranom tehnologijom i na nju se ne može utjecati. Primjer tehnološke redundancije je pri kodiranju dekadskih znamenki. Već je spomenuto da se za kodiranje skupa dekadskih znamenki koriste tetrade kao kombinacije 4 bita. Odgovarajuće zalihe znakova su:

$$ZZ_S = 10^1, \text{ odnosno } ZZ_F = 2^4$$

a odgovarajući sadržaji odlučivanja su:

$$S_F = 4 \text{ bita} , \text{ odnosno } S_S = 3,33 \text{ bita}$$

pa redundancija iznosi:

$$R = S_F - S_S = 4 - 3.33 = 0.67 \text{ bita}$$

što znači da u slučaju kodiranja dekadskih znamenki s 4 bita ostaje neiskorišteno 0.67 bita, odnosno da je dovoljno 3,33 bita. Budući je bit najmanja fizička jedinica podataka, ne može se dijeliti već se moraju upotrijebiti cijela 4 bita. Kod BCD koda redundancija se manifestira pseudotetradama.

Tehnička redundancija se namjerno proizvodi i služi za povećanje sigurnosti kodiranja podataka, te olakšanog otkrivanja i otklanjanja pogrešaka. Ovaj oblik redundancije je dodatno opterećenje u procesu obrade podataka što loše utiče na performanse i povećava troškove, ali nalazi svoje opravdanje u povećanoj sigurnosti.

Organizacijska redundancija se odnosi na pohranjivanje podataka. Baze podataka ne smiju imati redundanciju, odnosno jedan podatak smije u bazi podataka biti zapisan samo jedanput.

2.3. LOGIČKA ALGEBRA

2.3.1. OSNOVNI POJMOVI LOGIČKE ALGEBRE

2.3.1.1. Algebra propozicija

Algebra propozicija ili račun iskaza je dio matematike, odnosno matematičke ili simboličke logike koji se bavi izrazima koji sadrže nenumeričke varijable i konstante. Za razliku od aritmetike i algebre numeričkih varijabli, algebra propozicija se bavi varijablama i konstantama koje mogu poprimiti nenumeričke vrijednosti, a koje se nazivaju iskazi.

Iskazi su različite konstatacije o stanju univerzuma. Samo oni iskazi za koje se može ustanoviti njihova istinitost su domena algebre propozicija. Iskazi za koje se to ne može ustanoviti ili su neutralni, nisu argumenti računa iskaza. Rečenica "Jedan i nula su jedan." je iskaz, dok izraz "Učenje poslovne informatike." to nije. Vrijednost iskaza može biti samo *istina* ili *neistina*.

Osim jednostavnih iskaza, tvore se i složeni iskazi povezivanjem jednostavnih iskaza, čija ocjena istinitosti ovisi o istinitosti pojedinih jednostavnih iskaza, te o načinu na koji su logički povezani u cjelinu. Ti načini logičkog povezivanja se nazivaju logičke operacije, a tako dobiveni

složeni iskazi se nazivaju logičke funkcije. Osnovne logičke operacije pomoću kojih se tvore logičke funkcije su:

- **Konjunkcija** koja se u prirodnom jeziku iskazuje veznikom “i”, a koja u teoriji skupova ima analogiju s presjekom skupova.
- **Disjunkcija** koja se u prirodnom jeziku iskazuje veznikom “ili”, a koja u teoriji skupova ima analogiju s unijom skupova. Specifičnost veznika ili u bosanskom jeziku nije naglašena, ali se može govoriti o inkluzivnom i ekskluzivnom obliku. Inkluzivni oblik, odnosno “uključivo ili” je primarni oblik koji znači “bilo koji”. Ekskluzivni oblik, odnosno “isključivo ili” znači “ili jedan ili drugi”.
- **Negacija** se u prirodnom jeziku iskazuje riječju “ne”. Ona se primjenjuje na jedan iskaz i mijenja mu istinitost.
- **Implikacija** se u prirodnom jeziku obično iskazuje izrazom “ako ... onda”, što istinitost jednog iskaza uvjetuje istinitošću nekog drugog iskaza.
- **Ekvivalencija** je dvosmjerna implikacija, a u prirodnom jeziku se predstavlja izrazom “onda i samo onda” ili na slične načine.

Broj iskaza i logičkih operacija upotrijebljenih u logičkoj funkciji nije ograničen, a njena vrijednost može biti samo *istina* i *neistina*.

2.3.2. BOOLE-ova ALGEBRA

Engleski matematičar George Boole je u 19. vijeku uspio logičke operacije prikazati na algebarski način, te stvoriti matematičku strukturu koja je po njemu i nazvana, ali se često koristi i naziv logička algebra. Ta matematička struktura se sastoji od skupa B čiji elementi imaju domenu vrijednosti “0” i “1”, zatim linearnih operacija koje su po analogiji s aritmetikom nazvane *množenje* i *sabiranje*, te operacije *komplementa* nad kojima vrijedi određeni skup zakona. Ti zakoni su uglavnom analogija zakona algebre propozicija.

2.3.2.1. Logičke varijable

Logička algebra barata s logičkim varijablama koje sadrže izvornu logičku vrijednost ili odražavaju logičku vrijednosti nekog iskaza, te mogu poprimiti samo dvije spomenute vrijednosti. Konvencijom je utvrđeno da se vrijednost “0” odnosi na *neistinu*, a vrijednost “1” na *istinu*, te je i na taj način logička algebra povezana s algebrom propozicija.

2.3.2.2. Logički operatori

Linearne operacije logičke algebre se predstavljaju logičkim operatorima. Logički operatori se primjenjuju na logičke varijable ili izraze. Način njihova označavanja nije strogo propisan pa se koristi više leksičkih načina

označavanja, odnosno notacija ili simbola, od kojih su najčešći usporedno prikazani u tabeli 2.2. Postoje i druge notacije ali se ovdje neće navoditi niti posebno obrazlagati.

| Logička operacija | Simbol operatora I | Simbol operatora II | Simbol operatora III |
|-------------------|--------------------|---------------------|----------------------|
| Množenje | * | I | AND |
| Sabiranje | + | ILI | OR |
| Negacija | | NE | NOT |

Tabela: 2.2. Logički operatori

Izbor notacije ovisi o tehnologiji računanja logičkih izraza. U većini programskih jezika se koristi treća notacija iz prethodne tablice jer se radi o engleskim riječima, budući je engleski jezik uzor za većinu programskih jezika. U edukativne svrhe najčešće se koristi prva notacija zbog svoje jednostavnosti i analogije s aritmetičkim operacijama u binarnom brojevnom sistemu, pa će ona biti korištena u nastavku ovog poglavlja.

2.3.2.3. Poredbeni (relacijski) operatori

Vrlo često se logički operatori ne odnose neposredno na logičke varijable, već na iskaze zasnovane na usporedbi različitih vrijednosti. Unutar tih izraza se uspoređuju neke stvarne vrijednosti koje nisu iz domene logičke algebre, ali rezultat njihove usporedbe jest. Pretpostavka je da te vrijednosti pripadaju istom tipu podataka. Pri tome se uspoređuju njihovi fizički, odnosno binarni zapisi, koji se onda posmatraju kao binarni brojevi, pa je moguće uspoređivati i vrijednosti, odnosno veličinu dva slova i slično. Osnovni poredbeni ili relacijski operatori prikazani su u tabeli 2.3. s primjerom iskaza i opisom značenja. Naziv relacijski operatori se izbjegava da se ne unosi zabuna u vezi s pojmom relacijskih baza podataka.

| Operator | Iskaz | Opis |
|----------|--------|--------------------------------|
| = | A = B | Da li je A jednako B |
| <> | A <> B | Da li je A različito od B |
| < | A < B | Da li je A manje od B |
| > | A > B | Da li je A veće od B |
| <= | A <= B | Da li je A manje ili jednako B |
| >= | A >= B | Da li je A veće ili jednako B |

Tabela: 2.3. Poredbeni ili relacijski operatori

Uloga poredbenih operatora je da ustanove istinitost određenog iskaza, te da mu pridruže odgovarajuću vrijednost iz domene logičke algebre, koja onda sudjeluje u osnovnim logičkim operacijama.

2.3.2.4. Logičke funkcije

Logičke funkcije su izrazi koji se tvore od logičkih varijabli i logičkih operatora. Vrijednost logičke funkcije ovisi o vrijednosti logičkih varijabli i upotrijebljenih operatora, a računa se prema pravilima logičke algebre koja su određena tablicama istinitosti i pravilima redoslijeda izvođenja operacija.

Logičke funkcije u kojima se koristi samo jedan logički operator su osnovne i nazivaju se prema upotrijebljenom logičkom operatoru, pa se govori o **I**, **II** i **NE** logičkim funkcijama. Ove funkcije se definišu pomoću tablica istinitosti na način da se definiše vrijednost funkcije za svaku kombinaciju vrijednosti logičkih varijabli. Logički operatori **II** i **I** koriste dva argumenta, dok je operator **NE** unaran, odnosno primjenjuje se samo na jedan argument. Kako se radi o logičkim varijablama koje mogu imati samo dvije vrijednosti, ukupan broj kombinacija za dvije varijable je četiri za binarne funkcije, odnosno dva za unarne. U tabeli 2.4. su prikazane, odnosno sadržane tablice istinitosti za osnovne logičke funkcije.

| X | Y | X * Y | X + Y | X' |
|---|---|-------|-------|----|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | |

Tabela: 2.4. Tablice istinitosti za osnovne logičke funkcije

Kombinacijom više osnovnih logičkih operatora i logičkih varijabli stvaraju se složene logičke funkcije, kojima je moguće opisati praktično sve logičke probleme. Dopuštena je upotreba zagrada po pravilima aritmetike. Složena logička funkcija ilustrirana je slijedećim primjerom:

$$Q = X * (Y' + X) + Y * (Z' + X') + Y * Z'$$

Logičke funkcije se obično označavaju velikim slovima **P**, **Q**, **R** i **S**. Znak jednakosti, u navedenom primjeru, ima matematičko značenje, odnosno nije poredbeni ili relacijski operator. Pri računanju vrijednosti složenih logičkih funkcija treba poštivati redoslijed izvršavanja pojedinih operatora i korištenja zagrada, odnosno treba poštivati slijedeće prioritete:

1. **Zgrade.**
2. **NE operacije.**

3. I operacije.
4. III operacije.

2.3.3. ZAKONI I TEOREMI LOGIČKE ALGEBRE

Pri izračunavanju i transformaciji složenih logičkih funkcija mogu se primjenjivati zakoni i teoremi. Zakoni logičke algebre su prikazani u tabeli 2.5.

| | | |
|--------------------------|-----------------------|-----------------------|
| Komutacije | $X+Y=Y+X$ | $X*Y=Y*X$ |
| Asocijacije | $(X+Y)+Z=X+(Y+Z)$ | $(X*Y)*Z=X*(Y*Z)$ |
| Distribucije | $X+(Y*Z)=(X+Y)*(X+Z)$ | $X*(Y+Z)=(X*Y)+(X*Z)$ |
| Komplementarnosti | $X+X'=1$ | $X*X'=0$ |

Tabela: 2.5. Zakoni logičke algebre

Teoremi logičke algebre izvode se iz osnovnih zakona logičke algebre i mogu se dokazati. Ovi teoremi prikazani su u tabeli 2.6.

| | | |
|-------------------------------|----------------|----------------|
| Idempotentnosti | $X+X=X$ | $X*X=X$ |
| Ograđenosti | $X+1=1$ | $X*0=0$ |
| Apsorpcije | $X+(X*Y)=X$ | $X*(X+Y)=X$ |
| Komplement | $0'=1$ | $1'=0$ |
| Komplement komplementa | $(x')'=X$ | |
| De Morganov teorem | $(X+Y)'=X'*Y'$ | $(X*Y)'=X'+Y'$ |

Tabela: 2.6. Teoremi logičke algebre

Za ilustraciju će se predstaviti dokazivanje De Morganova teorema i teorema apsorpcije. De Morganov teorem iskazan je ekvivalentima $(X+Y)'=X'*Y'$, odnosno $(X*Y)'=X'+Y'$. Dokaz teorema proveden je pomoću tablica istinitosti i prikazan u tabeli 2.7.

| X | Y | X' | Y' | X+Y | (X+Y)' | X'*Y' | X*Y | (X*Y)' | X'+Y' |
|---|---|----|----|-----|--------|-------|-----|--------|-------|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Tabela: 2.7. Dokaz De Morganovog teorema

Teorem apsorpcije iskazan je ekvivalentima $X+(X*Y)=X$, odnosno $X*(X*Y)=X$. Dokaz teorema proveden je pomoću tablica istinitosti i prikazan u tabeli 2.8.

| X | Y | X*Y | X+(X*Y) | X+Y | X*(X+Y) |
|---|---|-----|---------|-----|---------|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Tabela: 2.8. Dokaz teorema apsorpcije

Zakoni i teoremi logičke algebre se koriste za pojednostavljivanje složenih logičkih funkcija, što je posebno bitno kod projektovanja logičkih sklopova pri izgradnji digitalnih računara.

2.4. PRIMJENA LOGIČKE ALGEBRE U INFORMATICI

2.4.1. IZRAČUNAVANJE LOGIČKIH FUNKCIJA

Pri izračunavanju vrijednosti logičkih funkcija moguća su dva pristupa koja ovise o tome da li se računa vrijednost logičke funkcije za sve moguće slučajeve logičkih varijabli ili se računa vrijednost logičke funkcije za određeni skup konkretnih vrijednosti logičkih varijabli.

Prvi pristup računanja svih mogućih vrijednosti logičke funkcije obično koristi tehniku tablica istinitosti. U tabeli 2.9. prikazan je primjer računanja složene logičke funkcije koja glasi:

$$Q = X * Y' + X * Y * Z' + X' * Y * Z'$$

| X | Y | Z | X' | Y' | Z' | X*Y' | X*Y*Z' | X'*Y*Z' | Q |
|---|---|---|----|----|----|------|--------|---------|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabela: 2.9. Izračunavanje vrijednosti logičke funkcije tablicom istinitosti

Ukoliko se izračunava vrijednost složene logičke funkcije za poznati skup vrijednosti logičkih varijabli obično se primjenjuje postupak sličan onom u aritmetici. Za primjer neka posluži složena logička funkcija:

$$Q = X * Z + Y * (X + Z')$$

a zadane vrijednosti logičkih varijabli su:

$$X = 0$$

$$Y = 1$$

$$Z = 1$$

Uvrštavanjem zadanih vrijednosti funkcija poprima izgled

$$Q = 0 * 1 + 1 * (0 + 1')$$

Prvi korak je izvršavanje operacije negacije kao unarne i najvišeg prioriteta, pa funkcija poprima oblik

$$Q = 0 * 1 + 1 * (0 + 0),$$

te izvršavanjem operacija unutar zagrada funkcija poprima oblik

$$Q = 0 * 1 + 1 * 0,$$

a izvršavanjem operacija množenja kao narednih po pravilima prioriteta funkcija poprima oblik

$$Q = 0 + 0,$$

i konačno izvršavanjem operacije zbrajanja rješenje funkcije Q za zadani skup vrijednosti logičkih varijabli je

$$Q = 0.$$

Ukoliko se izračunava vrijednost složene logičke funkcije sastavljene od poredbenih iskaza i za poznati skup vrijednosti općih varijabli u tim iskazima, primjenjuje se sličan postupak kome prethodi korak utvrđivanja istinitosti navedenih iskaza. Za primjer neka posluži funkcija

$$Q = A > B + (C < A + B < C)',$$

a zadane vrijednosti numeričkih varijabli su:

$$A = 3,$$

$$B = 5,$$

$$C = 7.$$

Utvrdivanjem istinitosti pojedinih poredbenih iskaza i uvrštavanjem dobivenih umjesto samih iskaza, funkcija poprima izgled

$$Q = 0 + (0 + 1)',$$

nakon čeka se izračunava vrijednost u zagradi, te primjenjuje operacija negacije, što nakon zbrajanja daje konačnu vrijednost logičke funkcije. Opisani koraci računanja su prikazani kako sljede:

$$Q = 0 + 1',$$

$$Q = 0 + 0,$$

$$Q = 0.$$

Poredbeni iskaz $A > B$ za zadane vrijednosti $A=3$ i $B=5$ nije istinit i umjesto njega je u funkciju uvrštena logička konstanta 0 . Sličan postupak je proveden i za ostale iskaze. Za neki drugi skup zadanih vrijednost varijabli A , B i C , vrijednost logičke funkcije Q se može promijeniti, ali uvijek će moći poprimiti samo logičke vrijednosti 0 i 1 .

2.4.2. ELEKTRONSKI SKLOPOVI

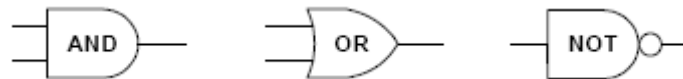
Razvoj digitalnih elektronskih računara usko je povezan s razvojem tehnologije izrade integrisanih kola, koja omogućavaju nebrojene mogućnosti kreacije spojeva osnovnih logičkih sklopova u složene elektroničke sklopove od kojih su izgrađeni računari.

Na ulaze osnovnih logičkih sklopova, koji predstavljaju elektroničku implementaciju logičkih funkcija, dovode se impulsi nad kojima se želi izvršiti određena logička operacija. Rezultat operacije se očituje na izlazu logičkog sklopa odmah po promjeni ulaznog stanja ili po dopuštenju kojim upravlja generator takta unutar elektroničkog sklopa, odnosno računara. Usklađeno djelovanje mnoštva osnovnih logičkih sklopova omogućava efikasno izvršavanje i najsloženijih logičkih zadataka.

Digitalni elektronski računar pomoću logičkih sklopova obavljaju i aritmetičke i logičke operacije. Svi podaci unutar računara su u binarnom zapisu što omogućava da se svi podaci obrađuju kombinacijama logičkih operacija, a rezultat se interpretira ovisno o vrsti podataka. Dakle, sve operacije u računaru se odvijaju prema pravilima logičke algebre. Kompleksne logičke funkcije mogu se svesti na tri osnovne logičke funkcije **I**, **ILI** i **NE**, odnosno kada je riječ o računarima obično se označavaju sa **AND**, **OR** i **NOT**.

2.4.2.1. Logički sklopovi

Logički sklopovi su elektronička implementacija logičkih funkcija. Osnovni logički sklopovi su **AND**, **OR** i **NOT** i pomoću njih je moguće realizirati bilo koju složenu logičku funkciju. Na slici 2.2. prikazani su uobičajeni grafički simboli za navedene logičke sklopove.



Slika 2.2. Grafički simboli osnovnih logičkih sklopova

Osim osnovnih logičkih sklopova u digitalnoj elektronici se za potrebe efikasnijeg definisanja složenih elektronskih sklopova uvode i pomoćni logički sklopovi koji su kombinacija osnovnih. To su prije svega sklopovi **NAND**, **NOR** i **XOR**. I ovi se sklopovi temelje na odgovarajućim složenim logičkim funkcijama odnosno mogu se iskazati ekvivalentnim funkcijama kako slijedi:

$$X \text{ NAND } Y = \text{NOT } (X \text{ AND } Y).$$

$$X \text{ NOR } Y = \text{NOT } (X \text{ OR } Y).$$

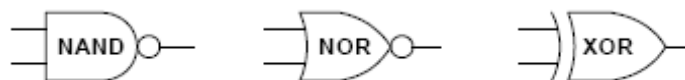
$$X \text{ XOR } Y = (X \text{ OR } Y) \text{ AND NOT}(X \text{ AND } Y).$$

Zbog veće efikasnosti izrade složenih logičkih sklopova s manjim brojem elemenata, ove se funkcije definišu s vlastitim tablicama istinitosti kako je prikazano u tabeli 2.10.

| X | Y | X NAND Y | X NOR Y | X XOR Y |
|---|---|----------|---------|---------|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

Tabela: 2.10. Tablica istinitosti za logičke funkcije NAND, NOR i XOR

Na slici 2.3. prikazani su uobičajeni grafički simboli za ove izvedene logičke sklopove.



Slika 2.3. Grafički simboli izvedenih logičkih sklopova

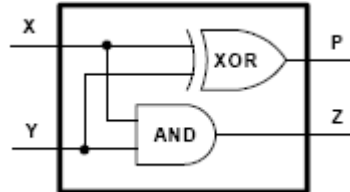
Primjena logičkih sklopova za izvođenje matematičkih operacija ilustrirana je sklopom koji zbraja dva jednoznamenkasta binarna broja i naziva se nepotpuni ili polu sabirač. Njegova tablica istinitosti dana je u tabeli 2.11.

| X | Y | P | Z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
|---|---|---|---|

Tabela: 2.11. Tablica istinitosti nepotpunog sabirača

Shema spajanja logičkih sklopova u nepotpunom sabiraču prikazana je na slici 2.4.. Potpuni sabirač se dobije spajanjem određenog broja nepotpunih.



Slika: 2.4. Shema nepotpunog sabirača

2.4.3. SOFTVERSKJE PRIMJENE

2.4.3.1. Programiranje

Logička operacija implikacije ili uvjetovanosti koja se u prirodnom jeziku izražava jezičnom konstrukcijom “*ako ... onda ...*”, ima istu ili sličnu interpretaciju u programskim jezicima. Kako se u pravilu radi o engleskom jeziku kao osnovi programskih jezika, logička operacija uvjetovanosti se iskazuje programskom konstrukcijom “*IF ... THEN ...*”, koja se ujedno i realizira u programu vrlo sličnom instrukcijom. Uobičajena sintaksa **IF** instrukcije je

IF uslov THEN akcija,

gdje je **uslov** logička funkcija, a **akcija** je skup instrukcija koje treba izvršiti samo ako je uslov istinit. Ako uslov nije ispunjen akcija se neće izvesti, a program će nastaviti s narednom instrukcijom iza **IF** instrukcije. Provjera istinitosti uslova se provodi izračunavanjem vrijednosti logičke funkcije sa trenutnim vrijednostima programskih varijabli, a prema pravilima logičke algebre. Promjenom vrijednosti varijabli u programu, pri narednom računanju istinitosti uslova, može doći do promjene vrijednosti logičke funkcije te promjene toka odvijanja programa. Primjena **IF** instrukcije u različitim oblicima opisuje uslovno grananje u računarskom programu i jedna je od okosnica upravljanja programskim strukturama.

2.4.3.2. Pretraživanje baza podataka

Logičke funkcije se koriste i za postavljanje uslova kod pretraživanja baza podataka, gdje će sistem prikazati samo one podatke koji zadovoljavaju postavljeni uslov. **SQL** je programski jezik kojim se između ostalog mogu

postavljati upiti nad relacijskom bazom podataka. Primjer **SQL** naloga kojim se traže svi podaci o studentima prve godine fakulteta bi izgledao kao:

```
SELECT * FROM STUDENT WHERE GODINA_STUDIJA = 1
```

2.4.3.3. Pretraživanje Web-a

Logičke funkcije se koriste i za efikasnije pretraživanje Web-a s internetskim pretraživačima. Uobičajeni način pretraživanja je da se u polje za pretraživanje upišu riječi koje mora sadržavati pojedina web stranica. Ako je potrebno smanjiti broj pronađenih stranica na mjeru koja je prihvatljiva za pojedinačno pregledavanje, potrebno je postaviti složeniji uslov u području za napredno pretraživanje. Na primjer ako netko želi pronaći podatke o svim hotelima u Konjicu i Mostaru može postaviti slijedeću uslov pretraživanja

(Konjic OR Mostar) AND hotel,

a pretraživač će popisati samo stranice u kojima se obvezno spominje riječ *hotel* i jedna od riječi *Konjic* ili *Mostar*. Treba obratiti pažnju da je veznik “i”, upotrijebljen u prirodnom jeziku, zapravo iskazan s logičkom funkcijom **OR**.