UNIVERZITET SINGIDUNUM

Violeta Tomašević

OSNOVI RAČUNARSKE TEHNIKE

BEOGRAD, 2012.

UNIVERZITET SINGIDUNUM

Prof. dr Violeta Tomašević

OSNOVI RAČUNARSKE TEHNIKE

Četvrto izdanje

Beograd, 2012.

OSNOVI RAČUNARSKE TEHNIKE

Autor:
Prof. dr. Violeta Tomašević

Recenzenti:
Prof. dr Milan Milosavljević
Prof. dr Jovan Đorđević

Izdavač:
UNIVERZITET SINGIDUNUM
Beograd, Danijelova 32

Za izdavača:
Prof. dr Milovan Stanišić

Tehnička obrada:
Violeta Tomašević

Dizajn korica:
Aleksandar Mihajlović

Godina izdanja:
2012.

Tiraž:
300 primeraka

Štampa:
Mladost Grup
Loznica

ISBN: 978-86-7912-406-7

P r e d g o v o r

Ova knjiga je nastala kao rezultat potrebe za odgovarajućim pisanim materijalom iz predmeta Osnovi računarske tehnike koji autor drži na prvoj godini Fakulteta za informatiku i računarstvo Univerziteta Singidunum u Beogradu. Pri pisanju je učinjen napor da knjiga bude prihvatljiva za čitaoce bez nekog većeg predznanja u oblasti računarstva, ali uz posedovanje osnovnog znanja iz matematike. Namenjena je i prilagođena prosečnom studentu, jer je osnovni cilj autora bio da svi studenti koji slušaju predmet Osnovi računarske tehnike mogu na razumljiv i lak način da savladaju predviđeno gradivo. Upravo iz tog razloga, knjiga ne sadrži značajnija teorijska razmatranja, niti prikazuje punu širinu i složenost razmatranih problema, već je prvenstveno orijentisana ka praktičnim aspektima računarske tehnike koji su ilustrovani brojnim primerima.

Knjiga je podeljena u sedam poglavlja: *Matematičke osnove računarske tehnike*, *Logički elementi*, *Memorijski elementi*, *Logičke funkcije*, *Logičke mreže*, *Osnovi organizacije računara* i *Personalni računar*.

U prvom poglavlju izložen je matematički aparat na kome se zasniva rad svakog računarskog sistema. S obzirom da se podaci u računaru predstavljaju i obrađuju u binarnom obliku, najveća pažnja posvećena je binarnom brojnom sistemu. Osim njega, razmatran je i heksadecimalni brojni sistem. U poglavlju su dati postupci konverzije brojeva između binarnog, decimalnog i heksadecimalnog sistema, kao i osnovne aritmetičke operacije nad binarnim brojevima. Takođe su opisani načini predstavljanja različitih vrsta podataka u računaru, uključujući cele brojeve, realne brojeve i podatake znakovnog tipa.

Osnovne logičke operacije, koje se intenzivno koriste na najnižem nivou obrade podataka u računaru, tema su drugog poglavlja. One su praćene odgovarajućim logičkim elementima koji služe za njihovu implementaciju u prekidačkim mrežama.

U trećem poglavlju obrađene su različite vrste flip-flopova kao osnovnih memorijskih elemenata koji se koriste u izgradnji sekvencijalnih prekidačkih mreža. Ukazano je na njihovu povezanost sa logičkim elementima. Posebna pažnja je posvećena taktovanim flip-flopovima i njihovom učešću u analizi i sintezi prekidačkih mreža.

Četvrto poglavlje uvodi pojam logičke funkcije kojom se mogu opisati složene strukture koje obezbeđuju potrebnu funkcionalnost računarskom sistemu. Prikazana su tri načina za predstavljanje logičke funkcije: pomoću kombinacionih tablica, u algebarskom obliku i pomoću Karnoove karte. Opisan je postupak

realizacije logičkih funkcija korišćenjem prekidačkih mreža. Na kraju je izložen metod minimizacije logičkih funkcija primenom Karnoove karte u cilju smanjenja složenosti njihove realizacije.

U petom poglavlju opisane su razne prekidačke mreže koje predstavljaju standardne module kombinacionog (koderi, dekoderi, multiplekseri, ...) i sekvencijalnog (registri, brojači, ...) tipa u realizaciji računarskog sistema. Za svaki modul dati su njegova funkcionalnost, osnovne osobine, mogućnosti primene i unutrašnja realizacija.

Opšta organizacija računarskog sistema i osnovni principi njegovog funkcionisanja opisani su u šestom poglavlju. Osim osnovnih pojmova, u poglavlju su izloženi i pojedini koncepti koji doprinose većoj efikasnosti rada računara: *pipeline*, DMA prenos i mehanizam prekida. U drugom delu poglavlja, organizacija računara je detaljno objašnjena na praktičnom primeru računarskog okruženja u kome centralno mesto zauzima procesor Intel 8086.

Poslednje poglavlje posvećeno je predstavljanju najbitnijih komponenata personalnog računara: matične ploče, procesora, memorija i ulazno/izlaznih uređaja. Naglašena je centralna uloga matične ploče i opisani su njeni delovi. Posebna pažnja posvećena je memorijama i to operativnoj memoriji, raznim vrstama spoljašnjih memorija i keš memoriji. Od ulazno/izlaznih uređaja predstavljeni su monitori i štampači.

Na kraju svakog poglavlja, u okviru *Vežbanja*, data su pitanja i zadaci za samostalno rešavanje koji studentima treba da posluže kao provera znanja na temu koja je razmatrana u poglavlju. Pitanja i zadaci su odabrani tako da u potpunosti pokrivaju predviđeno gradivo, pa se mogu iskoristiti za pripremanje ispita.

Biću zahvalna svima onima koji mi ukažu na greške ili daju korisne savete za buduće ispravke i dopune ovog materijala.

Beograd, decembar 2010.god. Autorka

S A D R Ž A J

UNIVERZITET
PRIVATNI
AKREDITOVANI
UNIVERZITET
SINGIDUNUM

1 Matematičke osnove računarske tehnike

Osnovna namena računara i drugih digitalnih sistema i uređaja je obrada informacija predstavljenih u binarnom obliku. Da bi se razumeo način njihovog rada, neophodno je najpre upoznati se sa osnovnim matematičkim aparatom na kome se taj rad zasniva. Tu se, pre svega, misli na binarni brojni sistem koji pripada klasi pozicionih brojnih sistema. Zatim, od velikog značaja su i osnovne aritmetičke operacije nad binarnim brojevima koje se u radu računara intenzivno koriste. Uvođenje heksadecimalnog brojnog sistema predstavlja sponu koja korisniku olakšava prihvatanje binarne predstave podataka. Radi potpunog razumevanja procesa rada, bitno je upoznati se sa postupcima konverzija podataka između binarnog, decimalnog i heksadecimalnog sistema. S obzirom da rešavanje svakog problema podrazumeva obradu različitih tipova podataka, potrebno je upoznati se sa načinom predstavljanja osnovnih tipova podataka (celih brojeva, realnih brojeva i podataka znakovnog tipa) u računaru.

1.1. Pozicioni brojni sistemi

Pozicioni brojni sistemi su sistemi zapisivanja brojeva u kojima vrednost broja zavisi od:

- cifara upotrebljenih za zapisivanje broja
- pozicije svake cifre u broju

Najčešće korišćeni pozicioni brojni sistem je decimalni (dekadni) brojni sistem. Osnova ovog sistema je 10, pa se mesne vrednosti na susednim pozicijama u broju razlikuju 10 puta. Tako, cifra na mestu jedinica (pozicija 0) doprinosi

vrednosti broja sa 1, cifra na mestu desetica (pozicija 1) sa 10, cifra na mestu stotina (pozicija 2) sa 100 itd. Decimalni brojevi $68_{(10)}$ i $291_{(10)}$ imaju različite vrednosti jer su zapisani različitim ciframa. Osim upotrebljenih cifara, na vrednost decimalnog broja utiču i pozicije na kojima se cifre nalaze. Tako, iako su zapisani istim ciframa, decimalni brojevi $276_{(10)}$ i $762_{(10)}$ imaju različite vrednosti jer su cifre u njima zapisane u drugačijem redosledu.

U opštem slučaju, bilo koji pozitivan ceo broj X u pozicionom brojnom sistemu može se zapisati u sledećem obliku:

$$X = a_n q^n + a_{n-1} q^{n-1} + \ldots + a_2 q^2 + a_1 q^1 + a_0 q^0 \qquad (1)$$

gde su:

n – broj cifara u zapisu broja X umanjen za 1, jer se najniža pozicija u broju smatra pozicijom 0

q – prirodan broj koji predstavlja osnovu brojnog sistema

a_i, $0 \leq i \leq n$ – cifre u zapisu broja X koje moraju pripadati dozvoljenom skupu cifara S_q za dati brojni sistem

U skladu sa navedenom formulom, decimalni broj $3827_{(10)}$ može se zapisati u obliku:

$$3827 = 3 \cdot 10^3 + 8 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0$$

U ovom primeru broj cifara u zapisu je 4, pa je $n = 3$, osnova brojnog sistema je $q = 10$, a cifre u zapisu broja, $a_3 = 3$, $a_2 = 8$, $a_1 = 2$ i $a_0 = 7$, pripadaju skupu cifara koji odgovara decimalnom brojnom sistemu $S_{10} = \{0,1,2,3,4,5,6,7,8,9\}$.

Pozicioni brojni sistemi mogu imati proizvoljnu osnovu, ali u praktičnim primenama najzastupljeniji su sistemi prikazani u tabeli 1.1.

Brojni sistem	Osnova brojnog sistema (q)	Skup dozvoljenih cifara (S_q)
binarni	2	$S_2 = \{0,1\}$
oktalni	8	$S_8 = \{0,1,2,3,4,5,6,7\}$
decimalni	10	$S_{10} = \{0,1,2,3,4,5,6,7,8,9\}$
heksadecimalni	16	$S_{16} = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Tabela 1.1 Najčešće korišćeni pozicioni brojni sistemi

1.1.1 Binarni brojni sistem

Binarni brojni sistem je sistem u kome se za predstavljanje brojeva koriste samo dve cifre: 0 i 1. Ovakav način predstavljanja informacija je vrlo pogodan za primenu u računarskim i drugim digitalnim sistemima. Naime, u ovim sistemima postoji česta potreba za opisivanjem stanja kada „ima signala“ ili „nema signala“, neki uređaj je „uključen“ ili „isključen“, podatak je „raspoloživ“ ili „nije raspoloživ“ i slično, što se efikasno može predstaviti binarnim vrednostima 0 i 1.

Pošto je osnova binarnog brojnog sistema 2, formula kojom je predstavljen pozitivan ceo broj X u ovom sistemu ima oblik:

$$X = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_2 2^2 + a_1 2^1 + a_0 2^0 \qquad (2)$$

Iz ove jednačine sledi da se mesne vrednosti na susednim pozicijama u binarnom broju razlikuju 2 puta. To znači da cifra na poziciji 0 (krajnja desno cifra u broju) doprinosi vrednosti binarnog broja sa $2^0 = 1$, cifra na poziciji 1 sa $2^1 = 2$, cifra na poziciji 2 sa $2^2 = 4$, cifra na poziciji 3 sa $2^3 = 8$ itd.

Binarno-decimalne konverzije brojeva

Iako binarni brojni sistem najviše odgovara mogućnostima savremene elektronske tehnologije, za ljude je mnogo prihvatljiviji decimalni brojni sistem sa kojim dolaze u dodir u ranom periodu svog života i dalje ga aktivno primenjuju i usvajaju. Stoga, da bi čovek mogao da razume način funkcionisanja računara, neophodno je da poznaje načine konvertovanja informacija iz binarnog u decimalni oblik i obrnuto.

Konverzija binarnog broja u decimalni obavlja se primenom jednačine (2). Decimalna vrednost koja odgovara zadatom binarnom broju dobija se kao suma mesnih vrednosti na kojima binarni broj ima vrednost 1.

Neka je dat binarni broj $10010110_{(2)}$. Primenom jednačine (2), decimalna vrednost ovog broja računa se na sledeći način:

$$X = 1\cdot 2^7 + 0\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0$$

$$X = 1\cdot 2^7 + 1\cdot 2^4 + 1\cdot 2^2 + 1\cdot 2^1 = 1\cdot 128 + 1\cdot 16 + 1\cdot 4 + 1\cdot 2 = 150_{(10)}$$

Zadati binarni broj ima vrednost 1 na pozicijama 1, 2, 4 i 7. Mesne vrednosti na tim pozicijama su 2^1, 2^2, 2^4 i 2^7, pa se decimalna vrednost broja dobija njihovim sabiranjem.

Konverzija decimalnog broja u binarni odvija se u dva koraka:

1. zadati decimalni broj podeliti sa 2 sa ostatkom; ostatak zapisati, a rezultat deljenja ponovo podeliti sa 2 sa ostatkom; dobijeni ostatak opet zapisati, a rezultat deljenja ponovo podeliti sa 2 sa ostatkom; ovaj postupak ponavljati sve dok se kao rezultat deljenja ne dobije vrednost 0
2. binarni broj formirati od zapisanih ostataka u obrnutom redosledu od onoga u kome su nastajali

Neka je dat decimalni broj $169_{(10)}$. Primenom koraka 1. dobija se sledeće:

	rezultat deljenja	ostatak	
169 : 2 =	84	(1)	(*LSB*)
84 : 2 =	42	(0)	
42 : 2 =	21	(0)	
21 : 2 =	10	(1)	
10 : 2 =	5	(0)	
5 : 2 =	2	(1)	
2 : 2 =	1	(0)	
1 : 2 =	0	(1)	(*MSB*)

U skladu sa korakom 2, od dobijenih ostataka može se formirati binarni broj koji odgovara zadatom decimalnom broju $169_{(10)}$. To se radi tako što poslednji ostatak predstavlja bit najveće težine u binarnom broju (*MSB – most significant bit*), a prvi ostatak bit najmanje težine (*LSB – least significant bit*). Dakle, decimalni broj $169_{(10)}$ se u binarnom obliku predstavlja vrednošću $10101001_{(2)}$.

Kao što se vidi, binarni zapis nekog broja zahteva upotrebu mnogo više cifara od njegovog decimalnog zapisa. Na primer, broj $169_{(10)}$ se u dekadnom sistemu zapisuje pomoću tri cifre, dok je za njegovu binarnu predstavu potrebno 8 cifara. Predugačak zapis predstavlja osnovni nedostatak binarnog brojnog sistema, jer nije pogodan za čoveka.

Aritmetičke operacije nad binarnim brojevima

Obrada binarnih podataka u računaru zahteva intenzivnu primenu osnovnih aritmetičkih operacija: sabiranja, oduzimanja, množenja i deljenja. Pri obavljanju ovih operacija, binarni brojevi koji u njima učestvuju tretiraju se kao celina, što znači da se uzima u obzir međusobni uticaj susednih pozicija u broju. Aritmetičke operacije nad binarnim brojevima izvode se po istim pravilima kao i aritmetičke operacije nad decimalnim brojevima, s tim što se mora uzeti u obzir da se mesne

vrednosti susednih pozicija razlikuju 2 puta (a ne 10 kao u decimalnom brojnom sistemu).

Sabiranje je binarna operacija jer se obavlja nad dva binarna broja. Slično sabiranju decimalnih brojeva, binarno sabiranje se vrši tako što se sabiraju vrednosti na istim pozicijama u binarnim brojevima koji učestvuju u operaciji. Ukoliko zbir na nekoj poziciji premaši vrednost 1 (bude 2 ili više), javlja se prenos za narednu poziciju. Prenos koji se javi na nekoj poziciji mora se uzeti u obzir pri sabiranju na toj poziciji.

Pošto se u binarnim brojevima na jednoj poziciji mogu naći samo 0 ili 1, moguće su sledeće situacije (prvi sabirak je cifra prvog broja, a drugi cifra drugog broja na datoj poziciji):

$0_{(2)} + 0_{(2)} = 0_{(2)}$ $\quad$ $0_{(2)} + 1_{(2)} = 1_{(2)}$

$1_{(2)} + 0_{(2)} = 1_{(2)}$ $\quad$ $1_{(2)} + 1_{(2)} = 10_{(2)}$

Takođe, ukoliko na nekoj poziciji postoji prenos sa niže pozicije, može nastati još jedna situacija (prvi i drugi sabirak imaju značenje kao i ranije, a treći sabirak predstavlja prenos):

$1_{(2)} + 1_{(2)} + 1_{(2)} = 11_{(2)}$

Kao što se vidi, u dve situacije javlja se prenos za narednu poziciju (premašena je vrednost 1), tj. kada su rezultati sabiranja $10_{(2)} = 2_{(10)}$ i $11_{(2)} = 3_{(10)}$,. U oba slučaja, stvarni rezultat sabiranja na tekućoj poziciji predstavlja cifru najmanje težine u broju (u prvom slučaju to je cifra 0, a u drugom 1), dok se 1 prenosi na narednu poziciju.

Neka su a i b cifre na istoj poziciji u binarnim brojevima koje treba sabrati. Označimo sa c_{ul} prenos sa prethodne pozicije, a sa c_{iz} prenos za narednu poziciju. Rezultat sabiranja na posmatranoj poziciji je s. Uz uvedene oznake, postupak sabiranja binarnih vrednosti a i b može se predstaviti tabelom 1.2.

Tabela 1.2 ilustruje prethodno opisane situacije. Na primer, ukoliko treba sabrati vrednosti $a = 1$ i $b = 1$, a prenos sa niže pozicije ne postoji $c_{ul} = 0$ (vrsta 4), rezultat sabiranja je $10_{(2)}$, što znači da je rezultat na posmatranoj poziciji $s = 0$, a prenos za narednu poziciju $c_{iz} = 1$. Slično, ako su $a = 1$ i $b = 0$, i postoji prenos sa niže pozicije $c_{ul} = 1$ (vrsta 7), rezultat sabiranja je $10_{(2)}$, što znači da je rezultat na tekućoj poziciji $s = 0$, a prenos za narednu poziciju $c_{iz} = 1$. Na sličan način se mogu analizirati i sve ostale vrste u tabeli.

c_{ul}	a	b	c_{iz}	s
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

Tabela 1.2 Sabiranje binarnih brojeva

U nastavku će operacija sabiranja biti prikazana na jednom primeru. Neka su dati binarni brojevi $A = 10110111_{(2)}$ i $B = 10011010_{(2)}$ koje treba sabrati. Postupak sabiranja se može predstaviti na sledeći način:

c_{ul}		1	0	1	1	1	1	1	0
A			1	0	1	0	1	1	1	1
B	+		1	0	0	1	1	0	1	0
$A+B$		1	0	1	0	0	1	0	0	1

Brojevi A i B sabiraju se tako što se najpre saberu cifre na poziciji 0. Kao rezultat dobija se 1+0 = 1 i nema prenosa za narednu poziciju. Zatim se sabiraju cifre na poziciji 1, tj. 1+1 = 10. Rezultat na poziciji 1 je 0, a prenos za narednu poziciju je 1. Na poziciji 2 sabiraju se cifre 1 i 0, ali se mora uključiti i prenos 1 koji dolazi sa niže pozicije. Tako se dobija 1+0+1 = 10, pa je na poziciji 2 rezultat 0 i postoji prenos za narednu poziciju. Cifre na poziciji 3 sabiraju se prenosom sa niže pozicije, tj. 1+1+1 = 11, pa je rezultat na ovoj poziciji 1 i postoji prenos za narednu poziciju. Ovaj postupak se nastavlja dok se ne dođe do najviših pozicija u brojevima koji se sabiraju.

Oduzimanje binarnih brojeva predstavlja binarnu operaciju koja se vrši tako što se oduzimaju vrednosti na istim pozicijama. Kao i kod oduzimanja decimalnih brojeva, i u ovom slučaju, ukoliko je vrednost od koje se oduzima veća od vrednosti koja se oduzima, neophodno je uzeti pozajmicu sa više pozicije. Ta pozajmica, kada pređe na nižu poziciju, ima vrednost 2 (u decimalnom sistemu je vredela 10), jer je u binarnom sistemu odnos mesnih vrednosti između susednih pozicija 2.

Pošto se binarni brojevi zapisuju samo nulama i jedinicama, pri oduzimanju su moguće sledeće situacije (umanjenik je cifra prvog broja, a umanjilac cifra drugog broja na datoj poziciji):

$0_{(2)} - 0_{(2)} = 0_{(2)}$ $1_{(2)} - 0_{(2)} = 1_{(2)}$ $1_{(2)} - 1_{(2)} = 0_{(2)}$

Ostalo je još da se razmotri situacija $0_{(2)} - 1_{(2)}$. Pošto je umanjenik veći od umanjioca, mora se uzeti pozajmica sa naredne više pozicije. To znači da se na narednoj poziciji vrednost umanjuje za 1, a na tekućoj poziciji se dodaje vrednost $10_{(2)} = 2_{(10)}$. Tako rezultat oduzimanja na tekućoj poziciji postaje 1 (2–1 = 1). Zatim se prelazi na oduzimanje na narednoj poziciji, pri čemu se mora voditi računa da je sa nje uzeta pozajmica.

Treba zapaziti da se na istoj poziciji mogu pojaviti dve pozajmice: jedna je pozajmica koja je od tekuće pozicije tražena sa niže pozicije, a druga je pozajmica koja je sa tekuće pozicije tražena od naredne, više pozicije.

Neka su *a* i *b* cifre na istoj poziciji u binarnim brojevima koje treba oduzeti. Označimo sa p_{ul} pozajmicu koja je data prethodnoj poziciji, a sa p_{iz} pozajmicu od naredne pozicije. Rezultat oduzimanja na posmatranoj poziciji je *r*. Sada se postupak oduzimanja binarnih vrednosti *a* i *b* može predstaviti tabelom 1.3.

p_{ul}	*a*	*b*	p_{iz}	*r*
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

Tabela 1.3 Oduzimanje binarnih brojeva

Razmotrimo drugu vrstu u tabeli. Dakle, treba naći razliku *a*–*b* = 0–1, pri čemu nema pozajmice sa niže pozicije (p_{ul} = 0). Da bi se to uradilo, mora se sa više pozicije uzeti pozajmica, pa je zato p_{iz} = 1. Ova pozajmica vredi 2 na nižoj poziciji, pa kada se od nje oduzme 1 dobija se rezultat *r* = 1. Nešto komlikovanija situacija data je u osmoj vrsti. U ovom slučaju treba oduzeti vrednosti *a*–*b* = 1–1, ali imajući u vidu da je sa ove pozicije već data pozajmica nižoj poziciji jer je p_{ul} = 1. Da bi se sprovelo oduzimanje, najpre treba uzeti pozajmicu sa više pozicije. Ova pozajmica na tekućoj poziciji ima vrednost 2, od čega treba jednu jedinicu odvojiti za pozajmicu p_{ul}, dok druga ostaje na razmatranoj poziciji. Tako se na tekućoj poziciji dobija 1+1 = 2, pa kad se od toga oduzme *b* = 1, dobija se rezultat *r* = 1. Na sličan način mogu se analizirati i sve ostale vrste u tabeli.

Operacija oduzimanja biće ilustrovana na sledećem primeru. Neka su dati binarni brojevi $A = 10110111_{(2)}$ i $B = 10011010_{(2)}$ koje treba oduzeti. Postupak oduzimanja može se predstaviti na sledeći način:

					$2_{(10)}$					
				0	0	$2_{(10)}$				
A		1	0	1	1	0	1	1	1	
B	-	1	0	0	1	1	0	1	0	
$A–B$		0	0	0	1	1	1	0	1	

Brojevi A i B oduzimaju se tako što se najpre oduzmu cifre na poziciji 0. Kao rezultat, dobija se 1–0 = 1. Na sličan način se oduzimaju i vrednosti na pozicijama 1 i 2. Na poziciji 3 potrebno je naći razliku 0–1. Da bi se to uradilo, sa više pozicije se uzima pozajmica, tako da na njoj vrednost postaje 0. Pozajmica dolazi na poziciju 3 kao decimalna vrednost 2 i nakon oduzimanja dobija se rezultat 1. Zatim se prelazi na oduzimanje na poziciji 4. Opet se pojavljuje isti slučaj da treba oduzeti 0–1. Stoga se sa naredne pozicije uzima pozajmica, pa na narednoj poziciji ostaje 0, a pozajmica na poziciji 4 postaje decimalna vrednost 2. Nakon oduzimanja, rezultat na poziciji 4 je 1. Postupak oduzimanja se nastavlja na pozicijama 5, 6 i 7 i dobija se konačni rezultat.

Množenje binarnih brojeva obavlja se po istim pravilima kao i množenje višecifrenih decimalnih brojeva, s tim što se prilikom sabiranja vodi računa da se radi u binarnom brojnom sistemu. Postupak množenja se može opisati sledećim koracima:

- svakom cifrom drugog činioca pomnožiti prvi činilac
- dobijene parcijalne proizvode napisati jedan ispod drugog, pomerene za jedno mesto ulevo
- sabrati sve parcijalne proizvode kao binarne brojeve

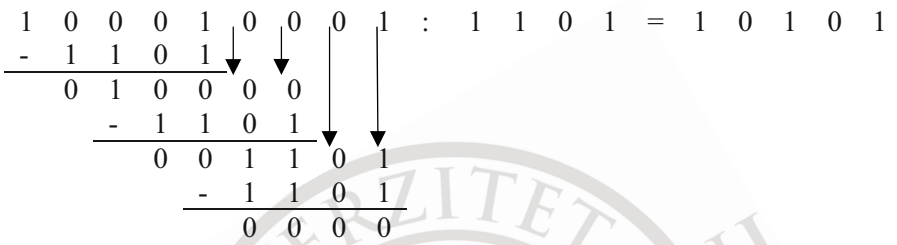
Dati postupak se može ispratiti na sledećem primeru:

1	1	0	0	·	1	1	0	1	=					1	1	0	0
												0	0	0	0	
											1	1	0	0		
									+		1	1	0	0		
										1	0	0	1	1	1	0	0

Deljenje binarnih brojeva vrši na sličan način kao deljenje decimalnih brojeva, s tim što se prilikom oduzimanja uzima u obzir da se radi u brojnom sistemu sa osnovom 2. Deljenje dva binarna broja obavlja se na sledeći način:

- grupu cifara deljenika (počevši sa leve strane) podeliti deliocem
- dobijeni rezultat pomnožiti deliocem, potpisati ispod grupe cifara u deljeniku i primeniti binarno oduzimanje
- spustiti sledeću cifru deljenika, a zatim ponavljati opisani postupak sve dok se ne dobije potpisani binarni broj koji je manji od delioca

Sledi primer deljenja dva binarna broja po opisanom postupku:



1.1.2 Heksadecimalni brojni sistem

Da bi se prevazišao problem sa dužinom zapisa binarnog broja, uveden je heksadecimalni sistem. Zapis broja u ovom sistemu zahteva manje cifara nego u dekadnom sistemu i znatno manje cifara nego u binarnoj predstavi, što je mnogo prihvatljivije za čoveka. Iako računar operiše nad binarnim brojevima, rezultati u binarnom obliku mogu se vrlo jednostavno prevesti u heksadecimalni oblik zahvaljajući pogodnom odnosu osnova ova dva sistema ($2^4 = 16$).

Heksadecimalni brojni sistem je sistem u kome se za predstavljanje brojeva koristi 16 heksadecimalnih cifara: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F. Pošto se u svakom brojnom sistemu za označavanje jedne cifre mora koristiti samo jedan simbol, to su u heksadecimalnom sistemu za predstavljanje dvocifrenih brojeva usvojene oznake početnih slova abecede. U tabeli 1.4 date su decimalne, heksadecimalne i binarne vrednosti brojeva od 0 do 15.

S obzirom da je osnova heksadecimalnog brojnog sistema 16, jednačina kojom se predstavlja pozitivan ceo broj X u ovom sistemu ima oblik:

$$X = a_n 16^n + a_{n-1} 16^{n-1} + ... + a_2 16^2 + a_1 16^1 + a_0 16^0 \qquad (3)$$

Iz jednačine (3) sledi da se mesne vrednosti na susednim pozicijama u heksadecimalnom broju razlikuju 16 puta. To znači da cifra na poziciji 0 doprinosi vrednosti heksadecimalnog broja sa $16^0 = 1$, cifra na poziciji 1 sa $16^1 = 16$, cifra na poziciji 2 sa $16^2 = 256$ itd.

Decimalna vrednost	Heksadecimalna vrednost	Binarna vrednost
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tabela 1.4 Heksadecimalne cifre i njihove decimalne i binarne vrednosti

Heksadecimalno-decimalne konverzije brojeva

Postupci konvertovanja heksadecimalnog broja u decimalni i obrnuto vrlo su slični ranije opisanim postupcima konvertovanja binarnog broja u decimalni i obrnuto. Jedina razlika je u osnovi brojnog sistema koja je sada 16.

Konverzija heksadecimalnog broja u decimalni obavlja se primenom jednačine (3).

Neka je dat heksadecimalni broj $5E3_{(16)}$. Primenom jednačine (3), decimalna vrednost ovog broja računa se na sledeći način:

$$X = 5 \cdot 16^2 + 14 \cdot 16^1 + 3 \cdot 16^0 = 5 \cdot 256 + 14 \cdot 16 + 3 \cdot 1 = 1507_{(10)}$$

U poslednjem izrazu, pošto se radi sa decimalnim vrednostima, heksadecimalna cifra E predstavljena je svojom decimalnom vrednošću 14.

Konverzija decimalnog broja u heksadecimalni odvija se u sledećim koracima:

1. zadati decimalni broj podeliti sa 16 sa ostatkom; ostatak zapisati, a rezultat deljenja ponovo podeliti sa 16 sa ostatkom; ovaj postupak ponavljati sve dok se kao rezultat deljenja ne dobije vrednost 0
2. heksadecimalni broj formirati od zapisanih ostataka u obrnutom redosledu od onoga u kome su nastajali; pre formiranja broja sve ostatke treba konvertovati u heksadecimalni oblik

Neka je dat decimalni broj $4328_{(10)}$. U skladu sa korakom 1 dobija se sledeće:

	rezultat deljenja	ostatak
4328 : 16 =	270	(8)
270 : 16 =	16	(14 = E)
16 : 16 =	1	(0)
1 : 16 =	0	(1)

Od dobijenih ostataka formira se heksadecimalni broj tako što se poslednji ostatak uzima kao cifra na najvišoj poziciji, a zatim se redom ispisuju heksadecimalne cifre sve dok se ne dođe do prvog ostatka koji predstavlja cifru na najnižoj poziciji u heksadecimalnom broju. Dakle, decimalni broj $4328_{(10)}$ se u heksadecimalnom obliku predstavlja sa $10E8_{(16)}$.

Konverzije brojeva između binarnog i heksadecimalnog sistema su znatno jednostavnije. Razlog za to leži u činjenici da osnova heksadecimalnog sistema odgovara četvrtom stepenu osnove binarnog sistema.

Heksadecimalno-binarne konverzije brojeva

Zahvaljujući pogodnom odnosu osnova binarnog i heksadecimalnog brojnog sistema, svaka heksadecimalna cifra može se predstaviti pomoću četiri binarne cifre. To znatno olakšava prevođenje brojeva iz jednog sistema u drugi, pa su postupci konverzije vrlo jednostavni.

Konverzija binarnog broja u heksadecimalni vrši se po sledećem postupku:

1. napraviti grupe od po 4 cifre u binarnom broju počevši sa desne strane, tj. od najniže pozicije
2. svaku grupu predstaviti jednom heksadecimalnom cifrom prema tabeli 1.4
3. heksadecimalni broj formirati od dobijenih heksadecimalnih cifara u redosledu u kome su odgovarajuće grupe raspoređene u binarnom broju

Neka je dat binarni broj $110111110_{(2)}$. On se deli u tri grupe od po 4 cifre na sledeći način:

1 | 1011 | 1110

Grupa 1110 odgovara heksadecimalnoj cifri E, grupa 1011 cifri B, a 1 cifri 1. Stoga se zadati binarni broj u heksadecimalnom obliku predstavlja kao $1BE_{(16)}$.

Konverzija heksadecimalnog broja u binarni odvija se u sledećim koracima:

1. svaku cifru heksadecimalnog broja predstaviti grupom od 4 binarne cifre prema tabeli 1.4
2. binarni broj formirati spajanjem grupa u redosledu u kome su i nastale

Neka je dat heksadecimalni broj $3A9_{(16)}$. Cifre u ovom broju se u binarnom obliku zapisuju sa:

$3_{(16)} = 3_{(10)} = 0011_{(2)}$

$A_{(16)} = 10_{(10)} = 1010_{(2)}$

$9_{(16)} = 9_{(10)} = 1001_{(2)}$

Binarni broj se formira jednostavnim spajanjem grupa, pa je:

$3A9_{(16)} = 0011\ 1010\ 1001_{(2)} = 1110101001_{(2)}$

1.2. Predstavljanje podataka u računaru

Širok spektar problema koji se mogu rešiti primenom računara zahteva korišćenje raznovrsnih tipova podataka za opisivanje raznih objekata, njihovih aktivnosti i međusobnih veza. Preglednosti radi, podaci su po tipu klasifikovani u dve osnovne grupe: statički tipovi i dinamički tipovi. Pod statičkim tipovima podrazumevaju se tipovi podataka kod kojih je unapred definisana unutrašnja struktura svakog podatka. Za razliku od njih, kod dinamičkih tipova struktura podataka se slobodno menja tokom rada.

Statički tipovi podataka obuhvataju skalarne i strukturirane podatke. Pod skalarnim tipovima podrazumevaju se najprostiji tipovi podataka čije su vrednosti skalari, tj. veličine koje predstavljaju elementarne celine i za koje nema potrebe da se dalju razlažu na komponente. Podatak je strukturiran ako se sastoji od više komponenata koje se nalaze u precizno definisanom odnosu (na primer matrice).

U ovom poglavlju detaljno će biti razmotrena tri skalarna tipa podataka koja se najčešće koriste: celobrojni (*integer*), realni (*real*) i znakovni tip podataka (*character*).

1.2.1 Predstavljanje označenih celih brojeva

U dosadašnjem izlaganju razmatrano je samo predstavljanje pozitivnih celih brojeva u binarnom obliku. Međutim, za rešavanje mnogih problema zahteva se korišćenje označenih, tj. pozitivnih i negativnih, celih brojeva.

Za označavanje brojeva u dekadnom brojnom sistemu koriste se oznake „+“ (za predstavljanje pozitivnih brojeva, ili se ovaj znak izostavlja) i „–“ (za predstavljanje negativnih brojeva). Ove oznake se pišu ispred cifara koje definišu apsolutnu vrednost decimalnog broja. U binarnom brojnom sistemu, ovakav način predstavljanja označenih brojeva nije moguć zato što je u njemu dozvoljeno korišćenje samo dva simbola, 0 i 1 (računar, takođe, može da prepozna samo ova dva znaka). Problem označavanja u binarnom brojnom sistemu najjednostavnije se može rešiti tako što se ispred apsolutne vrednosti broja doda jedna cifra koja predstavlja znak. Može se usvojiti, na primer, da 0 označava da je broj pozitivan, a 1 da je negativan. Ovakav način zapisivanja binarnih brojeva naziva se predstavljanje binarnih brojeva pomoću **znaka i apsolutne vrednosti**.

Neoznačeni broj $7_{(10)} = 111_{(2)}$ može se označiti pomoću znaka i apsolutne vrednosti na sledeći način:

$+ 7_{(10)} = 0111_{(2)}$ označen pozitivan binarni broj

$- 7_{(10)} = 1111_{(2)}$ označen negativan binarni broj

Iako je navedeni način označavanja vrlo jednostavan u pogledu formiranja zapisa broja, on nema značajniju primenu. Glavni razlog za to je što se nad binarnim brojevima zapisanim pomoću znaka i apsolutne vrednosti teško obavljaju aritmetičke operacije. Naime, ovako označeni brojevi se ne mogu posmatrati na jedinstven način, već se uvek mora posebno voditi računa o cifri na mestu najveće težine koja predstavlja znak. Mnogo efikasniji način označavanja binarnih brojeva je pomoću komplementa dvojke.

Komplement dvojke

Komplement dvojke ili puni komplement je postupak binarnog predstavljanja označenih celih brojeva koji se najčešće sreće u praksi. Po ovom postupku, označeni brojevi se binarno predstavljaju pomoću *n*-bitske reči čiji najstariji razred *S* označava predznak broja. Kao i u slučaju zapisa pomoću znaka i apsolutne

vrednosti, zapis u komplementu dvojke, takođe, počinje cifrom $S = 0$ ukoliko je broj nenegativan (pozitivan ili 0), a cifrom $S = 1$ ako je broj negativan. Broj 0 se zapisuje nulama u svim bitovima.

Neka je Z zadati ceo broj, a K binarni zapis broja Z. Pozitivan ceo broj $+|Z|$ i negativan ceo broj $-|Z|$ mogu se predstaviti u binarnom obliku na sledeći način:

$+|Z|$: $\quad K^{+} = (|Z|)_{(2)}$

$-|Z|$: $\quad K^{-} = (2^{n} - |Z|)_{(2)} = (2^{n} - K^{+})_{(2)}$

gde je $|Z|$ apsolutna vrednost broja Z. Veličina K^{-} se naziva *punim komplementom* veličine K^{+}, zato što važi $K^{+} + K^{-} = 2^{n}$. Takođe, može se reći i obrnuto, tj. da je veličina K^{+} puni komplement veličine K^{-}.

Na primer, neka je $Z = 15_{(10)}$. Pomoću 16-bitne reči ($n = 16$), brojevi +15 i –15 binarno se predstavljaju kao:

+15:	$K^{+} = (\|15\|)_{(2)} =$	0000 0000 0000 1111
–15:	$K^{-} = (2^{16} - \|15\|)_{(2)} =$	1 0000 0000 0000 0000
		– 0000 0000 0000 1111
		1111 1111 1111 0001

Iz primera se vidi da se efekat oduzimanja broja od 2^{n} može postići i invertovanjem tog broja (jedinice se zamene nulama, a nule jedinicama) uz dodavanje vrednosti 1. Tako važi:

polazni broj	0000 0000 0000 1111
invertovani broj	1111 1111 1111 0000
sabiranje sa 1	1111 1111 1111 0000 + 1 = 1111 1111 1111 0001

Na osnovu navedenog, može se zaključiti sledeće:

- pozitivan broj u komplementu dvojke dobija se dodavanjem cifre 0 ispred binarnog zapisa koji odgovara apsolutnoj vrednosti tog broja
- negativan broj u komplementu dvojke dobija se tako što se
  - ispred binarnog zapisa koji odgovara apsolutnoj vrednosti broja doda 0
  - zatim se sve binarne cifre invertuju
  - dobijeni broj se sabere sa 1

Slede primeri nalaženja komplementa dvojke jednog pozitivnog $(+7)_{(10)}$ i jednog negativnog $(-10)_{(10)}$ celog broja:

+7:	0000 0000 0000 0111	(dodavanje nule, puni komplement)
–10:	0000 0000 0000 1010	(dodavanje nule)
	1111 1111 1111 0101	(invertovanje)
	+1	(sabiranje sa jedinicom)
	1111 1111 1111 0110	(puni komplement)

Komplement dvojke ima osobinu da ukoliko se dva puta uzastopno primeni nad nekim brojem, kao rezultat se dobija polazni broj. Ovo je pokazano na primeru broja $13_{(10)}$:

polazni broj (+13):	0000 0000 0000 1101
invertovanje:	1111 1111 1111 0010
sabiranje sa 1:	+1
puni komplement (–13):	1111 1111 1111 0011
polazni broj (–13):	1111 1111 1111 0011
invertovanje:	0000 0000 0000 1100
sabiranje sa 1:	+1
puni komplement (+13):	0000 0000 0000 1101

Vrednost pozitivnog broja u komplementu dvojke se ne menja ako se ispred cifre najveće težine doda proizvoljan broj nula (vodeće nule). Na primer, važi $0111_{(2)} = 00000000111_{(2)}$. Takođe, vrednost negativnog broja u komplementu dvojke ostaje ista ako se ispred broja doda proizvoljan broj jedinica (vodeće jedinice). Na primer, $1001_{(2)} = 1111111001_{(2)}$.

Iako je postupak nalaženja komplementa dvojke vrlo jednostavan, može se dalje pojednostaviti. To se postiže tako što se:

- polazni binarni broj podeli na dva dela, levi i desni; desni deo čine prva jedinica sa desne strane u broju i sve nule koje slede iza nje (broj nula može biti i 0); preostale cifre čine levi deo broja
- komplement dvojke se dobija tako što se izvrši invertovanje levog dela broja, a desni deo broja ostaje nepromenjen

Kada se ovaj postupak primeni za nalaženje komplementa dvojke binarnog broja $01010010010000_{(2)}$ dobija se:

polazni broj	=	010100100	10000
		levi deo	desni deo
komplement dvojke	=	10101101110000	

Ovim pojednostavljenim postupkom izbegnuto je bilo kakvo računanje komplementa dvojke, već se za svaki broj može direktno napisati njegov puni komplement.

Pretpostavimo da je poznat n-bitni binarni broj $a_{n-1}.....a_1a_0$ koji je zapis nekog označenog celog broja u komplementu dvojke. Decimalna vrednost ovog broja, X, može se izračunati pomoću formule:

$$X = -a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + ... + a_1 2^1 + a_0 2^0 \qquad (4)$$

Primenom formule (4) na brojeve zapisane u komplementu dvojke iz ranijih primera, $0111_{(2)}$ i $10110_{(2)}$, dobijaju se njihove decimalne vrednosti $X1$ i $X2$, respektivno:

$$X1 = -0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 4 + 2 + 1 = +7_{(10)}$$

$$X2 = -1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = -16 + 4 + 2 = -10_{(10)}$$

Opseg neoznačenih brojeva

Pre određivanja opsega brojeva koji se mogu predstaviti u komplementu dvojke, biće analiziran jednostavniji slučaj opsega neoznačenih brojeva.

Pretpostavimo da zapis neoznačenog binarnog broja sadrži n binarnih cifara. Broj različitih kombinacija koje se mogu generisati sa ovim brojem cifara je 2^n. Pošto neoznačeni brojevi obuhvataju brojeve 0,1,2,3, ..., to se jedna od kombinacija mora upotrebiti za predstavljanje broja 0. Preostalih 2^n–1 kombinacija služe za predstavljanje brojeva od 1 do 2^n–1. Odavde sledi da se opseg neoznačenih binarnih brojeva zapisanih sa n cifara može dobiti po sledećoj formuli:

$$0 \le x \le 2^n - 1 \qquad \text{odnosno} \qquad x \in \{0, 1, \ldots, 2^n - 1\}$$

gde je x neoznačeni binarni broj.

Primenom ove formule dobijaju se opsezi neoznačenih brojeva koji se predstavljaju pomoću četiri, odnosno osam binarnih cifara na sledeći način:

$n = 4$: $0 \le x \le 2^4\text{-}1$ $0 \le x \le 15$

$n = 8$: $0 \le x \le 2^8\text{-}1$ $0 \le x \le 255$

Pomoću četiri binarne cifre mogu se zapisati neoznačeni brojevi iz opsega $x \in \{0, 1,\ldots, 15\}$, a pomoću osam cifara brojevi iz opsega $x \in \{0, 1,\ldots, 255\}$.

Opseg brojeva predstavljenih u komplementu dvojke

Za razliku od neoznačenih, označeni brojevi obuhvataju pozitivne, negativne brojeve i nulu. Pretpostavimo da zapis označenog binarnog broja predstavljenog u komplementu dvojke sadrži n binarnih cifara. Kao što je ranije rečeno, broj različitih kombinacija koje se mogu generisati sa ovim brojem cifara je 2^n. Pošto se jedna od kombinacija mora upotrebiti za predstavljanje broja 0, preostalih 2^n–1 kombinacija služe za predstavljanje pozitivnih i negativnih brojeva. S obzirom da se pozitivni i negativni brojevi ravnopravno koriste, može se uzeti da polovina kombinacija služi za predstavljanje pozitivnih, a druga polovina za predstavljanje negativnih brojeva. Međutim, broj 2^n–1 je neparan, pa se broj pozitivnih i broj negativnih brojeva koji se mogu predstaviti u komplementu dvojke moraju razlikovati za 1. Obično se uzima da je

broj pozitivnih brojeva = $(2^n–1–1)/2 = 2^n/2–2/2 = 2^{n-1}–1$

broj negativnih brojeva = $1+(2^n–1–1)/2 = 2^{n-1}$

Na osnovu navedenog, opseg označenih binarnih brojeva zapisanih u komplementu dvojke sa n cifara računa se po sledećim formulama:

$x \le 2^{n-1} – 1$ za $x \ge 0$

$|x| \le 2^{n-1}$ za $x < 0$ odnosno $x \in \{–2^{n-1}, ..., –1, 0, 1,\ldots, 2^{n-1}–1\}$

gde je x označeni binarni broj u komplementu dvojke.

Ako se ove formule primene za $n = 4$ i $n = 8$, mogu se naći opsezi četvorocifrenih i osmocifrenih binarnih brojeva predstavljenih u komplementu dvojke:

$n = 4$: $– 2^3 \le x \le 2^3\text{-}1$ $–8 \le x \le 7$

$n = 8$: $–2^7 \le x \le 2^7\text{-}1$ $–128 \le x \le 127$

Dakle, pomoću četiri binarne cifre mogu se zapisati označeni brojevi u komplementu dvojke koji pripadaju skupu $x \in \{-8, ..., -1, 0, 1,\ldots, 7\}$, a pomoću osam cifara brojevi iz skupa $x \in \{-128, ..., -1, 0, 1,\ldots, 127\}$.

Aritmetičke operacije nad brojevima u komplementu dvojke

Nad celim binarnim brojevima zapisanim u komplementu dvojke mogu se efikasno obavljati osnovne aritmetičke operacije sabiranja i oduzimanja. Prilikom izvršavanja ovih operacija, brojevi se posmatraju kao jedinstvene celine, tj znak se smatra sastavnim delom broja i nad njim se obavljaju operacije na isti način kao i nad svim ostalim ciframa broja.

Oduzimanje označenih binarnih brojeva A i B predstavljenih u komplementu dvojke, tj. $A - B$, može se realizovati pomoću operacije sabiranja. To se postiže tako što se umanjenik A uzima kao prvi sabirak, a umanjilac B sa negativnim predznakom kao drugi sabirak u zbiru:

$$A - B = A + (-B)$$

U nastavku će biti razmotrena samo operacija aritmetičkog sabiranja brojeva u punom komplementu, zato što se operacija oduzimanja, primenom prethodne formule, lako može svesti na operaciju sabiranja.

U računaru se operacija sabiranja (oduzimanja) kontroliše pomoću dva indikatora: indikatora prenosa C (*carry*) i indikatora prekoračenja V (*overflow*).

Indikator prenosa C predstavlja prenos iz najstarijeg razreda. Ukoliko je $C = 1$, to ukazuje da je rezultat operacije veći od raspoloživog prostora, što ne mora da znači da je dobijeni rezultat pogrešan. Može se pokazati da je i u ovom slučaju rezultat operacije ispravan ukoliko su prenosi iz dva najstarija razreda, C i P, isti (bilo dve nule, ili dve jedinice). Glavnu ulogu u utvrđivanju ispravnosti rezultata ima indikator prekoračenja V koji se računa na sledeći način:

ako je $C = P$, onda je $V = 0$

ako je $C \neq P$, onda je $V = 1$

Rezultat operacije je ispravan ako je $V = 0$, a neispravan ako je $V = 1$.

Sabiranje dva broja predstavljena u komplementu dvojke obavlja se na sledeći način:

a) oba sabirka se predstave u komplementu dvojke
b) dobijeni puni komplementi se saberu po pravilima binarnog sabiranja ne vodeći računa o znaku; prilikom sabiranja pamte se prenosi između razreda

c) na osnovu zapamćenih prenosa, odrede se vrednosti prenosa za dva najstarija razreda, C i P

d) na osnovu C i P izračuna se indikator V koji pokazuje da li je rezultat ispravan

Ovaj postupak je ilustrovan u nekoliko narednih primera za slučaj četvorocifrenih brojeva ($n = 4$).

Primer 1: Sabrati brojeve $A = +3$ i $B = +4$.

puni komplement A: 0011

puni komplement B: 0100

zbir: $0111 = -0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = +7$ (tačan rezultat)

prenosi: 0000

$C = 0$, $P = 0$, $V = 0$, ispravno

Primer 2: Sabrati brojeve $A = +3$ i $B = -4$.

puni komplement A: 0011

puni komplement B: 1100

zbir: $1111 = -1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -1$ (tačan rezultat)

prenosi: 0000

$C = 0$, $P = 0$, $V = 0$, ispravno

Primer 3: Sabrati brojeve $A = +7$ i $B = +1$.

puni komplement A: 0111

puni komplement B: 0001

zbir: $1000 = -1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = -8$ (netačan rezultat)

prenosi: 0111

$C = 0$, $P = 1$, $V = 1$, neispravno (rezultat van opsega)

Primer 4: Sabrati brojeve $A = +7$ i $B = -7$.

puni komplement A: 0111

puni komplement B: 1001

zbir: $0000 = -0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 = 0$ (tačan rezultat)

prenosi: 1111

$C = 1$, $P = 1$, $V = 0$, ispravno

Primer 5: Sabrati brojeve $A = -7$ i $B = -4$.

puni komplement A: 1001

puni komplement B: 1100

zbir: $0101 = -0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = +5$ (netačan rezultat)

prenosi: 1000

$C = 1$, $P = 0$, $V = 1$, neispravno (rezultat van opsega)

Da bi rezultat sabiranja binarnih brojeva zapisanih u komplementu dvojke bio ispravan, treba da pripada dozvoljenom opsegu brojeva koji se mogu predstaviti datim brojem cifara. Ukoliko to nije slučaj, dolazi do prekoračenja i moguće greške prilikom izračunavanja. Prekoračenje se javlja, na primer, pri sabiranju brojeva $+6_{(10)}$ i $+7_{(10)}$:

puni komplement $+6_{(10)}$: 0110

puni komplement $+7_{(10)}$: 0111

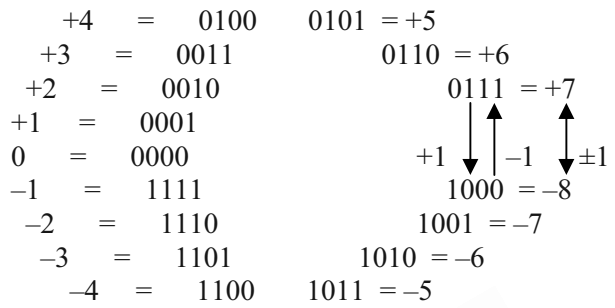
zbir: $1101 = -1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = -3$

prenosi: 0110

$C = 0$, $P = 1$, $V = 1$, neispravan rezultat

Kao što se vidi iz primera, nakon sabiranja dobijen je netačan rezultat $-3_{(10)}$, umesto ispravnog $+13_{(10)}$. Do greške je došlo zato što se u komplementu dvojke pomoću četiri binarne cifre mogu predstaviti samo brojevi iz skupa {-8,...,0,…,7}, a $+13_{(10)}$ ne pripada ovom skupu.

Može se pokazati da između celih brojeva predstavljenih u komplementu dvojke postoji kružna povezanost koja nastaje zbog toga što se najveća i najmanja vrednost iz dozvoljenog opsega razlikuju samo za 1. Tako, za $n = 4$, kružna povezanost se može prikazati kao na slici 1.1.



Slika 1.1 Kružna povezanost brojeva predstavljenih u komplementu dvojke

Kao što se vidi, kada se +7 (0111) sabere sa 1 dobija se –8 (1000), odnosno kada se od –8 oduzme 1 dobija se +7. Zbog kružne povezanosti, ukoliko se desi prekoračenje opsega, ono se propagira po krugu. Tako, u primeru kojim je ilustrovano prekoračenje, dobijeni rezultat $+13_{(10)}$ može se propagirati po krugu za 6 pozicija od broja +7 (+7 je najveći broj iz opsega, a ispravan rezultat je od njega veći za 6), čime se stiže upravo do broja –3, što je bio rezultat sabiranja.

1.2.2 Predstavljanje realnih brojeva

Rešavanje mnogih problema zahteva manipulaciju ne samo nad celim, već i nad realnim brojevima. Realni brojevi se u računaru obično predstavljaju u tzv. pokretnom zarezu (*floating point*).

Pokretni zarez

Pokretni zarez predstavlja način zapisivanja realnih brojeva pomoću tri komponente: znaka Z, eksponenta E i mantise M. Znak određuje da li je broj pozitivan ili negativan, mantisa sadrži značajne cifre broja, a eksponent služi za skaliranje. Decimalna vrednost V ovako zapisanog realnog broja računa se po formuli:

$$V = (Z)\ M{\cdot}q^{E}$$

gde je q osnova brojnog sistema.

Brojevi sa obično normalizuju tako što se decimalna tačka u mantisi postavlja desno od prve nenulte cifre. Na primer, brojevi $5.138{\cdot}10^{38}$ i $3.98{\cdot}10^{-24}$ jesu normalizovani, dok brojevi $513.8{\cdot}10^{36}$ i $39.8{\cdot}10^{-25}$ nisu normalizovani.

Postoje različiti standardi za predstavljanje brojeva u pokretnom zarezu. Broj cifara koje se koriste za predstavljanje navedenih komponenata, kao i formati u kojima su one zapisane razlikuju se u zavisnosti od primenjenog standarda. U

računaru se najčešće koristi binarna reprezentacija realnih brojeva zapisanih u formatu pokretnog zareza standardizovana od strane organizacije *IEEE* (*Institute of Electrical and Electronics Engineers*). Standard čiji je zvaničan naziv *IEEE* 754 definisan je u dva formata: sa jednostrukom ili običnom preciznošću i sa dvostrukom preciznošću. U nastavku će biti predstavljen samo jednostruki format *IEEE* 754 standarda.

Po standardu *IEEE* 754, realni broj u pokretnom zarezu zapisuje se u obliku 32-bitne binarne reči prikazane na slici 1.2 u kojoj je:

- bit na mestu najveće težine rezervisan za znak
- sledećih 8 bitova (1 bajt) namenjeno predstavljanju eksponenta
- preostalih 23 bita namenjeno predstavljanju mantise

Slika 1.2 Zapis u pokretnom zarezu po standardu IEEE 754 (jednostruki format)

Zapisani broj je pozitivan ako binarna cifra koja predstavlja znak (Z) ima vrednost 0, a negativan ako ova cifra ima vrednost 1.

Pomoću 8 bitova predviđenih za predstavljanje eksponenta mogu se zapisati decimalni brojevi od 0 do 255. Međutim, zbog potrebe da se omogući da eksponent, osim pozitivnih, može da ima i negativne vrednosti, navedeni opseg je oduzimanjem transliran (pomeren) za 127. Tako je dobijeno da vrednost eksponenta pripada opsegu od –127 (0 – 127 = –127) do 128 (255 – 127 = 128). Prilikom generisanja zapisa, stvarni eksponent treba najpre uvećati za 127, a zatim ga u binarnom obliku uključiti u zapis. Ovako definisan eksponent naziva se *uvećanim eksponentom*. Iz ovoga sledi da se prilikom određivanja decimalne vrednosti eksponenta mora od decimalne vrednosti 8-bitnog eksponenta u zapisu oduzeti broj 127.

Neka su m_1, m_2, ..., m_{23} biti mantise u redosledu prikazanom na slici. Po razmatranom standardu, mantisa je normalizovana, što znači da je njen bit najveće težine uvek jednak 1, pa stvarna mantisa ima 24-bitni oblik $1.m_1m_{2...}m_{23}$. Kao što se vidi, bit najveće težine je izostavljen iz zapisa, jer je unapred poznat. Decimalna vrednost mantise određuje se po formuli:

$$M = 2^0 + m_1 \cdot 2^{-1} + m_2 \cdot 2^{-2} + ... + m_{22} \cdot 2^{-22} + m_{23} \cdot 2^{-23} \qquad (5)$$

Pošto je $2^0 = 1$, a zbir ostalih članova u formuli manji od 1 (što se može matematički dokazati), sledi da vrednost mantise mora biti između 1 i 2.

Po standardu *IEEE* 754, decimalni broj 0 može se zapisati na dva načina:

- pomoću 32 nule
- pomoću jedinice i 31 nule

U nastavku je opisano kako se zadati decimalni broj može predstaviti u pokretnom zarezu i obrnuto, kako se na osnovu zadatog zapisa može pronaći decimalni broj kome taj zapis odgovara.

Postupak nalaženja decimalne vrednosti binarnog broja zapisanog u pokretnom zarezu po standardu *IEEE* 754 odvija se na sledeći način:

1) određivanje znaka Z: ako je bit najveće težine u zapisu 0, broj je pozitivan (+), a ako je 1, broj je negativan (–)
2) nalaženje eksponenta E: binarnu vrednost eksponenta (bajt koji sledi posle bita najveće težine) konvertovati u decimalnu vrednost, a zatim od dobijenog decimalnog broja oduzeti broj 127
3) određivanje mantise M: primeniti formulu (5) za računanje mantise na poslednja 23 bita u zapisu
4) računanje decimalne vrednosti broja V: primeniti formulu $V = (Z)\ M{\cdot}2^E$

Opisani postupak biće ilustrovan na jednom primeru. Neka je dat zapis broja u pokretnom zarezu $01000000111100000000000000000000_{(2)}$. Da bi se odredila decimalna vrednost ovog broja, najpre u zapisu treba uočiti sve njegove komponente.

0	10000001	11100000000000000000000
Z	E	M

Zatim, za uočene komponente, treba primeniti prethodno navedene korake:

1) određivanje znaka Z: bit najveće težine je 0, pa je broj pozitivan (+)
2) nalaženje eksponenta E: binarna vrednost eksponenta je 10000001, što odgovara decimalnoj vrednosti $X = 1{\cdot}2^7 + 1{\cdot}2^0 = 129$; pošto je eksponent u zapisu uvećan, njegova stvarna vrednost predstavlja razliku 129 – 127 = 2
3) određivanje mantise M: prema formuli za računanje mantise dobija se da je
$$M = 2^0 + 1{\cdot}2^{-1} + 1{\cdot}2^{-2} + 1{\cdot}2^{-3} = 1 + 1/2 + 1/4 + 1/8 = 15/8$$
4) računanje decimalne vrednosti broja V:
$$V = (Z)\ M{\cdot}2^E = +(15/8)\cdot 2^2 = +7.5_{(10)}$$

Po standardu *IEEE* 754, postupak konvertovanja decimalnih realnih brojeva u zapis u pokretnom zarezu obavlja se na sledeći način:

1) celobrojni deo realnog broja konvertuje se u binarni zapis po ranije opisanom standardnom postupku konverzije celih brojeva iz decimalnog u binarni brojni sistem
2) razlomljeni deo decimalnog broja konvertuje se u binarni zapis tako što se
   - pomnoži osnovom brojnog sistema 2
   - od dobijenog rezultata, ceo deo se proglasi rezultujućom cifrom
   - razlomljeni deo dobijenog rezultata se pomnoži osnovom brojnog sistema 2 i postupak se ponavlja kroz određen broj ciklusa
3) od binarnih zapisa dobijenih u koracima 1) i 2) formira se binarna vrednost broja tako što se najpre navede rezultat koraka 1), zatim tačka i nakon toga rezultat koraka 2)
4) dobijeni binarni broj se normalizuje
5) na osnovu normalizovanog oblika odrede se komponente Z, E i M i formira zapis u pokretnom zarezu

Ovaj postupak biće ilustrovan na primeru nalaženja zapisa u pokretnom zarezu koji odgovara realnom broju iz prethodnog primera $R = 7.5$. Ceo deo ovog broja je 7. On se u binarnom obliku predstavlja sa $111_{(2)}$. Razlomljeni deo broja R je 0.5. Binarni zapis ovog dela broja se dobija na sledeći način:

	rezultat množenja	ceo deo
$0.5 \cdot 2 =$	1.0	(1)
$0.0 \cdot 2 =$	0.0	(0)

binarni zapis razlomljenog dela = $10000.....{}_{(2)}$

Prema koraku 3) formira se binarna vrednost polaznog broja:

$$7.5_{(10)} = 111.100000000...{}_{(2)}$$

Normalizacijom desne strane prethodnog izraza dobija se da se polazni broj može napisati i u obliku:

$$7.5_{(10)} = 1.11100000000...{}_{(2)} \cdot 2^2$$

Iz normalizovanog oblika može se zaključiti da je:

- znak broja pozitivan (+)
- vrednost uvećanog eksponenta $E = 2 + 127 = 129_{(10)} = 10000001_{(2)}$

- vrednost mantise $M = 11100000000000000000000_{(2)}$

Tako je dobijeno da se broj 7.5 se u pokretnom zarezu po standardu *IEEE* 754 (jednostruki format) predstavlja u vidu binarnog niza:

$$0\ 10000001\ 11100000000000000000000_{(2)}$$

što je polazni broj u prethodnom primeru.

Sledi još jedan primer nalaženja zapisa u pokretnom zarezu koji odgovara realnom broju $R = 123.45$. Ceo deo ovog broja je 123. On se konvertuje u binarni oblik na sledeći način:

	rezultat deljenja	ostatak	
123 : 2 =	61	(1)	(LSB)
61 : 2 =	30	(1)	
30 : 2 =	15	(0)	
15 : 2 =	7	(1)	
7 : 2 =	3	(1)	
3 : 2 =	1	(1)	
1 : 2 =	0	(1)	(MSB)

$$123_{(10)} = 1111011_{(2)}$$

Razlomljeni deo polaznog broja je 0.45. Binarni zapis ovog dela dobija se na način opisan u koraku 2:

	rezultat množenja	ceo deo
$0.45 \cdot 2 =$	0.9	(0)
$0.9 \cdot 2 =$	1.8	(1)
$0.8 \cdot 2 =$	1.6	(1)
$0.6 \cdot 2 =$	1.2	(1)
$0.2 \cdot 2 =$	0.4	(0)
$0.4 \cdot 2 =$	0.8	(0)

binarni zapis razlomljenog dela = $011100.....{}_{(2)}$

Pošto je poslednji rezultat množenja 0.8, u narednim ciklusima će se vrednosti $1100_{(2)}$ ponavljati (razlomljeni deo 0.8 se već pojavio u trećoj vrsti). Tako se dobija da je tačna vrednost polaznog broja:

$$123.45_{(10)} = 1111011.011100110011001100110011001100110011001100110011001100...{}_{(2)}$$

Ovaj primer ilustruje tipičnu situaciju u kojoj se konačan racionalni decimalni broj konvertuje u beskonačno periodičan racionalni binarni broj. Dobijeni binarni broj se može napisati u normalizovanom obliku kao:

$$123.45_{(10)} = 1.1110110111001100110011001100110011001100110011001100...{}_{(2)} \cdot 2^6$$

Na osnovu poslednjeg izraza, formira se zapis u pokretnom zarezu na sledeći način:

- znak broja je pozitivan
- uvećani eksponent se računa kao $E = 6 + 127 = 133_{(10)} = 10000101_{(2)}$
- mantisa je $M = 11101101110011001100110_{(2)}$

Realan broj 123.45 se u pokretnom zerezu po standardu IEEE 754 (jednostruki format) predstavlja u obliku:

$$0\ 10000101\ 11101101110011001100110_{(2)}.$$

1.2.3 Predstavljanje podataka znakovnog tipa

Prilikom komunikacije korisnika računara sa računarom neophodno je obezbediti da se podaci pojavljuju u obliku koji je čitljiv za čoveka. Stoga podaci treba da budu predstavljeni u vidu znakova iz određenog skupa koji obično obuhvata velika i mala slova abecede, decimalne cifre, specijalne znakove i kontrolne znakove. Pod specijalnim znacima podrazumevaju se svi oni znaci koji se javljaju na tastaturama, a nisu ni cifre ni slova. Oni se mogu odštampati. Specijalni znaci su, na primer, !, #, $, %, *, (,), =, +, -, ? i drugi. Kontrolni znaci su znaci koji se ne mogu odštampati niti prikazati na ekranu računara, već služe za upravljanje ulazno/izlaznim uređajima. Primer kontrolnih znakova su zvučni signal, znaci za pomeranje papira štampača i slično.

U praksi se koristi više metoda za binarno predstavljanje znakova. Najpoznatiji od njih je standard *ASCII* (*American Standard Code for Information Interchange*). Po ovom standardu, znakovi se predstavljaju u vidu 8-bitnog binarnog broja. Skup *ASCII* znakova dat u tabeli 1.5 daje jednoznačnu vezu između znakova (kolone *ASCII*) i njihovih decimalnih (kolone *Dec*), odnosno heksadecimalnih (kolone *Hex*) kodova.

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
0	00	NUL	32	20		64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	“	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	’	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	TAB	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Tabela 1.5 Skup *ASCII* znakova

Vežbanja

1. Šta je pozicioni brojni sistem? Na koji način se bilo koji pozitivan ceo broj može predstaviti u pozicionom brojnom sistemu?

2. Naći decimalne i heksadecimalne vrednosti sledećih binarnih brojeva:

   a) $100111_{(2)}$
   b) $11010_{(2)}$
   c) $1000101_{(2)}$
   d) $1010001_{(2)}$
   e) $101001110_{(2)}$

3. Predstaviti u binarnom i heksadecimalnom obliku zadate decimalne brojeve:

   a) $56_{(10)}$
   b) $88_{(10)}$
   c) $124_{(10)}$
   d) $231_{(10)}$
   e) $426_{(10)}$

4. Konvertovati date heksadecimalne brojeve u binarne i decimalne brojeve:

   a) $58_{(16)}$
   b) $40C_{(16)}$
   c) $21E_{(16)}$
   d) $1F1_{(16)}$
   e) $426_{(16)}$

5. Data su dva neoznačena binarna broja A i B. Izračunati: $A+B$ i $A-B$.

   a) $A = 1011010$ $B = 11011$
   b) $A = 1011100$ $B = 110101$
   c) $A = 1001011$ $B = 101111$
   d) $A = 10011111$ $B = 101011$
   e) $A = 101001110$ $B = 11100010$

6. Data su dva neoznačena binarna broja A i B. Izračunati: $A \cdot B$ i $A : B$.

   a) $A = 100111$ $B = 1101$
   b) $A = 11001$ $B = 101$
   c) $A = 101010$ $B = 110$

7. Predstaviti u komplementu dvojke sledeće brojeve:

   a) $75_{(10)}$
   b) $355_{(10)}$
   c) $-33_{(10)}$
   d) $-82_{(10)}$
   e) $-100_{(10)}$

8. Izračunati decimalnu vrednost sledećih binarnih brojeva zapisanih u komplementu dvojke:

   a) $100101010_{(2)}$
   b) $0110110_{(2)}$
   c) $11010010_{(2)}$
   d) $1111101_{(2)}$
   e) $00011111_{(2)}$

9. Navesti formulu za određivanje mogućeg opsega neoznačenih binarnih brojeva zapisanih sa n cifara. Primenom ove formule odrediti sa koliko se najmanje cifara mogu zapisati sledeći neoznačeni brojevi:

   a) $47_{(10)}$
   b) $82_{(10)}$
   c) $100_{(10)}$
   d) $521_{(10)}$

10. Na osnovu formule za određivanje dozvoljenog opsega označenih binarnih brojeva zapisanih u komplementu dvojke sa n cifara odrediti sa koliko se najmanje cifara mogu zapisati sledeći označeni brojevi:

    a) $71_{(10)}$
    b) $59_{(10)}$
    c) $-121_{(10)}$
    d) $-99_{(10)}$

11. Navedene decimalne brojeve predstaviti u komplementu dvojke, a zatim naći njihov zbir.

a) $133_{(10)}$ i $-91_{(10)}$
b) $35_{(10)}$ i $-17_{(10)}$
c) $-22_{(10)}$ i $40_{(10)}$
d) $-14_{(10)}$ i $-7_{(10)}$

12. Brojeve $A = -53_{(10)}$ i $B = 9_{(10)}$ predstaviti u komplementu dvojke, a zatim naći njihovu razliku $A–B$.

13. Detaljno objasniti standard *IEEE* 754 za predstavljanje brojeva u pokretnom zarezu.

14. Odrediti decimalnu vrednost zadatih brojeva zapisanih u pokretnom zarezu po standardu *IEEE* 754:

a) $01000000111000000000000000000000_{(2)}$
b) $40C00000_{(16)}$
c) $42400000_{(16)}$
d) $11000010001110000000000000000000_{(2)}$
e) $C2700000_{(16)}$

15. Zadate decimalne brojeve zapisati u pokretnom zarezu u jednostrukom formatu po standardu *IEEE* 754.

a) $17.74_{(10)}$
b) $-8.25_{(10)}$
c) $240.1_{(10)}$
d) $-15_{(10)}$
e) $22.56_{(10)}$

2 Logički elementi

U osnovi funkcionisanja svakog računarskog sistema je prosleđivanje informacije o tome da li u datom trenutku negde u sistemu postoji signal ili ne. Ova vrsta informacije se može predstaviti binarnim ciframa 0 i 1. Na primer, može se usvojiti da vrednost 0 označava odsustvo signala, a vrednost 1 da signal postoji. Pošto binarne vrednosti 0 i 1 tako dobijaju logičko značenje, one se u računarskim sistemima često nazivaju i *logičkim vrednostima*.

U prethodnom poglavlju uveden je pojam binarnog broja koji predstavlja niz binarnih cifara 0 i 1 određene dužine. Takođe je pokazano kako se nad binarnim brojevima mogu obavljati različite aritmetičke operacije (sabiranje, oduzimanje, množenje i deljenje). Zajednička osobina svih aritmetičkih operacija jeste da se binarni brojevi nad kojima se one obavljaju posmatraju kao jedinstvena celina. To je neophodno zato što kod ovih operacija postoji međusobni uticaj između različitih pozicija u binarnim brojevima. Naime, na rezultat aritmetičke operacije na određenoj poziciji utiču ne samo binarne vrednosti brojeva koji učestvuju u operaciji na toj poziciji, već i binarne vrednosti koje se nalaze na susednim pozicijama.

U računarskim sistemima često se javlja potreba za poređenjem binarnih brojeva na nivou svake binarne cifre pojedinačno (na primer, utvrđivanje da li su dva skupa signala koja se pojavljuju na različitim lokacijama u sistemu istovetna, pri čemu se svaki signal predstavlja jednom logičkom vrednošću). S obzirom da se u tu svrhu ne mogu koristiti aritmetičke operacije, u računarskim sistemima svoju široku primenu nalaze i tzv. logičke operacije.

Logičke operacije se izvode nad binarnim brojevima na nivou pojedinačnih cifara. To znači da se binarni brojevi ne posmatraju kao jedinstvene celine (kao u slučaju aritmetičkih operacija), već kao niz međusobno nezavisnih binarnih vrednosti. Rezultat logičke operacije nad binarnim brojevima predstavlja niz

parcijalnih rezultata dobijenih primenom te logičke operacije nad binarnim ciframa na svim pozicijama u zadatim binarnim brojevima.

U nastavku će biti prikazane logičke operacije sabiranja, množenja, negacije i ekskluzivnog sabiranja (prve tri predstavljaju osnovu Bulove algebre). Osim njih, biće opisane i logičke operacije *NILI*, *NI* i *EKSNILI*.

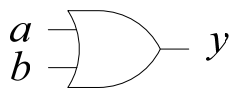
Pošto se logička operacija, u stvari, primenjuje nad pojedinačnim binarnim vrednostima, broj mogućih kombinacija ovih binarnih vrednosti je konačan. Stoga se značenje logičke operacije često opisuje pomoću kombinacione tablice. U tablici se navode sve moguće kombinacije ulaznih vrednosti (binarne vrednosti nad kojima se primenjuje logička operacija), kao i rezultat operacije koji odgovara svakoj ulaznoj kombinaciji. Time je logička operacija u potpunosti definisana.

Logičke operacije se u računarskim sistemima realizuju pomoću komponenata koje se nazivaju logičkim kolima. Logička kola imaju ulaze, na koje se dovode signali koji odgovaraju binarnim vrednostima nad kojima treba obaviti operaciju, i izlaz koji predstavlja rezultat operacije. Različitim logičkim operacijama odgovaraju različiti grafički simboli logičkih kola kojima se operacije realizuju.

2.1. Logičko sabiranje

Logičko sabiranje ili *ILI* operacija (*OR*) je binarna operacija jer omogućava sabiranje dve binarne vrednosti, a i b. Simbol za ovu operaciju je '+', pa se rezultat operacije y može predstaviti kao $y = a+b$. S obzirom da binarne vrednosti mogu biti samo 0 ili 1, broj mogućih kombinacija za sabiranje je 4. U kombinacionoj tablici na slici 2.1 navedene su sve moguće kombinacije za a i b, kao i rezultat sabiranja y za svaku od njih. Na istoj slici prikazan je i grafički simbol logičkog *ILI* kola koje realizuje logičko sabiranje.

a	*b*	*y*
0	0	0
0	1	1
1	0	1
1	1	1



Slika 2.1 Kombinaciona tablica i grafički simbol za logičku *ILI* operaciju

Na osnovu kombinacione tablice može se zaključiti da je rezultat logičkog sabiranja 0 samo ako su obe binarne vrednosti koje se sabiraju jednake 0. To je ekvivalentno tvrđenju da je rezultat logičkog sabiranja 1 ako je bar jedna od vrednosti koje se sabiraju jednaka 1.

Postupak logičkog sabiranja dva višecifrena binarna broja odvija se tako što se posebno sabiraju binarne vrednosti na svakoj poziciji, a zatim se tako dobijeni rezultati posmatraju kao jedan binarni broj. To će biti ilustrovano na jednom primeru.

Data su dva binarna broja koja treba logički sabrati: $A = 10101_{(2)}$ i $B = 11001_{(2)}$. Rezultat Y je, takođe, binarni broj od pet cifara koje se nalaze na pozicijama 4, 3, 2, 1 i 0, posmatrano s leva na desno.

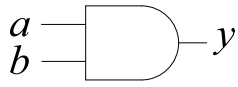
pozicija	4 3 2 1 0
A	1 0 1 0 1
B	1 1 0 0 1
$Y = A+B$	1 1 1 0 1

Vrednost na poziciji 0 u rezultatu se dobija kada se logički saberu vrednosti na poziciji 0 u brojevima A i B, tj. 1+1 = 1. Vrednost na poziciji 1 se dobija kada se logički saberu vrednosti na poziciji 1 u brojevima A i B, tj. 0+0 = 0. Na sličan način dobijaju se i vrednosti na pozicijama 2, 3 i 4, tako da se kao konačan rezultat logičkog sabiranja dobija $Y = A+B = 11101_{(2)}$. Treba uočiti da se dobijeni rezultat razlikuje od rezultata aritmetičkog sabiranja sprovedenog nad zadatim brojevima koji iznosi $101110_{(2)}$.

2.2. Logičko množenje

Logičko množenje ili *I* operacija (*AND*) je binarna operacija koja omogućava množenje dve binarne vrednosti, a i b. Simbol za ovu operaciju je '·', pa se rezultat operacije y može predstaviti kao $y = a \cdot b$, ili jednostavnije $y = ab$. Slično kao kod logičkog sabiranja, broj mogućih kombinacija za množenje je 4. Sve moguće kombinacije za a i b, kao i rezultat množenja y za svaku od njih prikazane se u kombinacionoj tablici na slici 2.2. Na slici je, takođe, prikazan i grafički simbol logičkog kola koje realizuje logičko množenje.

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1



Slika 2.2 Kombinaciona tablica i grafički simbol za logičku *I* operaciju

Iz prikazane kombinacione tablice može se videti da je rezultat logičkog množenja 1 samo ako su obe binarne vrednosti jednake 1, tj. rezultat je 0 ako je bar jedna od binarnih vrednosti koje se množe jednaka 0.

Dva višecifrena binarna broja se logički množe tako što se posebno množe binarne vrednosti na svakoj poziciji, a zatim se tako dobijeni rezultati posmatraju kao jedan binarni broj. Ovaj postupak će biti prikazan na primeru.

Neka su data dva binarna broja koja treba logički pomnožiti: $A = 11010_{(2)}$ i $B = 11001_{(2)}$. Rezultat Y ima, takođe, pet cifara koje se nalaze na pozicijama 4, 3, 2, 1 i 0, posmatrano s leva na desno.

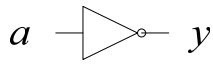
pozicija	4 3 2 1 0
A	1 1 0 1 0
B	1 1 0 0 1
Y=AB	1 1 0 0 0

Vrednost na poziciji 0 u rezultatu se dobija kada se logički pomnože vrednosti na poziciji 0 u brojevima A i B, tj. $0 \cdot 1 = 0$. Vrednost na poziciji 1 se dobija kada se logički pomnože vrednosti na poziciji 1 u brojevima A i B, tj. $1 \cdot 0 = 0$. Na sličan način dobijaju se i vrednosti na pozicijama 2, 3 i 4, tako da se kao konačan rezultat množenja dobija $Y = AB = 11000_{(2)}$.

2.3. Komplementiranje

Komplementiranje, negacija, invertovanje ili *NE* operacija (*NOT*) predstavlja unarnu operaciju jer se izvodi nad samo jednom binarnom vrednošću, a. Simbol za ovu operaciju je ’¯’, pa se rezultat operacije y može predstaviti kao $y = \bar{a}$. S obzirom da binarna vrednost može biti samo 0 ili 1, broj mogućih kombinacija za komplementiranje je 2. U kombinacionoj tablici na slici 2.3 navedene su obe kombinacije za a, kao i rezultat komplementiranja y za svaku od njih. Takođe, prikazan je i grafički simbol logičkog kola koje realizuje komplementiranje. U računarskoj tehnici ovo logičko kolo se obično naziva invertorom.

a	*y*
0	1
1	0



Slika 2.3 Kombinaciona tablica i grafički simbol za logičku *NE* operaciju

Na osnovu kombinacione tablice može se izvesti zaključak da je rezultat komplementiranja uvek suprotna logička vrednost od one koju treba komplementirati. Takođe treba zapaziti da je rezultat dvostrukog komplementiranja neke binarne vrednosti sama ta vrednost (komplement komplementa neke vrednosti je polazna vrednost).

Višecifreni binarni broj se komplementira tako što se posebno komplementira binarna vrednost na svakoj poziciji, a zatim se tako dobijeni rezultat posmatra kao jedan binarni broj. Ovaj postupak će biti prikazan na primeru.

Neka je dat binarni broj koga treba komplementirati: $A = 1001011_{(2)}$. Rezultat Y ima sedam cifara koje se nalaze na pozicijama 6, 5, ..., 1 i 0, posmatrano s leva na desno.

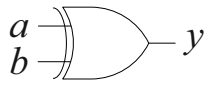
pozicija	6	5	4	3	2	1	0
A	1	0	0	1	0	1	1
$Y = \bar{A}$	0	1	1	0	1	0	0

Vrednost na poziciji 0 u rezultatu se dobija kada se komplementira vrednost na poziciji 0 u broju A, tj. $\bar{1} = 0$. Vrednost na poziciji 1 se dobija kada se komplementira vrednost na poziciji 1 u broju A, tj. $\bar{1} = 0$. Na sličan način se dobijaju i vrednosti na pozicijama 2, 3, 4, 5 i 6, tako da se kao konačan rezultat komplementiranja dobija $Y = \bar{A} = 0110100_{(2)}$.

2.4. Logička EKSILI operacija

Ekskluzivno sabiranje ili *EKSILI* operacija (*XOR*) je binarna operacija jer se obavlja nad dvema binarnim vrednostima, a i b. Simbol za ovu operaciju je '$\oplus$', pa se rezultat operacije y može predstaviti kao $y = a \oplus b$. Slično kao kod logičkog sabiranja i množenja, broj mogućih kombinacija za ekskluzivno sabiranje je 4. Sve moguće kombinacije za a i b, kao i rezultat ekskluzivnog sabiranja y za svaku od njih prikazane se u kombinacionoj tablici na slici 2.4. Na slici je, takođe, prikazan i grafički simbol logičkog kola koje realizuje logičko ekskluzivno sabiranje.

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



Slika 2.4 Kombinaciona tablica i grafički simbol za *EKSILI* operaciju

Rezultat ekskluzivnog sabiranja je 0 samo ako su obe binarne vrednosti koje se sabiraju međusobno jednake. Takođe, može se reći i da je rezultat ekskluzivnog sabiranja 1 samo ako su vrednosti koje se sabiraju međusobno različite. Iz ovog poslednjeg tvrđenja je potekao i naziv ove operacije o isključivosti tj. ekskluzivnosti.

Dva višecifrena binarna broja se ekskluzivno sabiraju tako što se posebno ekskluzivno sabiraju binarne vrednosti na svakoj poziciji, a zatim se tako dobijeni rezultati posmatraju kao jedan binarni broj. Ovaj postupak će biti prikazan na primeru.

Neka su data dva binarna broja koja treba sabrati ekskluzivno *ILI* operacijom: $A = 10010_{(2)}$ i $B = 11111_{(2)}$. Rezultat Y ima pet cifara koje se nalaze na pozicijama 4, 3, 2, 1 i 0, posmatrano s leva na desno.

pozicija	4 3 2 1 0
A	1 0 0 1 0
B	1 1 1 1 1
$Y = A \oplus B$	0 1 1 0 1

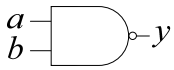
Vrednost na poziciji 0 u rezultatu se dobija kada se ekskluzivno saberu vrednosti na poziciji 0 u brojevima A i B, tj. $0 \oplus 1 = 1$. Vrednost na poziciji 1 se dobija kada se ekskluzivno saberu vrednosti na poziciji 1 u brojevima A i B, tj. $1 \oplus 1 = 0$. Na sličan način dobijaju se i vrednosti na pozicijama 2, 3 i 4, tako da se kao konačan rezultat ekskluzivnog sabiranja dobija $Y = A \oplus B = 01101_{(2)}$.

2.5. Logička NI operacija

Logička *NI* operacija (*NAND*) je binarna operacija koja predstavlja negaciju množenja dve binarne vrednosti, a i b. Rezultat ove operacije y zapisuje se kao $y = \overline{a \cdot b}$. Slično kao kod logičkog množenja, broj mogućih kombinacija ulaznih promenljivih je 4. Sve moguće kombinacije za a i b, kao i rezultat operacije y za svaku od njih, prikazane se u kombinacionoj tablici na slici 2.5. Na slici je, takođe, prikazan i grafički simbol logičkog kola koje realizuje ovu logičku operaciju.

Iz kombinacione tablice može se videti da je rezultat *NI* operacije 1 ako je bar jedna ulazna binarna vrednost jednaka 0, tj. rezultat je 0 ako su obe binarne vrednosti jednake 1.

a	*b*	*y*
0	0	1
0	1	1
1	0	1
1	1	0



Slika 2.5 Kombinaciona tablica i grafički simbol za logičku *NI* operaciju

Nad dva višecifrena binarna broja, *NI* operacija se primenjuje tako što se posebno primeni nad svakim razredom u brojevima, a zatim se tako dobijeni rezultati uzimaju kao jedan binarni broj. Ovaj postupak će biti prikazan na primeru.

Neka su data dva binarna broja nad kojima treba primeniti logičku *NI* operaciju: $A = 101010_{(2)}$ i $B = 110011_{(2)}$. Rezultat *Y* ima, takođe, šest cifara koje se nalaze na pozicijama 5, 4, 3, 2, 1 i 0, posmatrano s leva na desno.

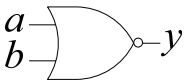
pozicija	5 4 3 2 1 0
A	1 0 1 0 1 0
B	1 1 0 0 1 1
$Y = \overline{A \cdot B}$	0 1 1 1 0 1

Vrednost na poziciji 0 u rezultatu se dobija kada se logički pomnože vrednosti na poziciji 0 u brojevima *A* i *B*, tj. 0·1 = 0, a zatim se ova vrednost komplementira i dobije rezultat 1. Vrednost na poziciji 1 se dobija kada se logički pomnože vrednosti na poziciji 1, tj. 1·1 = 1, pa se dobijena vrednost komplementira i postane 0. Na sličan način dobijaju se i vrednosti na pozicijama 2, 3, 4 i 5, tako da se kao konačan rezultat NI operacije dobija $Y = \overline{A \cdot B} = 011101_{(2)}$.

2.6. Logička NILI operacija

Logička *NILI* operacija (*NOR*) je binarna operacija koja predstavlja negaciju logičkog sabiranja dve binarne vrednosti, *a* i *b*. Rezultat ove operacije se izražava kao $y = \overline{a + b}$. Sve moguće kombinacije vrednosti za *a* i *b*, kao i rezultat operacije *y* za svaku od njih, navedene se u kombinacionoj tablici na slici 2.6. Na istoj slici je dat i grafički simbol logičkog *NILI* kola.

a	*b*	*y*
0	0	1
0	1	0
1	0	0
1	1	0



Slika 2.6 Kombinaciona tablica i grafički simbol za logičku *NILI* operaciju

Iz kombinacione tablice sledi da je rezultat *NILI* operacije 0 ako je bar jedna ulazna binarna vrednost jednaka 1, tj. rezultat je 1 samo ako su obe binarne vrednosti jednake 0.

Nad dva višecifrena binarna broja, *NILI* operacija se primenjuje tako što se posebno primeni nad svakim razredom u brojevima, pa se tako dobijeni rezultati posmatraju kao jedinstven binarni broj. Ovaj postupak će biti prikazan na primeru.

Neka su data dva binarna broja nad kojima treba primeniti logičku *NILI* operaciju: $A = 101010_{(2)}$ i $B = 110011_{(2)}$. Rezultat Y ima, takođe, šest cifara koje se nalaze na pozicijama od 5 do 0, posmatrano s leva na desno.

pozicija	5 4 3 2 1 0
A	1 0 1 0 1 0
B	1 1 0 0 1 1
$Y = \overline{A + B}$	0 0 0 1 0 0

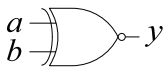
Vrednost na poziciji 0 u rezultatu se dobija kada se logički saberu vrednosti na poziciji 0 u brojevima A i B, tj. 0+1 = 1, a zatim se ova vrednost komplementira i dobije rezultat 0. Vrednost na poziciji 2 se dobija kada se logički saberu vrednosti na poziciji 2, tj. 0+0 = 0, pa se tako dobijena vrednost komplementira i postane 1. Na sličan način dobijaju se i vrednosti na ostalim pozicijama, tako da se kao konačan rezultat *NILI* operacije dobija $Y = \overline{A + B} = 000100_{(2)}$.

2.7. Logička EKSNILI operacija

Logička *EKSNILI* operacija (*XNOR*) je binarna operacija koja predstavlja negaciju ekskluzivnog logičkog sabiranja dve binarne vrednosti, a i b. Rezultat ove operacije se zapisuje u obliku $y = \overline{a \oplus b}$. Za sve moguće kombinacije vrednosti ulaznih promenljivih a i b, rezultat operacije y je dat u kombinacionoj tablici prikazanoj na slici 2.7. Na istoj slici je prikazan i grafički simbol logičkog *EKSNILI* kola.

.

a	*b*	*y*
0	0	1
0	1	0
1	0	0
1	1	1



Slika 2.7 Kombinaciona tablica i grafički simbol za logičku *EKSNILI* operaciju

Iz kombinacione tablice sledi da je rezultat *EKSNILI* operacije 1 ako su vrednosti ulaznih promenljivih jednake, a 0 ako su različite.

Kao i kod ostalih logičkih operacija, primena *EKSNILI* operacije nad dva višecifrena binarna broja obavlja se nad svakim razredom pojedinačno, a onda se od dobijenih pojedinačnih rezultata formira jedinstven binarni broj. Sledi primer koji ilustruje ovaj postupak.

Data su dva binarna broja nad kojima treba primeniti logičku *EKSNILI* operaciju: $A = 111000_{(2)}$ i $B = 100010_{(2)}$. Vrednost na poziciji 0 u rezultatu se dobija kada se vrednosti na poziciji 0 u brojevima A i B ekskluzivno saberu, tj. $0 \oplus 0 = 0$, a zatim se dobijena vrednost komplementira i dobije rezultat 1. Vrednost na poziciji 1 se dobija kada se logički saberu vrednosti na poziciji 1, tj. $0 \oplus 1 = 1$, pa se tako dobijena vrednost komplementira i postane 0. Na sličan način dobijaju se i vrednosti na ostalim pozicijama, tako da se kao konačan rezultat *EKSNILI* operacije dobija $Y = \overline{A \oplus B} = 100101_{(2)}$.

pozicija	5 4 3 2 1 0
A	1 1 1 0 0 0
B	1 0 0 0 1 0
$Y = \overline{A \oplus B}$	1 0 0 1 0 1

Vežbanja

1. Prikazati kombinacione tablice za sledeće logičke operacije:

   a) logičko množenje
   b) logičko sabiranje
   c) komplementiranje
   d) ekskluzivno sabiranje
   e) NI operaciju
   f) NILI operaciju
   g) EKSNILI operaciju

2. Nacrtati grafičke simbole za logička kola koja realizuju logičke operacije *I*, *ILI*, *NE*, *EKSILI*, *NI*, *NILI* i *EKSNILI*.

3. Izvršiti logičke operacije AB, $A+B$, $A \oplus B$ i $\bar{A}$ nad sledećim brojevima:

   a) $A=110101010_{(2)}$ i $B=101011111_{(2)}$
   b) $A=101011010_{(2)}$ i $B=110110101_{(2)}$
   c) $A=11010101_{(2)}$ i $B=01001100_{(2)}$
   d) $A=1101001010_{(2)}$ i $B=0110111000_{(2)}$
   e) $A=11110111_{(2)}$ i $B=11011111_{(2)}$
   f) $A=11011011_{(2)}$ i $B=10101111_{(2)}$
   g) $A=10110100_{(2)}$ i $B=01000001_{(2)}$
   h) $A=11010010_{(2)}$ i $B=01001010_{(2)}$
   i) $A=1101001010_{(2)}$ i $B=0010110101_{(2)}$
   j) $A=10111000_{(2)}$ i $B=10111000_{(2)}$
   k) $A=100000000_{(2)}$ i $B=100000000_{(2)}$

4. Naći aritmetički, a zatim i logički zbir sledećih binarnih brojeva:

   a) $11010110_{(2)}$ i $10011111_{(2)}$
   b) $100101100_{(2)}$ i $110011011_{(2)}$
   c) $1001011_{(2)}$ i $1000110_{(2)}$
   d) $1010101_{(2)}$ i $1100101_{(2)}$
   e) $1111111_{(2)}$ i $1111111_{(2)}$

5. Naći aritmetički, a zatim i logički proizvod sledećih binarnih brojeva:

   a) $1100_{(2)}$ i $1000_{(2)}$
   b) $1110_{(2)}$ i $1001_{(2)}$
   c) $1011_{(2)}$ i $1110_{(2)}$
   d) $101_{(2)}$ i $110_{(2)}$
   e) $111_{(2)}$ i $101_{(2)}$

6. Izvršiti logičke operacije $\overline{A \cdot B}$, $\overline{A + B}$ i $\overline{A \oplus B}$ nad sledećim brojevima:

   a) $A=101010_{(2)}$ i $B=101111_{(2)}$
   b) $A=1011010_{(2)}$ i $B=1110101_{(2)}$
   c) $A=11010101_{(2)}$ i $B=01001100_{(2)}$
   d) $A=1101001010_{(2)}$ i $B=0110111000_{(2)}$
   e) $A=11110111_{(2)}$ i $B=11011111_{(2)}$
   f) $A=11011011_{(2)}$ i $B=10101111_{(2)}$
   g) $A=110100_{(2)}$ i $B=010000_{(2)}$
   h) $A=11010010_{(2)}$ i $B=01001010_{(2)}$
   i) $A=1101010_{(2)}$ i $B=0110101_{(2)}$
   j) $A=10111000_{(2)}$ i $B=10111000_{(2)}$

UNIVERZITET
PRIVATNI
AKREDITOVANI
UNIVERZITET
SINGIDUNUM

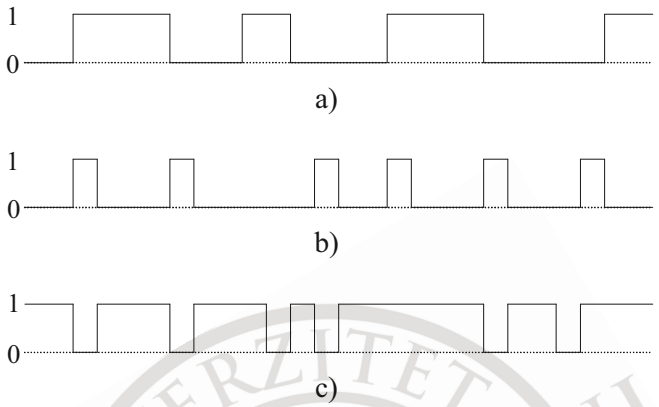
3 Memorijski elementi

Logička kola razmatrana u prethodnom poglavlju imaju osobinu da vrednost na njihovom izlazu zavisi isključivo od vrednosti binarnih signala prisutnih na njihovim ulazima. Međutim, osim logičkih, postoji još jedna široko rasprostranjena klasa digitalnih kola kod kojih vrednost na izlazu zavisi ne samo od postojećih binarnih vrednosti na ulazima, već i od stanja u kome se kolo trenutno nalazi, što je uslovljeno ponašanjem kola u prethodnom periodu. Pod stanjem se podrazumeva skup vrednosti logičkih signala na pojedinim mestima u digitalnom kolu. Ukoliko se promene vrednosti na ulazima, kolo može da pređe u novo stanje ili da ostane u istom stanju. Tokom vremena, stanja se menjaju, pa se kaže da kolo prolazi kroz sekvencu stanja. Zato se digitalno kolo sa opisanim ponašanjem često naziva sekvencijalnim kolom. Da bi se zapamtilo, tj. memorisalo stanje digitalnog kola, koriste se *memorijski elementi*.

Memorijski elementi su elementi koji zadržavaju, odnosno pamte uspostavljena stanja i po prestanku dejstva pobudnih signala koji su ih prouzrokovali. Osnovna karakteristika memorijskih elemenata jeste postojanje stabilnih stanja u kojima se mogu naći i ostati neograničeno vreme, a koja se mogu menjati pod uticajem ulaznih signala. U digitalnoj tehnici kao memorijski elementi koriste se *bistabilana kola* koja imaju dva stabilna stanja. Bistabilno kolo može da memoriše informaciju od jednog bita.

Bitan faktor za ispravan rad memorijskih elemenata je vreme. Postoje različite vrste signala u digitalnim kolima po pitanju njihovog trajanja. Ako trajanje binarnog signala nije ograničeno ni za jednu moguću binarnu vrednost (0 ili 1), kaže se da je signal *potencijalni* ili potencijalnog tipa. Ukoliko je trajanje binarnog signala ograničeno bar za jednu binarnu vrednost, signal je *impulsni* ili impulsnog tipa. Na slici 3.1 prikazana su oba tipa signala. Pod a) je dat potencijalni signal, pod b) impulsni signal kod koga je fiksno trajanje signala koji odgovara binarnoj

vrednosti 1, a pod c) impulsni signal kod koga je ograničeno i fiksno trajanje signala za vrednost 0.



Slika 3.1 Vrste signala: a) potencijalni b) c) impulsni

Impulsni signal može biti *periodičan*, što znači da mu je trajanje ograničeno i fiksno za obe binarne vrednosti 0 i 1. Primer ovakvog signala je dat na slici 3.2.

Slika 3.2 Impulsni periodični signal

Iako je na prethodnim slikama promena vrednosti binarnog signala sa 0 na 1 i obrnuto, sa 1 na 0, prikazana kao trenutna, u realnosti se ona odvija u vremenskom intervalu koji zavisi od tehnoloških karakteristika izvora koji je generisao posmatrani binarni signal. Ako se u nekom trenutku t_i promeni neki ulazni signal, proteći će izvesno vreme Δt dok se ne uspostavi odgovarajuća vrednost na izlazu. Vreme Δt predstavlja kašnjenje signala i zavisi od tehnoloških i strukturnih karakteristika kola. Da bi kolo ispravno radilo, sledeća promena na ulazu može se izvršiti u trenutku t_{i+1} samo ako je zadovoljen uslov $t_{i+1} - t_i \geq \Delta t$. U intervalu od t_i do $t_i + \Delta t$, u digitalnom kolu se odvija prelazni proces tokom koga izlazni signal nije definisan i ne može se koristiti.

Radi jednostavnije analize, kašnjenje Δt se zanemaruje i smatra se da se svaka promena na ulazu trenutno prosleđuje na izlaz kola. Ipak, da ne bi došlo do problema u radu kola, usvojeno je da su promene na ulazu dozvoljene samo u diskretnim vremenskim trenucima t_1, t_2, ..., t_i, t_{i+1} ,... , pri čemu mora da važi uslov

$t_{i+1} - t_i \geq \Delta t$. Zbog ovakvog načina funkcionisanja, kaže se da digitalna kola rade u *diskretnom vremenu*.

Vremenski interval između dva uzastopna trenutka naziva se intervalom takta ili samo *taktom*.

U savremenim računarskim i drugim digitalnim sistemima i uređajima, koriste se memorijski elementi zasnovani na poluprovodničkim tehnologijama koji se nazivaju flip-flop-ovima (FF). Flip-flop-ovi se obično realizuju kao kompozicije logičkih elemenata. Mogu se podeliti na *asinhrone* i *taktovane* ili *sinhrone* FF -ove.

U ovom poglavlju biće predstavljeni asinhroni *RS* FF i sinhroni *RS*, *D*, *T* i *JK* FF-ovi, kao osnovni memorijski elementi koji se koriste za realizaciju složenih sekvencijalnih kola.

3.1. Asinhroni RS flip-flop

RS flip-flop je memorijski element koji ima dva ulaza označena sa *R* i *S* i dva izlaza označena sa z_1 i z_2. Od oznaka ulaza *R* (*Reset*) i *S* (*Set*) potiče naziv samog elementa.

Asinhroni *RS* FF može se realizovati kao kombinaciona mreža logičkih *NILI* ili logičkih *NI* elemenata, kao što je prikazano na slici 3.3. U datim strukturnim šemama postoje povratne veze između izlaza i ulaza, što otežava njihovu analizu i zahteva definisanje posebnih postupaka za tu svrhu.

Slika 3.3 Ashinroni *RS* FF a) sa *NILI* elementima i b) sa *NI* elementima

Na osnovu strukturne šeme prikazane pod a) mogu se izvesti sledeće zavisnosti između izlaza i ulaza:

$$Q(t + \Delta t) = \overline{R} \cdot S + \overline{R} \cdot Q$$

$$z_1 = Q$$

$$z_2 = \overline{S} \cdot \overline{Q}$$

gde je Q izlazni signal na unutrašnjoj liniji koja predstavlja *liniju stanja*. Sa t je označen trenutak promene signala na ulazu, a sa $t + \Delta t$ trenutak kada je završena promena signala na liniji Q usled promene na ulazu. Tokom vremenskog intervala Δt odvija se prelazni proces, pa signali Q, z_1 i z_2 nisu definisani. Uobičajeno je da se $t + \Delta t$ označi sa t +1, što predstavlja *sledeći trenutak* u odnosu na *sadašnji trenutak t*. Slično, Q predstavlja *sadašnje stanje*, a $Q(t + 1)$ *sledeće stanje*. Tako se dobija funkcija

$$Q(t+1) = \overline{R} \cdot S + \overline{R} \cdot Q$$

koja se naziva *funkcijom prelaza* ili *funkcijom sledećeg stanja RS* FF-a sa *NILI* kolima. Funkcije z_1 i z_2 nazivaju se *funkcijama izlaza* posmatranog FF-a.

Za dve logičke promenljive a i b, važi $\overline{a+b} = \overline{a} \cdot \overline{b}$. Ovo tvrđenje se lako može dokazati pomoću tabele 3.1.

a	b	$\overline{a+b}$	$\overline{a} \cdot \overline{b}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Tabela 3.1 Kombinaciona tablica za nalaženje vrednosti $\overline{a+b}$ i $\overline{a} \cdot \overline{b}$

Izraz za funkciju sledećeg stanja izveden je na osnovu navedenog tvrđenja i date strukturne šeme na sledeći način:

$$Q(t+1) = \overline{R + \overline{S+Q}} = \overline{R + \overline{S} \cdot \overline{Q}} = \overline{R} \cdot \overline{\overline{S} \cdot \overline{Q}} = \overline{R} \cdot (S+Q) = \overline{R} \cdot S + \overline{R} \cdot Q$$

Funkcija izlaza z_2 sledi direktno iz navedenog tvrđenja, tj. $z_2 = \overline{S+Q} = \overline{S} \cdot \overline{Q}$.

Sličnom analizom dolazi se do funkcije prelaza i funkcija izlaza asinhronog *RS* FF-a sa *NI* elementima. Tako se dobija

$$Q(t+1) = \overline{S} + R \cdot Q$$

$$z_1 = Q$$

$$z_2 = \overline{R} + \overline{Q}$$

Osim ranije navedenog, za logičke promenljive a i b, važi i sledeće tvrđenje $\overline{a \cdot b} = \overline{a} + \overline{b}$. Dokaz ovog tvrđenja dat je u tabeli 3.2.

a	b	$\overline{a \cdot b}$	$\overline{a} + \overline{b}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Tabela 3.2 Kombinaciona tablica za nalaženje vrednosti $\overline{a \cdot b}$ i $\overline{a} + \overline{b}$

Izraz za funkciju prelaza izveden je na osnovu strukturne šeme prikazane na slici 3.3.b) i prethodnog tvrđenja na sledeći način:

$$Q(t+1) = \overline{S \cdot \overline{R \cdot Q}} = \overline{S} \cdot (\overline{R} + \overline{Q}) = \overline{S} + \overline{\overline{R}} + \overline{Q} = \overline{S} + \overline{\overline{R \cdot Q}} = \overline{S} + R \cdot Q$$

Funkcija izlaza z_2 dobija se direktno iz poslednjeg tvrđenja, tj. $z_2 = \overline{R \cdot Q} = \overline{R} + \overline{Q}$.

Funkcija prelaza i funkcije izlaza u potpunosti definišu način funkcionisanja bilo kog flip-flop-a. Na slici 3.4 date su kombinacione tablice kojima je opisan način funkcionisanja asinhronog *RS* FF-a sa *NILI* (pod a)) i sa *NI* (pod b)) elementima. Leva strana tablica sadrži sve moguće vrednosti signala na ulazima *R* i *S* u kombinaciji sa dva moguća stanja *Q* (0 i 1) u kojima se FF može zateći. Desna strana pokazuje vrednosti izlaznih signala z_1 i z_2, kao i naredno stanje $Q(t+1)$ u koje FF dolazi.

R	S	Q	$Q(t+1)$	z_1	z_2
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	0	1
1	1	0	0	0	0
0	0	1	1	1	0
0	1	1	1	1	0
1	0	1	0	1	0
1	1	1	0	1	0

a)

R	S	Q	$Q(t+1)$	z_1	z_2
0	0	0	1	0	1
0	1	0	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1
0	0	1	1	1	1
0	1	1	0	1	1
1	0	1	1	1	0
1	1	1	1	1	0

b)

Slika 3.4 Kombinaciona tablica funkcije prelaza i funkcija izlaza asinhronog *RS* FF-a a) sa *NILI* elementima i b) sa *NI* elementima

Stanje FF-a Q se smatra *stabilnim* ako za neku kombinaciju signala na ulazima važi $Q(t+1) = Q$. U suprotnom, stanje Q je *nestabilno* za tu ulaznu kombinaciju. FF ostaje u stabilnom stanju sve dok se ne promene signali na ulazu. U nestabilnom stanju, FF se ne zadržava, već odmah otpočinje prelaz u naredno stanje.

Kao što se vidi iz tabele pod a), asinhroni *RS* FF sa *NILI* elementima je u stabilnom stanju za ulaze $RS = 00$, bez obzira na trenutno stanje Q u kome se nalazi. Takođe, stanje $Q = 0$ je stabilno ako se na ulaze dovede neka od kombinacija $RS = 10$ ili $RS = 11$. Stanje $Q = 1$ je stabilno za ulaze $RS = 01$. Ostala stanja ovog FF-a su nestabilna.

Ako se FF nalazi u stabilnom stanju $Q = 0$, a na ulazu je kombinacija signala $RS = 11$, FF se ponaša kao dva nezavisna *NILI* elementa na čijim izlazima su nule. Ako se dogodi da se u tom trenutku na ulaze dovede kombinacija signala $RS = 00$, FF će preći u jedno od stabilnih stanja $Q = 0$ ili $Q = 1$, ali se ne može predvideti u koje. To zavisi od kašnjenja signala u *NILI* kolima i na vezama. Zbog toga, kod asinhronog *RS* FF-a sa *NILI* kolima, ulazna kombinacija $RS = 11$ se smatra zabranjenom. Ova zabrana se može definisati uslovom $R \cdot S = 0$ koji uvek mora ispunjen.

Iz tablice na slici 3.4 b) se vidi da je asinhroni *RS* FF sa *NI* kolima u stabilnom stanju uvek kada je na ulazima $RS = 11$. Stanje $Q = 0$ je stabilno za ulaze $RS = 01$, a $Q = 1$ za ulaze $RS = 10$ i $RS = 00$. Ostala stanja su nestabilna. Sličnom analizom kao kod *RS* FF-a sa *NILI* elementima, može se pokazati da je zabranjeno stanje na ulazima asinhronog *RS* FF-a sa *NI* kolima stanje $RS = 00$, što se može izraziti uslovom $R + S = 1$.

Da bi stanje FF-a sa *NILI* elementima ostalo nepromenjeno za vrednosti ulaznih signala $RS = 00$, tj. da bi važilo $Q(t + 1) = Q$, aktivna vrednost ulaznih signala mora biti 1, a neaktivna 0. Prelazak u naredno stanje je moguć samo ako bar na jedan od ulaza (R ili S) dođe signal sa vrednošću 1. Kod *RS* FF-a sa *NI* kolima je obrnuto, tj. aktivna vrednost signala na ulazima je 0, a neaktivna 1. Na taj način FF ostaje u sadašnjem stanju ako je na ulazima kombinacija $RS = 11$. Prelazak u naredno stanje moguć je samo ako bar na jedan ulaz (R ili S) dođe signal sa vrednošću 0.

Iz tablica na slici 3.4 može se zaključiti da za sve dozvoljene ulaze važi $z_1 = Q$ i $z_2 = \overline{Q}$, pa izlazne funkcije z_1 i z_2 i nisu potrebne (na izlaznim linijama strukturne šeme su stanja Q i $\overline{Q}$).

Primenom uslova $R \cdot S = 0$ na funkciju prelaza *RS* FF-a sa *NILI* elementima, dobija se

$$Q(t+1) = \overline{R} \cdot S + \overline{R} \cdot Q = \overline{R} \cdot S + \overline{R} \cdot Q + R \cdot S = S(\overline{R} + R) + \overline{R} \cdot Q = S + \overline{R} \cdot Q\,.$$

Na osnovu izraza za funkcije prelaza i navedenih uslova zabranjenih stanja za obe realizacije asinhronog *RS* FF-a može se dati funkcionalni opis njegovog rada.

Asinhroni *RS* FF radi tako što:

- ne menja postojeće stanje ako su na oba ulaza *R* i *S* dovedene neaktivne vrednosti signala
- prelazi u stanje $Q = 1$ ako je na ulazu *R* neaktivna, a na ulazu *S* aktivna vrednost signala
- prelazi u stanje $Q = 0$ ako je na ulazu *R* aktivna, a na ulazu *S* neaktivna vrednost signala
- istovremeno dovođenje aktivnih vrednosti signala na oba ulaza je zabranjeno

Iz ovog opisa sledi da aktivan signal na ulazu *S* postavlja stanje FF-a na 1(*set*), a aktivan signal na ulazu *R* postavlja stanje na 0 (*reset*), što je u direktnoj vezi sa nazivom samog FF-a.

Uzimajući u obzir sve dosad navedeno, kombinacione tablice prikazane na slici 3.4 mogu se predstaviti u kompaktnijem obliku u vidu tzv. *tablica prelaza* kao što je prikazano na slici 3.5.

R	*S*	*Q*(*t*+1)
0	0	Q
0	1	1
1	0	0
1	1	?

a)

R	*S*	*Q*(*t*+1)
0	0	?
0	1	0
1	0	1
1	1	Q

b)

Slika 3.5 Tablice prelaza asinhronog *RS* FF-a
a) sa *NILI* elementima i b) sa *NI* elementima

U tablicama prelaza, simbol „?“ označava da je odgovarajuća kombinacija signala na ulazima *R* i *S* zabranjena.

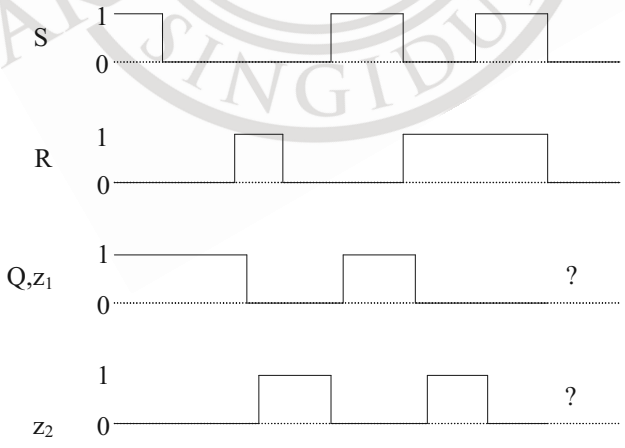
Grafički simboli za obe vrste razmatranih *RS* FF-ova prikazani su na slici 3.6. Kao što se vidi pod b), na ulazima FF-a se nalaze kružići koji označavaju negaciju, tj. da je aktivna vrednost ulaznih signala 0.

a) b)

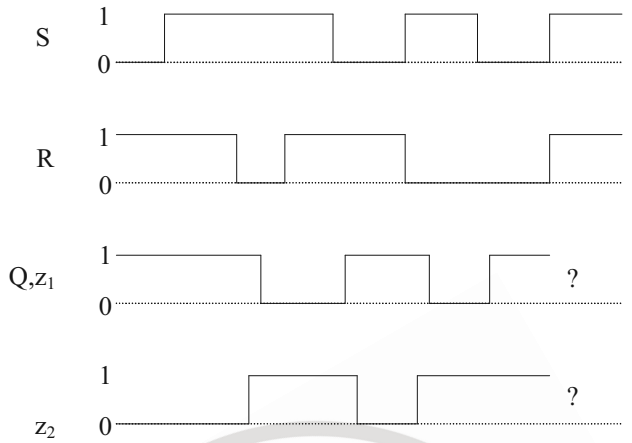
Slika 3.6 Grafički simbol za asinhroni *RS* FF
a) sa *NILI* elementima i b) sa *NI* elementima

Za rad asinhronih FF-ova bitna su dva vremenska parametra: vreme zadržavanja i kašnjenje signala. Vreme zadržavanja t_h (*hold time*) predstavlja minimalno vreme trajanja ulaznih signala nakon promene koje je neophodno da bi FF ispravno radio. Kašnjenje signala t_d (*delay time*) je vremenski interval od trenutka promene signala na ulazu do trenutka kada se promena stanja završi. Vrednosti ova dva paramtera zavise od strukture digitalnog kola i prelaznih procesa u korišćenim komponentama.

Na slici 3.7 dati su vremenski dijagrami asinhronih *RS* FF-a sa *NILI* (pod a)) i *NI* elementima (pod b)). Promene ulaznih signala *R* i *S* su nezavisne, dok promene signala stanja *Q* i izlaznih signala z_1 i z_2 zavise od promena na ulazima. Na dijagramima se mogu zapaziti kašnjenja signala koja odgovaraju kašnjenju u jednom logičkom elementu. Ukoliko oba ulazna signala *R* i *S* prelaze sa aktivne na neaktivnu vrednost, rezultujuće stanje nije definisano, što je na dijagramima označeno upitnikom.



a)



b)

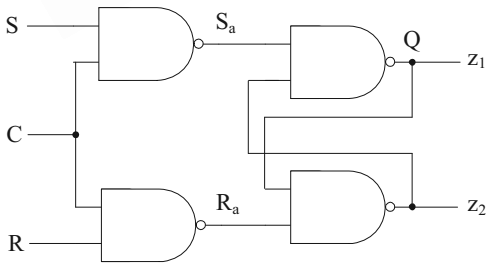
Slika 3.7 Vremenski dijagram signala za asinhroni *RS* FF sa a) *NILI* elementima i b) *NI* elementima

Najvažnija primena asinhronih FF-ova je njihovo intenzivno korišćenje u konstrukciji sinhronih FF-ova. U računarima se retko koriste sekvencijalne mreže sa asinhronim FF-ovima, već uglavnom mreže sa taktovanim FF-ovima.

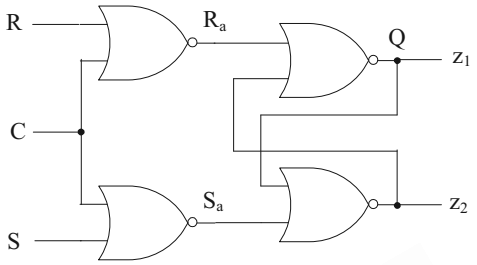
3.2. Sinhroni RS flip-flop

Mnoge aplikacije zahtevaju da se promene stanja *RS* FF-a, u skladu sa signalima *R* i *S* dovedenim na njegove ulaze, dodatno kontrolišu posebnim ulaznim signalom nazvanim *taktni signal* ili samo *takt*. Tako su nastali sinhroni ili taktovani *RS* FF-ovi.

Strukturna šema sinhronog *RS* FF-a realizovanog pomoću logičkih *NI* i logičkih *NILI* elemenata data je na slici 3.8 pod a) i b), respektivno.



a)



Slika 3.8 Sinhroni *RS* flip-flop a) sa *NI* elementima i b) sa *NILI* elementima

Signal takta je na slici označen sa *C* (*clock*). Jedna od mogućih binarnih vrednosti takta (0 ili 1) predstavlja *neaktivnu*, a druga *aktivnu* vrednost takta. Ako se na ulaz *C* dovede neaktivna vrednost takta, FF ostaje u stanju u kome se nalazi bez obrzira na vrednosti signala na ulazima *R* i *S*. Do promene stanja FF-a može da dođe samo ukoliko se na ulaz *C* dovede aktivna vrednost takta. U tom slučaju, od vrednosti signala na ulazima *R* i *S* zavisi da li će do promene stanja FF-a doći. Na ulazu *C* aktivna vrednost takta se zadržava samo onoliko vremena koliko je dovoljno da FF pređe u naredno stanje, a zatim se vraća na neaktivnu vrednost pre nego što bi mogao da započne novi prelaz. Promene ulaznih signala *R* i *S* su dozvoljene samo u vremenskim intervalima kada takt ima neaktivnu vrednost.

Sinhroni *RS* FF prikazan na slici 3.8 a) ima aktivnu vrednost signala takta 1 (to je i aktivna vrednost ulaznih signala *R* i *S*), a *RS* FF na slici 3.8 b) ima takt čija je aktivna vrednost 0, što je i aktivna vrednost njegovih signala na ulazima *R* i *S*.

Na slici 3.8 može se zapaziti da sinhroni *RS* FF u sebi sadrži asinhroni *RS* FF čiji su ulazi R_a i S_a, a izlazi odgovaraju izlazima sinhronog *RS* FF-a. Prema tome, funkcija prelaza sinhronog *RS* FF-a može se dobiti na osnovu ranije izvedene funkcije prelaza asinhronog *RS* FF-a.

Razmotrimo najpre slučaj pod a). Prema strukturnoj šemi, vrednosti signala R_a i S_a date su izrazima $S_a = \overline{S \cdot C}$ i $R_a = \overline{C \cdot R}$. Funkcija prelaza taktovanog *RS* FF-a dobija se kada se signali R_a i S_a sa ulaza asinhronog *RS* FF-a uvrste u njegovu funkciju prelaza. Tako se dobija

$$Q(t+1) = \overline{S_a} + R_a \cdot Q = S \cdot C + \overline{C \cdot R} \cdot Q = S \cdot C + \overline{C} \cdot Q + \overline{R} \cdot Q$$

Ovaj oblik funkcije prelaza naziva se *kompletnom funkcijom prelaza* taktovanog *RS* FF-a sa *NI* kolima. On se obično pojednostavljuje tako što se

posmatra samo za aktivnu vrednost taktnog signala, tj. za $C = 1$. Tako se dobija skraćeni oblik

$$Q(t+1) = S + \overline{R} \cdot Q$$

Upotreba skraćenog oblika funkcije prelaza je prihvatljiva, zato što se za neaktivnu vrednost signala takta funkcija prelaza svodi na $Q(t+1) = Q$.

Na sličan način se prema strukturnoj šemi na slici 3.8 pod b) može izvesti funkcija prelaza taktovanog *RS* FF sa *NILI* elementima. U ovom slučaju, vrednosti signala R_a i S_a mogu se predstaviti izrazima $S_a = \overline{C + S}$ i $R_a = \overline{C + R}$, pa kompletna funkcija prelaza dobija oblik

$$Q(t+1) = S_a + \overline{R_a} \cdot Q = \overline{C + S} + (C + R) \cdot Q = \overline{C} \cdot \overline{S} + C \cdot Q + R \cdot Q$$

Skraćeni oblik ove funkcije prelaza dobija se za aktivnu vrednost taktnog signala, tj. za $C = 0$ i dat je izrazom

$$Q(t+1) = \overline{S} + R \cdot Q$$

I u ovom slučaju može se koristiti skraćeni oblik funkcije prelaza zato što za neaktivnu vrednost taktnog signala takođe važi $Q(t+1) = Q$.

Iz navedenog sledi da su za neaktivnu vrednost taktnog signala oba stanja FF-a, tj. $Q = 0$ i $Q = 1$, stabilna za sve vrednosti ulaznih signala *R* i *S*. Kao i u slučaju asinhronih *RS* FF-ova, kod takovanih *RS* FF-ova je takođe zabranjeno istovremeno dovođenje aktivnih vrednosti na oba ulaza *R* i *S* iz ranije opisanog razloga.

Pošto su funkcije prelaza date istim izrazima kao i kod asinhronih *RS* FF-ova, tablice prelaza razmatranih sinhronih *RS* FF-ova odgovaraju tablicama prikazanim na slici 3.5 pod a) i b).

Na osnovu ovih tablica, način funkcionisanja sinhronih *RS* FF-ova može se opisati na sledeći način:

- za neaktivnu vrednost signala takta *C*, FF ostaje u tekućem stanju bez obzira na vrednosti signala na ulazima *R* i S
- kada se signal takta promeni sa neaktivne na aktivnu vrednost, FF
  - ostaje u tekućem stanju ako je na oba ulaza *R* i *S* neaktivna vrednost signala
  - prelazi u stanje $Q = 1$ ako je na ulazu *R* neaktivna, a na ulazu *S* aktivna vrednost signala

- prelazi u stanje $Q = 0$ ako je na ulazu R aktivna, a na ulazu S neaktivna vrednost signala

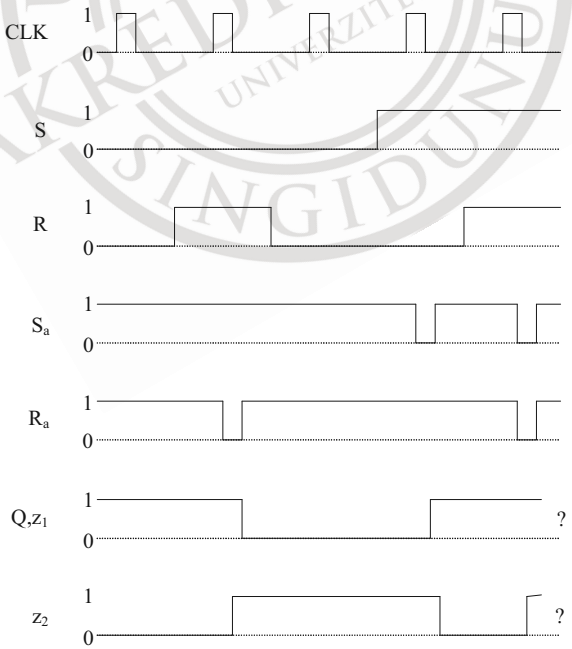
- istovermeno postojanje aktivnih vrednosti signala na ulazima R i S nije dozvoljeno u trenutku promene taktnog signala C sa neaktivne na aktivnu vrednost

Grafički simboli za taktovane *RS* FF-ove kod kojih je aktivna vrednost signala takta 1, odnosno 0 prikazani su na slici 3.9 pod a) i b), respektivno.

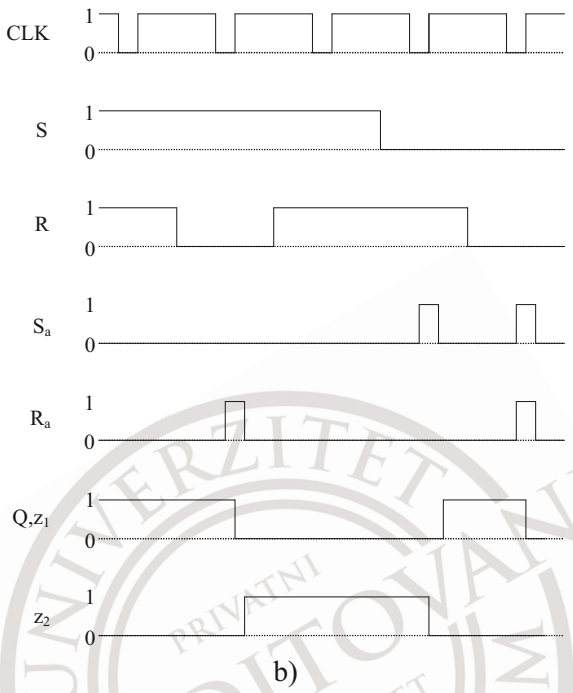
a) b)

Slika 3.9 Grafički simbol za sinhroni RS FF
a) sa *NI* elementima i b) sa *NILI* elementima

Na slici 3.10 dati su vremenski dijagrami sinhronog *RS* FF-a kod kojeg je 1 aktivna vrednost signala takta i ulaznih signala (pod a)) i kog koga je ova aktivna vrednost 0 (pod b)).



a)



Slika 3.10 Vremenski dijagram signala za sinhroni *RS* FF kog koga je aktivna vrednost signala takta i ulaznih signala a) 1 i b) 0

Kao što se vidi sa slike, signal takta *C* je impulsni, dok su svi ostali signali (*R*, *S*, *Q*) potencijalni. To je tako zato što se ne sme dozvoliti da signal takta predugo traje, jer bi u tom slučaju nakon jedne promene stanja *RS* FF-a odmah mogla da započne i sledeća promena, što ne bi vodilo ispravnom radu *RS* FF-a.

Funkcije prelaza se koriste u analizi sekvencijalnih mreža za određivanje načina njihovog funkcionisanja na osnovu date strukturne šeme. Proces sinteze mreže je obrnut. U njemu se na osnovu poznatog načina funkcionisanja mreže određuje njena strukturna šema.

Da bi se sprovela sinteza sekvencijalnih mreža čiji se rad zasniva na asinhronim i taktovanim FF-ovima, za svaki FF potrebno je definisati vrednosti signala koje treba dovesti na njegove ulaze da bi on prešao iz tekućeg stanja Q u naredno stanje $Q(t+1)$. Pri tome, moraju se razmotriti sve četiri moguće kombinacije prelaska $Q \rightarrow Q(t+1)$, tj. $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$ i $1 \rightarrow 1$. Funkcije koje definišu ove vrednosti ulaznih signala nazivaju se *funkcijama pobude FF-ova*. Funkcije pobude se obično predstavljaju kombinacionim tablicama koje se nazivaju *tablicama pobude* FF-ova. U levom delu tablice pobude nalaze se

nezavisno promenljive koje odgovaraju stanjima Q i $Q(t+1)$, a u desnom delu funkcije koje se označavaju isto kao i ulazni signali FF-ova.

Kod taktovanih FF-ova, prelazak iz sadašnjeg u naredno stanje moguć je samo kada se signal takta promeni sa neaktivne na aktivnu vrednost. Tablice pobude sinhronog *RS* FF-a prikazane su na slici 3.11. Pod a) je data tablica pobude za FF kod koga je 1 aktivna vrednost ulaznih signala, a pod b) za FF kod koga je 0 aktivna vrednost ulaznih signala.

Q	$Q(t+1)$	R	S
0	0	*b*	0
0	1	0	1
1	0	1	0
1	1	0	*b*

a)

Q	$Q(t+1)$	R	S
0	0	*b*	1
0	1	1	0
1	0	0	1
1	1	1	*b*

b)

Slika 3.11 Tablice pobude taktovanog *RS* FF-a kod koga je aktivna vrednost ulaznih signala a) 1 i b) 0

Tablice pobude se generišu na osnovu analize ponašanja FF-a. Tako na primer, tablica pod a) dobijena je analizom ponašanja sinhronog *RS* FF-a na sledeći način:

- FF koji se nalazi u stanju $Q = 0$ ostaje u tom stanju ako je na S ulazu signal 0, dok na R ulazu može biti bilo 0, bilo 1, što se označava sa $R = b$; tako je dobijena prva vrsta tabele
- FF prelazi iz stanja $Q = 0$ u stanje $Q(t+1) = 1$ ako je $R = 0$ i $S = 1$, što odgovara drugoj vrsti tabele
- FF prelazi iz stanja $Q = 1$ u stanje $Q(t+1) = 0$ ako je $R = 1$ i $S = 0$, što odgovara trećoj vrsti tabele
- FF koji se nalazi u stanju $Q = 1$ ostaje u tom stanju ako je na R ulazu signal 0, dok na S ulazu može biti bilo 0, bilo 1, što se označava sa $S = b$; tako je dobijena poslednja vrsta tabele

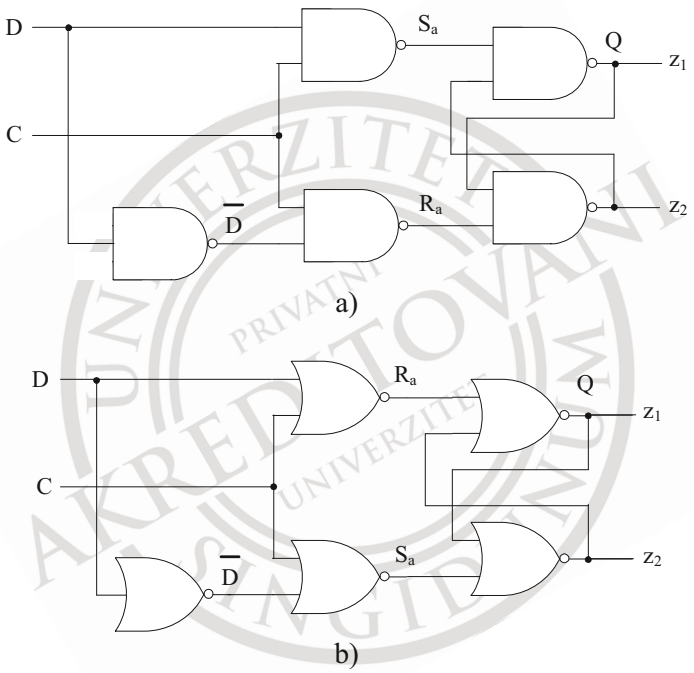
Na sličan način se dobija i tablica pobude prikazana na slici 3.11 b).

Pošto se način funkcionisanja asinhronih i sinhronih *RS* FF-ova opisuju istim funkcijama prelaza, to su i tablice pobude ovih FF-ova iste.

3.3. Sinhroni D flip-flop

Osim za konstrukciju sinhronog *RS* FF-a, asinhroni *RS* FF se takođe koristi i za realizaciju taktovanog *D* FF-a. Naziv *D* FF-a potiče od oznake njegovog ulaznog signala.

Strukturne šeme taktovanih *D* FF-ova realizovnih pomoću *NI* i *NILI* elemenata prikazane su na slici 3.12 pod a) i b), respektivno. NI i *NILI* elementi sa jednim ulazom predstavljaju *NE* element, tj. invertor.



Slika 3.12 Taktovani *D* FF a) sa *NI* elementima i b) sa *NILI* elementima

Na osnovu datih strukturnih šema, funkcije prelaza taktovanih *D* FF-ova mogu se izvesti tako što se najpre uspostave zavisnosti signala *S* i *R* (koji predstavljaju ulaze asinhronog *RS* FF-a) od ulaznih signala *D* FF-a *D* i *C*, a zatim se te zavisnosti uvrste u funkciju prelaza asinhronog *RS* FF-a.

Razmotrimo najpre slučaj pod a). Prema strukturnoj šemi, vrednosti signala S_a i R_a date su izrazima $S_a = \overline{D \cdot C}$ i $R_a = \overline{\overline{D} \cdot C}$. Kada se ove vrednosti uvrste u funkciju prelaza asinhronog *RS* FF-a dobija se

$$Q(t+1) = \overline{S_a} + R_a \cdot Q = D \cdot C + \overline{\overline{D} \cdot C} \cdot Q = D \cdot C + D \cdot Q + \overline{C} \cdot Q$$

Za $C = 0$ važi $Q(t+1) = Q$, a za $C = 1$ dobija se da je $Q(t+1) = D$.

Na sličan način može se analizirati i slučaj taktovanog *D* FF-a sa *NILI* elementima (slika 3.12 pod b)). U ovom slučaju, vrednosti signala R_a i S_a mogu se predstaviti izrazima $S_a = \overline{C + \overline{D}}$ i $R_a = \overline{C + D}$. Funkcija prelaza dobija oblik

$$Q(t+1) = S_a + \overline{R_a} \cdot Q = \overline{C + \overline{D}} + (C + D) \cdot Q = \overline{C} \cdot D + C \cdot Q + D \cdot Q$$

Za $C = 1$ važi $Q(t+1) = Q$, a za $C = 0$ dobija se da je $Q(t+1) = D$.

Prema tome, kod taktovanog *D* FF sa *NI* elementima aktivna vrednost signala takta je 1, a kod *D* FF-a sa *NILI* elementima aktivna vrednost je 0. Za aktivnu vrednost signala takta, funkcija prelaza za oba FF-a se svodi na $Q(t+1) = D$. To znači da su obe vrednosti signala *D* aktivne, tj. definisanje aktivne i neaktivne vrednosti signala *D* nema smisla.

Na osnovu navedenog, način funkcionisanja sinhronih *D* FF-ova može se opisati na sledeći način:

- za neaktivnu vrednost signala takta *C*, FF ostaje u tekućem stanju bez obzira na vrednost signala *D* na ulazu
- kada se signal takta *C* promeni sa neaktivne na aktivnu vrednost, FF prelazi u stanje *D*, tj. ako je *D* = 0, FF prelazi u stanje 0, a ako je *D* = 1, FF prelazi u stanje 1

Tablice prelaza sinhronih *D* FF-ova prikazane su na slici 3.13, a njihovi grafički simboli na slici 3.14.

D	*Q*(*t*+1) = D
0	0
1	1

a)

D	*Q*(*t*+1) = D
0	0
1	1

b)

Slika 3.13 Tablice prelaza taktovanog *D* FF-a kod koga je aktivna vrednost signala takta a) 1 i b) 0

a) b)

Slika 3.14 Grafički simbol za sinhroni *D* FF
a) sa *NI* elementima i b) sa *NILI* elementima

Tablice pobude sinhronog *D* FF-a prikazane su na slici 3.15. Pod a) je data tablica pobude za FF kod koga je 1 aktivna vrednost ulaznog signala, a pod b) za FF kod koga je 0 aktivna vrednost ulaznog signala. Kao što se vidi, tablice pobude su iste za obe vrste *D* FF-ova, jer su im i funkcije prelaza iste.

Q	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

a)

Q	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

b)

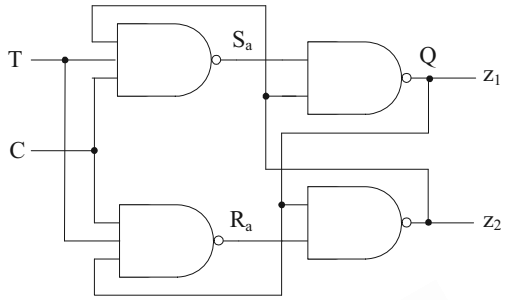
Slika 3.15 Tablice pobude taktovanog *D* FF-a kod koga je
aktivna vrednost ulaznih signala a) 1 i b) 0

Tablica pobude sinhronog *D* FF-a dobijena je na sledeći način:

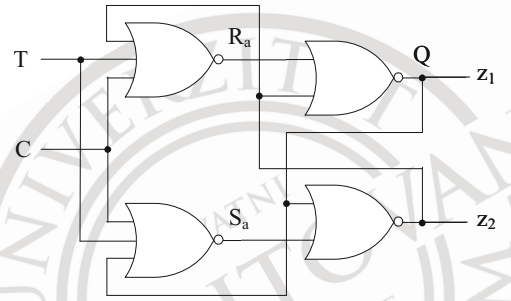
- FF prelazi u stanje $Q(t+1) = 0$ ako je na *D* ulazu signal 0 bez obzira na to u kom se stanju ranije nalazio (prva i treća vrsta tabele)
- FF prelazi u stanje $Q(t+1) = 1$ ako je na *D* ulazu signal 1 bez obzira na to u kom se stanju ranije nalazio (druga i četvrta vrsta tabele)

3.4. Sinhroni T flip-flop

Slično taktovanom *D* FF-u, taktovani *T* FF ima samo dva ulaza: signal takta *C* i ulaz *T*. Strukturne šeme taktovanih *T* FF-ova realizovnih pomoću *NI* i *NILI* elemenata prikazane su na slici 3.16 pod a) i b), respektivno.



a)



b)

Slika 3.16 Taktovani *T* FF a) sa *NI* elementima i b) sa *NILI* elementima

Kod taktovanog *T* FF-a sa *NI* elementima aktivna vrednost signala takta i ulaznog signala *T* su 1, a kod *T* FF-a sa *NILI* elementima njihove aktivne vrednosti su 0.

Iz datih strukturnih šema može se videti da se za realizaciju taktovanih *T* FF-ova takođe koriste asinhroni *RS* FF-ovi. Stoga se i funkcije prelaza taktovanih *T* FF-ova mogu dobiti pomoću funkcija prelaza asinhronih *RS* FF-ova tako što se ulazi S_a i R_a zamene odgovarajućim vrednostima. Pre izvođenja ovih funkcija, pokazaćemo da za oba taktovana *T* FF-a važi $z_2 = \overline{Q}$ za neaktivnu vrednost signala takta, tj. u stabilnom stanju.

Sa slike 3.16 pod a) vidi se da važi $z_2 = \overline{\overline{T \cdot C \cdot \overline{Q}} \cdot Q} = T \cdot C \cdot Q + \overline{Q}$. Za $C = 0$, dobija se da je $z_2 = \overline{Q}$. Na sličan način, sa slike 3.16 pod b) vidi se da važi $z_2 = \overline{\overline{T + C + \overline{Q}} + Q} = (T + C + Q) \cdot \overline{Q}$, pa se za $C = 1$ dobija da je $z_2 = \overline{Q}$.

Prema strukturnoj šemi datoj na slici 3.16 a), vrednosti signala S_a i R_a date su izrazima $S_a = \overline{T \cdot C \cdot \overline{Q}}$ i $R_a = \overline{T \cdot C \cdot Q}$. Kada se ove vrednosti uvrste u funkciju prelaza asinhronog *RS* FF-a dobija se

$$Q(t+1) = \overline{S_a} + R_a \cdot Q = T \cdot C \cdot \overline{Q} + \overline{T \cdot C \cdot Q} \cdot Q = T \cdot C \cdot \overline{Q} + \overline{T} \cdot Q + \overline{C} \cdot Q$$

Za $C = 0$ važi $Q(t+1) = Q$, a za $C = 1$ dobija se da je $Q(t+1) = T \cdot \overline{Q} + \overline{T} \cdot Q$.

Na sličan način se za slučaj taktovanog *T* FF-a sa *NILI* elementima (slika 3.16 pod b)) dobija da su vrednosti signala R_a i S_a date izrazima $R_a = \overline{T + C + \overline{Q}}$ i $S_a = \overline{T + C + Q}$, pa funkcija prelaza dobija oblik

$$Q(t+1) = S_a + \overline{R_a} \cdot Q = \overline{T + C + Q} + (T + C + \overline{Q}) \cdot Q = \overline{T} \cdot \overline{C} \cdot \overline{Q} + T \cdot Q + C \cdot Q$$

Za $C = 1$ važi $Q(t+1) = Q$, a za $C = 0$ dobija se da je $Q(t+1) = \overline{T} \cdot \overline{Q} + T \cdot Q$.

Pošto vrednost izraza $Q(t+1) = T \cdot \overline{Q} + \overline{T} \cdot Q$ za $T = 0$ postaje $Q(t+1) = Q$, a vrednost izraza $Q(t+1) = \overline{T} \cdot \overline{Q} + T \cdot Q$ za $T = 1$ je takođe $Q(t+1) = Q$, sledi da je ranije dato tvrđenje o aktivnim vrednostima signala takta ispravno (kod FF-a sa *NI* elementima aktivna vrednost signala takta je 1, a sa *NILI* elementima je 0).

Na osnovu navedenog, način funkcionisanja sinhronih *T* FF-ova može se opisati na sledeći način:

- za neaktivnu vrednost signala takta *C*, FF ostaje u tekućem stanju bez obzira na vrednost ulaznog signala *T*
- kada se signal takta *C* promeni sa neaktivne na aktivnu vrednost, FF
  - ostaje u tekućem stanju ako je na ulazu *T* neaktivna vrednost signala
  - menja stanje ako je na ulazu *T* aktivna vrednost signala

Tablice prelaza sinhronih *T* FF-ova prikazane su na slici 3.17, a njihovi grafički simboli na slici 3.18.

Tablice pobude sinhronog *T* FF-a prikazane su na slici 3.19. Pod a) je data tablica pobude za FF kod koga je 1 aktivna vrednost ulaznih signala, a pod b) za FF kod koga je 0 aktivna vrednost ulaznih signala.

T	$Q(t+1) = T \cdot \overline{Q} + \overline{T} \cdot Q$
0	Q
1	$\overline{Q}$

a)

T	$Q(t+1) = \overline{T} \cdot \overline{Q} + T \cdot Q$
0	$\overline{Q}$
1	Q

b)

Slika 3.17 Tablice prelaza taktovanog *T* FF-a kod koga je aktivna vrednost signala takta a) 1 i b) 0

a) b)

Slika 3.18 Grafički simbol za sinhroni *T* FF a) sa *NI* elementima i b) sa *NILI* elementima

Q	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

a)

Q	$Q(t+1)$	T
0	0	1
0	1	0
1	0	0
1	1	1

b)

Slika 3.19 Tablice pobude taktovanog *T* FF-a kod koga je aktivna vrednost ulaznih signala a) 1 i b) 0

Tablica pobude sinhronog *T* FF-a prikazana na slici 3.19 a) dobijena je na sledeći način:

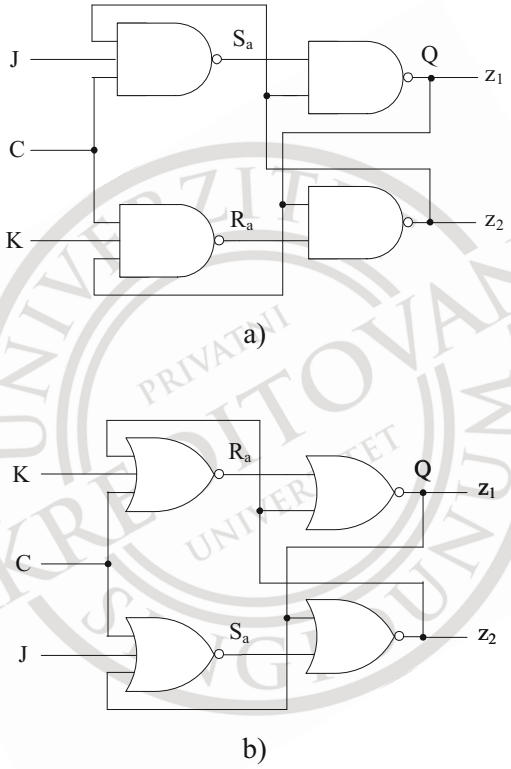
- FF ostaje u stanju u kome se nalazio ako je na *T* ulazu signal 0; tako su dobijene prva i četvrta vrsta tabele
- FF menja stanje (ako je bio u stanju $Q = 0$ prelazi u $Q(t+1) = 1$ i obrnuto, ako je bio u $Q = 1$ prelazi u $Q(t+1) = 0$) ako je na *T* ulazu signal 1; tako su dobijene druga i treća vrsta tabele

Na sličan način dobija se i tablica pobude prikazana na slici 3.19 b).

3.5. Sinhroni JK flip-flop

Sinhroni *JK* FF je dobio naziv po oznakama svojih ulaznih signala *J* i *K*. Osim njih, s obzirom da je FF taktovan, postoji i ulaz *C* na koji se dovodi signal takta.

Strukturne šeme taktovanih *JK* FF-ova realizovnih pomoću *NI* i *NILI* elemenata prikazane su na slici 3.20 pod a) i b), respektivno.



Slika 3.20 Taktovani *JK* FF a) sa *NI* elementima i b) sa *NILI* elementima

Kod taktovanog *JK* FF sa *NI* elementima aktivna vrednost signala takta i ulaznih signala *J* i *K* je 1, a kod *JK* FF-a sa *NILI* elementima, aktivna vrednost navedenih signala je 0.

Pre izvođenja funkcija prelaza, pokazaćemo da za oba taktovana *JK* FF-a važi $z_2 = \overline{Q}$ za neaktivnu vrednost signala takta, tj. u stabilnom stanju. Sa slike 3.20 pod a) vidi se da važi $z_2 = \overline{\overline{K \cdot C \cdot \overline{Q}} \cdot Q} = K \cdot C \cdot Q + \overline{Q}$. Za $C = 0$, dobija se

da je $z_2 = \overline{Q}$. Na sličan način, sa slike 3.20 pod b) vidi se da važi $z_2 = \overline{\overline{J + C + \overline{Q}} + Q} = (J + C + Q) \cdot \overline{Q}$, pa se za $C = 1$ dobija da je $z_2 = \overline{Q}$.

Slično kao kod taktovanih *D* i *T* FF-ova, funkcije prelaza taktovanih *JK* FF-ova mogu se dobiti kada se u funkcijama prelaza asinhronih *RS* FF-ova signali na ulazima S_a i R_a zamene odgovarajućim vrednostima.

Prema strukturnoj šemi datoj na slici 3.20 a), vrednosti signala S_a i R_a date su izrazima $S_a = \overline{J \cdot C \cdot \overline{Q}}$ i $R_a = \overline{K \cdot C \cdot Q}$. Kada se ove vrednosti uvrste u funkciju prelaza asinhronog *RS* FF-a dobija se

$$Q(t+1) = \overline{S_a} + R_a \cdot Q = J \cdot C \cdot \overline{Q} + \overline{K \cdot C \cdot Q} \cdot Q = J \cdot C \cdot \overline{Q} + \overline{K} \cdot Q + \overline{C} \cdot Q$$

Za $C = 0$ važi jednakost $Q(t+1) = Q$, a za $C = 1$ dobija se da je $Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q$.

Na sličan način se za slučaj taktovanog *JK* FF-a sa *NILI* elementima (slika 3.20 pod b)) dobija da su vrednosti signala R_a i S_a date izrazima $R_a = \overline{K + C + \overline{Q}}$ i $S_a = \overline{J + C + Q}$, pa funkcija prelaza dobija oblik

$$Q(t+1) = S_a + \overline{R_a} \cdot Q = \overline{J + C + Q} + (K + C + \overline{Q}) \cdot Q = \overline{J} \cdot \overline{C} \cdot \overline{Q} + K \cdot Q + C \cdot Q$$

Za $C = 1$ važi jednakost $Q(t+1) = Q$, a za $C = 0$ dobija se da je $Q(t+1) = \overline{J} \cdot \overline{Q} + K \cdot Q$.

Vrednost izraza $Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q$ za $JK = 00$ postaje $Q(t+1) = Q$, za $JK = 01$ $Q(t+1) = 0$, za $JK = 10$ $Q(t+1) = 1$, a za $JK = 11$ $Q(t+1) = \overline{Q}$. Takođe, vrednost izraza $Q(t+1) = \overline{J} \cdot \overline{Q} + K \cdot Q$ za $JK = 00$ postaje $Q(t+1) = \overline{Q}$, za $JK = 01$ $Q(t+1) = 1$, za $JK = 10$, a za $JK = 11$ $Q(t+1) = Q$. Iz ovoga sledi da je ranije dato tvrđenje o aktivnim vrednostima signala takta ispravno (kod FF-a sa *NI* elementima aktivna vrednost signala takta i ulaznih signala je 1, dok je kod FF-a sa *NILI* elementima 0).

Na osnovu navedenog, način funkcionisanja sinhronih *JK* FF-ova može se opisati na sledeći način:

- za neaktivnu vrednost signala takta *C*, FF ostaje u tekućem stanju bez obzira na vrednosti ulaznih signala *J* i *K*
- kada se signal takta *C* promeni sa neaktivne na aktivnu vrednost, FF

- ostaje u tekućem stanju ako su na oba ulaza *J* i *K* neaktivne vrednost signala
- prelazi u stanje 0 ako je na ulazu *J* neaktivna, a na ulazu *K* aktivna vrednost signala
- prelazi u stanje 1 ako je na ulazu *J* aktivna, a na ulazu *K* neaktivna vrednost signala
- menja stanje ako su na oba ulaza *J* i *K* aktivne vrednosti signala

Kao što se vidi, ulazi *J* i *K* kod taktovanog *JK* FF-a imaju istu funkciju kao ulazi *R* i *S* kod taktovanog *RS* FF-a. Jedina razlika je u tome što je kod taktovanog *JK* FF-a dopušteno da u trenutku promene signala takta sa neaktivne na aktivnu vrednost, oba ulaza *J* i *K* imaju aktivne vrednosti (tada FF menja stanje).

Tablice prelaza sinhronih *JK* FF-ova prikazane su na slici 3.21, a njihovi grafički simboli na slici 3.22.

J	K	$Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q$
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

a)

J	K	$Q(t+1) = \overline{J} \cdot \overline{Q} + K \cdot Q$
0	0	$\overline{Q}$
0	1	1
1	0	0
1	1	Q

b)

Slika 3.21 Tablice prelaza taktovanog *JK* FF-a kod koga je aktivna vrednost signala takta a) 1 i b) 0

a) b)

Slika 3.22 Grafički simbol za sinhroni *JK* FF
a) sa *NI* elementima i b) sa *NILI* elementima

Tablice pobude sinhronog *JK* FF-a prikazane su na slici 3.23. Pod a) je data tablica pobude za FF kod koga je 1 aktivna vrednost ulaznih signala, a pod b) za FF kod koga je 0 aktivna vrednost ulaznih signala.

Q	$Q(t+1)$	J	K
0	0	0	b
0	1	1	b
1	0	b	1
1	1	b	0

a)

Q	$Q(t+1)$	J	K
0	0	1	b
0	1	0	b
1	0	b	0
1	1	b	1

b)

Slika 3.23 Tablice pobude taktovanog *JK* FF-a kod koga je aktivna vrednost ulaznih signala a) 1 i b) 0

Tablica pobude data na slici 3.23 a) dobijena je analizom ponašanja sinhronog *JK* FF-a na sledeći način:

- FF koji se nalazi u stanju $Q = 0$ ostaje u tom stanju ako je na J ulazu signal 0, dok na K ulazu može biti bilo 0, bilo 1, što se označava sa $K = b$; tako je dobijena prva vrsta tabele
- FF prelazi iz stanja $Q = 0$ u stanje $Q(t+1) = 1$ ako je $J = 1$, dok na K ulazu može biti bilo 0, bilo 1, pa je $K = b$, što odgovara drugoj vrsti tabele
- FF prelazi iz stanja $Q = 1$ u stanje $Q(t+1) = 0$ ako je $K = 1$, dok na J ulazu može biti bilo 0, bilo 1, pa je $J = b$, što odgovara trećoj vrsti tabele
- FF koji se nalazi u stanju $Q = 1$ ostaje u tom stanju ako je na K ulazu signal 0, dok na J ulazu može biti bilo 0, bilo 1, što se označava sa $J = b$; tako je dobijena poslednja vrsta tabele

Na sličan način se dobija i tablica pobude prikazana na slici 3.23 b).

Vežbanja

1. Koje vrste signala postoje u digitalnim kolima?

2. Šta su to memorijski elementi i u čemu se oni razlikuju od logičkih elemenata?

3. Kakva je razlika između asinhronih i taktovanih flip-flop-ova?

4. Dati strukturne šeme asinhronih *RS* flip-flop-ova sa *NI* i sa *NILI* elementima i objasniti principe njihovog funkcionisanja.

5. Izvesti funkciju prelaza i funkcije izlaza za asinhroni *RS* FF sa:

   a) *NI* elementima
   b) *NILI* elementima

6. Nacrtati grafičke simbole za asinhrone *RS* FF-ove sa *NI* i *NILI* elementima. Dati tablice prelaza za navedene vrste *RS* FF-ova i objasniti ih.

7. Dati strukturne šeme taktovanih *RS* flip-flop-ova sa *NI* i sa *NILI* elementima i objasniti kako oni rade.

8. Izvesti funkciju prelaza za sinhroni *RS* FF sa:

   a) *NI* elementima
   b) *NILI* elementima

   Na osnovu funkcija prelaza, generisati tablice prelaza i objasniti ih.

9. Nacrtati grafičke simbole za taktovane *RS* FF-ove sa *NI* i *NILI* elementima.

10. Šta predstavljaju tablice pobude? Formirati tablice pobude koje odgovaraju taktovanim *RS* FF-ovima sa *NI* i *NILI* elementima i objasniti ih.

11. Dati strukturne šeme taktovanih *D* flip-flop-ova sa *NI* i sa *NILI* elementima i objasniti način njihovog funkcionisanja.

12. Izvesti funkcije prelaza za sinhrone *D* FF-ove sa *NI* i *NILI* elementima.

13. Na osnovu funkcija prelaza taktovanih *D* FF-ova, generisati odgovarajuće tablice prelaza i objasniti ih.

14. Nacrtati grafičke simbole za taktovane *D* FF-ove sa *NI* i *NILI* elementima, za svaku vrstu formirati tablicu pobude i objasniti je.

15. Dati strukturne šeme sinhronih *T* flip-flop-ova sa *NI* i sa *NILI* elementima i objasniti principe njihovog funkcionisanja.

16. Izvesti funkciju prelaza za sinhrone *T* FF-ove sa *NI* i *NILI* elementima.

17. Nacrtati grafičke simbole za taktovane *T* FF-ove sa *NI* i *NILI* elementima. Za svaku vrstu dati ogovarajuću tablicu prelaza i objasniti je.

18. Generisati tablice pobude za taktovane *T* FF-ove sa *NI* i *NILI* elementima i objasniti ih.

19. Dati strukturne šeme taktovanih *JK* flip-flop-ova sa *NI* i sa *NILI* elementima i objasniti kako oni rade.

20. Izvesti funkciju prelaza za sinhroni *JK* FF sa:
    a) *NI* elementima
    b) *NILI* elementima

    Na osnovu funkcija prelaza, generisati tablice prelaza i objasniti ih.

21. Nacrtati grafičke simbole za taktovane *JK* FF-ove sa *NI* i *NILI* elementima.

22. Formirati tablice pobude za sinhrone *JK* FF-ove sa *NI* i *NILI* elementima i objasniti ih.

4 Logičke funkcije

Već je rečeno da se rad računarskih sistema u suštini zasniva na primeni logičkih operacija nad binarnim vrednostima. Kombinovanjem različitih logičkih kola mogu se dobiti vrlo složene logičke strukture koje obezbeđuju potrebnu funkcionalnost u računarskom sistemu. Te strukture se opisuju logičkim funkcijama. S obzirom da se binarna logika (0 ili 1, ima ili nema signala) jednostavno može predstaviti prekidačkim elementom, logičke funkcije se često nazivaju i prekidačkim funkcijama.

Između aritmetičkih i logičkih funkcija postoje sličnosti, ali i značajne razlike. Logičke funkcije su jednostavnije od aritmetičkih jer operišu nad binarnim skupom vrednosti. Međutim, način razmišljanja karakterističan za logičke funkcije se često teško usvaja zbog navike stečene u radu sa aritmetičkim funkcijama u ranijem periodu.

Logička funkcija Y se, slično aritmetičkoj, može definisati nad proizvoljnim brojem promenljivih A, B, C, ... na sledeći način:

$$Y = f(A, B, C, ...)$$

Osnovna razlika u odnosu na aritmetičku funkciju je u tome što vrednost logičke funkcije Y mora pripadati skupu $\{0,1\}$. Takođe, i vrednosti promenljivih u logičkim funkcijama moraju pripadati istom skupu, tj. mogu biti samo 0 ili 1. Pošto se vrednosti 0 i 1 nazivaju logičkim, to se i promenljive u logičkim funkcijama nazivaju logičkim promenljivama.

Logičke promenljive u logičkoj funkciji su međusobno povezane logičkim operacijama (*I*, *ILI*, *NE*,...). Vrednost logičke funkcije za zadate vrednosti logičkih promenljivih dobija se kao rezultat izvršavanja logičkih operacija korišćenih u logičkoj funkciji.

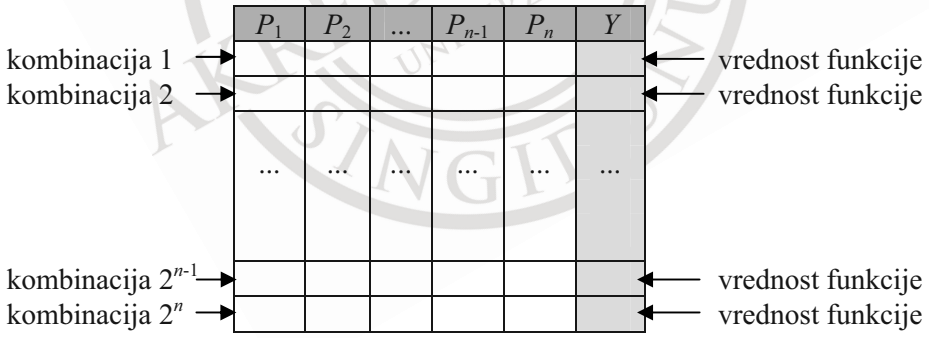
4.1. Predstavljanje logičkih funkcija

Logičke funkcije se mogu predstavljati na različite načine. U nastavku će biti opisana tri često primenjivana načina: pomoću kombinacione tablice, u vidu algebarskog izraza i pomoću Karnoovih karti. Zatim će biti pokazano kako se neka logička funkcija predstavljena na jedan od opisanih načina može prikazati na neki drugi način.

4.1.1 Kombinacione tablice

Pošto logičke funkcije zavise od konačnog broja logičkih promenljivih (n), pri čemu svaka promenljiva može imati samo dve vrednosti, 0 ili 1, to je broj mogućih različitih kombinacija promenljivih 2^n. Iz ovoga sledi da logičke funkcije imaju konačnu oblast definisanosti. Zahvaljujući tome, one se mogu predstaviti pomoću tablica koje se nazivaju kombinacionim tablicama ili tablicama istinitosti.

Kombinaciona tablica predstavlja tabelu u kojoj su date vrednosti logičke funkcije za sve moguće kombinacije vrednosti logičkih promenljivih koje se u njoj pojavljuju. Na slici 4.1 data je kombinaciona tablica kojom je predstavljena logička funkcija Y koja ima n logičkih promenljivih P_1, P_2, ..., P_n.



Slika 4.1 Izgled kombinacione tablice za logičku funkciju n promenljivih

Leva strana tabele 4.1 sadrži n kolona, za svaku promenljivu po jednu, i 2^n vrsta. Jedna vrsta odgovara jednoj kombinaciji vrednosti iz skupa {0,1} svih logičkih promenljivih koje se javljaju u funkciji. To znači da jedna kolona sadrži logičke vrednosti koje njoj odgovarajuća logička promenljiva ima u svim mogućim kombinacijama. Desna strana kombinacione tablice ima jednu kolonu koja odgovara samoj funkciji Y, i 2^n vrsta. Svaka vrsta sadrži vrednost koju logička

funkcija ima za kombinaciju vrednosti promenljivih navedenu u istoj vrsti u levom delu tabele.

Pri popunjavanju svih mogućih kombinacija u tabeli, lako može da dođe do grešaka (da se neke kombinacije ponove, ili da neke nedostaju) ukoliko se one popunjavaju nasumice. Stoga je dobro koristiti neki sistematičan pristup. Na primer, može se najpre popuniti poslednja kolona u levom delu tabele naizmeničnim upisivanjem vrednosti 0 i 1. Zatim se popunjava susedna kolona (levo od već popunjene) naizmeničnim upisom po dve vrednosti 0 i 1, pa sledeća susedna kolona sa po 4 vrednosti 0 i 1, pa sledeća po 8 vrednosti 0 i 1, i tako dalje (uvek po 2^i vrednosti 0 i 1) sve dok se ne popuni prva kolona u levom delu tabele. Ovakvo popunjavanje garantuje da su sve moguće kombinacije za zadati broj promenljivih unete.

Primer dobro popunjene kombinacione tablice koja predstavlja logičku funkciju četiri promenljive *A*, *B*, *C* i *D* dat je na slici 4.2.

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Slika 4.2 Izgled kombinacione tablice za logičku funkciju 4 promenljive

Može se zapaziti da su kombinacije unete u skladu sa postupkom izloženim u prethodnom pasusu. Značenje bilo koje vrste u tabeli slično je sledećem tumačenju koje se odnosi, na primer, na drugu vrstu: ako promenljive *A*, *B* i *C* imaju vrednost 0, a promenljiva *D* vrednost 1, vrednost logičke funkcije je 1.

Kombinacione tablice predstavljaju vrlo pregledan i jednostavan način za zadavanje logičkih funkcija. Međutim, s obzirom da imaju 2^n vrsta, predstavljanje logičkih funkcija na ovaj način postaje nepogodno već i pri malom broju promenljivih, tj. malim vrednostima n. Na primer, ako je $n = 5$, broj vrsta u tabeli je $2^5 = 32$, a ako je $n = 6$ broj vrsta raste na $2^6 = 64$. Ovakve dimenzije čine tabelu nepreglednom i nepraktičnom za rad.

Predstavljanje logičke funkcije pomoću tablice istinitosti biće prikazano na dva primera.

Prvi primer se odnosi na predstavljanje većinske logike pomoću kombinacione tablice. Pretpostavimo da tri glasača *A*, *B* i *C* glasaju za neki predlog tako što zaokružuju *za* (ako podržavaju predlog) ili *protiv* (ako ne podržavaju predlog). Označimo glas *za* predlog logičkom vrednošću 1, a glas *protiv* predloga logičkom vrednošću 0. Predlog može biti *usvojen* ako su dva ili više glasača glasala *za*, u suprotnom je *odbijen*. Označimo *usvojen* predlog logičkom vrednošću 1, a *odbijen* logičkom vrednošću 0. Zadatak je da se funkcija *Y* koja pokazuje da li je predlog usvojen ili ne predstavi tablicom istinitosti.

U rešavanju navedenog problema, najpre treba uočiti da funkcija *Y* zavisi od tri promenljive koje predstavljaju glasače *A*, *B* i *C*, pošto njihovi glasovi direktno utiču na vrednost funkcije, tj. da li je predlog usvojen ili ne. Dakle, leva strana kombinacione tablice ima 3 kolone i $2^3 = 8$ vrsta. Desna strana tablice ima jednu kolonu i 8 vrsta. Sada sledi popunjavanje svih mogućih kombinacija po vrstama leve strane tabele. Kombinacije se formiraju tako što, ukoliko je glasač glasao *za*, u polje njemu odgovarajuće kolone upisuje se vrednost 1, a ako je glasao *protiv*, upisuje se vrednost 0. Nakon što se popuni leva strana tabele, pristupa se određivanju vrednosti logičke funkcije *Y* za svaku od kombinacija i njenom upisivanju u odgovarajuće polje u desnom delu tablice. Vrednost funkcije je 1 ako je predlog *usvojen*, tj. ako je u odgovarajućoj kombinaciji u levom delu tabele više jedinica nego nula (to znači da su bar dva glasača glasala za predlog). U suprotnom, vrednost funkcije je 0. Nakon unosa vrednosti funkcije, dobija se tražena kombinaciona tablica koja je prikazana na slici 4.3.

Drugi primer ima za cilj predstavljanje logičke funkcije *Y* koja generiše signal za pokretanje lifta (označen logičkom vrednošću 1) u vidu kombinacione tablice. Pretpostavimo da se način funkcionisanja lifta može opisati pomoću tri logičke promenljive:

A - ima vrednost 1 ako su spoljna vrata lifta zatvorena, a 0 ako su otvorena

B - ima vrednost 1 ako su unutrašnja vrata lifta zatvorena, a 0 ako su otvorena

C - ima vrednost 1 ako se u liftu neko nalazi, a 0 ako je lift prazan

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Slika 4.3 Kombinaciona tablica koja odgovara većinskoj logici

Signal za pokretanje lifta treba generisati samo u dve situacije: ako su spoljna i unutrašnja vrata zatvorena i u liftu ima nekoga, ili ako su spoljna vrata zatvorena i lift je prazan.

Slično kao i u prethodnom primeru, najpre treba uočiti da funkcija *Y* zavisi od tri promenljive *A*, *B* i *C*, pošto njihove vrednosti direktno utiču na to da li će se lift pokrenuti ili ne. Dakle, leva strana kombinacione tablice ima 3 kolone i $2^3 = 8$ vrsta. Desna strana tablice ima jednu kolonu i 8 vrsta. Sledi popunjavanje svih mogućih kombinacija po vrstama leve strane tabele. Kombinacije se formiraju tako što se za svaku promenljivu, u polje njoj odgovarajuće kolone upisuje vrednost 1 ili 0 zavisno od stanja koje ta promenljiva ima u toj kombinaciji. Na primer, ako posmatrana kombinacija podrazumeva da su spoljna vrata lifta zatvorena, a unutrašnja otvorena i da u liftu ima nekog, promenljiva *A* će imati vrednost 1, promenljiva *B* vrednost 0 i promenljiva *C* vrednost 1. Nakon što se popuni leva strana tabele, pristupa se određivanju vrednosti logičke funkcije *Y* za svaku od kombinacija i njenom upisivanju u odgovarajuće polje u desnom delu tablice. Vrednost funkcije je 1 ako kombinacija opisuje jednu od dve navedene situacije za pokretanje lifta. U suprotnom, vrednost funkcije je 0. Tražena kombinaciona tablica prikazana je na slici 4.4.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Slika 4.4 Kombinaciona tablica koja opisuje funkcionisanje lifta

4.1.2 Algebarski oblik funkcije

Kao što je ranije rečeno, iako se svaka prekidačka funkcija može predstaviti kombinacionom tablicom, za funkcije većeg broja promenjivih to nije pogodna forma predstavljanja. U tom slučaju je bolje koristiti druge načine predstavljanja, kao na primer algebarski prikaz logičke funkcije.

U algebarskom prikazu, logička funkcija se predstavlja algebarskim izrazom koga čine logičke promenljive međusobno povezane logičkim operacijama (*I*, *ILI*, *NE*, ...). Postoje različiti načini algebarskog predstavljanja logičke funkcije. Jedan od često korišćenih načina je primena tzv. savršenih normalnih formi. Savršene normalne forme se pojavljaju u dva oblika, kao:

- savršena disjunktivna normalna forma (*SDNF*)
- savršena konjuktivna normalna forma (*SKNF*)

Neka je data logička funkcija Y koja zavisi od n logičkih promenljivih označenih sa A_1, A_2, ..., A_n. Bilo koja logička promenljiva, osim svoje originalne vrednosti (na primer A) ima i svoju negiranu vrednost ($\bar{A}$). Označimo sa $\tilde{A}$ promenljivu A ili njenu negiranu vrednost $\bar{A}$, tj. $\tilde{A} = A$ ili $\tilde{A} = \bar{A}$. Uvedimo sada dva nova pojma: potpuni proizvod i potpunu sumu.

Potpuni proizvod predstavlja logički proizvod $\tilde{A}_1\tilde{A}_2 \ldots \tilde{A}_n$. Dakle, to je proizvod u kome se pojavljuju sve promenljive od kojih zavisi logička funkcija (zbog toga se proizvod i zove potpunim), s tim što neke od promenjivih imaju svoju originalnu, a neke negiranu vrednost. Potpuni proizvod ima vrednost 1 samo za jednu kombinaciju vrednosti promenljivih, dok za sve ostale kombinacije ima vrednost 0.

Na sličan način, potpuna suma se definiše kao logički zbir $\tilde{A}_1+\tilde{A}_2+ \ldots +\tilde{A}_n$. U potpunoj sumi se, takođe, pojavljuju sve promenljive funkcije (što opravdava naziv potpuna) bilo u svojoj originalnoj ili negiranoj vrednosti. Potpuna suma ima vrednost 0 samo za jednu kombinaciju vrednosti promenljivih, dok za sve ostale kombinacije ima vrednost 1.

Primenom uvedenih pojmova mogu se definisati *SDNF* i *SKNF*. *SDNF* predstavlja logički zbir potpunih proizvoda, a *SKNF* logički proizvod potpunih suma.

Postupak predstavljanja logičke funkcije u algebarskom obliku korišćenjem *SDNF* opisan je teoremom čije će tvrđenje biti dato bez dokaza:

Teorema 1: Svaka logička funkcija $Y = f(A_1, A_2, ..., A_n)$, izuzev konstante nula, može se na jedinstven način napisati u obliku

$$Y = P_1 + P_2 + ... + P_m \qquad (m \leq 2^n)$$

gde su P_1, P_2, ..., P_m potpuni proizvodi koji odgovaraju kombinacijama vrednosti promenljivih za koje funkcija Y ima vrednost 1, tj. kao *SNDF*.

Na sličan način, logička funkcija se u algebarskom obliku može predstaviti i korišćenjem *SKNF* u skladu sa sledećom teoremom:

Teorema 2: Svaka logička funkcija $Y = f(A_1, A_2, ..., A_n)$, izuzev konstante jedinica, može se na jedinstven način napisati u obliku

$$Y = S_1 S_2 ... S_m \qquad (m \leq 2^n)$$

gde su S_1, S_2, ..., S_m potpune sume koje odgovaraju kombinacijama vrednosti promenljivih za koje funkcija Y ima vrednost 0, tj. kao *SKNF*.

Tvrđenja ovih teorema biće ilustrovana na jednom primeru. Neka je data logička funkcija Y koja zavisi od tri logičke promenljive, tj. $Y = f(A_1, A_2, A_3)$. Neka funkcija ima vrednost 1 za sledeće kombinacije vrednosti promenljivih A_1, A_2 i A_3: 010, 100, 101 i 111. Za sve ostale kombinacije vrednosti promenljivih, funkcija ima vrednost 0.

Prema prvoj teoremi, za kombinacije na kojima funkcija ima vrednost 1 mogu se formirati sledeći potpuni proizvodi:

$$P_1 = \bar{A}_1 A_2 \bar{A}_3, \quad P_2 = A_1 \bar{A}_2 \bar{A}_3, \quad P_3 = A_1 \bar{A}_2 A_3 \quad \text{i} \quad P_4 = A_1 A_2 A_3.$$

Potpuni proizvodi su formirani tako što, ukoliko je vrednost promenljive u kombinaciji 1, promenljiva ulazi u proizvod sa svojom originalnom vrednošću, a ako je vrednost 0, u proizvod se uključuje negirana vrednost promenljive. Sada se, po tvrđenju teoreme, logička funkcija Y može predstaviti u algebarskom obliku u vidu *SDNF*, ili tzv. sume proizvoda kao:

$$Y = \bar{A}_1 A_2 \bar{A}_3 + A_1 \bar{A}_2 \bar{A}_3 + A_1 \bar{A}_2 A_3 + A_1 A_2 A_3$$

Dakle, logička funkcija se predstavlja u vidu sume proizvoda tako što se operacijom logičkog sabiranja *ILI* povežu svi potpuni proizvodi koji odgovaraju kombinacijama vrednosti promenljivih za koje funkcija ima vrednost 1. Iz datog algebarskog izraza sledi da ukoliko logičke promenljive dobiju vrednosti koje odgovaraju jednoj od gore navedenih kombinacija, taj proizvod u zbiru postaje 1 (ostali proizvodi su 0), pa vrednost funkcije postaje 1 (rezultat logičkog sabiranja je 1 ako je bar jedan sabirak jednak 1).

Ista logička funkcija može se predstaviti i u vidu *SKNF*, ili tzv. proizvoda suma. Prema drugoj teoremi, za kombinacije na kojima funkcija ima vrednost 0, a to su kombinacije 000, 001, 011 i 110, mogu se formirati sledeće potpune sume:

$$S_1 = A_1+A_2+A_3, \quad S_2 = A_1+A_2+\bar{A}_3, \quad S_3 = A_1+\bar{A}_2+\bar{A}_3 \quad \text{i} \quad S_4 = \bar{A}_1+\bar{A}_2+A_3.$$

Potpune sume su formirane tako što, ukoliko je vrednost promenljive u kombinaciji 0, promenljiva ulazi u sumu sa svojom originalnom vrednošću, a ako je vrednost 1, u sumu se uključuje negirana vrednost promenljive. Sada se, po tvrđenju teoreme, logička funkcija *Y* može predstaviti u algebarskom obliku u vidu *SKNF* ili proizvoda suma kao:

$$Y = (A_1+A_2+A_3)(A_1+A_2+\bar{A}_3)(A_1+\bar{A}_2+\bar{A}_3)(\bar{A}_1+\bar{A}_2+A_3)$$

Dakle, logička funkcija se predstavlja u vidu proizvoda suma tako što se operacijom logičkog množenja *I* povežu sve potpune sume koje odgovaraju kombinacijama vrednosti promenljivih za koje funkcija ima vrednost 0. Iz navedenog algebarskog izraza sledi da ukoliko logičke promenljive dobiju vrednosti koje odgovaraju jednoj od kombinacija 000, 001, 011 ili 110, ta suma u proizvodu postaje 0, pa vrednost funkcije postaje 0 (rezultat logičkog množenja je 0 ako je bar jedan činilac jednak 0).

4.1.3 Karnoove karte

Osim kombinacionih tablica, za predstavljanje logičkih funkcija često se koristi još jedan tabelarni prikaz, a to je Karnoova (*Karnaugh*) karta. Ova karta, kao i kombinaciona tablica, predstavlja tabelu u kojoj su date vrednosti logičke funkcije za sve moguće kombinacije vrednosti logičkih promenljivih koje se u njoj pojavljuju. Razlika između ovih tabela je u njihovoj organizaciji.

Karnoova karta koja prikazuje logičku funkciju *Y* sa *n* logičkih promenljivih ima ukupno 2^n polja (koliko ima i mogućih kombinacija vrednosti promenljivih). Ukoliko je *n* paran broj, tabela ima $2^{n/2}$ vrsta i $2^{n/2}$ kolona, a ako je *n* neparan broj, $2^{(n-1)/2}$ vrsta i $2^{(n+1)/2}$ kolona (ili obrnuto, $2^{(n-1)/2}$ kolona i $2^{(n+1)/2}$ vrsta). Ovakvim rasporedom vrsta i kolona postiže se da Karnoova karta ima izgled što sličniji kvadratu, što doprinosi njenoj preglednosti.

Svakom polju u Karnoovoj karti odgovara jedna kombinacija vrednosti logičkih promenljivih funkcije. O kojoj kombinaciji je reč, određeno je binarnim oznakama vrsta i kolona. Naime, svakoj vrsti i koloni pridružena je binarna oznaka koja ukazuje na vrednosti koje odgovarajuća logička promenljiva ima u toj vrsti ili koloni. Da bi se pokazao način formiranja oznaka vrsta i kolona, najpre će biti uvedene neke pretpostavke.

Pretpostavimo da je skup svih promenljivih funkcije podeljen u dva podskupa. Broj elemenata u podskupovima je uslovljen brojem vrsta, odnosno kolona u Karnoovoj karti. Tako, ukoliko je n paran broj, podskupovi imaju po $n/2$ elemenata, a ako je n neparan broj, jedan podskup ima $(n-1)/2$ elemenata, a drugi $(n+1)/2$ elemenata. Pretpostavimo sada da je jedan podskup pridružen vrstama, a drugi kolonama (u slučaju parnog n, potpuno je svejedno koji podskup se pridružuje vrstama, a koji kolonama, dok u slučaju neparnog n, broj vrsta i kolona mora biti usklađen sa brojem elemenata u podskupu). Izbor konkretnih logičkih promenljivih koje će se naći u podskupovima može biti napravljen na različite načine i za svaki učinjeni izbor, Karnoova karta je ispravno definisana. Na osnovu uvedenih pretpostavki, oznake vrsta i kolona se formiraju kao sve moguće kombinacije vrednosti promenljivih koje se pojavljuju u podskupovima pridruženim vrstama, odnosno kolonama.

Pri raspoređivanju kombinacija po poljima Karnoove karte mora se poštovati pravilo da se kombinacije koje odgovaraju susednim poljima u Karnoovoj karti razlikuju samo u jednoj vrednosti (cifri). To se postiže tako što se poštuje pravilo da se i susedne oznake vrsta, odnosno kolona, takođe mogu razlikovati samo u jednoj vrednosti (cifri). Karnoova karta se popunjava tako što se u svako polje unosi vrednost koju funkcija ima za kombinaciju promenljivih koja odgovara tom polju.

U nastavku će biti opisan postupak generisanja Karnoove karte za funkciju n promenljivih po koracima:

1. Od skupa logičkih promenljivih formirati dva podskupa promenljivih V_1 i V_2 sa približno istim brojem članova (detaljna analiza broja elemenata u podkupovima data je ranije). Koje će promenljive biti u jednom, a koje u drugom podskupu, nije od značaja. Neka podskup V_1 ima $n1$ elemenata, a podskup V_2 $n2$ elemenata (mora da važi $n1 + n2 = n$).
2. Nacrtati tablicu sa 2^{n1} vrsta i 2^{n2} kolona.
3. U gornji levi ugao karte, iznad vrsta upisati nazive svih logičkih promenljivih koje pripadaju podskupu V_1, a levo od kolona nazive svih promenljivih koje pripadaju podskupu V_2.
4. Sve moguće kombinacije vrednosti promenljivih iz podskupa V_1 upisati kao oznake vrsta. Voditi računa da susedne oznake mogu da se razlikuju samo u jednoj binarnoj cifri.
5. Sve moguće kombinacije vrednosti promenljivih iz podskupa V_2 upisati kao oznake kolona. Takođe, mora se voditi računa da susedne oznake mogu da se razlikuju samo u jednoj binarnoj cifri.

6. U svako polje karte upisati binarnu vrednost koju funkcija ima za kombinaciju vrednosti promenljivih definisanu oznakom vrste i oznakom kolone.

Opisani postupak će biti primenjen za predstavljanje logičke funkcije Y koja zavisi od četiri promenljive A, B, C i D pomoću Karnoove karte. Neka funkcija Y ima vrednost 1 samo ako su vrednosti svih promenljivih međusobno jednake.

1. Od skupa logičkih promenljivih $\{A, B, C, D\}$ koji ima $n = 4$ elementa formiraju se dva podskupa V_1 i V_2 od po $n/2=2$ elementa ($n1=2$ i $n2=2$). Neka je $V_1 = \{A, B\}$, a $V_2 = \{C, D\}$ (raspored promenljivh po podskupovima je mogao da bude i drugačiji).
2. Sledi crtanje tablice sa $2^{n1} = 2^2 = 4$ vrste i $2^{n2} = 2^2 = 4$ kolone.
3. U gornji levi ugao karte, iznad vrsta upisuju se promenljive iz podskupa V_1, tj. AB, a levo od kolona promenljive iz podskupa V_2, tj. CD.
4. Kao oznake vrsta unose se sve kombinacije vrednosti promenljivih A i B (ima ih $2^2 = 4$), pri čemu se vodi računa da se susedne oznake razlikuju samo u jednoj binarnoj cifri. Moguć redosled oznaka vrsta je: 00, 01, 11, 10.
5. Kao oznake kolona unose se sve kombinacije vrednosti promenljivih C i D (ima ih $2^2 = 4$), pri čemu se, takođe, vodi računa da se susedne oznake razlikuju samo u jednoj binarnoj cifri. Moguć redosled oznaka kolona je: 00, 01, 11, 10.
6. Sledi popunjavanje karte vrednostima funkcije. Pošto funkcija Y ima vrednost 1 samo ako promenljive A, B, C i D imaju istu logičku vrednost, to postoje samo dve kombinacije za koje je taj uslov ispunjen: 0000 i 1111. Dakle, samo dva polja polja u tabeli imaju vrednost 1. Prvo polje ima oznaku vrste 00 i oznaku kolone 00, a drugo polje oznaku vrste 11 i oznaku kolone 11. Konačni izgled Karnoove karte prikazan je na slici.

AB \ CD	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

Sledi još jedan primer u kome će Karnoovom kartom biti predstavljena funkcija koja zavisi od tri promenjive *A*, *B* i *C*. Neka je vrednost funkcije 1 ako bar dve od promenljivih imaju vrednost 1.

1. Od skupa logičkih promenljivih {*A*,*B*,*C*} koji ima $n = 3$ elementa, formiraju se podskupovi V_1 od $(n\text{-}1)/2 = 1$ elementa i V_2 od $(n+1)/2 = 2$ elementa ($n1 = 1$ i $n2 = 2$). Neka je $V_1 = \{A\}$, a $V_2 = \{B, C\}$.
2. Sledi crtanje tablice sa $2^{n1} = 2^1 = 2$ vrste i $2^{n2} = 2^2 = 4$ kolone.
3. U gornji levi ugao karte, iznad vrsta upisuju se promenljive iz podskupa V_1, tj. *A*, a levo od kolona promenljive iz podskupa V_2, tj. *BC*.
4. Kao oznake vrsta unose se sve moguće vrednosti promenljive *A* ($2^1 = 2$).
5. Kao oznake kolona unose se sve kombinacije vrednosti promenljivih *B* i *C* (ima ih $2^2 = 4$), pri čemu se vodi računa da se susedne oznake razlikuju samo u jednoj binarnoj cifri. Moguć redosled oznaka kolona je: 00, 01, 11, 10.
6. Sledi popunjavanje karte vrednostima funkcije. Pošto funkcija *Y* ima vrednost 1 samo ako bar dve promenljive imaju vrednost 1, to su moguće kombinacije za koje je taj uslov ispunjen: 011, 101, 110 i 111. Dakle, četiri polja polja u tabeli imaju vrednost 1. Izgled popunjene Karnoove karte prikazan je na slici.

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Logička funkcija se može jednostavno definisati i zadavanjem polja u Karnoovoj karti u kojima se nalaze jedinice (ili nule). Da bi se to omogućilo, poljima u Karnoovoj karti pridružuju se indeksi. Indeks nekog polja predstavlja binarnu kombinaciju vrednosti promenljivih koja odgovara tom polju predstavljenu u decimalnom obliku. Ilustracije radi, na slici 4.5 date su dve Karnoove karte za funkciju četiri promenljive. U karti levo, u svakom polju je naznačena binarna kombinacija vrednosti promenljivih koja odgovara tom polju, dok su u karti desno naznačeni indeksi koji su dobijeni konverzijom binarnih kombinacija iz karte levo u decimalne brojeve.

AB \ CD	00	01	11	10
00	0000	0010	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Slika 4.5 Indeksiranje Karnoove karte za funkciju sa četiri promenljive

Sada se, uz ovako definisane indekse, logička funkcija koja ima vrednost 1 za kombinacije vrednosti promenljivih *ABCD*: 0100, 1101, 1010, 1000 i 1111, može jednostavno zadati izrazom

$Y(1) = \{4, 8, 10, 13, 15\}$ ili izrazom $Y(0) = \{0, 1, 2, 3, 5, 6, 7, 9, 11, 12\}$

i predstaviti Karnoovom kartom na slici.

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	1	1	0
10	1	0	0	1

Ovde treba imati na umu da raspored indeksa u Karnoovoj karti zavisi od izbora promenljivih koje se nalaze u podskupovima V_1 i V_2 pridruženim vrstama, odnosno kolonama. U primeru je dat uobičajen izbor promenljivih (koji će i nadalje biti korišćen) da skupu V_1 pripada „prvih“ $n1$ promenljivih, a skupu V_2 ostale promenljive funkcije.

Karnoove karte su vrlo pregledne i efikasne u slučaju predstavljanja logičkih funkcija sa malim brojem promenljivih. Već za funkcije pet promenljivih, javljaju se problemi vezani za susednost, koji utiču na otežanu dalju primenu karti. Ovi problemi se rešavaju na različite načine (crtanje više tablica manjih dimenzija, uvođenje tablica redukovanih dimenzija i sl.) koji ovom prilikom neće biti razmatrani.

4.1.4 Prelazak sa jednog načina predstavljanja funkcije na drugi

Pošto se ista logička funkcija može predstaviti na bilo koji od opisanih načina, u ovom poglavlju će biti pokazano kako se funkcija prikazana u jednom obliku jednostavno može prevesti u drugi oblik.

Prevođenje kombinacione tablice u sumu proizvoda i obrnuto

U poglavlju o predstavljanju logičke funkcije u algebarskom obliku, razmatrana su dva algebarska metoda: suma proizvoda i proizvod suma. S obzirom da će u nastavku fokus biti isključivo na sumi proizvoda, ovde će biti dat samo postupak prevođenja kombinacione tablice u ovaj način algebarskog predstavljanja logičke funkcije.

Suština postupka prevođenja kombinacione tablice u sumu proizvoda i obrnuto je u tome da je broj vrsta u kojima je vrednost funkcije 1 u kombinacionoj tablici jednak broju članova u sumi proizvoda (za svaku ovakvu vrstu formira se po jedan član). Dakle, kombinacije vrednosti promenljivih za koje funkcija ima vrednost 0 ne utiču na sumu proizvoda.

Na osnovu kombinacione tablice, suma proizvoda se može generisati na sledeći način:

1. U kombinacionoj tablici uočiti skup logičkih promenljivih od kojih funkcija zavisi, i pronaći sve vrste, tj. kombinacije vrednosti logičkih promenjivih, za koje logička funkcija ima vrednost 1.
2. Za svaku nađenu vrstu formirati potpuni proizvod koji joj odgovara (ako je vrednost neke promenjive u vrsti 0, ona u proizvod ulazi kao negirana vrednost, a ako je 1, kao originalna vrednost).
3. Napisati algebarski izraz koji logički sabira sve formirane potpune proizvode, čime je logička funkcija predstavljena sumom proizvoda.

Sledi primer prevođenja kombinacione tablice date na slici 4.6 u sumu proizvoda. Kao što se iz kombinacione tablice vidi, radi se o funkciji tri promenjive A, B i C, što znači da svaki potpuni proizvod sadrži te tri promenjive. Suma proizvoda se dobija jednostavnim logičkim sabiranjem svih potpunih proizvoda koji odgovaraju vrstama u kojima logička funkcija ima vrednost 1.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

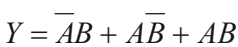
Slika 4.6 Generisanje sume proizvoda na osnovu kombinacione tablice

Postupak generisanja kombinacione tablice na osnovu date sume proizvoda je sledeći:

1. Na osnovu sume proizvoda uočiti skup n logičkih promenljivih od kojih funkcija zavisi (n odgovara broju promenljivih u bilo kom članu sume).
2. Za taj skup promenjivih nacrtati kombinacionu tablicu koja ima $1+2^n$ vrsta i $1+n$ kolona. U tablicu uneti nazive promenljivih i naziv funkcije Y.
3. U levi deo tablice uneti sve moguće kombinacije vrednosti promenljivih.
4. Za svaki potpuni proizvod u sumi proizvoda pronaći odgovarajuću kombinaciju u levom delu kombinacione tablice, pa za nju uneti vrednost logičke funkcije 1 u poslednju kolonu tablice. Preostala polja u poslednjoj koloni popuniti nulama.

Primer generisanja kombinacione tablice na osnovu sume proizvoda dat je na slici 4.7. Pošto svaki potpuni proizvod u sumi ima samo dve promenljive, to kombinaciona tablica ima 5 vrsta i 3 kolone. Kombinacije za koje funkcija ima vrednost 1, direktno se čitaju iz sume proizvoda.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Slika 4.7 Generisanje kombinacione tablice na osnovu sume proizvoda

Prevođenje kombinacione tablice u Karnoovu kartu i obrnuto

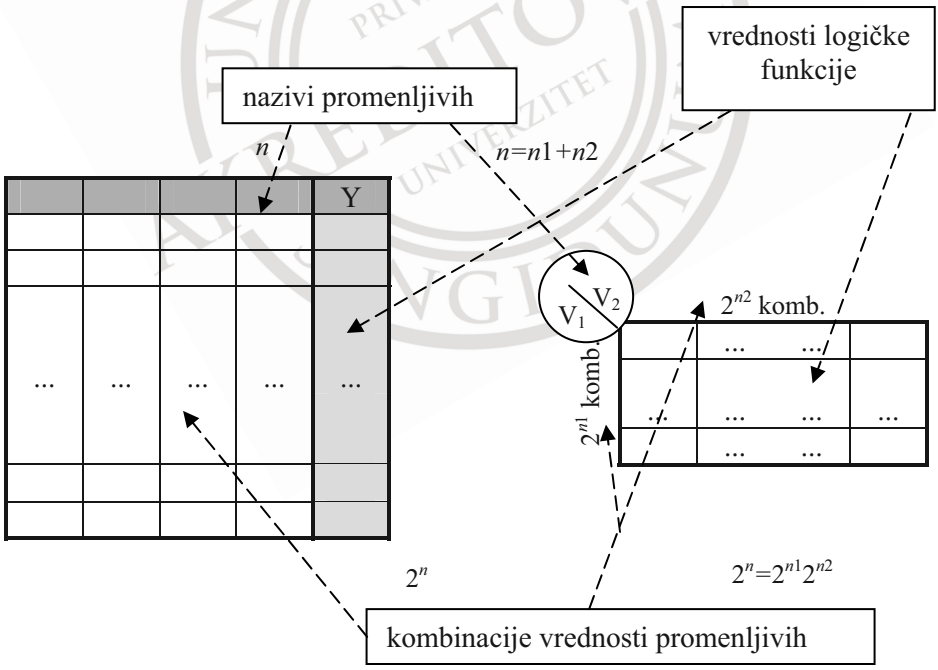
Kombinaciona tablica i Karnoova karta sadrže iste informacije o logičkoj funkciji, samo što su one u njima drugačije organizovane (videti sliku 4.8).

To su sledeće informacije:

a) *nazivi logičkih promenljivih*. U kombinacionoj tablici, imena promenljivih predstavljaju nazive kolona u levom delu tablice, dok se u Karnoovoj karti nalaze u levom gornjem uglu, kao elementi podskupova V_1 i V_2.

b) *kombinacije vrednosti logičkih promenljivih*. U kombinacionoj tablici, sve moguće kombinacije vrednosti promenljivih nalaze se u vrstama levog dela tablice. U Karnoovoj karti, svaka kombinacija je podeljena u dva dela: oznaku vrste i oznaku kolone, pa se tek njihovim spajanjem dobija kombinacija koja odgovara nekom polju u karti.

c) *vrednosti logičke funkcije za kombinacije logičkih promenljivih*. U kombinacionoj tablici, vrednosti funkcije se nalaze u koloni u desnom delu tablice, dok se u Karnoovoj karti nalaze u poljima karte.



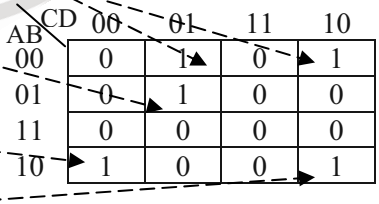
Slika 4.8 Poređenje kombinacione tablice i Karnoove karte

Imajući u vidu navedeno, kombinaciona tablica se prevodi u Karnoovu kartu na sledeći način:

1. Skup promenjivih zadat u kombinacionoj tablici podeliti u dva podskupa V_1 i V_2. Zatim, na osnovu dimenzija ovih podskupova, izračunati dimenzije Karnoove karte. Nakon crtanja karte, u gornji levi ugao upisati promenljive iz skupova V_1, odnosno V_2.
2. Formirati oznake vrsta (kao sve moguće kombinacije vrednosti promenjivih iz skupa V_1) i oznake kolona (kao sve moguće kombinacije vrednosti promenljivih iz skupa V_2).
3. U svako polje Karnoove karte uneti vrednost logičke funkcije za kombinaciju koja odgovara tom polju. Vrednost funkcije za neku kombinaciju čita se iz poslednje kolone kombinacione tablice i vrste koja odgovara toj kombinaciji u kombinacionoj tablici.

Primer prevođenja kombinacione tablice u Karnoovu kartu dat je na slici 4.9. Kao što se iz kombinacione tablice vidi, radi se o funkciji Y koja zavisi od četiri promenjive A, B, C i D koje su, radi formiranja Karnoove karte svrstane u dva podskupa $\{A, B\}$ i $\{C, D\}$, pa su dimenzije Karnoove karte 4 x 4. Vrednosti funkcije su direktno preuzete iz kombinacione tablice kao što je označeno strelicama na slici.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Slika 4.9 Generisanje Karnoove karte na osnovu kombinacione tablice

Postupak prevođenja Karnoove karte u kombinacionu tablicu je sledeći:

1. Formirati skup svih promenjivih od kojih funkcija zavisi sjedinjavanjem podskupova V_1 i V_2 u Karnoovoj karti. Za taj skup promenjivih nacrtati kombinacionu tablicu (n promenjivih zahteva kombinacionu tablicu koja ima $1+2^n$ vrsta i $1+n$ kolona). U tablicu uneti nazive promenljivih i naziv funkcije Y.
2. U levi deo tablice uneti sve moguće kombinacije vrednosti promenljivih.
3. Za svaku kombinaciju u kombinacionoj tablici, naći polje u Karnoovoj karti koje joj odgovara (po oznakama vrsta i oznakama kolona) i iz njega pročitati vrednost funkcije. Pročitanu vrednost uneti u kombinacionu tablicu kao vrednost funkcije za tu kombinaciju.

Primer prevođenja Karnoove karte u kombinacionu tablicu dat je na slici 4.10. Kao što se iz Karnoove karte vidi, radi se o funkciji tri promenjive A, B i C, pa su dimenzije kombinacione tablice 9 x 4. Vrednosti funkcije se direktno preuzimaju iz Karnoove karte iz polja koja odgovaraju kombinacijama u kombinacionoj tablici.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A \ BC	00	01	11	10
0	0	0	0	1
1	0	1	1	0

Slika 4.10 Generisanje kombinacione tablice na osnovu Karnoove karte

Prevođenje Karnoove karte u sumu proizvoda i obrnuto

Postupak prevođenja Karnoove karte u sumu proizvoda i obrnuto u osnovi je isti kao i u slučaju kombinacione tablice.

Na osnovu Karnoove karte, suma proizvoda se može generisati na sledeći način:

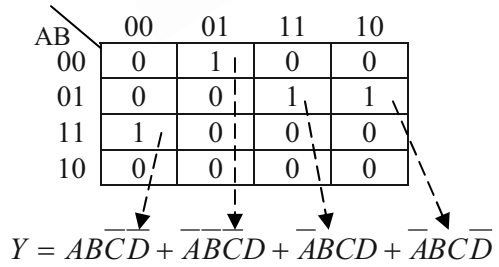
1. U Karnoovoj karti uočiti skup logičkih promenljivih od kojih funkcija zavisi (unija podskupova V_1 i V_2), i pronaći sva polja za koje logička funkcija ima vrednost 1.

2. Za svako nađeno polje formirati potpuni proizvod koji odgovara kombinaciji vrednosti promenljivih za to polje (kombinacija se dobija spajanjem oznake vrste i oznake kolone). Ako je vrednost promenljive u kombinaciji 0, ona u proizvod ulazi kao negirana vrednost, a ako je 1, kao originalna vrednost.
3. Napisati algebarski izraz koji logički sabira sve formirane potpune proizvode, čime je logička funkcija predstavljena sumom proizvoda.

Sledi primer prevođenja Karnoove karte date na slici 4.11 u sumu proizvoda. Kao što se iz Karnoove karte vidi, radi se o funkciji četiri promenjive A, B, C i D, pa svaki potpuni proizvod sadrži te četiri promenljive. Suma proizvoda se dobija logičkim sabiranjem svih potpunih proizvoda koji odgovaraju poljima u kojima logička funkcija ima vrednost 1.

Postupak formiranja Karnoove karte na osnovu sume proizvoda je sledeći:

1. Na osnovu sume proizvoda formirati skup logičkih promenljivih od kojih funkcija zavisi (sve promenljive koje se pojavljuju u bilo kom članu sume).
2. Skup promenljivih podeliti (po ranije opisanom postupku) u dva podskupa V_1 i V_2. Zatim, na osnovu dimenzija ovih podskupova, izračunati dimenzije Karnoove karte. Nakon crtanja karte, u gornji levi ugao upisati promenljive iz skupova V_1, odnosno V_2.
3. Formirati oznake vrsta (kao sve moguće kombinacije vrednosti promenjivih iz skupa V_1) i oznake kolona (kao sve moguće kombinacije vrednosti promenljivih iz skupa V_2).
4. Svaki potpuni proizvod u sumi proizvoda predstaviti odgovarajućom kombinacijom vrednosti promenljivih. Ako se promenljiva u proizvodu pojavljuje kao originalna vrednost, ona u kombinaciju ulazi kao logička vrednost 1, a ukoliko se javlja kao negirana, u kombinaciju ulazi kao 0.
5. Za dobijene kombinacije, u Karnoovoj karti pronaći polja koja im odgovaraju i u njih uneti vrednost logičke funkcije 1. U ostala polja Karnoove karte uneti vrednost 0.



Slika 4.11 Generisanje sume proizvoda na osnovu Karnoove karte

Primer generisanja Karnoove karte na osnovu sume proizvoda dat je na slici 4.12. Pošto svaki potpuni proizvod u sumi ima četiri promenljive, to Karnoova karta ima dimenzije 4 x 4. Prvi član u sumi proizvoda predstavlja potpuni proizvod koji odgovara kombinaciji vrednosti promenljivih 1000, pošto se u njemu promenljiva *A* javlja kao originalna, a promenljive *B*, *C* i *D* kao negirane vrednosti. Ovoj kombinaciji odgovara polje koje se nalazi u četvrtoj vrsti i prvoj koloni Karnoove karte, pa u njega treba uneti vrednost funkcije 1. Na sličan način se za sve ostale članove sume proizvoda popunjavaju polja u kojima Karnoova karta ima vrednost 1. Preostala polja imaju vrednost 0.

$$Y = A\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}\overline{B}C\overline{D}$$

AB \ CD	00	01	11	10
00	0	0	0	1
01	0	1	0	0
11	0	0	0	0
10	1	0	0	0

Slika 4.12 Generisanje Karnoove karte na osnovu sume proizvoda

4.2. Realizacija logičkih funkcija

Način funkcionisanja računarskog sistema u velikoj meri se može opisati logičkim funkcijama. S obzirom da one, kao i sve druge funkcije, predstavljaju apstraktan pojam, da bi se prešlo sa opisa sistema na sistem koji zaista radi, neophodno je fizički realizovati logičke funkcije koje opisuju sistem. Logičke funkcije se realizuju pomoću prekidačkih mreža koje predstavljaju najvažnije komponente računara i drugih digitalnih sistema.

Prekidačke mreže se sastoje od skupa elemenata povezanih tako da realizuju zadatu logičku funkciju. To znači da kada se na ulaze mreže dovedu određeni binarni signali, na njenim izlazima se dobijaju, u skladu sa logičkom funkcijom koju realizuju, očekivane binarne vrednosti.

Prema funkcijama koje realizuju, prekidačke mreže se dele na kombinacione i sekvencijalne. Glavna osobina kombinacione mreže je da vrednost na njenom izlazu (vrednost logičke funkcije) zavisi samo od trenutnog stanja na njenom ulazu (vrednosti logičkih promenljivih funkcije). Kombinacione mreže se realizuju kao kompozicija logičkih elemenata. Logički elementi, po pravilu, realizuju samo jednu jednostavnu prekidačku funkciju (imaju samo jedan izlaz). Opisuju se funkcijom koju realizuju, grafičkim simbolom i nazivom elementa. U logičke elemente,

između ostalih, spadaju logička kola *I*, *ILI*, *NE*, ekskluzivno *ILI*, itd. Za razliku od kombinacione, kod sekvencijalne mreže vrednost na izlazu (vrednost logičke funkcije) zavisi, ne samo od trenutnog stanja na ulazu (vrednosti logičkih promenljivih), već i od stanja u kome se mreža nalazi u datom trenutku (vrednosti na unutrašnjim linijama mreže). Sekvencijalne mreže se realizuju kao kompozicija logičkih i memorijskih elemenata, ili samo logičkih elemenata. Da li će kompozicija logičkih elemenata predstavljati kombinacionu ili sekvencijalnu mrežu, zavisi od načina povezivanja. Memorijski elementi se obično realizuju kao kompozicija logičkih elemenata. Glavna osobina memorijskih elemenata je da imaju samo dva stanja na koja utiču povratne veze koje postoje u strukturi ovih elemenata.

Rad sa prekidačkim mrežama podrazumeva rešavanje dve vrste problema. U nekim slučajevima potrebno je za postojeću prekidačku mrežu odrediti funkciju koju ona obavlja. Rešavanje ovog problema je predmet analize prekidačke mreže. Međutim, moguća je i obrnuta situacija u kojoj je potrebno zadatu logičku funkciju realizovati pomoću prekidačke mreže. Ovaj postupak se naziva sintezom prekidačke mreže.

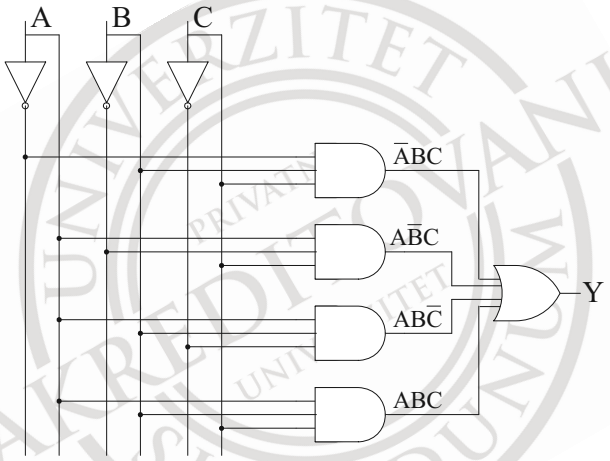
U ovom poglavlju će biti opisan jedan postupak sinteze prekidačke mreže koja realizuje logičku funkciju zadatu u aglebarskom obliku (ukoliko je funkcija predstavljena na neki drugi način, ranije opisanim postupcima može se prevesti u algebarski oblik).

Neka je data logička funkcija *Y* koja zavisi od *n* promenljivih međusobno povezanih logičkim operacijama. Ona se može realizovati prekidačkom mrežom koja:

- ima *n* ulaza koji odgovaraju logičkim promenljivama i jedan izlaz koji predstavlja vrednost funkcije *Y*
- ima onoliko različitih vrsta logičkih kola (*NE*, *I*, *ILI*, ...) koliko ima različitih logičkih operacija u funkciji
- ima onoliko logičkih kola jedne vrste koliko ih je potrebno za obavljanje svih logičkih operacija te vrste u funkciji

Ovo će biti ilustrovano na primeru funkcije $Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$. Prekidačka mreža koja realizuje ovu funkciju ima tri ulaza koja odgovaraju promenljivama *A*, *B* i *C* i jedan izlaz *Y*. U strukturi prekidačke mreže postoje tri vrste logičkih kola: *NE*, *I* i *ILI*. Pošto se svaka promenljiva u funkciji pojavljuje i kao negirana vrednost, prekidačka mreža mora da sadrži tri invertora (*NE* kola). Osim toga, za realizaciju svakog člana u zbiru potrebno je po jedno *I* kolo, što ukupno zahteva četiri logička *I* kola (ovde se dopušta da logičko *I* kolo ima više ulaza i jedan izlaz). Takođe, potrebno je i jedno *ILI* kolo sa više ulaza za realizaciju logičkog zbira.

U praktičnoj realizaciji, pogodno je poći od izlaza prekidačke mreže. Kao što se iz funkcije vidi, vrednost *Y* se dobija kao logički zbir četiri binarna signala. Stoga izlaz prekidačke mreže treba istovremeno da bude i izlaz logičkog *ILI* kola. Na ulaze *ILI* kola treba dovesti četiri binarna signala koji odgovaraju članovima zbira. To se može postići tako što se na svaki ulaz *ILI* kola dovede izlaz *I* kola koje realizuje odgovarajući član zbira. Da bi *I* kolo realizovalo član zbira, na njegove ulaze treba dovesti odgovarajuću kombinaciju logičkih promenljivih koje se u njemu pojavljuju. Ako se promenljiva u članu javlja kao originalna vrednost, na ulaz *I* kola se dovodi direktno sa ulaza prekidačke mreže, a ako se javlja kao negirana vrednost, mora se dovesti sa izlaza logičkog *NE* kola na čiji je ulaz dovedena originalna vrednost te promenljive sa ulaza prekidačke mreže. Opisani postupak realizacije se može ispratiti na slici 4.13 koja predstavlja konačni izgled prekidačke mreže koja realizuje zadatu logičku funkciju.



Slika 4.13 Realizacija logičke funkcije pomoću prekidačke mreže

4.3. Minimizacija logičkih funkcija

Struktura prekidačke mreže (broj logičkih kola i način njihovog povezivanja) zavisi od algebarskog izraza kojim je predstavljena logička funkcija. U većini algebarskih načina predstavljanja, ista logička funkcija se može napisati u vidu različitih algebarskih izraza koji ne moraju biti jednako pogodni za praktičnu realizaciju. Sa stanovišta ekonomičnosti i efikasnosti, cilj je da se za realizaciju funkcije upotrebi što manji broj logičkih kola uz što manje veza između njih. Minimizacijom se naziva postupak određivanja najprostijeg između više algebarskih izraza kojima se logička funkcija može predstaviti. Naziv postupka

potiče od toga što se za poređenje izraza koristi neka veličina koja u najprostijem izrazu ima minimalnu vrednost (na primer broj operacija, broj simbola i sl.).

Postoje različite metode minimizacije logičkih funkcija. Za ranije opisane algebarske načine predstavljanja funkcija, najopštija podela metoda minimizacije je na grafičke i algoritamske. Grafičke metode se zasnivaju na vizuelnoj analizi grafički predstavljene logičke funkcije. Ove metode su teorijski vrlo jednostavne, a pogodne su za funkcije sa manje promenljivih (do 6). Algoritamske metode koriste razne algoritme za transformisanje analitički ili tabelarno predstavljene funkcije. Složenije su od grafičkih metoda, ali su zato efikasne i u slučaju funkcija sa znatno većim brojem promenljivih.

U ovom poglavlju će biti prikazana najčešće korišćenja grafička metoda minimizacije koja se zasniva na primeni Karnoove karte. Metoda će biti samo opisana, tj. tvrdnje i pravila na kojima ona zasniva neće biti dokazivani.

Postupak minimizacije logičke funkcije pomoću Karnoove karte sprovodi se u tri koraka:

1. zadatu logičku funkciju predstaviti popunjenom Karnoovom kartom na ranije opisan način
2. od polja Karnoove karte u kojima logička funkcija ima vrednost 1 formirati pravougaone površine poštujući unapred definisana pravila
3. na osnovu pravougaonih površina, po određenim pravilima, ispisati minimalni zapis logičke funkcije u vidu sume proizvoda

Pri formiranju pravougaonih površina, moraju biti poštovana sledeća pravila:

- pravougaone površine sadrže samo ona polja Karnoove karte u kojima logička funkcija ima vrednost 1 (nijedno polje sa vrednošću 0 ne može pripadati pravougaonoj površini)
- broj polja u pravougaonoj površini može biti samo 2^k, $k = 0,1,2,...$ (površina može imati samo 1,2,4,8, ... polja)
- jednu pravougaonu površinu mogu da čine samo susedna polja u kojima je vrednost logičke funkcije 1
- pravougaone površine treba da budu što je moguće veće (da sadrže što više polja), a njihov broj što manji
- prema potrebi, isto polje može se naći u više pravougaonih površina

Pod susednim poljima u Karnoovoj karti podrazumevaju se:

- dva polja koja imaju jednu zajedničku stranicu

- dva polja koja se nalaze u istoj vrsti, s tim što je jedno polje u prvoj, a drugo u poslednjoj koloni (ova susednost se može videti ako se Karnoova karta urola kao cilindar po vertikali)
- dva polja koja se nalaze u istoj koloni, s tim što je jedno polje u prvoj, a drugo u poslednjoj vrsti (ova susednost se može videti ako se Karnoova karta urola kao cilindar po horizontali)

Nakon formiranja pravougaonih površina, može se pristupiti ispisivanju minimalnog zapisa logičke funkcije. Minimalni zapis ima oblik sume proizvoda sa onoliko članova (proizvoda) koliko ima pravougaonih površina, jer se za svaku pravougaonu površinu generiše po jedan član (proizvod) sume. Proizvod koji odgovara jednoj pravougaonoj površini dobija se analizom oznaka vrsta i kolona za sva polja koja pripadaju toj površini. On se formira tako što se za svaku logičku promenljivu pojedinačno analizira da li ona ulazi u proizvod i, ako ulazi, na koji način. Ova analiza se obavlja po sledećim pravilima:

- ako u oznakama vrsta koje odgovaraju pravougaonoj površini promenljiva ima vrednost i 0 i 1 (u oznaci jedne vrste ima vrednost 1, a u oznaci neke druge vrste vrednost 0), ta promenljiva ne ulazi u proizvod
- ako u svim oznakama vrsta koje odgovaraju pravougaonoj površini promenljiva ima vrednost 1, ta promenljiva ulazi u proizvod sa svojom originalnom vrednošću
- ako u svim oznakama vrsta koje odgovaraju pravougaonoj površini promenljiva ima vrednost 0, ta promenljiva ulazi u proizvod sa svojom negiranom vrednošću
- ako u oznakama kolona koje odgovaraju pravougaonoj površini promenljiva ima vrednost i 0 i 1 (u oznaci jedne kolone ima vrednost 1, a u oznaci neke druge kolone vrednost 0), ta promenljiva ne ulazi u proizvod
- ako u svim oznakama kolona koje odgovaraju pravougaonoj površini promenljiva ima vrednost 1, ta promenljiva ulazi u proizvod sa svojom originalnom vrednošću
- ako u svim oznakama kolona koje odgovaraju pravougaonoj površini promenljiva ima vrednost 0, ta promenljiva ulazi u proizvod sa svojom negiranom vrednošću

Iako opisani postupak minimizacije pomoću Karnoove karte izgleda prilično složeno, već nakon nekoliko praktično urađenih primera, on prelazi u rutinu. U nastavku će biti prikazano nekoliko karakterističnih primera minimizacije logičke funkcije Y koja zavisi od četiri logičke promenljive A, B, C i D.

Primer 1. Minimizirati logičku funkciju zadatu Karnoovom kartom na slici.

AB \ CD	00	01	11	10
00	0	1	0	1
01	0	0	0	1
11	1	1	0	1
10	0	0	0	1

Rešenje: Pri formiranju pravougaonih površina, pogodno je poći od onih polja sa vrednošću 1 koja nemaju susednih polja, ili, ako takvih nema, od polja sa najmanjim brojem susednih polja. Ovaj princip se dalje može koristiti do kraja postupka formiranja pravougaonih površina.

U ovom primeru postoji samo jedno polje koje nema susednih, a to je polje kome odgovara kombinacija 0001. Stoga se od njega formira prva pravougaona površina koja ima samo jedno polje. Sledeće polje sa najmanje suseda odgovara kombinaciji 1101. Ono ima samo jednog suseda (polje 1100) i stoga mora sa njim formirati pravougaonu površinu. Polje 1100 ima i drugog suseda, polje 1110, ali za njih nema potrebe formirati pravougaonu površinu, jer je polje 1110 bolje uključiti u veću pravougaonu površinu sa ostalim poljima poslednje kolone.

Pošto su formirane pravougaone površine, može se ispisati minimalna forma logičke funkcije. Ona predstavlja sumu proizvoda od tri člana. Neka prvi član odgovara pravougaonoj površini od jednog polja, drugi od dva polja, a treći površini od četiri polja.

Prvi član se generiše sledećom analizom:

- pošto u oznaci prve vrste promenljivama *A* i *B* odgovaraju vrednosti 0, obe promenljive ulaze u proizvod kao negirane vrednosti
- pošto u oznaci druge kolone promenljivoj *C* odgovara vrednost 0, a promenljivoj *D* vrednost 1, onda promenljiva *C* ulazi u proizvod kao negirana, a promenljiva *D* kao originalna vrednost

Drugi član se generiše sledećom analizom:

- pošto u oznaci treće vrste promenljivama *A* i *B* odgovaraju vrednosti 1, obe promenljive ulaze u proizvod kao originalne vrednosti
- pošto u oznakama prve i druge kolone promenljivoj *C* odgovara vrednost 0, ona u proizvod ulazi kao negirana vrednost
- pošto u oznakama prve i druge kolone promenljivoj *D* odgovaraju vrednost 0 u prvoj koloni, a vrednost 1 u drugoj, to vrednost ove promenljive ne

utiče na vrednost funkcije u ovoj površini, pa promenljiva D ne ulazi u proizvod

Treći član se generiše sledećom analizom:

- pošto u oznakama vrsta (od prve do četvrte) promenljivama A i B odgovaraju vrednosti bilo 0 bilo 1, ove promenljive ne ulaze u proizvod
- pošto u oznaci četvrte kolone promenljivoj C odgovara vrednost 1, a promenljivoj D vrednost 0, onda promenljiva C ulazi u proizvod kao originalna, a promenljiva D kao negirana vrednost

Na osnovu sprovedene analize, dobija se sledeća minimalna forma funkcije:

$$Y = \overline{A}\,\overline{B}\,\overline{C}D + AB\overline{C} + C\overline{D}$$

Iz dobijenog izraza se vidi da je proizvod u sumi jednostavniji ukoliko mu odgovara pravougaona površina sa više polja.

Primer 2. Naći minimalnu formu logičke funkcije zadate skupom indeksa

$$Y(1) = \{2, 3, 6, 8, 9, 10, 11, 12\}.$$

Rešenje: U ovom primeru postoje dva polja (0011 i 0110) koja imaju samo po jedno susedno polje. Stoga se za svako od ovih polja mora napraviti pravougaona površina koja obuhvata to polje i njemu susedno. Pošto je susedno polje u oba slučaja isto (polje 0010), to će ono biti uključeno u dve pravougaone površine, što je dozvoljeno. Poslednja pravougaona površina obuhvata četiri preostale susedne jedinice.

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	1	0	0
10	1	1	0	0

Minimalna forma logičke funkcije ima tri člana. Neka prvi član odgovara površini koju čine polja 0011 i 0010, drugi površini koju čine polja 0010 i 0110, a treći površini od četiri polja.

Prvi član se generiše sledećom analizom:

- pošto u oznaci prve vrste promenljivama A i B odgovaraju vrednosti 0, obe promenljive ulaze u proizvod kao negirane vrednosti
- pošto u oznakama treće i četvrte kolone promenljivoj C odgovara vrednost 1, a promenljivoj D i 0 i 1, to u proizvod ulazi samo promenljiva C i to kao originalna vrednost

Drugi član se generiše sledećom analizom:

- pošto u oznaci poslednje kolone promenljivoj C odgovara vrednost 1, a promenljivoj D vrednost 0, to promenljiva C ulazi u proizvod kao originalna vrednost, a promenljiva D kao negirana
- pošto u oznakama prve i druge vrste promenljivoj A odgovara vrednost 0, a promenljivoj B i 0 i 1, to u proizvod ulazi samo promenljiva A i to kao negirana vrednost

Treći član se generiše sledećom analizom:

- pošto u oznakama treće i četvrte vrste promenljivoj A odgovara vrednost 1, a promenljivoj B i 0 i 1, to u proizvod ulazi samo promenljiva A i to kao originalna vrednost
- pošto u oznakama prve i druge kolone promenljivoj C odgovara vrednost 0, a promenljivoj D i 0 i 1, to u proizvod ulazi samo promenljiva C i to kao negirana

Na osnovu sprovedene analize, dobijena je sledeća minimalna forma funkcije:

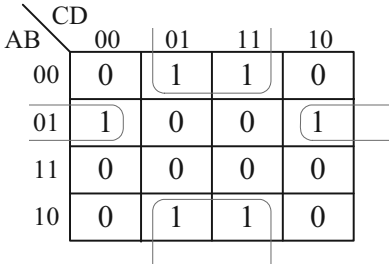
$$Y = \overline{A}\,\overline{B}C + \overline{A}C\overline{D} + A\overline{C}$$

Iz navedenog izraza može se zaključiti da pravouganim površinama sa istim brojem polja odgovaraju proizvodi sa istim brojem promenljvih (za površine od dva polja, proizvodi sadrže tri promenljive, a za površinu od četiri polja, samo dve). Ova činjenica može da posluži za brzu proveru ispravnosti formiranih proizvoda.

Primer 3. Minimizirati logičku funkciju zadatu skupom indeksa

$$Y(1) = \{1, 3, 4, 6, 9, 11\}.$$

Rešenje: Ovaj primer ilustruje susednost prve i poslednje vrste, odnosno prve i poslednje kolone. U skladu sa rasporedom polja sa vrednošću 1, mogu se napraviti dve pravougaone površine: prvu čine polja 0001, 0011, 1001 i 1011, a drugu polja 0100 i 0110.



Minimalna forma logičke funkcije ima dva člana. Neka prvi član odgovara površini od četiri polja, a drugi površini od dva polja.

Prvi član se generiše sledećom analizom:

- pošto u oznakama prve i četvrte vrste promenljivoj *B* odgovara vrednost 0, a promenljivoj *A* i 0 i 1, to u proizvod ulazi samo promenljiva *B* i to kao negirana vrednost
- pošto u oznakama druge i treće kolone promenljivoj *D* odgovara vrednost 1, a promenljivoj *C* i 0 i 1, to u proizvod ulazi samo promenljiva *D* i to kao originalna vrednost

Drugi član se generiše sledećom analizom:

- pošto u oznaci druge vrste promenljivoj *A* odgovara vrednost 0, a promenljivoj *B* vrednost 1, to promenljiva *B* ulazi u proizvod kao originalna vrednost, a promenljiva *A* kao negirana
- pošto u oznakama prve i četvrte kolone promenljivoj *D* odgovara vrednost 0, a promenljivoj *C* i 0 i 1, to u proizvod ulazi samo promenljiva *D* i to kao negirana vrednost

Na osnovu sprovedene analize, dobijena je sledeća minimalna forma funkcije:

$$Y = \overline{B}D + \overline{A}B\overline{D}$$

Primer 4. Minimizirati logičku funkciju zadatu Karnoovom kartom na sledećoj slici.

Rešenje: Ovaj primer ilustruje susednost uglova Karnoove karte. Naime, u skladu sa ranije datim opisom susednosti polja, polja u uglovima Karnoove karte (0000, 0010, 1000 i 1010) su susedna i formiraju prvu pravougaonu površinu. Drugu pravougaonu površinu formira svih osam polja u levom delu karte (cilj je da površine budu što veće).

AB \ CD	00	01	11	10
00	1	1	0	1
01	1	1	0	0
11	1	1	0	0
10	1	1	0	1

Minimalna forma logičke funkcije ima dva člana. Neka prvi član odgovara površini od četiri polja, a drugi površini od osam polja.

Prvi član se generiše sledećom analizom:

- pošto u oznakama prve i četvrte vrste promenljivoj *B* odgovara vrednost 0, a promenljivoj *A* i 0 i 1, to u proizvod ulazi samo promenljiva *B* i to kao negirana vrednost
- pošto u oznakama prve i četvrte kolone promenljivoj *D* odgovara vrednost 0, a promenljivoj *C* i 0 i 1, to u proizvod ulazi samo promenljiva *D* i to kao negirana vrednost

Drugi član se generiše sledećom analizom:

- pošto u oznakama vrsta (od prve do četvrte) promenljivama *A* i *B* odgovaraju bilo 0 bilo 1, to one ne ulaze u proizvod
- pošto u oznakama prve i druge kolone promenljivoj *C* odgovara vrednost 0, a promenljivoj *D* i 0 i 1, to u proizvod ulazi samo promenljiva *C* i to kao negirana vrednost

Na osnovu sprovedene analize, dobijena je sledeća minimalna forma funkcije:

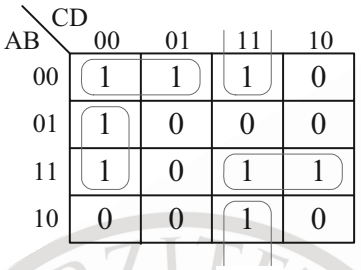
$$Y = \overline{B}\,\overline{D} + \overline{C}$$

Primer 5. Minimizirati logičku funkciju zadatu Karnoovom kartom na slici.

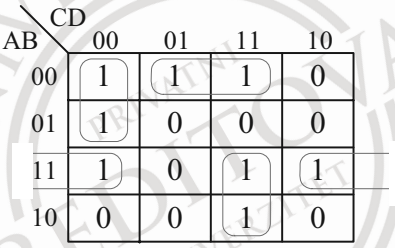
AB \ CD	00	01	11	10
00	1	1	1	0
01	1	0	0	0
11	1	0	1	1
10	0	0	1	0

Rešenje: U ovom primeru biće pokazano da se za istu logičku funkciju može generisati više minimalnih formi. Naime, na osnovu date Karnoove karte mogu se formirati četiri pravougaone površine od po dva polja na dva načina.

I način: 0100 i 1100, 0000 i 0001, 1111 i 1110, 0011 i 1011



II način: 0000 i 0100, 0001 i 0011, 1111 i 1011, 1100 i 1110



Ako se pravougaone površine odaberu u skladu sa I načinom, dobija se minimalna forma:

$$Y = B\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C} + AB\overline{C} + \overline{B}CD$$

Ukoliko se pravougaone površine izaberu u skladu sa II načinom, minimalni oblik funkcije je:

$$Y = \overline{A}\overline{C}\overline{D} + \overline{A}\overline{B}D + ACD + AB\overline{D}$$

Iako se dobijene minimalne forme međusobno razlikuju po promenljivama koje se u njima javljaju, sa stanovišta realizacije, one imaju istu složenost. To znači da je prilikom minimizacije bitno da se postigne što je moguće manji broj pravougaonih površina sa što više polja u njima, a koja će to konkretna polja biti, nije od većeg interesa.

Vežbanja

1. Logičke funkcije (Y) prikazane datim kombinacionim tablicama predstaviti:
   a) u obliku suma proizvoda
   b) u vidu Karnoovih karti

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
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	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
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

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

2. Logičke funkcije zadate u algebarskom obliku predstaviti:
   a) kombinacionim tablicama
   b) u vidu Karnoovih karti

$$Y = \overline{A}BCD + \overline{A}BC\overline{D} + AB\overline{C}D + A\overline{B}\overline{C}D$$
$$Y = \overline{A}\overline{B}CD + AB\overline{C}D + ABCD + A\overline{B}\overline{C}D + A\overline{B}CD$$
$$Y = \overline{A}B\overline{C}D + \overline{A}BCD + ABCD$$

$Y = \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D} + ABC\overline{D} + A\overline{B}\overline{C}\overline{D}$

$Y = \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC + AB\overline{C}$

3. Logičke funkcije prikazane Karnoovim kartama na slici predstaviti:
   a) u algebarskom obliku
   b) pomoću kombinacionih tablica

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	0	1	0	0
10	0	0	1	0

AB \ CD	00	01	11	10
00	0	1	0	1
01	0	0	1	0
11	0	1	0	0
10	0	1	0	0

AB \ C	0	1
00	1	0
01	0	1
11	0	0
10	0	1

4. Zadate logičke funkcije realizovati pomoću prekidačkih mreža:
   a) $Y = \overline{A}\overline{B}C + A\overline{B}\overline{C} + AB\overline{C}$
   b) $Y = \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}C\overline{D}$
   c) $Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C}$

5. Primenom Karnoove karte minimizirati logičke funkcije predstavljene sledećim sumama proizvoda:
   a) $Y = \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}C\overline{D} + A\overline{B}\overline{C}\overline{D}$
   b) $Y = \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}D + A\overline{B}C\overline{D}$
   c) $Y = \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + A\overline{B}CD$
   d) $Y = \overline{A}\overline{B}CD + AB\overline{C}D + ABCD + A\overline{B}\overline{C}D + A\overline{B}CD$
   e) $Y = \overline{A}\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + A\overline{B}CD$
   f) $Y = \overline{A}B\overline{C}\overline{D} + \overline{A}BC\overline{D} + AB\overline{C}D + ABCD$
   g) $Y = \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D} + ABC\overline{D} + A\overline{B}\overline{C}\overline{D}$
   h) $Y = \overline{A}\overline{B}\overline{C}D + \overline{A}BCD + ABCD + A\overline{B}CD$
   i) $Y = \overline{A}\overline{B}C + \overline{A}BC + AB\overline{C}$
   j) $Y = \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}$

6. Izvršiti minimizaciju logičkih funkcija (Y) koje zavise od četiri logičke promenljive (A, B, C i D), a zadate su pomoću indeksa u Karnoovoj karti:

a) $Y(1) = (0, 1, 2, 5, 8, 10)$
b) $Y(1) = (2, 4, 6, 8, 10, 14)$
c) $Y(1) = (3, 9, 12, 13, 14)$
d) $Y(1) = (0, 6, 7, 8, 14, 15)$
e) $Y(1) = (4, 6, 7, 8, 12)$
f) $Y(0) = (4, 5, 7, 12, 14, 15)$

5 Logičke mreže

Pri konstrukciji računarskih i drugih digitalnih sistema izuzetno važnu ulogu imaju logičke, tj. prekidačke mreže. Pomoću njih se mogu ispuniti najrazličitiji funkcionalni zahtevi. Naravno, da bi se to postiglo, potrebno je uložiti dosta rada i vremena kako za projektovanje, tako i za samu realizaciju mreža. To se posebno odnosi na složene sisteme, kao što su na primer računarski sistemi, koji imaju vrlo veliki broj raznovrsnih funkcionalnih zahteva koje treba da ispune. Olakšavajuća okolnost je ta što se u različitim modelima i verzijama računarskih i drugih digitalnih sistema javljaju mnogi zajednički zahtevi. Radi uštede u vremenu potrebnom za razvoj sistema, prekidačke mreže koje se često koriste izdvojene su kao standardni moduli. Procedure i postupci za razvoj standardnih modula su definisani, ali njihovo poznavanje nije neophodno za upotrebu modula. Naime, za korišćenje i uključivanje modula u složenije logičke strukture dovoljno je znati način njihovog funkcionisanja, tj. vezu između njihovih ulaza i izlaza.

U zavisnosti od elemenata od kojih su napravljeni, standardni moduli mogu biti kombinacionog ili sekvencijalnog tipa. U nastavku će biti opisano po nekoliko najčešće primenjivanih modula svakog tipa.

5.1. Standardni kombinacioni moduli

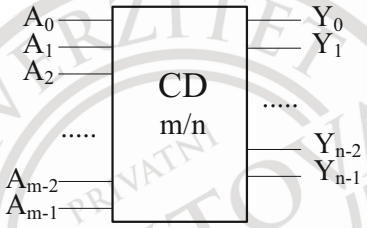
Standardni kombinacioni moduli su moduli koji sadrže samo logičke elemente. Od brojnih modula ovog tipa, za ovu priliku izdvojeni su sledeći: koder, dekoder, multiplekser, demultiplekser, sabirač i aritmetičko-logička jedinica.

Za svaki od navedenih modula biće opisano čemu služi i kako se realizuje, a za neke od njih (dekoder, multiplekser, ...) biće dati i primeri korišćenja u drugim logičkim strukturama.

5.1.1 Koderi

Koderi su kombinacione mreže sa više ulaza i više izlaza koje služe za kodiranje informacija, tj. predstavljanje informacija u binarnom obliku. Informacija koju treba kodirati dovodi se na jedan od ulaza kodera, a njena binarna vrednost dobija se na njegovim izlazima. Ulaz kodera na koji je doveden signal koji predstavlja informaciju obično se označava vrednošću 1. Da bi koder ispravno radio, na svim ostalim ulazima mora biti vrednost 0. Dakle, kod kodera, u datom trenutku može biti aktivan jedan i samo jedan ulaz. Ukoliko se istovremeno dovedu signali na dva ili više ulaza kodera (što je fizički moguće uraditi), kodirana vrednost na izlazu neće biti ispravna.

Grafički simbol kodera sa *m* ulaza i *n* izlaza prikazan je na slici 5.1.



Slika 5.1 Koder *m*/*n*

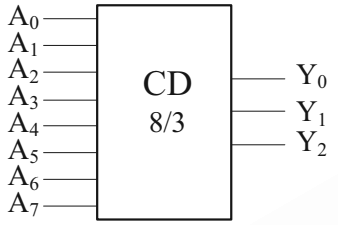
Koder koji ima *n* izlaza može da generiše najviše 2^n različitih kodnih kombinacija na svom izlazu. Pošto svaka kombinacija predstavlja kodiranu vrednost informacije dovedene na jedan od ulaza kodera, to znači da broj ulaza kodera ne može da bude veći od 2^n. U zavisnosti od broja ulaza i izlaza, razlikuju se dve vrste kodera:

- *potpuni koderi*, kod kojih važi $m = 2^n$ (imaju 2^n ulaza i *n* izlaza, što znači da su u potpunosti iskorišćene mogućnosti kodovanja)
- *nepotpuni koderi*, kod kojih važi $m < 2^n$ (imaju manje od 2^n ulaza i *n* izlaza, što znači da se neke kodne kombinacije nikad ne mogu pojaviti na izlazu)

U nastavku će biti detaljno opisan primer potpunog kodera 8/3. Ovaj koder ima 8 ulaza i 3 izlaza kao što je prikazano na slici 5.2.

Kao što je rečeno, u datom trenutku može biti aktivan (vrednost 1) samo jedan od ulaza kodera. Neka se kod razmatranog kodera u zavisnosti od rednog broja aktivnog ulaza, na izlazu generiše kombinacija bitova koja odgovara tom rednom broju. Na primer, ako se signal dovede na ulaz broj 3 (A_3=1, ostali ulazi imaju vrednost 0), na izlazima će se generisati binarna vrednost 011 (Y_2=0, Y_1=1 i Y_0=1).

Ovakav način funkcionisanja kodera može se opisati kombinacionom tablicom prikazanom na slici 5.3.



Slika 5.2 Koder 8/3

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

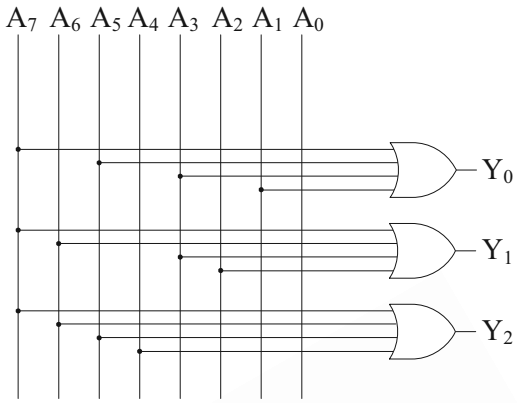
Slika 5.3 Kombinaciona tablica kodera 8/3

Leva strana tablice ne sadrži sve moguće kombinacije ulaznih promenljivih (njihov broj je $2^8 = 256$), već samo one u kojima je samo jedan od ulaza aktivan (8 kombinacija). To je u skladu sa već opisanim načinom funkcionisanja kodera (ostale kombinacije se ne mogu pojaviti na ulazu pri regularnom radu kombinacione mreže, pa nema potrebe unositi ih u tablicu). Na desnoj strani tablice date su funkcije izlaza Y_2, Y_1 i Y_0 koje, za datu ulaznu kombinaciju, formiraju odgovarajuću 3-bitnu binarnu vrednost.

Zavisnost izlaza od ulaza se može predstaviti sledećim prekidačkim funkcijama (videti kombinacionu tablicu):

$$Y_0 = A_1 + A_3 + A_5 + A_7$$
$$Y_1 = A_2 + A_3 + A_6 + A_7$$
$$Y_2 = A_4 + A_5 + A_6 + A_7$$

Na osnovu datih funkcija izlaza, razmatrani koder 8/3 može se realizovati pomoću prekidačke mreže predstavljene na slici 5.4.

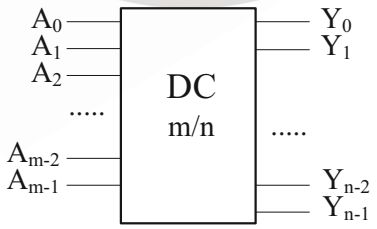


Slika 5.4 Prekidačka mreža koja realizuje koder 8/3

5.1.2 Dekoderi

Dekoderi su kombinacione mreže sa više ulaza i više izlaza čija je funkcija u osnovi inverzna funkciji kodera. Oni služe za dekodovanje binarno kodiranih informacija, tj. generisanje informacija u izvornom obliku. Binarno kodirana informacija se dovodi na ulaze dekodera, a izvorna informacija se dobija na jednom od njegovih izlaza. Izlaz dekodera na kome se pojavljuje informacija obično se označava vrednošću 1, dok je na svim ostalim izlazima vrednost 0. Dakle, kod dekodera, u datom trenutku aktivan je jedan i samo jedan izlaz. Ovde ne može da dođe do greške kao u slučaju kodera zato što korisnik može da postavlja samo ulazne signale, dok se izlazni signali generišu automatski u skladu sa logikom rada samog dekodera.

Grafički simbol dekodera sa *m* ulaza i *n* izlaza prikazan je na slici 5.5.



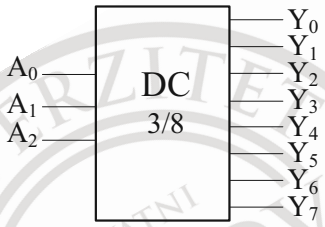
Slika 5.5 Dekoder *m*/*n*

Kod dekodera koji ima *m* ulaza, na njih se može dovesti najviše 2^m različitih binarno kodiranih informacija. Pošto svakoj ulaznoj informaciji odgovara po jedan

izlaz, to znači da broj izlaza dekodera ne može da bude veći od 2^m. U zavisnosti od broja ulaza i izlaza, razlikuju se dve vrste dekodera:

- *potpuni dekoderi*, kod kojih važi $n = 2^m$ (imaju m ulaza i 2^m izlaza, što znači da su u potpunosti iskorišćene mogućnosti dekodovanja)
- *nepotpuni dekoderi*, kod kojih važi $n < 2^m$ (imaju m ulaza i manje od 2^m izlaza, što znači da se neke binarne kombinacije nikad ne pojavljuju na ulazu)

Ilustracije radi, sledi detaljan opis primera potpunog dekodera 3/8 koji ima 3 ulaza i 8 izlaza kao što je prikazano na slici 5.6.



Slika 5.6 Dekoder 3/8

U datom trenutku na ulaze dekodera dovodi se binarna kombinacija od 3 bita koja predstavlja ranije kodovanu informaciju. U zavisnosti od dovedene kombinacije, aktivira se samo jedan od izlaza dekodera i to onaj koji odgovara ulaznoj kombinaciji. Neka kod razmatranog dekodera ulaznoj kombinaciji odgovara onaj izlaz čiji redni broj predstavlja decimalnu vrednost kombinacije bita na ulazu. Na primer, ako se na ulaze dekodera dovede kombinacija 011 (A_0=0, A_1=1 i A_2=1), aktiviraće se izlaz sa rednim brojem 3 (Y_3=1, ostali izlazi imaju vrednost 0). Ovakav način funkcionisanja dekodera može se opisati kombinacionom tablicom prikazanom na slici 5.7.

A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	1
1	1	0	0	1	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0

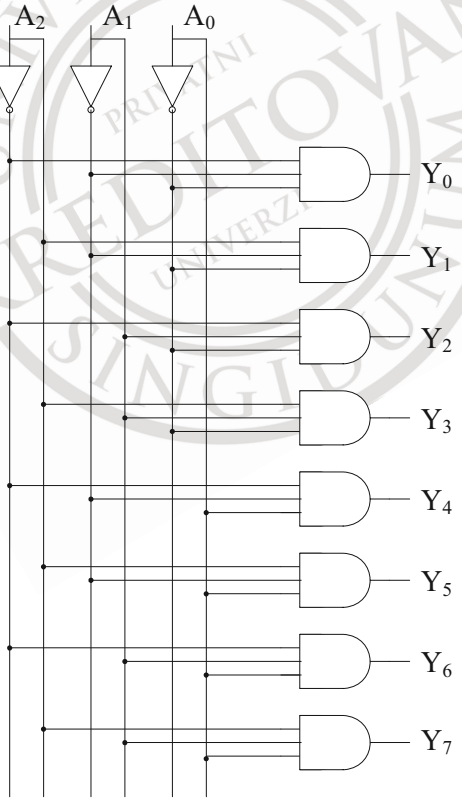
Slika 5.7 Kombinaciona tablica dekodera 3/8

Pošto se radi o potpunom dekoderu, leva strana tablice sadrži sve moguće kombinacije ulaznih promenljivih ($2^3 = 8$ kombinacija). Na desnoj strani tablice date su funkcije izlaza Y_7, Y_6, Y_5, Y_4, Y_3, Y_2, Y_1 i Y_0 koje za datu ulaznu kombinaciju aktiviraju samo njoj odgovarajući izlaz.

Zavisnost izlaza od ulaza može se predstaviti sledećim prekidačkim funkcijama (videti kombinacionu tablicu):

$$Y_0 = \bar{A}_0 \bar{A}_1 \bar{A}_2 \qquad Y_4 = A_0 \bar{A}_1 \bar{A}_2$$
$$Y_1 = \bar{A}_0 \bar{A}_1 A_2 \qquad Y_5 = A_0 \bar{A}_1 A_2$$
$$Y_2 = \bar{A}_0 A_1 \bar{A}_2 \qquad Y_6 = A_0 A_1 \bar{A}_2$$
$$Y_3 = \bar{A}_0 A_1 A_2 \qquad Y_7 = A_0 A_1 A_2$$

Na osnovu prikazanih funkcija izlaza, dekoder 3/8 može se realizovati pomoću prekidačke mreže predstavljene na slici 5.8.

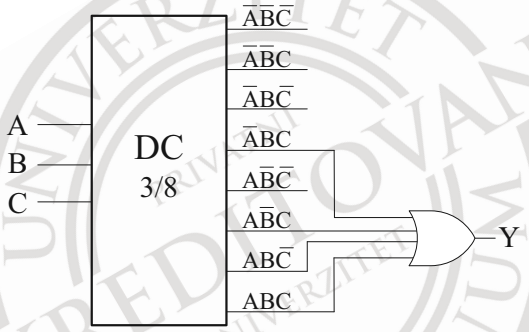


Slika 5.8 Prekidačka mreža koja realizuje dekoder 3/8

Dekoderi, kao i ostale logičke mreže, nalaze svoju primenu i u složenijim logičkim strukturama. U nastavku će biti opisan primer korišćenja dekodera za realizaciju zadate logičke funkcije. Neka je data funkcija

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC\,.$$

Za realizaciju ove funkcije pogodno je koristiti jedno *ILI* kolo sa četiri ulaza. Na svaki od ulaza treba dovesti po jedan član u sumi, tj. jedan proizvod. S obzirom da svaki proizvod predstavlja kombinaciju 3 binarne promenljive, može se zapaziti da on odgovara jednoj od prekidačkih funkcija izlaza prethodno opisanog potpunog dekodera 3/8. Na primer, proizvodu $\bar{A}BC$ odgovara funkcija izlaza Y_3. To znači da se za realizaciju proizvoda u zadatoj funkciji može upotrebiti pomenuti dekoder na način prikazan na slici 5.9.



Slika 5.9 Realizacija logičke funkcije primenom dekodera 3/8

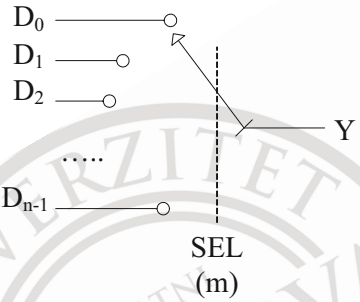
Kao što se vidi, logičke promenljive *A*, *B* i *C* od kojih zavisi funkcija predstavljaju ulaze dekodera, dok se vrednost same funkcije*Y* dobija na izlazu *ILI* kola. Kada se na ulaz dekodera dovede kombinacija koja odgovara nekom članu u funkciji, onda se na odgovarajućem izlazu generiše vrednost 1 koja se direktno prosleđuje na izlaz *ILI* kola, pa funkcija ima vrednost 1. Ako se na ulaz dekodera dovede kombinacija koja ne odgovara nijednom članu funkcije, onda se na odgovarajućem izlazu dekodera generiše vrednost 1, ali taj izlaz nije povezan sa *ILI* kolom, pa ne utiče na njegov izlaz. S obzirom da su svi ostali izlazi dekodera neaktivni, na izlazu *ILI* kola funkcija ima vrednost 0. Ovime je pokazano da data šema zaista realizuje zadatu logičku funkciju.

5.1.3 Multiplekseri

Multiplekseri su kombinacione mreže sa više ulaza i jednim izlazom koje obavljaju funkciju digitalnog višepoložajnog prekidača. Oni imaju dve vrste ulaza:

- *informacione ulaze*, na koje se dovode ulazni signali multipleksera
- *selekcione ulaze*, koji selektuju jedan od informacionih ulaza

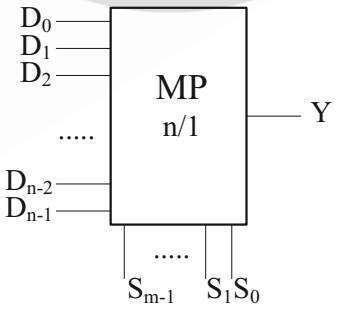
Multiplekser radi tako što samo jedan od binarnih signala dovedenih na informacione ulaze prosleđuje na izlaz. Izbor binarnog signala koji će biti prosleđen na izlaz zavisi od signala dovedenih na selekcione ulaze. Stoga se multiplekseri često nazivaju i *selektorima*. Ovakva funkcionalnost odgovara digitalnom višepoložajnom prekidaču prikazanom na slici 5.10.



Slika 5.10 Višepoložajni prekidač koji odgovara multiplekseru

U skladu sa binarnom kombinacijom dovedenom na selekcione ulaze, prekidač se postavlja u odgovarajući položaj i neposredno povezuje sa jednim od informacionih ulaza. Signal koji postoji na tom informacionom ulazu direktno se prosleđuje na izlaz.

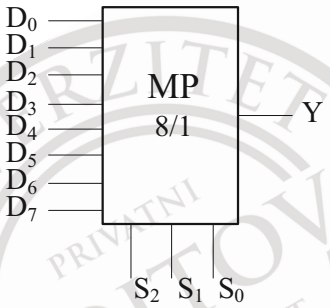
Grafički simbol multipleksera sa *n* informacionih ulaza, *m* selekcionih ulaza i jednim izlazom prikazan je na slici 5.11.



Slika 5.11 Multiplekser *n*/1

Broj informacionih ulaza multipleksera je povezan sa brojem njegovih selekcionih ulaza. Kod multipleksera koji ima m selekcionih ulaza, broj različitih binarnih kombinacija koje se na njima mogu generisati je 2^m. Pošto svaka kombinacija selektuje samo jedan informacioni ulaz, broj ovih ulaza takođe iznosi $n = 2^m$. Ovaj odnos se može sagledati i na drugi način. Ako multiplekser ima n informacionih ulaza, za njihovu selekciju potrebno je obezbediti isto toliko različitih binarnih kombinacija na selekcionim ulazima. Stoga je broj potrebnih selekcionih ulaza $m = \log_2 n$.

U nastavku će detaljno biti opisan multiplekser 8/1 koji ima 8 informacionih ulaza, 3 selekciona ulaza i 1 izlaz, kao što je to prikazano na slici 5.12.



Slika 5.12 Multiplekser 8/1

Neka su na informacione ulaze multipleksera dovedeni željeni signali. U datom trenutku, dovođenjem selekcionih signala, na selekcionim ulazima se formira binarna kombinacija koja predstavlja redni broj informacionog ulaza sa koga se binarna vrednost (0 ili 1) direktno prosleđuje na izlaz. Ovakav način funkcionisanja multipleksera može se opisati kombinacionom tablicom prikazanom na slici 5.13.

S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

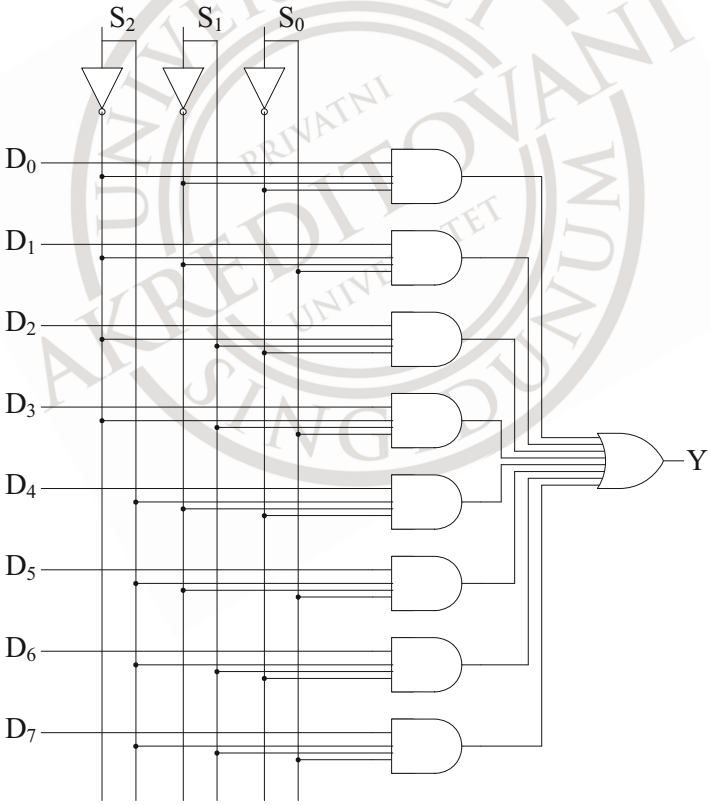
Slika 5.13 Kombinaciona tablica multipleksera 8/1

Leva strana tablice sadrži sve moguće kombinacije koje se mogu generisati na selekcionim ulazima ($2^3 = 8$ kombinacija). U desnom delu tablice data je funkcija izlaza koja zavisi, ne samo od selekcionih signala, već i od trenutnih vrednosti signala na informacionim ulazima.

Zavisnost izlaza od ulaza može se, na osnovu kombinacione tablice, predstaviti sledećom prekidačkom funkcijom:

$$Y = D_0 \overline{S_2}\,\overline{S_1}\,\overline{S_0} + D_1 \overline{S_2}\,\overline{S_1} S_0 + D_2 \overline{S_2} S_1 \overline{S_0} + \ldots + D_7 S_2 S_1 S_0$$

Kao što se vidi, za zadatu kombinaciju na selekcionim ulazima (na primer 001) izlaz dobija vrednost odgovarajućeg informacionog ulaza (za navedeni primer to je vrednost D_1). Na osnovu prikazane funkcije izlaza, multiplekser 8/1 može se realizovati pomoću prekidačke mreže predstavljene na slici 5.14.



Slika 5.14 Prekidačka mreža koja realizuje multiplekser 8/1

Multiplekser se može koristiti za realizaciju zadate logičke funkcije.

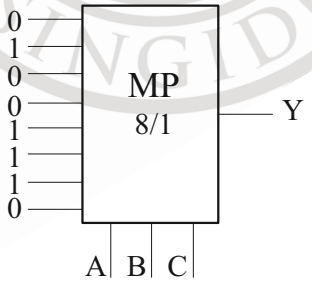
Neka je data funkcija

$$Y = \overline{A}\overline{B}C + A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C}.$$

Ova funkcija se može predstaviti kombinacionom tablicom datom na slici.

A	*B*	*C*	*Y*
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	1
1	1	1	0

Iz tablice se vidi da je vrednost funkcije *Y* jednaka 1 samo za sledeće kombinacije promenljivih *A*, *B* i *C*: 001, 100, 101 i 110. Ukoliko se usvoji da promenljive *A*, *B* i *C* predstavljaju selekcione signale multipleksera 8/1 (tip multipleksera je uslovljen brojem selekcionih signala kojih ima 3), a funkcija *Y* izlaz multipleksera, onda se zadata funkcija može realizovati tako što se na informacione ulaze koji odgovaraju navedenim kombinacijama dovede vrednost 1, a na sve ostale informacione ulaze vrednost 0 (videti sliku 5.15).



Slika 5.15 Realizacija logičke funkcije primenom multipleksera 8/1

Na ovaj način, svaki put kad se na selekcione ulaze dovede kombinacija koja odgovara nekom od proizvoda koji se javljaju u zadatoj funkciji, prekidač se povezuje na informacioni ulaz na kome je vrednost 1, pa se ona prosleđuje na izlaz. Ako se na selekcione ulaze dovede neka od preostalih kombinacija, prekidač se

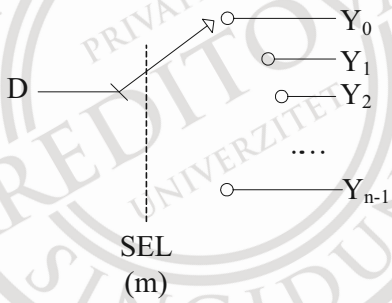
povezuje na informacioni ulaz na kome je vrednost 0, pa je i vrednost funkcije na izlazu multipleksera 0.

5.1.4 Demultiplekseri

Demultiplekseri su kombinacione mreže sa više ulaza i više izlaza čija je funkcija inverzna funkciji multipleksera. Oni takođe obavljaju funkciju digitalnog višepoložajnog prekidača, ali u obrnutom smeru. Imaju dve vrste ulaza:

- *informacioni ulaz*, na koji se dovodi ulazni signal
- *selekcione ulaze*, koji selektuju jedan od izlaza

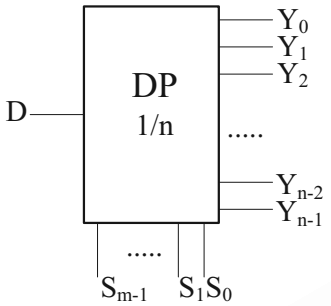
Demultiplekser radi tako što binarni signal doveden na informacioni ulaz prosleđuje na jedan od izlaza. Izbor izlaza na koji se prosleđuje ulazni signal zavisi od signala dovedenih na selekcione ulaze. Zbog toga se demultiplekser ponekad naziva i *distributorom*. Ovakva funkcionalnost odgovara digitalnom višepoložajnom prekidaču prikazanom na slici 5.16.



Slika 5.16 Višepoložajni prekidač koji odgovara demultiplekseru

U skladu sa binarnom kombinacijom dovedenom na selekcione ulaze, prekidač se postavlja u odgovarajući položaj i neposredno povezuje sa jednim od izlaza. Signal koji postoji na informacionom ulazu direktno se prosleđuje na taj izlaz.

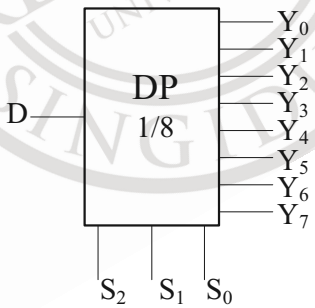
Grafički simbol demultipleksera sa jednim informacionim ulazom, *m* selekcionih ulaza i *n* izlaza prikazan je na slici 5.17.



Slika 5.17 Demultiplekser 1/*n*

Broj izlaza demultipleksera uslovljen je brojem njegovih selekcionih ulaza. Kod demultipleksera koji ima m selekcionih ulaza, broj različitih binarnih kombinacija koje se na njima mogu generisati je 2^m. Pošto svaka kombinacija selektuje samo jedan izlaz, broj izlaza takođe iznosi $n = 2^m$. Takođe se može reći i sledeće: ako demultiplekser ima n izlaza, za njihovu selekciju potrebno je obezbediti isto toliko različitih binarnih kombinacija na selekcionim ulazima. Stoga je broj potrebnih selekcionih ulaza $m = \log_2 n$.

Realizacija demultipleksera će biti objašnjena na primeru demultipleksera 1/8 koji ima 1 informacioni ulaz, 3 selekciona ulaza i 8 izlaza, kao što je to prikazano na slici 5.18.



Slika 5.18 Demultiplekser 1/8

Neka je na informacioni ulaz demultipleksera doveden željeni signal. U datom trenutku, dovođenjem selekcionih signala, na selekcionim ulazima se formira binarna kombinacija koja predstavlja redni broj izlaza na koji će biti prosleđen signal sa informacionog ulaza. Ovakav način funkcionisanja demultipleksera može se opisati kombinacionom tablicom prikazanom na slici 5.19.

S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	D
0	0	1	0	0	0	0	0	0	D	0
0	1	0	0	0	0	0	0	D	0	0
0	1	1	0	0	0	0	D	0	0	0
1	0	0	0	0	0	D	0	0	0	0
1	0	1	0	0	D	0	0	0	0	0
1	1	0	0	D	0	0	0	0	0	0
1	1	1	D	0	0	0	0	0	0	0

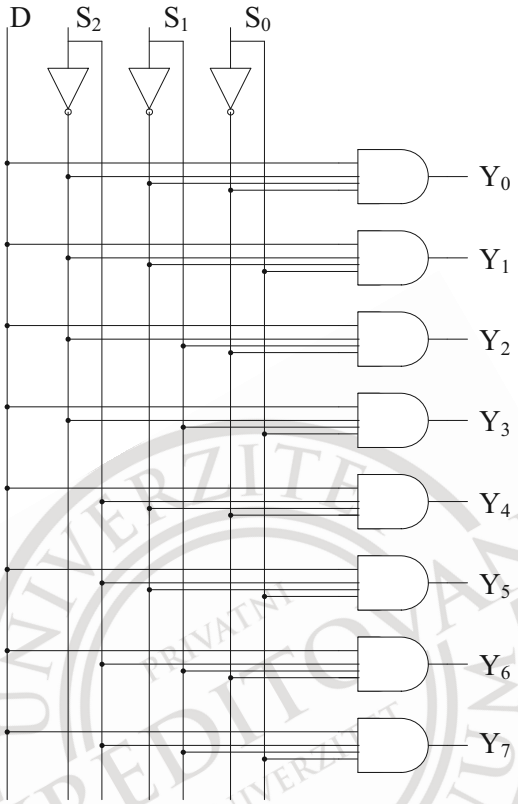
Slika 5.19 Kombinaciona tablica demultipleksera 1/8

Leva strana tablice sadrži sve moguće kombinacije koje se mogu generisati na selekcionim ulazima ($2^3 = 8$ kombinacija). U desnom delu tablice date su funkcije izlaza koje zavise, ne samo od selekcionih signala, već i od trenutne vrednosti signala na informacionom ulazu.

Na osnovu kombinacione tablice, zavisnost izlaza od ulaza može se predstaviti sledećim prekidačkim funkcijama:

$$Y_0 = D\overline{S_2}\,\overline{S_1}\,\overline{S_0} \qquad Y_4 = DS_2\,\overline{S_1}\,\overline{S_0}$$
$$Y_1 = D\overline{S_2}\,\overline{S_1}S_0 \qquad Y_5 = DS_2\,\overline{S_1}S_0$$
$$Y_2 = D\overline{S_2}S_1\,\overline{S_0} \qquad Y_6 = DS_2S_1\,\overline{S_0}$$
$$Y_3 = D\overline{S_2}S_1S_0 \qquad Y_7 = DS_2S_1S_0$$

Kao što se vidi, za zadatu kombinaciju na selekcionim ulazima (na primer 101) odgovarajući izlaz (za dati primer izlaz 5) dobija vrednost sa informacionog ulaza (D). Na osnovu prikazanih funkcija izlaza, demultiplekser 1/8 može se realizovati pomoću prekidačke mreže predstavljene na slici 5.20.



Slika 5.20 Prekidačka mreža koja realizuje demultiplekser 1/8

5.1.5 Sabirači

Sabirači su kombinacione mreže koje realizuju aritmetičku operaciju sabiranja dva jednobitna binarna broja. U zavisnosti od načina sabiranja, razlikuju se dve vrste sabirača:

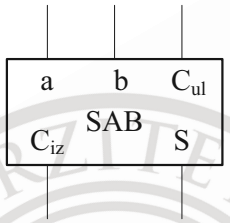
- polusabirači
- potpuni sabirači

Polusabirač predstavlja mrežu koja ima dva ulaza na koja se dovode ulazni signali (*a* i *b*) koje treba sabrati. Ulazni signali predstavljaju binarne cifre 0 ili 1. Mreža ima i dva binarna izlaza *S* i C_{iz}. Na izlazu *S* dobija se rezultat sabiranja, dok se na izlazu C_{iz} dobija prenos u stariji (viši) razred. Kao što se vidi, polusabirač nema mogućnost potpunog sabiranja zato što ne koristi prenos iz prethodnog

(nižeg) razreda. Stoga se on ne može primeniti za sabiranje višecifrenih binarnih brojeva.

Za razliku od polusabirača, potpuni sabirač uzima u obzir i prenos iz prethodnog razreda. On ima iste ulaze i izlaze kao polusabirač, samo što mu je dodat još jedan, treći ulaz C_{ul} na koji se dovodi binarni signal koji predstavlja prenos iz prethodnog razreda. Zahvaljujući ovom ulazu, potpuni sabirač može da se koristi za sabiranje višecifrenih binarnih brojeva.

Grafički simbol potpunog sabirača prikazan je na slici 5.21.



Slika 5.21 Potpuni sabirač

Postupak sabiranja dva jednobitna binarna broja može se predstaviti kombinacionom tablicom prikazanom na slici 5.22.

Leva strana tablice sadrži sve moguće kombinacije koje se mogu dovesti na ulaze potpunog sabirača, dok su na desnoj strani date vrednosti koje se pojavljuju na izlazima sabirača za svaku ulaznu kombinaciju.

C_{ul}	a	b	C_{iz}	S
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

Slika 5.22 Kombinaciona tablica potpunog sabirača

Zavisnost izlaza od ulaza može se predstaviti sledećim funkcijama izlaza:

$$S = a \oplus b \oplus C_{ul}$$
$$C_{iz} = (a \oplus b)C_{ul} + ab$$

Ove prekidačke funkcije izvedene su na osnovu kombinacione tablice i izraza kojima se opisuje funkcija izlaza logičkog ekskluzivno *ILI* kola (videti kombinacionu tablicu za ovu operaciju operaciju, poglavlje 2) koji glase:

$$a \oplus b = \bar{a}b + a\bar{b} \qquad \text{ili} \qquad \overline{a \oplus b} = \bar{a}\bar{b} + ab$$

Primenom ovih jednačina, funkcije izlaza potpunog sabirača mogu se dobiti na sledeći način:

$$S = \overline{C_{ul}}\,\bar{a}b + \overline{C_{ul}}\,a\bar{b} + C_{ul}\,\bar{a}\bar{b} + C_{ul}\,ab = (\bar{a}b + a\bar{b})\overline{C_{ul}} + (\bar{a}\bar{b} + ab)C_{ul}$$

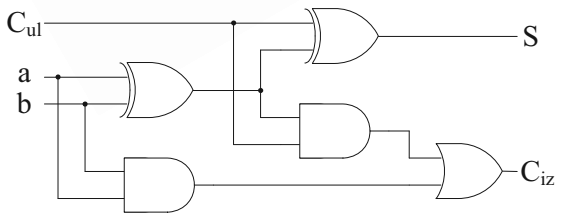
$$S = (a \oplus b)\overline{C_{ul}} + (\overline{a \oplus b})C_{ul} = a \oplus b \oplus C_{ul}$$

$$C_{iz} = \overline{C_{ul}}\,ab + C_{ul}\,\bar{a}b + C_{ul}\,a\bar{b} + C_{ul}\,ab = (\bar{a}b + a\bar{b})C_{ul} + ab(\overline{C_{ul}} + C_{ul})$$

$$C_{iz} = (a \oplus b)C_{ul} + ab$$

Na osnovu funkcija izlaza, potpuni sabirač se može realizovati pomoću prekidačke mreže prikazane na slici 5.23.

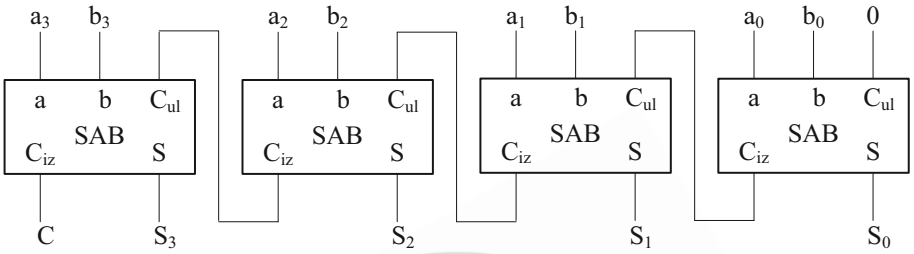
Potpuni sabirači se najčešće koriste za sabiranje višecifrenih binarnih brojeva. Pošto svaki potpuni sabirač vrši sabiranje samo na nivou jedne binarne cifre (jednog bita), sabiranje višecifrenih brojeva se ostvaruje kaskadnim povezivanjem onoliko potpunih sabirača koliko ima cifara u binarnim brojevima koje treba sabrati. Tako svaki sabirač obavlja sabiranje samo na nivou razreda u kome se nalazi. Kaskadno povezivanje potpunih sabirača ostvaruje se tako što se izlaz za prenos u viši razred (C_{iz}) jednog sabirača vezuje na ulaz za prenos iz nižeg razreda (C_{ul}) sabirača koji se nalazi u sledećem razredu.



Slika 5.23 Prekidačka mreža koja realizuje potpuni sabirač

Pomoću potpunih sabirača mogu se sabirati kako neoznačeni brojevi, tako i označeni brojevi predstavljeni u komplementu dvojke.

U nastavku će biti opisan četvorobitni sabirač koji omogućava sabiranje dva četvorocifrena binarna broja $A = a_3a_2a_1a_0$ i $B = b_3b_2b_1b_0$. Za realizaciju ovog sabirača potrebno je kaskadno povezati četiri potpuna sabirača (za svaki razred po jedan) kao što je to prikazano na slici 5.24.



Slika 5.24 Realizacija četvorobitnog sabirača

Kao što se vidi, na ulaz C_{ul} sabirača u najnižem razredu dovedena je vrednost 0 zato što nema prenosa iz nižeg razreda. U svim ostalim razredima, sabirači su povezani tako što je na njihov ulaz C_{ul} doveden signal sa izlaza C_{iz} sabirača u prethodnom razredu, dok je njihov izlaz C_{iz} vezan za ulaz sabirača C_{ul} u narednom razredu. Na taj način je obezbeđeno ispravno sabiranje koje uključuje prenose između razreda. Izuzetak predstavlja izlaz C_{iz} sabirača u najvišem razredu koji nije nigde povezan, već se uključuje u rezultat.

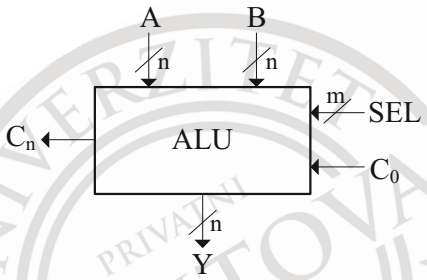
Postupak sabiranja na primeru četvorocifrenih brojeva $A = 1011$ i $B = 101$, čije su cifre u vidu binarnih signala dovedene na ulaze sabirača, odvija se na sledeći način:

- sabirač za najniži razred (razred 0) izvrši binarno sabiranje vrednosti dovedenih na njegove ulaze, tj. $a_0+b_0+0 = 1+1+0 = 10_{(2)}$; pošto rezultat ima dve cifre, cifra 0 predstavlja rezultat sabiranja u nultom razredu i signal koji odgovara vrednosti 0 pojavljuje se na izlazu S_0, a cifra 1 prelazi u viši razred tako što se odgovarajući signal pojavljuje na izlazu C_{iz} i prosleđuje kao prenos do sabirača u višem razredu
- sabirač koji odgovara prvom razredu radi na isti način, tj. nalazi zbir $a_1+b_1+C_{ul} = 1+0+1 = 10_{(2)}$; na njegovim izlazima se pojavljuju vrednosti $S_1 = 0$ i $C_{iz} = 1$
- sabirač koji odgovara drugom razredu sabira $a_2+b_2+C_{ul} = 0+1+1 = 10_{(2)}$; na njegovim izlazima se pojavljuju vrednosti $S_2 = 0$ i $C_{iz} = 1$
- sabirač koji odgovara najvišem razredu sabira $a_3+b_3+C_{ul} = 1+1+1 = 11_{(2)}$; na njegovim izlazima se pojavljuju vrednosti $S_3 = 1$ i $C_{iz} = 1$; konačan rezultat sabiranja je binarni broj 11000 koji se dobija na izlazima C_{iz} i S_3 sabirača u najvišem razredu i izlazima S_2, S_1 i S_0 ostalih sabirača

5.1.6 Aritmetičko-logičke jedinice

Aritmetičko-logička jedinica (*ALU – Arithmetic Logic Unit*) je višefunkcionalna kombinaciona mreža koja realizuje neki skup aritmetičkih operacija i neki skup logičkih operacija nad *n*-bitskim binarnim brojevima. Zbog svoje funkcionalnosti, ona predstavlja centralni deo svakog procesora i njene karakteristike direktno utiču na mogućnosti procesora. Česta upotreba uslovila je da se *ALU* obično izrađuje kao integrisana komponenta.

Grafički simbol aritmetičko-logičke jedinice prikazan je na slici 5.25.



Slika 5.25 Aritmetičko-logička jedinica

ALU ima sledeće ulaze:

A i *B* – *n*-bitski binarni brojevi nad kojima se obavlja operacija
C_0 – ulaz bitan za pojedine operacije
SEL – *m*-bitski upravljački ulaz

Izlazi *ALU* su:

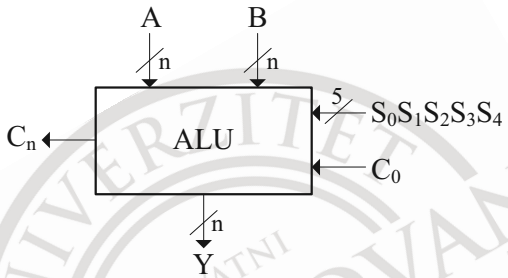
C_n – izlaz bitan za pojedine operacije
Y – *n*-bitski binarni broj koji predstavlja rezultat operacije

U datom trenutku, *ALU* može da obavlja samo jednu artimetičku ili logičku (može i kombinovanu) operaciju. Izbor te operacije (koja se vrši nad binarnim brojevima dovedenim na ulaze *A* i *B*) uslovljen je signalom koji u tom trenutku postoji na upravljačkom ulazu *SEL*. Pošto se radi o *m*-bitskom ulazu, na njega je moguće dovesti 2^m različitih signala, što znači da je broj mogućih operacija koje *ALU* može da izvrši takođe 2^m. Ukoliko selektovana operacija zahteva početnu vrednost ili neki parametar, oni se dovode na ulaz C_0. Rezultat operacije se dobija

na izlazu Y, dok se izlaz C_n koristi po potrebi u slučaju da selektovana operacija to zahteva.

Može se zapaziti da se upravo opisani način rada *ALU* odnosi samo na dovođenje operanada, izbor operacije i dobijanje rezultata, a ne i na realizaciju samih operacija. Mreže koje realizuju operacije nalaze se u bloku koji predstavlja *ALU* u grafičkom simbolu i o njima će biti reči malo kasnije.

Rad sa *ALU* biće objašnjen na primeru aritmetičko-logičke jedinice sa 5-bitnim upravljačkim ulazom *SEL* ($S_4S_3S_2S_1S_0$), kao što je prikazano na slici 5.26.



Slika 5.26 Aritmetičko-logička jedinica sa 5 selekcionih signala

Ova *ALU* može da izvrši $2^5 = 32$ različite operacije. Uobičajeno je da su u skupu operacija podjednako zastupljene aritmetičke i logičke operacije, pa se može usvojiti da *ALU* iz primera realizuje 16 aritmetičkih i 16 logičkih operacija navedenih u tabeli 5.1.

U spisku operacija treba voditi računa o oznakama operacija, jer se iste oznake (na primer „+“) koriste i za aritmetičke („+“ je oznaka za sabiranje) i za logičke operacije („+“ je oznaka za logičku *ILI* operaciju). Zato je u tabeli usvojeno da oznake *plus* i *minus* predstavljaju aritmetičke operacije sabiranja i oduzimanja.

Upravljački signal S_4 obično se koristi za selekciju tipa operacije, tj. određuje da li će se izvršavati neka od aritmetičkih ili neka od logičkih operacija. U primeru, ako je $S_4 = 1$ obavljaju se logičke operacije, a ako je $S_4 = 0$, aritmetičke (videti tabelu). Nakon izbora tipa operacije, potrebno je selektovati konkretnu operaciju iz odabranog skupa (skupa aritmetičkih ili skupa logičkih operacija). To se obavlja pomoću signala $S_3S_2S_1S_0$. Ova četiri signala mogu da formiraju $2^4 = 16$ različitih kombinacija što je dovoljno za selekciju operacija u odabranom skupu.

S_3	S_2	S_1	S_0	$S_4 = 1$	$S_4 = 0$
0	0	0	0	$Y = \overline{A}$	$Y = A$ *plus* C_0
0	0	0	1	$Y = \overline{A + B}$	$Y = (A+B)$ *plus* C_0
0	0	1	0	$Y = \overline{A}B$	$Y = (A+\overline{B})$ *plus* C_0
0	0	1	1	$Y = 0000$	$Y = 0$ - C_0
0	1	0	0	$Y = \overline{AB}$	$Y = A$ *plus* $(A\overline{B})$ *plus* C_0
0	1	0	1	$Y = \overline{B}$	$Y = (A+B)$ *plus* $(A\overline{B})$ *plus* C_0
0	1	1	0	$Y = A \oplus B$	$Y = A$ *minus* B *minus* $\overline{C}_0$
0	1	1	1	$Y = A\overline{B}$	$Y = (A\overline{B})$ *minus* $\overline{C}_0$
1	0	0	0	$Y = \overline{A} + B$	$Y = A$ *plus* (AB) *plus* C_0
1	0	0	1	$Y = \overline{A \oplus B}$	$Y = A$ *plus* B *plus* C_0
1	0	1	0	$Y = B$	$Y = (A+\overline{B})$ *plus* (AB) *plus* C_0
1	0	1	1	$Y = AB$	$Y = (AB)$ *minus* C_0
1	1	0	0	$Y = 1111$	$Y = A$ *plus* A *plus* C_0
1	1	0	1	$Y = A + \overline{B}$	$Y = (A+B)$ *plus* A *plus* C_0
1	1	1	0	$Y = A + B$	$Y = (A+\overline{B})$ *plus* A *plus* C_0
1	1	1	1	$Y = A$	$Y = A$ *minus* C_0

Tabela 5.1 Operacije *ALU* sa pet selekcionih ulaza

Neka su na ulaze *ALU* dovedeni binarni signali $A = 1001$ i $B = 1100$, kao i upravljački signal *SEL* = 10001. Pošto je prethodno opisana logika implementirana u okviru bloka *ALU*, mreža će na osnovu signala *SEL* zaključiti da treba obaviti logičku operaciju $\overline{A + B}$ (iz $S_4 = 1$ sledi da je operacija logička, a kombinacija 0001 određuje samu operaciju). Kao rezultat, na izlazu *Y* će se pojaviti vrednost 0010.

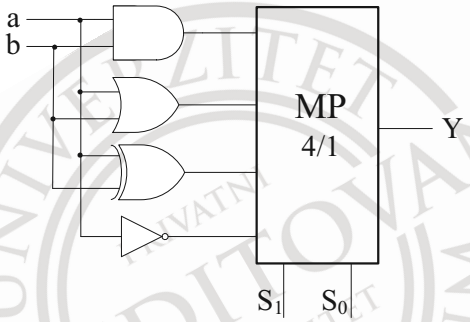
U nastavku će biti reči o realizaciji samih operacija u okviru bloka koji predstavlja *ALU*. S obzirom da realizacija zavisi od broja i vrste operacija koje *ALU* može da obavi, ona može biti vrlo složena. Međutim, princip realizacije je jednostavan i biće objašnjen na primeru *ALU* koja obavlja osnovne logičke operacije (*I*, *ILI*, *EKSILI* i *NE*) nad jednobitnim binarnim brojevima *a* i *b*.

Za realizaciju *ALU* sa četiri logičke operacije dovoljno je koristiti 2-bitni upravljački signal *SEL* (S_1S_0). Neka su vrednosti ovog signala koje odgovaraju pojedinim operacijama date u tabeli 5.2.

S_1	S_0	Y
0	0	ab
0	1	a+b
1	0	$a \oplus b$
1	1	$\bar{a}$

Tabela 5.2 Skup osnovnih operacija *ALU*

Ukoliko se signali S_1 i S_0 iskoriste kao selekcioni signali multipleksera 4/1, *ALU* se može realizovati tako što se izlaz multiplesera proglasi izlazom *ALU Y*, a na svaki od ulaza multipleksera (prema tabeli 5.2) dovede izlaz prekidačke mreže koja realizuje konkretnu operaciju (videti sliku 5.27).



Slika 5.27 Realizacija *ALU* koja obavlja osnovne logičke operacije

U primeru, prekidačke mreže koje realizuju operacije su vrlo jednostavne jer se sastoje samo od jednog logičkog kola koje odgovara operaciji. U slučaju većeg broja složenijih operacija i prekidačke mreže su složenije. Ovakvim načinom realizacije, kada se na ulaze dovedu signali koji odgovaraju binarnim brojevima nad kojima treba obaviti operaciju i upravljački signali S_1 i S_0, na ulazima multipleksera dobijaju se rezultati svih operacija. Međutim, na izlaz multipleksera (tj. izlaz *ALU*) prosleđuje se rezultat samo one operacije koja je selektovana (sa odgovarajućeg ulaza multipleksera).

5.2. Standardni sekvencijalni moduli

Standardni sekvencijalni moduli su moduli koji, osim logičkih, sadrže i memorijske elemente. U ovom poglavlju biće predstavljena tri važna modula ovog tipa: registri, brojači i memorije. Za svaki modul biće opisani njegova funkcionalnost, osnovne karakteristike i situacije u kojima se koristi. Za registre i brojače takođe će biće data i njihova unutrašnja realizacija.

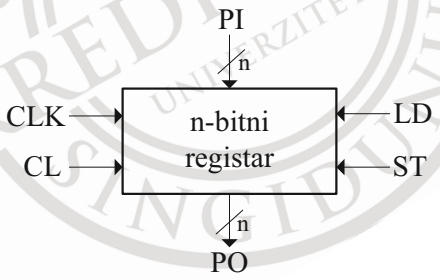
5.2.1 Registri

Registri su sekvencijalne mreže koje se u računarima i drugim digitalnim sistemima koriste za pamćenje binarnih podataka, tj. binarnih reči. Registar za pamćenje binarne reči od *n* bita može se posmatrati kao sekvencijalna mreža sa *n* razreda. Veličina registra zavisi od dužine binarne reči (broja bitova u njoj) koja se u njemu čuva. Najčešće se koriste 8-bitni i 16-bitni registri. Binarne reči se po potrebi mogu upisivati u registar, ili se iz njega čitati. U zavisnosti od načina upisa i čitanja, razlikuju se dve vrste registara:

- registri sa *paralelnim* upisom i čitanjem
- registri sa *serijskim* upisom i čitanjem

Paralelni registri

Kod paralelnih registara (registara sa paralelnim upisom i čitanjem), svi biti binarne reči upisuju se u registar istovremeno, tj. u jednom taktu, a takođe se i svi biti sačuvane reči čitaju istovremeno, tj. u jednom taktu. Grafički simbol paralelnog registra dat je na slici 5.28.



Slika 5.28 Paralelni registar

PI (*Parallel Inputs*) predstavlja informacione paralelne ulaze na koje se dovodi *n*-bitska binarna reč koju treba sačuvati u registru. *PO* (*Parallel Outputs*) su informacioni paralelni izlazi na kojima se pojavljuje *n*-bitska binarna reč koja je pročitana iz registra. Upravljački ulaz *LD* (*Load*) omogućava upis sadržaja sa *PI* u registar. S obzirom da se često javlja potreba za brisanjem sadržaja registra, tj. postavljanjem *n*-bitske binarne reči koja se sastoji samo od nula u registar, izdvojen je poseban ulaz *CL* (*Clear*) koji to omogućava. Osim ovog asinhronog ulaza, ponekad se koristi i ulaz *ST* (*Set*) koji podržava asinhroni upis jedinice u sve razrede registra. Za rad registra neophodan je i takt koji obezbeđuje sinhronizaciju

prilikom obavljanja operacija upisa i čitanja. Takt se dovodi na ulaz *CLK* (*Clock*) i samo njegova aktivna vrednost može da promeni stanje registra.

S obzirom da su kod paralelnog registra svi razredi međusobno jednaki, za njegovu realizaciju dovoljno je razmotriti samo jedan razred. Neka se u *n*-bitski registar $A = A_{n-1}A_{n-2}A_{n-3}...A_0$ upisuje binarna reč $I = I_{n-1}I_{n-2}I_{n-3}...I_0$. Funkcija prelaza *i*-tog razreda registra *A* prilikom upisa data je u kombinacionoj tablici na slici 5.29. Kao što se vidi, usvojeno je da se stanje *i*-tog bita registra, A_i, može menjati samo ako je signal $LD = 1$ (aktivna vrednost *LD* signala) i tada je novo stanje određeno bitom koji se upisuje, tj. $A_i(t+1) = I_i$. Ako je $LD = 0$ (neaktivna vrednost *LD* signala), stanje registra se ne menja pa je $A_i(t+1) = A_i$.

LD	I_i	A_i	$A_i(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Slika 5.29 Tablica prelaza *i*-tog razreda paralelnog registra pri upisu

Na osnovu tablice prelaza *i*-tog razreda mogu se formirati kombinacione tablice i odrediti funkcije pobude za različite vrste flip-flop-ova pomoću kojih paralelni registar može biti realizovan. U nastavku će biti opisana realizacija paralelnog registra pomoću *D* FF-ova. Sličnom analizom se mogu dobiti i realizacije pomoću *RS*, *T* i *JK* FF-ova.

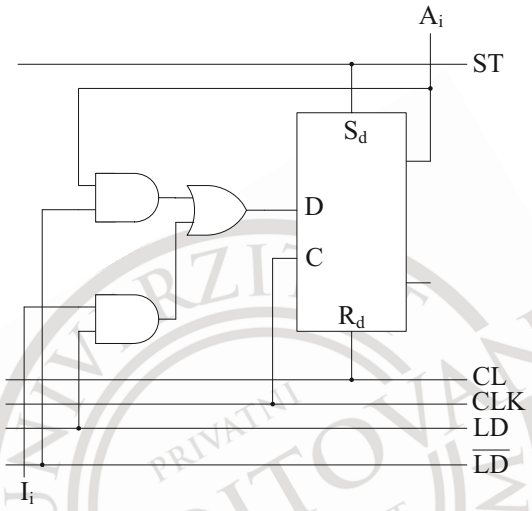
Paralelni registar se pomoću taktovanih *D* FF-ova može realizovati na različite načine. Jedan od njih je primenom sinhronih *D* FF-ova kod kojih je aktivna vrednost ulaznih signala 1. Pošto $A_i(t+1)$ predstavlja novo stanje *i*-tog razreda paralelnog registra, poslednja kolona u tabeli prelaza odgovara izlazu *D* FF-a kojim se realizuje *i*-ti razred (signal D_i). Prevođenjem tablice prelaza u algebarski oblik, dobija se da je

$$D_i = A_i \cdot \overline{I_i} \cdot \overline{LD} + A_i \cdot I_i \cdot \overline{LD} + \overline{A_i} \cdot I_i \cdot LD + A_i \cdot I_i \cdot LD = I_i \cdot LD + A_i \cdot \overline{LD}$$

Pored paralelnog upisa binarne reči u registar pomoću taktovanih ulaza *D* FF-ova, često se uvode još dva ulaza FF-ova koja služe za asinhroni upis vrednosti 0

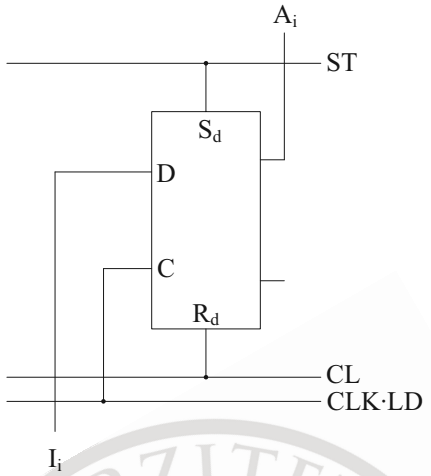
(ulaz R_d) ili vrednosti 1 (ulaz S_d) u sve razrede registra. Upravljački signali za ove operacije su *CL* i *ST*, respektivno.

Na slici 5.30 prikazana je strukturna šema kojom se realizuje upis u *i*-ti razred paralelnog registra. Ona sadrži tri logička kola, dva *I* i jedno *ILI*, povezana tako da generišu ulazni signal taktovanog *D* FF-a prema poznatom izrazu za D_i.



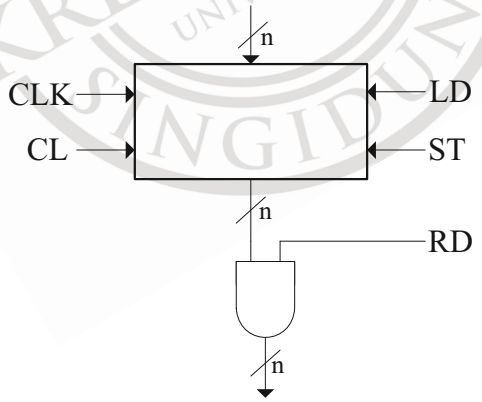
Slika 5.30 Upis u *i*-ti razred paralelnog registra sa *D* FF-ovima

Strukturna šema na slici može se pojednostaviti korišćenjem signala takta za upravljanje.To se postiže dovođenjem aktivne vrednosti signala takta na ulaze *C* svih FF-ova u registru samo kada i signal *LD* ima aktivnu vrednost. Tada funkcija prelaza *i*-tog razreda registra postaje $A_i(t+1) = D_i = I_i$. Za neaktivnu vrednost signala *LD*, funkcija prelaza je $A_i(t+1) = A_i$. Na osnovu ovoga, može se konstruisati strukturna šema *i*-tog razreda paralelnog registra sa *D* FF-ovima data na slici 5.31.



Slika 5.31 Upis u *i*-ti razred paralelnog registra sa *D* FF-ovima sa upravljanjem pomoću signala takta

Iako je sadržaj paralelnog registra uvek dostupan na izlazima *PO*, često je potrebno i njime upravljati. Za razliku od operacije upisa, operacija čitanja ne zavisi od vrste FF-ova pomoću kojih je registar realizovan. Upravljanje čitanjem sadržaja iz registra najjednostavnije se može ostvariti pomoću logičkog I elementa kao što je prikazano na slici 5.32.



Slika 5.32 Čitanje iz paralelnog registra pomoću I elementa

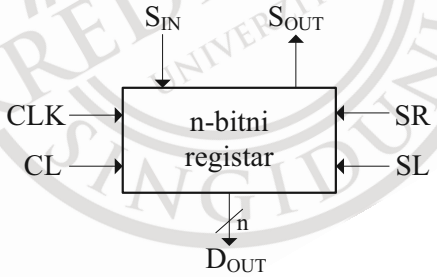
Upravljački signal za operaciju čitanja je *RD* (*Read*). On se dovodi na jedan od dva ulaza svih *n* I elementa.

Serijski registri

Serijski ili pomerački registar (registar sa serijskim upisom i čitanjem) omogućava pomeranje binarne reči koja se u njemu nalazi za jedno mesto od ulaza ka izlazu. Zavisno od načina pomeranja, postoje različite vrste pomeračkih registara. Kod registara sa pomeranjem udesno, bit iz najnižeg razreda se briše, a svi ostali biti se pomeraju u jedan razred niže. U najviši razred, koji ostaje prazan, upisuje se novi bit sa serijskog ulaza. Na sličan način rade i serijski registri sa pomeranjem ulevo. Kod njih se bit najveće težine briše, ostali biti se pomeraju za jedno mesto ulevo, a na mesto bita najmanje težine upisuje se novi bit sa ulaza. Oba pomenuta registra se mogu realizovati i tako da operacija pomeranja predstavlja rotaciju, tj. da se bit koji je ranije brisan upisuje na mesto novog bita. Osim opisane, pomerački registri mogu imati i dodatnu funkcionalnost o kojoj ovde neće biti reči, kao na primer, pomeranje u oba smera (bidirekcioni registri), paralelni upis binarne reči i sl.

Pomerački registri se koriste u slučajevima kada je potrebno omogućiti serijski prijem ili slanje podataka (bit po bit). U praksi postoji znatno veća potreba za korišćenjem paralelnih nego pomeračkih registara, ali i pomerački registri imaju značajnu primenu.

Grafički simbol pomeračkog registra prikazan je na slici 5.33.



Slika 5.33 Serijski registar

S_{OUT} (*Serial Output*) je jednobitni serijski informacioni izlaz na kome se pojavljuje bit koji je pročitan i izbrisan iz registra prilikom pomeranja sadržaja. S_{IN} (*Serial Input*) predstavlja informacioni jednobitni serijski ulaz na koji se dovodi bit koji treba dodati u registar. Informacioni izlaz D_{OUT} (*Data Output*) omogućava čitanje trenutnog sadržaja registra, tj. *n*-bitske reči koja se u registru nalazi u tom trenutku. I kod pomeračkog, kao i kod paralelnog registra, postoje ulazi *CL* i *CLK* koji imaju ranije opisanu ulogu. Serijski registar takođe ima i ulaze *SR* (*Shift Right*) i *SL* (*Shift Left*) na koje se dovode signali za pomeranje sadržaja za jedno mesto udesno, odnosno ulevo.

Različite vrste pomeračkih registara realizuju se različitim strukturnim šemama. U nastavku je opisana realizacija serijskog registra sa pomeranjem za jedan razred udesno pomoću *D* FF-ova. Usvojeno je da je 1 aktivna vrednost signala *SR*. Funkcija prelaza *i*-tog razreda razmatranog pomeračkog registra označenog sa $A = A_{n-1}A_{n-2}A_{n-3}...A_0$ prikazana je na slici 5.34. Postojeće stanje *i*-tog bita registra se ne menja ako je signal *SR* neaktivan i tada je $A_i(t+1) = A_i$. Registar prelazi u sledeće stanje tek pošto signal *SR* postane aktivan i tada novom stanju odgovara vrednost narednog razreda registra, tj. važi $A_i(t+1) = A_{i+1}$.

SR	A_{i+1}	A_i	$A_i(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Slika 5.34 Tablica prelaza *i*-tog razreda serijskog registra sa pomeranjem udesno

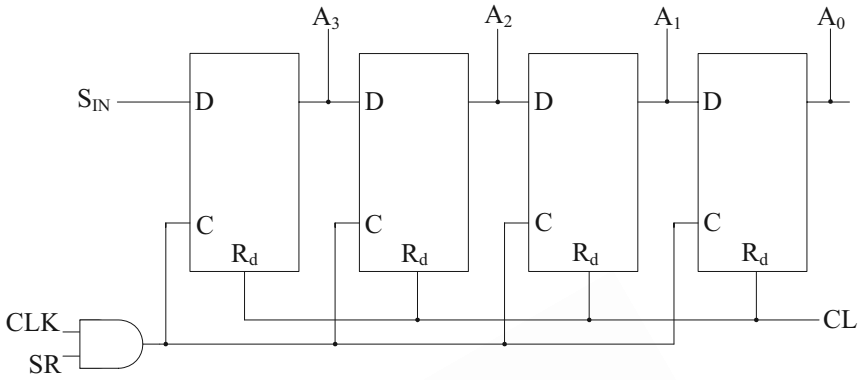
Kao što se vidi, data tablica prelaza je vrlo slična tablici prelaza *i*-tog razreda paralelnog registra pri operaciji upisa. Stoga je dovoljno u formulama zameniti *LD* i I_i sa *SR* i A_{i+1}. Tako se dobija da je

$$A_i(t+1) = A_{i+1} \cdot SR + A_i \cdot \overline{SR}$$

osim (*n*-1) -og razreda za koga važi

$$A_{n-1}(t+1) = S_{IN} \cdot SR + A_{n-1} \cdot \overline{SR}$$

Na slici 5.35 prikazana je strukturna šema koja realizuje 4-bitni pomerački registar sa taktovanim *D* FF-ovima i upravljanjem pomoću signala takta.



Slika 5.35 Pomerački registar sa četiri razreda

5.2.2 Brojači

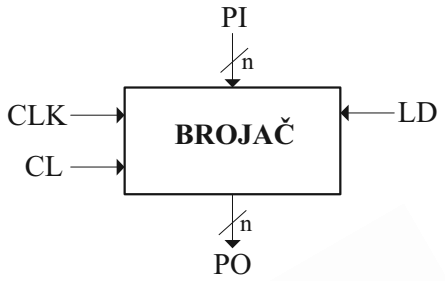
Brojači su sekvencijalne mreže koje se u digitalnim sistemima i uređajima koriste za brojanje različitih događaja i merenje vremenskih intervala. Oni rade tako što broje promene nekog signala dovedenog na njihov ulaz i na izlazu generišu binarni broj koji odgovara broju promena učinjenih do posmatranog trenutka. Obično je reč je o taktovanim mrežama, što znači da se stanje na njihovom izlazu menja samo po nailasku takta, tj. promene. U zavisnosti od načina brojanja, razlikuju se:

- *inkrementirajući* brojači koji broje unapred, tj. po rastućem redosledu brojanja
- *dekrementirajući* brojači koji broje unazad, tj. po opadajućem redosledu brojanja

Broj različitih binarnih brojeva koje brojač može da generiše na svom izlazu naziva se *moduo* brojača. Inkrementirajući brojač po modulu *m* radi tako što, u skladu sa taktom, generiše binarne brojeve od 0 do *m*-1 (ukupno *m* brojeva), a zatim se resetuje i ponovo počinje da broji od 0. Dekrementirajući brojač po modulu *m* generiše binarne brojeve od *m*-1 do 0, a zatim se vraća u stanje *m*-1 i nastavlja sa radom.

Brojači se mogu realizovati na mnogo različitih načina. Na primer, kao inkrementirajući, dekrementirajući, brojači sa korakom 1, 2, 4 ili 8, kombinovani inkrementirajući/dekrementirajući, brojači sa više operacija (paralelni upis, taktovanjo brisanje i sl.). Za njihovu realizaciju mogu se koristiti i različiti memorijski elementi (*RS* ili *JK* flip-flop-ovi). Kaskadnim povezivanjem može se od brojača po jednom modulu dobiti brojač po drugom modulu. U ovom poglavlju

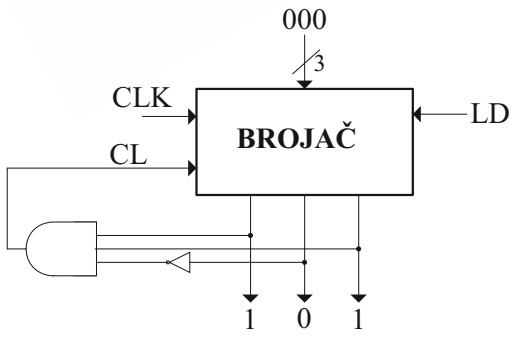
biće prikazan samo jedan tip inkrementirajućeg brojača čiji je grafički simbol prikazan na slici 5.36.



Slika 5.36 Brojač

Na ulaz *PI* (*Parallel Inputs*) dovodi se *n*-bitska binarna reč koja predstavlja početnu vrednost od koje brojač počinje da broji. Upravljački ulaz *LD* (*Load*) omogućava upis početnog stanja sa ulaza *PI* u brojač. Na izlazu *PO* (*Parallel Outputs*) generiše se *n*-bitska binarna reč koja predstavlja trenutno stanje brojača. Za resetovanje, tj. brisanje sadržaja brojača postavljanjem *n*-bitske binarne reči koja se sastoji samo od nula u brojač, izdvojen je poseban ulaz *CL* (*Clear*). Za rad brojača neophodan je i takt koji omogućava promenu stanja brojača i dovodi se na ulaz *CLK* (*Clock*). Ovde treba uočiti da kod ovog brojača ne postoji mogućnost eksplicitnog zadavanja tipa brojača (inkrementirajući ili dekrementirajući), jer je taj izbor već ugrađen u blok koji predstavlja unutrašnjost brojača.

Brojač predstavljen grafičkim simbolom broji po modulu 2^n, što je uslovljeno dužinom binarne reči koja može da se smesti u brojač (n). Međutim, često se javlja potreba za brojačima čiji moduo nije stepen dvojke (na primer, brojač po modulu 6, 10 i sl.). Da bi se realizovao brojač po željenom modulu, potrebno je dodati eksternu (spoljašnju) logiku koja bi to omogućila. Uloga eksterne logike će biti objašnjena na primeru brojača po modulu 6, čija je realizacija data na slici 5.37.



Slika 5.37 Realizacija brojača po modulu 6

Brojač po modulu 6 na svom izlazu generiše binarne brojeve od 0 do 5, tj. od $000_{(2)}$ do $101_{(2)}$. Stoga je dovoljno koristiti 3-bitski brojač na čijem se ulazu *PI* i izlazu *PO* pojavljuju 3-bitske binarne vrednosti. Pošto početna vrednost nije eksplicitno zadata, može se usvojiti da brojač počinje da broji od 0, pa je na ulaz *PI* dovedena vrednost 000. Ova vrednost se upisuje u brojač aktiviranjem ulaza *LD*. Sa dovođenjem signala takta na ulaz *CLK*, brojač počinje da broji po rastućem redosledu. Pri svakom narednom impulsu takta, vrednost brojača (na izlazu *PO*) se uvećava za 1. U trenutku kada brojač stigne do broja 5, potrebno ga je vratiti na 0, što se može postići brisanjem njegovog sadržaja, tj. aktiviranjem ulaza *CL*. Signal za aktiviranje ulaza *CL* generiše se od strane eksterne logike. Naime, ova logika je napravljena tako da registruje trenutak kada se na linijama izlaza *PO* pojavi kombinacija koja odgovara najvećem broju do kojeg brojač može da stigne (moduo umanjen za 1), što je u ovom primeru 101. U tom trenutku, logičko I kolo na svom izlazu generiše vrednost 1 koja se prosleđuje na ulaz *CL* i brojač se resetuje. Da bi logičko I kolo proizvelo ovaj signal, na njegove ulaze su direktno dovedeni signali sa linija *PO* izlaza na kojima kombinacija ima vrednost 1, a preko invertora sa linija na kojima kombinacija ima vrednost 0.

U nastavku će biti opisana unutrašnja realizacija inkrementirajućeg brojača sa taktovanim *RS* FF-ovima koji broji po modulu 2^n. Aktivna vrednost ulaznih signala FF-ova je 1. Neka je stanje brojača označeno sa $A = A_{n-1}A_{n-2}A_{n-3}...A_0$. Naredno stanje je definisano izrazom $A(t+1) = A+1$, tj. zbirom tekućeg binarnog stanja A i vrednosti 1. Tablica prelaza za i-ti razred ovog brojača data je na slici 5.38.

A_i	C_i	$A_i(t+1)$	C_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Slika 5.38 Tablica prelaza i-tog razreda inkrementirajućeg brojača

U tablici prelaza sa C_i je označen prenos iz nižeg razreda u i-ti, a sa C_{i+1} prenos iz i-tog u viši razred. Prevođenjem tablice prelaza u algebarski oblik, dobija se da je

$$A_i(t+1) = \overline{A_i} \cdot C_i + A_i \cdot \overline{C_i}$$

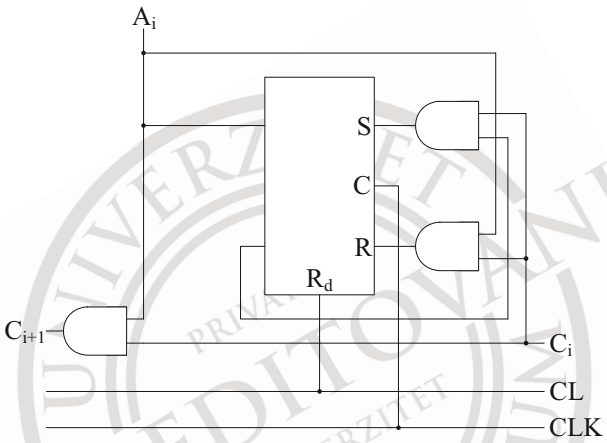
$$C_{i+1} = A_i \cdot C_i$$

Na osnovu tablice prelaza mogu se dobiti funkcije pobude *RS* FF-ova koji se koriste u realizaciji brojača u obliku

$$R_i = A_i \cdot C_i$$

$$S_i = \overline{A_i} \cdot C_i$$

Strukturna šema *i*-tog razreda razmatranog brojača, generisana u skladu sa datim jednačinama, prikazana je na slici 5.39.



Slika 5.39 Razred *i* inkrementirajućeg brojača sa taktovanim *RS* FF-ovima

5.2.3 Memorije

Memorije su komponente koje se u digitalnim sistemima koriste za pamćenje velikog broja binarnih reči. Idealna memorija bi trebalo da ima sledeće karakteristike:

- visoku gustinu pakovanja koja direktno utiče na kapacitet memorije
- trajno čuvanje podataka
- kratko vreme upisa i čitanja podataka, što bitno utiče na brzinu rada
- veliki broj upisa pre otkaza memorije, što utiče na pouzdanost
- nisku potrošnju električne energije
- nisku cenu

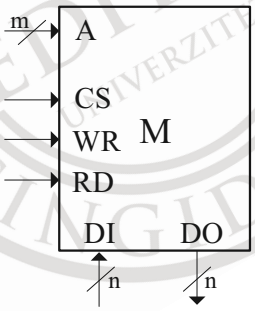
S obzirom da je praktično nemoguće zadovoljiti sve ove karakteristike, razvijeno je više vrsta memorija koje, prema potrebi, poseduju samo neke od

navedenih karakteristika i to one koje su bitne za konkretnu primenu. Po opštoj podeli, memorije mogu biti:

- *nepermanentne*
- *permanentne*

Nepermanentne memorije su memorije čiji se sadržaj gubi u slučaju prestanka napajanja (isključenja sistema, nestanka struje i sl.). Najčešće korišćena nepermanentna memorija je memorija sa ravnopravnim pristupom *RAM* (*Random Access Memory*). Naziv ove memorije potiče od činjenice da kod nje vreme potrebno za upis i čitanje binarne reči ne zavisi od mesta na kome se binarna reč nalazi u memoriji.

RAM memorija se sastoji od memorijske matrice sa 2^n ćelija (ovaj broj ćelija predstavlja kapacitet memorije), pri čemu u svaku ćeliju može da se smesti jedna *n*-bitska reč. Svakoj ćeliji je pridružen broj, tj. adresa. Pristup ćeliji omogućava adresni dekoder koji takođe ulazi u sastav *RAM* memorije. Kada se neka adresa u binarnom obliku dovede na ulaz dekodera, na odgovarajućoj izlaznoj liniji pojavljuje se aktivna vrednost signala čime se adresirana ćelija selektuje za upis ili čitanje. Na slici 5.40 prikazan je grafički simbol *RAM* memorije.



Slika 5.40 RAM memorija

Kada se na ulaz *A* (*Address*) dovede adresa, dekoder (koji se nalazi unutar bloka) selektuje odgovarajuću ćeliju. Ukoliko u tom trenutku na ulazima *WR* (*Write*) i *CS* (*Chip Select*) postoje aktivne vrednosti signala (na primer 1), u selektovanu ćeliju se upisuje podatak koji trenutno postoji na ulazu *DI* (*Data Input*). Međutim, ako aktivne vrednosti signala postoje na ulazima *RD* (*Read*) i *CS*, iz selektovane ćelije će biti pročitana binarna reč koja će se pojaviti na izlazu *DO* (*Data Output*). Čitanjem se ne menja binarna reč u selektovanoj ćeliji. Operacije čitanja i upisa se ne mogu obavljati istovremeno. Da bi se to osiguralo, memorije obično imaju jedinstven ulaz za signale čitanja i upisa, pa jedna binarna vrednost

(na primer 0) predstavlja aktivni signal za čitanje, a druga binarna vrednost (1) aktivni signal za upis. Na ovaj način postaje nemoguće da oba signala budu aktivna u istom trenutku.

RAM memorije se mogu realizovati na različite načine, što zavisi od primenjenje poluprovodničke tehnologije. Za njihovo konstruisanje, osim logičkih i memorijskih elemenata, mogu se koristiti i elektronske komponente, na primer kondenzatori. U skladu sa ovim, uvedena je podela *RAM* memorija na:

- statičke – *SRAM* (*Static RAM*)
- dinamičke – *DRAM* (*Dynamic RAM*)

Statičke memorije su napravljene od memorijskih elementa koji omogućavaju pamćenje sadržaja sve dok se on ne promeni, ili dok se ne isključi napajanje (reč je o nepermanentnim memorijama). Njihove osnovne karakteristike su: mala gustina pakovanja, velika brzina rada pošto je vreme pristupa vrlo kratko i reda je *ns*, mala verovatnoća greške, mala potrošnja električne energije, visoka cena nabavke.

Dinamičke memorije sadrže memorijske elemente koji imaju kondenzatore, tako da se njihov sadržaj mora povremeno osvežavati. Upravo zbog stalnog osvežavanja, u naziv ovih memorija uveden je termin „dinamička“. Proces osvežavanja je vrlo brz (reda *ns*), u potpunosti se obavlja u hardveru memorije (na svakih par *ms*) i neprimetan je za korisnika. Karakteristike dinamičkih memorija su: jednostavna proizvodnja, velika gustina pakovanja, mala potrošnja električne energije, niska cena nabavke, mala brzina rada, manja pouzdanost, tj. veća verovatnoća greške.

U tabeli 5.3 dat je uporedni pregled razmatranih vrsta memorija po zadatim osobinama.

Osobina	Statička memorija	Dinamička memorija
brzina	znatno veća	
kapacitet po čipu		znatno veći
potrošnja	troši manje el.energije	
cena		niža
verovatnoća greške	pouzdanija	
broj upisa pre otkaza	praktično beskonačan	praktično beskonačan

Tabela 5.3 Poređenje statičkih i dinamičkih memorija

Da bi se prevazišao glavni nedostatak nepermanentnih memorija, a to je gubitak njihovog sadržaja sa prestankom napajanja, konstruisane su permanentne memorije.

Permanentne memorije su one memorije kod kojih gubitak napajanja ne utiče na ono što je u njima do tog trenutka zapisano. Za razliku od nepermanentnih memorija koje su čitajuće/upisne (*RW – Read/Write*), u permanentne memorije obično se ne može upisivati u toku rada sistema, već je moguće samo iz njih čitati sadržaj. Takve memorije se nazivaju fiksnim, ili *ROM* (*Read Only Memory*).

Način upisa podataka u *ROM* zavisi od korišćene tehnnologije. Najjednostavniji postupak je upis sadržaja prilikom proizvodnje čipa. Ovako upisan sadržaj se ne može menjati.

Veću fleksibilnost imaju programabilne *ROM*, tj. *PROM* (*Programmabile ROM*) u koje sam korisnik može da upiše sadržaj koji se kasnije, takođe, ne može menjati.

Najveću fleksibilnost imaju *EPROM* (*Erasable PROM*) koje, osim upisa, imaju i mogućnost brisanja sadržaja. Brisanje se obavlja tako što se kroz mali stakleni prozor koji se nalazi na kućištu memorije propusti ultraljubičasta svetlost koja osveli unutrašnjost čipa u trajanju od dvadesetak minuta. Nakon toga, čip je prazan i može se ponovo programirati. Ipak, jedan čip se može izbrisati samo nekoliko desetina puta, nakon čega postaje neupotrebljiv.

Noviji oblik permanentne memorije je fleš (*flash* – munja), čiji naziv asocira na veliku brzinu pristupa. Ova memorija koristi tehnologiju elektronskog upisa i brisanja podataka, pri čemu se tada može pristupati samo blokovima podataka, ali ne i svakom bajtu pojedinačno. Čitanje je obavlja bajt po bajt. Karakteristike ove memorije su: velika brzina čitanja, veliki kapacitet i velika pouzdanost.

Vežbanja

1. Šta su koderi i čemu služe? Objasniti princip rada kodera. Koje vrste kodera postoje i u čemu se one razlikuju?

2. Realizovati potpuni koder 8/3 (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

3. Realizovati potpuni koder sa 4 ulaza (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

4. Šta su dekoderi i čemu služe? Objasniti princip rada dekodera. Koje vrste dekodera postoje i u čemu se one razlikuju?

5. Realizovati potpuni dekoder 3/8 (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću prekidačke mreže).

6. Realizovati potpuni dekoder sa 2 ulaza (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

7. Primenom dekodera realizovati zadate logičke funkcije:

   a) $Y = A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C} + \overline{A}B\overline{C}$
   b) $Y = \overline{A}\overline{B}C + A\overline{B}\overline{C} + AB\overline{C}$
   c) $Y = \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC + AB\overline{C}$

   Objasniti postupak realizacije.

8. Šta je multiplekser i koja je njegova uloga? Objasniti princip rada multipleksera.

9. Realizovati multiplekser 8/1 (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

10. Realizovati multiplekser sa 4 ulaza (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

11. Primenom multipleksera realizovati zadate logičke funkcije:

    a) $Y = \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}\overline{B}C + AB\overline{C}$
    b) $Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C}$
    c) $Y = \overline{A}\overline{B}C + \overline{A}BC + AB\overline{C}$

    Objasniti postupak realizacije.

12. Šta je demultiplekser i koja je njegova uloga? Objasniti princip rada demultipleksera.

13. Realizovati demultiplekser 1/8 (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću logičkih kola).

14. Realizovati demultiplekser sa 4 izlaza (dati kombinacionu tablicu, funkcije izlaza i način realizacije pomoću prekidačke mreže).

15. Šta je sabirač? Koje vrste sabirača postoje i u čemu se one razlikuju? Dati grafički prikaz potpunog sabirača i objasniti princip njegovog rada kroz funkcionalnost njegovih ulaza i izlaza.

16. Pomoću potpunih sabirača realizovati 4-bitni sabirač. Objasniti postupak realizacije.

17. Realizovati 3-bitni sabirač primenom potpunih sabirača. Objasniti postupak realizacije.

18. Šta je aritmetičko-logička jedinica i čemu služi? Nacrtati grafički simbol *n*-bitne *ALU* i objasniti princip njenog rada kroz funkcionalnost njenih ulaza i izlaza.

19. Realizovati 1-bitnu ALU koja može nad dva jednobitna binarna broja *a* i *b* da obavi zadate logičke operacije.

    a) $\bar{a}$, $a \oplus b$ i ab
    b) $\bar{a}$, $\bar{a} \oplus b$, $a+b$ i ab
    c) $\bar{a} \oplus b$, $\bar{a}+b$ i ab
    d) $\bar{a}$, $a \oplus b$, $\bar{a}+b$ i ab

e) ab, $a+b$, $\bar{a}b$ i $a \oplus b$

Objasniti postupak realizacije.

20. Šta su registri i čemu služe? Koje vrste registara postoje i u čemu se one razlikuju?

21. Dati grafički prikaz n-bitnog paralelnog registra i objasniti princip njegovog rada kroz funkcionalnost njegovih ulaza i izlaza.

22. Izvesti funkciju prelaza i-tog razreda paralelnog registra prilikom operacije upisa i predstaviti je pomoću tablice prelaza.

23. Realizovati operaciju upisa u i-ti razred paralelnog registra pomoću D flip-flop-ova.

24. Dati strukturnu šemu koja problem čitanja sadržaja iz paralelnog registra rešava pomoću logičkog I elementa.

25. Dati grafički prikaz n-bitnog serijskog registra i objasniti princip njegovog rada kroz funkcionalnost njegovih ulaza i izlaza.

26. Realizovati 4-bitni serijski registar sa pomeranjem za jedno mesto udesno pomoću taktovanih D FF-ova sa upravljanjem pomoću signala takta. Dati detaljan opis realizacije.

27. Šta su brojači i čemu služe? Šta je moduo brojača? Koje vrste brojača postoje i u čemu se one razlikuju?

21. Nacrtati grafički simbol n-bitnog brojača i objasniti princip njegovog rada kroz funkcionalnost njegovih ulaza i izlaza.

22. Realizovati brojač po modulu:

a) 7
b) 15
c) 24

Objasniti postupak realizacije.

23. Objasniti postupak realizacije inkrementirajućeg brojača koji broji po modulu 2^n pomoću taktovanih RS FF-ova. Dati strukturnu šemu i-tog razreda ovog brojača.

24. Čemu služe memorije? Navesti karakteristike idealne memorije. Koja je osnovna podela memorija?

25. Objasniti razliku između permanentne i nepermanentne memorije.

26. Šta je *RAM* memorija? Nacrtati grafički simbol *RAM* memorije i na osnovu njega objasniti kako se podatak upisuje, a kako čita iz memorije.

27. Navesti vrste *RAM* memorija u zavisnosti od njihove realizacije. Ukratko opisati svaku od njih.

28. Dati uporedni prikaz statičkih i dinamičkih *RAM* memorija po sledećim karakteristikama: brzini, kapacitetu, potrošnji, ceni, verovatnoći greške i broju upisa pre otkaza.

29. Šta su *ROM*, *PROM* i *EPROM*? Navesti njihove mogućnosti.

30. Šta je *flash* memorija? Dati njene osnovne karakteristike.

UNIVERZITET
PRIVATNI
AKREDITOVANI
UNIVERZITET
SINGIDUNUM

6 Osnovi organizacije računara

Računari su digitalni sistemi koji služe za obradu podataka. Na osnovu ulaznih podataka koje zadaje korisnik, a koji predstavljaju veličine bitne za rešavanje nekog problema, računar generiše izlazne podatke koji predstavljaju rešenje tog problema. Iako podaci koje korisnik unosi, kao i rezultati koji mu se prezentuju mogu biti u različitim formatima (tekst, odmerci nekog signala itd.), oni se u računaru konvertuju u binarni oblik, jer samo tako mogu da budu obrađivani. Binarne reči koje predstavljaju podatke su nizovi binarnih cifara 0 i 1 sa definisanim značenjem. To znači da se za svaku binarnu reč u sistemu zna šta ona predstavlja (podatak, instrukciju, i sl.).

Obrada podataka zasniva se na izvršavanju relativno malog broja operacija definisanih nad binarnim rečima. Operacije koje se izvršavaju nad jednom binarnom reči nazivaju se *unarnim*, za razliku od operacija koje se obavljaju nad dvema binarnim rečima koje se zovu *binarnim* operacijama. Sve operacije se mogu svrstati u četiri klase: *aritmetičke*, *logičke*, operacije *pomeranja* i operacije *prenosa*. Kao što je ranije rečeno, aritmetičke operacije posmatraju binarne reči kao binarne brojeve (neoznačene, u pokretnom zarezu, u komplementu dvojke itd.). Logičke operacije se izvode nad pojedinačnim razredima binarnih reči. Operacije pomeranja su unarne i definišu se kao pomeranja binarne reči za određen broj razreda ulevo ili udesno uz odgovarajuće popunjavanje praznih razreda nastalih usled pomeranja. Operacije prenosa služe za prenos binarnih reči sa jedne lokacije na drugu, pri čemu se reč koja se prenosi ne menja. Operacije se u računaru realizuju pomoću kombinacionih i sekvencijalnih mreža od kojih su neke opisane u prethodnom poglavlju (registri, memorije, dekoderi, multiplekseri itd.).

Da bi uneo podatke u računar, korisnik mora da pristupi nekoj od jedinica za ulaz koje računar poseduje. To su obično tastatura ili miš. Rezultate dobija putem izlazne jedinice računara, na primer ekrana, štampača, plotera i sl. Ulazno-izlazne

jedinice se često nazivaju periferijama. Putem periferija računar ostvaruje komunikaciju sa okruženjem (na primer sa drugim računarima).

Podaci koje korisnik unese u računar smeštaju se u operativnu memoriju. Ona komunicira sa ostalim delovima računara, pri čemu ima najživlju komunikaciju sa procesorom računara. Procesor obavlja potrebe operacije nad podacima koji se nalaze u operativnoj memoriji. S obzirom da je operativna memorija brza i zbog toga skupa, ona je ograničenog, relativno malog kapaciteta.Zato se svi podaci sa kojima računar trenutno ne radi smeštaju u eksternu, tj. spoljašnju memoriju koja je znatno sporija, ali ima mnogo veći kapacitet od operativne memorije. Primeri eksterne memorije su hard disk, fleš disk, *DVD*, *CD* i sl. Ulazno-izlazne jedinice, procesor, operativna memorija i eksterne memorije čine glavne fizičke delove računara koji se jednim imenom nazivaju hardver (*hardware*). Sam hardver nije dovoljan za rad računara. Da bi računar mogao da izvrši neku obradu podataka, neophodno je definisati program rada računara u kome će precizno biti zadate sve aktivnosti u okviru obrade, kako po pitanju sadržaja, tako i po pitanju redosleda njihovog izvođenja. Stoga program predstavlja niz pojedinačnih instrukcija ili naredbi koji se čuva u memoriji računara. U računaru postoji veći broj različitih programa koji se jednim imenom nazivaju softver (*software*). Treba uočiti da se izborom programa menja način obrade podataka iako hardver računara ostaje neizmenjen. Programi se, u zavisnosti od porekla, mogu svrstati u dve kategorije: *sistemski* i *aplikativni* softver. Sistemskim softverom se obično nazivaju programi koje proizvođači računara isporučuju zajedno sa računarom. To su najčešće programi bez kojih računar ne može da radi (na primer operativni sistem), kao i neki uslužni programi koje korisnici primenjuju nezavisno od problema kojim se bave (prevodioci, sistemi za upravljanje fajlovima, i sl.). Aplikativni softver čine programi koje primenjuju razne kategorije korisnika za potrebe rešavanja svojih problema. Strogu granicu između sistemskog i aplikativnog softvera ponekad je teško definisati. Na primer, još uvek ne postoji saglasnost oko toga u koju kategoriju spada program *Internet Explorer* web pretraživač, tj. da li je on deo operativnog sistema *Windows* ili nije.

Procesor izvršava program koji je prethodno smešten u operativnu memoriju instrukciju po instrukciju i to onim redosledom kojim su instrukcije navedene u programu, sve dok ne dođe do kraja programa. Ovaj postupak se može predstaviti ciklusima koji sadrže sledeća četiri koraka:

- dohvatanje instrukcije iz memorije
- dohvatanje operanada, ukoliko se to zahteva u instrukciji
- izvršavanje instrukcije
- upis rezultata u memoriju ili slanje na neku izlaznu jedinicu ako se to u instrukciji zahteva

Operandi su podaci u binarnom obliku nad kojima se vrše željene operacije. Na primer, brojevi koje treba sabrati su operandi za operaciju sabiranja. Operandi se obično nalaze u operativnoj memoriji (kao i programi), mada se mogu dobiti i od neke periferne jedinice.

Kao što se vidi iz navedenih koraka, između procesora i memorije odvija se vrlo intenzivna komunikacija, jer je potreba za čitanjem instrukcija i podataka (operanada) iz memorije, kao i upisivanjem rezultata u nju vrlo česta. Prenos instrukcija i podataka između procesora i memorije obavlja se preko bidirekcionih (dvosmernih) veza koje se nazivaju *magistralom podataka*. Izbor memorijske lokacije iz koje treba uzeti podatak ili u koju treba upisati podatak obavlja se pomoću adrese koju određuje procesor. Adresa se šalje memoriji preko unidirekcionih (jednosmernih) veza koje se nazivaju *adresnom magistralom*. Prenos po magistralama je paralelan, što znači da se svi biti memorijske reči prenose istovremeno. Širina adresne magistrale, u većini slučajeva, određuje kapacitet operativne memorije. Tako, ako adresa ima n bita, pomoću nje se može adresirati $C = 2^n$ različitih memorijskih lokacija, pa C predstavlja maksimalni kapacitet memorije.

Upis binarnog podatka u memoriju obavlja se u sledećim koracima:

- procesor adresira željenu memorijsku lokaciju postavljanjem njene adrese na adresnu magistralu
- procesor postavlja podatak koji treba da se upiše na magistralu podataka
- procesor šalje komandu memoriji da obavi upis podatka

Postupak čitanja binarnog podatka iz memorije odvija se na sledeći način:

- procesor adresira memorijsku lokaciju iz koje želi da pročita podatak postavljanjem njene adrese na adresnu magistralu
- procesor šalje komandu memoriji da na magistralu podataka postavi adresirani podatak
- procesor preuzima podatak sa magistrale podataka

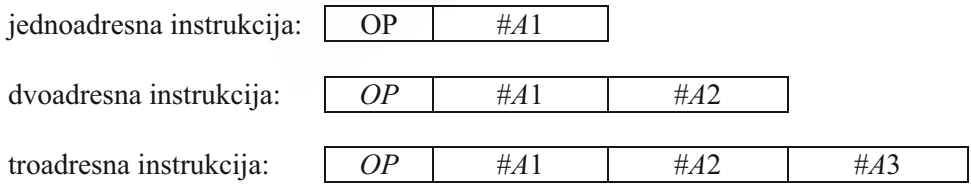
Pošto je u organizaciji računara procesor odgovoran za izvršavanje instrukcija, on mora da ima mogućnost lokalnog čuvanja izvesne količine podataka. Za to se koriste brzi procesorski registri. Registri mogu da imaju različite namene. Na primer, postoji registar u kome se čuva adresa naredne instrukcije koju treba dohvatiti, zatim registar sa instrukcijom koju treba izvršiti, registri u koje se smeštaju operandi, registri opšte namene itd. U zavisnosti od načina pristupa, registri se mogu klasifikovati u: *adresibilne* i *interne* registre. Adresibilni su oni registri kojima programer može da pristupa, tj. da čita i menja njihov sadržaj.

Nasuprot njima, interni registri nisu dostupni programeru, već se njihov sadržaj postavlja i menja automatski tokom rada procesora.

Da bi dohvatio neku instrukciju, procesor mora da ima informaciju o tome na kojoj lokaciji u memoriji se ona nalazi. Na početku rada, adresa prve instrukcije programa se hardverski unosi u registar procesora u kome se čuva adresa naredne instrukcije. Pošto se ovaj registar obično realizuje kao brojač, adresa svake sledeće instrukcije programa dobija se automatski inkrementiranjem brojača (ukoliko nije naredba skoka). Na osnovu poznate adrese, iz memorije se doprema odgovarajuća instrukcija u smešta u registar za instrukcije, nakon čega sledi njeno izvršavanje.

Izvršavanje instrukcije se svodi na obavljanje onih operacija koje su njome specificirane. Radi jednostavnije realizacije, tj. manje složenosti kombinacionih i sekvencijalnih mreža kojima se realizuju, mnoge složenije operacije se izvode preko nekoliko jednostavnijih operacija. Na primer, množenje se često realizuje preko operacija sabiranja i pomeranja, deljenje preko oduzimanja i pomeranja i sl. U opštem slučaju, instrukcija se razlaže na više koraka sa jednom ili više operacija u svakom koraku. U jedan korak može se uključiti više operacija ukoliko one mogu da se obavljaju paralelno, tj. istovremeno. Treba napomenuti da ovakvo izvršavanje ne predstavlja obavezu, jer se za svaku definisanu operaciju može konstruisati jedinstvena kombinaciona mreža koja je realizuje.

Da bi instrukcija mogla da se izvrši, neophodno je prethodno obezbediti raspoloživost operanada bitnih za njeno izvršenje. Stoga, instrukcija u svom formatu mora da specificira, osim operacije koju procesor treba da obavi, i podatke (ili adrese na kojima se podaci nalaze) nad kojima se operacija obavlja. Svaka instrukcija se sastoji iz dva dela: operacionog koda (koji definiše operaciju) i adresnog dela (koji specificira podatke, tj. operande). Da bi procesor znao koju operaciju treba da obavi, najpre se mora dekodovati kod operacije pomoću adresnog dekodera, a zatim se na osnovu adresnog dela instrukcije dopremaju operandi. Adresni deo može da sadrži jednu ili više adresa (ili podataka) u zavisnosti od broja operanada, kao što je prikazano na slici 6.1.



Slika 6.1 Struktura instrukcije

Na primer, ako operacioni kod odgovara operaciji inkrementiranja, dovoljno je u adresnom delu navesti samo jednu adresu (#*A*1) na kojoj se nalazi sadržaj koji treba uvećati za 1.

Pošto se operandi dopreme u registre za operande, aritmetičko-logička jedinica izvršava operaciju definisanu kodom operacije. Razultat se po potrebi prenosi u operativnu memoriju, ili se prosleđuje nekoj izlaznoj jedinici.

U organizaciji računara, osim procesora koji zauzima centralno mesto, značajnu ulogu imaju i brojni koncepti koji su vremenom uvođeni sa ciljem poboljšanja efikasnosti rada računara. U nastavku su ukratko opisana tri značajna koncepta.

Pipeline predstavlja koncept koji je uveden sa ciljem da se ubrza proces izvršavanja programa. Koncept se zasniva na korišćenju konkurentnosti prilikom izvršavanja instrukcija. Pretpostavimo da se proces izvršavanja instrukcije odvija u četiri faze:

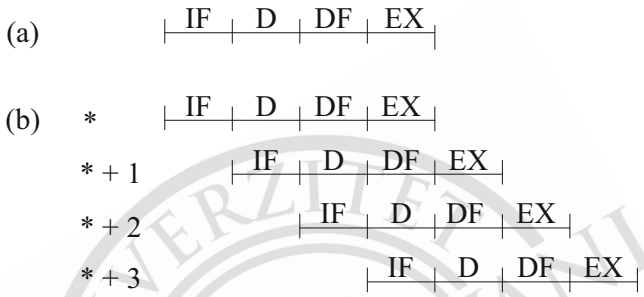
- *IF* (*Instruction Fatch*): dohvatanje instrukcije iz memorije i njeno smeštanje u registar za instrukcije
- *D* (*Decode*): dekodiranje operacionog koda instrukcije
- *DF* (*Data Fatch*): dohvatanje podataka (operanada) bilo iz memorije ili iz odgovarajućih registara
- EX (*Execute*): izvršavanje instrukcije

Tipičan postupak izvršavanja instrukcije prikazan je na slici 6.2 (a). Instrukcija se izvršava tako što prolazi kroz faze *IF*, *D*, *DF* i na kraju *EX*.

Ideja u *pipeline* konceptu je da se omogući da se u istom trenutku različite instrukcije nalaze u različitim fazama izvršavanja i da se tako ostvari znatna ušteda u vremenu izvršavanja programa. To znači, na primer, da naredna instrukcija može da se dohvata iz memorije dok traje dekodiranje tekuće instrukcije. S obzirom da su ove dve aktivnosti međusobno nezavisne, nema nikakvih prepreka da se obavljaju istovremeno. *Pipeline* mašine upravo pokušavaju da obezbede stalnu zauzetost po svim fazama, tj. da u svakom taktu postoji po jedna instrukcija u svakoj od faza izvršavanja. Na slici 6.2 (b) prikazan je *pipeline* koncept na primeru izvršavanja četiri uzastopne instrukcije od kojih se prva nalazi na nekoj adresi *.

Kao što se vidi, u prvom taktu dohvata se prva instrukcija iz memorije, u drugom se ona dekodira i istovremeno dohvata druga instrukcija iz memorije. U trećem taktu dohvataju se operandi za prvu instrukciju, druga instrukcija se dekodira i istovremeno se dohvata treća instrukcija iz memorije. U četvrtom taktu je ostvareno najveće ubrzanje pošto u svim fazama postoji po jedna instrukcija koja

se obrađuje: prva instrukcija je u fazi izvršenja, druga u fazi dohvatanja operanada, treća u fazi dekodovanja i četvrta u fazi dopremanja iz memorije. U petom taktu izvršava se druga instrukcija, dohvataju se operandi za treću, a dekodira četvrta. U šestom taktu treća instrukcija se izvršava, a za četvrtu se dopremaju operandi. Na kraju, u sedmom taktu, izvršava se četvrta instrukcija. Dakle, sve četiri instrukcije su izvršene u periodu od 7 taktova, dok bi bez primenjenog *pipeline* koncepta za njihovo izvršenje bilo potrebno 16 taktova, po 4 za svaku instrukciju.



Slika 6.2 Izvršavanje instrukcija (a) obično (b) u *pipeline*-u

Ubrzanje koje se može ostvariti primenom *pipeline* koncepta zavisi od broja stepena (*stages*) ili dubine *pipeline*-a, tj. faza u izvršavanju instrukcije. U datom primeru broj stepeni je 4, pa je maksimalno ubrzanje četiri puta pod uslovom da je mašina dobro podešena. Naravno, postizanje ubrzanja ima svoju cenu. Ona se ogleda u većoj složenosti i većoj ceni procesora sa ovakvim mehanizmom izvršavanja instrukcija. Ipak, skoro svi današnji procesori podržavaju *pipeline* koncept.

DMA (*Direct Memory Access*) je koncept prisutan u svim modernim računarima koji omogućava direktan prenos podataka između periferije i memorije, bez posredovanja procesora. Zahvaljujući njemu, procesor, kao najskuplji i najopterećeniji resurs u računarskom sistemu, oslobođen je mnogih nepotrebnih poslova. Naime, u sistemu koji nema *DMA*, da bi podaci bili prebačeni od izvora do odredišta (na primer sa neke periferije do memorije) procesor bi morao da kopira deo po deo podataka sa izvora i da ih prosleđuje na odredište, pri čemu, tokom celog tog procesa ne bi bio raspoloživ ni za kakvu drugu aktivnost. Otežavajuća okolnost je još i ta što su periferne jedinice mnogo sporije od operativne memorije, zbog čega se gubi dodatno vreme. Uvođenjem *DMA* koncepta, procesor se oslobađa ovog posla i za vreme transfera podataka od izvora do odredišta može da obavlja druge poslove. Ipak, treba imati u vidu da u tim poslovima ne može da koristi magistralu kojom se obavlja *DMA* prenos.

Pri *DMA* prenosu važnu ulogu ima *DMA* kontroler (on preuzima poslove oko prenosa koje procesor ima u sistemu bez *DMA*). *DMA* prenos inicira procesor slanjem odgovarajuće komande *DMA* kontroleru. Nakon toga, potpunu odgovornost za prenos preuzima *DMA* kontroler (generisanje adresa, transfer podataka). Po završetku prenosa, *DMA* kontroler obaveštava procesor da je prenos završen i da je magistrala slobodna.

DMA transfer se primenjuje u sledećim slučajevima:

- kada je potrebno preneti veliku količinu podataka između periferije i memorije, pa bi posredovanje procesora znatno usporilo transfer
- kada je periferija relativno brza (kao na primer hard disk)

Mehanizam prekida je vrlo važan koncept u računarskim sistemima koji je uveden kako bi se izbeglo da procesor troši vreme čekajući na spoljašnje događaje. Značaj ovog mehanizma biće ilustrovan na konkretnom primeru rešavanja problema koji sledi.

Problem: Poznato je da su periferije znatno sporije od procesora. Pretpostavimo da procesor treba da pošalje veću količinu podataka štampaču da ih odštampa. Zbog ograničenih mogućnosti štampača, podaci se moraju slati u više delova. Kada procesor pošalje jedan deo na štampanje, on mora da sačeka da štampač završi svoj posao, tj. da odštampa prispele podatke, kako bi poslao sledeći deo. Čekanje na periferiju (dok ona završi svoj posao) predstavlja izgubljeno vreme za procesor, jer je on tada besposlen. S obzirom da je procesorsko vreme dragoceno, bilo je neophodno naći neko prihvatljivo rešenje za prevazilaženje ovog problema.

Rešenje problema: Problem je rešen tako što je uveden mehanizam prekida koji dopušta procesoru da izvršava druge aktivnosti za vreme dok periferija obavlja svoj posao.

Mehanizam prekida obezbeđuje efikasniji rad računara sa periferijama tako što, za razliku od principa čekanja na periferiju da završi svoj posao, uvodi novi princip opsluživanja periferije na njen zahtev. To znači da kada procesor pošalje podatke periferiji, odmah nastavlja dalje sa radom ne čekajući da periferija obavi svoj posao. U trenutku kada periferija završi posao, ona obaveštava procesor o tome i tada procesor preduzima mere za slanje naredne količine podataka periferiji.

Postupak opsluživanja prekida odvija se na sledeći način:

- Kada periferija postane spremna za prijem podataka od procesora, ona to signalizira procesoru slanjem *zahteva za prekidom*. Zahtev sadrži *kod prekida* koji odgovara periferiji koja ga je poslala.

- Po prijemu zahteva, procesor završava izvršavanje tekuće instrukcije i na kratko prekida izvršavanje tekućeg programa kako bi opslužio prispeli zahtev.
- Svaki procesor ima skup prekida koje je u stanju da opsluži. Za svaki prekid unapred je definisana tzv. *prekidna rutina* koja predstavlja podprogram koji treba da se izvrši u slučaju prispeća zahteva za tim prekidom. Prekidne rutine su smeštene u memoriji na određenim adresama. Sve adrese prekidnih rutina smeštene su u tabelu prekida *(Interrupt Pointer Table*) koja se, takođe, nalazi u memoriji. Procesor opslužuje prekid tako što na osnovu koda prekida iz prispelog zahteva pronalazi u tabeli prekida adresu odgovarajuće prekidne rutine i izvršava rutinu. U prekidnoj rutini se opslužuje odgovarajuća periferija.
- Po završetku prekidne rutine, procesor se vraća tamo gde je stao u programu koji je izvršavao u trenutku nailaska zahteva za prekidom i nastavlja sa radom.

Da bi opisani mehanizam prekida mogao uspešno da funkcioniše, neophodno je obezbediti da se pri povratku iz prekidne rutine procesor nađe i potpuno istim uslovima u kojima je bio kada je počeo opsluživanje prekida. Ti uslovi se nazivaju *kontekstom procesora*. To znači da kada procesor prihvati zahtev za prekidom, pre nego što pređe na njegovo opsluživanje, mora da sačuva kontekst u kome je izvršavao tekući program. Kontekst obično čine trenutni sadržaji pojedinih registara procesora. Nakon završetka prekidne rutine, sačuvani kontekst procesora se restaurira (postavljaju se zapamćene vrednosti u registre) i procesor zna odakle treba da nastavi izvršavanje započetog programa i kakav je bio status nakon poslednje operacije koju je *ALU* izvršila. Pamćenje konteksta procesora je neophodno zato što se prekidna rutina izvršava kao i svaki drugi program, što znači da se tom prilikom koriste isti procesorski registri, pa obično dolazi do izmene njihovog sadržaja.

Prekidi se mogu klasifikovati na različite načine. Jedna od podela je na:

- spoljašnje ili eksterne prekide
- unutrašnje ili interne prekide

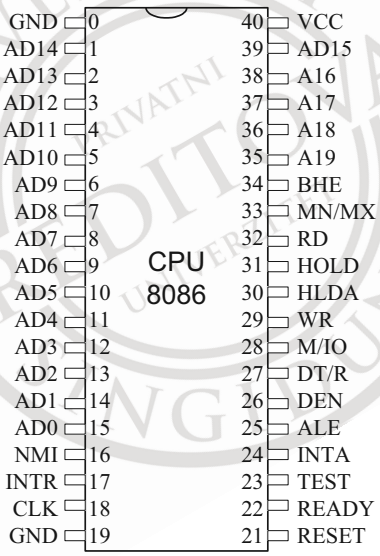
Spoljašnji prekidi dolaze od periferija, dok unutrašnji prekidi mogu biti posledica izvršavanja instrukcije prekida, ili posledica neke neregularnosti u izvršavanju tekuće instrukcije. Prekidima se pridružuju *prioriteti* koji ukazuju na njihovu važnost. Prednost u opsluživanju imaju prekidi sa većim prioritetom. Interni prekidi su višeg prioriteta od eksternih, tako da ako u toku opsluživanja nekog eksternog prekida pristigne zahtev za internim prekidom, procesor prekida opsluživanje eksternog prekida i prelazi na opsluživanje internog prekida.

Međutim, ako u toku opsluživanja internog prekida stigne zahtev za eksternim prekidom, procesor će ga prihvatiti tek kada završi započeti posao.

6.1. Organizacija računara sa procesorom Intel 8086

Organizacija računara, kao i koncepti koji su teorijski obrađeni u prethodnom poglavlju, u ovom poglavlju biće prikazani na praktičnom primeru. Posmatraćemo računarsko okruženje u kome centralno mesto zauzima procesor *Intel* 8086. Ovaj procesor je izabran zato je pogodan za edukativnu svrhu, a i sve kasnije generacije Intel-ovih procesora zadržale su kompatibilnost i sličnu strukturu kao i on.

Čip koji odgovara procesoru *Intel* 8086 ima 40 pinova. Raspored pinova (*pin-out*) prikazan je na slici 6.3, dok je u tabeli 6.1 dato objašnjenje šta koji pin predstavlja.



Slika 6.3 Procesor *Intel* 8086

Procesor *Intel* 8086 ima sledeće karakteristike:

- radni takt procesora je, u zavinosti od verzije, 2, 5, 8 ili 10 MHz
- postoji 16-bitna magistrala podataka
- postoji 20-bitna adresna magistrala koja dopušta adresiranje 2^{20} memorijskih ćelija, što odgovara 1MB memorije

- magistrale podataka i adresa su multipleksirane, što znači da se za njih koriste isti pinovi na procesorskom čipu
- postoje dve linije za spoljašnje prekide
- postoji poseban ulazno/izlazni (*I*/*O*) adresni prostor veličine 64KB za adresiranje perifernih jedinica
- postoji podrška za *DMA*

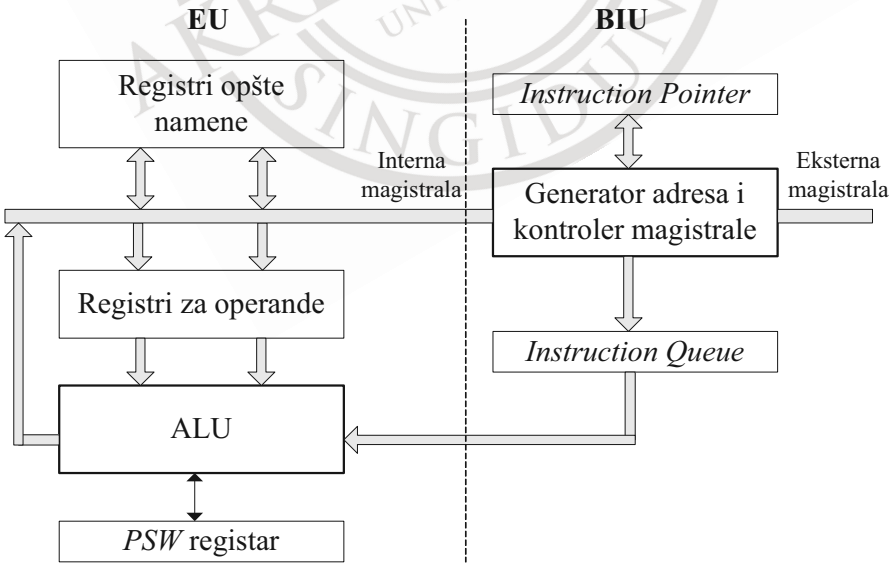
Pin	Opis
VCC	pozitivan kraj napajanja procesora (+5V)
GND	negativan kraj napajanja procesora (masa)
*AD*0-*AD*15	multipleksirane magistrale podataka i adresa
*A*16-*A*19	dodatne 4 linije za adresnu magistralu
NMI [*Not Masked Interrupt*]	ulaz na koji se dovode nemaskirani prekidi
INTR [*Interrupt Request*]	ulaz na koji stiže zahtev za spoljašnjim prekidom
INTA [*Interrupt Acknowledge*]	izlaz za signaliziranje da će zahtev za prekidom biti prihvaćen
CLK [*Clock*]	ulaz za takt na kome radi procesor
BHE [*Bus High Enable*]	izlaz kojim procesor signalizira da je u 16-bitnom modu rada
MN/*MX* [*Minimal*/*Maximal*]	ulaz za signaliziranje min. ili max. moda rada (da li je procesor jedini u sistemu ili ne)
RD [*Read*]	izlaz kojim procesor zahteva čitanje podatka iz memorije ili sa periferije
WR [*Write*]	izlaz kojim procesor šalje signal za upis u memoriju ili periferiju
HOLD	ulaz kojim *DMA* kontroler traži od procesora da mu prepusti magistralu
HLDA [*Hold Acknowledge*]	izlaz kojim procesor signalizira *DMA* kontroleru da mu je prepustio magistralu
M/*IO*	izlaz kojim procesor obaveštava ostatak sistema da li je adresa koju je postavio na adresnu magistralu adresa u memoriji ili adresa periferije

ALE [*Address Latch Enable*]	izlaz kojim procesor signalizira da je na magistralu postavio adresu
DEN [*Data Enable*]	izlaz kojim procesor signalizira da je na magistralu postavio podatak
READY	ulaz kojim memorija obaveštava procesor da je spremna za sledeći ciklus upisa ili čitanja
RESET	ulaz kojim se resetuje procesor
TEST	ulaz koji obaveštava procesor da je sistem u test modu i da instrukcije izvršava jednu po jednu (*step mode*)

Tabela 6.1 Funkcije pinova procesora *Intel* 8086

6.1.1 Unutrašnja struktura procesora

Osnovna uloga procesora *Intel* 8086 u posmatranom okruženju je da omogući proces izvršavanja instrukcije koji se odvija u četiri faze: dohvatanje instrukcije iz memorije i dekodovanje njenog operacionog koda, dohvatanje operanada ako to instrukcija zahteva, izvršavanje instrukcije i upis rezultata u memoriju ili slanje na odgovarajuću perifernu jedinicu. Da bi odgovorio ovom zadatku, po svojoj unutrašnjoj strukturi, procesor *Intel* 8086 podeljen je u dve osnovne jedinice prikazane na slici 6.4:



Slika 6.4 Struktura procesora *Intel* 8086

- *EU* (*Execution Unit*) – izvršna jedinica koja izvršava instrukcije
- *BIU* (*Bus Interface Unit*) – jedinica za spregu sa magistralom koja dohvata instrukcije iz memorije, čita operande i šalje rezultate u memoriju ili ka periferijama

EU i *BIU* su međusobno nezavisne jedinice i najčešće svoje aktivnosti obavljaju istovremeno.

Izvršna jedinica nije direktno povezana sa spoljašnjom magistralom, tako da jedino posredno, preko jedinice za spregu, može da ostvari komunikaciju sa memorijom i periferijama.

Centralni deo *EU* je aritmetičko-logička jedinica (*ALU*) koja, u zavisnosti od instrukcije, izvršava aritmetičke ili logičke operacije nad operandima. Informaciju o tome koju instrukciju treba da obavi, *ALU* dobija iz *Instruction Queue* koji je sastavni deo *BIU*, dok operande preuzima iz dva 16-bitna registra za operande. *EU*, takođe, sadrži i osam 16-bitnih registara opšte namene, kao i *PSW* (*Processor Status Word*) registar koji služi za pamćenje tekućeg statusa procesora.

Operandi mogu da dođu u registre za operande na tri načina: iz memorije, sa periferije ili iz registara opšte namene ukoliko su rezultat prethodne operacije koju je izvršila *ALU* (često je rezultat jedne operacije operand za narednu operaciju).

Registri opšte namene su podeljeni u dve grupe od po četiri registra. Prvu grupu čine registri za podatke, a drugu indeksni i pokazivački registri. Registi za podatke služe za čuvanje podataka generisanih tokom izvršavanja programa. Reč je o adresibilnim registrima, tako da korisnik može da im pristupa bez ograničenja radi čitanja ili menjanja trenutnog sadržaja. Njihova specifičnost je u tome što je moguće posebno adresirati njihovu gornju i donju polovinu (1 bajt), što znači da se svaki registar za podatke može koristiti kao jedan 16-bitni, ili kao dva 8-bitna registra.

PSW je 16-bitni registar koji sadrži više informacija, pri čemu je svaka od njih predstavljena jednim bitom registra koji se naziva flegom (eng. *flag* – zastavica). *PSW* registar obuhvata 6 statusnih flegova, 3 kontrolna flega, a 7 neiskorišćenih bitova.

Statusni flegovi predstavljaju informacije o bitnim osobinama rezultata poslednje operacije koju je izvršila *ALU*. Ovi flegovi su internog karaktera, tako da korisnik može samo da ih čita, ali ne i da ih menja. Njihove vrednost može da postavlja jedino *EU*. Značenja statusnih flegova data su u tabeli 6.2.

Kontrolni flegovi predstavljaju informacije o parametrima bitnim za rad procesora. Njihove vrednosti postavlja korisnik. Kontrolni flegovi su: *DF*

(*Direction flag*), *IF* (*Interrupt flag*) i *TF* (*Trap flag*). Za ovo izlaganje od interesa je samo *IF* fleg koji zabranjuje ili dozvoljava spoljašnje prekide koji dolaze po *INTR* liniji. Spoljašnji prekidi koji stižu po *NMI* liniji i unutrašnji prekidi se ne mogu zabraniti ovim flegom. Više detalja o ulozi ovog flega biće izneto kasnije.

Statusni flegovi	Opis
CF [*Carry flag*]	Setovan na 1 ako je bilo prenosa ili pozajmica pri računanju najvišeg bita rezultata. Dobija novu vrednost nakon svake instrukcije sabiranja, oduzimanja ili rotiranja.
PF [*Parity flag*]	Setovan na 1 ako rezultat ima paran broj jedinica. Koristi se za proveru ispravnosti prenosa podataka do periferija.
AF [*Auxiliary Carry flag*]	Setovan na 1 ako je pri računanju rezultata bilo prenosa ili pozajmice između nižeg i višeg bajta.
ZF [*Zero flag*]	Setovan na 1 ako je rezultat operacije 0.
SF [*Sign flag*]	Setovan na 1 ako je rezultat negativan broj, tj. počinje sa 1 (negativni brojevi se predstavljaju u komplementu dvojke).
OF [*Overflow flag*]	Setovan na 1 ako je došlo do prekoračenja, tj. zbog svoje veličine, rezultat ne može da bude upisan u predviđenu lokaciju.

Tabela 6.2 Statusni flegovi *PSW* registra

Jedinica za spregu (*BIU*) obavlja svu potrebnu razmenu podataka i instrukcija između procesora sa jedne strane i memorije i periferija sa druge. Osim generatora adresa i kontrolera magistrale koji ovde neće biti detaljno analizirani, *BIU* sadrži još *Instruction Pointer* (*IP*) registar i *Instruction Queue* (*IQ*).

IP je registar koji sadrži adresu u memoriji na kojoj se nalazi sledeća instrukcija koja treba da bude prosleđena procesoru. Pošto se naredbe programa u memoriji čuvaju na sukcesivnim adresama, *IP* registar se obično realizuje kao brojač koji se inkrementira nakon preuzimanja svake nove instrukcije iz memorije. Ukoliko se u programu naiđe na naredbu skoka (na primer *GOTO*, *CALL*, *JUMP* i dr.) koja prebacuje kontrolu na neku drugu lokaciju u memoriji, neophodno je promeniti vrednost IP registra na adresu te druge lokacije. Nakon toga, prva sledeća instrukcija koju treba izvršiti je ona koja se nalazi na adresi upisanoj u *IP*.

IQ predstavlja internu memoriju procesora u kojoj se čuvaju naredne instrukcije koje čekaju na izvršenje. Kod procesora *Intel* 8086 u *IQ* se može smestiti do 6 instrukcija. Korišćenje *IQ* obezbeđuje da *EU* uvek bude snabdevena

novim instrukcijama. U slučaju pojave instrukcije skoka, sadržaj *IQ* postaje neodgovarajući, pa se mora promeniti. To se radi tako što *BIU* najpre resetuje *IQ*, zatim dohvata instrukciju sa nove adrese postavljene u *IP* i direktno je prosleđuje procesoru (da on ne bi čekao), a nakon toga se *IQ* puni narednim instrukcijama.

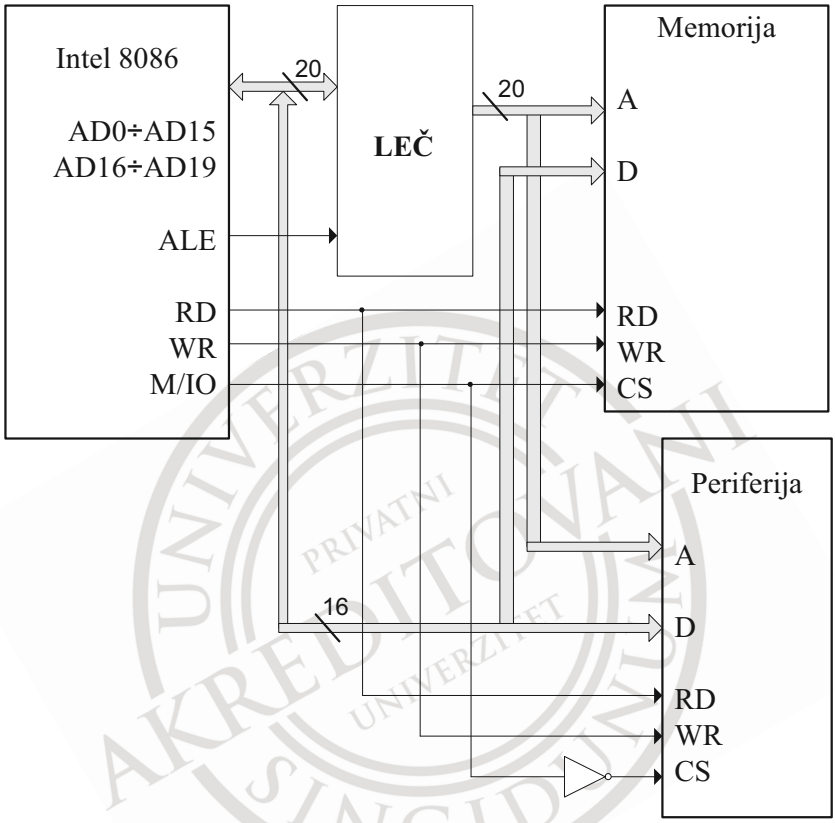
U nastavku će biti opisan princip rada procesora *Intel* 8086. Pretpostavimo da procesor izvršava neki program i da je u datom trenutku stigao do neke instrukcije u programu. Ta i narednih 5 instrukcija u programu već se nalaze u *IQ*. U *IP* se nalazi adresa instrukcije koja sledi nakon instrukcija smeštenih u *IQ*. *ALU* uzima prvu instrukciju iz *IQ*, dekodira njen operacioni kod i, ukoliko to instrukcija zahteva, pokušava da dohvati potrebne operande. Ako su operandi već u registrima za operande (na primer generisani prilikom izvršavanja prethodne instrukcije), *ALU* izvršava instrukciju. Ako operandi nisu raspoloživi, već moraju biti dopremljeni iz memorije ili sa neke periferije, *EU* šalje zahtev *BIU* da pribavi operande. *BIU* pronalazi podatke i smešta ih u registre za operande, nakon čega *ALU* izvršava instrukciju. Tokom izvršavanja instrukcije u *EU*, *BIU* dohvata narednu instrukciju iz memorije (na osnovu adrese u *IP* registru) i dopunjava *IQ*. Rezultat rada *ALU* može da bude operand za narednu instrukciju, podatak bitan za dalji rad programa, ili podatak koji treba smestiti u memoriju ili poslati nekoj periferiji. Stoga se on može poslati u registre za operande, registre opšte namene ili proslediti *BIU* sa zahtevom da ih pošalje u memoriju ili ka određenoj periferiji. Po izvršenju instrukcije obično se postavljaju i statusni flegovi *PSW* registra koji opisuju dobijeni rezultat. Zatim se prelazi na izvršavanje naredne instrukcije koja se nalazi u *IQ*.

6.1.2 Razmena podataka sa okruženjem

Potreba za razmenom binarnih informacija između procesora i okruženja (memorije, periferija) je vrlo česta, jer se podaci i programi nalaze na jednoj lokaciji (u memoriji i na periferijama), a njihova obrada se obavlja na drugoj lokaciji (u procesoru). Na slici 6.5 prikazan je način povezivanja procesora *Intel* 8086 sa memorijom i periferijama u cilju ostvarivanja međusobne komunikacije.

Procesor *Intel* 8086 razmenjuje informacije sa ostatkom sistema putem 20-bitne magistrale. Magistrala je jednim delom multipleksirana zato što se njenih 16 bita (*AD*0-*AD*15) koristi za prenos i adresa i podataka, dok se preostala četiri bita (*A*16-*A*19) koriste samo za adrese. To znači da se magistalom podataka prenose 16-bitni podaci, dok je adresna magistrala 20-bitna i omogućava adresiranje memorije kapaciteta 1MB. Problem koji ovde nastaje je u tome što nije moguće da se na istim pinovima istovremeno pojave i adresa i podatak, a njihova istovremena raspoloživost je neohodna prilikom upisa i čitanja sadržaja. Ovaj problem je rešen upotrebom leča (*latch*). Leč je sekvencijalno kolo slično registru, s tom razlikom

što se nakon upisa podatka u leč, podatak automatski pojavljuje na njegovom izlazu. Ova osobina leča je iskorišćena za privremeno pamćenje adrese.



Slika 6.5 Povezivanje procesora *Intel* 8086 sa okruženjem

Procesor razmenjuje podatke sa memorijom dvosmerno, tj. može da upisuje podatak u memoriju ili da ga iz nje čita. Postupak upisa binarnog podatka u memoriju odvija se na sledeći način:

- procesor adresira željenu memorijsku lokaciju postavljanjem njene adrese na adresnu magistralu (*AD*0-*A*19)
- procesor aktivira *ALE* signal kojim se adresa upisuje u leč i prosleđuje na njegov izlaz; pošto je izlaz leča direktno povezan sa adresnim ulazom memorije *A*, adresa se prosleđuje do memorije

- na magistralu podataka (*AD*0-*AD*15) procesor postavlja podatak koji treba da bude upisan u memoriju i on se direktno prosleđuje do memorijskog ulaza za podatke *D*
- aktiviranjem signala *WR*, koji je direktno povezan sa ulazom za signal upisa u memoriju *WR*, procesor zadaje komandu memoriji da izvrši upis podatka sa *D* na lokaciju čija je adresa na *A* ulazu memorije

Čitanje binarnog podatka iz memorije obavlja se u sledećim koracima:

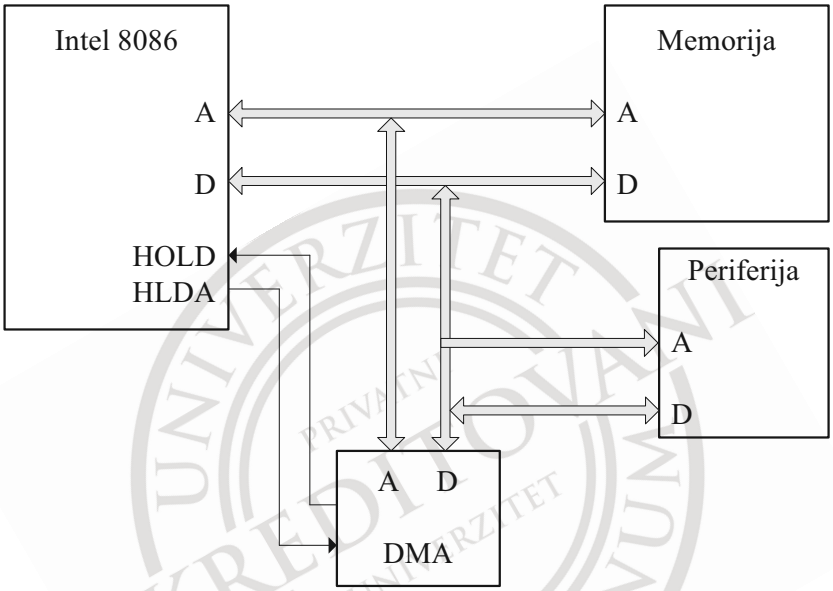
- procesor adresira memorijsku lokaciju iz koje želi da pročita podatak postavljanjem njene adrese na adresnu magistralu (*AD*0-*A*19)
- procesor aktivira *ALE* signal kojim se adresa upisuje u leč i prosleđuje na njegov izlaz; pošto je izlaz leča direktno povezan sa adresnim ulazom memorije *A*, adresa se prosleđuje do memorije
- aktiviranjem signala *RD*, koji je direktno povezan sa ulazom za signal čitanja iz memorije *RD*, procesor zadaje komandu memoriji da sa adrese na njenom *A* ulazu pročita podatak i postavi ga na magistralu podataka *D*
- zbog direktne veze *D* izlaza memorije i magistrale podataka (*AD*0-*AD*15), procesor preuzma podatak

Osim sa memorijom, procesor na sličan način razmenjuje podatke i sa periferijma. Kao i memorijske ćelije, svaka periferija ima svoju adresu. Prilikom obraćanja periferiji, procesor postavlja njenu adresu na adresnu magistralu. Odgovor koji dobija potiče od adresirane periferije.

Problem nastaje zbog činjenice da se adrese memorijskih lokacija poklapaju sa adresama periferija. Ukoliko ne bi postojala nikakva dodatna informacija o tome kome procesor želi da se obrati, desilo bi se da mu istovremeno odgovore i memorija i periferija, što bi dovelo do greške u sistemu zbog sudara podataka koji dolaze sa različitih strana na istu magistralu podataka. Ovaj problem je rešen uvođenjem signala *M*/*IO* na osnovu koga se tačno zna čiju adresu je procesor postavio na magistralu. Ako *M*/*IO* signal ima vrednost 1, procesor se obraća memoriji, a ako ima vrednost 0, obraća se periferiji (videti sliku 6.5). Signal *M*/*IO* koji generiše procesor vodi se na *CS* (*Chip Select*) ulaze memorije i periferija. *CS* ulazi omogućavaju prelazak čipova u aktivan režim. To znači da ako signal *M*/*IO* ima vrednost 1, on se prosleđuje na *CS* ulaz memorije i onda memoriji može da se pristupa. Istovremeno, zbog invertora, na *CS* ulaz periferija dolazi vrednost 0 i pristup periferijama nije moguć. Obrnuto, ako se signal *M*/*IO* postavi na 0, rad sa memorijom je blokiran, a može se pristupati periferijama jer je na njihovm *CS* ulazu vrednost 1 (zbog invertora).

6.1.3 DMA mehanizam

Procesor *Intel* 8086 podržava koncept direktnog pristupa memoriji – *DMA*. Na slici 6.6 prikazan je način povezivanja procesora sa *DMA* kontrolerom kako bi se omogućio transfer podataka između periferija i memorije bez posredovanja procesora.



Slika 6.6 Povezivanje procesora *Intel* 8086 sa *DMA* kontrolerom

Preglednosti radi, na slici nije prikazan leč, ali se podrazumeva da on postoji unutar procesora.

Svaki prenos podataka između periferija i memorije predstavlja *DMA* ciklus. On se odvija na sledeći način:

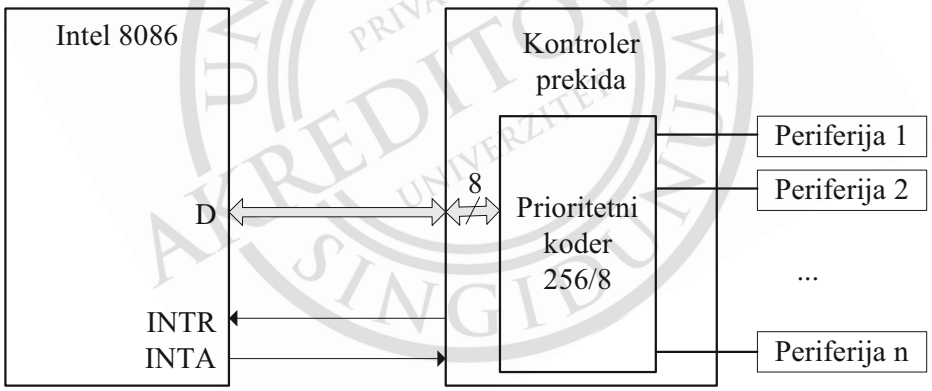
- *DMA* kontroler zahteva od procesora da mu omogući pristup magistrali tako što aktivira sinal *HOLD*
- kada procesor primi signal *HOLD*, završava tekući ciklus na magistrali i aktivira signal *HLDA*, čime predaje magistralu *DMA* kontroleru
- dok *DMA* kontroler koristi magistralu za transfer podataka, procesor ne sme da joj pristupa (nema upisa i čitanja iz memorije), ali može da nastavi sa obradom podataka koji se trenutno u njemu nalaze
- kada završi *DMA* ciklus, *DMA* kontroler deaktivira signal *HOLD*, čime oslobađa magistralu i predaje je procesoru

6.1.4 Mehanizam prekida

Procesor *Intel* 8086 ima mogućnost opsluživanja kako eksternih, tako i internih prekida. Za prijem zahteva za eksternim prekidima predviđena su dva pina procesora: *INTR* (*Interrupt Request*) i *NMI* (*Non Masked Interrupt*).

Linija *INTR* je povezana sa programabilnim kontrolerom prekida (najčešće je to *Intel* 8259). Uloga kontrolera prekida je da prihvata zahteve za prekidima koje upućuju periferije povezane sa njim i da ih prosleđuje procesoru. U datom trenutku kontroler prekida može da dobije jedan ili više zahteva za prekidom. Ukoliko je dobio više zahteva, kontroler pomoću prioritetnog kodera određuje koji od prispelih zahteva ima najviši prioritet i ako je taj zahtev višeg prioriteta od zahteva koji procesor trenutno opslužuje, kontroler aktivira liniju *INTR*. Prioritetni koder je vrsta kodera koja dopušta istovremeno dovođenje aktivnih signala na više ulaza, pri čemu se na izlazu pojavljuje kodirana informacija koja odgovara aktivnom ulazu sa najvišim prioritetom.

Na slici 6.7 prikazan je način povezivanja procesora sa kontrolerom prekida.

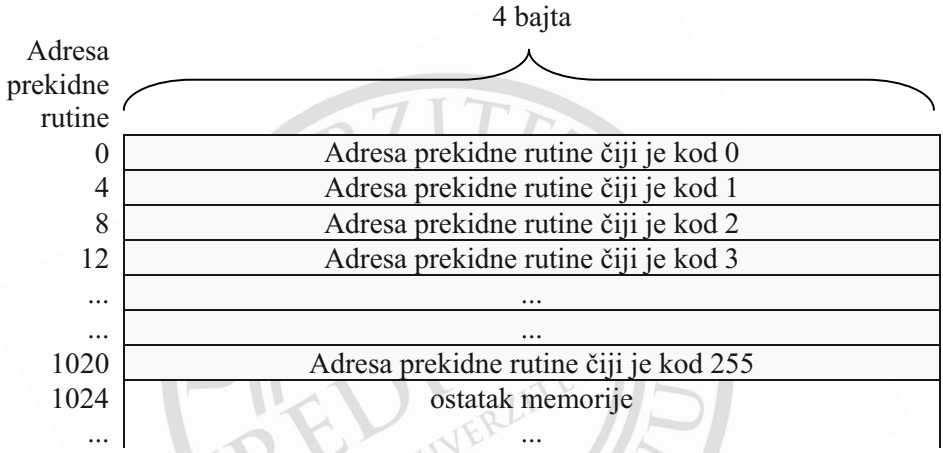


Slika 6.7 Povezivanje procesora *Intel* 8086 sa kontrolerom prekida

Kada dobije signal po liniji *INTR*, procesor prihvata ili ne prihvata zahtev za prekidom u zavisnosti od vrednosti *IF* flega u *PSW* registru. Ako je ovaj fleg postavljen na 1, zahtev se prihvata, a ako je 0, zahtev se ignoriše. Ukoliko procesor prihvati zahtev, najpre mora da završi izvršavanje tekuće instrukcije za šta je potrebno neko vreme. Ovo vreme se naziva vremenom kašnjenja prekida i zavisi od instrukcije koja se trenutno izvršava (najveće je kašnjenje ukoliko su bile u toku instrukcije množenja i deljenja). Nakon završetka instrukcije, procesor odgovara na zahtev aktivranjem linije *INTA*, čime se kontroler prekida obaveštava da na magistralu podataka treba da pošalje 8-bitni podatak koji predstavlja kod prekida i

nosi informaciju o tome koja periferija je zahtevala prekid. Na osnovu dobijenog koda prekida procesor određuje koju prekidnu rutinu treba da pozove radi opsluživanja periferije.

Adresa prekidne rutine dobija se iz tabele prekida (*Interrupt Pointer Table*) na osnovu dobijenog koda prekida. Procesor *Intel* 8086 može da opsluži 256 različitih prekida. Stoga tabela prekida ima 256 ulaza (za svaku vrstu prekida po jedan). Za pamćenje pojedinačnih adresa prekidnih rutina koriste se 4 bajta, pa je veličina tabele prekida 1KB (256 x 4B = 1024B). Tabela prekida se uvek nalazi u prvom kilobajtu memorije kao što je prikazano na slici 6.8.



Slika 6.8 Tabela prekida

Pre nego što pređe na izvršavanje prekidne rutine, procesor mora da sačuva kontekst u kome je izvršavao tekući program. To je neophodno kako bi nakon završetka prekidne rutine procesor mogao da se vrati tamo gde je stao i nastavio sa izvšavanjem programa koji je prekinut. U okviru konteksta procesora, treba sačuvati:

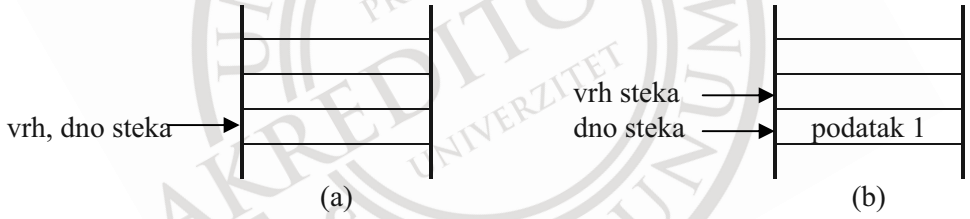
- sadržaj *PSW* registra, jer se u njemu nalazi trenutni status procesora
- sadržaj *IP* registra, jer se u njemu nalazi adresa naredne instrukcije programa

Pamćenje konteksta se vrši automatski, neposredno pre prelaska na izvršavanje prekidne rutine. Takođe, po završetku prekidne rutine, pomenuti registri se automatski reinicijalizuju na stare, sačuvane vrednosti. U praksi, osim *PSW* i *IP* registara, često je neophodno sačuvati i sadržaje pojedinih registara opšte namene. Naime, u zavisnosti od vrste prekida, tokom izvršavanja prekidne rutine, pojedini registri opšte namene mogu promeniti svoj sadržaj. Da bi procesor mogao da se

vrati u prvobitno stanje, programer treba da utvrdi koji registri opšte namene mogu da promene sadržaj, a zatim da eksplicitno, softverskim putem, zahteva pamćenje njihovog sadržaja. Nakon završetka prekidne rutine, programer mora sam da reinicijalizuje ove registre opšte namene, takođe softverskim putem, tj. da ispiše deo koda koji obavlja njihovu reinicijalizaciju.

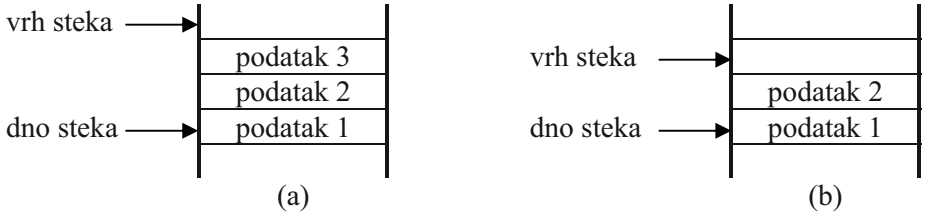
Kontekst procesora se čuva u posebnom delu memorije koji se naziva stek (*stack*). Stek podržava *LIFO* (*last in*, *first out*) disciplinu pristupa, što znači da se sa steka prvo uzimaju podaci koji su poslednji u njega uneti. Deo memorije predviđen za stek ograničen je pokazivačima na memorijske lokacije koje predstavljaju dno steka i vrh steka. Pristup steku, bilo u cilju upisa, bilo radi uzimanja nekog podatka, uvek se vrši preko vrha steka. Stek može biti implementiran na različite načine. Jedna od implementacija prikazana je na slikama 6.9 i 6.10.

Kada je stek prazan, vrh steka i dno steka ukazuju na istu memorijsku lokaciju (slika 6.9 (a)). Pri stavljanju prvog podatka na stek, podatak se upisuje u memorijsku lokaciju na koju ukazuje vrh steka, a zatim se vrh steka pomera na prvu narednu lokaciju (slika 6.9 (b)). Pokazivač na dno steka se ne pomera (uvek ukazuje na istu lokaciju). U ovoj implementaciji vrh steka uvek ukazuje na narednu praznu lokaciju na koju treba upisati sledeći podatak.



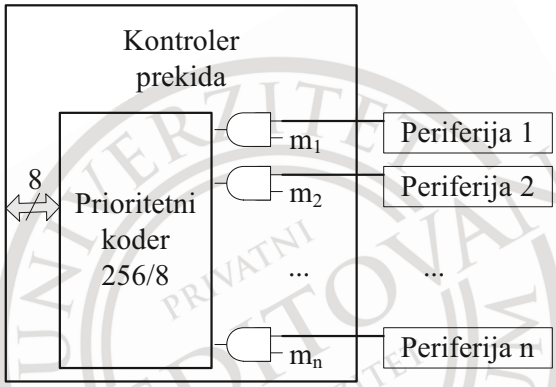
Slika 6.9 Upis podatka u stek

Čitanje podatka sa steka obavlja se tako što se najpre proveri da li je stek prazan (oba pokazivača ukazuju na istu lokaciju). Ako stek nije prazan (slika 6.10 (a)), pokazivač na vrh steka se pomeri na prethodnu lokaciju, a zatim se iz nje preuzima podatak (slika 6.10 (b)). Vrh steka ostaje na praznoj lokaciji iz koje je uzet podatak, tako da postoje uslovi za eventualni budući upis novog podatka.



Slika 6.10 Čitanje podatka sa steka

Procesor *Intel* 8086 ima mogućnost maskiranja prekida koji dolaze po *INTR* liniji. Maskiranje se obavlja tako što procesor šalje kontroleru prekida odgovarajuću komandu koja sadrži masku sa određenim brojem bita m_i, kao što je prikazano na slici 6.11. Svaki bit m_i maskira zahtev za prekidom upućen od strane jedne periferije tako što se zajedno sa zahtevom dovodi na ulaz *I* kola. Ako je vrednost bita maske 1, zahtev za prekidom se prosleđuje na izlaz *I* kola, a ako je bit maske 0, zahtev za prekidom se ignoriše. Kontroler prekida razmatra samo one zahteve koji su prošli kroz *I* kola, tj. došli do prioritetnog kodera i prosleđuje ih procesoru.



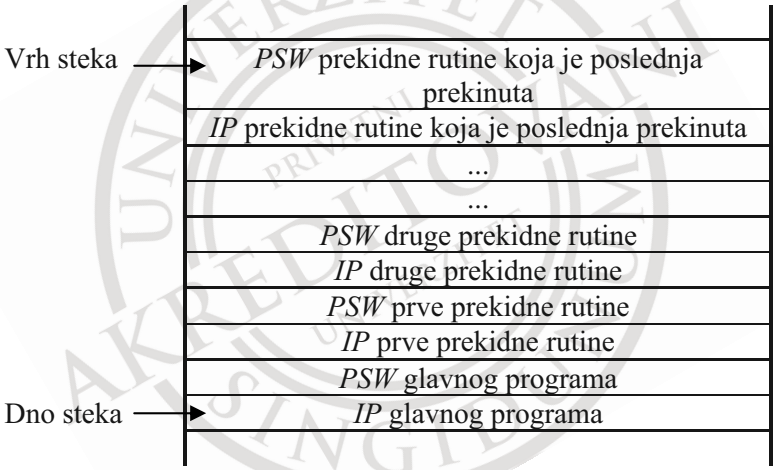
Slika 6.11 Maskiranje prekida

Osim spoljašnjih prekida koji dolaze po *INTR* liniji, procesor *Intel* 8086 može da opsluži i spoljašnje prekide koji dolaze po *NMI* liniji, kao i unutrašnje prekide. Postupak njihovog opsluživanja je isti kao i već opisani postupak za opsluživanje prekida koji stižu po *INTR* liniji.

Prekidi koji stižu po *NMI* liniji nazivaju se nemaskirajućim prekidima. Ovaj naziv potiče od činjenice da ne postoji mogućnost njihovog neprihvatanja. To znači da se oni ne mogu ignorisati ni postavljanjem maske, ni postavljanjem flega *IF* u *PSW* registru. Razlog za ovakav tretman je u tome što se radi o prekidima koji mogu da imaju katastrofalne posledice po procesor. Na primer, oni nastaju kada dođe do gubitka napajanja, greške prilikom transfera na magistrali, greške u radu memorije itd. Prekidi koji dolaze po *NMI* liniji su višeg prioriteta od bilo kog prekida koji stiže po *INTR* liniji.

Unutrašnji prekidi nastaju prilikom izvršavanja instrukcije internog prekida, tj. asemblerske instrukcije *INT*. Ova instrukcija sadrži kod prekida na osnovu koga procesor aktivira izvršavanje odgovarajuće prekidne rutine. Interni prekidi su višeg prioriteta od eksternih prekida.

Zbog velike raznovrsnosti mogućih zahteva za prekidom, uvedena je stroga disciplina u njihovom opsluživanju koja se zasniva na prioritetima. Kao što je rečeno, najviši prioritet imaju interni prekidi, zatim spoljašnji koji dolaze po *NMI* liniji i na kraju spoljašnji koji stižu po *INTR* liniji. Ova hijerarhija u prioritetu je bitna ukoliko tokom izvršavanja jedne prekidne rutine dođe novi zahtev za prekidom. Ako je prispeli zahtev nižeg prioriteta, mora na sačeka da bi bio opslužen. Međutim, ako je višeg prioriteta, procesor prekida izvršavanje tekuće prekidne rutine i prelazi na prekidnu rutinu koja odgovara prispelom zahtevu. Kada je završi, procesor se vraća na rutinu koju je prekinuo i nastavlja sa njenim izvršavanjem. Ovakav način opsluživanja prekida može da ide i u „dubinu“, tj. da postoji više prekidnih rutina čije je izvršavanje u toku. Da bi svi prispeli zahtevi bili korektno opsluženi, pri svakom prelasku na novu prekidnu rutinu neophodno je sačuvati tekući kontekst procesora. On se čuva na steku kao što je prikazano na slici 6.12.



Slika 6.12 Čuvanje konteksta procesora na steku

U ovom primeru, procesor je izvršavao glavni program kada je stigao zahtev za prekidom. Da bi opslužio prekid, procesor je prekinuo glavni program, tekući kontekst stavio na stek i prešao na izvršavanje prekidne rutine. Tokom izvršavanja prekidne rutine, došao je zahtev za prekidom višeg prioriteta, pa je i izvršavanje prekidne rutine moralo biti zaustavljeno. Pre prelaska na sledeću prekidnu rutinu, na steku je sačuvan kontekst koji odgovara trenutnom stanju u kome je prekidna rutina prekinuta. Sličan postupak se ponavlja za sve naredne prekide višeg prioriteta. Kada se prekid najvišeg prioriteta opsluži, sa steka se uzima zapamćeni kontekst koji odgovara prethodnoj rutini i nastavlja se sa njenim izvršavanjem. Kada se opsluže svi prekidi, sa steka se reinicijalizuje kontekst koji odgovara glavnom programu i procesor nastavlja njegovo izvršavanje.

6.2. Počeci razvoja Intel familije procesora

Sedamdesetih godina prošlog veka, na tržištu su se pojavili prvi procesori. U toku svoje kratke istorije (poslednjih četrdesetak godina), prosecori su doživeli ogroman tehnološki napredak. Njihove performanse u pogledu brzine su poboljšane više od hiljadu puta, arhitektura je znatno unapređena, mogućnosti značajno proširene. Ovako intenzivan razvoj bio je praćen često bespoštednim nadmetanjem firmi koje su proizvodile procesore za prevlast na svetskom tržištu.

Kada je vodeća kompanija u proizvodnji računara *IBM* projektovala svoj prvi personalni računar (*PC – Personal Computer*), postojala je dilema oko izbora procesora koji će biti u njega ugrađen. Na raspolaganju su bila dva proizvođača: *Intel* i *Motorola*. Iako su *Motorola* procesori, po mišljenju *IBM*-ovih inženjera bili prihvatljiviji jer su imali bolje karakteristike, rukovodstvo *IBM*-a je izabralo Intel. Pri izboru su prevagnule sledeće činjenice: *Intel*-ov procesor je bio jeftiniji, kompatibilan sa svim ranijim verzijama, a garantovana je i kompatibilnost sa svim budućim generacijama procesora, *IBM* je dobio prava za proizvodnju *Intel*-ovog procesora, za *Intel*-ov procesor je postojao gotov operativni sistem *CP*/*M* firme *Digital Research* koji je u to vreme bio standard za mikroračunare.

U novembru 1971.godine proizveden je prvi *Intel*-ov procesor sa oznakom 4004 koji je bio upakovan u samo jedan čip (videti sliku 6.13). Za njegovu izradu je korišćena tada dostupna 10μm tehnologija. Radni takt ovog procesora je bio od 0.4 do 0.8MHz, imao je 4-bitnu magistralnu podataka i 640 bajta spoljašnje memorije.

Slika 6.13 Izgled kućišta procesora *Intel* 4004

Već i aprilu 1972.godine proizveden je procesor *Intel* 8008 sa 8-bitnom magistralom i 16KB spoljašnje memorije. Radni takt nije bitno promenjen.

U prvi komercijalno rapoloživ računar *Altair* ugrađen je Intel-ov procesor sledeće generacije sa oznakom 8080. Za samo nekoliko meseci prodato je na hiljade komada ovih računara, što im je donelo prvo mesto na listi prodaje. Ovaj procesor je radio na taktu od 2MHz i imao je 64KB memorije.

Značajni pomak desio se 1978.godine kada je proizveden prvi 16-bitni procesor *Intel* 8086 koji je *IBM* ugradio u svoj prvi personalni računar *PC*/*XT*. U to

vreme se koristila 3μm tehnologija. Ovaj procesor se proizvodio sa taktom u verzijama 2, 5, 8 i 10MHz, imao je 20-bitnu adresnu magistralu koja je adresirala 1MB memorije. Interesantno je da se u tom trenutku nije uočavala buduća potreba za većom memorijom.

Godine 1982. nastao je 16-bitni procesor *Intel* 80286, koga je *IBM* ugradio u svoj *Advanced Technology* personalni računar *PC/AT* koji je za šest godina proizvodnje doživeo 15 miliona instalacija, čime je prevazišao sve ranije rekorde. To je bio prvi pravi procesor jer je uveo koncept zaštićenog moda (*protected mode*) koji je omogućavao da više programa rade istovremeno, nezavisno jedan od drugog (*multitasking*). Ova mogućnost je kasnije našla primenu u operativnim sistemima, kao na primer u *Windows OS*. Radni takt ovog procesora je dostizao i do 20MHz i imao je 16MB RAM.

Prvi 32-bitni procesor *Intel* 80386 (videti sliku 6.14) proizveden je 1985.godine i predstavlja veliki tehnološki napredak u *Intel*-u. Razvijena je cela familija ovih procesora. Radni takt je bio do 33MHz, a poslednja verzija je mogla da adresira 4GB RAM. U ovom procesoru prvi put je korišćen *pipeline* princip za izvršavanje instrukcija, svi čipovi u ovoj familiji su bili *pin-to-pin* kompatibilni, a postojala je softverska kompatibilnost sa familijom 286 (mogli su da se koriste programi za 286), uvedeni su standardi u razvoju čipova koji su se kasnije primenjivali.

Slika 6.14 Izgled kućišta procesora *Intel* 80386

Početkom 1989.godine nastao je 32-bitni procesor *Intel* 80486 čiji je radni takt u poslednjim verzijama dostizao 100MHz. Bio je duplo brži od svog prethodnika jer je imao poboljšanu arhitetkturu. To je prvi procesor sa integrisanom keš memorijom veličine 8KB u koju su se, korišćenjem *pipeline* mehanizma, stavljali sledeća instrukcija i podatak, tako da procesor nije morao da pristupa spoljašnjoj memoriji. Ovo je bio veliki pomak u brzini obrade podataka. Veliko poboljšanje je bilo i uvođenje integrisanog matematičkog koprocesora koji je bio namenjen izvršavanju aritmetičkih operacija u jednom taktu nad brojevima predstavljenim u pokretnom zarezu.

Godine 1993. otpočinje proizvodnja novih generacija *Intel*-ovih procesora pod novim imenom *Pentium*. Do promene imena (umesto 586) došlo je zbog pravnih problema.

Pentium I je imao radni takt do 233MHz, dva nezavisna bloka keš memorije (za naredbe i podatke), a mogao je da izvršava dve instrukcije u toku jednog takta. Takođe, dva ovakva procesora mogla su da rade zajedno u sistemu. Najpoznatiji u ovoj familiji bili su *Pentium Pro* i *Pentium MMX*.

Pentium II predstavlja neuspeli pokušaj objedinjavanja najboljih osobina prethodnika. Imao je takt do 450MHz, dva bloka keš memorije različitih brzina i kapaciteta (32KB *L1* keša i do 512KB *L2* keša koji je znatno sporiji od *L1*, ali i dalje mnogo brži od operativne memorije).

Pentium III je imao proširen skup *MMX*-ovih instrukcija (za 70 novih instrukcija) čime je poboljšano procesiranje 3D animacija. Radni takt ovog procesora je dostizao i do 1GHz. Da bi poboljšao sigurnost *online* transakcija, *Intel* je odlučio da na svaki procesor ovog tipa ugradi serijski broj čipa (*PSN – processor serial number*) koji je mogao da se pročita preko mreže ili *Internet*-a. Pošto su korisnici smatrali da je to napad na njihovu privatnost, *Intel* je bio prinuđen da dopusti isključenje *PSN*-a putem *BIOS*-a.

Pentium IV je imao novu, *NetBurst* arhitekturu koja je omogućavala povećavanje brzine u budućnosti. Ova arhitektura obuhvatala je četiri nove tehnologije: *Hyper Pipelined Technology* (mogućnost povećanja brzine procesora proširenjem *pipeline*-a, tako da se može izvršavati više instrukcija istovremeno), *Rapid Execution Engine* (kao posledica proširenja *pipeline*-a, uvedene su dve *ALU* koje rade duplo brže), *Execution Trace Cache* (urađene su neophodne prepravke na keš memoriji) i 400MHz sistemsku magistralu. Radni takt ovog procesora je dostizao do 3.6GHz.

Vežbanja

1. U kom obliku se podaci predstavljaju u računaru? Šta su unarne, a šta binarne operacije? Koje klase operacija postoje? Ukratko opisati svaku od njih.

2. Objasniti postupak izvršavanja programa od strane procesora.

3. Čemu služi magistrala? Koje vrste magistrala postoje i u čemu se one razlikuju? Kako širina magistrale utiče na kapacitet memorije?

4. Prikazati i objasniti strukturu programske instrukcije. Dati primer jednoadresne i dvoadresne instrukcije.

5. Objasniti *pipeline* mehanizam izvršavanja instrukcija.

6. Šta predstavlja direktan pristup memoriji i kada se primenjuje?

7. Koji problem rešava mehanizam prekida i koje rešenje nudi?

8. Opisati postupak opsluživanja prispelog zahteva za prekidom.

9. Navesti vrste prekida, ukratko ih objasniti i uporediti po prioritetu.

10. Koje su osnovne jedinice u strukturi procesora *Intel* 8086? Ukratko opisati njihovu ulogu.

11. Navesti elemente u strukturi izvršne jedinice (*Execution Unit*) procesora *Intel* 8086 i objasniti ulogu svakog od njih.

12. Navesti elemente u strukturi jedinice za spregu sa magistralom (*Bus Interface Unit*) procesora *Intel* 8086 i objasniti ulogu svakog od njih.

13. Šta je *PSW* (*Processor Status Word*) registar i čemu služi? Detaljno opisati njegov sadržaj. Da li korisnik može da menja sadržaj ovog registra?

14. Objasniti princip rada procesora *Intel* 8086.

15. Objasniti postupak upisa podatka u memoriju na primeru okruženja sa procesorom *Intel* 8086 (dati sliku). Kako se rešava problem multipleksirane magistrale?

16. Objasniti postupak čitanja podatka iz memorije na primeru okruženja sa procesorom *Intel* 8086 (dati sliku). Kako se rešava problem multipleksirane magistrale?

17. Objasniti kako se obavlja *DMA* transfer (*DMA* ciklus) na primeru okruženja sa procesorom *Intel* 8086.

18. Navesti vrste prekida koje može da opsluži procesor *Intel* 8086, ukratko ih objasniti i uporediti po prioritetu.

19. Detaljno opisati postupak opsluživanja zahteva za prekidom prispelog po *INTR* liniji procesora *Intel* 8086 (dati sliku).

20. Šta predstavlja tabela prekida (*Interrupt Pointer Table*)? Gde se nalazi i šta sadrži (dati izgled tabele) u slučaju procesora *Intel* 8086?

21. Zašto je neophodno sačuvati kontekst procesora u slučaju pojave zahteva za prekidom? Šta sadrži kontekst procesora *Intel* 8086 i gde se čuva?

22. Šta je stek (*stack*)? Objasniti princip njegovog funkcionisanja na primeru upisa, odnosno čitanja nekog podatka sa steka.

23. Objasniti pojam maskiranja prekida na primeru okruženja sa procesorom *Intel* 8086.

24. Opisati disciplinu zasnovanu na prioritetima koja se mora poštovati prilikom opsluživanja većeg broja prispelih zahteva za prekidom.

UNIVERZITET
PRIVATNI
AKREDITOVANI
UNIVERZITET
SINGIDUNUM

7 Personalni računar

Personalni računar se sastoji od velikog broja hardverskih komponenata međusobno povezanih tako da omogućavaju efikasno izvršavanje različitih programa u cilju obrade određenih podataka. Centralni deo personalnog računara je matična ploča (*motherboard*). Ona objedinjuje ostale komponente računara, kontroliše i sinhroniše njihov rad. Na matičnoj ploči nalaze se konektori preko kojih se u računarski sistem uključuju procesor i radna memorija, kao njegovi najbitniji delovi. Tu je i priključak za hard disk kao najvažniji vid spoljašnje memorije. Na raspolaganju su i razne vrste portova za priključivanje perifernih ulazno/izlaznih jedinica, kao i razne vrste ekspanzionih slotova za proširivanje mogućnosti računarskog sistema. Osim objedinjujuće, matična ploča ima i kontrolnu funkciju koju realizuje pomoću brojnih kontrolera integrisanih u okviru čipseta. Na njoj se nalazi i *BIOS*, program koji upravlja procesom startovanja računara.

Osim matične ploče, važne komponente personalnog računara su, pre svega procesor i operativna memorija, a zatim razne vrste spoljašnjih memorija (hard disk, *CD*, *DVD* itd.), kao i ulazno/izlazne jedinice (monitori, štampači itd.).

7.1. Matična ploča

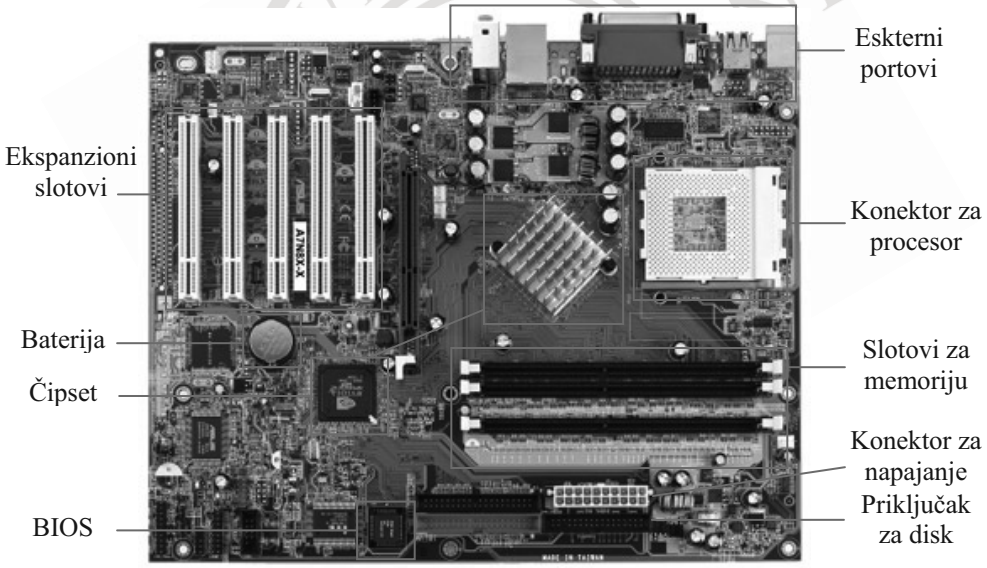
Matična ploča služi za povezivanje i sinhronizaciju rada najbitnijih delova računara. Na njoj se nalaze procesor i operativna memorija. Ostale komponente, kao što su spoljašnje memorije, razni kontroleri i periferne jedinice priključuju se na matičnu ploču bilo kablovima, bilo *plug-in* karticama koje se stavljaju u odgovarajuće slotove. Moderni računari imaju tendenciju integrisanja periferija u samu matičnu ploču.Važna komponenta na matičnoj ploči je i čipset koji određuje karakteristike i mogućnosti matične ploče.

Na matičnoj ploči nalaze se:

- konektor za procesor

- slotovi za operativnu memoriju
- priključak za hard disk
- ekspanzioni slotovi
- čipset
- priključci eksternih portova
- *BIOS* i *CMOS* sa baterijom
- sistemski sat
- konektor za napajanje i naponski regulatori napajanja za procesor
- konektori za *LE* diode i tastere za indikaciju koji se nalaze na prednjoj strani kućišta računara

Slika 7.1 prikazuje matičnu ploču na kojoj su označene navedene komponente.



Slika 7.1 Matična ploča

Brzina rada matične ploče zavisi od takta sistemske magistrale i takta na kome radi čipset. Kod prvih personalnih računara, takt matične ploče je bio isti kao i takt tadašnjih procesora. Međutim, tokom kasnijeg razvoja, takt matične ploče nije pratio napredak u brzini procesora, tako da su današnji procesori mnogo brži od matične ploče. Razlog ovako malog napretka u brzini matične ploče jeste u tome što se njene dimenzije nisu bitno promenile u odnosu na ploče kod prvih

personalnih računara. Zbog velikih dimenzija matične ploče, vreme koje je potrebno za propagaciju signala sa kraja na kraj ploče nije zanemarljivo i utiče na maksimalni takt na kome ploča može da radi.

Prlikom izbora matične ploče koja će se biti ugrađena u računar treba povesti računa ne samo o njenim mogućnostima, već i o njenom kvalitetu. Razlozi za to su sledeći:

- nadogradnja (*upgrade*) matične ploče nije moguća; na primer, ako matična ploča nema neki od ekspanzionih slotova, nije ih moguće naknadno dodati
- neispravnost na ploči obično dovodi do prestanka rada celog sistema; ponekad, neispravnost ploče može da dovede i do otkaza drugih komponenata u računarskom sistemu

7.1.1 Konektori

Procesor i operativna memorija uključuju se u računarski sistem preko odgovarajućih konektora koji se nalaze na matičnoj ploči računara.

Postoji više standarda za veličinu, oblik i raspored pinova konektora za procesor (na primer *Socket* 478, *Socket* 775 itd.). Zbog velikog broja pinova na procesorskim čipovima, konektori za procesor primenjuju *ZIF – Zero Insert Force* sistem za priključivanje čipa na konektor. *ZIF* sistem omogućava da se procesor uz primenu minimalne sile stavi u konektor, a zatim „zaključa“ pomoću polugice koja se nalazi sa strane konektora. Pritiskanjem polugice ostvaruju se električni kontakti između pinova procesora i konektora, a takođe se uspostavlja i čvrsta mehanička veza između njih.

Na matičnoj ploči obično postoje 2, 3 ili 4 slota za priključivanje operativne memorije. Slotovi mogu biti različiti u zavisnosti od tipa memorije koji je predviđen za datu matičnu ploču.

7.1.2 Ekspanzioni slotovi

Ekspanzioni slotovi predstavljaju priključke na matičnoj ploči (slotove) koji služe za proširenje mogućnosti računarskog sistema. Na njih se priključuju periferni uređaji kao što su: grafički adapter (video kartica), modem, zvučna kartica, *SCSI* (*Small Computer System Interface*) kontroler, razne vrste specijalizovanog hardvera itd.

Postoji više vrsta ekspanzionih slotova koje se međusobno razlikuju po brzini i mogućnostima koje pružaju. Danas su u širokoj upotrebi sledeći ekspanzioni slotovi:

- *PCI – Peripheral Component Interconnect*

- *PCI Express – Peripheral Component Interconnect Express*
- *AGP – Accelerated Graphics Port*

Na matičnoj ploči može se nalaziti samo jedan *AGP* slot, dok *PCI* i *PCI Express* slotova može biti više. Po pitanju brzine, najbrži je *PCI Express* slot.

***PCI* slot** je najčešće korišćeni ekspanzioni slot. On služi za povezivanje perifernih jedinica kao što su modem, zvučna kartica, mrežni adapter, *SCSI* kontroler, *PCI* grafičke kartice i dr. na matičnu ploču. Može biti integrisan, ili u vidu slota u koji se stavlja *PCI* kartica (videti sliku 7.2). Matična ploča obično ima 2 do 6 *PCI* slotova.

PCI magistrala je u osnovi 32-bitna, ali postoje i 64-bitne verzije. Takt na magistrali je 33MHz, a noviji *PCI* slotovi podržavaju takt od 66MHz. Ako se koristi 64-bitna magistrala podataka i takt od 66MHz, na magistrali se teorijski može ostvariti protok do 512MB/s. Ovo je u praksi veoma teško postići, tako da se ova magistrala obično koristi na znatno nižim brzinama (oko 266MB/s).

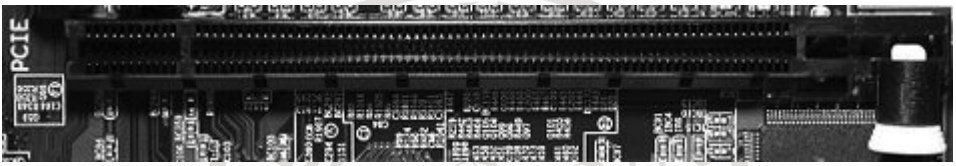
Slika 7.2 Tipična 32-bitna *PCI* kartica za *SCSI* adapter

PCI Express je standard koji je uveo *Intel* 2004.godine. Nastao je iz potrebe za brzim ekspanzionim slotom koji bi objedinio sve postojeće standarde. Ideja je bila da matične ploče imaju samo jednu vrstu ekspanzionog slota čija bi brzina zavisila od kartice koja se u njemu nalazi.

Da bi se ostvario postavljeni zahtev, uvedena je nova koncepcija prenosa podataka za ekspanzione slotove, a to je serijski prenos podataka. Podaci se serijski prenose parom jednosmernih linija koji se naziva „*lane*". Na jednom *PC Express* priključku može da bude više „*lane*"-ova, a u zavisnosti od njihovog broja, slot dobija nastavak u imenu. Na primer, ako slot ima 8 serijskih parova, ime mu je *PC Express* x8. Na slikama 7.3 i 7.4 prikazani su *PCI Express* slotovi x1 i x16. Brzina

prenosa po jednom „*lane*"-u za osnovnu verziju *PCI Express* slota (*PCIe* 1.1) je konstantna i iznosi 250MB/s. Ako slot koristi više „*lane*"-ova, brzina mu se proporcionalno povećava. Tako, na primer, *PCI Express* x8 može da prenosi podatke brzinom od 8x250MB/s = 2000MB/s. Novije verzije, PCIe 2.0 iz 2007.godine i PCIe3.0 koja je predložena za 2010.godinu, ostvaruju brzine prenosa od 500MB/s i 1GB/s, respektivno.

Slika 7.3 *PCI Express* x1 slot



Slika 7.4 *PCI Express* x16 slot

***AGP* slot** se koristi isključivo za priključivanje grafičke kartice na matičnu ploču. Nastao je kao odgovor na sve veće zahteve po pitanju brzine grafičkih kartica, posebno u primenama kao što su 3D animacije ili računarske igrice.

AGP slot podržava 32-bitnu magistralu i osnovni takt od 66MHz. Pri ovom taktu, brzina razmene informacija sa *AGP* karticom je 256MB/s i ona predstavlja *AGP* 1x standard. Standardi *AGP* 2x, 4x i 8x imaju 2, 4, odnosno 8 puta veći protok podataka zahvaljujući povećanju takta na magistrali na 133MHz, 266MHz, odnosno 533MHz.

7.1.3 Čipset

Na početku razvoja računara, na matičnim pločama se nalazio veliki broj pojedinačnih čipova koji su služili za kontrolu rada računara i upravljanje periferijama. Tu su, kao posebni čipovi, bili izdvojeni generator takta, sistemski časovnik, kontroler prekida, *DMA* kontroler itd. Ovako veliki broj čipova značajno je uticao na cenu prvobitnih matičnih ploča. Sa razvojem integrisane tehnologije, pojavila se ideja da se svi čipovi integrišu u samo nekoliko integrisanih kola koja su dobila naziv čipset. Tako je čipset postao jedan od najbitnijih delova matične ploče. Takt na kome radi čipset je obično nekoliko puta brži od takta na kome radi

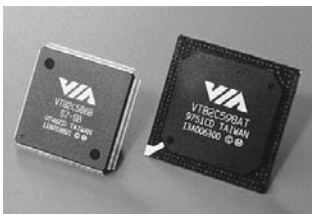
sistemska magistrala, pa mogućnosti čipseta značajno utiču na performanse celog sistema.

Dakle, čipset predstavlja skup pojedinačnih kontrolera koji ostvaruju veze između glavnih delova računara, kao i veze prema periferijama i uređajima priključenim na ekspanzione slotove. U čipsetu se nalaze:

- *DMA* kontroler koji upravlja *DMA* prenosom podataka
- kontroler prekida koji upravlja prekidima
- kontroler memorije koji određuje tip i maksimalni kapacitet operativne memorije koja može da instalira u računaru, utiče na brzinu rada sa njom, upravlja keš memorijom i dr.
- *EIDE* i *S-ATA* kontroleri koji kontrolišu protok podataka na *Enhanced IDE* i *Serial ATA* magistralama koje se koriste za povezivanje hard diskova i optičkih uređaja za skladištenje podataka (kao na primer *CD-ROM* i *DVD*) na sistem
- kontroleri ekspanzionih slotova koji kontrolišu rad uređaja priključenih na ove slotove
- kontroleri portova koji kontrolišu uređaje priključene na ove portove
- sat realnog vremena

Čipset se obično realizuje u vidu dva integrisana kola (videti sliku 7.5):

- severnog mosta (*north bridge*), koji se nalazi u blizini komponenata koje zahtevaju brz protok informacija, kao što su procesor, memorija i *AGP* ekspanzioni slot; u ovom čipu su objedinjeni kontroleri koji upravljaju radom ovih delova sistema
- južnog mosta (*south bridge*), koji objedinjuje kontrolere periferija (hard diskova, optičkih uređaja i sl.) i kontrolere portova; ovo je obično manji čip i udaljen je od glavnih delova matične ploče



Slika 7.5 Čipset: severni i južni most

Danas u svetu postoji veliki broj proizvođača čipsetova, među kojima su najpoznatiji *VIA*, *nForce*, *Intel*, *SiS*, *ALI* itd. Prva tri od navedenih proizvođača pokrivaju oko 95% svetskog tržišta čipsetova.

7.1.4 Portovi

Portovi služe za jednostavno i brzo povezivanje različitih perifernih uređaja na personalni računar. Preko portova povezuju se miš, tastatura, štampač, skener, digitalna kamera i drugi uređaji.

U zavisnosti od načina prenosa podataka koji podržavaju, portovi se svrstavaju u dve grupe:

- paralelne portove
- serijske portove

Paralelni port omogućava prenos binarnih podataka u kome se svi bitovi podatka šalju istovremeno po odgovarajućim linijama. To je jedan od portova kojim su bili opremljeni prvi personalni računari. On je prvobitno bio namenjen samo povezivanju štampača na računar. Po polaznom konceptu, pomoću njega je bio moguć prenos podataka samo u jednom smeru i to od računara ka uređaju koji je na njega priključen. Kasnija rešenja omogućila su protok podataka u oba smera, tako da je paralelni port mogao da se koristi i kao ulazni.

Personalni računar podržava rad do tri paralelna porta koji se označavaju sa *LPT1*, *LPT2* i *LPT3*. Ovi portovi su 8-bitni jer se prenos podataka obavlja istovremeno (paralelno) preko 8 linija. Osim linija za podatke, paralelni port koristi još nekoliko dodatnih linija za sinhronizaciju prenosa i signalizaciju, tako da je ukupan broj linija 18. Konektor za paralelni port je 25-pinski i nalazi se na zadnjoj strani kućišta računara.

Preko paralelnog porta mogu se prenositi informacije na udaljenosti do 4m. Osim štampača, na ovaj port mogu se priključivati i drugi uređaji, kao na primer programatori različitih vrsta permanente memorije.

Serijski port podržava princip prenosa podataka bit po bit. Obično se za serijski prenos koriste dve linije, pri čemu je jedna rezervisna za prijem, a druga za slanje podataka. Zbog prenosa bit po bit, brzina prenosa (protok) u slučaju serijske veze izražava se u bitima u sekundi (b/s ili bps), a ne u bajtovima u sekundi (B/s) kao kod paralelnog prenosa.

Korišćenje serijskog porta je pogodno u situacijama kada su rastojanja na koja je potrebno preneti informaciju relativno velika, pri čemu se ne zahteva velika

brzina prenosa. Maksimalna brzina prenosa preko serijskog porta je reda 100 Kb/s, a pri toj brzini podaci se mogu preneti na udaljenost do 10m.

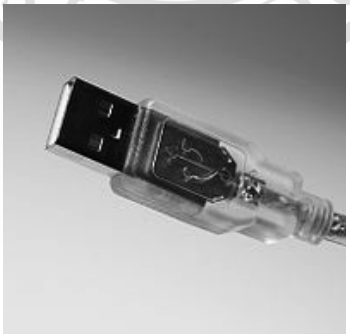
U poređenju sa paralelnim prenosom, serijski prenos zahteva manji broj linija, pa je povezivanje jeftinije i sigurnije, a verovatnoća greške u prenosu manja. Naravno, brzina prenosa koja se može ostvariti je znatno manja.

Personalni računari koriste standardni *RS*-232 serijski port. Obično postoje jedan ili dva takva priključka na zadnjoj strani kućišta računara, mada ih po potrebi može biti i više. Osim dve linije za prenos podataka, *RS*-232 port koristi još 6 linija za sinhronizaciju prenosa i jednu liniju za masu. Ovo ukupno čini 9 linija. Konektori za serijski port su uglavnom 9-pinski, ali se mogu naći i oni koji imaju 25 pinova. U tom slučaju se od 25 koristi samo 9 pinova.

RS-232 serijski port se koristi za povezivanje sporih periferija kao što su eksterni modem ili *RS*-232 miš. Zbog izražene otpornosti na smetnje, ovaj port se pokazao vrlo pouzdanim u industriji i drugim profesionalnim primenama.

U računaru postoje različite vrste portova, od kojih će neke biti opisane u nastavku.

USB* port** (Universal Serial Bus***) je univerzalna serijska magistrala i predstavlja jedan od novijih interfejsa u personalnim računarima (slika 7.6). Koncipiran je kao fleksibilno, ekonomično i jednostavno rešenje povezivanja velikog broja perifernih jedinica (do 127), kao što su: fleš memorije, štampači, skeneri, modemi, tastature, miševi, digitalni fotoaparati, digitalne kamere itd.



Slika 7.6 Port USB Series „A“ plug

Personalni računari obično imaju 2 do 6 *USB* priključaka, ali se njihov broj može povećati dodavanjem tzv. *USB* haba (*USB hub*), čiji je izgled dat na slici 7.7.

USB hab se priključuje na jedan od *USB* priključaka računara, a zatim se na njega može priključiti nekoliko *USB* uređaja (obično 4 do 8).

Slika 7.7 *USB hub*

USB port ima četiri linije od kojih se dve koriste za prenos podataka, a preostale dve za prenos napona napajanja za uređaje koji se priključuju na port. Stoga se uređaji mogu povezivati i dok su personalni računar ili sam uređaj uključeni.

Postoji više verzija *USB* porta: *USB* 1, *USB* 2 i *USB* 3. Brzina prenosa podataka kod *USB* porta je od 12Mb/s (*USB* 1) do 4.8Gb/s (*USB* 3), a udaljenost na koju je moguće preneti podatke je do 5m.

***FireWire* port (*IEEE* 1394)** je veoma brz port koji podatke prenosi serijski, brzinama od 100 do 3200 Mb/s. *FireWire* je naziv zaštićen od strane kompanije *Apple*, dok zvanični naziv, standard *IEEE* 1394, potiče od *IEEE – Institute of Electrical and Electronic Engineers*.

Ovaj port pruža napajanje uređaju koji je priključen na njega, tako da može nesmetano da se priključuje i isključuje tokom rada računara. Na port se može priključiti veliki broj uređaja, a udaljenost do koje se mogu preneti podaci je 4.5m.

***PS*/2 port** služi za povezivanje tastature i miša na *PC*. Njegov naziv potiče od *IBM Personal System*/2 serije personalnih računara u kojoj je uveden 1987.godine.

PS/2 je serijski port kod koga se prenos podataka obavlja preko 6-pinskog konektora koji se nalazi na zadnjoj strani kućišta računara. Sam prenos je vrlo spor, što odgovara brzini periferija kojima je namenjen.

Da bi se smanjila mogućnost pogrešnog povezivanja tastature i miša, *PS* konektori na kućištu računara obojeni su istom bojom kao i konektori na tastaturi, odnosno mišu. Za tastaturu je rezervisana ljubičasta boja, a za miša zelena.

Infracrveni port (***IrDA* port**, *IrDA* označava ***Infra**red **D**ata **A**ssociation*, grupu proizvođača uređaja koji su razvili ovaj standard) omogućava prenos podataka svetlosnim putem. To je sasvim novi pristup u odnosu na uobičajeno korišćenje bakarnih vodova (kablova) za prenos podataka.

Infracrvena svetlost je pogodna za prenos podataka zato što je njen spektar nevidljiv za ljudsko oko, a osim toga u prirodi postoji malo izvora ove svetlosti koji bi mogli da ometaju prenos.

Da bi se obavio prenos podataka između računara i periferije, potrebno je da oba uređaja imaju infracrvene transivere. Infracrveni transiver *desktop* računara povezuje se na konektor za infracrveni port koji se nalazi na matičnoj ploči. U slučaju *laptop* računara, transiver se nalazi u sklopu računara, tako da nije potreban nikakav dodatni hardver. Takođe, postoje i vrlo jeftini infracrveni transiveri koji se priključuju na *USB* port. Osim ovoga, za prenos podataka neophodno je obezbediti i optičku vidljivost između transivera na računaru i transivera na periferiji.

Udaljenost na koju se mogu preneti podaci na ovaj način nije velika i iznosi par metara. Infracrveni port podržava manje brzine prenosa (do 16Mb/s)

Infracrveni port se najčešće koristi za razmenu podataka između računara i mobilnog telefona ili za slanje podataka na štampač koji ima ugrađen infracrveni port.

7.1.5 BIOS i CMOS

BIOS (*Basic Input/Output System*) predstavlja program na najnižem nivou koji se aktivira prilikom startovanja računara. On se učitava pre operativnog sistema i uspostavlja vezu između hardvera i softvera u sistemu. Nakon uključenja, prve instrukcije koje računar izvršava su one koje dobija iz *BIOS*-a.

BIOS podržava širok spektar funkcija:

- obezbeđuje pristup uređajima za skladištenje podataka (na primer hard disku) i tako omogućava učitavanje operativnog sistema (OS) sa nekog od njih; pomoću opcije *BIOS*-a *Boot Device* može se definisati uređaj za skladištenje sa koga će biti učitan OS; takođe, moguće je zadati i listu više uređaja za skladištenje koja bi se ispitivala kako bi se pronašao prvi uređaj iz liste na kome se nalazi OS koji bi zatim bio učitan
- obavlja autodetekciju i formatiranje hard diska
- vrši *power-on self* test
- određuje način i brzinu pristupa operativnoj memoriji

- dozvoljava ili zabranjuje upotrebu keš memorije
- konfiguriše ekspanzione slotove kao i eksterne portove koji se nalaze na matičnoj ploči
- setuje takt procesora i matične ploče
- pruža mogućnost *PnP* (*Plug and Play*), tj. dopušta dodavanje novih jedinica bez rekonfigurisanja ili intervencije korisnika u razrešavanju mogućih konflikata
- konfiguriše matičnu ploču

BIOS se nalazi na matičnoj ploči personalnog računara u memoriji permanentnog tipa. Korišćenje memorije ovog tipa je naophodno da bi nakon isključivanja računara, sadržaj *BIOS*-a ostao zapamćen.

Sadržaj *BIOS* programa se, po potrebi, može promeniti (*upgrade*). Način promene *BIOS*-a zavisi od vrste memorije u kojoj se nalazi. Ipak, njegova izmena se ne preporučuje bez preke potrebe, jer u nekim slučajevima to može da dovede do trajne neupotrebljivosti matične ploče.

Parametri i opcije koje su podešene u *BIOS*-u čuvaju se u memoriji malog kapaciteta (reda stotinu bajtova) koja se naziva *CMOS* (*Complementary Metal-Oxide-Semiconductor*). To je nepermanentna memorija čiji bi se sadržaj izgubio u slučaju da ostane bez napajanja. Da se to ne bi desilo, na matičnoj ploči postoji baterija koja, dok je računar isključen, obezbeđuje napajanje *CMOS*-u. Pošto je potrošnja *CMOS* memorije veoma mala, vek trajanja baterije je od 5 do 10 godina.

Osim napajanja *CMOS*-a, uloga baterije je i da obezbedi rad sata realnog vremena dok je računar isključen. Netačan rad sata računara prvi je znak da bi bateriju trebalo zameniti novom.

U poslednje vreme, *BIOS* program, kao i podešeni parametri čuvaju se u istoj fleš memoriji.

7.2. Procesor

Među proizvođačima savremenih procesora za personalne računare, izdvojile su se dve kompanije koje pokrivaju najveći deo svetskog tržišta. To su *Intel* i *AMD*. *Intel* je u vrhu po proizvodnji najnovijih i najmodernijih procesora, dok je u klasi procesora srednje brzine *AMD* uspeo da postigne i zadrži bolji odnos brzina/cena.

Kod savremenih procesora, proizvođačima nije ostalo još puno prostora za povećavanje takta na kome radi procesor, tako da se u poslednje vreme mnogo više pažnje posvećuje poboljšanju unutrašnje arhitekture procesora.

Glavni pravci razvoja novih procesora kreću se ka:

- povećanju veličine keš memorije i njenom boljem iskorišćenju
- korišćenju većeg i preciznijeg skupa instrukcija
- usavršavanju tehnologija kao što je *pipeline* i sl.
- razvoju procesora sa više jezgara (*multicore*)

7.3. Memorije

Osnovna funkcija memorijske jedinice je čuvanje programa i podataka. To se može postići primenom različitih hardverskih komponenata. U personalnom računaru, razlikuju se tri klase memorijskih jedinica:

- operativna memorija, koja se koristi prilikom izvršavanja programa
- spoljašnje memorije, koje se obično koriste za arhiviranje podataka
- keš (*cache*) memorija, koja služi za povećanje brzine rada računarskog sistema

7.3.1 Operativna memorija

Operativna memorija služi za smeštanje velike količine podataka koji se koriste u radu računara. Ona se realizuje kao *RAM – Random Access Memory*.

U operativnoj memoriji nalaze se operativni sistem koji računar koristi, kao i svi programi koje je korisnik trenutno aktivirao. Nakon završetka rada programa, memorija koju je program koristio oslobađa se i stavlja na raspolaganje drugim programima.

Operativna memorija je nepermanentnog tipa, tako da se isključivanjem računara njen sadržaj nepovratno gubi. Zato je bitno da se svi dokumenti na kojima se radi dok je računar uključen snime na neki od uređaja ili medijuma za trajno skladištenje podataka (hard disk, *CD-ROM* i sl.) pre nego što se računar isključi. U suprotnom, postoji rizik od trajnog gubitka podataka (oni mogu biti sačuvani samo ako ih je operativni sistem prethodno *backup*-ovao na hard disk).

Da bi se postigao što veći kapacitet, operativna memorija je dinamičkog tipa. Za ispravan rad ove vrste memorije potrebno je neprestano „osvežavanje“ njenog sadržaja za šta su zaduženi specijalni kontroleri.

Količina operativne memorije kojom računar raspolaže je veoma bitna za performanse računara. Ukoliko računar nema dovoljno *RAM* memorije, on će deo hard diska proglasiti za tzv. virtuelnu memoriju. Sve one podatke koje ne može da

smesti u operativnu memoriiju smestiće na hard disk (pod *Windows XP* operativnim sistemom u fajl pod imenom *pagefile.sys*). Ovime, ne samo da se gubi deo prostora na hard disku, nego se i s obzirom da je hard disk znatno sporiji od memorije, usporava pristup ovim podacima. Kako je pristup podacima na hard disku i do 100 puta sporiji nego pristup podacima koji se nalaze u memoriji, jasno je zašto manjak operativne memorije bitno utiče na performanse personalnog računara. Savremeni računari raspolažu obično sa 256MB ili više operativne memorije. Ova količina memorije se preporučuje kao minimalna za rad sa *Windows XP* operativnim sistemom. Količina memorije u sistemu se može povećati jednostavnim dodavanjem odgovarajućeg tipa memorije u slobodne memorijske slotove na matičnoj ploči.

7.3.2 Spoljašnje memorije

Spoljašnje memorije predstavljaju uređaje i medijume za skladištenje podataka u računarima. Omogućavaju čuvanje (uključujući *backup*) i prenos velikih količina podataka i u vreme dok je *PC* isključen.

Najrasprostranjeniji uređaji za skladištenje podataka su:

- hard disk
- optički uređaji za skladištenje podataka, *CD* i *DVD-ROM*
- *USB flash* disk

Najbitnije karakteristike ovih uređaja su kapacitet, brzina upisa i čitanja podataka i trajnost čuvanja podataka koji se na njima nalaze.

Hard disk

Hard disk (*HDD - Hard Disc Drive*) je uređaj za skladištenje podataka u računaru, koji je, osim procesora, u poslednjih dvadesetak godina najviše napredovao. Značajna poboljšanja ostvarena su kako u tehnologiji izrade, tako i u pogledu kapaciteta, performansi, pouzdanosti i cene diska.

U početku, hard diskovi su bili glomazni i teški za proizvodnju. Prvi hard diskovi koji su ličili na današnje imali su glave za čitanje i upis koje su ostvarivale fizički kontakt sa površinom diska i na taj način omogućavale odgovarajućem elektronskom sklopu da bolje očita magnetno polje sa površine. Međutim, zbog fizičkog kontakta, glave su se brzo trošile i uz to grebale površinu diska, što je ugrožavalo sigurnost podataka na disku. Do nastanka modernih hard diskova dovelo je otkriće *IBM*-ovih inženjera koje se dogodilo 50-tih godina prošlog veka. Oni su došli do zaključka da bi, uz odgovarajući dizajn, glave mogle da lebde iznad površine diska i da pristupaju podacima na disku dok oni prolaze ispod njih. *IBM* je

napravio prvi komercijalno raspoloživ disk 1956. godine. Disk je imao kapacitet od 5MB i sastojao se od 50 ploča prečnika 24 inča. Gustina zapisa podataka je bila oko 2000 bita po kvadratnom inču, a brzina prenosa podataka je bila za to vreme vrlo velika, 8800B/s.

Konstrukcija hard diska

Hard disk se sastoji od više kružnih ravnih diskova (ploča) koji su sa obe strane presvučeni specijalnim materijalom koji ima mogućnost skladištenja informacija u magnetnoj formi. Svaki bit binarnog podatka upisuje se na površinu diska primenom specijalnih metoda kodiranja koje binarne vrednosti, 0 i 1, prevode u magnetni fluks. Ploče imaju otvor u centru i pričvršćene su na valjkasti nosač (*spindle*). Pokreću se pomoću specijalnog motora i rotiraju velikom brzinom.

Za upis i čitanje podataka sa diska koriste se specijalni elektromagnetni uređaji koji se nazivaju glavama (*heads*). Njihova uloga je da povežu magnetni medijum diska na kome se nalaze podaci sa elektronskim komponentama ostatka diska. Dakle, glave rade kao konvertori energije, jer transformišu magnetne signale u električne i obrnuto, u zavisnosti od toga da li se trenutno obavlja čitanje ili upis podataka. Zbog ovakve svoje funkcije, glave predstavljaju kritičnu komponentu u određivanju performansi diska i jedna su od najskupljih njegovih komponenata. Glave su montirane na nosač. Uređaj nazvan aktuator postavlja nosač zajedno sa glavama u određenu poziciju u odnosu na površinu diska.

Tehnologija koja se danas koristi za izradu glava za hard disk je tzv. *MR* tehnologija. *MR* glave koriste princip magnetorezistivnosti, tj. menjaju svoju otpornost kada se podvrgnu različitim magnetnim poljima. Upotrebom *MR* glava omogućena je mnogo veća gustina zapisa jer su one veoma osetljive, što znači da se bitovi podataka mogu postaviti bliže jedan drugom (povećava se gustina, a time i kapacitet hard diska).

Da bi mogao da obavlja svoju funkciju, hard disk mora biti izrađen sa velikom preciznošću. Njegova unutrašnjost izolovana je od spoljašnjeg sveta, kako bi se sprečilo da prašina dospe na površinu ploča, jer bi to moglo da dovede do trajnog oštećenja glava ili površine diska.

Sa donje strane hard diska nalazi se štampana ploča na koju je smeštena integrisana inteligentna kontrolerska logika. Njena uloga je da kontroliše rad svih komponenata diska, kao i da komunicira sa ostatkom računara.

Štampana ploča kontrolera na disku sadrži mikroprocesor, internu memoriju i ostale komponente koje kontrolišu rad diska. Ona predstavlja pravi računar u malom. Kako diskovi vremenom postaju napredniji i brži, sve više funkcija se dodaje kontrolerskoj logici, pa se u okviru nje koriste sve moćniji procesori i

prateći čipovi, kao i veća memorija da bi se dobili brži interfejsi i veći propusni opseg.

Osnovne funkcije mikroprocesora hard diska su:

- kontrola rada *spindle* motora
- kontrola rada aktuatora
- upravljanje vremenskim signalima za operacije čitanja i upisa
- keširanje podataka koji se čitaju sa ili upisuju na hard disk
- implementacija *power management* funkcije

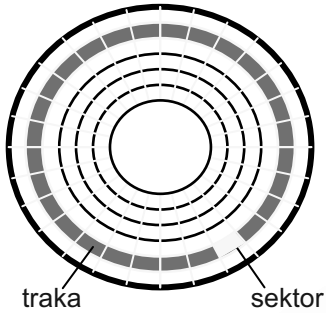
Memorija na štampanoj ploči hard diska je nepermanentnog tipa i koristi se kao keš memorija. Ona služi da uskladi razliku u brzini koja postoji između interfejsa prema matičnoj ploči i rada mehaničkih delova diska koji su relativno spori. Upotrebom keš memorije značajno se poboljšavaju performanse i smanjuje broj pristupa disku. Podaci sa diska se neprestano prebacuju u keš memoriju, bez obzira da li je magistrala na matičnoj ploči slobodna ili ne. Sa druge strane, računar može da šalje podatke na disk iako on nije spreman za upis novih podataka. Prispeli podaci se privremeno smeštaju u keš memoriju, a na disk će biti upisani kada on bude slobodan za upis.

Organizacija podataka na hard disku

Svaka ploča hard diska ima dve korisne površine (gornju i donju) na kojima se čuvaju podaci. Za svaku korisnu površinu postoji po jedna glava koja omogućava upis ili čitanje podataka sa nje. Tako, na primer, hard disk sa 3 ploče ima 6 glava.

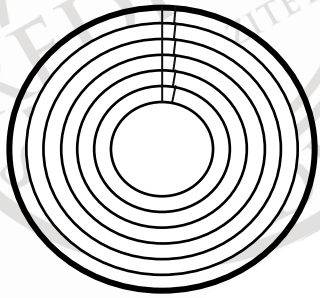
Iznad površina ploča, glave opisuju koncentrične kružnice koje se nazivaju trakama (*tracks*). Skupovi kružnica istih prečnika na svim korisnim površinama ploča nazivaju se cilindrima (*cylinders*). Radi lakšeg i bržeg pristupa podacima, svaka traka ugaono je podeljena na sektore (*sectors*) koji sadrže po 512 bajtova. Sektor predstavlja najmanji blok podataka kome se može pristupiti, tj. najmanji blok koji može da se adresira. Slika 7.8 prikazuje opisanu organizaciju podataka na hard disku.

Kao što se vidi na slici, broj sektora je isti po svim trakama, iako se one međusobno razlikuju po dužini. To ukazuje na činjenicu da prostor na disku nije optimalno iskorišćen.



Slika 7.8 Organizacija podataka na hard disku

ZBR – Zoned Bit Recording je tehnologija koja obezbeđuje ravnomerniju gustinu zapisa na disku i optimalno korišćenje cilindara bližih spoljašnjem obimu diska. Ova tehnologija izjednačava površine svih sektora na disku tako što odustaje od ugaone podele traka i uvodi podelu traka kao što je to prikazano na slici 7.9. Na ovaj način, dobija se da je broj sektora veći na spoljašnjim, nego na unutrašnjim trakama diska. Kao posledica toga, javlja se neravnomerna brzina transfera podataka sa različitih delova diska. Podaci se brže prenose sa spoljašnjih nego sa unutrašnjih cilindara.



Slika 7.9 ZBR tehnologija

Cylinder skew tehnologija je još jedno odstupanje od jednostavne organizacije po sektorima koje je uvedeno u cilju povećanja brzine čitanja i upisa podataka. Ova tehnologija rešava sledeći problem:

> Kada se pri sekvencijalnom čitanju sadržaja pročitaju svi sektori jednog cilindra diska, glave se pomeraju na sledeći cilindar. Pošto je glavi potrebno konačno vreme za pomeraj, a ploče ne prestaju da se okreću, glava bi se našla u sredini sektora koji treba da pročita na sledećem cilindru, ili čak iza njega. Da bi se pristupilo pravim podacima, ploče hard diska treba da obiđu ceo krug

kako bi se glave ponovo postavile iznad traženog sektora, čime se gubi mnogo vremena.

Rešenje opisanog problema ogleda se u tome što se disk realizuje tako da je prvi sektor narednog cilindra pomeren za nekoliko mesta u odnosu na poslednji sektor prethodnog cilindra.

Performanse hard diska

Performanse hard diska su jedan od faktora koji najviše utiču na ukupne performanse *PC* sistema. U pogledu protoka podataka, hard disk predstavlja jedno od uskih grla u sistemu, tako da se povećanje njegove brzine uvek primeti u svakodnevnom radu (brže učitavanje OS i korisničkih programa).

Brzina hard diska zavisi od primenjenog fajl sistema, kao i od više drugih parametara:

- vremena pristupa podacima na ploči diska
- interne i eksterne brzine prenosa podataka
- brzine rotacije ploča
- gustine zapisa podataka
- dimenzija ploča

Vreme pristupa podacima na ploči (*access time*) predstavlja zbir vremena traženja i vremena latencije.

Vreme traženja (*seek time*) predstavlja prosečno vreme koje je potrebno da bi se glave pomerile između dve trake na slučajnoj udaljenosti. Ono zavisi od mehaničkih karakteristika diska, kao i od udaljenosti između traka i izražava se u milisekundama. Osim prosečnog vremena traženja, koriste se još i vreme traženja između dve susedne trake (*track to track seek*) i vreme traženja između dve najudaljenije trake (*full stroke seek time*).

Latencija (*latency*) predstavlja vreme koje je potrebno ploči diska da se okrene da bi se glava, koja se već nalazi na odgovarajućoj traci, postavila iznad željenog sektora. To vreme najviše zavisi od brzine rotacije ploča. Takođe se koristi i prosečna latentnost (*average latency)* koja predstavlja vreme potrebno za rotaciju od 180˚.

Interna brzina prenosa podataka presudno utiče na ukupne performanse diska. Ona se izražava u MB/s i predstavlja brzinu kojom se podaci mogu čitati sa površine diska. Brzina prenosa se računa na osnovu fizičkih specifikacija, a to su brzina rotacije diska i gustina zapisa podataka. Ukoliko je primenjena *ZBR*

tehnologija, interna brzina prenosa podataka nije konstantna i zavisi od toga na kom delu diska se podaci nalaze. Brzina je znatno veća na obodu diska nego na njegovoj unutrašnjosti.

Eksterna brzina prenosa podataka predstavlja maksimalnu brzinu prenosa podataka između hard diska i matične ploče. Ova brzina najviše zavisi od brzine primenjenog interfejsa.

Brzina rotacije ploča u velikoj meri utiče na ukupne performanse diska. Njenim povećavanjem se istovremeno povećava brzina prenosa i smanjuje vreme pristupa podacima. Ova brzina se izražava u obrtajima u minuti (*RPM – Rounds Per Minute*). To je broj koji najviše govori o performansama diska, jer će skoro uvek disk koji se vrti većim brojem obrtaja biti brži od diska koji se vrti manjim brojem obrtaja. Trend povećanja brzine rotacije hard diska je veoma spor, ali će se sigurno nastaviti, jer se time najviše ubrzava njegov rad.

Gustina zapisa podataka po hard disk ploči povećava se drastično iz godine u godinu, tako da prevazilazi sva ranija optimistička predviđanja. U odnosu na prve *IBM* diskove, postignuta poboljšanja su reda veličine desetina miliona puta. Gustina zapisa direktno utiče na kapacitet hard diskova. Od početnih 10MB u 1981.godini, u januaru 2008.godine kapacitet komercijalno dostupnih hard diskova personalnih računara kretao se između 120GB i 1TB, dok je u julu 2010.godine ostvaren kapacitet od 3TB.

Dimenzije ploča hard diskova imaju tendenciju smanjivanja. Tako su hard diskovi dimenzije 5.25“ danas potpuno nestali sa tržišta, dok diskovi dimenzije 3.5" dominiraju u desktop računarima i računarima za serverske primene. Kod prenosivih računara, diskovi od 2.5" su standard, ali se koriste i diskovi manjih dimenzija. Smanjenje dimenzija donosi sa sobom povećanje čvrstine ploča diskova, kao i smanjenje njihove mase što omogućava veće brzine rotacije kao i veću pouzdanost. Što se tiče broja ploča, hard diskovi danas najčešće koriste između jedne i četiri ploče.

Težnja za povećanjem brzine diska ima smisla samo ako su podaci na disku sigurni. Stoga je važna karakteristika hard diska i njegova pouzdanost.

Pouzdanost se izražava pomoću:

- *MTBF* vrednosti (*mean time between failures*), koja predstavlja srednje vreme između dva otkaza
- broja uključenja/isključenja (*start/stop cycles*) koje disk može da izdrži

Za *MTBF* i broj uključenja/isključenja mogu se odrediti samo teorijske vrednosti (milioni sati za *MTBF*, a stotine hiljada uključenja/isključenja), ali ne i statističke, jer se u praksi ne mogu sprovesti merenja vremena do otkaza diskova u trajanju od nekoliko godina.

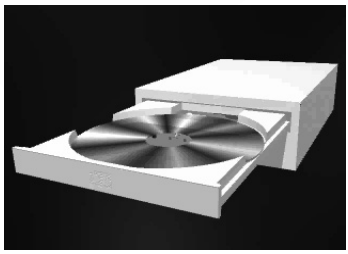
CD-ROM

CD-ROM (*Compact Disc - Read Only Memory*) je optički medijum za skladištenje podataka koji je u protekloj deceniji prešao put od skupe do jeftine komponente prisutne u svakom personalnom računaru. S obzirom na relativno veliki kapacitet koji pruža uz veoma nisku cenu, *CD-ROM* se pokazao vrlo pogodnim ne samo u standardnim, već i u raznim multimedijalnim primenama. Zahvaljujući njegovoj pojavi, oblast multimedija je doživela pravu ekspanziju što je dovelo do naglog povećanja broja multimedijlnih aplikacija koje do tada nisu bile prisutne u većoj meri.

CD-ROM je nastao 1978.godine kao rezultat udruženog rada kompanija *Philips* i *Sony*. Godine 1984. *CD* tehnologija je standardizovana specifikacijom formata zapisa i kodova za detekciju i korekciju grešaka (*ECC - Error Correction and Control*).

Konstrukcija *CD-ROM* uređaja

Po konstrukciji, *CD* uređaji su veoma slični drugim uređajima za skladištenje podataka koji koriste rotirajuće ploče (na primer, hard diskovima). Međutim, između njih postoji značajna razlika u postupku upisa i čitanja podataka. Za razliku od pomenutih uređaja koji koriste magnetni medijum, *CD* uređaji koriste optički metod zapisa i čitanja podataka. Princip rada *CD-ROM* uređaja zasniva se na pretvaranju optički uskladištenih podataka u električne signale. Spoljašnji izgled *CD-ROM* uređaja prikazan je na slici 7.10.



Slika 7.10 *CD-ROM* uređaj

Postupak čitanja podatka sa *CD-ROM* diska odvija se tako što se najpre na površinu diska usmeri laserski zrak, a zatim se detektuje intenzitet reflektovane

svetlosti. Na disku postoje jame (*pit*) i površi (*land*) koje predstavljaju binarne vrednosti 0 i 1. Intenzitet svetlosti reflektovane iz jame je mnogo slabiji od intenziteta svetlosti reflektovane od površi. Reflektovana svetlost sa površi i jama, preko složenog sistema sočiva i ogledala, prenosi se do foto dioda koje detektuju razlike u intenzitetu svetlosti i te razlike pretvaraju u električne signale. Ove impulse zatim dekoduje kontrolerska logika *CD-ROM* uređaja i u obliku digitalnih podataka (1 i 0) šalje na matičnu ploču računara.

Organizacija podataka na CD-ROM disku

Za smeštaj podataka na *CD* predviđena je spiralna staza koja počinje od centra diska a završava se na 5mm od njegovog oboda. Rastojanje između dve susedne trake na stazi je 1.6μm. Duž spiralne staze nalaze se površi i jame dužine oko 1μm.

Podaci se na *CD* upisuju počevši od centra diska ka periferiji. Gustina zapisa je konstantna po jedinici površine, tj. ne zavisi od toga da li se podaci nalaze bliže obodu ili centru diska. S obzirom da se disk u *CD* uređaju okreće konstantnom ugaonom brzinom, podaci se brže čitaju sa spoljnih nego sa unutrašnjih staza.

Osim podataka, *CD* sadrži i dodatne informacije koje služe za sinhronizaciju prenosa i korekciju grešaka. Ove informacije doprinose većoj preciznosti i pouzdanosti u radu. Zauzimaju oko 13% kapaciteta diska i nevidljive su za korisnika *CD* uređaja. Preostalih 87% diska predstavlja deklarisani kapacitet diska koji služi za skladištenje podataka. Kapacitet *CD*-a može biti 650MB ili 700MB.

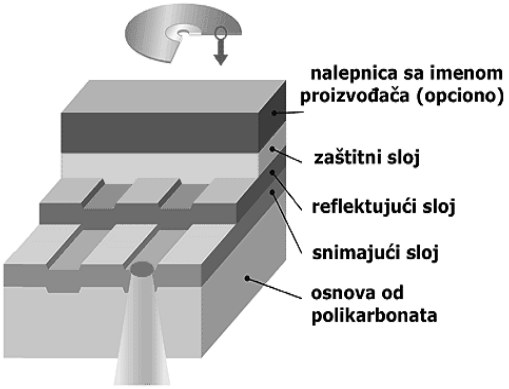
Brzina rada *CD* uređaja deklariše se u umnošcima brzine čitanja muzičkog *CD*-a, koja iznosi 150KB/s. Tako, jednobrzinski *CD* uređaj (oznaka 1x) čita podatke brzinom od 150KB/s, dok pedeset-dvo-brzinski *CD* uređaji (52x) čitaju podatke brzinom od 7800KB/s. Treba napomenuti da se deklarisana brzina odnosi na brzinu čitanja podataka sa krajnje spoljne trake diska, dok je brzina čitanja sa krajnje unutarnje trake više nego duplo manja.

CD-R

Glavni nedostatak *CD-ROM* uređaja je nemogućnost upisa podataka na disk, što je posledica činjenice da se kod ove tehnologije nule i jedinice fizički utiskuju u plastični supstrat. Ovaj problem je rešen novom, *CD-R* (*Compact Disc - Recordable*) tehnologijom koja je specificirana 1990.godine. Prvi *CD-R* uređaji na tržištu su se pojavili 1993.godine, a proizvođač je bila kompanija Philips. *CD-R* uređaji podržavaju sve *CD* formate, a osim snimanja, rade i kao *CD-ROM* čitači.

CD-R diskovi imaju supstrat na koji je naneta “prazna" spirala (*spiral pre-groove*). Ova spirala služi *CD-R* uređaju da je prati prilikom upisa. Na supstrat se nanosi specijalan fotosenzitivni (snimajući) sloj, na njega veoma tanak reflektujući

sloj od srebra ili zlata i na kraju dolazi zaštitni sloj, koji predstavlja gornju površinu diska (videti sliku 7.11).



Slika 7.11 Presek CD-R diska

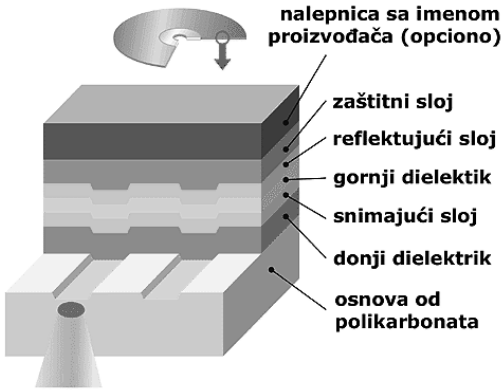
Boja *CD-R* diskova zavisi od boje i tipa fotosenzitivnog sloja i reflektujućeg sloja, tako da kombinacije ova dva sloja imaju zelenu, zlatnu, plavu ili srebrnu boju.

Fotosenzitivni sloj ima osobinu da se, kada se osvetli laserskom svetlošću određenog tipa i intenziteta, brzo greje i menja hemijski sastav. Kao rezultat ovog "prženja", tj. promene hemijskog sastava površine koja je "spržena" (*burned*), ova površina reflektuje manje svetla nego ona koja nije "spržena". Na ovaj način, ceo snimljeni disk je izdeljen na delove koji su "sprženi" (0) ili nisu "sprženi" (1). Ovako snimljeni *CD-R* diskovi mogu da se čitaju na svakom *CD-ROM* čitaču, kao da se radi o fabrički narezanom *CD-ROM* disku.

Pošto *CD-R* medijum na "sprženim" delovima trajno menja hemijsku strukturu i fizička svojstva, jednom snimljen disk se ne može presnimiti ili obrisati. Tehnikom *multi-session* dozvoljava se da se na disk koji nije iskorišćen do kraja dosnimi još podataka, pri čemu se gubi 13MB za svaku novu sesiju.

CD-RW

CD-RW (*Compact Disc - Rewritable*) radi na sličnom principu kao *CD-R* disk, s tim što umesto fotosenzitivnog sloja ima tri nova sloja: donji dielektrik, fazno promenljivi (snimajući) sloj i gornji dielektrik, kao što je prikazano na slici 7.12.



Slika 7.12 Presek *CD-RW* diska

Dielektrični slojevi služe da odvlače toplotu sa snimajućeg sloja. Kad je disk prazan, snimajući sloj je kristalizovan i u tom stanju reflektuje svu svetlost. Kad laser za snimanje zagreje tačke na snimajućem sloju iznad temperature topljenja (500-700˚C), smeša na tom mestu prelazi u tečno stanje, a ako se mesto odmah ohladi, prelazi u amorfno stanje u kojem skoro potpuno apsorbuje svetlost. Brisanje se vrši kad se amorfni sloj zagreje na temperaturu kristalizacije i tako drži određeno vreme, a zatim ohladi, čime se vraća u kristalizovano stanje. Podaci se na *CD-RW* diskove generalno snimaju sporije nego na klasične *CD-R* diskove.

DVD-ROM

DVD-ROM (*Digital Video Disc - Read Only Memory*) je tehnologija novijeg datuma koja omogućava skladištenje znatno većih količina podataka nego *CD* tehnologija, uz znatno veće brzine transfera podataka. Prvi *DVD-ROM* uređaji postali su komercijalno dostupni 1997.godine. Kako se tehnologija razvijala, slično kao i kod *CD-ROM* uređaja, pojavili su se *DVD-ROM* uređaji sa više brzina.

Spoljašnji izgled diskova, kao i spoljašnjost i unutrašnjost *DVD-ROM* i *CD-ROM* uređaja na prvi pogled se ne razlikuju mnogo. Obe tehnologije podatke smeštaju na diskove prečnika 120mm i debljine 1.2mm. Međutim, međusobno rastojanje između traka na *DVD* disku je znatno manje od rastojanja između traka *CD* diska, tako da je ukupna dužina razvijene spirale kod *DVD* diska 11km, što je duplo duže nego kod *CD* diska. Takođe, dužina površi ili jame kod *DVD* diska je duplo manja nego kod *CD* diska i iznosi oko 0.5μm, čime je bitno povećan kapacitet diska. Da bi ispravno pročitao podatke koji odgovaraju površima i jamama ovih dimenzija, *DVD* uređaj mora da obezbedi bolji fokus laserske svetlosti, što se može postići primenom crvenih lasera (*CD* uređaji koriste

infracrvene lasere). Još bolji rezultati u budućnosti očekuju se od upotrebe plavih lasera.

DVD-ROM uređaji okreću diskove manjom brzinom nego *CD-ROM* uređaji, ali s obzirom na veću gustinu zapisa podataka, ukupni protok podataka je znatno veći nego kod *CD-ROM* uređaja ekvivalentne brzine rotacije. Na primer, dok 1x *CD-ROM* ima maksimalni protok od 150KB/s, 1x *DVD-ROM* ima protok od oko 1.3MB/s, što je skoro 9 puta veća brzina.

DVD tehnologija podržava čitanje podataka sa dvoslojnih *DVD* diskova promenom fokusa lasera za čitanje. Za prvi informacioni sloj se umesto reflektujućeg sloja koristiti providni sloj, a za drugi (unutrašnji) informacioni sloj se koristi normalan reflektujući sloj. Ova tehnika ne povećava kapacitet diska za 100%, ali dozvoljava da na takav disk stane 8.5GB podataka.

Osim što mogu da imaju dva sloja, *DVD* diskovi mogu biti i dvostrani. Da bi se omogućilo preciznije fokusiranje lasera na manje jame i smanjila osetljivost na neravnine, proizvođači su odlučili da smanje debljinu zaštitnog plastičnog sloja kroz koji laser mora da prođe da bi dospeo do informacionih površina. Korišćenje tanjeg plastičnog supstrata nego što je to slučaj kod *CD* diska, rezultiralo je diskovima debljine od samo 0.6mm. Ovakvi diskovi bili su isuviše tanki, pa su se lomili i krivili jer nisu imali potrebnu čvrstinu. Zbog smanjene izdržljivosti diskova, da bi oni ipak ostali ostali ravni, odlučeno je da se dva ovakva diska zalepe "leđa o leđa", pa da se na taj način opet dobije disk debljine 1.2mm. Na ovaj način kapacitet diska je povećan za 100% u odnosu na jednostrani disk.

U skladu sa navedenim, *DVD* tehnologija podržava četiri standarda za kapacitet diska:

- *DVD*-5 za jednostrani-jednoslojni disk kapaciteta 4.7GB
- *DVD*-9 za jednostrani-dvoslojni disk kapaciteta 8.5GB
- *DVD*-10 za dvostrani-jednoslojni disk kapaciteta 9.4GB
- *DVD*-18 za dvostrani-dvoslojni disk kapaciteta 17GB

DVD-ROM uređaji imaju mogućnost čitanja kako *DVD*, tako i *CD* diskova. Znatno su tolerantniji prema problematičnim *CD* diskovima, koje mnogi *CD-ROM* čitači teško ili uopšte ne mogu da pročitaju.

DVD disk ima veću izdržljivost i pouzdanost u odnosu na *CD* disk. Naime, *DVD* diskovi imaju ugrađen mnogo bolji i efikasniji algoritam za korekciju grešaka (*ECC*), koji može da ispravi grešku koja se javila na 2000 uzastopnih bajtova podataka (dužina od oko 4mm trake).

Zbog ogromne rasprostranjenosti *CD* formata, jedan od glavnih zadataka projektanata bio je da naprave uređaj koji bi omogućio upotrebu i *CD* i *DVD*

diskova. To je zahtevalo projektovanje specijalnog optičkog sklopa koji može da podesi fokus kako za tanke (0.6mm) nosioce *DVD* formata, tako i za stare 1.2mm nosioce *CD* formata. Rešenje je postignuto upotrebom specijalnog sočiva u čiji centar je utisnut hologramski element. Laserska svetlost koja prolazi po obodu sočiva, van hologramа, fokusirana je tako da na površini diska stvara dovoljno malu tačku pogodnu za čitanje *DVD* formata. Jedna trećina zraka prolazi kroz hologram u centru i reaguje sa njim tako da je takav zrak fokusiran i sočivom i hologramom i na površini diska stvara tačku pogodnu za čitanje *CD* formata.

USB fleš disk

USB Memory Drive ili *Keydrive* je mali prenosni uređaj za skladištenje podataka koji koristi fleš memoriju (*flash memory*) i *USB* konektor na računaru. Njegov izgled je prikazan na slici 7.13. Za razliku od ostalih prenosivih medijuma za skladištenje podataka, *USB* fleš koristi poluprovodničku tehnologiju (čipove) za čuvanje podataka. Ovo ga čini otpornim na fizička oštećenja i prašinu.

Kapacitet ovih diskova vremenom se menjao od početnih 16MB do današnjih 256MB. U 2003.godini većina *USB* fleševa radila je na USB 1.0/1.1 standardu brzinom od 12Mb/s. U 2004.godini novi *USB* fleševi podržavaju USB 2.0 interfejs, sa maksimalnom brzinom čitanja od oko 200Mb/s i brzinom upisa od oko 100MB/s. U idealnim uslovima, ovako sačuvani podaci mogu da opstanu oko 10 godina.

Slika 7.13 *USB Memory Drive*

USB fleš disk se priključuje na normalan tip-A *USB* priključak, bilo na računaru, bilo na *USB hub*-u. Uređaj dobija napajanje preko *USB* priključka na računaru, tako da mu nije potrebno spoljašnje napajanje.

Nove tehnologije

Zbog sve veće potrebe za uređajima i medijumima za skladištenje većih količina podataka, danas se u svetu paralelno razvija nekoliko novih tehnologija za optičko skladištenje podataka. Među najperspektivnijim od njih mogu se izdvojiti:

- *BluRay Disc* (*BD*)
- Holografski disk (*HVD*)

BluRay Disc je optički disk za skladištenje podataka koji je nastao 2003.godine. To je prvi video format visoke definicije koji nije razvio *DVD* forum (telo koje podržava već uspešan i priznat *DVD* format). *BluRay* format je razvio konzorcijum od devet priznatih proizvođača nazvan *Blu-ray Disc Founders* koga čine: *Hitachi*, *LG Electronics*, *Matsushita Electric Industrial*, *Pioneer*, *Royal Philips Electronics*, *Samsung Electronics*, *Sharp*, *Sony* i *Thompson*. Na slici 7.14 prikazan je uređaj koji koristi *Blu-ray* diskove.

Slika 7.14 *Blu-ray Disc* plejer

Ideja je bila da se za čitanje i upis podataka na disk koristi novi, plavi laser (odatle potiče i naziv formata) koji radi sa talasnim dužinama od 405nm. Za konvencionalne *DVD* i *CD* formate koriste se crveni, odnosno infracrveni laseri koji rade sa talasnim dužinama od 650nm i 780nm, respektivno. Korišćenje kraćih talasnih dužina značajno smanjuje prostor potreban za predstavljanje jednog bita na površini diska. Stoga se na *Blu-ray* disk istih dimenzija (prečnik ploče 12cm) može smestiti šest puta više podataka nego na *DVD*, a dvanaest puta više nego na *CD*. Maksimalan kapacitet jednoslojnog *Blu-ray* diska je 25GB, dok je za dvoslojni 50GB. Na *Blu-ray* disk može da stane dva sata TV programa visoke definicije.

Glavni konkurent *Blu-ray* formatu bio je *HD DVD* format (*High Definition DVD format*), poznat i kao *AOD* (*Advanced Optical Disc*). *AOD* format su razvili *Toshiba* i *NEC*. Specifikacija tehnologije je završena 2004.godine, a prvi uređaji su se pojavili 2006.godine. *HD DVD* format je takođe koristio plave lasere sa talasnim dužinama od 405nm, ali je imao nešto manji kapacitet (20 GB). Prednost mu je bila u nižoj ceni. Iako je *HD DVD* neko vreme smatran pravim naslednikom *DVD* formata, u februaru 2008.godine, *Toshiba* je odustala od daljeg razvoja, proizvodnje i plasmana *HD DVD* uređaja i time je *Blu-ray* dobio primat na tržištu.

Holografski disk (*HVD - Holographic Versatile Disc*) je optički disk koji je proizvod japanske korporacije *Optware*. Korporacija je formirana 1999.godine od šest giganata elektronske industrije sa ciljem pronalaženja načina da se tehnologija holografskog beleženja podataka pretoči u komercijalne proizvode.

Tehnologija se zasniva na tzv. kolineranoj holografiji koja podrazumeva korišćenje dva lasera, crvenog i zelenog. *HVD* je istih dimenzija kao i standardni *DVD* i *CD* diskovi (12 cm u prečniku), ali su mu karakteristike znatno bolje. Njegov kapacitet je do 3.9 TB informacija, što je oko 5800 puta više od *CD*, 850 puta više od kapaciteta *DVD*, 160 puta više od jednoslojnog *Blu-ray* diska, a dva puta više od najvećih hard diskova u 2008.godini. Izgled holografskog diska dat je na slici 7.15.

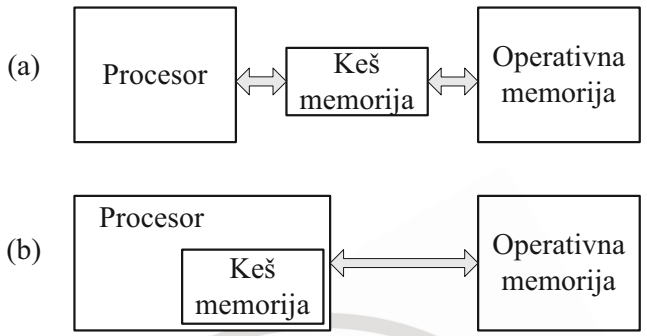
Slika 7.15 Holografski disk

7.3.3 Keš memorija

Keš memorija predstavlja veoma brzu memoriju relativno malog kapaciteta koja se nalazi u okviru procesora ili u njegovoj neposrednoj blizini. Njena osnovna funkcija je poboljšanje performansi računarskog sistema. Zbog brzine i lakoće rukovanja, keš memorija je statičkog tipa, pa samim tim ima visoku cenu. Cena keš memorije, kao i veličina silicijumske pločice koju ona zauzima glavni su razlozi za njenu upotrebu u relativno malim količinama.

Princip korišćenja keš memorije je sledeći: kada procesor zahteva neki podatak iz operativne memorije, tada se iz operativne u keš memoriju, osim traženog podatka, prenosi i određena količina podataka koji se nalaze iza traženog podatka u operativnoj memoriji. Ubrzanje rada ostvaruje se zahvaljujući tome što je velika verovatnoća da će naredni potrebni podaci biti među podacima koji su već preneti u keš memoriju. Na primer, programske instrukcije se u operativnoj memoriji nalaze na sukcesivnim adresama. Ako se u keš memoriju prenese blok instrukcija, velika je verovatnoća da će naredna instrukcija koju procesor treba da izvrši već biti u keš memoriji, s obzirom da se instrukcije izvršavaju sukcesivno. Kako je keš memorija znatno brža od operativne, ovime je obezbeđen znatno brži pristup podacima, a samim tim i brži rad celog sistema.

Keš memorija se može nalaziti unutar procesora (slika 7.16 (b)), ili izvan njega (slika 7.16 (a)).



Slika 7.16 Veza keš memorije sa okruženjem

Postoje dve vrste keš memorije: *L1* i *L2* keš.

***L1* (*level* 1) keš** memorija se nalazi u okviru samog procesora i radi na istom taktu kao i procesor. Može se javiti u dva oblika:

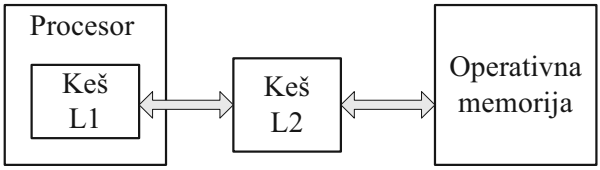
- kao jedinstveni blok memorije u kome se čuvaju i podaci i instrukcije
- kao podeljeni keš koji se sastoji od keša za podatke i keša za instrukcije

Jedinstveni keš daje bolje efekte prilikom rada sa multimedijalnim podacima kada velike količine podataka prolaze kroz procesor. Podeljeni keš daje bolje efekte kada, osim podataka, u procesor dolaze i brojne instrukcije.

Količina *L1* keš memorije ograničena je tehnološkim mogućnostima proizvodnje procesora. Zbog velikog broja tranzistora koji realizuju jednu memorijsku ćeliju, *L1* keš zahteva veliku površinu na silicijumskoj pločici na kojoj se pravi procesor. Takođe, *L1* keš bitno doprinosi zagrevanju procesora.

***L2* (*level* 2) keš** memorija se nalazi van procesora, ali veoma blizu njega. Ona radi na taktu koji je jednak polovini takta procesora, ili na taktu čipseta. Iako je sporiji od *L1*, *L2* keš je još uvek znatno brži od operativne memorije, tako da u velikoj meri doprinosi brzini rada sistema. Kapacitet *L2* keš memorije je znatno veći od kapaciteta *L1* keš memorije. Kontroler *L2* keš memorije integrisan je u okviru severnog mosta čipseta.

Sa stanovišta računarskog sistema poželjno je imati što više keš memorije, naročito tipa *L1*. U savremenim računarskim sistemima obično su zastupljena oba tipa keš memorije. Struktura ovakvih sistema prikazana je na slici 7.17.



Slika 7.17 Sistem sa *L1* i *L2* keš memorijom

7.4. Ulazno/izlazni uređaji

Ulazno/izlazni uređaji (*I/O devices*) personalnog računara omogućavaju razmenu podataka sa okruženjem. Ovde se pod okruženjem podrazumeva ne samo korisnik računara (čovek), već i drugi računari i digitalni sistemi sa kojima se može ostvariti komunikacija. Ulazni uređaji omogućavaju prijem, a izlazni slanje podataka. Ulazni uređaji su tastatura, miš i sl., dok su najznačajniji izlazni uređaji monitori i štampači, o kojima će biti reči u nastavku.

7.4.1 Monitori

Monitor predstavlja važan deo personalnog računara koji omogućava vizualizaciju programa koji se trenutno izvršava. Postoji više tehnologija koje se koriste za izradu ekrana monitora, a najzastupljenije su:

- *CRT* (*Cathode Ray Tube*) – ekran sa katodnom cevi
- *LCD* (*Liquid Crystal Display*) – ekran sa tečnim kristalom

***CRT* tehnologija** je starijeg datuma i zasniva se na korišćenju katodnih cevi sličnih onima koje se koriste kod televizijskih prijemnika. Na jednom kraju katodne cevi nalazi se elektronski top, a na drugom površina ekrana na kojoj se formira slika. Elektronski top šalje snop elektrona koji veoma brzo prelazi preko ekrana sa leva na desno u linijama od vrha do dna ekrana. Trag koji snop elektrona ostavlja na fosforom premazanoj površini ekrana formira sliku. Iako je u jednom trenutku mlazom pogođena samo jedna tačka na ekranu, korisnik ima utisak postojanja kontinualne slike i to iz dva razloga:

- fosfor, kojim je presvučena unutrašnja strana ekrana, nastavlja da svetli neko vreme nakon što je pogođen elektronskim mlazom
- zbog tromosti ljudskog oka (nemogućnosti da primeti veoma brze promene) korisniku se čini da su tačke još uvek osvetljene, iako to nije slučaj

S obzirom da slika koju formira elektronski top *CRT* monitora vremenom bledi, elektronski mlaz mora stalno da prelazi preko ekrana kako bi održao sliku. Ovaj postupak naziva se osvežavanjem ekrana, a učestanost kojom se osvežavanje obavlja naziva se učestanošću osvežavanja ili vertikalnom frekvencijom. Učestanost osvežavanja predstavlja broj formiranih slika u sekundi i izražava se u hercima (Hz). Za većinu modernih monitora ovaj parametar se kreće u rasponu od 60Hz do 150Hz.

***LCD* tehnologija** je novijeg datuma u odnosu na *CRT* tehnologiju. Prvobitno je bila namenjena samo za primenu kod prenosivih računara i uređaja.

Postupak formiranja slike na površini *LCD* ekrana zasniva se na korišćenju tečnog kristala. Iza *LCD* ekrana nalazi se izvor svetlosti. Svetlost najpre prolazi kroz polarizujući filtar, posle koga su svi zraci koji prođu usmereni u istom pravcu koji je normalan na površinu ekrana. Iza ovog, nalazi se drugi polarizujući filtar koji se sastoji od ćelija sa tečnim kristalom. Te ćelije mogu da menjaju ugao polarizacije svetlosti i na taj način utiču na količinu svetlosti koja kroz njih prolazi. Svaki piksel (tačka) na *LCD* ekranu sastoji se od tri ćelije, po jedna za prikazivanje crvene, zelene i plave boje.

LCD ekrani vremenom dobijaju „mrtve“ piksele (tačke). To su tačke ne ekranu koje su stalno tamne (crni „mrtvi“ piksel) ili stalno emituju maksimalnu količinu svetlosti (svetli „mrtvi“ piksel, znatno više smeta korisniku od crnog). One smanjuju kvalitet slike i otežavaju rad na računaru. Obično su posledica procesa proizvodnje monitora.

Razlike između monitora sa *CRT* i *LCD* ekranom su:

- *LCD* monitori imaju savršenu geometriju slike, dok se kod *CRT* monitora ona podešava relativno komplikovanim skupom operacija (od kojih neke nisu podržane komandama sa prednje strane monitora) i nikad nije savršena
- *LCD* monitori često imaju mogućnost fizičke rotacije ekrana za 90º, čime se dobija odnos dimenzija slike 3:4, koji je povoljniji za čitanje teksta i rad u profesionalnom okruženju
- *LCD* monitori imaju znatno manju dubinu i nekoliko puta su lakši od *CRT* monitora, pa zauzimaju manje mesta na radnom stolu i lakši su za manipulaciju

Osnovne karakteristike monitora su:

- veličina i dimenzija ekrana
- zakrivljenost ekrana
- rezolucija ekrana
- frekvencija osvežavanja ekrana
- maksimalni osvetljaj i kontrast slike
- ugao pod kojim se slika može videti
- način povezivanja sa okruženjem
- potrošnja električne energije
- zračenje

Veličina ekrana monitora predstavlja dužinu dijagonale ekrana i izražava se u inčima (1" = 2.54cm). Današnji ekrani monitora personalnih računara imaju veoma širok opseg veličina, od 12" za male *notebook* računare, preko standardnih 17" za *desktop* računare, pa sve do 25" za monitore računara koji se koriste za grafičke primene.

Kod *CRT* monitora vidljiva dijagonala je za 1" do 1.5" manja od one koja je deklarisana. Tako, na primer, vidljiva dijagonala 17" monitora je obično od 15.5" do 16". Ovo je posledica toga što proizvođači *CRT* monitora deklarišu ukupnu dijagonalu ekrana u koju su uračunati i delovi ekrana koji nisu u funkciji prikaza slike. Kod *LCD* monitora ovo pravilo ne važi, tako da vidljiva dijagonala odgovara deklarisanoj.

Osim veličine dijagonale, bitan je i odnos širine i visine slike. Postoje dva standarda za definisanje ovog odnosa:

- širina:visina = 4:3; iako je stariji, ovaj standard se još uvek češće koristi
- širina:visina = 16:9 (drugi naziv za ovaj standard je *Wide Screen*); ovo je noviji standard koji više odgovara dimenzijama vidnog polja čoveka, tako da je slika na monitorima koji su izrađeni po ovom standardu preglednija, što monitor čini pogodnijim za rad

Zakrivljenost ekrana je osobina koja se javlja samo kod *CRT* monitora. To je nepoželjna osobina zato što samo ekrani sa ravnom površinom garantuju mali odsjaj i dobru vidljivost iz različitih uglova gledanja. Ekrani *CRT* monitora najčešće imaju zakrivljeni oblik koji predstavlja deo sfere. Ovakva konstrukcija je posledica toga što je za projektovanje slike na površini ekrana potrebno da sve tačke na ekranu budu jednako udaljene od izvora elektronskog mlaza. Pred moderne *CRT* ekrane postavlja se zahtev da imaju što ravniji ekran. Da bi se ovo

postiglo koristi se moderna elektronska tehnologija koja kompenzuje različite udaljenosti pojedinih delova ekrana od izvora elektronskog mlaza. Ipak, i pored toga, veoma malo ekrana je potpuno ravno. Proizvođači ih veoma često konstruišu tako da im je spoljna površina potpuno ravna dok sa unutrašnjom stranom, na kojoj se projektuje slika, to nije slučaj. Ovime korisnik stiče utisak da sedi za potpuno ravnim ekranom iako to nije tako.

Ekrani *LCD* monitora, zbog drugačijeg načina projektovanja slike, mogu da budu potpuno ravni, što predstavlja bitnu prednost u odnosu na *CRT* tehnologiju.

Rezolucija ekrana predstavlja broj piksela po horizontali i vertikali ekrana (piksel je najmanji deo površine ekrana koji može da generiše proizvoljnu boju). Odnos broja piksela po vertikali i po horizontali je isti kao i odnos dimenzija ekrana, tj. za većinu monitora iznosi 4:3. Standardne vrednosti rezolucija ekrana su od 640x480 do 1600x1200. Korisnik može da podešava rezoluciju prema veličini ekrana i sopstvenim potrebama. Veća rezolucija garantuje sliku sa većim brojem detalja i većom radnom površinom, ali objekti na slici postaju manji, što prilikom dužeg rada može da dovede do nepotrebnog naprezanja vida.

Frekvencija osvežavanja ekrana predstavlja broj slika koje se prikazuju na ekranu tokom jedne sekunde i izražava se u Hz. Ako je frekvencija osvežavanja previše niska, slika na ekranu manje ili više treperi, što prilikom dužeg gledanja može da dovede do zamora očiju. Stoga minimalna preporučena frekvencija osvežavanja za dugotrajan rad na računaru iznosi 75Hz. Na maksimalnu frekvenciju osvežavanja utiče rezolucija slike na ekranu. Povećanjem rezolucije, maksimalna frekvencija osvežavanja se smanjuje i obrnuto.

Maksimalni osvetljaj i kontrast slike utiču na subjektivni kvalitet slike i mogućnost rada u prostorijama sa puno svetla.

Osvetljaj ekrana meri se u kandelama po kvadratnom metru ($1cd/m^2$), što se u SAD izražava u fizičkim jedinicama koje se zovu "nit". Prosečni monitori imaju od 150 do 250 nita, pri čemu su ove vrednosti nešto veće kod *LCD* nego kod *CRT* ekrana.

Kontrast ekrana predstavlja odnos jačine osvetljaja između najsvetlije i najtamnije tačke koja se može reprodukovati na ekranu. Veći kontrast doprinosi oštrijem tekstu i življim bojama. *CRT* monitori imaju nešto bolji kontrast od *LCD* monitora. Kod modernih monitora kontrast iznosi od 250:1 do 750:1.

Uglovi vidljivosti predstavljaju maksimalne uglove po horizontali i vertikali ekrana pod kojima se slika može videti bez promena u boji i kontrastu i bez većih deformacija u izgledu slike.

Uglovi vidljivosti kod *CRT* ekrana u mnogome zavise od njegove zakrivljenosti. Ona doprinosi da slika gledana sa strane deluje izduženo, a delovi slike na suprotnoj ivici se čak ne mogu ni videti. Takođe, postoji i mala promena u bojama, ali taj problem je manje izražen.

Pri promeni ugla vidljivosti, *LCD* ekrani imaju manji problem sa deformacijom slike jer su potpuno ravni. Međutim, veliki problem kod ovog tipa monitora je promena boja i smanjenje kontrasta pri gledanju sa strane. *LCD* ekrani se konstruišu tako da omoguće što veću promenu ugla gledanja po horizontali na račun manje vidljivosti sa promenom ugla gledanja po vertikali. Moderni *LCD* ekrani postižu sve veće uglove vidljivosti tako da su se po toj karakteristici dosta približili *CRT* ekranima. Dobar *LCD* monitor ima ugao vidljivosti veći od 140º po horizontali i od 120º po vertikali.

Postoje dva ***načina povezivanja monitora i personalnog računara***: putem analogne i digitalne veze (interfejsa).

Analogno povezivanje je starijeg datuma i u upotrebi je kod gotovo svih *CRT* monitora. Signali u analognom obliku za crvenu, zelenu i plavu boju generišu se u grafičkoj kartici računara i dovode do monitora. Pošto je *CRT* monitor po svojoj prirodi analogan, dobijeni signali su za njega sasvim odgovarajući. Analognim povezivanjem postiže se brži rad i praktično beskonačan spektar boja koji se može poslati na monitor.

Digitalno povezivanje ili *DVI* (*Digital Visual Interface*) je u upotrebi uglavnom kod *LCD* monitora, koji su po svojoj prirodi digitalni. Ako se kod *LCD* monitora koristi analogno povezivanje, tada se najpre u grafičkoj kartici digitalni signal pretvara u analogni, a potom se u monitoru analogni signal vraća u digitalni oblik. Ova dvostruka konverzija doprinosi smanjenju kvaliteta slike, što se na ekranu obično manifestuje kao treperenje ili „plivanje“ piksela.

Potrošnja električne energije od strane monitora je vrlo bitna karakteristika, jer monitor troši skoro polovinu električne energije celog *PC* sistema. Potrošnja monitora najviše zavisi od tipa i veličine ekrana. Monitori sa *LCD* ekranom troše znatno manje energije od *CRT* monitora (u proseku oko 2.5 puta manje za istu veličinu ekrana).

Kod novijih monitora, računar može da upravlja radom monitora. Ukoliko se tokom određenog vremena monitor ne koristi, računar može da ga delimično ili potpuno isključi i tako doprinese štednji električne energije i dužem životnom veku samog monitora.

Zračenje monitora je vrlo bitno jer može loše da utiče na zdravlje korisnika. Problem predstavljaju zračenja veoma niske učestanosti. Opasnost od ozbiljnijeg

oboljenja je relativno mala, ali ako korisnik provodi trećinu dana za računarom, onda svakako spada u rizičnu grupu. Radi zaštite od zračenja, propisano je nekoliko standarda koje treba poštovati (*MPR* I, *MPR* II, *TCO* i dr.). Takođe, treba imati u vidu da monitori znatno više zrače sa zadnje nego sa prednje strane, pa u skladu sa tim treba obratiti pažnju na raspored monitora u prostoriji. Intenzitet zračenja znatno opada sa povećanjem rastojanja od izvora zračenaja. To znači da je na dovoljnoj udaljenosti od monitora zračenje veoma malo. Minimalno rastojanje na kom korisnik treba da se nalazi, prema preporukama, jednako je dvostrukoj dužini dijagonale ekrana monitora.

Količina zračenja kod *LCD* monitora je samom tehnologijom izrade svedena na minimum, dok je kod *CRT* monitora problem zračenja izraženiji.

7.4.2 Štampači

U mnogim računarskim aplikacijama nije dovoljno samo prikazati podatke na ekranu, ili ih sačuvati u elektronskom obliku u memoriji, već je neophodno imati ih i u papirnoj formi. To omogućavaju različite vrste štampača. Najčešće korišćeni štampači su:

- matrični štampač
- inkdžet (*ink-jet*) štampač
- laserski štampač

Matrični štampači spadaju u grupu štampača sa dodirnim mehanizmom. Selektivnim pritiskom na traku natopljenu mastilom, mehanizam za štampanje ostavlja trag na papiru. Na pokretnoj glavi štampača, poređane u vertikalnoj liniji, nalaze se iglice. Ovih iglica obično ima 9 ili 24, što utiče na kvalitet štampe. Glava se pokreće horizontalno, dok se između nje i papira nalazi mastiljava traka. U toku pomeranja glave, pojedine iglice kratkotrajno izleću i udarajući traku ostavljaju otisak na papiru.

Ovakvi štampači su bučni, spori i male su rezolucije. Pored toga, sve odštampane tačkice su istog intenziteta, tako da su veoma loši za štampanje slika i uglavnom se koriste za štampanje teksta i tabela. Međutim, za razliku od ostalih vrsta štampača, korišćenjem indigo papira omogućavaju štampanje više kopija istovremeno i obično su veoma robustni i dugotrajni.

Inkdžet štampači formiraju sliku tako što kroz male otvore raspršuju naelektrisane kapljice boje na papir. Štampanje slike u boji se ostvaruje korišćenjem 3 ili 4 ovakva otvora kroz koje se raspršuju kapljice različite boje. Kombinacijom tri boje (žuta, cijan i magenta) i regulisanjem njihovog intenziteta

mogu se dobiti sve ostale boje. Ukoliko se sve tri boje pomešaju u jednakom intenzitetu, dobija se crna boja.

Radi uštede kolor kertridža (u kome se nalaze pomenute tri boje), često se koristi dodatni crni kertridž koji predstavlja četvrtu boju u sistemu.

Prednosti ovakvih štampača su jeftina izrada, tih rad i lako formiranje slike u boji. Osnovne mane su im: lošiji kvalitet štampe i sporiji rad u odnosu na laserske štampače i relativno visoka cena po otisku (odštampanom papiru).

Laserski štampači su najzastupljenija vrsta štampača. To duguju pre svega veoma dobrom kvalitetu otiska i velikoj brzini rada. Iako je sam štampač relativno skup, cena po otisku je vrlo niska. Sa jednim tonerom laserski štampači mogu da odštampaju od 2500 do 7000 papira formata A4 u zavisnosti od kapaciteta tonera.

Oni rade na sledećem principu: laserski snop promenljivog inteziteta osvetljava valjak čija je površina osetljiva na svetlost. Valjak se okreće i osvetljena površina ulazi u komoru sa toner prahom. Na površinu valjka koja je osvetljena lepi se toner prah, koji se zatim daljom rotacijom prenosi na papir. Toner se na papiru učvršćuje pomoću dodatnog grejača, preko koga papir prolazi na putu do izlaska iz štampača.

Brzina štampanja laserskih štampača iznosi od 12 do 30 strana u minuti za klasične laserske štampače, pa sve do 300 strana u minuti za štampače koji se koriste u velikim sistemima za opsluživanje više korisnika.

Pored brzine, laserski štampači se razlikuju i po kvalitetu štampe. Kvalitet štampe je u najvećoj meri određen brojem tačaka po inču. Danas se proizvode štampači koji imaju 1200 tačaka po inču ili više, dok su raniji standardi bili 300 i još uvek zastupljeni 600 tačaka po inču.

U poslednjih desetak godina razvijeni su i laserski štampači u boji. Mada su nekoliko puta skuplji od crno-belih laserskih štampača, sve više ulaze u upotrebu jer formiraju veoma kvalitetan otisak u boji.

Vežbanja

1. Čemu služi matična ploča personalnog računara? Koji su njeni najvažniji delovi? O čemu se mora voditi računa pri izboru matične ploče?

2. Kako se procesor i operativna memorija priključuju na matičnu ploču? Šta je *ZIF* sistem?

3. Čemu služe i gde se nalaze ekspanzioni slotovi? Navesti vrste ekspanzionih slotova i uporediti ih po brzini.

4. Čemu služi i koje osobine ima *PCI* slot?

5. Navesti razloge nastanka i glavne osobine *PCI Express* slota.

6. Ukratko opisati osobine *AGP* slota.

7. Objasniti ulogu čipseta. Navesti od čega se čipset sastoji i gde je smešten unutar računara.

8. Čemu služe portovi? Navesti vrste portova i ukratko ih opisati.

9. Koje su osnovne osobine *USB* porta?

10. Dati glavne osobine *FireWire* porta.

11. Navesti karakteristike *PS*/2 porta.

12. Dati osnovne karakteristike infracrvenog (*IrDA*) porta.

13. Šta predstavlja *BIOS* i koje funkcije obavlja? Gde se nalazi i kog je tipa memorija u kojoj je smešten *BIOS*?

14. Šta je *CMOS* i čemu služi? Odakle dobija napajanje kada je računar isključen?

15. Čemu služi operativna memorija i šta se u njoj nalazi. Kog je tipa i kakve posledice to prouzrokuje? Objasniti pojam „virtuelne“ memorije.

16. Opisati konstrukciju hard diska. Kakva je uloga glava hard diska? U kom obliku se binarne vrednosti (0 i 1) pamte na hard disku?

17. Opisati prostornu organizaciju podataka na hard disku (ilustrovati slikom).

18. Opisati *ZBR* (*Zone Bit Recording*) tehnologiju (problem koji rešava i rešenje).

19. Opisati *Cylinder skew* tehnologiju (problem koji rešava i rešenje).

20. Koji parametri utiču na performanse hard diska? Šta su interna i eksterna brzina prenosa podataka?

21. Kako se određuje vreme pristupa podacima na hard disku?

22. Kako brzina rotacije ploča, gustina zapisa i dimenzije ploča utiču na performanse hard diska?

23. Objasniti princip rada *CD-ROM* uređaja.

24. Opisati organizaciju podataka na *CD-ROM* disku.

25. Ukratko predstaviti osnovne karakteristike *CD-R* tehnologije.

26. Dati kratak prikaz *CD-RW* tehnologije.

27. Dati uporedni prikaz *DVD* i *CD* tehnologija.

28. Navesti osnovne osobine *USB flash* diska.

29. Šta je keš memorija i čemu služi? Objasniti princip rada keš memorije.

30. Opisati *L1* keš memoriju (gde se nalazi, koliki je radni takt, u kojim oblicima se javlja i kada se koji od njih koristi, cena, kapacitet).

31. Opisati *L2* keš memoriju (gde se nalazi, koliki je radni takt, kapacitet i cena u odnosu na *L1*).

32. Navesti osnovne karakteristike *CRT* tehnologije za izradu ekrana monitora personalnih računara.

33. Navesti osnovne karakteristike *LCD* tehnologije za izradu ekrana monitora personalnih računara.

34. Koja je razlika između monitora sa *CRT* i *LCD* ekranom?

35. Navesti osnovne karakteristike *PC* monitora i ukratko ih objasniti.

36. Objasniti princip rada matričnog štampača.

37. Objasniti princip rada inkdžet štampača.

38. Objasniti princip rada laserskog štampača.

UNIVERZITET
PRIVATNI
AKREDITOVANI
UNIVERZITET
SINGIDUNUM

Literatura

1. Borivoj Lazić, *Osnovi računarske tehnike: prekidačke mreže*, Akademska misao, 2006.

2. Carl Hamacher, Zvonko Vranešić, Safwat Zaky, *Computer Organization*, McGraw-Hill, Inc., 1984.

3. Jozo Dujmović, *Programski jezici i metode programiranja*, Naučna knjiga, 1990.

4. Jovan Đorđević, *Osnovi računarske tehnike - Zbirka rešenih zadataka*, Viša poslovna škola, Blace, 2006.

5. Michael Flynn, *Computer Architecture: Pipelined and Parallel Procesor Design*, Jones and Bartlett Publishers, Inc., 1995.

6. Mirko Vermezović, Prezentacija iz predmeta Osnovi računarske tehnike na Fakultetu za poslovnu informatiku Univerziteta Singidunum, 2006.

7. www.wikipedia.org

Na osnovu člana 23. stav 2. tačka 7. Zakona o porezu na dodatu vrednost („Službeni glasnik RS”, br. 84/04... i 61/07), Odlukom Senata Univerziteta Singidunum, Beograd, broj 260/07 od 8. juna 2007. godine, ova knjiga je odobrena kao osnovni udžbenik na Univerzitetu.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004(075.8)

ТОМАШЕВИЋ, Виолета, 1965-
Osnovi računarske tehnike / Violeta Tomašević. - 4. izd. - Beograd : Univerzitet Singidunum, 2012 (Loznica : Mladost Grup). - V, 205 str. : ilustr. ; 24 cm

Tiraž 300. - Bibliografija: str. [207].

ISBN 978-86-7912-406-7

a) Рачунарство
COBISS.SR-ID 189069836