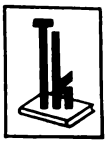
Časlav Dinić

PECOM 64

Tehnička knjiga



Recenzent: Dr Boško Damjanović

Urednik: Radivoje Grbović

YUISBN 86-325-0135-6

Časlav Dinić, dipl. inž.

PECOM 64

Tehnička knjiga

Sadržaj

Predgovor

Ova knjiga predstavlja prvi priručnik posvećen domaćem mikroračunaru PECOM 64. Autor se trudio da na pristupačan način izloži organizaciju ovog računara, elemente BASIC-a, arhitekturu mikroprocesora CDP 1802, asembler i računarsku učionicu.

Knjiga je namenjena, pre svega, učenicima i nastavnicima osmog razreda osnovne škole, kao i učenicima i profesorima prvog razreda srednje škole za nastavu iz informatike.

U knjizi je urađeno dosta primera koji ilustruju mogućnosti ovog kućnog računara, kao i niz praktičnih saveta koji će, svakako, biti korisni imaocima PECOM-a 64.

Autor

1. Uvod

Organizacija kućnog računara PECOM 64 slična je organizaciji mnogih mikrokompjuterskih sistema. Na jednoj ploči elekronike nalaze se sledeće jedinice:

- Osmobitni mikroprocesor CDP 1802
- sekcija za generisanje video signala i tona, takozvani video interfejs sistem (skraćeno VIS), koji u sebi sadrži 4 Kbajta memorije za generisanje veličine, oblika i boje prikazanih znakova na ekranu
- ROM i RAM memorije ukupnog kapaciteta 64 Kbajta
- sekcija za obavljanje ulazno-izlaznih funkcija, koja podržava rad sa magnetnom trakom, štampačem i rad u računarskoj učionici.

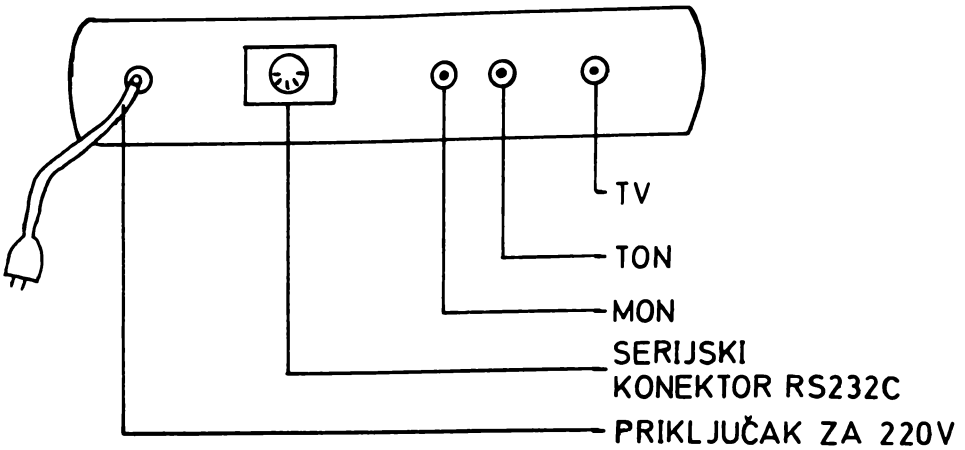
U PECOM-u 64 smešten je RF modulator čija je uloga generisanje video signala za prikazivanje na ekranu i signala tona. Tastatura se nalazi na poklopcu kutije. Ona je poluprofesionalna i sadrži 55 tastera za generisanje znakovnih podataka, grafičkih i kontrolnih znakova. Za generisanje tona PECOM 64 koristi tonski deo priključenog TV prijemnika.

Napajanje elektronskih komponenti računara vrši se pomoću stabilisanih izvora napajanja, koji se nalaze na ploči elektronike i ulazni napon dobijaju sa trafoa. Zbog male potrošnje elektronskih komponenti dimenzije trafoa su takve da je bilo moguće njegovo unošenje u istu kutiju zajedno sa pločom elektronike i tastature.

Za prikazivanje znakova i grafike može se koristiti standardni TV prijemnik (kolor ili crno-beli) ili monitor. Znaci i simboli prikazuju se u 24 linije sa 40 znakova u liniji. U memoriji se nalazi standardni skup od 128 znakova, simbola i grafičkih simbola. Znaci se mogu programski modifikovati po veličini i boji. Na ekranu se mogu prikazivati znaci i u 8 boja, s tim što se osnovna boja ekrana može predstavljati takođe u 8 boja. Kod posebnih verzija PECOM-a 64 može se koristiti grafika "visoke rezolucije" (240x216 tačaka), gde se čitav set od 128 znakova može programski predefinisati. Za generisanje tona koristi se tonski deo TV prijemnika, čime se dobijaju kvalitetne tonske karakteristike sa mogućnošću menjanja tona, oktave i jačine.

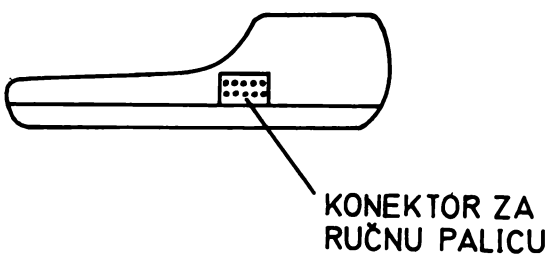
Na zadnjoj strani kutije PECOM-a 64 (sl.1.1.) nalaze se sledeći priključci:

- TV konektor za antenski ulaz u TV prijemnik
- TON — poseban konektor za izdvajanje tonskog signala
- MON — konektor za antenski ulaz u monitor
- RS232C — serijski konektor, koji omogućava povezivanje PECOM-a 64 za rad sa kasetofonom i štampačem. Takođe, ovim konektorom vrši se povezivanje računara za rad u računarskoj učionici.
- priključak za 220V



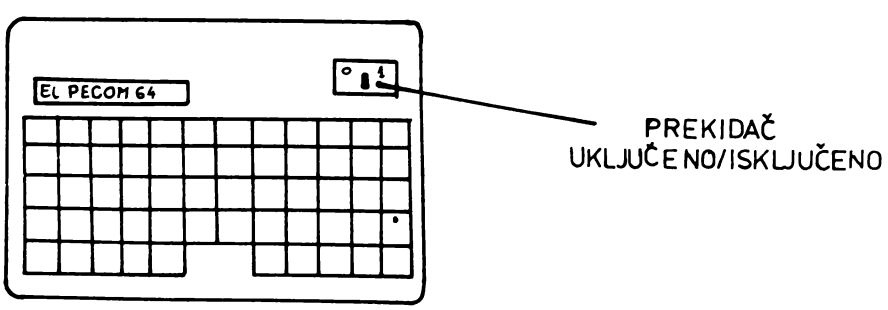
Sl.1.1

Na desnoj strani kutije (slika 1.2.) smešten je konektor za priključivanje ručne palice, kojom se kursor može pomerati levo, desno, gore i dole.



Sl.1.2

Na prednjoj strani kutije (slika 1.3.) nalazi se prekidač za uključivanje i isključivanje napajanja računara. Kada se prekidač nalazi u krajnjem levom položaju računar je isključen, a kada se prebaci u krajnji desni položaj tada je uključen u mrežno napajanje.



Sl.1.3

1.2. ORGANIZACIJA KUĆNOG RAČUNARA PECOM 64

PECOM 64, kao i svi ostali kućni računari, da bi obavljao potrebne funkcije u sebi mora da sadrži memoriju, centralnu procesorsku jedinicu i ulazno-izlaznu jedinicu.

Na sl.1.4. predstavljena je detaljna šema organizacije PECOM-a 64. Memorijski prostor od 64 Kbajta podeljen je na 32 Kbajta ROM memorije smeštene u EPROM-u i 32 Kbajta RAM memorije u kojoj korisnik može da smešta programe i podatke.

Ulazno-izlazna jedinica omogućava vezu memorijskih delova računara i mikroprocesora sa perifernim uređajima računara. Na taj način omogućeno je prikazivanje podataka i poruka na ekranu, komunikacija sa kasetofonom i štampačem i povezivanje računara za rad u sistemu računarske učionice.

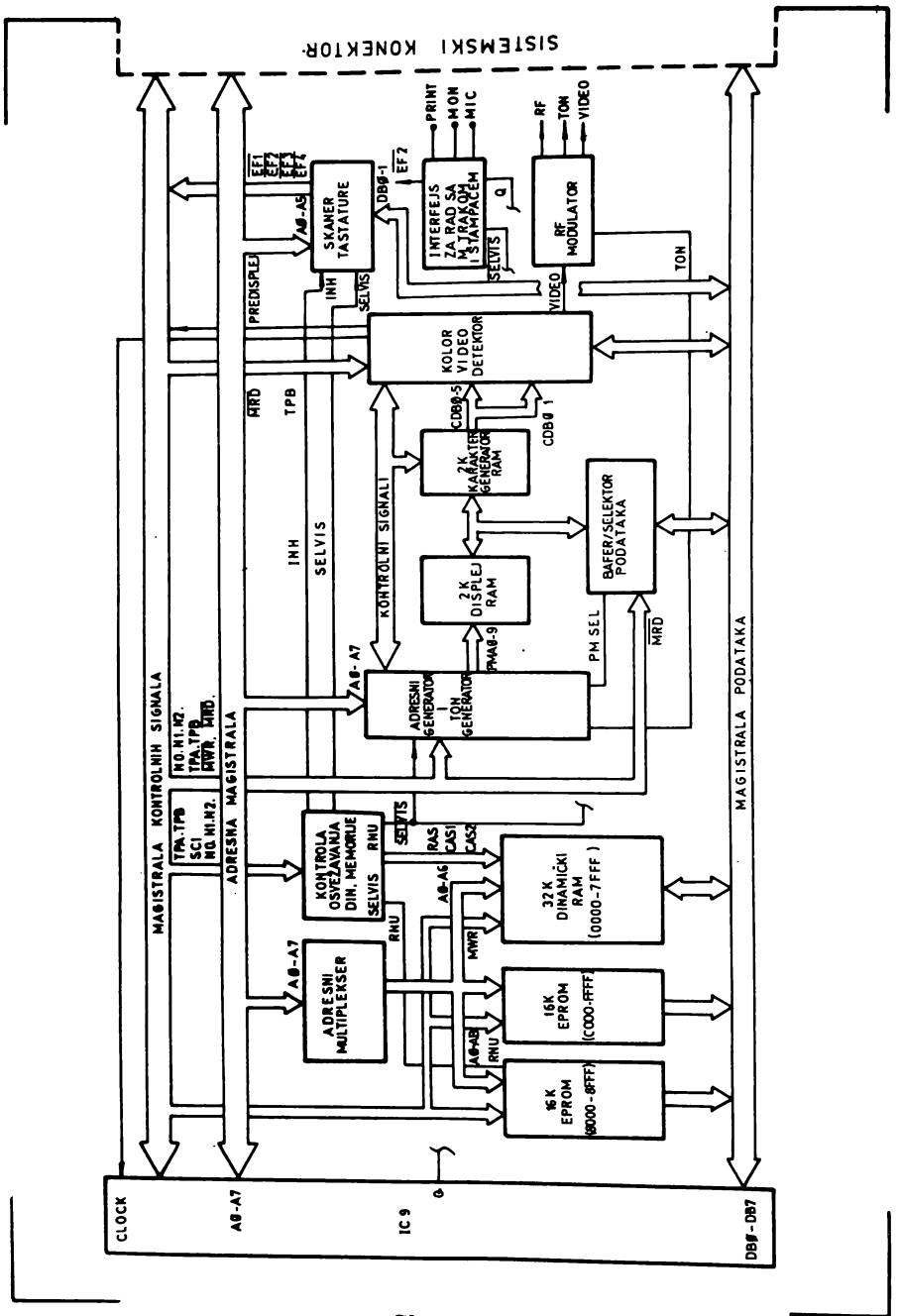
Mikroprocesor CDP 1802 je osmobitno integralno kolo koje radi sa učestanošću od 2,8 MHz, koju dobija iz kolor video generatora (kolo CDP 1870). Mikroprocesor CDP 1802 obavlja kontrolu ili je u vezi sa ostalim jedinicama računara preko 3 standardne ulazno-izlazne magistrale:

— adresne magistrale
— magistrale podataka
— magistrale kontrolnih signala.

Kada mikroprocesor želi da stupi u vezu sa nekom komponentom računara, on mora na neki način da obavesti čitav sistem o tome koja ga komponenta konkretno interesuje (koji bajt memorije, ili koji periferni uređaj). Drugim rečima, mikroprocesor mora da adresira željenu komponentu sistema. To postiže tako što odgovarajuće električne nivoe dovede na linije, takozvane **adresne magistrale**. U jednom trenutku, jedna linija magistrale može biti ili na naponu 0V (logička nula), ili na naponu +5V (logička jedinica). Svaka kombinacija nula i jedinica na magistrali predstavlja adresu jednog bajta u memoriji, ili jednog uređaja na periferiji. Kod PECOM-a 64 adresna magistrala se sastoji od 8 paralelnih linija, preko kojih se prenose adrese označene sa A0 do A7. Sa ovih 8 adresa moguće je napraviti $2^8 = 256$ kombinacija, a samim tim i adresirati 256 različitih memorijskih lokacija, što je premalo i za najjednostavnije računare. U PECOM-u postoje kola, takozvani adresni multiplekseri, koji formiraju još 8 novih adresa označenih A8 do A15. Na taj način dobijamo šesnaestobitnu adresu, sačinjenu od adresa A0 do A7 koje nazivamo nižim, i adresa A8 do A15 koje nazivamo višim adresama. Šesnaestobitnom adresom može se adresirati 64 Kbajta memorije.

Razmena informacija između mikroprocesora i ostalih elemenata računara vrši se preko posebnih električnih linija koje čine takozvanu **magistralu podataka,** koju čine 8 paralelnih linija, gde jednu liniju nazivamo DB linijom (skraćenica od DATA BUS = magistrala podataka), a celu magistralu DB magistralom, koja sadrži 8 DB linija obeleženih od DB0 do DB7.

Mikroprocesor može da postavi signale na magistralu podataka proizvoljno, tada on upisuje sadržaj. Sadržaj će pročitati neki od perifernih uređaja, ili će biti upisan u neku od memorijskih lokacija. Isto tako, procesor može da pročita sadržaj magistrale, u kom slučaju upisivanje vrši neka od preostalih komponenti računara. Zato kažemo da je magistrala podataka dvosmerna.


SISTEMSKI KONEKTOR
PRINT
MON
MIC
RF
TON
VIDEO
SKANER TASTATURE
INTERFEJS ZA RAD SA MAGNETOFONOM I ŠTAMPAČEM
RF MODULATOR
PREDSPLEJ
INH
SELVIS
KOLOR VIDEO DETEKTOR
MRD
TPB
KONTROLNI SIGNALI
2K KARAKTER GENERATOR RAM
2K DISPLEJ RAM
BAFER/SELEKTOR PODATAKA
ADRESNI GENERATOR I TON GENERATOR
PM SEL
MAGISTRALA KONTROLNIH SIGNALA
ADRESNA MAGISTRALA
MAGISTRALA PODATAKA
KONTROLA OSVEŽAVANJA DIN. MEMORIJE
RAS
CAS1
CAS2
32K DINAMIČKI RAM (0000-7FFF)
ADRESNI MULTIPLEKSER
16K EPROM (C000-FFFF)
16K EPROM (8000-BFFF)
CLOCK
IC 9
DB0-DB7

Sl.1.4

Što se tiče adresne magistrale, ona je jednosmerna, jer jedino mikroprocesor vrši adresiranje.

Magistralu kontrolnih signala čine linije sa specijalnom namenom, preko kojih mikroprocesor stavlja na znanje svim ostalim delovima računara šta upravo radi, ili šta namerava da uradi. Isto tako, preko kontrolnih linija mikroprocesor prima informacije o tome šta ostali uređaji rade.

1.2.1. Memorijska jedinica PECOM-a 64

Već smo se upoznali sa načinom upisivanja i čitanja podataka i naredbi u računar. Sada ćemo objasniti gde se ti podaci smeštaju. Podaci se čuvaju u posebnom delu računara koji se naziva **memorija** (memory).

Računar za svoj rad koristi dve osnovne vrste memorije: unutrašnju i spoljašnju memoriju. **Unutrašnja memorija** je smeštena u samom računaru i odlikuje se velikom brzinom u radu. Kapacitet unutrašnje memorije često se uzima kao jedan od kriterijuma prilikom upoređivanja kvaliteta i snage srodnih klasa računara.

Unutrašnja memorija prihvata sve dolazne informacije i predstavlja izvor svih izlaznih podataka. Ona čuva program i podatke koji treba da se obrade za neodređeni period i bez promene, sve dok se ne dovedu novi podaci koji će ih zameniti. Istovremeno ona čuva podatke na takav način da su pristupačni za kasniju upotrebu u sistemu. Vreme potrebno da se prenese informacija iz date ćelije do centralne procesorske jedinice naziva se **vreme pristupa**, i predstavlja važnu osobinu memorija. Vreme pristupa treba da je što kraće i kod postojećih računarskih sistema je reda 0,1 mikro sekundi.

Relativno visoka cena unutrašnje memorije, kao i ograničen prostor, onemogućavaju smeštanje velikog broja informacija, koje su karakteristične za veći broj različitih primena. Zbog toga se koristi **spoljašnja memorija**, koja je u stanju da prihvati veliki broj informacija i da ih u svakom trenutku stavi na raspolaganje računaru. Takve informacije su statičke, i nalaze se na magnetnoj traci, disku, dobošu i sl. Sve su to elektromehanički sistemi, koji imaju znatnu inerciju. Zbog toga su spoljašnje memorije dosta sporije od unutrašnjih.

Unutrašnju memoriju je najlakše shvatiti kao dugačak niz ćelija u koje mikroprocesor može da upisuje sadržaje, i iz kojih može te sadržaje da čita. Svaka memorijska ćelija ima svoju adresu — ceo broj koji jednoznačno određuje njegov položaj u memoriji. Kada mikroprocesor želi da komunicira sa nekom memorijskom ćelijom, mora da postavi odgovarajuću adresu na adresnu magistralu, i tek tada da pozove memoriju. Jedna memorijska ćelija sastoji se od 8 elementarnih ćelija, koje su u stanju da upamte samo jednu od dve osnovne informacije, nulu ili jedinicu.

U zavisnosti od fizičke realizacije, struktura memorije može biti takva da mikroprocesor jedino ima pristup čitanju sadržaja, ili takva da mikroprocesor može po volji da upisuje i čita sadržaje. Memorija iz koje se može samo čitati naziva se ROM (Read Only Memory). Sadržaj ROM-a ostaje neizmenjen čak i ako isključimo računar. Deo memorije računara uvek je organizovan kao ROM, i tu su smešteni neki osnovni programi i podaci bez kojih sistem ne bi mogao da funkcioniše.

Drugi deo memorije, sa proizvoljnim pristupom nazivamo RAM (Random Access Memory). U RAM memoriju možemo upisivati podatke i smeštati naše programe. Prestankom napajanja čitav sadržaj RAM-a se nepovratno gubi.

Skup svih memorijskih lokacija kojima mikroprocesor može da pristupi čini takozvani memorijski adresni prostor ili memorijsku mapu. Pošto svakoj memorijskoj lokaciji odgovara jedna adresa, adresni prostor je jednak ukupnom broju svih različitih rasporeda nula i jedinica na 16 adresnih linija. Ovde se radi o varijacijama sa ponavljanjem 16-te klase od dva elemenata a njih ima tačno $2^{16} = 65\,536$. To znači da mikroprocesor CDP 1802 adresira 65 536 bajtova, odnosno 64 kilobajta.

Kod PECOM-a 64 32 Kbajta zauzima ROM memorija i 32 Kbajta RAM memorija.

ROM memorija se koristi da se u njoj smeste programi koji su neophodni za rad računara. Ona je postojana i ne gubi sadržaj sa prestankom napajanja. Na PECOM-u 64 ROM memorija je izvedena sa dva EPROM-a kapaciteta po 16 Kbajta. Naziv EPROM potiče od termina Električno programirljiv ROM, i označava ROM memoriju koja jedino električnim putem može da izmeni sadržaj.

U prvom EPROM-u, čije se memorijske lokacije adresiraju heksadecimalnim adresama od 8000 do BFFF, smešten je program koji prevodi naredbe BASIC jezika na mašinski jezik, koji računar razume i može da ga izvrši. U njemu je smeštena tabela svih naredbi programskog jezika BASIC. Kada se neka naredba preko tastature unese u računar, mikroprocesor je pronalazi u postojećoj tabeli, a zatim prelazi na tačno definisane memorijske lokacije u kojima su smeštene osnovne instrukcije koje računar treba da uradi da bi se izvršila uneta naredba.

U drugom EPROM-u, čije se memorijske lokacije adresiraju heksadecimalnim adresama od COOO do FFFF, smešteni su sledeći sistemski programi:

- MONITOR + — koji omogućava rad sa memorijom i rad na mašinskom jeziku
- ekranski EDITOR — koji služi za unošenje programa u računar na ASEMBLER-u, višem programskom jeziku, i njegovo prevođenje na mašinski jezik
- programi za podršku rada računara u računarskoj učionici
- programi za prevođenje slova iz latinice u ćirilicu i obrnuto.

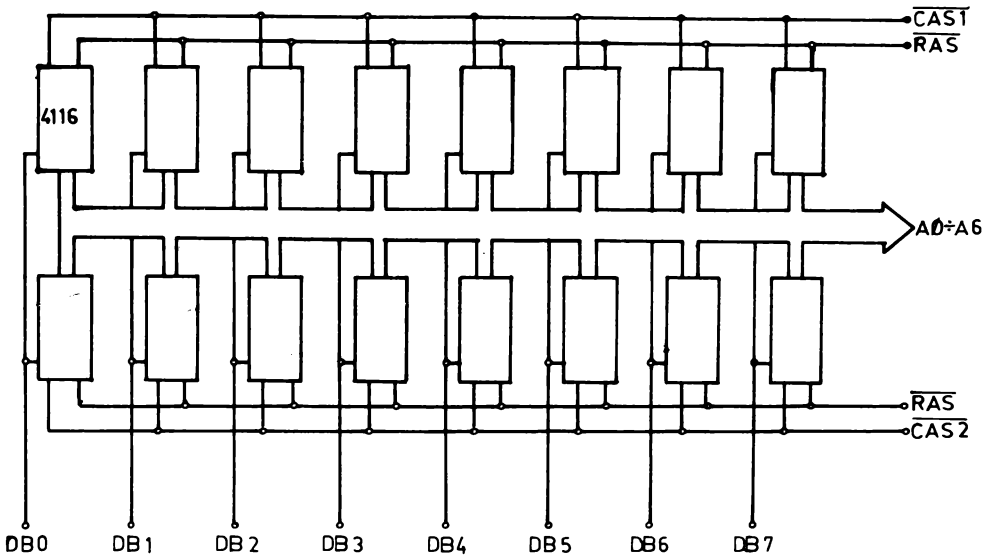
Iz ROM memorije mikroprocesor može samo da čita sadržaje memorijskih lokacija i ne može ih menjati.

RAM memorija zauzima deo memorijskog prostora koji je dostupan korisniku, zbog čega se naziva korisnička oblast računara. U nju mogu da se unose programi u neposrednom režimu (preko tastature), ili u posrednom režimu (preko magnetne trake ili flopi diska ako je računar povezan za rad u računarskoj učionici). Iz RAM memorije mikroprocesor može da čita sadržaje memorijskih lokacija ili da u određene memorijske lokacije upisuje podatke.

RAM memorija je kapaciteta 32 Kbajta i formirana je sa 16 memorijskih integralnih kola MK 4116 kapaciteta po 2 Kbajta. MK 4116 spada u grupu dinamičkih RAM memorija kod kojih sadržaj memorijskih lokacija mora stalno da se obnavlja. U tu svrhu se koriste posebni kontrolni signali računara.

Od celokupnog memorijskog prostora RAM memorija zauzima memorijski prostor čije se memorijske lokacije adresiraju heksadecimalnim adresama od 0000 do 7FFF. Deo tog memorijskog prostora, od 0000 do 02FF, pripada radnoj oblasti BASIC-a programskog jezika u koji smešta parametre potrebne za njegov rad. Organizacija RAM memorije prikazana je na slici 1.5.

16 memorijskih komponenti MK 4116 smešteno je u dva reda, pri čemu se u gornjem redu nalaze komponente u kojima su smeštene memorijske lokacije sa adresa-



Sl.1.5

ma od 0000 do 3FFF, a u donjem redu memorijske lokacije sa adresama od 4000 do 7 FFF.

Za adresiranje jednog reda memorija, kapaciteta 16 Kbajta, koristi se sedmobitna adresna magisrala (A0 do A7). Pristupanje memorijskim lokacijama RAM memorije vrši se u intervalima koje diktira kontrolni signal $\overline{CAS}$. Da li će se pristupiti gornjem ili donjem redu memorija odlučuju kontrolni signali $\overline{CAS1}$ i $\overline{CAS2}$.

Na koji način se vrši upis i čitanje podataka u RAM memoriji? U RAM memoriji se smeštaju osmobitni podaci na taj način što se jedan bit podatka smešta u jednu memorijsku lokaciju jedne od memorija 4116. Svi bitovi osmobitnog podatka smeštaju se u memorijske lokacije čija je adresa ista, i to najniži bit podataka u krajnju levu memoriju, a najviši bit podatka u krajnju desnu memoriju.

Kada mikroprocesor pristupa određenoj memorijskoj lokaciji, prvo postavlja kontrolne signale na osnovu kojih odlučuje da li će se podatak upisati u memoriju ili će se čitati iz nje. Zatim, na adresnoj magistrali postavlja adresne signale koji kodiraju adresu memorijskih lokacija kojima se pristupa. Na osnovu kodirane adrese mikroprocesor odlučuje da li će se pristupiti memorijskim lokacijama čija je heksadecimalna adresa manja ili veća od 4000. U tu svrhu se koriste kontrolni signali CAS1 i CAS2. Sedmobitnom adresom određuje se po jedna memorijska lokacija u svakom kolu 4116 u kojoj se može smestiti jedan bit podatka. Na kraju se, preko magistrale podataka, smešta 8-bitni podatak u izabrane memorijske lokacije ili se osmobitni podatak iz izabranih memorijskih lokacija šalje na magistralu podataka.

RAM memorije su nepostojane, i posle isključenja računara njihov sadržaj se briše.

Sadržaj ROM i RAM memorije može se prikazati na ekranu korišćenjem određenih naredbi koje će biti opisane kasnije, prilikom analize programskih jezika PECOM-a 64.

Memorija displeja i karakter generatora

U PECOM-u postoji još 4 Kbajta RAM memorije, koja ne ulazi u sastav opisanih 64 Kbajta memorije računara. Tu spadaju:

— memorija displeja
— memorija karakter generatora.

Memorija displeja je kapaciteta 2 Kbajta i u njoj se smeštaju podaci kojima se definiše sadržaj koji se prikazuje na ekranu. Svaka pozicija na ekranu ima svoju memorijsku lokaciju u displej memoriji. Zavisno od toga kakav se sadržaj upisuje u određenu memorijsku lokaciju displej memorije, zavisiće šta će se prikazati na ekranu u poziciji koja odgovara datoj memorijskoj lokaciji.

Karakter generator je, takođe, kapaciteta 2 Kbajta i u njega se smeštaju podaci koji određuju oblik svakog pojedinačnog znaka koji se prikazuje na ekranu. Za definisanje jednog znaka potrebno je 9 bajtova podataka, a samim tim i 9 memorijskih lokacija u karakter generatoru. Za definisanje svih 128 znakova potrebno je nešto više od 1 Kbajta memorije. Može se zaključiti da postoji mogućnost proširenja karakter memorije.

Adresiranje memorijskih lokacija displej i karakter memorije vrši adresni generator uz pomoć adresa i naredbi centralne procesorske jedinice.

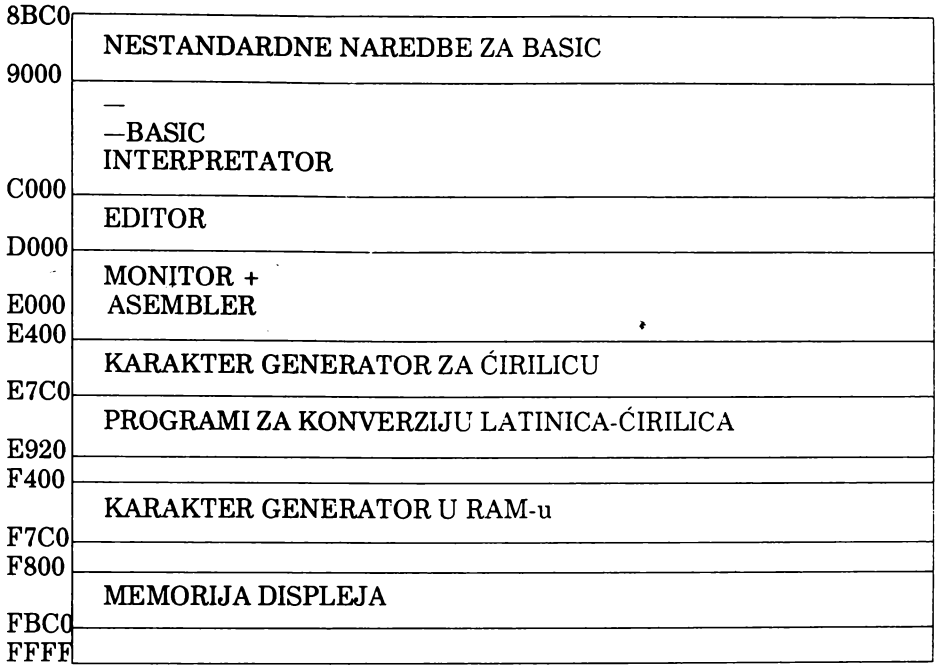
Sadržaj displej i karakter memorije računar može da čita u toku izvršavanja naredbi, koje kao rezultat izvršenja daju određen prikaz na ekranu. U tu grupu spadaju naredbe za grafiku i boju ekrana i znakova.

O tome, da li će se neki sadržaj upisivati u displej i karakter memoriju ili će se iz njih čitati, odlučuju posebna kola računara, takozvani bafer-selektori podataka. Oni služe kao neka vrsta kapije koja, u zavisnosti od kontrolnih signala koje dobijaju sa mikroprocesora, u displej i karakter memorije upisuje podatke koji se nalaze na magistrali podataka, ili iz displej i karakter memorije šalje podatke na magistrfalu podataka.

Displej i karakter memorija razlikuju se od RAM i ROM memorija u tome što se sadržaj njihovih memorijskih lokacija ne može prikazati na ekranu.

U tabeli 1.1. dat je prikaz dodele ukupnog memorijskog prostora u PECOM-u 64.

0000	RADNA OBLAST BASIC-a
0300	— — — KORISNIČKA — OBLAST — —
7000	
7CA0	STEK
7CB0	BAFER TASTATURE — UPIS-ČITANJE SA TRAKE
8000	USLUŽNI PROGRAM
8800	INICIJALNI KARAKTER GENERATOR



1.2.2 Centralna procesorska jedinica (mikroprocesor) PECOM-a 64

Mikroprocesor 1802 je 8-bitna centralna procesorska jedinica organizovana sa registrima. Koristi se kao računarski ili kontrolni element sa upamćenim programom. Mikroprocesor 1802 sadrži sva kola koja su neophodna za pripremu i izvršenje naredbi smeštenih u memoriju. Za njegov rad potreban je samo jedan taktni signal od 2,8 MHz, koji dobija iz kolor video generatora.

Arhitektura mikroprocesora zasnovana je na registarskoj matrici, koja sadrži šesnaest 16-bitnih registara opšte namene. Svaki registar iz registarske matrice R označen je jednom heksadecimalnom cifrom od 0 do F.

Za one programere koji svoje programe pišu isključivo u BASIC programskom jeziku, nije potrebno poznavanje funkcija pojedinih registara u mikroprocesoru, zato što BASIC programski dodeljuje potrebne uloge pojedinim registrima. Međutim, za one koji svoje programe pišu u asembleru ili mašinskom jeziku, bitno je poznavanje uloge pojedinih registara. Zbog toga ćemo se vratiti opisu funkcija pojedinih registara mikroprocesora 1802, u poglavljima koja obrađuju programski jezik asembler i mašinski jezik.

Mikroprocesor 1802 u svom radu u stalnoj je vezi sa memorijskim i ulazno-izlaznim jedinicama računara. U tu svrhu koristi:

— osmobitnu adresnu magistralu
— osmobitnu magistralu podataka
— ulazno-izlaznu magistralu od 15 kontrolnih signala.

U RAM memoriju mikroprocesor smešta programe koji se preko tastature unose u računar, ili se usnimavaju sa neke od spoljašnjih memorija, kao i rezultate dobijene izvršenjem unetih programa. Mikroprocesor adresnom magistralom određuje memorijske lokacije u kojima se smeštaju programi i rezultati izvršenja programa. Samo smeštanje podataka vrši se preko magistrale podataka.

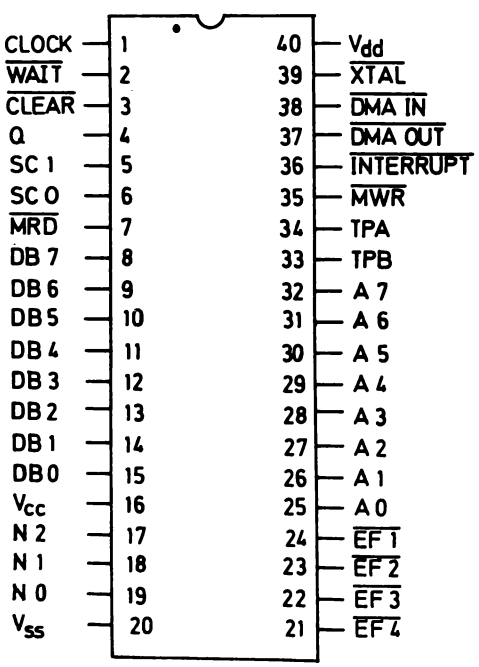
Iz ROM memorije mikroprocesor može samo da čita sadržaje određenih memorijskih lokacija, u kojima su smeštene osnovne instrukcije koje treba da uradi da bi izvršio program koji se nalazi smešten u RAM memoriji.

Za adresiranje ROM memorije mikroprocesor koristi 16-bitnu adresnu magistralu koja je sastavljena od:

— 8 adresnih linija adresne magistrale
— 8 adresnih linija, koje se formiraju u posebnom kolu računara (adresni multiplekser). Ove adrese su označene sa A8 do A15. Ulazno-izlaznom magistralom mikroprocesor komunicira sa perifernim jedinicama računara: tastaturom, ekranom, kasetofonom, štampačem itd.

Pomoću ovih magistrala mikroprocesor ima potpunu kontrolu nad celokupnim računarom, i on je taj koji odlučuje da li će se neki podatak ili naredba preneti nekoj jedinici računara ili ne. Iz tih razloga mikroprocesor nazivaju srcem računara.

Na sl.1.6. prikazana je konfiguracija mikroprocesora 1802 sa nazivima signala.



Sl.1.6

Funkcije pojedinih signala su:

DB0+DB7 je 8-bitna dvosmerna magistrala podataka

A0+A7 je 8-bitna adresna magistrala

$\overline{EF1}$+$\overline{EF4}$ su četiri ulazna indikatora stanja spoljnih uređaja koje postavljaju ulazno-izlazna kola. Na taj način se mikroprocesoru daje informacija o stanju u kom se nalaze pojedina ulazno-izlazna kola

Q je jednobitni serijski izlaz mikroprocesora preko koga se šalju podaci na magnetnu traku.

N0+N2 su programski kontrolisani izlazi koji pokazuju koja je od maksimalno 8 perifernih jedinica priključena na mikroprocesor.

$\overline{INTERRUPT}$ — signalom ulazno-izlazna kola postavljaju zahtev za prekid rada računara

$\overline{DMA\text{-}IN}$, $\overline{DMA\text{-}OUT}$ su signali kojima se ostvaruje direktan pristup memoriji (upis i čitanje)

SC0, SC1 su kodovi stanja koji definišu da li se mikroprocesor nalazi u stanju pripreme, izvršenja naredbe ili stanju programskog prekida.

TPA, TPB su vremenski signali koji definišu u kom trenutku su podaci na magistrali podataka važeći

$\overline{MWR}$ je signal koji definiše upis podataka u memoriju

$\overline{MRD}$ je signal koji definiše čitanje sadržaja memorije

CLOCK je taktni impuls učestanosti 2,8 MHz

$\overline{WAIT}$, $\overline{CLEAR}$ su signali koji definišu četiri moguća načina rada mikroprocesora: punjenje, reset, pauza i rad

V_{cc}, V_{dd} je napajanje mikroprocesora jednosmernim naponom od +5V

V_{ss} je masa

Mikroprocesor I802 je programirajuća komponenta, čiji se rad određuje naredbama (instrukcijama) koje izvršava. Naredba se iskazuje nizom bitova koji definišu rad računara, i osnovna je komanda koju računar razume. Naredbom se može:

— preneti podatak između mikroprocesora i memorije
— izvršiti aritmetička ili logička operacija nad podatkom
— upravljati ulaznim i izlaznim uređajima računara
— doneti odluka sa kojom naredbom nastaviti izvršenje programa
— uticati na stanje mikroprocessora.

Naredbe mikroprocesora mogu se izražavati:

— na mašinskom jeziku (nizovima bitova)
— na simboličkom jeziku (nizovima znakova).

Memorija računara puni se programima napisanim na mašinskom jeziku, koji mikroprocesor jedino razume. Ako program pišemo na simboličkom jeziku (na primer: BASIC, ASEMBLER), pre unošenja programa u memoriju on mora biti preveden na mašinski jezik mikroprocesora.

U fazi uzimanja naredbi mikroprocesor obavlja sledeće aktivnosti:

— Poseban registar ima ulogu brojača naredbi. Sadržaj tog registra predstavlja adresu memorijske lokacije iz koje mikroprocesor uzima naredbu za obradu i prenosi je u posebne registre koji obrazuju registar naredbi.
— Uvećava za jedan sadržaj registra koji ima ulogu brojača naredbi. Time je brojač naredbi pripremljen da ukazuje na adresu memorijske lokacije u kojoj je smeštena sledeća naredba.

Uzetu naredbu analizira upravljački deo mikroprocesora. On je prepoznaje i generiše odgovarajući niz upravljačkih signala, neophodnih za njeno izvršenje.

1.2.3. Ulazno-izlazna jedinica PECOM-a 64

PECOM 64 ne bi mogao obavljati nikakve operacije samo pomoću memorije i mikroprocesora. On mora da poseduje tastaturu preko koje će programer izdavati instrukcije da izvršava konkretne naredbe i vrši različita izračunavanja. Rezultati takvih izračunavanja nalaze se smešteni u memoriji. Da bismo ih koristili moraju biti preneti iz memorije na ekran ili na papir štampača.

Za trajno čuvanje programa moramo koristiti neku od spoljašnjih memorija, jer po isključenju računara sadržaj programa smeštenog u RAM memoriji briše se. Najrasprostranjenija spoljašnja memorija je magnetna traka kasetofona. Program koji smo usnimili na kasetu možemo u svakom trenutku uneti u računar.

Da bi bila omogućena veza računara sa tastaturom, ekranom, kasetofonom i štampačem moraju postojati elektronske komponente koje će to omogućiti. Skup tih komponenti čini ulazno-izlaznu jedinicu računara.

1.2.4. Tastatura PECOM-a 64

Za direktno komuniciranje sa PECOM-om 64 konstruisano je nekoliko programskih jezika kod kojih se naredbe predstavljaju skupom znakova koje koristimo u svakodnevnom pisanju. Naredbe u takvom obliku unose se u računar preko tastature, ulazne periferne jedinice računara.

Tastatura je sa ostalim delom računara povezana preko posebnih kola, takozvanih skanera tastature, čija je uloga da svakoj pritisnutoj dirki dodeli određeni broj (znakovni podatak). Skup tih brojeva računar će prepoznati kao konkretnu instrukciju i kao takvu će je izvršiti.

Tastatura PECOM-a 64 (sl.1.7.) je poluprofesionalna sa 55 dirki, kojima je obuhvaćen standardni skup znakova (uključujući i skup znakova Č, Ć, Đ, Ž i Š) i funkcionalne dirke: CAPS LOCK, ESC, DEL, BREAK, CTRL, SHIFT, RETURN i LINE FEED.

Pritiskom na dirku CAPS LOCK vrši se izbor korišćenja malih ili velikih slova. Pritiskom na neku dirku sa dva simbola za vreme dok se drži pritisnuta dirka SHIFT, generiše se gornji simbol.

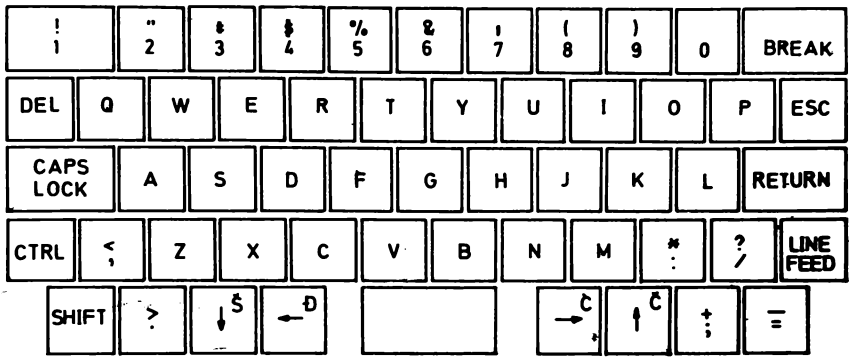
Dirkom RETURN vrši se vraćanje kursora u krajnji levi položaj na ekranu u sledećem redu. njome se, takođe, računaru šalje poruka koja označava kraj programskog reda prilikom pisanja programa.

Ako je dirka CTRL pritisnuta a naknadno se pritisne neka određena dirka, računar će na ekranu odštampati neki od specijalnih znakova ili će obaviti izvesne akcije. Na primer, pritisak na dirku H dok je dirka CTRL pritisnuta, prouzrokovaće da računar obriše prethodno uneti znak. Ako je pritisnuta dirka C, računar će ignorisati programski red koji se kuca, a kursor će pomeriti u krajnji levi položaj u sledećem redu.

Pritiskom na dirku LINE FEED računar će pomeriti kursor za jednu poziciju naniže.

Dirke DEL i ESC imaju ulogu pri radu PECOM-a 64 u računarskoj učionici.

Dirka BREAK omogućava zaustavljanje programa koji se izvršava.

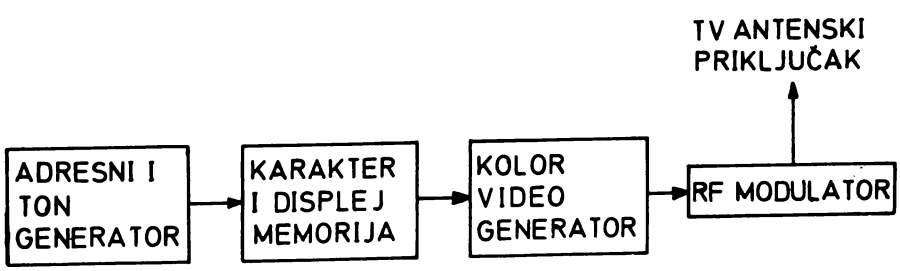


Sl.1.7

1.2.5. Ekran PECOM-a 64

Da bi smo mogli koristiti rezultate koje računar daje izvršavanjem programa, moramo ih prikazati na ekranu ili na papiru štampača. Kao ekran PECOM 64 koristi standardni TV prijemnik ili monitor. Ako se kao ekran koristi TV prijemnik, tada se na njegov antenski ulaz dovodi signal sa priključka TV, koji u sebi sadrži video signal slike i tonski signal. Ako se kao ekran koristi monitor, tada se na antenski ulaz monitora dovodi signal sa priključka MON, koji u sebi sadrži samo video signal.

Za generisanje slike na ekranu računar koristi posebnu izlaznu jedinicu video interfejs sistem (VIS) (sl.1.8.).



Sl.1.8

Uloga VIS-a je da izvrši sve one instrukcije računara koje kao rezultat treba da daju određeni prikaz na ekranu i da svaki znak, koji se preko tastature unese u računar, prikaže na ekranu.

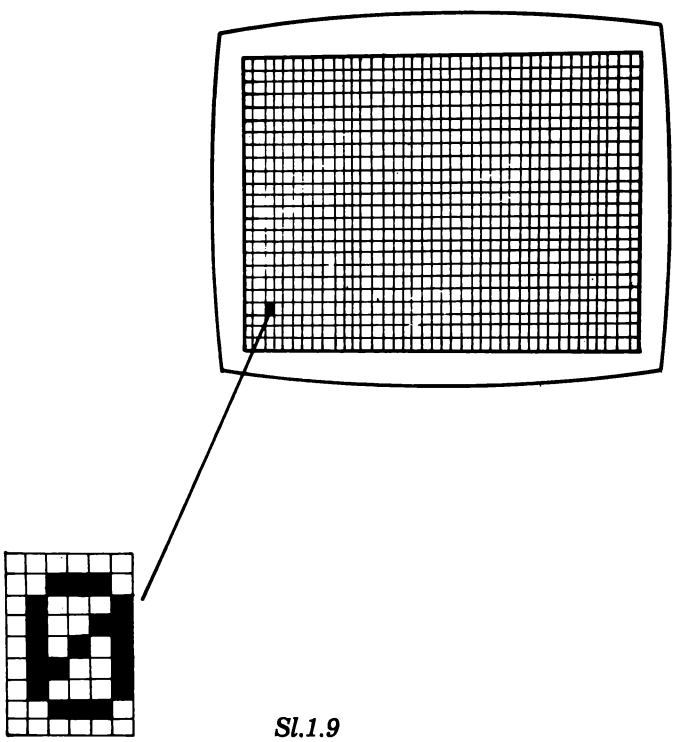
Adresni generator ima ulogu da adresira memorijske lokacije u displej i karakter memoriji, u kojima se nalaze zapisani podaci o veličini, obliku i boji znaka koji se prikazuje na ekranu.

Ton generator definiše učestanost, jačinu i oktavu signala tona, koji se u zvučni efekat pretvara u samom TV prijemniku.

Kolor i video generator ima za zadatak da sve ono što treba da se prikaže na ekranu generiše u signal koji će se slati u TV prijemnik. Preko njega mikroprocesor generiše jednu od 8 boja znaka, koji se prikazuje na ekranu, i jednu od 8 boja ekrana.

Kao posebna celina u računaru se nalzi RF modulator, koji spaja signal slike i tonski signal u jedinstveni signal, i koji se pretvara u oblik kakv imaju signali za prijem klasičnog TV programa.

Znaci se na ekranu prikazuju u 24 linije, po 40 znakova u liniji. Svaki znak se prikazuje matricom od 6x9 tačaka (sl.1.9.).

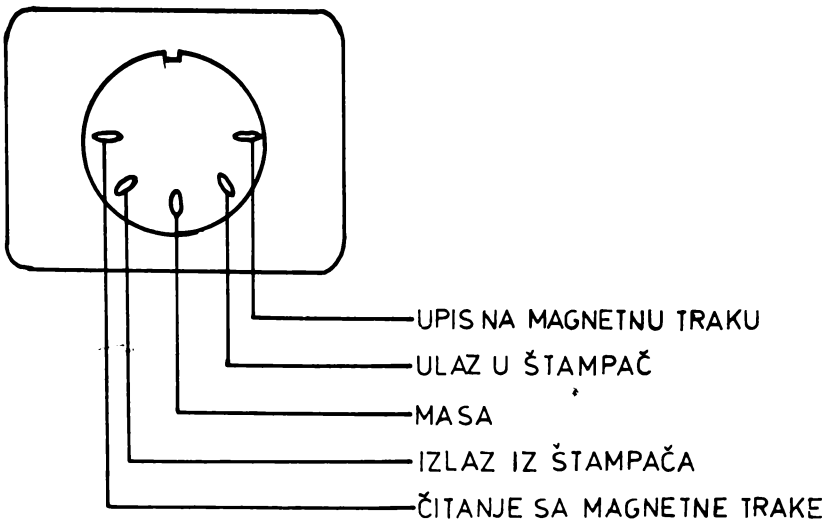


Sl.1.9

1.3. RAD PECOM-a 64 SA KASETOFONOM I ŠTAMPAČEM

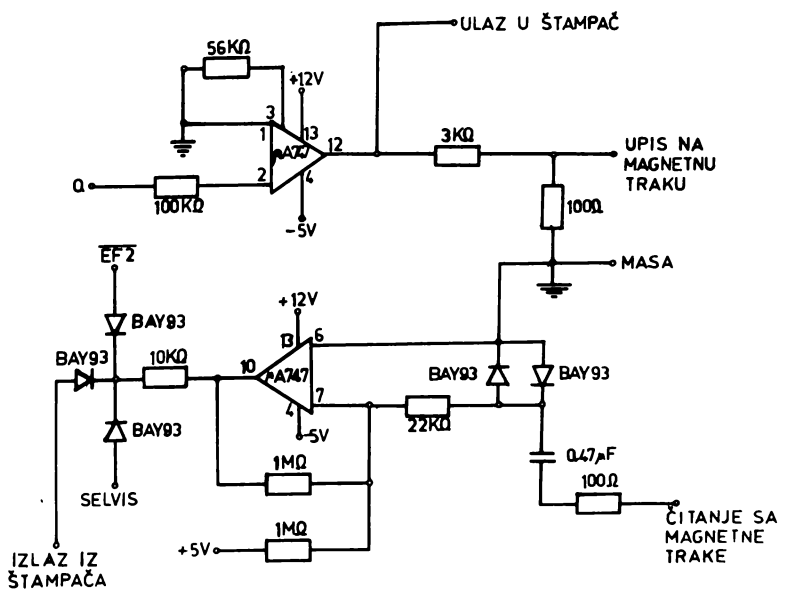
Za komunikaciju sa kasetofonom i štampačem, PECOM 64 koristi serijski konektor RS232C, koji se nalzi na zadnjoj strani kutije (sl.1.1.). Na konektoru (sl.1.10.) se nalaze priključci za sledeće signale:

- signal za upis programa na magnetnu traku
- signal za čitanje programa sa magnetne trake
- ulazni signal u štampač
- izlazni signal iz štampača
- masa.



Sl.1.10

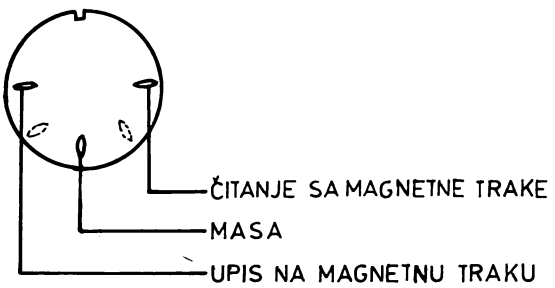
Šema interfejsa za povezivanje unutrašnjih delova računara sa kasetofonom i štampačem prikazana je na slici 1.11.



Sl.1.11

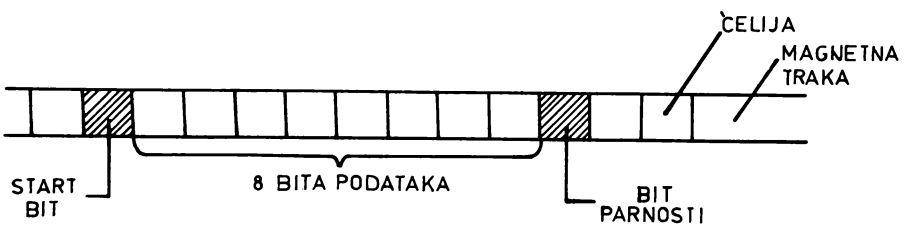
1.3.1. Rad PECOM-a 64 sa kasetofonom

Komunikaciju sa kasetofonom PECOM 64 obavlja preko kabla sa dva konektora, gde su oba konektora identično povezana na način prikazan na slici 1.12.



Sl.1.12

Podaci za upis na magnetnu traku šalju se preko serijskog izlaza Q mikroprocesora (sl.1.11.), a preko operacionog pojačivača 747 na izlaz "UPIS NA MAGNETNU TRAKU". Operacioni pojičavač ima ulogu da pojača signal koji se šalje na magnetnu traku. Iz računara se bitovi podatka šalju kao kombinacija jedinica i nula. Na traci se bitovi pamte signalima učestanosti 2 KHz za logičku nulu i 800 Hz za logičku jedinicu. Magnetna traka je izdeljena na ćelije, gde u svakoj ćeliji može da se smesti po jedan bit podatka (sl.1.13.).

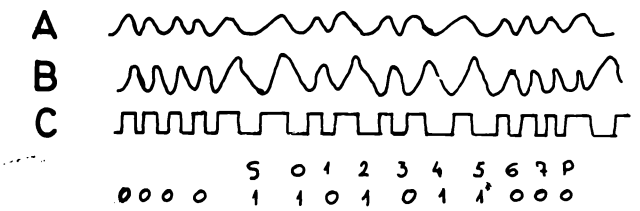


Sl.1.13

Podaci se na traku smeštaju serijski (jedan za drugim) u grupi po 8 bitova (1 bajt) podataka. Svaki bajt počinje signalom logičke jedinice (START BIT), koji definiše početak bajta podataka. Zatim slede 8 bita podataka, i završava se bitom parnosti. Bit parnosti se primenjuje da bi se utvrdilo da li je podatak tačno prenesen na traku. Na prijemnoj strani odbrojavaju se sve "1" u bajtu podatka koji je primljen. Ako je definisan neparan broj, a pri prijemu se dobije paran broj jedinica, onda će se javiti znak da je došlo do greške, tj. da je bar jedan bit podatka nestao ili je nastao tokom prenosa, jer je na prijemu bio paran broj. Da bi se na pogodan način kodirali potrebni znaci koristi se kod znaka, koji smo ranije opisali.

Podaci se sa trake čitaju preko ulaza "ČITANJE SA MAGNETNE TRAKE", a zatim se preko operacionog pojačivača 747, linijom EF2 šalju u mikroprocesor. Svako čitanje podataka sa magnetne trake počinje blokom signala na nivou logičke nule u trajanju oko 15 sec. Ovim blokom mikroprocesor obezbeđuje prepoznavanje početka zapisa.

Na slici 1.14. prikazani su vremenski dijagrami signala koji se generišu u toku komunikacije računara sa kasetofonom.



Sl.1.14

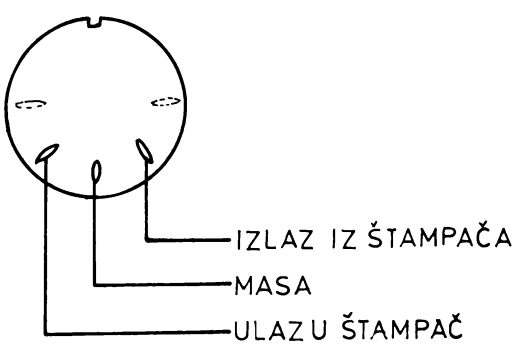
Signal A generiše se na 12-om izvodu operacionog pojačivača 747 (sl.3.2.) i predstavlja pojačani signal iz mikroprocesora koji u sebi sadrži bitove podataka, koji treba da budu usnimljeni na magnetnu traku.

U toku čitanja programa sa magnetne trake formira se signal B. Posle obrade u ulaznom stepenu računara na 12-om izvodu operacionog pojačivača dobija se signal C iz kog mikroprocesor može da prepozna bitove podataka.

U jednom bloku podataka računar prepoznaje signal S (START BIT), zatim 8 bita podataka, i na kraju bit parnosti P.

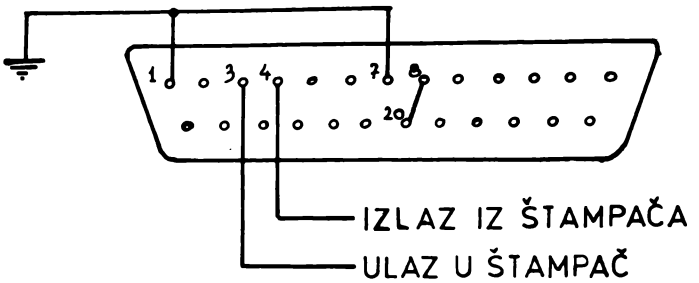
1.3.2. Rad sa štampačem

Komunikaciju sa štampačem PECOM 64 obavlja preko serijskog izvoda RS232C posebnim kablom, koji ima dva različita konektora. Šema povezivanja konektora za računar prikazana je na sl.1.15.



Sl.1.15

Izrada konektora za štampač zavisi od tipa štampača koji se koristi. U sistemu računarske učionice PECOM 64 radi sa štampačem RGB 105. Šema povezivanja konektora za ovaj tip štampača prikazana je na slici 1.16.



Sl.1.16

Sa izvoda "IZLAZ IZ ŠTAMPAČA" mikroprocesor prima pobudni signal iz štampača, iz čega zaključuje da je štampač spreman za prijem podataka iz računara.

Preko izvoda "ULAZ U ŠTAMPAČ" PECOM 64 serijski šalje podatke u štampač na osnovu kojih će zaključiti šta treba odštampati na papiru.

Da bi se omogućio rad računara sa kasetofonom i štampačem mora postojati i odgovarajuća programska podrška. Računar mora da zna da li će preuzeti aktivnosti za čitanje programa sa magnetne trake, ili upis programa na magnetnu traku, ili će određeni sadržaj prikazati na papiru štampača. Vrsta aktivnosti se određuje skupom naredbi koje se unose preko tastature.

1.4. PROGRAMSKA OPREMA (SOFTVER) PECOM-a 64

Procesor zna da izrši samo nekoliko matematičkih i logičkih operacija. On sam ne zna da pročita znak sa tastature, niti da taj znak ispiše na ekranu. Da bi te, i ostale funkcije, procesor uradio, u računaru mora da postoji nekoliko osnovnih programa. Ti programi su smešteni u ROM memoriji, odakle se mogu čitati i izvršavati.

Najosnovniji i najvažniji skup programa koji omogućavaju rad računara naziva se — **operativni sistem.**

Operativni sistem PECOM-a 64 ima nekoliko zadataka. Pre svega, po uključenju računara ovaj program treba da pristupi video memoriji, pomoću koje će obrisati ekran i ispisati poruku o izvršenoj inicijalizaciji.

Posle inicijalizacije, računar izvršava naredbe korisnika sve dok ne bude isključen. Operativni sistem nije neposredno zadužen za izvršavanje naredbi. To će činiti BASIC interpretator. Operativni sistem mora da održava rad računara, prima podatke koje korisnik otkuca i šalje ih dalje, kontroliše i osvežava sliku na ekranu.

Osnovna delatnost operativnog sistema su rad sa tastaturom i ekranom i komunikacija sa perifernim uređajima. On brine o zvuku koji računar generiše. Bilo bi dosta složeno kada bi ovaj zadatak radio BASIC interpretator, jer bi na taj način korisniku bilo otežano korišćenje zvuka.

Osim generisanja slike i tona, operativni sistem se bavi načinom unošenja programa u operativnu memoriju. On to može izvesti na sledeća dva načina:

— u posrednom kontaktu sa računarskim sistemom
— u neposrednom kontaktu sa računarskim sistemom.

Kada se problem rešava u posrednom kontaktu sa računarskim sistemom program sa ulaznim podacima se prvo unosi na neku od spoljašnjih memorija računara (magnetna traka, kartica, disk i sl.), a zatim se preko ulazne jedinice računara smešta u unutrašnju memoriju računara, u obliku koji centralna jedinica računara razume.

Kada se problem rešava u neposrednom kontaktu sa računarskim sistemom, program se unosi sa tastature, ili se naredbama unesenih preko tastature, smešta iz spoljašnje memorije u računar (ako se program već nalazi u spoljašnjoj memoriji). Ulazni podaci se unose preko tastature, a izlazni podaci i poruke se prikazuju na ekranu ili papiru štampača.

Tastatura (kao ulazni organ) s jedne, i ekran i štampač (kao izlazni organi) s druge strane, čine terminal računara.

PECOM 64 za svoj rad koristi modifikovani standardni programski jezik BASIC, takozvani BASIC 3. U BASIC-u 3 naredbe i komande standardnog BASIC-a su prilagođene arhitekturi kućnog računara PECOM 64, i njegovoj komunikaciji sa perifernim uređajima (kasetofonom, štampačem, ekranom).

U EPROM-u računara smešten je BASIC INTERPRETATOR, skup programa pomoću kojih se naredbe napisane u programskom jeziku BASIC prevode na mašinski jezik koji računar razume i može da izvrši.

BASIC interpretator ispituje svaku naredbu koja se preko tastature unosi u računar. Pre svega upoređuje da li je naredba korektno upisana ili ne. Ako je naredba nekorektno upisana, tada se na terminalu prikazuje poruka o greški. Ako je naredba korektno napisana tada interpretator ispituje da li naredba sadrži broj programskog reda ili ne. Ako je naredba bez broja programskog reda računar je odmah izvršava, a ako je sa brojem programskog reda tada se ona izvršava tek posle startovanja programa posebnom naredbom za startovanje.

Pored BASIC INTERPRETATORA u EPROM-u PECOM-a 64 nalaze se smešteni sledeći programi:

- programi koji nestandardne naredbe BASIC-a prevode u naredbe na mašinskom jeziku
- programi koji omogućavaju komunikaciju računara sa tastaturom, štampačem i kasetofonom
- karakter generator za generisanje slovnih znakova u ćirilici
- MONITOR+: programi za rad sa memorijom i programi za rad u mašinskom jeziku
- EKRANSKI EDITOR — CREDIT: skup programa koji omogućavaju unošenje programa u računar i vršenje njihovih ispravki
- programi za rad PECOM-a 64 u sistemu računarske učionice.

1.5. POVEZIVANJE PECOM-A 64 SA TV PRIJEMNIKOM I KASETOFONOM

Za povezivanje PECOM-a 64 za rad sa TV prijemnikom i kasetofonom, pored standardnog TV prijemnika i standardnog kasetofona, potrebni su sledeći kablovi:

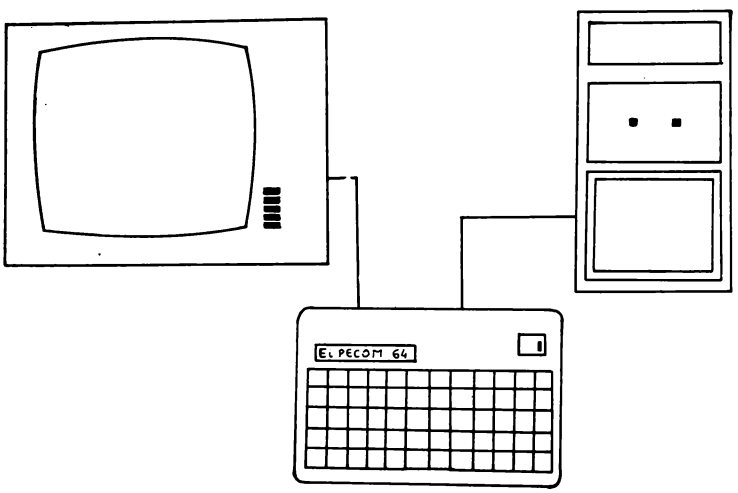
- antenski kabli sa konektorom za priključenje na antenski ulaz TV aparata
- kabli za povezivanje računara sa kasetofonom.

U toku povezivanja računar treba da je isključen, odnosno prekidač na prednjoj strani kutije (sl.1.3.) prebačen u krajnji levi položaj.

Da bi smo TV prijemnik povezali sa računarom, antenski kabl priključujemo na TV priključak računara (sl.2.1.1.), i na antenski ulaz TV prijemnika (obično se nalazi na poleđini TV prijemnika). Nije nam potreban spoljni RF modulator, pošto je isti ugrađen u sami računar.

Da bi smo kasetofon povezali sa računarom, jedan kraj kasetofonskog kabla priključujemo na serijski konektor RS232C računara, a drugi kraj kabla na petopolni priključak kasetofona.

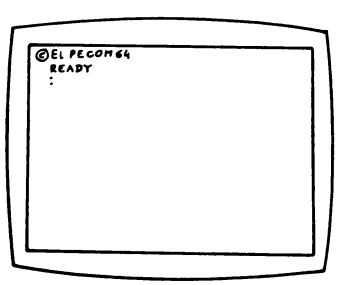
Na ovaj način dobijamo minimalnu konfiguraciju sistema (sl.1.17.).



Sl.1.17

Umesto standardnog TV prijemnika kao ekran računara možemo koristiti monitor. Način povezivanja monitora sa računarom sličan je načinu povezivanja standardnog TV prijemnika. Koristimo isti antenski kabl, s tom razlikom što kao izlaz iz računara koristimo priključak MON.

Podešavanje TV prijemnika za rad sa računarom vrši se u donjem delu VHF područja (između kanala2 i 3 za Pal sistem), sve dok se ne pojavi poruka u gornjem levom uglu ekrana (sl.1.18.).



Sl.1.18

Poruku koju PECOM 64 ispisuje u gornjem levom uglu ekrana zovemo **inicijalizacijom računara.**

Ako kao ekran koristimo monitor, tada nije potrebno nikakvo podešavanje na njemu, već se poruka o inicijalizaciji računara ispisuje odmah po uključenju sistema. Treba napomenuti da se u toku rada sa monitorom ne mogu videti rezultati izvršenja naredbi za boje znakova i ekrana, niti čuti rezultati izvršenja naredbi za ton.

1.6. POVEZIVANJE ZA RAD ŠTAMPAČEM

Konfiguraciju sistema za rad sa štampačem čini PECOM 64, standardni TV prijemnik (ili monitor) i serijski štampač RGB 105, kao i kabl sa dva konektora: konektor za priključivanje na računar i konektor za priključivanje na štampač.

2. Elementi BASIC jezika

BASIC je skraćenica engleskih reči Beginner's All-purpose Symbolic Instruction Code, koje u slobodnom prevodu znače: višenamenski simbolički jezik za početnike. Grupa asistenata i studenata Dortmouth univerziteta pod rukovodstvom profesora J.G.Kemenya i T.E.Kurtza, 1965. godine razvila je BASIC jezik za potrebe firme General Electric. Prve verzije BASIC jezika realizovane su za računare DATA—NET—30 i GE—235.

Cilj BASIC jezika je ostvarivanje aktivnog rada između korisnika i računara. U ovom režimu rada korisnik komunicira sa računarom posredstvom terminala i u procesu komunikacije rešava svoj zadatak. Ovakav režim rada odgovara ljudima kojima računarstvo i informatika nije profesija.

Odlike BASIC jezika su jednostavnost i pogodnost za rad i učenje. Ne postoji standardni BASIC jezik, već se u praksi susreću razne njegove varijante, zbog toga što verzija programskog jezika u velikoj meri zavisi i od hardverske strukture računara: vrste mikroprocesora, memorija i ulazno-izlaznih jedinica.

Za potrebe kućnog računara PECOM 64 formirana je verzija BASIC jezika nazvana BASIC 3. U daljim izlaganjima umesto termina BASIC 3 upotrebljavaćemo standardni termin BASIC.

2.1. KAKO SE ULAZI U REŽIM BASIC-a

Ako smo uspešno izvršili povezivanje PECOM-a 64 sa TV prijemnikom ili monitorom, odmah po uključenju, računar u gornjem levom uglu ekrana ispisuje tekst inicijalizacije (poglavlje 2.5.).

Ispisana poruka

```
© Ei PECOM 64
READY
:
```

znači da je računar obavio sve potrebne aktivnosti za prihvatanje i obradu podataka od strane programera.

Porukom READY računar daje doznanja da se nalazi u režimu rada BASIC programskog jezika. Programiranje vršimo na taj način što računaru naređujemo kako da izvrši postavljene zadatke. Te zadatke i sami možemo uraditi, ali bi nam za to bilo potrebno mnogo više vremena. Da bismo postupak programiranja pojednostavili, zadatak delimo na manje korake koje nazivamo naredbama.

2.2. ŠTA JE NAREDBA

Osnovne instrukcije koje računar razume i koje može izvršiti nazivamo **naredbom**. Reči pomoću kojih se formiraju naredbe čine programski jezik. BASIC, za formiranje naredbi, koristi sledeće simbole:

— slova: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
— cifre: 0 1 2 3 4 5 6 7 8 9
— interpunkcijske znake: . , : ! ? ()
— aritmetičke znake: + — * / ↑
— relacijske znake: > < =
— pomoćne znake: # $ % &

Sve ovo spada u **osnovne simbole.** Oni nisu dovoljni da bi se potpuno definisao BASIC. Da se ne bi izmišljali novi simboli od postojećih osnovnih formirani su **izvedeni simboli,** koji predstavljaju nazive naredbi koje se u BASIC-u koriste.

2.2.1. Naredbe sa direktnim izvršenjem

Naredimo računaru da sabere brojeve 5 i 6 i da rezultat prikaže na ekranu. To izvodimo na taj način što sa tastature unesemo naredbu

```
PRINT 5+6 (RETURN)
```

Oznaka (RETURN) znači da smo pritiskom na dirku RETURN završili upis programskog reda. Odmah posle toga računar na ekranu ispisuje rezultat, broj 11. Naredba PRINT biće u kasnijim razmatranjima detaljnije opisana. Za sada je dovoljno da znamo da njome izdajemo instrukcije računaru da prikaže na ekranu rezultat obrade naredbe koja sledi posle PRINT.

Ako bismo hteli da ponovimo računanje morali bismo kompletnu naredbu uneti sa tastature. Na ovakav način izvršavaju se **naredbe sa direktnim izvršenjem.** Ovakve naredbe još nazivamo **komandama.**

U slučaju naredbi sa direktnim izvršenjem pritisak na dirku RETURN ne označava samo završetak programskog reda, već vrši ulogu naredbe kojom se startuje izvršenje naredbe sabiranja i prikazivanja rezultata na ekranu.

2.2.2. Upotreba PECOM-a 64 kao kalkulatora

Preko tastature unesimo sledeću naredbu:

```
:PRINT 8+3
```

Naredbom izdajemo instrukciju računaru da na ekranu prikaže rezultat aritmetičke operacije 8+3. Naredbe ovog tipa nazivamo **kalkulatorskim,** a režim rada u kome se računar nalazi nazivamo **kalkulatorskim režimom** rada.

Posle pritiska na dirku RETURN računar će na ekranu prikazati rezultat

```
11
```

U kalkulatorskom režimu rada PECOM 64 može da obavlja 5 aritmetičkih operacija:

— sabiranje
— oduzimanje
— množenje

— deljenje
— stepenovanje.

Sabiranje i oduzimanje računar obavlja pomoću znakova + i —, koji se koriste i u matematici.

Preko tastature unesimo sledeću naredbu:

```
:PRINT 12+13
```

Posle pritiska na dirku RETURN računar će na ekranu prikazati rezultat

```
25
```

Pokušajmo sledeće:

```
PRINT 16-12 (RETURN)
```

Računar će odgovoriti rezultatom

```
4
```

Operaciju množenja računar obavlja pomoću znaka *. Unesimo u računar naredbu:

```
:PRINT 7*8
```

Posle pritiska na dirku RETURN dobićemo rezultat

```
56
```

Operaciju deljenja računar obavlja pomoću znaka /. Unesimo u računar narebu:

```
:PRINT 36/4
```

Posle pritiska na dirku RETURN dobićemo rezultat

```
9
```

Ponekad je potrebno pomnožiti neki broj samim sobom određeni broj puta. Pokušajmo sledeće:

```
:PRINT2*2*2*2*2
```

Posle pritiska na dirku RETURN dobićemo rezultat

```
32
```

Međutim, umesto da pišemo sve ovo, isto možemo predstaviti na sledeći način:

```
:PRINT 2↑5
```

Posle pritiska na dirku RETURN dobićemo isti rezultat kao u prethodnom primeru. Stoga je znak za stepenovanje (↑) jednostavna skraćenica računara za ponovno množenje istim brojem.

Operacija 2↑5 ima isto značenje kao i 2^5 u matematici, ali je računar tako projektovan da može prepoznati samo 2↑5 a ne 2^5. Slično tome, računar može da interpretira 7*8 i 36/4 ali ne i 7x8 i 36:4.

Svi gornji primeri sadrže dva broja i jednu operaciju. PECOM 64 može da rešava i složenije probleme sa više brojeva i mnogo različitih operacija. Na primer, kao rezultat narebe

```
:PRINT(7-4)*(19-14)/(8-3)(RETURN)
```

računar će prikazati rezultat

```
3
```

Zagrade "()" koristimo da bismo računaru saopštili koja operacija treba prvo da se obavi.

Kod izvršenja aritmetičkih operacija sa decimalnim brojevima, bitno je znati da decimalni zarez u matematici, u računaru zamenjuje decimalna tačka.

Unesimo u računar sledeću naredbu:

```
:PRINT 2.3+4.6
```

Posle pritiska na dirku RETURN računar će prikazati rezultat

```
6.9
```

Ako nam se desi da umesto decimalne tačke unesemo decimalni zarez, naredba će biti

```
:PRINT 2,3+4,6 (RETURN)
```

Računar će na ekranu prikazati sledeće:

```
2 7 6
```

Šta se u stvari desilo? Računar ne prepoznaje decimalni zarez, i zbog toga neće prikazati očekivan rezultat. Znak zarez (,) u naredbi PRINT ima ulogu da razdvoji prikazane podatke za 8 pozicija. Naredbu

```
:PRINT 2,3+4,6
```

računar će protumačiti na sledeći način:

— štampati vrednost 2
— posle 8 praznih mesta štampati rezultat aritmetičke operacije 3+4
— posle 8 praznih mesta štampati vrednost 6

U nekim slučajevima računar neće na ekranu prikazati nikakav rezultat, samo će prijaviti grešku.

2.2.3. Programsko izvršenje naredbe

Želimo li naredbu

```
PRINT 5+6
```

izvršiti više puta bez ponovnog unošenja sa tastature, moramo je sačuvati u memoriji. Programskom redu pridodaćemo **broj reda** sa kojim će naredba biti zapamćena:

```
:10 PRINT 5+6
```

Ovako napisana naredba predstavlja **program**, a za naredbu PRINT kažemo da je **programski izvršena naredba.** U ovom slučaju pritiskom na dirku RETURN označavamo da smo završili unošenje programskog reda i da je program smešten u memoriju.

Da bi računar izvršio upisan program moramo upotrebiti naredbu za izvršenje programa (RUN), koju aktiviramo u režimu direktnog izvršenja:

```
:RUN (RETURN)
```

Posle pritiska na dirku RETURN računar na ekranu prikazuje rešenje programa "11". Ovaj program možemo startovati više puta, sve dok se nalazi u memoriji.

U ovom primeru za broj programskog reda uzet je broj 10. U opštem slučaju to može biti bilo koji pozitivan ceo broj od 1 do 65534. Upotreba broja većeg od 65534 dovešće do uništenja programa. Preporučujemo vam da započnete svoj program sa brojem programskog reda 10 i dalje sa 20, 30, 40,..., ostavljajući na taj način prostore za još 9 programskih linija.

Ovakav način biranja vrednosti za brojeve programskih redova dolazi do izražaja u toku pisanja složenijih programa, kada je potrebno između postojećih programskih redova ubaciti nove, sa novim naredbama, a da pri tom ne prepravljamo već upisane programske redove.

Računar izvršava program po rastućim vrednostima brojeva programskih redova.

2.3. RAD SA PROGRAMIMA

2.3.1. Unošenje novog programa

Programi koje pišemo u jednom programskom redu, veoma su jednostavni tako da pomoću njih ne možemo rešiti nijedan imalo ozbiljniji zadatak. Moramo se pripremiti za pisanje dužih programa, sa više programskih redova. Da nam to ne bi zadavalo velike probleme podsetimo se osnovnih pravila:

- svaki programski red započinje brojem programskog reda
- pritiskom na dirku RETURN završava se unošenje jednog programskog reda, a kursor se postavlja u krajnji levi položaj narednog reda
- broj novog programskog reda mora biti veći od broja prethodnog programskog reda.

Preko tastature unesimo sledeći program u računar:

```
10 PRINT 5*6 (RETURN)
20 PRINT 8+2 (RETURN)
30 PRINT 3-2 (RETURN)
```

Izvršenje programa startujemo naredbom

```
:RUN (RETURN)
```

Na ekranu se štampa rešenje programa:

```
30
10
1
```

Prethodno napisan program može se, umesto u 3 programska reda, napisati u jednom na sledeći način:

```
10 PRINT 5*6:PRINT 8+2:PRINT 3-2
```

gde dve tačke (:) služe za razdvajanje naredbi u jednoj liniji (separator naredbi).

2.3.2. Ispravke programa

Prilikom unošenja programskih redova nekog dužeg programa mogu se pojaviti greške. Ukoliko se greška ranije primeti, lakši je postupak za njeno otklanjanje. Postoji nekoliko postupaka kojima se program može korigovati.

Poništavanje zadnjeg unetog znaka

Poništavanje zadnjeg unetog znaka vrši se istovremenim pritiskom na dirke CTRL i H. Ako želimo izbrisati n unetih znakova treba n puta pritisnuti dirku H dok držimo pritisnutu dirku CTRL.

Poništavanje zadnjeg programskog reda

Ukoliko smo počeli sa unošenjem ovog programskog reda, i ako želimo da računar taj red ignoriše, upotrebićemo posebnu komandu sa tastature istovremenim pritiskom na dirke CTRL i C. Efekat ove naredbe je sledeći: kursor se pomera u novi red u krajnji levi položaj, a programski red koji smo počeli da unosimo postaće nevažeći.

Važno je napomenuti da ova naredba ima efekta samo u slučaju da unošenje programskog reda, koji želimo da izbacimo, nije završeno pritiskom na dirku RETURN. Ukoliko je dirka RETURN pritisnuta mogu se pojaviti sledeće situacije:

— ako smo u poslednjem programskom redu pogrešno upotrebili neku naredbu ili komandu, tada će računar na ekranu štampati poruku o greški a programski red biće ignorisan
— ukoliko su naredbe u poslednjem programskom redu tačno upotrebljene, računar će ceo programski red upisati u memoriju. Tada željeni programski red mora,mo izbaciti na drugi način.

Izbacivanje programskog reda iz programa

Izbacivanje programskog reda iz programa može se vršiti upisivanjem broja programskog reda, koji želimo izbaciti, i pritiskom na dirku RETURN. Pri tome se povećava raspoloživa memorija računara za nove programske redove. Neka se u memoriji računara nalazi sledeći program:

```
10 PRINT 1+2
20 PRINT 3+4
30 PRINT 5+6
```

Ako želimo izbaciti programski red sa brojem 30 potrebno je upisati

```
:30 (RETURN)
```

U memoriji računara nalaziće se sledeći program:

```
10 PRINT 1+2
20 PRINT 3+4
```

Programski red 30 je izbrisan.

Ispravke unutar programskog reda

BASIC poseduje editorski režim rada u kome korisnik može vršiti ispavke unutar unetog programskog reda. U ovaj režim ulazi se posebnom naredbom

```
EDIT < brojni izraz > (RETURN)
```

Oznaka (RETURN) znači da EDIT pripada grupi naredbi sa direktnim izvršenjem.

<brojni izraz> ima vrednost broja programskog reda u okviru kog se vrši ispravka.

Ova komanda otvara broj reda koji je određen brojnim izrazom i dopušta korisniku da menja red. Kod upisa, sa EDIT <brojni izraz> dolazi se na određeni red a kursor se vraća na početak sledećeg reda. Korisnik tada pomera kursor pomoću tastera za razmak jedan znak ispred znaka koji treba da se menja. U ovom momentu na raspolaganju stoje tri EDIT mogućnosti. Korisnik može da unese sa tastature I (ubacivanje), D (brisanje) ili C (promena). Pošto se unese znak za određenu aktivnost, kursor se pomera ispod znaka koji treba da se modifikuje, i dopušta da se unesu znaci ispred svakog položaja znaka u redu. Operacija D uklanja znake iz reda sekvencijalno sa svakim pritiskom dirke za razmak. Operacija C pojedinačno zamenjuje svaki znak novim koji se unosi sa tastature.

Pošto korisnik obavi željenu modifikaciju istovremenim pritiskom na dirke CTRL i S na tastaturi, red se ponovo pojavi na ekranu sa izvršenom modifikacijom. Korisnik može da menja položaj kursora i da ponovo modifikuje red ili da izađe iz režima EDIT sa CTRL i S. Obratite pažnju na to da je kod jednog prolaza dozvoljena samo jedna operacija (I, D ili C). Međutim, korisnik može da primeni svaku operaciju koliko puta želi na određenom redu pre nego što izađe iz režima EDIT. Pritiskom na dirke CTRL i A obavlja se funkcija uklanjanja. Ako se ovo uradi posle znaka za mo-

difikaciju ali pre pritiska na dirke CTRL i S, ponovo se prikazuje red u neizmenjenom obliku. Ovo je značajno kada se pogreši prilikom obavljene operacije I,D ili C.

Gore iznetu proceduru ilustrovaćemo primerima. Taster za razmak označen je oznakom "u". Pretpostavimo da u računaru postoji neki program i u okviru njega u liniji 10 naredba

```
10 PRINT "ZDRAVO"
```

Želimo ispred teksta ZDRAVO da ubacimo tekst KAŽI. Naredbom

```
:EDIT 10 (RETURN)
```

postavljamo liniju 10 u stanje za izmenu. Dobićemo

```
10 PRINT"ZDRAVO"
```

Postavićemo kursor ispod znaka za navod, izabraćemo operaciju I i unećemo tekst KAŽI.

```
10 PRINT "ZDRAVO"
IKAŽIu
```

Pritiskom na dirke CTRL i S prikazuje se izmenjen red:

```
10 PRINT "KAŽI ZDRAVO"
```

Sada želimo da izbrišemo tekst KAŽI i umesto ZDRAVO stavimo MILAN.

Postavimo kursor ispod prvog znaka za navod i ukucajmo operaciju D i 5 znakova za razmak:

```
10 PRINT"KAŽI ZDRAVO"
Duuuuu
```

Svaki znak za razmak briše po jedan znak. Istovremenim pritiskom na dirke CRTL i S dobijamo:

```
10PRINT"ZDRAVO"
```

Pošto izmena još nije završena, ne izlazimo iz režima EDIT u ovom momentu. Pozicioniramo kursor ispod znaka za navod i biramo operaciju 6 za izmenu:

```
10PRINT"ZDRAVO"
CMILANu
```

Istovremenim pritiskom na dirke CTRL i S računar će na ekranu prikazati izmenjen programski red:

```
10PRINT"MILAN"
```

Pošto smo izvršili željene izmene, izlazimo iz režima EDITOR-a ponovnim pritiskom na dirke CTRL i S, i vraćamo se u normalni režim rada porukom

```
READY
::
```

Ponovno unošenje programskog reda

Ispravka programskog reda, korišćenjem naredbe EDIT, efikasna je u slučaju dužeg programskog reda. Kada je programski red kraći bolje je izvršiti izmenu ponovnim unošenjem celog programskog reda. Računar će automatski ignorisati raniji programski red sa istim brojem programskog reda a tretiraće onaj koji smo naknadno uneli.

Ilustrovaćemo ovaj postupak konkretnim primerom. Neka se u računaru nalazi sledeći program:

```
10 PRINT 2*3
20 PRINT 2+3
30 PRINT 5-4
```

Želimo da u programskom redu sa brojem 30 umesto postojeće naredbe stoji naredba PRINT 2—3. Unećemo u računar sledeći programski red

```
30 PRINT 2-3
```

Program će tada biti sledeći

```
10 PRINT 2*3
20 PRINT 2+3
30 PRINT 2-3
```

Na ovaj način smo izvršili izmenu jednog dela programa.

2.3.3. Izvršavanje programa

Startovanje programa, koji smo uneli u računar, vršimo posebnom komandom RUN sa kojom smo se već susreli. Sada ćemo je detaljnije objasniti. Format ove komande može biti

```
RUN (RETURN)
RUN <brojni izraz> (RETURN)
RUN+ (RETURN)
```

RUN

Ova komanda postavlja BASIC u režim izvršenja programa. BASIC počinje traženje najmanjeg broja programskih redova i počinje izvršenje svakog reda po numeričkom redosledu. Međutim, pre nego što počne izvršenje, komanda RUN briše svu oblast polja i niza da bi načinila prostor za nove podatke.

RUN <brojni izraz>

Ova komanda započinje izvršenje programa sa programskim redom čiji je broj određen brojnim izrazom. Ona je slična komandi RUN koju ne prati brojni izraz u tome što postavlja BASIC u režim izvršenja. Ako broj reda, koji određuje brojni izraz, ne postoji računar na ekranu prikazuje grešku. Zbog toga što ova komanda RUN ne briše oblast podataka, korisnik može da izvrši program sa prethodno definisanim podacima.

Unesimo u računa sledeći program

```
10 PRINT 2*3
20 PRINT 2+3
30 PRINT 2-3
```

ako izvršenje programa startujemo komandom

```
:RUN (RETURN)
```

računar će, kao rezultat izvršenja programa, na ekranu štampati

```
6
5
-1
READY
:
```

Na ovaj način je izvršen ceo program.

Želimo li izvršenje samo jednog dela programa upotrebićemo komandu RUN <brojni izraz>. Ako želimo izvršenje programa od programskog reda 20, program startujemo komandom

```
:RUN 20 (RETURN)
```

Kao rezultat izvršenja ovog dela programa, računar na ekranu prikazuje sledeće

```
5
-1
READY
:
```

Pokušamo li da startujemo program komandom

```
:RUN 15 (RETURN)
```

računar na ekranu prijavljuje grešku o nepostojećem redu programa.

RUN+ (RETURN)

Ova komanda ima efekta kod izvršenja programa koji u sebi sadrže naredbe bezuslovnog skoka (GO TO). Naredbe bezuslovnog skoka dosta usporavaju izvršenje programa zbog toga što računar mora adresu na koju se program grana prvo da prevede u apsolutnu a tek onda da izvrši grananje na nju. Kada se startuje naredba RUN+, računar pretražuje korisnički program i sve adrese grananja prevodi u apsolutne adrese, a zatim pristupa izvršenju. Ovaj korak u velikoj meri povećava brzinu izvršenja programa. Posle početnog ciklusa RUN+, RUN će omogućiti izvršenje u ovom bržem režimu. Program koji se poziva sa RUN+ ne treba da se čuva u ovom obliku zbog dodeljivanja apsolutnih adresa.

2.3.4. Čuvanje programa na kaseti

Za trajno cuvanje programa, koje smo sami napravili, moramo koristiti neku od spoljašnjih memorija, jer nakon isključenja briše se sadržaj programa u memoriji računara. Najrasprostranjenija spoljašnja memorija je magnetna traka u kaseti radio-kasetofona. Prilikom upoznavanja sa arhitekturom PECOM-a 64 naučili smo kako se podaci iz memorije računara smeštaju na magnetnu traku.

U ovom delu upoznaćemo se sa naredbama BASIC-a, koje nam omogućavaju smeštanje programa na magnetnu traku. Ovde treba razlikovati smeštanje kompletnog programa i smeštanje podataka.

Za smeštanje celog programa iz memorije računara na magnetnu traku koristimo naredbu PSAVE.

PSAVE

Kada startujemo ovu naredbu sadržaj dela memorije, koji je dostupan korisniku za smeštanje programa, smešta se na traku. Programu možemo dodeliti naziv programa, kojim ga možemo prepoznati. Naredbe PSAVE tada ima format

PSAVE <naziv programa>

gde <naziv programa> može biti proizvoljan skup alfanumeričkih znakova.

Postoje slučajevi kada nemamo potrebu za smeštanjem kompletnog programa, već samo podataka koje generiše program (nizovi, polja). U delu memorije, dostupne korisniku, vrši se smeštanje programa a odmah iza toga smeštanje podataka. Naredbom

DSAVE

sadržaj memorije upamćen na traci počinje pri kraju programa (početak podataka) i završava se pri kraju podataka. Ova naredba omogućava pamćenje svih podataka koje generiše program (nizovi i polja). Takođe, i ova naredba može da se upotrebi u formatu

```
DSAVE <naziv programa>
```

gde je <naziv programa> skup alfanumeričkih znakova koji čine ime programa.

Postupak smeštanja na traku isti je i za programe i za podatke, i biće izložen po fazama:

1. proveravamo da li je program ispravan i da ga takvog možemo smestiti na traku
2. kasetofonskim kablom uspostavljamo komunikaciju računara sa kasetofonom
3. praznu kasetu smeštamo u kasetofon
4. preko tastature unosimo naredbu PSAVE ili PSAVE <naziv programa> ili DSAVE ili DSAVE <naziv programa>
5. startujemo kasetofon za usnimavanje na traku
6. pritiskom na dirku RETURN startujemo naredbu za usnimavanje programa iz računara na traku.

Kao kontrola uspešnog usnimavanja programa na traku služe nam horizontalne pruge na ekranu u plavoj i žutoj boji i tonski signal promenljive jačine. Po završetku usnimavanja ekran se ponovo boji u belo, a računar prikazuje poruku

```
READY
:
```

Ako želimo prekinuti postupak usnimavanja programa na traku, to možemo postići pritiskom na dirku BREAK.

2.3.5 Čitanje programa sa trake

Program koji smo smestili na traku možemo u svakom trenutku ponovo uneti u računar, korišćenjem naredbe

PLOAD

Ovom naredbom se program sa trake smešta u memorijski prostor računara koji je dostupan korisniku. Ukoliko smo pre unošenja programa sa kasete u računaru imali neki program, naredba PLOAD ga kompletno briše, kao i podatke koje je taj program generisao.

Ako ne znamo tačan početak programa na kaseti, možemo ga uneti u računar modifikovanom naredbom PLOAD koja ima format

```
PLOAD <naziv programa>
```

Računar će sam pronaći početak programa pod tim nazivom i početi da ga smešta u memoriju. Sve ostale programe, na koje naiđe u postupku pretraživanja, računar ignoriše. Ovaj postupak je izvodljiv samo u slučaju da smo taj isti program na traku

smestili korišćenjem naredbe PSAVE <naziv programa>. Takođe, moramo da znamo tačan naziv programa.

DLOAD

Ovom naredbom smeštamo prethodno generisane podatke sa trake u računar, i to na kraju bilo kog programa u BASIC-u, koji se u tom trenutku nalazi u računaru. Naredba DLOAD, u delu memorije dostupne korisniku, ne briše program već samo podatke koje je postojeći program u računaru već generisao.

Ako su podaci na traku smešteni naredbom DSAVE <naziv programa>, i ako znamo tačan naziv programa, onda naredbu DLOAD možemo upotrebiti u formatu

DLOAD <naziv programa>

U tom slučaju ne moramo da znamo tačan početak programa na traci, već će to sam računar pronaći postupkom pretraživanja.

Postupak unošenja programa ili podataka sa kasete u računar biće izložen po fazama:

1. Kasetu smeštamo u kasetofon. Ukoliko znamo gde se nalazi početak programa, kasetu premotavamo na tu poziciju, a ako ne znamo tačan početak programa kasetu premotavamo na početak.
2. Kasetofonskim kablom uspostavljamo komunikaciju računara sa kasetofonom
3. Ukoliko smo kasetu premotali na tačan početak programa sa tastature unosimo naredbu PLOAD i DLOAD. U slučaju da smo kasetu premotali na početak (kada ne znamo tačan početak programa), unosimo naredbu PLOAD n<naziv programa> ili DLOAD "<naziv programa>".
4. Startujemo kasetofon za reprodukciju sa trake.
5. Pritiskom na dirku RETURN startujemo naredbu za usnimavanje programa sa trake u računar.

Kao kontrola uspešnog usnimavanja programa u računar služe nam horizontalne pruge na ekranu u beloj i zelenoj boji i tonski signal promenljive jačine. Po završetku usnimavanja ekran se oboji u belo, a računar prikazuje poruku

```
READY
:
```

na osnovu koje zaključujemo da je program korektno unesen u računar.

Postupak unošenja programa sa kasete u računar moguće je prekinuti jedino isključivanjem i ponovnim uključivanjem računara (resetovanjem).

2.3.6. Prikazivanje (listanje) programa

Posle izvršenih ispravki često je potrebno videti izgled kompletnog programa na ekranu, ili ga prikazati na papiru štampača. Umesto termina "prikazivanje programa" češće se koristi termin "**listanje programa**". Program se može izlistati na ekranu ili na papiru štampača.

Listanje programa na ekranu

Za listanje kompletnog programa na ekranu koristi se komanda

LIST

Ukoliko želimo listanje samo jednog programskog reda tada koristimo komandu LIST u formatu

LIST <brojni izraz>

gde je <brojni izraz> pozitivan ceo broj, koji predstavlja broj onog programskog reda koji želimo da prikažemo na ekranu.

LIST <brojni izraz 1>, <brojni izraz2>

Ovom komandom lista se deo programa.

<brojni izraz 1> je pozitivan ceo broj, koji predstavlja broj onog programskog reda sa kojim se počinje listanje

<brojni izraz 2> je pozitivan ceo broj, koji predstavlja broj onog programskog reda sa kojim se završava listanje.

Može se zaključiti da <brojni izraz 1> mora biti veći od <brojni izraz 2>.

Ako je u bilo kom momentu neki brojni izraz jednak brojnom redu koji ne postoji, tada će se listati programski red čiji je broj najbliži datom.

Sledećim primerom demonstriraćemo sve tri varijante komande LIST.

Preko tastature unesimo sledeći program u računar:

```
10 PRINT 1+2 (RETURN)
20 PRINT 3+4 (RETURN)
30 PRINT 5+6 (RETURN)
40 PRINT 7+8 (RETURN)
50 PRINT 9+10 (RETURN)
```

Ako koristimo format naredbe LIST, posle pritiska na dirku RETURN na ekranu se pojavljuje izlistan ceo program.

Ako koristimo format naredbe LIST <brojni izraz> onda će biti listan samo onaj programski red čiji broj odgovara vrednosti brojnog izraza. Na primer

```
:LIST 40 (RETURN)
```

na ekranu prikazuje

```
40 PRINT 7+8
```

Ako koristimo format naredbe LIST sa dva brojna izraza, međusobno odvojena zarezom, tada listanje započinje programskim redom čiji broj reda odgovara brojnom izrazu 1, a završava se programskim redom čiji broj reda odgovara brojnom izrazu 2. Na primer

```
:LIST 20,40
```

Posle izvršenja ove naredbe na ekranu se štampaju sledeći programski redovi:

```
20 PRINT 3+4
30 PRINT 5+6
40 PRINT 7+8
```

Listanje programa na štampaču

Želimo li trajno da sačuvamo program na papiru, možemo ga izlistati na štampaču. Pre toga štampač treba povezati sa računarom posebnim kablom za štampač. Sa detaljima povezivanja štampača i računara upoznali smo se prilikom prezentira-

nja organizacije PECOM-a 64. Pre listanja programa štampač mora biti selektovan, odnosno postavljen u stanje za prijem podataka iz računara.

LLIST

Ova komanda se koristi za štampanje na štampaču listinga celog programa, koji je smešten u memoriji računara. Štampanje programa može se prekinuti pritiskom na dirku BREAK.

2.3.7. Brisanje programa i podataka

Brisanje programa i promenljivih, koje programa generiše, može se postići komanodm

NEW

Istovremeno ova komanda briše i sadržaj memorije ekrana. Ekran se briše, a kursor se postavlja u gornjem levom uglu ekrana.

CLD

Kada se ova komanda izvrši brišu se svi podaci koje je program generisao, a koji se nalaze u memorijskom delu dostupnom korisniku, odmah iza programa. Program ostaje u onakvom obliku kakav je bio i pre upotrebe komande CLD.

CLS

Ovom komandom briše se samo memorija ekrana, dok program i podaci u računaru ostaju nepromenjeni. Posle izvršenja komande CLS kursor se smešta u gornji levi ugao na ekranu.

2.3.8. Promena brojeva programskih redova

Posle više ispravki u programu i dopisivanja novih programskih redova, može se javiti potreba za promenom brojeva programskih redova. Razlozi za ovakvu aktivnost mogu biti:

— između dva susedna programska reda ne može se ubaciti novi programski red
— razlog preglednosti programa.

U ovim slučajevima treba primeniti komandu

RENUMBER

Ova komanda brojevima programskih redova dodeljuje vrednosti 10,20,30... itd. Ako se koristi format komande

RENUMBER <brojni izraz>

tada <brojni izraz> određuje početni broj programskog reda, a istovremeno i razmak između redova.

Kada se komanda RENUMBER izvrši, računar šalje poruku

```
0 COMP BR
```

kojom nam daje do znanja da je izvršen postupak promene brojeva programskih redova. Ako dođe do pogrešnog uređivanja programa računar šalje poruku o greški.

Razmotrićemo primer programa pre i posle sprovođenja komande RENUMBER. Neka je izvorni program sledeći:

```
1 CLS (RETURN)
2 PRINT 7*6 (RETURN)
9 PRINT 5+3 (RETURN)
```

Ako primenimo komandu

```
:RENUMBER(RETURN)
```

računar šalje poruku

```
0 COMP BR
READY
:
```

Posle listanja programa komandom

```
LIST (RETURN)
```

na ekranu dobijamo preuređen program:

```
10 CLS
20 PRINT 7*6
30 PRINT 5+3
```

Primenimo li komandu

```
RENUMBER 3 (RETURN)
```

i pozovemo li listanje programa na ekranu komandom LIST, dobićemo

```
3 CLS
6 PRINT 7*6
9 PRINT 5+3
```

2.3.9. Dužina programa

Ako želimo, možemo proveriti koliki memorijski prostor zauzima program i podaci koje generiše sam program. Takođe možemo proveriti koliko je ostalo slobodnog memorijskog prostora u računaru.

EOP

Kada se izvrši ova komanda, računar na ekranu automatski prikazuje heksadecimalnu adresu memorijske lokacije kojom se završava program upisan u memoriji. Ako znamo da se program u memoriji upisuje počev od memorijske lokacije sa heksadecimalnom adresom 0200, možemo lako izračunati koliki memorijski prostor zauzima program.

EOD

Kada se izvrši ova komanda računar na ekranu prikazuje heksadecimalnu adresu memorijske lokacije kojom se završavaju podaci upisani u memoriji. Ako znamo, da se podaci u memoriji upisuju odmah posle upisivanja programa, i ako znamo heksadecimalnu adresu memorijske lokacije gde se završava upisivanje programa, možemo izračunati koliko je memorijskih lokacija upotrebljeno za upis podataka.

Primenićemo, istovremeno, komande EOP i EOD na program:

```
10 CLS (RETURN)
20 PRINT 7*6 (RETURN)
30 PRINT 5+3 (RETURN)
:EOP:EOD(RETURN)
```

Računar na ekranu prikazuje poruku:

```
READY
© 0236 © 023F
```

Na osnovu ovih podataka zaključujemo da je program u memoriji zauzeo 54 memorijskih lokacija. Kako smo to izračunali?

Program zauzima memorijski prostor od memorijske lokacije sa heksadecimalnom adresom 0200 do 0236, što je ukupno 36 heksadecimalnih brojeva adresa. Heksadecimalni broj 36 je decimalni 54.

Podaci su zauzeli memorijske lokacije sa heksadecimalnim adresama od 0237 do 023F, što čini ukupno 9 memorijskih lokacija.

MEM

Ova komanda omogućava korisniku da sazna koliko mu je preostalo slobodnog memorijskog prostora u RAM memoriji. Kada se izvrši, MEM daje decimalni broj koji predstavlja broj preostalih slobodnih memorijskih lokacija. U slučaju prethodnog programa, kada unesemo komandu MEM:

```
:PRINT MEM (RETURN)
```

računar odgovara sledećom porukom na ekranu

```
31073
READY
:
```

što znači da je ostalo još 31073 slobodnih memorijskih lokacija u RAM memoriji.

Bitno je napomenuti da se komande EOP, EOD i MEM mogu upotrebiti u bilo kom delu programa, jer su to naredbe sa direktnim izvršenjem.

2.3.10. Pomeranje početka programa

DEFUS & <brojni izraz>

Naredbom DEFUS definiše se start korisničke oblasti, u kojoj može da se smešta program napisan na BASIC-u. Ona mora biti upotrebljena pre početka pisanja programa i pripada naredbi sa direktnim pristupom. Ova naredba dopušta da se početak programske oblasti pomeri dalje u memoriju. Ona omogućava programeru da kreira "rupu" u memoriji u kojoj može da generiše programe na mašinskom jeziku. Brojni izraz definiše gde programska oblast treba da počne. Kod BASIC-a korisnička oblast počinje sa heksadecimalnom adresom 0200. Brojni izraz mora da počne od broja koji je veći od 0200.

Ako se načini pokušaj da se definiše korisnička oblast na nekoj adresi ispod 0200, BASIC će sam sebe uništiti. Kada se naredba DEFUS izvrši jedini način da se programska oblast vrati na 0200 jeste pomoću druge DEFUS naredbe na 0200. Linija uništava korisnički program koji je u memoriji.

Interesantna karakteristika pomeranja početka oblasti korisničkog programa leži u naredbi PSAVE. Ako je neki program generisan u pomerenoj lokaciji i sa njom je

doveo u vezu neke programe na mašinskom jeziku, naredba PSAVE će upamtiti sve od 0200 do kraja programske oblasti. Programi na mašinskom jeziku biće uključeni kao i dati programi u BASIC-u. Naredba PLOAD će učitati sve gornje programe uključujući programe na mašinskom jeziku i ponovo će definisati početak korisničke oblasti.

U slučaju da nije upotrebljena naredba DEFUS, tada će se program na mašinskom jeziku u memoriji naći iza programa BASIC-a. Na taj način program na mašinskom jeziku neće biti učitan na traku.

Primer za DEFUS naredbu dat je u daljem tekstu.

```
DEFUS & 2D00
DEFUS & 2DCC     pomera početak korisničke oblasti na 2DCC
DEFUS & 0200     pomera početak programske oblasti u njegov početni položaj.
```

Kada se koristi rupa koju je načinila naredba DEFUS, treba obratiti pažnju na to da se memorijski prostor od 0200 do 0210 koristi za smeštanje konstanti programa pisanog u BASIC-u i treba da se izbegava kod pisanja programa na mašinskom jeziku.

2.4. KOMANDE SA TASTATURE PECOM-a 64

2.4.1. Komande koje formira funkcionalna dirka CTRL

Ako se izvesne dirke pritisnu dok je pritisnuta dirka CTRL računar će preduzeti izvesne akcije.

CTRL-C

Ako se pritisne dirka C dok je pritisnuta dirka CTRL (operacija čija je skraćenica "CTRL-C"), pre pritiska na dirku RETURN, red koji se prikazuje biće ignorisan i kursor će se postaviti u krajnji levi položaj u sledećem redu. CTRL-C omogućava korisniku da dâ računaru informaciju o tome da želi da ignoriše delimično otkucanu naredbu.

CTRL-H

Pritiskom na dirku H dok se drži dirka CTRL, kursor se pomera za jedan znak unazad, a znak preko koga se prelazi automatski se briše. Na ovaj način je moguća korekcija pogrešno unetih naredbi i podataka.

CTRL-A

Ukoliko se računar nalazi u fazi izvršenja EDIT naredbe u nekoj od tri mogućnosti, pritiskom na dirku A dok se drži dirka CTRL, ignoriše se izvršena aktivnost i može se izaći iz EDIT naredbe ili ponovo otpočeti neka od C, D ili I mogućnosti.

CTRL-D

Pritiskom na dirku D za vreme dok je pritisnuta i dirka CTRL briše se ekran.

CTRL-S

U toku izvršenja EDIT funkcije, aktiviranjem CTRL-S pojavljuje se nova varijanta reda sa ispravkama, a ponovnim startovanjem CTRL-S završava se editovanje i upravljanje se vraća na READY.

2.4.2. Funkcija dirke BREAK

Pomoću dirke BREAK može se izvršavati prevremeno prekidanje sledećih komandi BASIC-a:

— komande izvršenja programa: RUN, RUN <brojni izraz>
— komande listanja programa na ekranu: LIST
— komande listanja programa na štampaču: LLIST.

2.5. IZVEŠTAJ O GREŠKAMA

U toku pisanja programa postoji mogućnost da se pogreši, greške na prvi pogled nisu vidljive, ali doprinose da se program ne izvršava. Računar ispituje svaku naredbu i svaki programski red. Ako otkrije grešku u nekom programskom redu, izdaje sledeću poruku:

```
ERR CODE <broj greške>
AT LINE <broj reda>
```

BASIC u sebi sadrži spisak svih grešaka, koje mogu da se pojave u toku realizacije programa. Svaka greška imaće svoj broj od 0 do 72, na osnovu kog možemo saznati kakve je prirode.
<broj reda> predstavlja broj programskog reda u kome je pogrešeno.

Ako je greška otkrivena u programskom redu bez broja reda ili u komandi, tada poruka o greški sadrži samo broj greške i imaće format

ERR CODE <broj greške>

Da bi program funkcionisao i da bi dao očekivane rezultate, grešku u programu treba otkloniti na jedan od ranije opisanih načina.

Upoznaćemo se sa još jednom situacijom u kojoj se možemo naći prilikom izvršavanja programa. Već smo se upoznali sa funkcijom dirke BREAK. njome prekidamo izvršenje tekućeg programa. Posle pritiska na dirku BREAK, na ekranu će se pojaviti poruka:

```
ERR CODE 0
```

AT LINE <broj reda>

U ovom slučaju ne radi se ni o kakvoj greški. Računar samo daje poruku da smo prekinuli izvršenje programa kod naredbe koja se nalazi u programskom redu sa naznačenim brojem reda.

2.6. FUNKCIJA, PODATAK, PROMENLJIVA, IZRAZ

2.6.1. Funkcije

Funkcije su ključne reči koje nakon izvršenja daju neku vrednost. Mogu da stoje samostalno ili da se koriste sa drugim funkcijama, da obrazuju izraze. Lista funkcija, koje koristi BASIC, data je u tabeli 6.1.

ABS	INT	PI
ASC	INUM	RND
ATN	LEN	SGN
COS	LOG	SIN
CHR $	MEM	SQR
EXP	MID $	TAB
FNUM	MOD	USR
FVAL	PEEK	KEY

Tabela 6.1.

Postoje izvesna ograničenja u pogledu korišćenja pojedinih funkcija. Na primer, funkcija TAB može da se koristi u naredbi PRINT. Na druga ograničenja biće ukazano prilikom detaljnog opisivanja pojedinih funkcija.

2.6.2. Podatak

Podatak je oznaka informacije koja se obrađuje na računaru. Zavisno od toga da li je oznaka broj ili slovo, postoje brojni i slovni podaci. Brojni podaci mogu biti celobrojni ili mešoviti.

2.6.3. Promenljiva

Svaki programski jezik gradi se nad skupom osnovnih simbola koje nazivamo azbuka. Od njih se grade elementarne i složene konstrukcije programskog jezika. Jedna od najelementarnijih konstrukcija koju možemo ugraditi u programe jeste **promenljiva.**

Kad od računara zatražimo da sačuva neke informacije, on mora da zna kako da ih prepozna. Drugim rečima, svakom podatku koji je potrebno ponovo koristiti treba da se dodeli ime po kome će ga računar prepoznavati i pronalaziti kad zatreba.

Jedna od tipičnih informacija koju je potrebno memorisati jeste broj koji se razvijanjem programa uvećava ili smanjuje. Takođe je često potrebno ponoviti neki postupak određen broj puta. Za sve korake koji obrazuju taj postupak kažemo da čine ciklus ili petlju, a moramo im pridružiti brojač ciklusa, čiji je zadatak da kontroliše da li je postupak ponovljen traženi broj puta.

Svaka promenljiva mora imati:

— tip
— tekuću vrednost
— ime.

U BASIC-u koristimo dva osnovna tipa promenljivih:

— brojne promenljive, ako je podatak broj
— slovne promenljive, ako je podatak slovo.

Brojne promenljive mogu biti:

— celobrojne

— realne.

s tim što se vrednosti kod realnih brojnih promenljivih mogu zapisivati u obliku fiksnog ili pokretnog zareza.

Brojnoj promenljivoj se kao tekuća vrednost može dodeliti celobrojni ili razlomljeni podatak, ali se uvek vrši prilagođavanje na tip promenljive.

Ime brojne promenljive može biti slovo

```
A
B
.
.
Z
```

ili slovo sa brojem iza

```
A0
A1
.
.
.
A255
.
.
.
Z0
Z1
.
.
.
Z255
```

Računar, sve ono što se nalazi između znakova navoda, tretira kao samo jedan slovni podatak, ne zalazeći u to iz čega se on sastoji i šta znači. Između navodnika možemo pisati sve znake: slova, cifre, razmake, interpunkcijske znake, grafičke simbole.

Ime slovne promenljive sadrži slovo sa dolarskim znakom ($):

```
A$
.
.
Z$
```

Slovni podaci se ne mogu upotrebljavati ni u jednom brojnom izrazu ili relaciji. Oni ne mogu biti ni argumenti numeričkih funkcija. Računar nad slovnim podacima može vršiti jedino slovne operacije:

— spajanje dva slovna podatka u jedan

— razdvajanje jednog slovnog podatka na delove.

Tip promenljive definiše skup vrednosti koje se mogu dodeliti promenljivoj. U suštini su sve promenljive definisane kao realne (razlomljene), dok se celobrojne promenljive definišu posebnim naredbama BASIC-a.

2.6.4. Izraz

Izraz je vrsta zapisa postupka za dobijanje jednog rezultujućeg podatka na osnovu jednog ili više zadatih podataka. Postoje aritmetički i slovni izrazi.

Aritmetički izraz je formula koja može da se sastoji od:
- samo jednog broja
- jedne promenljive
- jedne funkcije
- od grupe funkcija, povezanih međusobno aritmetičkim operatorima.

Neki primeri aritmetičkih izraza su:

```
5 SIN (45)
A 3*SIN (45)
K(5)6*(2+A)/(B+C)
```

Aritmetički izrazi mogu biti celobrojni i mešoviti. U prethodnim primerima celobrojni izraz je 5 a svi ostali su mešoviti.

Slovni izraz predstavlja zapis postupka za dobijanje rezultujućeg slovnog podatka na osnovu jednog ili više slovnih podataka.

2.7. BROJEVI

BASIC radi sa celim brojevima i brojevima u pokretnom zarezu. Celi brojevi mogu da se koriste u opsegu
od —2147483647 do +2147483647
i oni su tačni u celom opsegu. Brojevi u pokretnom zarezu mogu da se koriste u opsegu
od —0.70141E39 do +0.70141E39

Dekadni broj u pokretnom zarezu može se napisati u:
- pozicionom obliku
- eksponencijalnom obliku.

Pozicioni oblik dekadnog broja u pokretnom zarezu u matematici se piše na sledeći način

```
-15,83
| | └────────razlomljeni deo
| └──────────decimalni zarez
└────────────celi deo
```

a u BASIC-u

```
-15.83
```

Eksponencijalni oblik dekadnog broja u pokretnom zarezu u matematici piše se na sledeći način:

```
-0,1583·10²
   |     | └─ eksponent
   |     └───baza
   └─────────mantisa
```

a u BASIC-u

```
-.1583E2
 └────────── decimalna tačka
```

Za pozicioni zapis dekadnog broja u pokretnom zarezu, u BASIC-u važe sledeća pravila:

— pozicioni zapis može imati maksimalno 10 cifara
— najmanji broj je —.170141E39
— najveći broj je +.170141E39
— umesto decimalnog zareza u BASIC-u se koristi decimalna tačka
— nula kao celi ili razlomljeni deo broja može se izostaviti
— znak + može se izostaviti.

Primer

```
110.0
110.        nula kao razlomljeni deo broja može se izostaviti

0.124
 124        nula kao celi deo broja može se izostaviti
.

1234567890
1234567890.1    broj je neispravno napisan jer ima više od 10 cifara
```

Ako je uneseno više od 10 cifara, BASIC javlja grešku

```
ERR CODE 44
```

Za eksponencijalni zapis dekadnog broja u pokretnom zarezu u BASIC-u važe sledeća pravila:

— mantisa može biti bilo koji ceo ili razlomljeni dekadni broj poziciono zapisan
— mantisa može imati najviše 10 cifara
— opseg eksponenta je od —39 do +39
— znak + može se izostaviti i u mantisi i u eksponentu.

Primer: Brojeve u levoj koloni napisati u eksponencijalnoj notaciji.

54378.03	5.437803E-5
$3.14.1\sigma^5$	3.14E5
10^{-18}	1,E-18
-10^{15}	-1.E15

Primer: U sledećim eksponencijalnim zapisima brojeva postoje greške.

E02	nema mantise
12345678900E+5	suviše cifara u mantisi
15.5E-40	eksponent je manji od -39
-5.40E53	eksponent veći od +39

Za predstavljanje **heksadecimalnih brojeva** koriste se heksadecimalne konstante:

& — definiše heksadecimalni 16-bitni broj u formatu

```
& hhhh
```

#— definiše heksadecimalni 8-bitni broj u formatu

```
#hh
```

Za predstavljanje 8-bitnih binarnih brojeva koriste se binarne konstante (%), u formatu

```
%bbbbbbbb%
```

Tačnost brojeva u pokretnom zarezu je približno na 6 cifara, mada je dozvoljen ulaz devet cifara.

Svi brojevi koji se unose sa tastature pamte se onako kako se upisuju. Ako se unesu bez decimalnog zareza, pamte se kao celi brojevi, a ako se unesu sa decimalnim zarezom ili oznakom "E", pamte se kao brojevi sa pokretnim zarezom. Ako se oba tipa podataka jave u istom izrazu, tj. u mešovitom izrazu, prvi termin izraza određuje tip preostalog dela izraza. Ako izraz počinje brojem, pretpostavlja se da je izraz sa pokretnim zarezom, bez obzira da li je izraz unet kao ceo broj ili sa decimalnim zarezom.

2.7.1. Definisanje celoborojnih promenljivih

Ako u toku pisanja programa koristimo neke promenljive, koje treba da dobijaju celobrojne vrednosti, postoji mogućnost da te promenljivę definišemo kao celoborojne. To činimo na početku programa, naredbom

DEFINT <slovni izraz>

Kada upotrebimo ovu naredbu, tada računar sve promenljive od A do slovnog izraza (uključujući slovni izraz) postavlja kao celobrojne. Na primer, naredba

```
DEFINT D
```

definiše imena promenljivih A, B, C i D kao celobrojne.

Kada se upotrebi naredba DEFINT bez slovnog izraza, u formatu

```
DEFINT
```

tada računar sve promenljive, označene slovima od A do Z, definiše kao promenljive sa pokretnim zarezom.

Razmotrimo sledeći primer:

```
10 DEFINT A
20 A=23.2
30 PRINT A
```

U redu broj 10 promenljiva A postavlja tip celog broja. Broj 23.2 je broj sa pokretnim zarezom. Stoga se pretvara u ceo broj i dodeljuje se promenljivoj A. Važno je obratiti pažnju na to da kada se broj sa pokretnim zarezom ili funkcija pretvaraju u ceo broj, zaokruživanje se vrši na najbliži ceo broj. Rezultat reda broj 30 biće štampanje celo broja 23 bez decimalnog zareza. Razmotrimo sledeći primer:

```
10 DEFINT B
20 B=3.45+2.23
30 PRINT B
```

Promenljivoj B dodelili smo celobrojnu vrednost, i zbog toga se vrednost aritmetičkog izraza zaokružuje na najbliži ceo broj. Rezultat programskog reda 30 biće štampanje celog broja 6 bez decimalnog zareza.

Naredba NEW će uvek pretvoriti sve promenljive u oblik sa pokretnim zarezom. Zbog toga, ako u programu treba da se upotrebe celobrojne promenljive, programski red sa naredbom DEFINT treba da se javi na početku programa.

2.7.2. Određivanje broja mesta za prikazivanje brojeva

U toku pisanja programa može se definisati broj mesta za prikazivanje brojeva u pokretnom zarezu. U tu svrhu se koristi komanda

FORMAT N

Ova komanda određuje broj mesta za prikazivanje brojeva. Svaka naredba PRINT koja dolazi posle komande FORMAT, bilo da prikazuje vrednost promenljive ili da prikazuje broj, imaće određen broj mesta za prikazivanje brojeva. Na primer, kada računar izvrši program

```
10 A=12345.6
20 FORMAT 7
30 PRINT A
40 PRINT 67890.1
:RUN (RETURN)
```

štampaće na ekranu

```
12345.6
67890.1
```

Broj N u naredbi FORMAT definiše sa koliko pozicija treba predstavljati brojeve. Kod brojeva sa pokretnim zarezom i decimalna tačka zauzima jednu poziciju. Ukoliko brojem pozicije (koju određuje naredba FORMAT) ne može da se prikaže neki broj, računar će posle naredbe za prikazivanje tog broja, odštampati onoliko zvezdica koliko dozvoljava broj N u naredbi FORMAT.

Ako se u prethodnom programu red 20 zameni sa

```
20 FORMAT 6
```

računar će posle startovanja programa, na ekranu štampati

```
******
******
```

Ako se red 40 zameni sa

```
40 PRINT -67890.1
```

rezultat je

```
******
-*****
```

Da rezimiramo: ako je broj mesta, koji je potreban za prikazivanje broja, neki pozitivan broj koji je veći od N, prikazaće se N zvezdica. Ako je broj negativan, prikazaće se znak — iza kog dolazi (N—1) zvezdica. Opseg za N je od 1 do 15.

FORMAT 0 znači da se ograničenje usled FORMAT-a isključuje. Na primer, program

```
10 FORMAT 5
20 PRINT 123456
30 FORMAT 0
40 PRINT 123456
:RUN (RETURN)
```

da će

```
*****
123456
```

Komandom FORMAT 5 ograničili smo prikazivanje brojeva sa pet pozicija. Na ovakav način nije moguće prikazati broj 123456, na šta računar odgovara štampanjem pet zvezdica. Komandom FORMAT 0 ukida se ograničenje za prikazivanje brojeva.

2.7.3. Formatiranje brojeva sa pokretnim zarezom

FIXED <brojni izraz>

Kada se ova naredba izvrši ona formatira prikazivanje svih brojeva sa pokretnim zarezom. Vrednost brojnog izraza definiše koliko će se cifara prikazati zdesna od decimalnog zareza. Prateće nule se dopisuju da bi se kompletirao broj. Ako je potrebno broj će biti zaokružen.

Ilustrovaćemo ovu funkciju primerom. Preko tastature unesimo sledeći program u računar:

```
10 PRINT 123
20 PRINT 44.5566
30 PRINT 5E7
```

Posle startovanja programa, računar će štampati na ekranu:

```
123
44.5566
.5E+08
```

Ako u postojeći program ubacimo liniju sa naredbom FIXED 3, dobićemo program u obliku:

```
1 FIXED 3
10 PRINT 123
20 PRINT 44.5566
30 PRINT .5E+8
```

Startovanjem programa dobićemo prikazane brojeve u formatu

```
123.000
44.556
.500E+08
```

Zaključujemo da su svi brojevi (ceo, decimalni i eksponencijalni) zaokruženi na 3 decimale.

2.8. OPERATORI

Posmatrajmo neke osnovne operacije koje se mogu izvršiti nad dvema veličinama:

```
C=A+B
C=A-B
C=A*B
C=A/B
```

Veličine A, B, C čine operande, a operacije +,—,*,/,= čine operatore.

Operand je svaka veličina koja ulazi ili izlazi iz aritmetičke ili logičke operacije. **Operatori** su oznake koje ukazuju na vrstu operacije koja treba da se izvrši na operandu da bi se dobio željeni rezultat. Operatori se mogu podeliti u pet kategorija:

— aritmetički
— za dodeljivanje
— relacioni

— logički
— razni.

U daljem tekstu sledi opis operatora u svakoj od ovih kategorija.

2.8.1. Aritmetički operatori

U sledećem spisku dati su aritmetički operatori sa njihovim simbolima, koji prikazuju prioritete svake matematičke operacije.

stepenovanje	↑najviši prioritet
množenje	*prioritet drugog nivoa
deljenje	/prioritet drugog nivoa
sabiranje	+prioritet trećeg nivoa
oduzimanje	—prioritet trećeg nivoa

Na operatore istog nivoa prioriteta deluje se sleva udesno. Simbol "—" se, takođe, koristi za označavanje negativnih brojeva. Redosled izvršavanja operacija može se menjati upotrebom zagrada. U daljem tekstu slede primeri koji ilustruju redosled prioriteta.

u matematici	**u BASIC-u**
a/b*c	A/B*C
a^b+cd	A*B+C*D
a_{bc}	A↑B
a^{bc}	A↑B↑C
a	a↑(B*C)
—ab	—A*B
ab/cd	
c	A*(B/C)*D
$3a_2^2$+2b—5	3*A↑2+2*B—5
$-a^2$	—A↑2

Stepenovanje brojeva se izvodi na sledeći način:

— ako je eksponent ceo broj, stepenovanje se svodi na množenje
— ako je eksponent razlomljen broj, tada se stepenovanje svodi na logaritmovanje i antilogaritmovanje. Posledica toga je zahtev da osnova ne sme biti negativan razlomljeni broj.

Primer:

2↑3	je isto što i	2*2*2	i daje 7.99999
5.2↑2	je isto što i	5.2*5.2	i daje 27.4
—2↑3	je isto što i	(—2)*(—2)*(—2)	i daje —8

(—2)↑1.2 nije dozvoljeno i BASIC javlja grešku ERR CODE 8.

2.8.2. Operatori za dodeljivanje

Simbol za dodeljivanje je znak "—". Kada se ovaj znak upotrebi za dodeljivanje, on dobija definiciju "dobija vrednost za". Primeri:

LET A=5 — promenljivoj A se dodeljuje vrednost 5

LET B=B+A*2 — promenljivoj B se dodeljuje stara vrednost promenljive B uvećane za vrednost A*2

2.8.3. Relacioni operatori

Relacioni operatori su:

jednako	=
nejednako	< >
veće od	>
manje od	<
veće ili jedanko od	> =
manje ili jednako od	< =

Izrazi koji sadrže relacione operatore ocenjuju se pre kao istiniti ili lažni nego kao numerička vrednost.

2.8.4. Logički operatori

Logički operatori su: AND, OR, XOR, NOT. Ovi operatori obavljaju ili logičku operaciju nad datim izrazima ili logički komplement datog izraza. Treba obratiti pažnju na to da logički operatori pretvaraju izraze u celobrojne pre nego što obave naznačenu operaciju i da izraz koji dolazi posle operatora NOT mora da bude u zagradi.

2.8.5. Razni operatori

U grupu raznih operatora spadaju:

: odvajanje linije u istom redu
; ograničavač za naredbu PRINT
, ograničavač za naredbu PRINT
" ograničavač niza
(grupni ograničavač
) grupni ograničavač

Upotreba zagrada u BASIC-u ima sledeća pravila:

— postoje samo male zagrade
— koriste se kao u matematici
— menjaju uređeni prioritet aritmetičkih operacija
— deo izraza koji je u zagradi najvišeg je prioriteta
— unutrašnje zagrade su višeg prioriteta od spoljnih
— u jednom izrazu broj unutrašnjih zagrada mora biti jednak broju spoljašnjih zagrada.

Primer:

u matematici	u BASIC-u
a^{b+c}	A↑(B+C)
a^{-b}	A↑(—B)
a^{bc}	A↑(B*C)
$(a^{b})^{c}$	A↑B↑C
$\frac{ab}{cd}$	(A*B)/(C*D)
$(a+b)^2+(a-b)^2$	(A+B)↑2+(A—B)↑2
$\frac{a}{(b+c)^2}+\frac{4}{d}+3$	A/(B+C)↑2+4/D+3

2.9. NIZOVI I POLJA

U mnogim oblastima primene javlja se potreba da se izvode operacije nad skupom podataka (kao što su vektori, matrice). Da se ne bi pojedinačno imenovali podaci, ovakav skup podataka dobija zajedničko ime: niz ili polje.

Pre nego što se upustimo u detaljniju analizu nizova i polja u BASIC-u definisaćemo ove pojmove.

Niz je redosled znakova (alfanumeričkih i specijalnih znakova kao što su +,—,*,$ i sl.). U niz se može uključiti i znak za blanko ali ne može navodnik (znak"), zbog toga što su navodnici ograničavači niza (upotrebljavaju se da obeleže početak i kraj niza). BASIC može da sadrži najviše 127 znakova u jednom nizu.

Poljem nazivamo skup podataka, pri čemu svi podaci nose isti naziv grupe, ali gde je svaki označen još i brojem (ili skupom brojeva).

U BASIC-u se može definisati maksimalno 26 polja (svako polje označeno sa jednim slovom engleske azbuke). Polja mogu biti jednodimenzionalna i dvodimenzionalna. Jednodimenzionalno polje još nazivamo **niz**, a dvodimenzionalno polje nazivamo — **tabela.**

Element jednodimenzionalnog polja određen je nazivom polja i jednim brojem koji predstavlja redni broj elementa u polju u odnosu na prvi element polja. Primer jednodimenzionalnog polja (niza) je:

```
A(5)=1,2,3,45,66
```

Element dvodimenzionalnog polja određen je nazivom polja i brojnom vrednošću vrste i kolone u kojoj se element u polju nalazi.

Primer dvodimenzionalnog polja je:

$$A(m,n)=\begin{Vmatrix} a_{11}\,a_{12}\ldots a_{1n} \\ a_{21}\,a_{22}\ldots a_{2n} \\ \cdot\quad\cdot\quad\cdot \\ \cdot\quad\cdot\quad\cdot \\ \cdot\quad\cdot\quad\cdot \\ a_{m1}\,a_{m2}\ldots a_{mn} \end{Vmatrix}$$

2.9.1. Dimenzionisanje nizova i polja

Računar u svojoj memoriji sadrži jedan deo memorijskog prostora u koji smešta nizove i polja koji se koriste u programu. On mora tačno da zna koliki deo memorije da odvoji za to. Taj postupak nazivamo — dimenzionisanje nizova i polja.

Ukoliko program, koji sami kreiramo, sadrži nizove i polja tada ih moramo dimenzionisati. U tu svrhu koristimo naredbu DIM.

Format naredbe DIM za dimenzionisanje nizova je:

DIM <naziv niza> <ceo broj>

gde je:

<naziv niza> — naziv jednog niza
<ceo broj> — ceo broj koji definiše koliko je polje veliko.

Format naredbe DIM za dimenzionisanje polja je:

DIM <naziv polja> <ceo broj>, <ceo broj>

gde je:

<naziv polja> — naziv jednog polja
<ceo broj> — ceo broj koji definiše koliko je polje veliko u jednoj od dve dimenzije.

Tip polja (pokretni zarez ili celobrojan) zavisi od toga da li je njegov prateći naziv definisan kao celobrojan ili sa pokretnim zarezom. Na primer, ako su sve promenljive od A do Z sa pokretnim zarezom, tada će sva polja biti sa pokretnim zarezom (nezavisno od režima izraza koji definiše njegove granice). Ako su promenljive A,B i C definisane kao celobrojne, tada će polja A, B i C (ako se upotrebe) biti definisana kao celobrojna. Višestruka polja mogu se dimenzionisati u istoj DIM naredbi sve dok je svako polje odvojeno zarezom.

Ako se oblast memorije prekorači dok se dimenzioniše polje, rezultat će biti poruka o greški. Pokušaj da se upotrebi element nekog polja koje prethodno nije dimenzionisano, imaće za rezultat poruku o greški. Brisanje oblasti za podatke vrši se naredbom CLD.

Primer DIM naredbe je:

```
DIM A(10,10), B(20)
```

Gornji primer ostavlja prostor za 100 brojeva (10x10), pri čemu nazivi idu od A(1,1) do A(10,10) i 20 brojeva sa nazivima od B(1) do B(20).

Naredba za dimenzionisanje može da se upotrebi, ako se želi, u direktnom režimu izvršenja. Svako polje koje program generiše ostaje nedirnuto nakon završetka izvođenja programa. Naredbom PRINT u direktnom režimu može da se sazna sadržaj nekog polja.

Polje ostaje netaknuto sve dok se ne izvrši naredba CLD ili dok se ne obavi neka izmena programa. Podaci polja, kao i podaci niza, nalaze se neposredno iza korisničke oblasti. Svaka promena prvobitnog programa izmeniće korisničku oblast i na taj način uništiti podatke. Treba obratiti pažnju na to da neko polje može ponovo da se dimenzioniše a da se ne unište ostala polja ili nizovi.

2.10. NAREDBE ZA DODELJIVANJE

Naredbe ove grupe dodeljuju vrednosti promenljivama ili memorijskim lokacijama. U ovu grupu spadaju naredbe: LET i POKE.

2.10.1. Naredba LET

Opšti format naredbe LET je:

LET <promenljiva> = <izraz>

Glavno dejstvo ove naredbe je da promenljivoj dodeli vrednost izraza. Promenljiva se uvek nalazi na levoj strani znaka "=", a izraz čija se vrednost dodeljuje promenljivoj, na desnoj strani.

Promenljiva i izraz mogu biti brojnog ili znakovnog tipa, iz čega sledi da postoje dve verzije naredbe LET.

LET <brojna promenljiva> = <brojni izraz>

Ova naredba dodeljuje vrednost brojnog izraza brojnoj promenljivoj. Tip brojne promenljive definiše režim u kome će biti brojni izraz. Reč LET se može izostaviti.

Treba napomenuti da znak"=" ne znači "jednako", već simbolizuje proces koji se sastoji od 3 koraka:

1. izračunavanje vrednosti izraza
2. prevođenje izračunate vrednosti brojnog izraza sa desne strane znaka =, u onaj režim u kom je definisana promenljiva. Ako je vrednost brojnog izraza data kao broj sa pokretnim zarezom, a brojna promenljiva kao celi broj, onda se vrednost brojnog izraza prevodi u ceo broj.
3. dodeljivanje vrednosti brojnog izraza iz koraka 2. brojnoj promenljivoj.

Daćemo neke primere naredbe LET. Preko tastature unesimo u računar sledeći program:

```
10 LET I=3*8
20 LET K=2*SIN(PI/2)
30 LET M=3
40 PRINT I*M/K
```

Kada računar izvrši ovaj program, dobićemo rešenje 36. Na ovaj način smo sigurni da smo sve ove oblike naredbe LET korektno upotrebili.

Ako su promenljiva i izraz znakovnog tipa, onda je format naredbe LET

LET <slovna promenljiva> = <slovni izraz>

U narednom primeru upotrebićemo naredbu LET za slovne promenljive i izraze. Preko tastature unesimo sledeći program u računar:

```
10 LET A$="DOBRO JE UPOTREBLJENA"
20 B$=A$+ "NAREDBA LET"
30 PRINT B$
```

Ako izvršimo ovaj program

```
:RUN (RETURN)
```

dobićemo sledeći tekst ispisan na ekranu:

```
DOBRO JE UPOTREBLJENA NAREDBA LET
```

Svi nizovi znakova u naredbi LET moraju biti pod navodnicima.

Sada ćemo se upoznati sa izvesnim ograničenjima u korišćenju promenljivih kod naredbi dodeljivanja. Preko tastature unesimo sledeći program:

```
10 M=5
20 M=M+1
30 PRINT M
```

Ako startujemo izvršenje ovog jednostavnog programa, računar će na ekranu prikazati grešku. O čemu se radi?

PECOM 64 u skupu naredbi, koje se mogu pozivati iz BASIC-a, poseduje naredbu

```
M+
```

kojom se direktno iz BASIC-a vrši prelazak na rad u režimu MONITOR-a+. Kada BASIC interpretator naiđe na naredbu u programskom redu 20, on je neće protumačiti kao matematičku operaciju, već će je tumačiti kao naredbu za prelaz u režim MONITOR-a+. Međutim, pošto naredba M+ mora ići na početku programskog reda (a u našem slučaju pre toga imamo M=), računar će je protumačiti kao nekorektno napisanu, i zato će prijaviti grešku.

Zaključak: U programu ne koristiti promenljivu M ako će se u nekom njegovom delu pojaviti M+.

2.10.2. Naredba POKE

Ovom naredbom se određenoj memorijskoj lokaciji dodeljuje zadata vrednost. Format naredbe je:

POKE <brojni izraz 1>, <brojni izraz 2>

Ova naredba, koja treba da se upotrebljava sa krajnjom pažnjom, koristi se da se nekoj memorijskoj lokaciji dodeli podatak. Lokacija u memoriji, u koju se smešta podatak, definiše se pomoću brojnog izraza1, a podatak koji se smešta definiše se pomoću brojnog izraza 2. Brojni izraz1 i brojni izraz2 moraju biti decimalni ili heksadecimalni brojevi. Na primer:

```
POKE(5000,255)
```

smešta heksadecimalni ekvivalent za 255 (FF) u memorijsku lokaciju sa decimalnom adresom 5000. Uobičajeniji i sigurniji pristup je

```
POKE(&1388,#FF)
```

što smešta heksadecimalne podatke FF u heksadecimalnu lokaciju 1388.

Naredbe Poke neće se izvršiti, ako je jedan brojni izraz decimalni broj, a drugi heksadecimalni broj.

Primer 10.2. U memorijsku lokaciju sa heksadecimalnom adresom 500 upisati heksadecimalni podatak 88.

Ova operacija se izvršava sa jednom naredbom POKE. Program je:

```
10 POKE (&0500,#88)
:RUN (RETURN)
```

Da bismo proverili da li je naredba POKE korektno izvršena, upotrebićemo neke od naredbi za rad u mašinskom jeziku računara.

Ove naredbe biće kasnije obrađene. Ne zalazeći u njihovu suštinu, preduzmimo sledeće korake:

1. preko tastature ukucajmo PROB i pritisnimo dirku RETURN
2. kada se na ekranu pojavi znak ">" ukucajmo sledeće:

```
D 500 0
```

i pritisnimo dirku RETURN

3. na ekranu će se pojaviti sledeći tekst:

```
0500 88
```

što znači da smo u memorijsku lokaciju sa heksadecimalnom adresom 0500 upisali heksadecimalni podatak 88.

4. povratak u BASIC izvršićemo pritiskom na dirku B.

Treba napomenuti da naredbu POKE koriste iskusniji programeri za kombinovano pisanje programa u BASIC-u i na mašinskom jeziku. Postoje slučajevi kada se određeni delovi programa, napisani na mašinskom jeziku, izvršavaju u zavisnosti od vrednosti pojedinih parametara. Ukoliko znamo tačnu memorijsku lokaciju iz koje računar čita vrednost nekog parametra, naredbom POKE možemo iz BASIC-a upisivati određenu vrednost u tu memorijsku lokaciju. Na taj način menjamo vrednost parametra, a samim tim i izvršenje dela programa napisanog na mašinskom jeziku.

2.11. NAREDBA IZLAZA

U toku izvršavanja naredbi, računar generiše određene rezultate, koje smešta u deo memorije predviđene za to. Da bi te rezultate mogli koristiti, oni moraju biti preneti iz memorije računara i prikazani na ekranu ili papiru štampača. U tu svrhu se koriste naredbe izlaza: PRINT i LPRINT.

2.11.1. Naredba PRINT

Ovom naredbom omogućava se prikazivanje rezultata obrade računara ili neke poruke na ekranu. Za prikazivanje neke poruke na ekranu koristi se format naredbe PRINT

PRINT "<niz>"

gde je
<niz> — skup alfanumeričkih znakova koji se štampa na ekranu.

Za prikazivanje rezultata obrade koristi se format naredbe PRINT

PRINT <naziv promenljive>

gde je:
<naziv promenljive> — oznaka promenljive čija se trenutna vrednost u programu prikazuje na ekranu.

Ako se vrši prikazivanje trenutne vrednosti promenljive, tada promenljiva u programu mora biti izraz ili funkcija niza. Na primer, u programu

```
10 A=4+8
20 B$="VREDNOST PROMENLJIVE A JE:"
30 PRINT B$
40 PRINT A
```

U programskom redu 10 definiše se način određivanja vrednosti promenljive A, a u programskom redu 20 funkcije niza B$. Naredbom u programskom redu 30 prikazuje se sadržaj niza, a naredbom u programskom redu 40 prikazuje se vrednost promenljive A. Posle startovanja programa na ekranu se prikazuje sledeći tekst:

```
VREDNOST PROMENLJIVE A JE:
12
```

Svaka od stavki koja treba da se prikaže na ekranu mora da se odvoji prihvatljivim ograničavačem: zarezom ili tačkom i zarezom. Kada se upotrebi zarez, sledeći znak koji treba da se prikaže postavlja se nakon određenog razmaka. Kada se upotrebi tačka i zarez, ne ubacuju se razmaci i sledeći znak se prikazuje direktno posle poslednjeg.

Primer: Kada se izvrši program

```
10 PRINT "1", "2", "3"
20 PRINT "1";"2";"3"
```

na ekranu se ispisuje sledeći tekst

```
1     2     3
123
```

U toku pisanja programa umesto PRINT možemo pisati skraćeno PR. Prilikom listanja programa, računar će sve PR zameniti sa PRINT.

Primer: Naredba PRINT oblika

```
:PRINT2.4+3.6,20,3*6
```

prikazaće rezultate na ekranu u obliku

```
6         20          18
```

Primer: Naredba PRINT oblika

```
:PRINT"A=";,"B=";4
```

prikazaće rezultate na ekranu u obliku

```
A=3               B=6
```

Primer: Naredba PRINT oblika

```
:PRINT "PECOM";" ";64
```

prikazaće poruku na ekranu u obliku

```
PECOM 64
```

Primer: Program kojim se predstavlja raspodela ukupnog memorijskog prostora u PECOM-u 64.

```
10 CLS
20 PRINT "RASPODELA UKUPNOG MEMORIJSKOG PROSTORA U PECOM-u 64"
30 PRINT
40 PRINT
50 PRINT "ROM MEMORIJA"," 32KB"
60 PRINT "RAM MEMORIJA"," 32KB"
70 PRINT "DISPLEJ MEMORIJA";" 2KB"
80 PRINT "KARAKTER MEMORIJA";" 2KB
:RUN(RETURN)
```

Kao rezultat izvršenja ovog programa imaćemo obrisan ekran i ispisan sledeći tekst:

RASPODELA UKUPNOG MEMORIJSKOG PROSTORA U PECOM-u 64

ROM MEMORIJA	32KB
RAM MEMORIJA	32KB
DISPLEJ MEMORIJA	2KB
KARAKTER MEMORIJA 2KB	

Primer: Izračunati vrednost izraza:

$$y = \frac{1}{a - \frac{x^2}{b - x^3}}$$

za sledeće vrednosti argumenata:

$$a=2,\ b=3,\ x=5.3E-4$$

Program izgleda ovako:

```
10 CLS
20A=2:B=3:X=5.3E-4
30 Y=1(A-X↑2/(B-X↑3))
40 PRINT "A";A "B=";B,"X=";X
50 PRINT "Y=";Y
60 END
```

Vrednosti promenljivih definišemo naredbom sa programskim redom 20. U programskom redu 30 izračunava se vrednost funkcije Y za date vrednosti argumenata A,B,X. Ispisivanje na ekranu vrši se u dva reda, što se ostvaruje sa dve PRINT naredbe (programski redovi 40 i 50):

A=2 B=3 X=.00053
Y=.5

2.11.2. Naredba LPRINT

Svrha ove naredbe je da štampa sadržaj memorije na štampaču. Način korišćenja je isti kao i za naredbu PRINT. Može da se izvršava direktno ili programski (bez ili sa brojem programskog reda). Kao i naredba PRINT i naredba LPRINT ima dva formata.

LPRINT "<niz>"

Posle izvršenja ove naredbe na štampaču se štampa niz definisan izrazom u naredbi pod znacima navoda. Niz može biti bilo koji skup do 128 alfanumeričkih znakova, osim znaka ", koji se upotrebljava za definisanje početka i kraja niza. Na primer, naredbom

```
:LPRINT"ABCDEF"
```

posle pritiska dirke RETURN na štampaču će se odštampati niz:

```
ABCDEF
```

LPRINT <naziv promenljive>

Ovom naredbom se štampaju na štampaču trenutne vrednosti promenljive, koja se određuje nazivom promenljive. Primeri prihvatljivih naredbi su:

```
LPRINT A$
LPRINT A
LPRINT A$,A
```

2.12. NAREDBA ULAZA

Programi u kojima postoje samo naredbe obrade i naredbe izlaza, daju pri svakom izvršavanju iste rezultate. Ovakve programe ima smisla izvršiti samo jednom. Međutim, programi koji sadrže promenljive koje dobijaju vrednosti pri izvršavanju programa, preko ulaznog organa <tastature>, daju različite rezultate pri svakom izvršavanju programa u zavisnosti od ulaznih veličina.

Naredba kojom se dodeljuju vrednosti promenljivim preko ulaznog organa računara, zove se **naredba ulaza.** Ova naredba prenosi podatke sa ulaznog organa (tastature) u memoriju računara. Naredba ulaza ima format

INPUT <naziv promenljive>, <naziv promenljive>

INPUT <naziv promenljive niza>

Kada računar naiđe na neku ulaznu naredbu on se zaustavlja i izdaje na ekranu znak pitanja. Tada čeka na odgovor programera. Posle naredbe INPUT može da se pojavi niz naziva promenljivih. Svaki od naziva promenljive mora biti odvojen zarezom. Kao odgovor na upitnik (?) programer može da odgovori nekim izrazom ili grupom izraza odvojenih zarezima.

Razmotrimo sledeći program

```
10 INPUT A,B,C
20 D=A+B+C
30 PRINT D
```

Kada startujemo ovaj program, na ekranu se pojavljuje znak pitanja.

Računar očekuje unošenje podataka, neophodnih za izvršenje programa. Moramo uneti 3 podatka, međusobno odvojena zarezima. Unošenje podataka završavamo pritiskom na dirku RETURN. Posle toga, računar će izvršiti potrebne operacije nad unetim podacima (sabraće ih), a rezultat će prikazati na ekranu. Na primer, ako posle postavljenog znaka pitanja unesemo sledeće:

```
? 1,2,3
```

posle pritiska na dirku RETURN, na ekranu će se odštampati zbir ova tri broja

```
 6
```

Šta znače sledeća dva programska reda?

```
10 INPUT A$
20 PRINT A$
```

U ovom primeru naredba INPUT koristi se za unošenje elemenata određenog niza. Posle postavljenog znaka pitanja, od strane računara, vršimo unošenje elemenata niza, jedan za drugim bez razdvajanja zarezima. Unošenje niza završavamo priti-

skom na dirku RETURN, posle čega će računar nastaviti izvršenje programa štampanjem unetog niza.

Naredbom INPUT ne sme istovremeno da se unosi vrednost brojne promenljive i elementi niza. Svaka naredba INPUT može da sadrži jedan i samo jedan naziv niza. Prihvatljive naredbe INPUT su:

```
10 INPUT A,B,C(1)
10 INPUT A(1), A(B)
10 INPUT A$
10 INPUT B$(B)
```

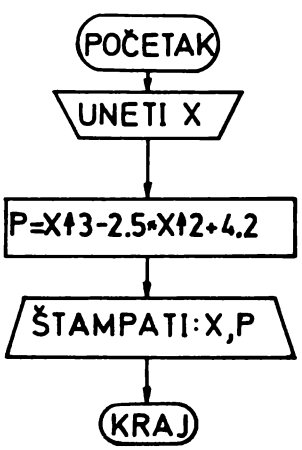
Sledeće naredbe INPUT su pogrešne:

```
10 INPUT AB - nema zareza izmedu promenljivih
10 INPUT A$,B$ - zabranjeni su višestruki nazivi niza
```

Primer: Nacrtati algoritamsku šemu i napisati program za izračunavanje vrednosti polinoma:

$$P(x)=x^3-2{,}5x^2+4{,}2$$

Algoritamska šema je data na sl.2.1.



Sl.2.1

Na postavljen znak pitanja, od strane računara, preko tastature unosimo vrednost promenljive X. Unošenje završavamo pritiskom na dirku RETURN. U drugom koraku računar izračunava vrednost P za unetu vrednost X. U trećem koraku računar štampa na ekranu unetu vrednost X i vrednost polinoma za datu vrednost X.

Program po datoj algoritamskoj šemi je sledeći:

```
10 CLS
20 INPUT X
30 P=X↑3 - 2.5*X↑2 + 4.2
40 PRINT "ZA X=";X;" P(x)=";P
:RUN (RETURN)
```

Naredba INPUT može da se koristi i u modifikovanom obliku:

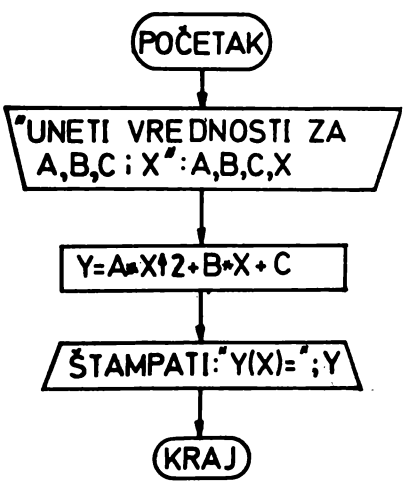
INPUT "niz" <naziv promenljive>, <naziv promenljive>

Ova INPUT naredba je identična sa prethodnom, osim što se direktno posle INPUT naredbe javlja navodnik koji dopušta da se poruka prikaže pre prikazivanja znaka pitanja. Pre prvog navodnika ili posle drugog ne koristi se zarez.

Primer: Nacrtati algoritamsku šemu i napisati program za izračunavanje vrednosti funkcije

$$y(x)=ax^2+bx+c$$

Algoritamska šema je data na sl.2.2.



Sl.2.2

U ovom primeru INPUT naredbu primenjujemo u drugačijoj formi od prethodnog primera. Ovde vrednosti promenljivih unosimo nakon prikazivanja sledeće poruke na ekranu "UNETI VREDNOSTI A,B,C,X". Dalji postupak izvršenja programa je isti kao i u prethodnom primeru, s tom razlikom što se vrednost funkcije izračunava po drugoj formuli.

Program po datoj algoritamskoj šemi je sledeći:

```
10 CLS
20 INPUT "UNETI VREDNOSTI ZA A,B,C,X"A,B,C,X
30 Y=A*X↑2+B*X+C
40 PRINT "Y(X)=";Y
```

Posle startovanja programa na ekranu se štampa poruka

```
UNETI VREDNOSTI ZA A,B,C,X?
```

nakon koje unosimo vrednosti za A,B,C,X. Vrednosti za svaku promenljivu moraju biti razdvojene zarezom. Posle unošenja poslednje vrednosti, pritiskom na dirku RETURN računar nastavlja izvršenje programa sa programskim redom 30.

2.13. NAREDBA ZA KRAJ PROGRAMA

Format ove naredbe je:

END

Ova naredba (kraj programa) služi za zaustavljanje ili završetak programa i vraća BASIC na režim dirketnog izvršenja. Naredba END može da se koristi bilo gde u programu u BASIC-u i to proizvoljan broj puta. Ona se, ako se želi, može izostaviti u programu.

2.14. NAREDBA ZA KOMENTAR U PROGRAMU

U programu je često potrebno pisati komentare, koji objašnjavaju program ili delove programa. Ovo je omogućeno naredbom za komentar REM, koja ima format:

REM <osnovni simbol>

Kada se REM postavi na početak nekog programskog reda, ceo red se prikazuje kod listanja, ali i ignoriše za vreme izvršenja programa. njena svrha je da omogući programeru da komentariše program tekstom, zadanim u osnovnom simbolu.

2.15. ELEMENTARNE FUNKCIJE

Elementarne funkcije mogu biti:
- aritmetičke (numeričke)
- slovne
- konverzione
- ostale

2.15.1. Aritmetičke funkcije

Aritmetičke funkcije omogućavaju izračunavanje trigonometrijskih i algebarskih funkcija na jednostavan način. Ove funkcije se pišu na sličan način kao i u matematici. Vrednost ovakvih funkcija je brojni podatak.

U aritmetičke funkcije spadaju: INT, FNUM, INUM, MOD, ABS, SGN, SQR, LOG, EXP, SIN, COS, ATN, PI.

Korišćenje celih brojeva

U skoro svim programima, koji kao rezultat izvršenja treba da daju grafički prikaz na ekranu, neophodno je pretvaranje decimalnih brojeva u cele. Za obavljanje takvih aktivnosti programeru na raspolaganju stoji nekoliko funkcija: INT, FNUM, INUM, MOD. Ove funkcije sa različitom tačnošću vrše zaokrugljivanje decimalnih brojeva na cele.

INT <brojni izraz>

Kao rezultat ova funkcija daje celobrojni deo brojnog izraza sa pokretnim zarezom, odbacujući razlomljeni deo. Rezultat je i dalje u režimu pokretnog zareza. Napišimo sledeći program:

```
10 A=26.47
20 B=2.99
30 PRINT INT (A)
40 PRINT INT (B)
:RUN (RETURN)
```

Kao rezultat izvršenja ovog programa računar će na ekranu odštampati broj 26, što predstavlja celobrojni deo broja A. Tačnost ove funkcije zavisi od broja koji se nalazi u brojnom izrazu. Kao što ćemo kasnije videti, i drugim funkcijama bi se broj 26.47 zaokružio na 26. Mešutim, ako funkcijom INT zaokružujemo broj 2.99 na ceo, dobićemo kao rezultat broj 2, što u mnogim slučajevima nije dovoljna tačnost.

FNUM <brojni izraz>

Za razliku od funkcije INT, funkcija FNUM zaokružuje vrednost brojnog izraza na najbliži ceo broj i pretvara ga u broj sa pokretnim zarezom. Napišimo sledeći program:

```
10 A=26.47
20 B=2.99
30 PRINT FNUM (A)
40 PRINT FNUM (B)
:RUN (RETURN)
```

Funkcija FNUM će broj 26.47 zaokružiti na 26. kao i funkcija INT, ali će zato broj 2.99 zaokružiti na 3.

Ako je vrednost brojnog izraza ceo broj, funkcija FNUM neće imati potrebe za zaokruživanjem na najbliži ceo broj, ali će zato taj broj pretvoriti u broj sa pokretnim zarezom. Na primer, naredba

```
PRINT FNUM (2*3)
```

imaće za rezultat

```
6.
```

INUM <brojni izraz>

Za razliku od funkcije FNUM, funkcija INUM vrednost brojnog izraza, sa pokretnim zarezom, pretvara u režim celobrojnog broja sa zaokruživanjem na najbliži ceo broj. Funkcija INUM pruža mogućnost za prebacivanje neke posebne funkcije u ceo broj. Napišimo sledeći program:

```
10 A=26.47
20 B=2.99
30 PRINT INUM (A)
40 PRINT INUM (B)
:RUN (RETURN)
```

Funkcija INUM će brojeve 26.47 i 2.99 zaokružiti, kao i funkcija FNUM, na najbliži ceo broj, s tom razlikom što vrednost zaokruženog broja postaje celobrojna (nema tačke iza najniže cifre).

Na kraju ovog dela objasnićemo posebnu funkciju

MOD <brojni izraz1>,<brojni izraz2>

koja kao svoju vrednost daje sledeći ekvivalent
<brojni izraz1>—<brojni izraz1>/<brojni izraz2>*<brojni izraz2> gde se svaki brojni izraz prvo pretvara u ceo broj, pa je i vrednost funkcije ceo broj.

Na primer, naredba

```
PRINT MOD (10,3.)
```

imaće za rezultat

```
1
```

Određivanje apsolutne vrednosti i znaka broja

U BASIC-u, za dobijanje apsolutne vrednosti broja postoji funkcija ABS, koja ima sledeći format

ABS(<brojni izraz>)

Ova funkcija daje apsolutnu vrednost brojnog izraza. Funkcija ABS ne menja vrednost pozitivnog broja, ali zato ignoriše znak negativnog. Na primer, naredba

```
PRINT ABS (-10*2)
```

imaće za rezultat

```
20
```

Znak broja dobija se funkcijom SGN, formata

SGN(<brojni izraz>)

Vrednost ove funkcije je +1, 0 ili —1, zavisno od znaka brojnog izraza:

— ako je brojni izraz pozitivan, funkcija SGN ima vrednost +1
— ako je brojni izraz 0, funkcija SGN ima vrednost 0
— ako je brojni izraz negativan, funkcija SGN ima vrednost —1

Korenovanje i stepenovanje na računaru

Od matematičkih funkcija najčešće se koriste one za korenovanje i stepenovanje brojeva.

Kvadratni koren nekog brojnog izraza izračunava se naredbom SQR, čiji je format

SQR(<brojni izraz>)

Ova funkcija daje kvadratni koren vrednosti brojnog izraza i to u pokretnom zarezu.

Stepenovanje se obeležava stelicom nagore (↑). Na primer, 2^3 se piše 2↑3. S obzirom da programi za stepenovanje rade sporije i netačno, poželjno je stepenovanje izbegavati i prevoditi ga u množenje kada je to moguće. Na primer, naredba

```
PRINT 2↑3
```

imaće za rezultat

```
 7.99999
```

a naredba

```
PRINT 2*2*2
```

imaće za rezultat

```
8.
```

Logaritamska i eksponencijalna funkcija

BASIC PECOM-a 64 može da izvršava prirodan logaritam, odnosno logaritam za osnovu e=2.71828. U tu svrhu koristi naredbu LOG, formata

LOG(<brojni izraz>)

Ova funkcija daje broj sa pokretnim zarezom, čija je vrednost jednaka prirodnom logaritmu vrednosti brojnog izraza. Na primer, naredba

```
PRINT LOG (2.71828)
```

imaće za rezultat

```
.999999
```

Operacija suprotna logaritmovanju je eksponenciranje. Eksponencijalna funkcija ima format

EXP(<brojni izraz>)

Ova funkcija daje broj sa pokretnim zarezom, čija je vrednost broj e=2.71828 dignut na stepen vrednosti brojnog izraza. Na primer, naredba

```
PRINT EXP (2)
```

imaće za rezultat

```
7.38906
```

Trigonometrijske funkcije

BASIC može da izračunava trigonometrijske funkcije: sinus, kosinus, arcus tangens, dok se ostale funkcije mogu izraziti pomoću ovih.

SIN(<brojni izraz>)

Ova funkcija daje sinus ugla određenog vrednošću brojnog izraza izraženog u radijanima. Rezultat funkcije SIN je broj sa pokretnim zarezom. Na primer, naredba

```
PRINT SIN (PI/6)
```

izračunavaće vrednost sinusa ugla PI/6 u radijanima, kome odgovara ugao od 30 stepeni. Rezultat izvršenja naredbe je

```
.5
```

COS(<brojni izraz>)

Ova funkcija daje kosinus ugla određenog vrednošću brojnog izraza, izraženog u radijanima. Rezultat funkcije COS je broj sa pokretnim zarezom. Na primer, naredba

```
PRINT COS (PI/3)
```

imaće za rezultat

```
.500001
```

ANT <brojni izraz>

Ova funkcija kao svoju vrednost daje ugao izražen u radijanima, čiji je tangens jednak vrednosti brojnog izraza. U matematici se ova funkcija zove ARCUS TANGENS.

Ostale aritmetičke funkcije

PI

Ova funkcija daje kao svoju vrednost broj 3.14159.

2.15.2. Slovne funkcije

Slovne funkcije kao vrednost mogu imati slova ili brojni podatak. Ako funkcija ima vrednost slovo, tada se ime funkcije završava znakom $. Drugačije, ove funkcije se nazivaju funkcije niza. U ovu grupu spadaju funkcije LEN i MID$.

Definisanje dužine specificiranog niza

U ovu svrhu BASIC koristi funkciju LEN, formata:

LEN(<naziv niza>)

Ova funkcija daje kao svoju vrednost broj znakova u nizu specificiranom nazivom niza. Vrednost se daje u pokretnom zarezu osim ako joj ne prethodi neki ceo broj ili funkcija, u kom slučaju se automatski pretvara u ceo broj. Na primer, u programu:

```
10 A=1
20A$="ARITMETIKA"
30 PRINT LEN (A$)
40 PRINT A*LEN (AS)
:RUN (RETURN)
```

naredba

```
PRINT LEN (A$)
```

daje

```
10. (broj znakova u nizu A$)
```

a

```
PRINT A*LEN (A$)
```

daje 10 (pretpostavlja se da je promenljiva A definisana kao ceo broj sa jedinicom upamćenom kao njenom vrednošću). Važno je napomenuti da niz mora biti prethodno definisan.

Primer: Napisati program koji će izračunavati koliko znakova sadrži proizvoljno unet tekst. U tekstu se i znak za blanko tretira kao znak.

Program je sledeći:

```
10 CLS
20 INPUT A$
30 PRINT "TEKST SADRŽI ";LEN A$;" ZNAKOVA."
:RUN (RETURN)
```

Posle startovanja programa, na postavljen znak pitanja treba uneti proizvoljan tekst (do 95 znakova). Posle pritiska na dirku RETURN računar odgovara tekstom:

```
TEKST SADRŽI xx ZNAKOVA
```

gde je xx broj znakova koje sadrži proizvoljno unet tekst.

Definisanje podniza iz specificiranog niza

BASIC poseduje funkciju kojom se iz datog niza može izdvojiti deo niza. Format ove naredbe je:

MID$ <naziv niza>, <brojni izraz1>, <brojni izraz2>

Ovom funkcijom se vrši izvlačenje dela niza definisanog nazivom niza. Podsetimo se da naziv niza mora da sadrži slovo i znak $. <brojni izraz1> definiše koji znak sleva treba da započne podniz <brojni izraz2> definiše broj znakova koji treba da se upotrebe za generisanje podniza, počev od znaka zadatog brojnim izrazom1

Na primer, iz niza

```
A$(10)="EKSPERIMENT"
```

izdvojićemo podniz B$, definisan funkcijom MID$ na sledeći način

```
B$=MID$(A$(10),4,6)
```

Podniz B$ generišemo od elemenata niza A$(10) na taj način što će prvi element podniza B$ biti četvrti element niza A$(10), i još pet elemenata sleva udesno. Podniz B$ će tada biti

```
B$="PERIME"
```

što se može proveriti sledećim programom:

```
10A$="EKSPERIMENT"
20B$MID$(A$(10),4,6)
30 PRINT B$
:RUN (RETURN)
```

koji nakon što se izvrši štampa na ekranu podniz B$:

```
PERIME
```

Funkcija MID$ može se koristiti u formatu

MID$ <naziv niza>, <brojni izraz1>

Kada se izostavi brojni izraz2, tada će funkcija MID$ koristiti sve znakove desno od znaka definisanog brojnim izrazom1. U ovom slučaju program iz prethodnog primera bio bi sledeći

```
10A$="EKSPERIMENT"
20B$=MID$(A$(10),4)
30 PRINT B$
:RUN (RETURN)
```

a posle njegovog izvršenja na ekranu se štampa podniz

```
PERIMENT
```

2.15.3. Konverzione funkcije

Konverzionim funkcijama se vrši prebacivanje slovnog podatka u brojni i obrnuto. Funkcije ove grupe su: ASC, FVAL, PEEK, CHR$, STR$.

Konverzija slovnog podatka u brojni podatak

Ova funkcija daje, kao svoju vrednost, decimalnu vrednost prvog znaka u nizu definisanog nazivom niza, i to u pokretnom zarezu. Ako ovoj funkciji prethodi neka celobrojna funkcija, pretvoriće se u ceo broj. Na primer, program:

```
10A$(5)="ARITMETIKA"
20B=ASC(A$(5))
30 PRINT B
:RUN (RETURN)
```

posle izvršenja, na ekranu štampa vrednost promenljive B, koja će kao svoju vrednost sadržati broj 65, decimalni kod slova A.

Za izračunavanje izraza niza kao aritmetički izraz koristi se funkcija FVAL, formata

FVAL(<niz>)

Ova funkcija izračunava izraz kao aritmetički izraz i daje njegovu vrednost. Na ovaj način korisnik može da kreira matematičke funkcije.

Primer datog formata funkcije FVAL je sledeći

```
10 PRINT FVAL ("SQR(3)")
:RUN (RETURN)
```

Na ekranu se štampa vrednost niza "SQR(3)", a to je

```
1.73205
```

Funkcija FVAL se može koristiti i u formatu

FVAL <naziv niza>

gde niz specificiran nazivom niza mora biti prethodno definisan. Primer ovakvog formata naredbe FVAL je sledeći:

```
10 A$="8+4"
20 PRINT FVAL (A$)
30 PRINT FVAL(A$+"/2")
:RUN (RETURN)
```

Izvršavanjem programskog reda sa brojem 20 računar će na ekranu štampati vrednost prethodno definisanog niza A$. U programskom redu 30 računar nizu A$ pridodaje niz "/2" i definiše novi:

"8+4/2"

a zatim na ekranu štampa njegovu vrednost.

Posle izvršavanja programa na ekranu se štampa rezultat

```
12
10
```

Kod funkcije FVAL dužina niza je ograničena na 48 znakova.

PEEK(<brojni izraz>)

Ova funkcija daje decimalnu vrednost sadržaja memorijske lokacije čija je vrednost određena decimalnim brojem brojnog izraza. Decimalni rezultat prikazuje se u pokretnom zarezu. Na primer:

```
10 PRINT PEEK (36864)
```

Posle izvršenja ove naredbe računar će na ekranu štampati broj 113. Proverimo šta smo dobili ovom naredbom. Brojni izraz u PEEK naredbi je decimalni broj, a memorijske lokacije u računaru su numerisane heksadecimalnim brojevima. Prevedemo li decimalni broj 36864 u heksadecimalni, dobićemo broj 9000. Naredba PEEK i sadržaj određene memorijske lokacije prikazuje decimalnim brojem. Prevedemo li decimalni broj 113 u heksadecimalni, dobićemo broj 71. Sada možemo zaključiti da

se u memorijskoj lokaciji sa heksadecimalnom adresom 9000 nalazi heksadecimalni podatak 71.

Ovaj zaključak možemo proveriti prikazivanjem sadržaja memorijske lokacije sa adresom 9000 na ekranu. U tu svrhu moramo upotrebiti neke od naredbi mašinskog jezika, koje će kasnije biti objašnjene. Za nas je dovoljno da sprovedemo sledeći postupak:

1. preko tastature ukucajmo naredbu PROB i pritisnimo dirku RETURN

2. kada se pojavi znak ">" ukucajmo D9000 i pritisnimo dirku za blanko a zatim dirku RETURN

3. na ekranu će se odštampati tekst

```
9000 71
```

Na osnovu ovoga zaključujemo da je u memorijskoj lokaciji 9000 smešten podatak 71. Za povratak u BASIC dovoljno je pritisnuti dirku B.

Konverzija brojnog podatka u slovni podatak

CHR$ (<brojni izraz>, <brojni izraz>,...)

Funkcija CHR$ se prepoznaje i koristi samo u naredbi PRINT. Ona brojni izraz tretira kao decimalni kod određenog znaka i taj znak štampa na ekranu. Brojni izraz može biti decimalni ili heksadecimalni broj. Ako je brojni izraz decimalni broj, on može uzimati vrednost od 0 do 127. Na primer, naredba

```
PRINT CHR$(65)
```

imaće za rezultat štampanje na ekranu znaka A.

Ako je brojni izraz heksadecimalni broj, tada se mora označiti znakom # i može uzimati vrednosti od 0 do 7F.

Naredba

```
PRINT CHR$(#42)
```

imaće za rezultat štampanje na ekranu znaka B.

Naredba

```
PRINTCHR$(#41,#42,#43)
```

imaće za rezultat

```
ABC
```

Primer: Na ekranu ispisati tekst

```
OVO JE
TEST
```

primenom samo jedne naredbe CHR$.

Da bismo rešili ovaj zadatak treba da znamo decimalne kodove upotrebljenih znakova:

```
O=79        T=84
V=86        S=83
J=74        blanko=32
E=69
```

Decimalni kod naredbe LF (pomeranje kursora za jedan red naniže) je 10. Decimalni kod naredbe CTRL-H (pomeranje kursora za jednu poziciju ulevo) je 8.

Nizom decimalnih kodova

```
79,86,79,32,74,69
```

ispisujemo tekst OVO JE. U novi red prelazimo sa decimalnim kodom 10. Da bismo kursor vratili ispod slova V moramo 5 puta vršiti pomeranje decimalnim kodom 8 (CTRL-H). Ostatak teksta ispisujemo nizom decimalnih kodova

```
84,69,83,84
```

Program je sledeći:

```
10 CLS
20PRINTCHR$(79,86,79,32,74,69,10.8,8,8,8,8,84,69,83,84)
:RUN (RETURN)
```

Rezultat ovog programa je ispisivanje teksta

```
OVO JE
TEST
```

STR$(<brojni izraz>)

Ovom naredbom cifre brojnog izraza postaju elementi novonastalog niza.

Primer: Sledećim programom:

```
10A$=STR$(55555)
20 PRINT A$
:RUN (RETURN)
```

nizu A$ dodeljuje se vrednost cifara koje se nalaze u brojnom izrazu naredbe STR$. Novi niz A$ štampa se na ekranu.

Brojni izraz ne mora biti isključivo numerička vrednost, već to može biti i neki izraz.

Primer:

```
10 A$="TEST"
20PRINTSTR$(ASC(MID$(A$,1,1)))
:RUN (RETURN)
```

Naredbom MID$(A$,1,1) izdvaja se prvi znak niza A$. Naredbom ASC dodeljuje se funkciji MID$ decimalni kod izdvojenog prvog znaka niza A$, koji postaje element novoformiranog niza. Isti efekat se postiže sledećim programom:

```
10 D$="TEST"
20 E=ASC(D$)
30 E$=STR$(E)
40 PRINT E$
:RUN (RETURN)
```

2.15.4. Ostale funkcije

Generisanje slučajnih brojeva

RND

Ova funkcija, kao rezultat, daje slučajan broj sa pokretnim zarezom koji je veći ili jednak nuli a manji ili jednak jedinici.

RND(<brojni izraz>)

Ova funkcija daje slučajan broj veći ili jednak nuli a manji ili jednak vrednosti brojnog izraza.

Uređenje izlaznog reda

TAB (<brojni izraz>)

Ova funkcija nije funkcija u istom smislu kao prethodne funkcije zbog toga što ne daje nikakvu vrednost. Ona se koristi i prepoznaje samo u naredbi PRINT i postavlja kursor u horizontalni položaj određen vrednošću brojnog izraza. Novi položaj kursora određuje se u odnosu na kolonu 1, a ne u odnosu na prethodni položaj kursora. Prikazivanje se nastavlja od te kolone pa nadalje.

Na primer, naredba

```
PRINT TAB(10);1
```

prikazuje znak 1 u koloni 10

Naredba

```
PRINTTAB(10);1;TAB(20);2
```

prikazuje 1 u koloni 10 a 2 u koloni 20.

Naredba

```
PRINT TAB(A);"*" , A<-39
```

prikazuje zvezdicu u koloni A.

Postavljanje jedinice mere za ugaone funkcije

DEG

Ova funkcija postavlja jedinicu mere za ugaone funkcije na stepene.

Kada uključimo računar, njegov BASIC ugaone funkcije izračunava u radijanima. Na primer, naredba

```
PRINT SIN(10)
```

posle pritiska na dirku RETURN na ekranu štampa vrednost funkcije SIN(10) u radijanima

```
-.544022
```

Naredbom DEG računar prelazi u režim rada u kome ugaone funkcije izračunava u stepenima. Ako ponovo unesemo naredbu

```
PRINT SIN(10)
```

i pritisnemo dirku RETURN, računar će na ekranu štampati vrednost ove funkcije u stepenima

```
.173684
```

RAD

Ovom funkcijom ponovo možemo postaviti jedinicu mere za ugaone funkcije na radijane. Funkciju RAD možemo upotrebiti u bilo kom delu programa.

Određivanje decimalnog ekvivalenta koda pritisnute dirke na tastaturi

KEY

Kada računar naiđe na ovu naredbu on ispituje koja je od dirki na tastaturi pritisnuta. Kada se pritisne neka dirka, tada računar dodeljuje funkciji KEY vrednost koja odgovara decimalnom kodu pritisnute dirke.

Naredba za definisanje programskog kašnjenja

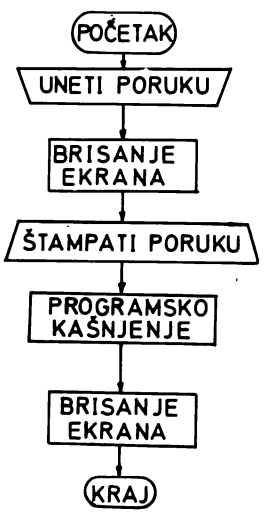
Često je u toku izvršavanja programa potrebno da se određena poruka ili uputstvo određeni vremenski interval prikažu na ekranu, a da se zatim nastavi sa izvršenjem programa. U BASIC—u postoji naredba WAIT kojom se može uneti željeno kašnjenje u izvršenju programa. Format ove naredbe je

WAIT (<brojni izraza>)

Ova naredba omogućava unošenje kašnjenja u izvršenje programa. Dužina kašnjenja je jednaka broju ciklusa određenim brojnim izrazom. Jedan ciklus kašnjenja traje oko 8 mili sekundi.

Primer: Nacrtati algoritamsku šemu i napisati program koji će prikazivati poruku na ekranu oko 4 sekundi (500 ciklusa kašnjenja) a zatim će obrisati ekran.

Algoritamska šema je data na sl.2.3.



Sl.2.3

Program za datu algoritamsku šemu je sledeći:

```
10 INPUT A$
20 CLS
30 PRINT "PORUKA JE":";A$
40 WAIT (500)
50 CLS
60 END
:RUN (RETURN)
```

Posle startovanja programa na ekranu se pojavljuje znak pitanja, posle koga unosimo željeni tekst. Unošenje završavamo pritiskom na dirku RETURN. Uneseni tekst se štampa. Posle 4 sekunde ekran se briše i izbacuje poruku o završetku programa (READY).

Naredba za trasiranje programskih redova

Prilikom analize i ispitivanja tačnosti složenijih programa može se javiti potreba da se prati redosled izvršavanja naredbi startovanog programa. U BASIC-u postoji naredba TRACE koja omogućuje da se na ekranu pored poruka, koje izdaje računar u toku izvršavanja programa, prikazuju i brojevi onih programskih redova koji se trenutno izvršavaju.

Naredba TRACE je posebno pogodna za analizu razgranatih programa, kod kojih redosled izvršavanja naredbi nije unapred poznat, već zavisi od trenutnih vrednosti parametara programa. Na taj način programer tačno zna koji se deo programa u datom trenutku izvršava (trasira svoj program).

Format naredbe TRACE je

TRACE (<brojni izraz>)

Ako je brojni izraz bilo šta osim nule, vršiće se trasiranje, odnosno svaki broj programskog reda koji se izvršava štampaće se na ekranu. Poruka o trasiranju je:

```
TR $<broj programskog reda>$
```

Primer: Izvršićemo trasiranje sledećeg programa:

```
1 TRACE
10 PRINT "OVO JE"
20 WAIT(500)
30 PRINT "ISPITIVANJE"
40 WAIT(500)
50 PRINT "NATEDBE TRACE"
60 END
:RUN (RETURN)
```

Posle startovanja programa računar će na ekranu štampati sledeći tekst:

```
TR $10$
OVO JE
TR $20$
TR $30$
ISPITIVANJE
TR $40$
TR $50$
NAREDBE TRACE
TR $65535$
```

Poruka na kraju teksta govori da je ispitana svaka memorijska lokacija do kraja korisničkog memorijskog prostora.

2.16. NAREDBE SKOKA

U toku kreiranja programa često je potrebno bezuslovno promeniti redosled izvršavanja programskih redova. Ovo se postiže naredbom bezuslovnog skoka (GO TO). Pored nje postoji naredba uslovnog skoka (IF) koja omogućuje skok na ukazan broj programskog reda u zavisnosti od toga da li je navedeni uslov ispunjen ili nije.

2.16.1. Naredba bezuslovnog skoka

Često je potrebno da se neki deo programa preskoči i, u određenim uslovima, ne izvrši. Za vraćanje na neki prethodni programski red, ili za preskakanje nekoliko sledećih, koristi se naredba GO TO. PECOM 64 prepoznaje naredbu ako su reči GO i TO napisane spojeno. GO TO bukvalno znači IDI NA, pa u kombinaciji sa brojem programskog reda BASIC programa omogućava da se delovi programa ponove ili preskoče.

Format naredbe bezuslovnog skoka je

GO TO <brojni izraz>

Kada računar u toku izvršavanja programa naiđe na ovu naredbu, on vrši bezuslovno grananje i prenosi izvršenje programa na broj programskog reda koji je određen brojnim izrazom. Ako ne postoji programski red sa brojem zadatim brojnim izrazom, generiše se poruka o greški.

Primer 16.1.1. Nacrtati algoritamsku šemu i napisati program koji će neprekidno prikazivati određeni sadržaj na ekranu.

Algoritamska šema je data na sl.2.4

Sl.2.4

Posle štampanja određenog teksta na ekranu, računar će izvršiti naredbu programskog kašnjenja. Posle toga će naredbom bezuslovnog skoka preći na ponovno izvršenje naredbe kašnjenja. Na taj način je formiran beskonačni ciklus koji se može prekinuti naredbom za prekid programa (izvršava se pritiskom na dirku BREAK). Program prema datoj algoritamskoj šemi je sledeći:

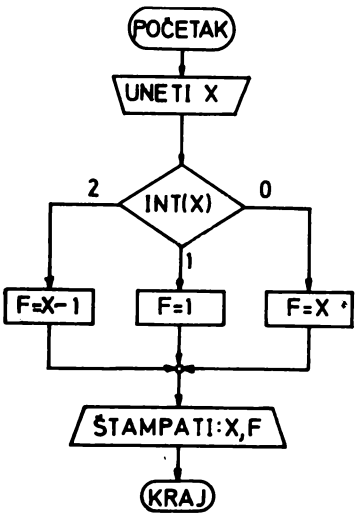
```
10 CLS
20 PRINT "OVO JE PROGRAM ZA DEMONSTRIRANJE"
30 PRINT "NAREDBE GOTO."
40 PRINT "PROGRAM ĆE SE IZVRŠAVATI SVE DOTLE"
50 PRINT "DOK NE PRITISNETE DIRKU BREAK."
60 WAIT (10)
70 GOTO 60
```

Primer: Funkcija F(x) definisana je sa:

$$F(x)=\begin{cases} x & 0\leqslant x<1 \\ 1 & 1\leqslant x<2 \\ x-1 & 2\leqslant x<3 \end{cases}$$

Napisati program kojim se izračunava i štampa vrednost funkcije F za vrednost argumenta X, koji se učitava.

Algoritamska šema je data na sl.2.5.



Sl.2.5

Program za datu algoritamsku šemu je sledeći:

```
10 INPUT"X="X
20 GOTO INT(X)+30
30 F=X:GOTO 40
31 F=1:GOTO40
32 F=X-1
40 PRINT "ZA X=";X;"F=";F
50 END
```

U programskom redu 20 naredba INT(X) izdvaja celobrojni deo unete vrednosti argumenta X. Za vrednost promenljive X između 0 i 1 vrednost funkcije INT(X) je 0 i program se nastavlja sa naredbom u programskom redu 30, gde je F=X.

Za vrednost promenljive X između 1 i 2, vrednost funkcije INT(X) je 1 i program se nastavlja sa naredbom u programskom redu 31, gde je F=1.

Za vrednost promenljive X između 2 i 3, vrednost funkcije INT(X) je 2 i program se nastavlje sa naredbom u programskom redu 32, gde je F=X—1.

U svim slučajevima se prelazi na izvršenje naredbe u programskom redu 40 (štampanje vrednosti promenljive i funkcije F).

2.16.2. Naredba uslovnog skoka

Mogućnost donošenja odluka je jedna od najznačajnijih prednosti računara nad kalkulatorima, i zasniva se na naredbama grananja, koje obezbeđuju upravljanje redosledom izvršavanja naredbi programa u zavisnosti od toga da li je ispunjen neki uslov.

IF...THEN (ako...onda) je jedna od naredbi prenosa upravljanja koja može narušiti uobičajeni redosled izvršavanja naredbi. Podrazumeva se da se naredbe

izvršavaju onim redom kojim su fizički smeštene u program. Na početku se izvršava prva naredba, kada se njena realizacija okonča prelazi se na sledeću i tako redom. Da bi se ovaj redosled mogao promeniti, imamo mogućnost da naredbama dodelimo adrese (brojeve programskih redova), na koje prenosimo upravljanje naredbama bezuslovnog skoka (GO TO) ili naredbama uslovnog skoka (IF...THEN).

Format naredbe IF...THEN je

IF <relacijskii zraz> THEN <naredba>

Ova naredba je uslovna naredba ali nije uslovno grananje kao kod ostalih verzija BASIC jezika. Ona testira uslov i ako je uslov zadovoljen, naredba i grupa naredbi (odvojenih dvema tačkama) se izvode. Ako uslov nije zadovoljen, tada se izvođenje nastavlja sa sledećim programskim redom. Relacijski izraz može biti:

— izraz nizova
— aritmetički izraz

između kojih se nalaze relacioni operatori. Kada se upotrebljavaju izrazi nizova jedini prihvatljivi operatori su znak jednako i nejednako. Prihvatljive naredbe IF sa izrazima nizova date su u sledećem programu:

```
10 A$="ACA"
20 B$="ANA"
30 IF A$=B$ THEN GOTO 50
40 IF A$=B$ THEN GOTO 60
50 PRINT "NIZOVI SU JEDNAKI"
60 PRINT "NIZOVI SU NEJEDNAKI"
70 END
```

U datom programu, naredbama u programskim redovima 30 i 40 vrši se upoređivanje nizova zbog jednakosti. Ako su nizovi jednaki izvršava se naredba u programskom redu 50, a ako su nejednaki izvršava se naredba u programskom redu 60.

U slučaju aritmetičkih izraza prihvatljivi operatori su:

```
= jednako
<> nejednako
veće od
< manje od
>= veće ili jednako
<= manje ili jednako
```

Tip prvog izraza definiše tip drugog izraza. Primer uslovne naredbe je:

```
10 IF A=B THEN PRINT A:WAIT(200):CLS:GOTO 200
20 PRINT B
```

U ovom slučaju, ako je vrednost za A jednaka vrednosti za B, tada će se vrednost za A prikazati, doći će do programskog kašnjenja, ekran će biti obrisan i izvršenje će se nastaviti sa programskim redom 200. Ako vrednost za A nije jednaka vrednosti za B, tada se vrednost za B prikazuje, a izvršenje programa se nastavlja od programskog reda 20 pa nadalje.

Još jedna stvar koju treba napomenuti je da je ključna reč THEN opciona i ne mora da se upotrebi. Uobičajena greška koja treba da se izbegne je:

```
IF A>N THEN 200
```

Sve zdesna od THEN, ako se javi, treba da bude izvršna naredba. Broj 200 nije izvršna naredba. Tačna verzija bi bila:

```
IF A>B THEN GOTO 200
```

ili

```
IF A>B GOTO 200
```

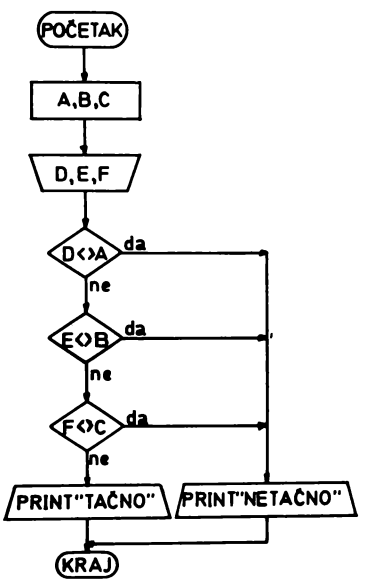
Obratimo pažnju na sledeću naredbu:

```
10 IF A=INT(S/5) THEN IF B>10 THEN IF C<5 THEN GOTO 500
```

U ovom primeru postoje višestruki uslovi. Ako neki od uslova nisu zadovoljeni izvršenje se nastavlja u sledećem programskom redu. Ako se zadovolje svi uslovi, izvršenje će se konačno preneti na programski red 500 pomoću naredbe GOTO. Umesto naredbe GOTO može stajati bilo koja izvršna naredba.

Primer: Napravićemo kratak test program iz istorije. Na postavljeno pitanje: Kada je održano drugo zasedanje AVNOJ-a?, treba dati odgovor. Računar odlučuje o tome da li je odgovor tačan ili ne.

Algoritamska šema za dati primer je data na sl.2.6.



Sl.2.6

Tačni podaci su dati vrednostima za promenljive A,B,C. Na postavljeno pitanje unosimo vrednosti za promenljive D,E,F. Računar ispituje da li je D=A,E=B,F=C i ako su svi uslovi ispunjeni štampa poruku o tačnosti rešenja. Ako bar jedan od uslova nije ispunjen javlja se poruka o netačnosti rešenja i daje prikaz tačnog rešenja.

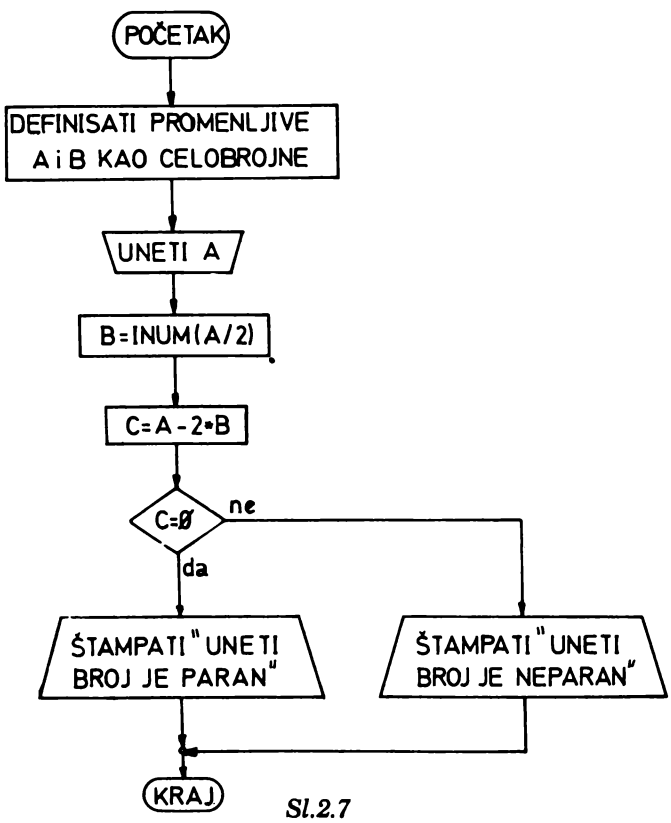
Program po datoj algoritamskoj šemi je sledeći:

```
10 REM *** TEST IZ ISTORIJE ***
20 CLS
30 PRINT "KADA JE ODRŽANO DRUGO ZASEDANJE AVNOJ-a"
40 A=29
50 B=11
60 C=1943
70 INPUT "DAN(DD):"D
80 INPUT "MESEC(MM):"E
```

```
90 INPUT "GODINA(GGGG):" F
100 IF D<>A GOTO 150
110 IF E<>B GOTO 150
120 IF F<>C GOTO 150
130 PRINT "ODGOVOR JE TAČAN"
140 END
150 PRINT "ODGOVOR NIJE TAČAN"
160 PRINT "DRUGO ZASEDANJE AVNOJ-a ODRŽANO JE"
170 PRINT "29.11.1943. GODINE"
180 END
```

Primer: Nacrtati algoritamsku šemu i napisati program koji za uneti ceo broj izdaje poruku da li je paran ili neparan.

Algoritamska šema za ovaj primer data je na sl.2.7.



Sl.2.7

Na početku programa se promenljive A i B definišu kao celobrojne. Posle unošenja broja A, računar izračunava vrednost promenljive B po obrascu:

```
B=INUM(A/2)
```

B se definiše kao najbiliži ceo broj broja A podeljenog sa 2. Vrednost promenljive C se izračunava po obrascu:

```
C=A-2*B
```

IF naredba ispituje vrednost za C: ako je jednako 0, tada se štampa poruka da se radi o parnom broju, a ako taj uslov nije ispunjen štampa se poruka da se radi o neparnom broju.

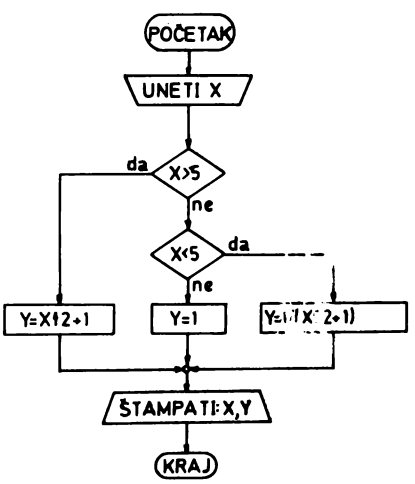
Program za datu algoritamsku šemu je sledeći:

```
10 DEFINT B
20 INPUT A
30 B=INUM(A/2)
40 C=A-2*B
50 IF C=0 THEN GOTO 80
60 PRINT "UNETI BROJ JE NEPARAN"
70 GOTO 90
80 PRINT "UNETI BROJ JE PARAN"
90 END
```

Primer: Nacrtati algoritamsku šemu i napisati program za izračunavanje vrednosti funkcije:

$$y=\begin{cases} x^2+1 & x>5 \\ 1 & x=5 \\ \dfrac{1}{x^2+1} & x<5 \end{cases}$$

Algoritamska šema je data na sl.2.8.



Sl.2.8

Program po datoj algoritamskoj šemi je sledeći:

```
1 REM *** REŠAVANJE KVADRATNE JEDNAČINE ***
10 CLS
20 INPUT X
30 IF X>5 GOTO 70
40 IF X<5 GOTO 90
```

```
50 Y=1
60 GOTO100
70 Y=X*X+1
80 GOTO 100
90 Y=1/(X*X+1)
100 PRINT "ZA X=";X
110 PRINT "Y=";Y
120 END
```

U mnogim programima, gde se komunikacija programera sa računarom vrši preko tastature, koristi se kombinacija IF naredbe sa naredbom KEY kao relacijskim izrazom. U tom slučaju, format naredbe IF mogao bi glasiti:

IF KEY= <brojni izraz1> THEN GOTO <brojni izraz2>

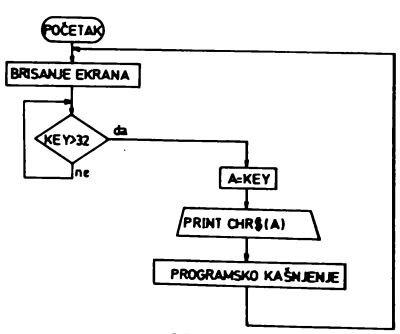
gde je
<brojni izraz1> — decimalni kod nekog od 128 znakova <brojni izraz2> — broj nekog programskog reda u programu.

Kada računar naiđe na programski red sa ovakvom kombinacijom naredbi, on očekuje pritisak one dirke čiji je decimalni kod jednak vrednosti brojnog izraza 1. Ako je taj uslov ispunjen vrši se bezuslovni skok na programski red čiji je broj određen brojnim izrazom 2. Ukoliko odgovarajuća dirka nije pritisnuta nastavlje se izvršenje programskog reda koji sledi.

U primeru koji sledi ilustrovaćemo primenu kombinacije naredbi IF i KEY.

Primer: Nacrtati algoritamsku šemu i napisati program koji će prikazivati na ekranu poruku o pritisnutoj dirci sa tastature.

Algoritamska šema za ovaj primer je data na sl.2.9.



Sl.2.9

Program za datu algoritamsku šemu je sledeći:

```
10 CLS
20 IF KEY>32 GOTO 40
30 GOTO 20
40 A=KEY
50 PRINT "PRITISNULI STE DIRKU KOJA GENERIŠE"
60 PRINT "ZNAK"; CHR$(A)
70 WAIT(300)
80 GOTO 10
```

2.17. PROGRAMSKI CIKLUSI

koje čine programski ciklus mora postojati barem jedna naredba koja omogućava izlazak iz ciklusa. Ukoliko takva naredba ne postoji, takav ciklus nazivamo **beskonačnim programskim ciklusom.**

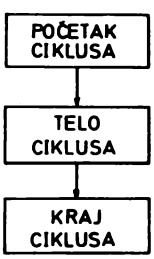
Ako je izlaz iz ciklusa uslovljen brojem ponavljanja ciklusa, tada se ciklus zove **brojački programski ciklus.**

Programski ciklusi mogu slediti jedan iza drugog u programu i takav program nazivamo **linijski formiran ciklus.** Ako se programski ciklusi nalaze jedan unutar drugog tada imamo **koncentrično formiran ciklus.**

2.17.1. Naredbe programskog ciklusa

U programiranju se najčešće koriste brojački programski ciklusi. BASIC poseduje skup naredbi koje omogućavaju lako programiranje ovakvih programskih ciklusa.

Jedan programski ciklus možemo predstaviti sledećim algoritmom (sl.8.10).



Sl.2.10

Telo ciklusa predstavlja skup naredbi BASIC-a, kojima se izvršavaju obrade podataka u okviru jednog koraka programskog ciklusa.

Sada ćemo detaljnije obraditi naredbe početka i kraja programskog ciklusa.

Naredba početka programskog ciklusa

Svaki programski ciklus počinje naredbom

FOR<kontrolna promenljiva> = <početna vrednost> TO

<krajnja vrednost> STEP<priraštaj>

Ova naredba kontrolnoj promenljivoj dodeljuje vrednost početne vrednosti. Program nastavlja sa izvršavanjem sve dok ne naiđe na naredbu NEXT. U tom momentu vrednost promenljive se uvećava za iznos koji se definiše priraštajem i koji može biti ili pozitivan ili negativan. Krajnja vrednost promenljive se onda procenjuje i upoređuje sa znakom priraštaja. Pretpostavlja se da nula ima pozitivan znak. Ako ne dođe do poklapanja znaka izvršenje se preuzima pri prvoj naredbi posle poslednje naredbe FOR i nastavlje se kroz petlju. U bilo kom momentu za vreme petlje korisnik može da modifikuje vrednost kontrolne promenljive nekim raspoloživim sredstvom (na primer, naredbom LET).

Početna, krajnja vrednost i priraštaj mogu biti pozitivni ili negativni brojevi, celi ili brojevi sa pokretnim zarezom. Ako se u naredbi izostavi STEP i priraštaj, računar pretpostavlja da je veličina priraštaja 1.

Naredba kraja programskog ciklusa

Svaki programski ciklus završava se naredbom kraja programskog ciklusa, koja ima format:

NEXT

NEXT <kontrolna promenljiva>

Ova naredba zatvara programski ciklus kao što je opisano u naredbi FOR. Ako se izostavi naziv kontrolne promenljive, naredba NEXT vraća upravljanje na poslednju naredbu FOR i proces se nastavlja. Ako je dat naziv kontrolne promenljive, BASIC će izvršiti proveru da ustanovi da li on odgovara nazivu kontrolne promenljive upotrebljene u poslednjoj naredbi FOR. Ako ne odgovara, izdaje se na ekranu poruka o greški.

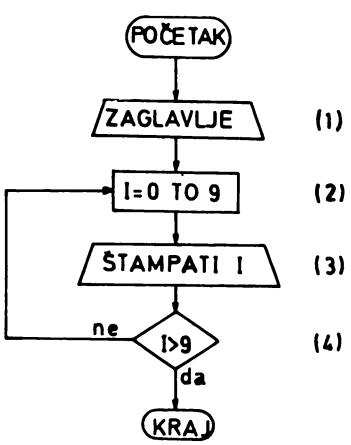
Formiranje FOR/NEXT petlje

Programski ciklus drugačije nazivamo FOR/NEXT petljom. Ako su poznati početna, krajnja vrednost i priraštaj, može se izračunati koliko puta se obavljaju naredbe iz tela programske petlje, primenom sledeće formule:

$$n = \text{INT}\ \frac{\text{poslednja vrednost-početna vrednost}}{\text{priraštaj}} + 1$$

Primer: Napisati program za štampanje na ekranu sa svim jednocifrenim brojevima.

Algoritamska šema je data na sl.2.11.

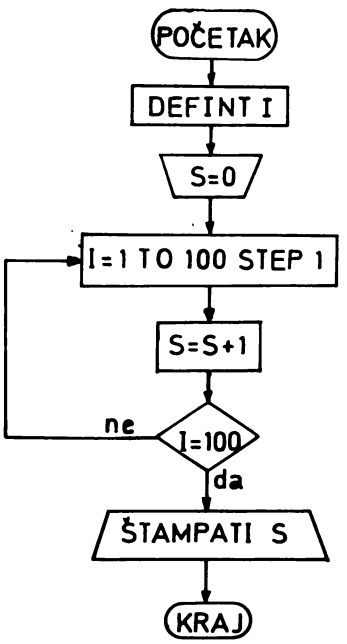


Sl.2.11

Program ya datu algoritamsku šemu je sledeći:

```
10 PRINT "SVI JEDNOCIFRENI BROJEVI SU:"
20 FOR I=0 TO 9
30 PRINT I
40 NEXT I
50 END
```

Primer: Napisati program za sabiranje prvih 100 celih pozitivnih brojeva. Algoritamska šema je data na sl.2.12.



Sl.2.12

Program za datu algoritamsku šemu je sledeći:

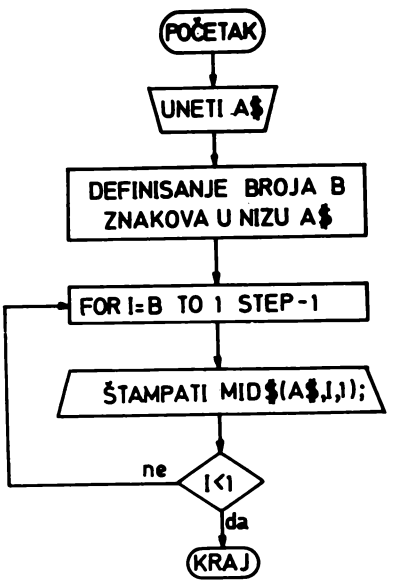
```
10 DEFINT I
20 S=0
30 FOR I=1 TO 100 STEP 1
40 S=S+1
50 NEXT I
60 PRINT "SUMA PRVIH 100 CELIH BROJEVA JE";S
70 END
```

Naredbom DEFINT u programskom redu 10 definišemo promenljivu u FOR/NEXT petlji kao celobrojnu. Naredba FOR u redu 30 definiše početnu i krajnju vrednost, kao i priraštaj promenljive. U toku svakog izvršavanja ciklusa FOR/NEXT petlje vrednost sume S se uvećava za trenutnu vrednost promenljive I. Posle izlaska iz petlje računar na ekranu štampa rezultat programa:

```
SUMA PRVIH 100 PRIRODNIH BROJEVA JE 5050.
```

Primer: Sastaviti program koji za uneti tekst štampa na ekranu tekst koji se dobija kada se uneti tekst pročitą sdesna nalevo.

Algoritamska šema je data na sl.2.13.



Sl.2.13

Program po datoj algoritamskoj šemi je sledeći:

```
10 INPUT A$
20 B=LEN(A$)
30 FOR I=B TO 1 STEP -1
40 PRINT MID$(A$,I,1)
50 NEXT I
```

Ako je proizvoljno uneta reč BEOGRAD, računar će posle izvršenja programa na ekranu prikazati tekst

```
DARGOEB
```

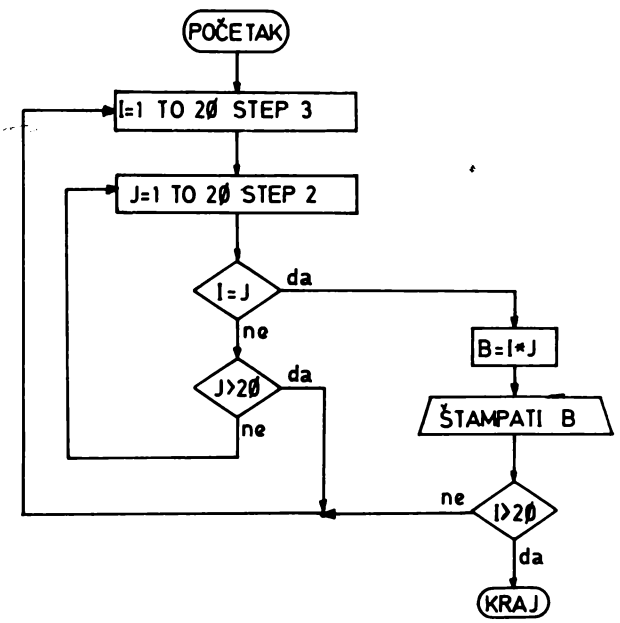
Naredba za prevremeni izlazak iz FOR/NEXT petlje

EXIT <brojni izraz>

Ova naredba je u stvari bezuslovno grananje na broj programskog reda definisanog brojnim izrazom. Namenjena je za prevremeni izlazak iz FOR/NEXT petlje. Treba obratiti pažnju na jednu meru predostrožnosti. Ako imamo slučaj korišćenja FOR/NEXT petlje u petlji, naredba EXIT služi da prenese upravljanje na broj reda koji je za jedan nivo niži od povezivanja. Ova naredba se koristi umesto standardne GOTO naredbe.

Primer: Nacrtati algoritamsku šemu i napisati program koji iz dva skupa brojeva množi iste brojeve.

Algoritamska šema je data na sl.2.14.



Sl.2.14

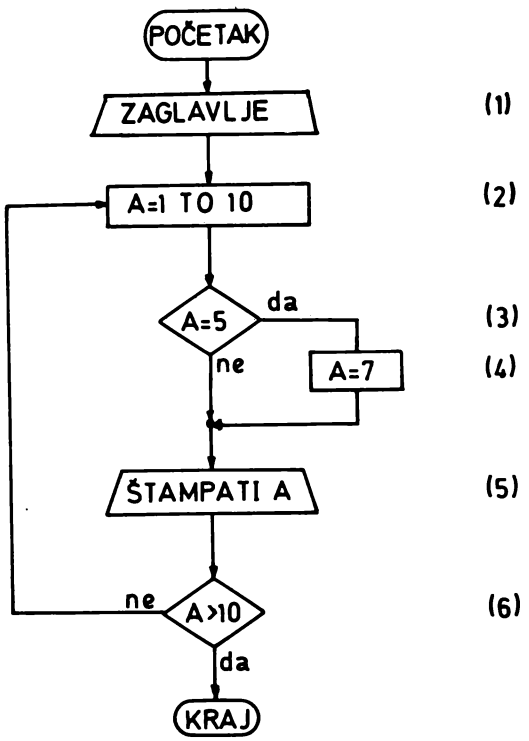
Program za dátu algoritamsku šemu je sledeći:

```
10 FOR I=1 TO 20 STEP 3
20 FOR J=1 TO 20 STEP 2
30 IF I=J EXIT 50
40 NEXT J
50 B=I*J
60 PRINT B
70 NEXT I
80 END
```

Izmena trenutne vrednosti promenljive u FOR/NEXT petlji

U jednoj FOR/NEXT petlji vrednost promenljive se menja sa zadatim prirastajem. Pomoću naredbe LET može se u nekom koraku promeniti vrednost promenljive u neku drugu vrednost. Ako želimo da se program i dalje izvršava u FOR/NEXT petlji, tada dodeljena vrednost promenljivoj mora biti u granicama početne i krajnje vrednosti zadate u naredbi FOR. Ako to nije slučaj program će odmah u tom istom koraku izaći iz petlje. Ovakvu mogućnost ilustrovaćemo primerom.

Primer: Štampati na ekranu sve cele brojeve od 1 do 4 i od 7 do 10.
Algoritamska šema je data na sl.2.15.



Sl.2.15

Da bi izvršio ovaj program računar treba da obavi sledeće aktivnosti, označene po koracima:

(1) — štampa na ekranu poruku "TRAŽENI BROJEVI SU:"
(2) — postavlja ulaznu naredbu FOR/NEXT petlje
(3) — ispituje da li je promenljiva A dobila vrednost 5. Ako nije izvršava se korak (5), a ako jeste onda se obavlja korak (4).
(4) — promenljivoj A dodeljuje novu vrednost
(5) — štampa na ekranu trenutnu vrednost promenljive A
(6) — ispituje uslov kraja FOR/NEXT petlje

Program po datoj algoritamskoj šemi biće:

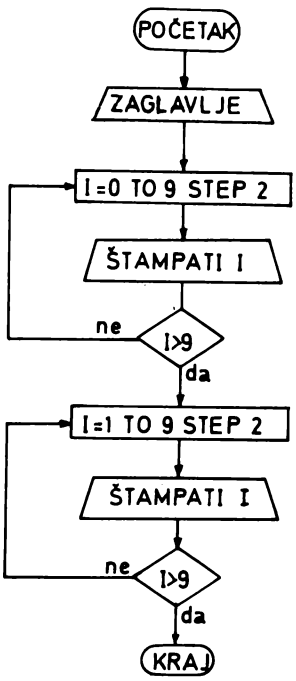
```
10 PRINT "TRAŽENI BROJEVI SU:"
20 FOR A=1 TO 10
30 IF A=5 LET A=7
40 PRINT A
50 NEXT A
60 END
```

2.17.2. Linijska struktura FOR/NEXT petlje

U složenijim programima javlja se često više FOR/NEXT petlji.

Primer: Štampati na ekranu sve parne a u nastavku sve neparne jednocifrene brojeve.

Algoritamska šema je data na sl.2.16.



Sl.2.16

U ovom primeru ilustrovana je primena algoritamske strukture sa dve FOR/NEXT petlje. Prvom FOR/NEXT petljom se štampaju svi parni, a drugom FOR/NEXT petljom svi neparni jednocifreni brojevi.

Redosled algoritamskih koraka odgovara redosledu programskih redova u programu:

```
10 PRINT "PARNI JEDNOCIFRENI BROJEVI SU:"
20 FOR I=0 TO 9 STEP 2
30 PRINT I
40 NEXT I
50 PRINT "NEPARNI JEDNOCIFRENI BROJEVI SU:"
60 FOR I=1 TO 9 STEP 2
70 PRINT I
80 NEXT I
90 END
```

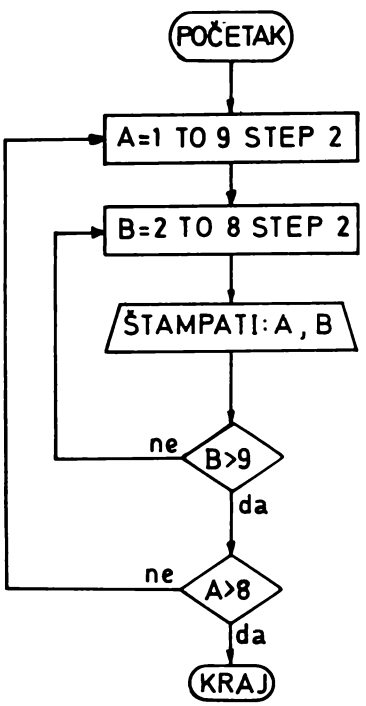
2.17.3. Koncentrične FOR/NEXT petlje

Osnovno pravilo za koncentrične FOR/NEXT petlje je da jedna petlja mora započeti i završiti se u okviru druge petlje. Presecanje petlji nije dozvoljeno.

Funkcionisanje dve koncentrične FOR/NEXT petlje analiziraćemo sledećim primerom.

Primer: Napraviti kombinaciju svih neparnih sa svim parnim jednocifrenim brojevima.

Algoritamska šema je data na sl.2.17.



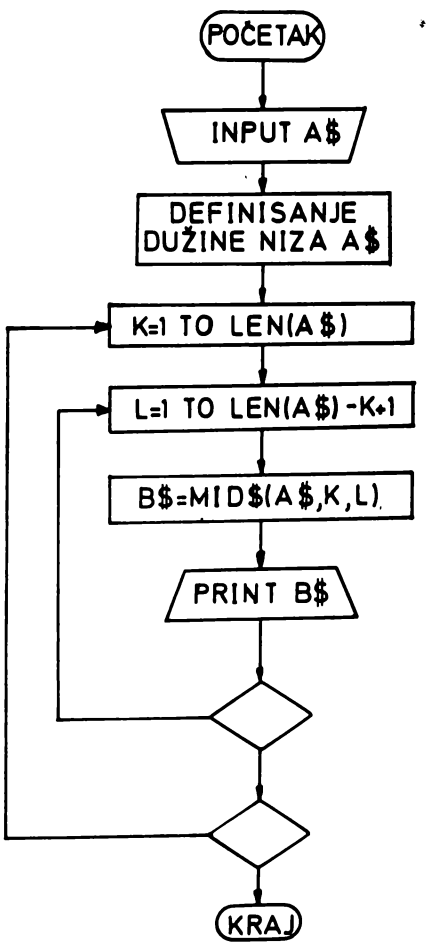
Sl.2.17

U ovom primeru imamo dve FOR/NEXT petlje: petlju A i unutar nje petlju B. Za jednu vrednost promenljive u petlji A obradiće se kombinacije sa svim vrednostima promenljive u petlji B. Kada se ovaj korak završi uvećaće se vrednost promenljive u petlji A za vrednost priraštaja. Program po datoj algoritamskoj šemi biće:

```
10 FOR A=1 TO 9 STEP 2
20 FOR B=2 TO 8 STEP 2
30 PRINT B, PRINT A
40 NEXT B
50 NEXT A
60 END
```

Primer: Nacrtati algoritamsku šemu i napisati program za nalaženje svih međusobno različitih podreči date reči.

Algoritamska šema je prikazana na sl.2.18. Preko tastature unosimo niz čije različite podreči računar treba da pronađe. U narednom koraku definišemo broj znakova unetog niza, korišćenjem naredbe LET. Podreči određujemo pomoću dve FOR/NEXT petlje. Prvo FOR/NEXT petljom definišemo od kog znaka, unetog niza (gledano sleva udesno), počinjemo generisanje podniza. Ako je dužina unetog niza ceo broj, tada se vrednost promenljive u prvoj FOR/NEXT petlji menja od 1 do tog celog broja.



Sl.2.18

Drugom FOR/NEXT petljom definišemo koliko znakova iz unetog niza uzimamo počev od rednog broja znaka niza određenog prvom FOR/NEXT petljom. Vrednost promenljive u drugoj FOR/NEXT petlji zavisi od trenutne vrednosti promenljive u prvoj FOR/NEXT petlji.

U algoritmu smo upotrebili sledeće oznake promenljivih:

A$ — oznaka unetog niza

D — broj znakova unetog niza

K — promenljiva prve FOR/NEXT petlje, koja uzima vrednosti od 1 do D

L — promenljiva druge FOR/NEXT petlje, koja uzima vrednosti od 1 do D—K+1

U okviru FOR/NEXT petlji određujemo a zatim štampamo na ekranu trenutnu vrednost podniza B$.

Program prema datoj algoritamskoj šemi biće sledeći:

```
10 CLS
20 INPUT "REČ JE:"A$
30 D=LEN(A$)
40 FOR K=1 TO D
50 FOR L=1 TO D-K+1
60B$=MID$(A$,K,L)
70 PRINT B$
80 NEXT L
90 NEXT K
```

Proverićemo izvršenje programa na jednom konkretnom primeru. Posle startovanja programa na postavljen znak pitanja REČ JE:? unosimo niz A$. Unošenje niza završavamo pritiskom na dirku RETURN. Nakon toga računar će štampati sve različite podreči niza A$. Ako smo uneli niz "PECOM", računar će štampati sledeće podreči:

```
P       C
PE      CO
PEC     COM
PECO    O
PECOM   OM
E       M
EC
ECO
ECOM
```

2.18. NESTANDARDNE NAREDBE BASIC-A

Standardni BASIC jezik ne sadrži naredbe iz ove grupe, pa su zato i nazvane — nestandardnim. Formate i funkcije ovih naredbi određuje sam proizvođač računara. U grupu nestandardnih naredbi za BASIC PECOM-a 64 spadaju: SCR, TONE, CPOS, COLOR, CHRGEN. Koriste se za definisanje oblika znakova, njihovo pozicioniranje na ekranu, zadavanje boja znakova i pozadine i generisanje tona.

2.18.1. Generisanje boje ekrana

Za generisanje boje ekrana koristi se naredba SCR, čiji je format

SCR<brojni izraz>

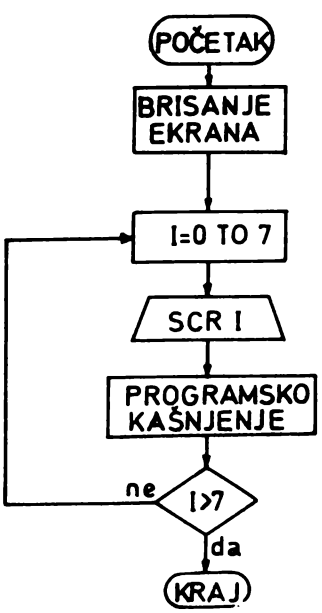
Vrednost brojnog izraza definiše boju ekrana prema sledećoj tabeli:

brojni izraz	boja ekrana
0	crna
1	zelena
2	plava
3	maslinasta
4	crvena
5	žuta
6	ljubičasta
7	bela

Primer: Napisati program koji će prikazivati svaku boju ekrana određeni vremenski interval.

Algoritamska šema je data na sl.2.19.

Na početku programa računar briše sadržaj ekrana. Zatim postavlja parametre



Sl.2.19

FOR/NEXT petlje za promenljivu I. U svakom koraku petlje ekran se boji onom bojom koju definiše promenljiva I. Programskim kašnjenjem zadržava se boja ekrana određeni vremenski interval.

Program za datu algoritamsku šemu je sledeći:

```
10 CLS
20 FOR I=0 TO 7
30 SCR I
40 WAIT (300)
50 NEXT I
60 END
```

2.18.2. Generisanje tona

Za generisanje zvučnih efekata BASIC PECOM-a 64 koristi naredbu TONE čiji je format:

TONE (<brojni izraz1>, <brojni izraz2>,<brojni izraz3>)

Naredba TONE daje kontinualni ton pri čemu:

brojni izraz1 — određuje učestanost i može da uzima vrednosti od 0 do 127

brojni izraz 2 — vrši izbor jedne od 8 oktava i uzima vrednosti od 0 do 7

brojni izraz 3 — određuje amplitudu (jačinu) tona i može da uzima vrednosti od 0 do 15

Korišćenjem naredbi TONE i WAIT u programu, mogu se dobiti različite melodije. Na primer, generisanje tona C u lestvici vrši se sledećim programom:

```
10 TONE (119,4,8)
20 WAIT(32)
30 TONE(0,0,0)
40 END
```

U programskom redu 10 generiše se ton naredbom TONE. Naredbom WAIT određuje se trajanje tona, a naredbom TONE (O,O,O) ukida se generisani ton.

Naredba

TONE (<brojni izraz1>,<brojni izraz2>)

je mala modifikacija prethodne naredbe. Ona se koristi za generisanje belog šuma, kojim se mogu simulirati razni pucnji i sl. U ovoj naredbi:

brojni izraz 1 — određuje frekventni opseg belog šuma i može da varira od 1 do 8

brojni izraz 2 — određuje amplitudu i može da varira od 0 do 15.

Nakon unošenja naredbe

```
TONE(1,1)
```

i pritiska na dirku RETURN čuće se efekat belog šuma. Naredbom

```
TONE (1,2)
```

čuće se pojačanje šuma (ili amplitude).

Za ukidanje šuma unosi se:

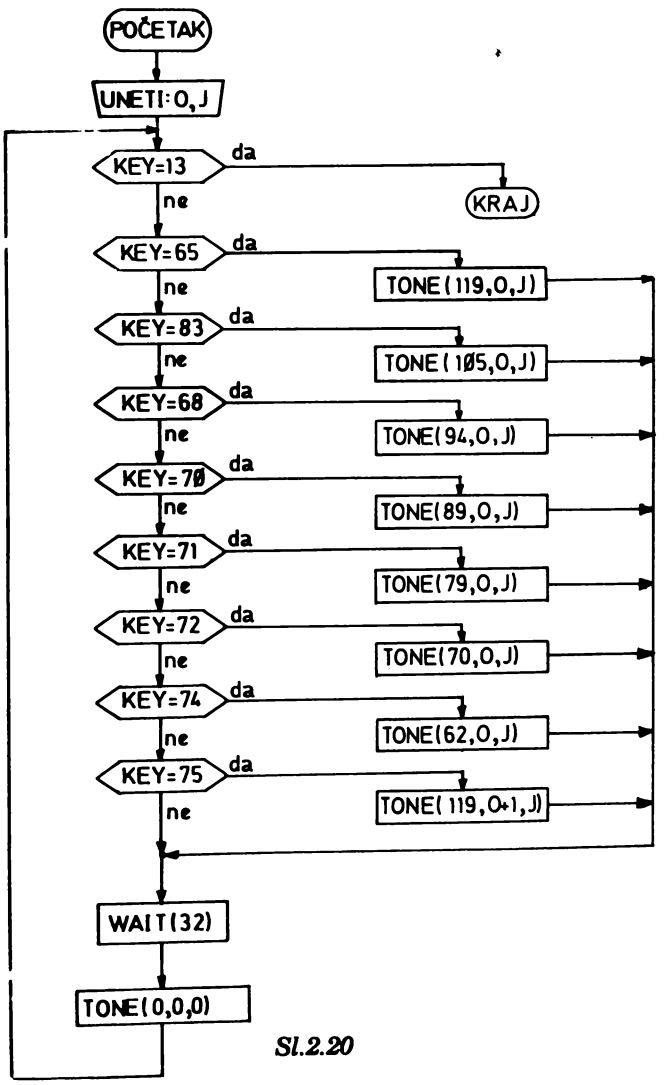
```
TONE(0,0)
```

Za komponovanje melodija potrebno je znati učestanosti tonova u osnovnoj tonskoj skali. U tabeli su date vrednosti učestanosti (brojnog izraza1 u naredbi TONE) za osnovne tonove.

ton	C	D	E	F	G	A	H	C
učestanost	119	105	94	89	79	7	0	62	119

Primer: Napisati program za komponovanje muzike. Dirke:A,S,D,F,G,H,J,K daju redom tonove C,D,E,F,G,A,H,C. Oktavu i jačinu izabrati sam. Učestanosti tonova uzeti iz prethodne tabele. Komponovanje prekinuti pritiskom na dirku RETURN.

Algoritamska šema je data na sl.2.20.



Sl.2.20

Program za datu algoritamsku šemu je sledeći:

```
10 REM *** KOMPONOVANJE MUZIKE ***
20 SCR 2
30 PRINT"DIRKE A S D F G H J K"
40 PRINT"DAJU TONOVE C D E F G A H C"
50 PRINT"AKO NE ŽELITE VIŠE DA KOMPONUJETE PRITISNITE DIRKU RETURN"
60 PRINT"IZABERITE JEDNU OD 8 OKTAVA":INPUT O
70 PRINT"IZABERITE JAČINU OD 0 DO 15": INPUT J
71 PRINT"POČNITE SA KOMPONOVANJEM"
75 IF KEY=13 GOTO 270
80 IF KEY=65 GOTO 170
90 IF KEY=83 GOTO 180
100 IF KEY=68 GOTO 190
110 IF KEY=70 GOTO 200
120 IF KEY=71 GOTO 210
130 IF KEY=72 GOTO 220
140 IF KEY=74 GOTO 230
150 IF KEY=75 GOTO 240
160 GOTO 75
170 TONE (119,0,J):GOTO 250
180 TONE(105,0,J):GOTO 250
190 TONE(94,0,J):GOTO 250
200 TONE(89,0,J):GOTO 250
210 TONE(79,0,J):GOTO 250
220 TONE(70,0,J):GOTO 250
230 TONE(62,0,J):GOTO 250
240 TONE(11:,0+1,J): GOTO 250
250 WAIT(32)
260 TONE(0,0,0)
265 GOTO 75
270 END
```

2.18.3. Pozicioniranje znaka na ekranu

U toku pisanja programa, koji kao rezultat treba da da određen grafički prikaz na ekranu, programeru je od velike koristi naredba CPOS, čiji je format:

CPOS(<brojni izraz1>,<brojni izraz2>,<brojni izraz3>)

Ovom naredbom vrši se pozicioniranje određenog znaka na ekranu, pri čemu je:
brojni izraz1 — redni broj kolone na ekranu, i kreće se od 0 do 39
brojni izraz2 — redni broj vrste na ekranu, i kreće se od 0 do 23
brojni izraz3 — decimalni kod znaka koji se pozicionira (uzima vrednost od 0 do 127).

Programom

```
10 CLS
20CPOS(19,12,67)
```

pozicioniraće se slovo C na sredini ekrana.

Naredbom

CPOS(<brojni izraz1>,<brojni izraz2>)

vrši se pozicioniranje kursora na ekranu.

Naredba CPOS može da se upotrebi zajedno sa naredbom PRINT za prikazivanje poruka i zaglavlja, počev od definisanog položaja na ekranu, ili sa naredbom CHRGEN za kreiranje grafike u boji. Na primer, programom

```
10 CPOS(14,14)
20 PRINT "OVO JE TEST"
```

prikazaće se poruka OVO JE TEST na sredini ekrana.

Naredba CPOS omogućava korisniku lakše i efikasnije generisanje grafike na ekranu. Navešćemo neke elementarne primere koji, kada se kombinuju, mogu dati dosta dobru grafiku.

Primer: Za crtanje 10 znakova po vertikali koristimo sledeći program:

```
10 CLS
20 FOR I=5 TO 14
30 CPOS(20,I,7)
40 NEXT I
50 END
```

Naredbom CPOS pozicioniramo karakter sa decimalnim kodom 7 (vertikalna crta). Redni broj kolone je konstantan, dok je redni broj vrste sa promenljivom I. FOR/NEXT petljom se postiže promena vrednosti promenljive I do 5 do 14.

Posle izvršenja ovog programa, računar će na ekranu iscrtati vertikalnu liniju dužine 10 karaktera.

Primer: Programom koji sledi postižemo kretanje određene poruke na ekranu.

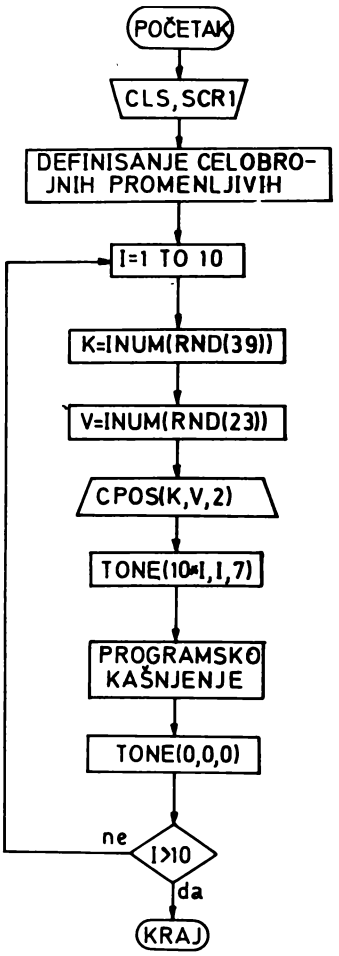
```
10 CLS
20 FOR I=0 TO 23
30 CPOS(0+I,12):PRINT" KREĆE SE UDESNO "
40 NEXT I
45 CLS
50 FOR I=24 TO 1 STEP -1
60 CPOS(I,12):PRINT"KREĆE SE ULEVO"
70 NEXT I
80 GOTO 10
```

Prvom FOR/NEXT petljom menja se vrednost kolone od koje počinje štampanje poruke "KREĆE SE UDESNO". Kada se tekst ispiše u krajnjem desnom položaju na ekranu, počinje izvršenje druge FOR/NEXT petlje. Ona omogućava menjanje vrednosti kolone od koje počinje štampanje poruke "KREĆE SE ULEVO". Kada se tekst ispiše na ekranu u krajnjem levom položaju, prelazi se na izvršenje prve FOR/NEXT petlje (dejstvo naredbe GOTO 20 u programskom redu 80).

Primer: Nacrtati algoritamsku šemu i napisati program koji štampa na ekranu 10 znakova sa decimalnim kodom 2 na slučajno odabranim pozicijama.

Algoritamska šema je data na sl.2.21.

Program za datu algoritamsku šemu je sledeći:

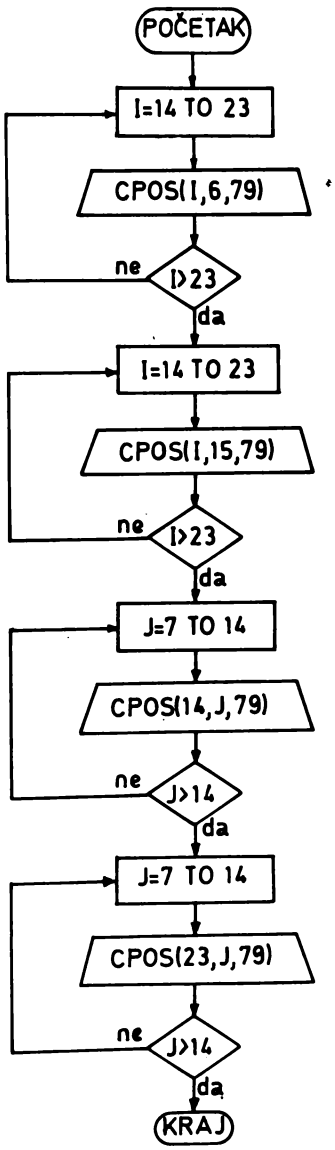


Sl.2.21

```
10 CLS
20 SCR1
30 DEFINT Z
40 FOR I=1 TO 10
50K=INUM(RND(39))
60V-INUM(RND(23))
70CPOS(K,V,2)
80TONE(I*10,I,7)
90 WAIT(50)
100TONE(0,0,0)
110 NEXT I
120 END
```

Primer: Nacrtati algoritamsku šemu i napisati program za crtanje kvadrata sa 10X10 znakova na sredini ekrana. Za crtanje koristiti slovo O čiji je decimalni kod 79.

Ovaj primer najjednostavnije možemo rešiti crtanjem svake stranice kvadrata pojedinačno. U tom slučaju upotrebićemo četiri FOR/NEXT petlje (po jednu petlju za crtanje svake stranice). Algoritamska šema za ovakav način rešavanja zadatka data je na sl.2.22.



Sl.2.22

Za crtanje znakova na ekranu koristimo naredbu CPOS. U prve dve FOR/NEXT petlje broj vrste je konstantan, a broj kolone se menja za vrednosti promenljive I u FOR/NEXT petlji. Za svaku vrednost promenljive I na ekranu se crta po jedan znak (slovo O sa decimalnim kodom 79). Na taj način se dobijaju dve horizontalne stranice kvadrata. Istim postupkom se crtaju i vertikalne stranice kvadrata. Razlika je u tome što je sada u CPOS naredbi broj kolone konstantan a broj vrste se menja kao promenljiva J u trećoj i četvrtoj FOR/NEXT petlji.

Prema datoj algoritamskoj šemi program je sledeći:

```
10 CLS
20 FOR I=14 TO 23
30CPOS(I,6,79)
40 NEXT I
50 FOR I=14 TO 23
60CPOS(I,15,79)
70 NEXT I
80 FOR J=7 TO 14
90CPOS(14,J,79)
100 NEXT J
110 FOR J=7 TO 14
120CPOS(23,J,79)
130 NEXT J
140 END
```

Ako bolje analiziramo prethodni algoritam, možemo uvideti da prve dve FOR/NEXT petlje možemo podvesti pod jednu. Isti slučaj je i sa trećom i četvrtom FOR/NEXT petljom. U ovom slučaju program bi bio sledeći:

```
10 CLS
20 FOR I=14 TO 23
30CPOS(I,6,79)
40CPOS(i,15,79)
50 NEXT I
60 FOR J=7 TO 14
70CPOS(14,J,79)
80CPOS(23,J,79)
90 NEXT J
100 END
```

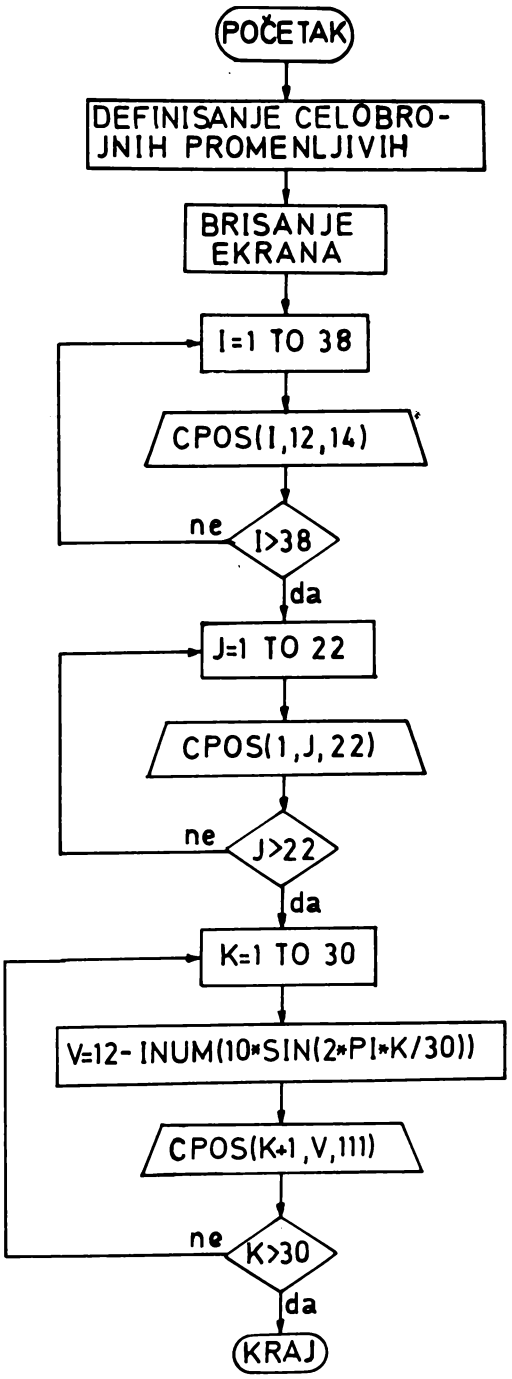
Prednosti ovako rešenog zadatog primera su sledeće:

— program je kraći a samim tim i jednostavniji
— zauzima manji memorijski prostor
— brzina izvršavanja programa je znatno veća.

Primer: Nacrtati algoritamsku šemu i napisati program za crtanje funkcije SIN.

Algoritamska šema je data na sl.2.23. Na početku programa sve promenljive u programu definišu se kao celobrojne. To činimo iz razloga što se kolone i vrste, u kojima se vrši prikazivanje znakova na ekranu, definišu vrednostima promenljivih iz programa.

Posle brisanja ekrana, prvom FOR/NEXT petljom crtamo X-osu. To činimo znakom čiji je decimalni kod 14, koji se štampa na pozicijama sa konstantnom vrednošću vrste (12), a vrednost kolone se menja od 1 do 38.


POČETAK
DEFINISANJE CELOBRO-
JNIH PROMENLJIVIH
BRISANJE
EKRANA
I=1 TO 38
CPOS(I,12,14)
ne
I>38
da
J=1 TO 22
CPOS(1,J,22)
ne
J>22
da
K=1 TO 30
V=12- INUM(10*SIN(2*PI*K/30))
CPOS(K+1,V,111)
ne
K>30
da
KRAJ

Sl.2.23

Drugom FOR/NEXT petljom se crta Y-osa, znakom čiji je decimalni kod 22. Znak se štampa na pozicijama koje imaju konstantnu vrednost kolone (1), a vrednost vrste se menja od 1 do 22.

Treća FOR/NEXT petlja definiše broj tačaka u kojima se izračunava vrednost funkcije SIN. U ovom primeru, vrednost funkcije se izračunava u 30 tačaka. Funkcijom

```
V=12-INUM(10*SIN(2*PI*K/30))
```

definiše se vrednost vrste u kojoj se crta znak sa decimalnim kodom 111 (malo slovo) za datu vrednost kolone K, definisanom FOR/NEXT petljom.

Program za datu algoritamsku šemu je sledeći:

```
10 DEFINT Z
20 CLS
30 FOR I=0 TO 38
40CPOS(I,12,14)
50 NEXT I
60 FOR J=1 TO 22
70CPOS(0,J,22)
80 NEXT J
90 FOR K=0 TO 30
100V=12-INUM(10*SIN(2*PI*K/30))
110CPOS(K,V,111)
120 NEXT K
130 END
```

Primer: Za crtanje funkcije COS može se upotrebiti isti program, kao u primeru 18.3.7., s tom razlikom što bi programski red 100 bio sledeći:

```
V=12-INUM(10*COS(2*PI*K/30))
```

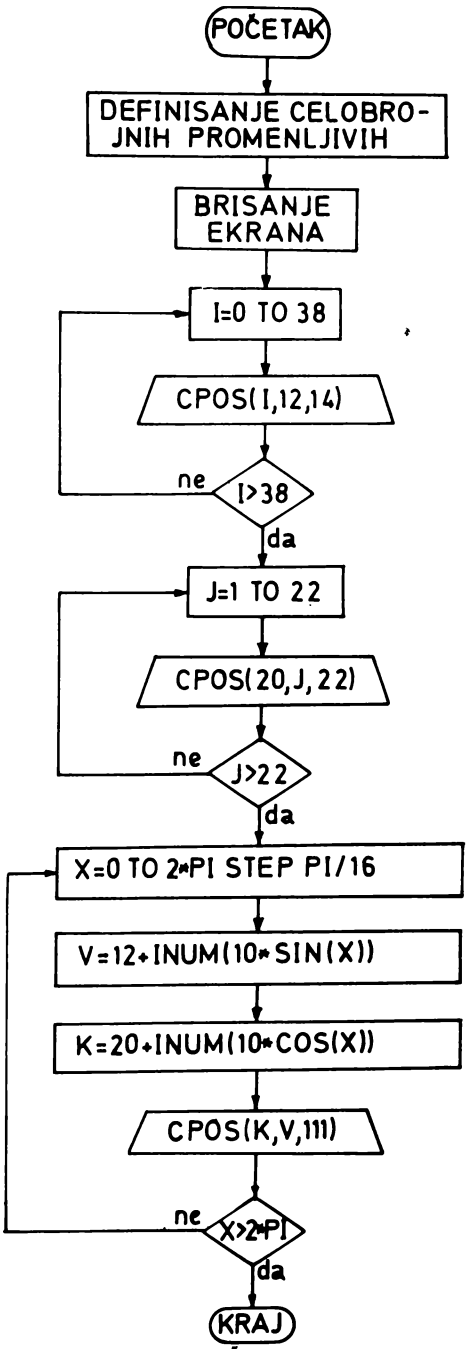
Primer: Nacrtati algoritamsku šemu i napisati program za crtanje kruga na ekranu.

Algoritamska šema je data na sl.2.24.

Na početku programa definišemo kao celobrojne one promenljive, koje koristimo u programu. Jedino je promenljiva X u pokretnom zarezu. Posle brisanja ekrana crtaju se koordinate X i Y pomoću dve FOR/NEXT petlje. Trećom FOR/NEXT petljom se definišu koordinate za crtanje znaka sa decimalnim kodom 111 (malo slovo), a koje su određene vrednostima funkcije kruga.

Program za datu algoritamsku šemu je sledeći:

```
10 DEFINT V
20 CLS
30 FOR I=0 TO 38
40CPOS(I,12,14)
50 NEXT I
60 FOR J=1 TO 22
70CPOS(20,J,22)
80 NEXT J
90 FOR X=0 TO 2*PI STEP PI/16
100V=12+INUM(10*SIN(X))
110K= 20+INUM(10*COS(X))
120CPOS(K,V,111)
130 NEXT X
140 END
```


POČETAK
DEFINISANJE CELOBRO-
JNIH PROMENLJIVIH
BRISANJE
EKRANA
I=0 TO 38
CPOS(I,12,14)
ne
I>38
da
J=1 TO 22
CPOS(20,J,22)
ne
J>22
da
X=0 TO 2*PI STEP PI/16
V=12+INUM(10*SIN(X))
K=20+INUM(10*COS(X))
CPOS(K,V,111)
ne
X>2*PI
da
KRAJ

Sl.2.24

2.18.4. Generisanje boje znakova

Kada se uključi PECOM 64, njegov karakter generator prikazuje znakove na ekranu u crnoj boji. Za bojenje tih istih karaktera u nekoj od 7 boja BASIC poseduje naredbu COLOR, čiji je format:

COLOR(<brojni izraz1>,<brojni izraz2>,<brojni izraz3>)

gde je:
brojni izraz1 — decimalni kod znaka od koga se počinje sa bojenjem
brojni izraz2 — decimalni kod znaka sa kojim se završava sa bojenjem
brojni izraz3 — kod boje kojom se boje znaci i može imati vrednost od 0 do 15

Tabela koja sledi prikazuje vrednosti za brojni izraz 3 i odgovarajuće boje pri izvršenju naredbe COLOR.

brojni izraz 3	**boja znaka**
0 ili 4 ili 12	crna
1	crvena
2 ili 9	plava
3	ljubičasta
6 ili 10	zelena
7	žuta
11	maslinasta

Primer:

```
10CPOS(20,12,1)
20COLOR(1,1,2)
30 END
```

U navedenom primeru znak sa decimalnim kodom 1 postavlja se u kolonu 20 i vrstu 12 na ekranu. Naredbom COLOR ovaj znak se boji u plavu boju.

Primer: Štampati sva velika slova na sredini ekrana u crvenoj, plavoj i ljubičastoj boji.

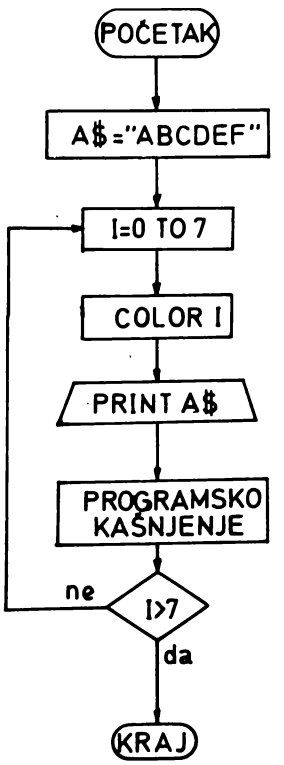
Program je sledeći:

```
10 CLS
20COLOR(65,74,1)
30COLOR(75,84,2)
40COLOR(85,94,3)
50CPOS(4,12)
60PRINT"ABCDEFGHIJKLMNOPQRSTUVWXYZŠĆČ"
70 END
```

Primer: Nacrtati algoritamsku šemu i napisati program koji omogućava praćenje na ekranu promene boja znakova datog niza.

U ovom primeru vrednost brojnog izraza3 u naredbi COLOR je promenljiva, čija se vrednost definiše FOR/NEXT petljom.

Algoritamska šema je data na sl.2.25.



Sl.2.25

Program za datu algoritamsku šemu je sledeći:

```
10 CLS
20 A$="ABCDEF"
30 FOR I=0 TO 7 STEP 1
40COLOR(65,91,I)
50 CPOS(17,12):PRINT A$
60 WAIT(400)
70 NEXT I
80 END
```

2.18.5. Generisanje novih karaktera

Prilikom crtanja složenije grafike, skup raspoloživih specijalnih znakova na PECOM-u 64 nije dovoljan da zadovolji potrebe. Naredba CHRGEN omogućava programeru da programski generiše nove karaktere. Format ove naredbe je

CHRGEN(<brojni izraz>, "18 heksadecimalnih cifara")

Nakon izvršenja naredbe CHRGEN, automatski se vrši promena sadržaja memorije karakter generatora onog znaka koji je pregenerisan.

Brojni izraz je ceo broj koji je u stvari decimalni kod znaka, i može da uzima vrednosti od 0 do 127. Ukoliko znak koji se generiše ima isti decimalni kod kao i znak koji je prikazan na ekranu pre izvršenja naredbe CHRGEN, moći ćemo, nakon izvršenja naredbe CHRGEN, na ekranu pratiti efekat ove naredbe.

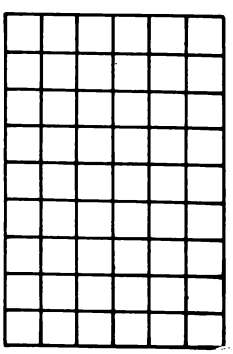
Znak koji se generiše naredbom CHRGEN, na ekranu se prikazuje tačkastom matricom 6X9 (šest kolona i devet vrsta tačaka). 18 heksadecimalnih brojeva, u formatu naredbe CHRGEN, smeštenih pod znacima navoda predstavljaju 9 parova heksadecimalnih brojeva vrsta znaka koji generišemo.

Način na koji se generiše znak naredbom CHRGEN najbolje je objasniti na konkretnom primeru. Najpre treba postaviti konkretne zahteve:

— koji decimalni kod treba da ima znak koji se generiše
— u kojoj od četiri boje (crna, crvena, plava i ljubičasta) generisati znak
— oblik znaka koji se generiše.

Definišimo sada konkretne zahteve:

— novogenerisani znak treba da ima decimalni kod 65 (kod koji u normalnom režimu rada ima slovo A)
— generisati znak u crvenoj boji
— oblik znaka je sledeći:



Matrici 6X9 docrtaćemo sleve strane još dve kolone u koje smeštamo po dva bita za boju svake vrste. Bitove za boju biramo iz sledeće tabele:

bitovi za boju	**boja znaka**
0 0	crna
0 1	crvena
1 0	plava
1 1	ljubičasta

U kasnijim primerima videćemo da jedan znak može sadržati četiri boje.

Nacrtajmo sada proširenu matricu 8X9

8	4	2	1	8	4	2	1
0	1	0	0	1	1	0	
0	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	0	1	1	0	0

U dve krajnje leve kolone upisujemo dva bita za boju znaka. Mi smo zahtevali da je znak generisan u zelenoj boji, a iz gornje tabele vidimo da su bitovi za crvenu boju 0 i 1.

U zatamnjene kvadratiće znaka upisujemo 1, a u preostale upisujemo 0. Matricu dimenzije 8X9 podelimo na dve matrice dimenzija 4X9 i svakoj koloni dodelimo odgovarajući težinski kod.

Sada izračunavamo vrednosti leve i desne polovine vrsta. Svaka polovina vrste daje jedan heksadecimalni broj, a cela vrsta formiraće par heksadecimalnih cifara. Kako imamo 9 vrsta, imaćemo 9 parova heksadecimalnih cifara, koji se smeštaju u formatu naredbe CHRGEN pod znacima navoda.

Pošto svaki kvadratić u jednoj od polovina vrste ima svoju težinu (1,2,4 ili 8), sabraćemo one težine kvadratića u koje je upisana 1. Dobićemo decimalne vrednosti koje treba pretvoriti u heksadecimalne.

vrsta	vrednost levog dela vrste	vrednost desnog dela vrste
1	4	$12_D = C_H$
2	7	$15_D = F_H$
3	7	$15_D = F_H$
4	4	$12_D = C_H$
5	4	$12_D = C_H$
6	4	$12_D = C_H$
7	7	$15_D = F_H$
8	7	$15_D = F_H$
9	4	$12_D = C_H$

Vrednosti vrsta pišemo jednu do druge počev od najviše vrste:

```
4C7F7F4C4C4C7F7F4C
```

što čini 9 parova heksadecimalnih cifara koje smeštamo u format naredbe pod znacima navoda.

Naredba CHRGEN tada dobija oblik:

```
CHRGEN(65,"4C7F7F4C4C4C7F7F4C")
```

Dejstvo naredbe CHRGEN proverićemo konkretnim programom:

```
10 CLS
20PRIN"ABABABABA"
30 WAIT(300)
40 CHRGEN (65,"4C7F7F4C4C4C7F7F4C")
```

Posle startovanja programa na ekranu možemo pratiti sledeće:

— ekran se briše (dejstvo naredbe CLS)
— odštampaće se niz ABABABABA
— posle kraće pauze (dejstvo naredbe WAIT) pregenerisaće se slovo A u novi znak koji smo generisali

Pored standardnog formata naredbe CHRGEN može se koristiti i sledeći format:

CHRGEN(<brojna promenljiva>, <ime niza>)

gde:
brojna promenljiva — predstavlja promenljivu kojom se definiše decimalni kod znaka koji pregenerišemo CHRGEN naredbom
ime niza — predstavlja naziv brojnog niza od 18 heksadecimalnih cifara koji definiše 9 vrsta znaka koji se generiše.

Format naredbe CHRGEN možemo napisati u obliku:

CHRGEN(I,A$)

gde je:

```
I - brojna promenljiva
A$ - ime niza
```

Funkcionisanje ovakvog oblika naredbe CHRGEN ilustrovaćemo sledećim primerom.

Primer: Nacrtati algoritamsku šemu i napisati program za generisanje punog znaka u crnoj boji sa decimalnim kodovima 65 do 69.

Za generisanje znaka koristimo sledeću matricu:

0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1

Pošto novogenerisani znak treba da je u crnoj boji, u kolone koje definišu bitove za određivanje boje upisujemo vrednost O. Za generisanje punog znaka u svim kolonama, koje definišu bitove za oblik znaka upisujemo 1.

Iz matrice možemo videti da su heksadecimalne vrednosti svih vrsta u matrici znak 3F.

Definišimo parametre koje ćemo koristiti u programu. Neka vrednost promenljive I određuje decimalni kod znaka koji zamenjujemo novim znakom. A$ neka predstavlja niz od 18 heksadecimalnih cifara za definisanje boje i oblika znaka. Na osnovu zahteva u zadatku, promenljiva I uzima vrednosti od 65 do 69, a nizu A$ se dodeljuje:

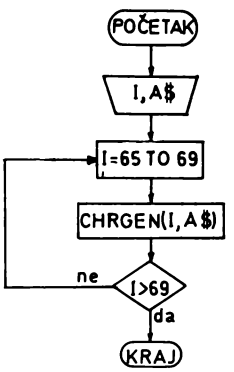
```
A$="3F3F3F3F3F3F3F3F3F"
```

Algoritamska šema je data na sl.2.26.

FOR/NEXT petljom postižemo izvršenje naredbe CHRGEN za date vrednosti promenljive I.

Program za datu algoritamsku šemu je sledeći:

```
10 FOR I=65. TO 69
20A$="3F3F3F3F3F3F3F3F"
30CHRGEN(I,A$)
40 NEXT I
```



Sl.2.26

Naredbom u programskom redu 30 vršimo predefinisanje znakova čiji su decimalni kodovi od 65 do 69, u nove karaktere definisane nizom A$ u liniji 20.

Posle izvršenja programa, pritiskom na dirke A,B,C,D,E, na ekranu će se štampati znak koji smo generisali naredbom CHRGEN.

Generisanje novih boja znakova

Ranije smo naučili da PECOM 64 radi sa 128 znakova, čiji su decimalni kodovi od 0 do 127. Međutim, PECOM 64 u svom karakter generatoru sadrži još 128 znakova sa decimalnim kodovima od 128 do 255. Oni su istog oblika kao i prethodni znaci, ali u zelenoj boji. Ovu konstataciju ilustrovaćemo primerom.

Naredbom CPOS, koja može da se izvrši u direktnom režimu rada (bez broja programskog reda), pozicioniraćemo na sredini ekrana znak sa decimalnim kodom 65 (veliko slovo A):

```
CPOS(20,12,65)(RETURN)
```

Kao rezultat izvršenja ove naredbe računar će na sredini ekrana štampati slovo A u crnoj boji. Uvećamo li decimalni kod slova A za 128, dobićemo:

```
65+128=193
```

Ponovnim korišćenjem naredbe

```
CPOS(20,12,193)(RETURN)
```

računar će na ekranu, na istom mestu gde se nalazilo prethodno slovo A, štampati slovo A u zelenoj boji.

Standardnim oblikom naredbe CHRGEN mogli smo generisati znakove u četiri boje (crna, crvena, plava i ljubičasta). Upotrebom decimalnih kodova znakova od 128 do 255, pomoću naredbe CHRGEN, možemo generisati još četiri boje znakova: zelenu, belu, maslinastu i žutu.

Nove boje znakova generisaćemo na primeru punog znaka zbog lakšeg rada.

Generisanje znaka u zelenoj boji

Matrica novog znaka biće sledeća:

0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1

- — u kolone za definisanje boje znaka upisaćemo sve 0
- — u kolone za definisanje oblika znaka upisaćemo 1 (zato što generišemo pun znak)
- — sve vrste matrice imaće vrednosti 3F

Sledećim programom generisaćemo novi znak na decimalnom kodu dirke A uvećanom za 128:

```
65+128=193
```

Tako generisan znak pozicioniraćemo i štampati na sredini ekrana korišćenjem naredbe CPOS:

```
10CHRGEN(193,"3F3F3F3F3F3F3F3F3F")
20CPOS(20,12,193)
```

Posle izvršenja ovog programa računar će na ekranu štampati pun znak u zelenoj boji.

Generisanje znaka u beloj boji

Za generisanje znaka u beloj boji upotrebićemo istu matricu kao i u prethodnom primeru, s tom razlikom, što se u kolonama koje definišu bitove za boju upisuje 1. Vrste u matrici sada dobijaju vrednosti FF.

Program za ovaj slučaj je sledeći:

```
10CHRGEN(193,"FFFFFFFFFFFFFFFFFF")
20CPOS(20,12,193)
```

Kada se izvrši ovaj program, računar će na sredini ekrana odštampati novogenerisani znak u beloj boji.

Generisanje znaka u žutoj boji

Za generisanje znaka u žutoj boji u matrici znaka, u kolone za definisanje bitova boje, upisujemo 0 i 1. Vrste u matrici sada dobijaju vrednosti 7F.

Program za ovaj slučaj je sledeći:

```
10CHRGEN(193,"7F7F7F7F7F7F7F7F")
20CPOS(20,12,193)
```

Kada se izvrši ovaj program, računar će na ekranu štampati novogenerisani znak u žutoj boji.

Generisanje znaka u maslinastoj boji

Za generisanje znaka u maslinastoj boji u matrici znaka u kolone za definisanje bitova boje upisujemo 1 i 0. Vrste u matrici dobijaju vrednosti BF.

Program za ovaj slučaj je sledeći:

```
10CHRGEN("BFBFBFBFBFBFBFBF")
20CPOS(20,12,193)
```

Kada se izvrši ovaj program, računar na sredini ekrana štampa novogenerisani znak u maslinastoj boji.

2.19. POTPROGRAMI

Često se prilikom rešavanja zadataka javlja potreba da se isti postupak ponovi na više mesta u programu. Na primer, korišćenje istih formula ali za različite vrednosti, ili izvršavanje iste procedure koju nije moguće uklopiti u ciklus. Program, napisan za rešavanje takve vrste problema, sadržavao bi isti niz naredbi na različitim mestima. Poželjno je da se takav niz naredbi izdvoji u posebnu celinu, koja se može koristiti po potrebi. Niz naredbi, izdvojen na takav način naziva se **potprogram.**

Formiranje potprograma ima više prednosti:

- kraći zapis programa
- lakše prenošenje programa na računar
- manje angažovanje memorijskog prostora u računaru
- lakše testiranje programa, jer se potprogrami kao posebne celine mogu odvojeno testirati
- isti potprogram se može koristiti u raznim programima.

Da bi se u programskom jeziku stvorila mogućnost korišćenja potprograma, moraju biti definisani sledeći problemi:

- kako se prepoznaje skup naredbi koji čini potprogram
- kako se definišu ulazne i izlazne veličine potprograma
- kako se prelazi iz programa u potprogram
- kako se vraća iz potprograma u program.

2.19.1 Naredba za pozivanje potprograma

Iz glavnog programa potprogram se poziva naredbom

GOSUB<brojni izraz>

Ovom naredbom se prekida izvršenje tekućeg programa i prelazi se na izvršenje potprograma koji počinje na adresi datoj brojnim izrazom. Ova naredba je identična

naredbi GOTO sa izuzetkom što program pamti gde je došlo do GOSUB. Kada BASIC naiđe na naredbu RETURN, on će se vratiti na izvršenje naredbe koja dolazi posle poslednje izvršene naredbe GOSUB u tekućem programu. Na ovaj način potprogrami se mogu povezivati onoliko duboko koliko to memorija dopušta.

Takođe se iz jednog potprograma naredbom GOSUB može preći u drugi potprogram, s tim što se prvo vrši povratak iz drugog potprograma u prvi potprogram a tek onda u tekući program.

2.19.2. Naredba za povratak iz potprograma

Povratak iz potprograma u glavni program vrši se naredbom

RETURN

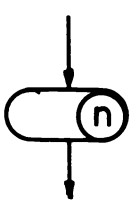
Ovom naredbom označava se kraj potprograma. Izvršenje glavnog programa nastavlja se naredbom koja sledi iza poslednje izvršene GOSUB naredbe.

2.19.3. Potprogramski segmenti

Pošto smo se upoznali sa naredbama za poziv i povratak iz potprograma pristupićemo analizi programskih segmenata. **Potprogramskim segmentom** smatraćemo niz programskih redova koji se završava naredbom povratka u program.

U algoritamskim šemama prelazak na potprogramski segment označavaćemo simbolom

gde je n — broj koji označava obeležje prvog programskog reda u programskom segmentu.



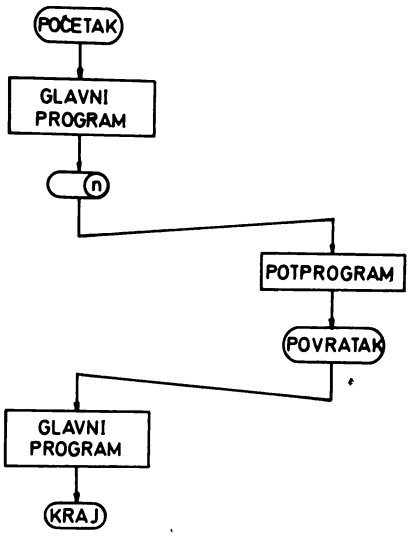
Povratak iz potprograma označavaćemo simbolom:

Sada možemo dati opštu algoritamsku šemu glavnog programa sa potprogramskim segmentom (sl.2.27.).

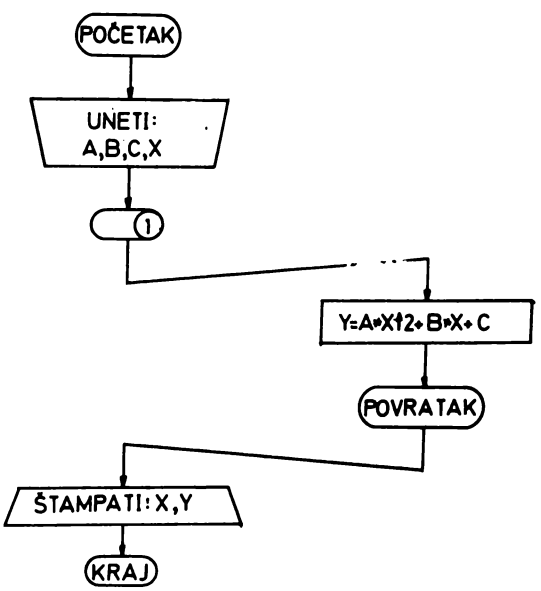
Primer: Nacrtati algoritamsku šemu i napisati program za izračunavanje vrednosti kvadratne funkcije

$y=Ax^2+Bx+C$

Parametre A,B,C i promenljivu X uneti u glavni program a vrednost kvadratne funkcije izračunati u potprogramu. Algoritamska šema je data na sl.2.28.


POČETAK
GLAVNI PROGRAM
n
POTPROGRAM
POVRATAK
GLAVNI PROGRAM
KRAJ

Sl.2.27


POČETAK
UNETI: A,B,C,X
1
Y=A*X↑2+B*X+C
POVRATAK
ŠTAMPATI: X,Y
KRAJ

Sl.2.28

Program po datoj algoritamskoj šemi je sledeći:

```
10 CLS
20 CPOS(2,3):INPUT"UNETI A=";A
30 CPOS(2,5):INPUT"UNETI B=";B
40 CPOS(2,6):INPUT"UNETI C=";C
50 CPOS(2,7):INPUT"UNETI X=";X
60 GOSUB 200
70 CPOS(2,10):PRINT"VREDNOST KVADRATNE FUNKCIJE"
80 CPOS(2,12):PRINT"Y=";A"*X↑2";B;"*X+";
90 CPOS(2,14):PRINT"ZA X=";X;" JE Y=";Y
100 END
```

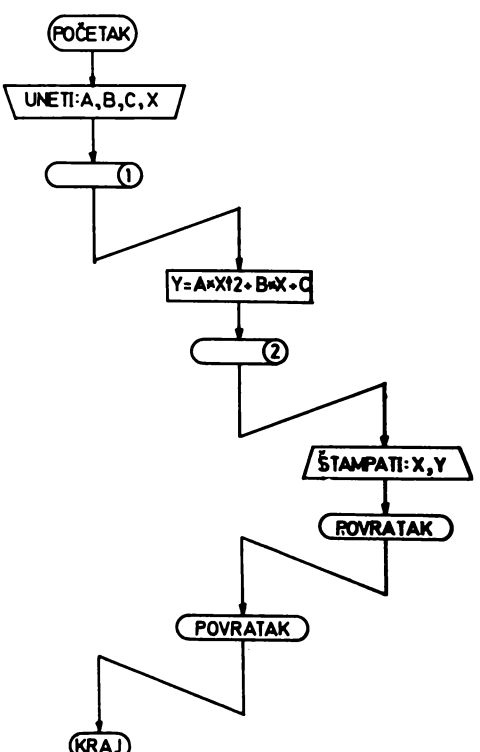
Za A=1, B=2, C=3, X=4

jedan od izlaznih izveštaja može biti sledeći:

```
VREDNOST KVADRATNE FUNKCIJE
Y=1*X↑2 + 2*X + 3
za X=4 JE Y=27
```

Primer: Rešićemo prethodni zadatak sa dva potprograma, gde se u drugi potprogram ulazi iz prvog potprograma.

Algoritamska šema je data na sl.20.29.



Sl.2.29

U glavnom programu se unose vrednosti parametara A,B,C i vrednost promenljive X. Računar, zatim poziva potprogram 1 u kome se izračunava vrednost funkcije Y za date parametre A,B,C i promenljive X. Iz ovog potprograma računar prelazi u potprogram 2 u kome se izdaju naredbe za štampanje vrednosti kvadratne funkcije.

Povratak iz potprograma 2 u glavni program ide preko potprograma 1. Direktan povratak u glavni program nije moguć.

Program po datoj algoritamskoj šemi je sledeći:

```
10 CPOS(2,4):INPUT"UNETI A="A
20 CPOS(2,5):INPUT"UNETI B="B
30 CPOS(2,6):INPUT"UNETI C="C
40 CPOS(2,7):INPUT"UNETI X="X
50 GOSUB 200
60 END
```

Potprogram 1 je sledeći:

```
200 Y=A*X*X + B*X + C
210 GOSUB 300
220 RETURN
```

Potprogram 2 je sledeći:

```
300 CPOS(2,10):PRINT"VREDNOST KVADRATNE FUNKCIJE"
310 CPOS(2,12):PRINT"Y=";A;*X↑2 + ";B 310 CPOS;"*X +";C
320 CPOS(2,14):PRINT"ZA X=";X;" JE Y=";Y
330 RETURN
```

2.19.4. Opšti potprogram

Potprogram napisan na mašinskom jeziku, koji je moguće pozvati iz BASIC-a zove se **opšti potprogram.** Naredba za poziv opšteg potprograma je

CALL(<brojni izraz>)

Ova naredba obezbeđuje vezu između BASIC-a i programiranja na mašinskom jeziku, ona služi kao poziv potprograma i prenosi izvršenje. Adresa potprograma se određuje pomoću brojnog izraza. Kako su adrese memorijskih lokacija označene heksadecimalnim brojevima, tada i brojni izraz treba da bude heksadecimalni broj, kome prethodi oznake &.

Naredba CALL se može proširiti na još dva načina:

1. Naredbom CALL čiji je format

CALL(<brojni izraz1>,<brojni izraz2>)

može se pozivati program napisan na mašinskom jeziku, počevši od adrese definisane brojnim izrazom1. Ako se upotrebi brojni izraz2, tada se njegova vrednost prenosi na potprogram napisan na mašinskom jeziku kao 16. bitni binarni ceo broj u registar R8.

2. Format naredbe CALL

CALL(<brojni izraz1>,<brojni izraz2>,<brojni izraz3>)

koristi se kada je u potprogramu, pored smeštanja podataka u registar R8, potrebno smestiti i podatak u registar RA. Sadržaj registra RA biće vrednost brojnog izraza 3.

Pored potprograma koje korisnik piše u mašinskom jeziku ili asembleru, naredbom CALL se mogu pozivati i potprogrami koji već postoje smešteni u ROM memoriji računara. Tu spadaju potprogrami za konverziju u rad sa ćirilicom i latinicom i slično.

Ako želimo preći u režim rada sa ćirilicom pozivamo potprogram koji je smešten na heksadecimalnoj adresi CE00:

```
CALL((&CE00)
```

Želimo li ponovo da se vratimo u režim rada sa latinicom, pozivamo potprogram koji je smešten na heksadecimalnoj adresi CE70:

```
CALL(&CE70)
```

Pozovemo li potprogram na heksadecimalnoj adresi 8002 izvršiće se identična operacija kao da smo isključili računar i ponovo ga uključili (resetuje se sadržaj računara):

```
CALL(&8002)
```

Sa ovom naredbom potrebno je krajnje oprezno baratati.

2.19.5. Funkcija za prelaz na mašinski jezik

Funkcija ove grupe je USR, i može imati sledeće formate:

USR(<brojni izraz>)

USR(<brojni izraz1>,<brojni izraz2>)

USR(<brojni izraz1>,<brojni izraz2>,<brojni izraz3>)

Ova funkcija ima dejstvo kao naredba CALL koja je opisana u prethodnom odeljku, ali s tom razlikom da je USR funkcija koja će se upotrebiti kao deo nekog izraza. Kada se naiđe na USR, vrši se poziv potprograma na mašinskom jeziku, čija je adresa definisana brojnim izrazom. Podaci se mogu preneti na potprogram na potpuno isti način kao naredbom CALL. Kod funkcije USR, kada se naiđe na naredbu D5 u programu na mašinskom jeziku, BASIC će dati 32-bitni ceo broj kao vrednost za funkciju USR. Ovaj 32-bitni broj obrazuje sadržaj registara R8 i RA. Registar R8 daje 16 bitova nižeg reda a RA daje 16 bitova višeg reda.

2.20. NAREDBA ZA PODATKE PROGRAMA

Do sada smo se upoznali sa dva načina unošenja numeričkih podataka potrebnih računaru da izračunava i rešava postavljene zadatke. Prvi način je bio preko naredbe LET. Međutim, ako je taj podatak trebalo promeniti, morao se menjati program, odnosno naredba LET.

Drugi način, koji je prihvatljiviji u komunikaciji sa računarom, jeste preko naredbe INPUT.

Naredbe READ, DATA i RESTORE omogućavaju treću vrstu unošenja podataka, koja pomalo podseća na način unošenja podataka pomoću kartica.

Naredba READ nalaže računaru da pročita podatak koji se nalazi upisan u naredbi DATA. Uz naredbu READ nalaze se samo nazivi promenljivih, čije vrednosti

računar mora da pročita iz naredbe DATA. Naredba DATA nije ništa drugo, nego magacin podataka.

Isto kao i naredbe FOR i NEXT, tako se i naredbe READ i DATA moraju pojaviti u programu.

Naredba READ u programu dolazi na ono mesto koje određuje korisnik već prema logici programa, a naredba DATA se može nalaziti bilo gde unutar programa. Kada računar naiđe na naredbu READ, odmah traži i odgovarajuću naredbu DATA.

Osnovna razlika između naredbi READ i INPUT je u vremenu unošenja podataka u program. Programi u kojima se za unošenje podatka koristi naredba INPUT su univerzalni i nikada se ne moraju prepravljati za izvršenje sa novim podacima. Preko INPUT naredbe podaci se unose u toku samog izvršavanja programa. Naredba READ traži da se podaci u naredbi DATA unesu u računar pre početka izvršenja programa. Ako se takav program ponovo želi, ali sa novim podacima, stari podaci se izbacuju a ukucavaju se novi.

Kada je potebno da se počne sa čitanjem podataka ponovo iz početka, koristi se naredba RESTORE, koja usmerava pokazivač podataka na prvi podatak u prvoj naredbi DATA u programu.

2.20.1. Definisanje podataka

Za definisanje podataka, koji će se koristiti u programu, upotrebljava se naredba DATA, čiji je format:

DATA<podatak>,<podatak>,...

Naredba DATA sadrži podatke koje će koristiti naredba READ. Svaki podatak u naredbi DATA mora biti odvojen zarezom. Podatak može biti i niz znakova, gde svaki niz znakova mora biti pod navodnicima. Naredba DATA može se javiti bilo gde u programu. Za složenije programe može se koristiti više naredbi DATA, koje se obađuju redom.

Primeri ispravnih naredbi DATA su:

```
10 DATA 1,2,3,4, - elementi su brojne konstante
20 DATA "ANA","MIRA","VESNA" - elementi su nizovi
30 DATA SIN(45), SIN(60),COS(45),COS(60) - elementi su funkcije
40 DATA 2*A,A*A,B,C,D - elementi su izrazi
```

2.20.2. Čitanje podataka

Za čitanje podataka, koji su definisani u naredbi DATA, koristi se naredba READ, čiji je format:

READ<promenljiva>,<promenljiva>,...

Naredba READ se koristi za čitanje podataka iz naredbe DATA sleva na desno i za dodeljivanje ovih podataka nekoj novoj promenljivoj. Svaki put kada naredba READ zahteva još podataka, Interni ukazatelj BASIC-a se pomera do sledećeg podatka u naredbi DATA. Kada je naredba DATA bez podataka, BASIC će nastaviti sa pretraživanjem zbog neke druge naredbe DATA. Kada ne nađe ni jednu, šalje se poruka o greški koja govori o nedostatku podataka. Važno je voditi računa o vrstama

promenljivih u naredbi READ i odgovarajućih podataka u naredbi DATA. Oni moraju biti saglasni po pitanju oblika: nizovi idu sa nizovima, brojevi sa brojevima itd. Primer prihvatljivog para naredbi DATA/READ je:

```
5 DIM A(4)
10 DATA 10,20,30,40
20 FOR A=1 TO4
30 READ A(A)
40 NEXT A
50 READ A$(1),B,C
60 PRINT A$(1),B+C
70 DATA "B+C JEDNAKO JE",20,30
```

Rezultat rada ciklusa sa kontrolnom promenljivom A biće punjenje polja A na sledeći način:

```
A(1)=10      A(2)=20      A(3)=30      A(4)=40
```

Rezultat reda sa obeležjem 60 biće:

```
B+C JEDNAKO JE 50
```

Primer: Program kojim se učitavaju elementi matrice:

```
10 DIM A(3,3)
20 FOR I=1 TO 3
30 FOR J=1 TO 3
40 READ A(I,J)
50 PRINT A(I,J);"";
60 NEXT J
70 PRINT
80 NEXT I
90 DATA 5,10,8,11
100 DATA 3,2,8
110 DATA 1,2,
```

Naredbom DIM u programskom redu 10 definiše se matrica 3X3. FOR naredbama (programski redovi 20 i 30) i naredbom READ (programski red 40) učitavaju se vrednosti elemenata matrice iz naredbe DATA (programski redovi 90,100 i 110). Kada se upotrebe svi elementi jedne naredbe DATA prelazi se na drugu.

Ovim programom definisali smo matricu:

$$A = \begin{vmatrix} 5 & 10 & 8 \\ 11 & 3 & 2 \\ 8 & 1 & 2 \end{vmatrix}$$

Primer: Napisati program za štampanje poruke na ekranu primenom naredbi DATA i READ.U ovom primeru demontriran je jedan od načina prikazivanja teksta na ekranu. Decimalni kodovi znakova koji se štampaju, smeštaju se u memoriju naredbama DATA. Na početku programa vršimo dimenzionisanje polja A, kome će se dodeljivati vrednosti iz DATA.

Prvom FOR/NEXT petljom štampa se prvi red teksta. Naredbom READ A(I) čitaju se redom podaci iz naredbe DATA u programskom redu 110. Pozicije u kojima se štampaju pojedini znaci su u funkciji promenljive I, čije su vrednosti definisane FOR/NEXT petljom.

Drugom FOR/NEXT petljom se obavljaju iste aktivnosti kao i u prethodnoj FOR/NEXT petlji, s tom razlikom što se u polje A(J) vrši čitanje podataka iz naredbe DATA u programskom redu 120.

Program bi izgledao ovako:

```
10 DIM A(30)
20 CLS
30 FOR I=1 TO 21
40 READ A(I)
50 CPOS(10+i,11): PRINT CHR$(A(I))
60 NEXT I
70 FOR J=1 TO 19
80 READ A(J)
90 CPOS(11+J,13):PRINTCHR$(A(J))
100 NEXT J
110 DATA 79,86,32,74,69,32,68,69,77,79,78,83,84, 82.73,82,65,78,74,69
120 DATA 78,65,82,69,68,66,73,32,68,65,84,65,32,73,32,82,69,65,68,
130 END
```

2.20.3. Naredba RESTORE

Ova naredba resetuje ukazatelj BASIC-a na početak prve naredbe DATA u programu, dopuštajući na taj način višestruku upotrebu iste naredbe DATA u jednom programu.

Primer: Program kojim se učitava matrica sa istim vrednostima u svakoj od vrsta:

```
10 DIM A(10,10)
20 INPUT "N"N
30 FOR I=1 TO N
40 FOR J=1 TO N
50 READ A(I,J)
60 PRINT A(I,J)
70 NEXT J
80 RESTORE
90 PRINT
100 NEXT I
110DATA1,2,3,4,5,6,7,8,9,10
120 END
```

Naredbom DIM definiše se matrica A sa maksimalno 10 kolona i 10 vrsta. Naredbom INPUT bira se dimenzija matrice koja se generiše. Naredbom DATA u programskom redu 110 definišu se podaci za elemete vrsta u matrici. Cikličnom strukturom dve FOR/NEXT petlje definišu se kolone i vrste matrice.

U toku izvršavanja unutrašnje FOR/NEXT petlje, naredbom READ čitaju se podaci za elemente vrste matrice. Podaci se čitaju iz naredbe DATA. Posle završetka svakog ciklusa unutrašnje FOR/NEXT petlje, naredbom RESTORE vraća se čitanje na prvi podatak iz naredbe DATA. Na ovaj način se generiše matrica gde su sve vrste sa istim elementima.

Za uneto N=2 program generiše matricu:

$$A=\begin{vmatrix} 1 & 2 \\ 1 & 2 \end{vmatrix}$$

Za N=3 matrica je:

$$A=\begin{vmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{vmatrix}$$

Primer: Nacrtati algoritamsku šemu i napisati program koji će prikazati pune znakove u svih 8 boja u 20 redova.

Algoritamska šema je data na sl.2.30. Na početku programa računar briše ekran i boji ga zelenom bojom. Zatim dimenzioniše polje kojim će se čitati podaci smešteni u DATA. Spoljašnjom FOR/NEXT petljom se postiže crtanje znakova u vrstama od 1 do 20. Unutrašnjom FOR/NEXT petljom definiše se promenljiva K od 1 do 4, kojom se obavljaju sledeće aktivnosti:

1 — čita se K-ti podatak iz naredbe DATA

2 — definišu se decimalni kodovi znakova koji se pregenerišu

3 — štampaju se svi pregenerisani znaci.

Za generisanje novih znakova koristimo metode koje su obrađene u poglavlju 3.18.5. Naredbu CHRGEN koristimo u obliku

```
CHRGEN(K,A$)
```

gde je:

K — decimalni kod pregenerisanog znaka

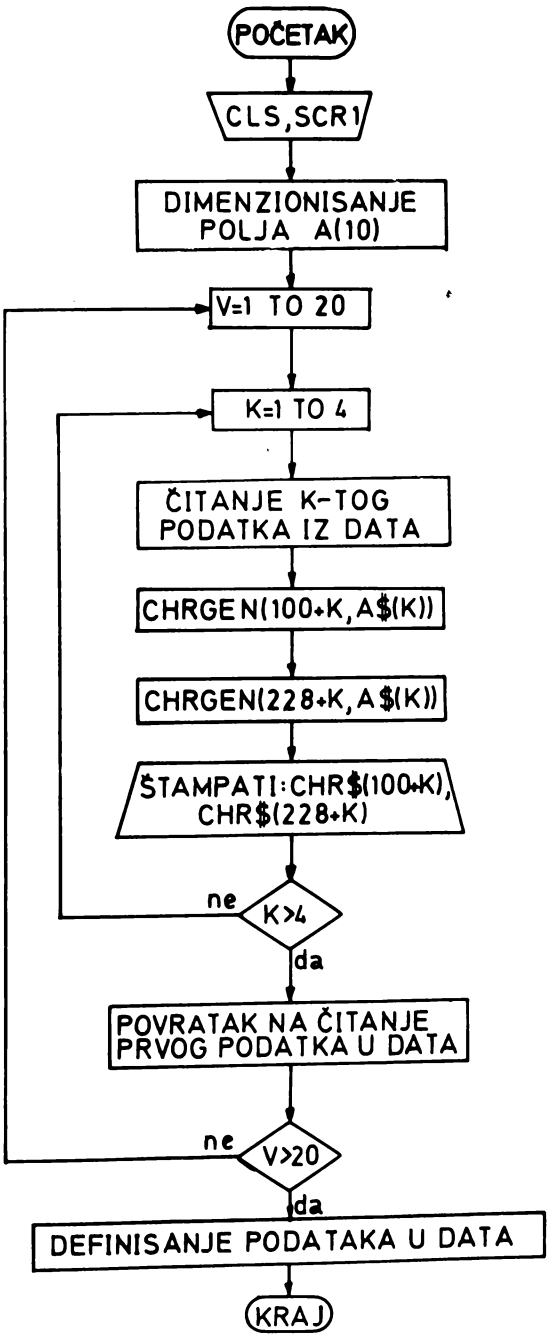
A$ — niz od 18 heksadecimalnih cifara koje generišu boju i oblik znaka

U svakom od 4 koraka unutrašnje FOR/NEXT petlje naredbom READ A$(K) čita se K-ti niz iz naredbe DATA. Posle izlaska iz unutrašnje FOR/NEXT petlje, naredbom RESTORE vrši se povratak na prvi podatak iz naredbe DATA, a zatim započinje naredni korak spoljašnje FOR/NEXT petlje.

Nizovi 18 heksadecimalnih cifara za naredbu CHRGEN su definisani u naredbi DATA na kraju programa.

Program za datu algoritamsku šemu je sledeći:

```
10 CLS
20 SCR 1
30 DIM A(10)
40 FOR V=1 TO 20
50 FOR K=1 TO 4
60 READ A$(K)
70 CHRGEN(100+K,A$(K))
80 CHRGEN(228+K,A$(K))
90 CPOS(2*K+14,V):PRINT CHR$(100+K);CHR$(228+K)
100  EXT K
110 RESTORE
120 NEXT V
130 DATA "3F3F3F3F3F3F3F3F3F","7F7F7F7F7F7F7F7F7F",
"BFBFBFBFBFBFBFBFBF","FFFFFFFFFFFFFFFFFF"
140 END
```


POČETAK
CLS,SCR1
DIMENZIONISANJE POLJA A(10)
V=1 TO 20
K=1 TO 4
ČITANJE K-TOG PODATKA IZ DATA
CHRGEN(100+K,A$(K))
CHRGEN(228+K,A$(K))
ŠTAMPATI:CHR$(100+K), CHR$(228+K)
ne
K>4
da
POVRATAK NA ČITANJE PRVOG PODATKA U DATA
ne
V>20
da
DEFINISANJE PODATAKA U DATA
KRAJ

Sl.2.30

```
40 FOR V=1 TO 20
50 FOR K=1 TO 4
60 READ A$(K)
70 CHRGEN(100+K,A$(K))
80 CHRGEN(228+K,A$(K))
90 CPOS(2*K+14,V):PRINT CHR$(100+K);CHR$(228+K)
100  EXT K
110 RESTORE
120 NEXT V
130 DATA "3F3F3F3F3F3F3F3F","7F7F7F7F7F7F7F7F",
"BFBFBFBFBFBFBFBF","FFFFFFFFFFFFFFFF"
140 END
```

Kada se startuje program, računar na ekranu štampa 8 raznobojnih vertikalnih linija dužine po 20 karaktera.

2.21. NAREDBA ZA PRELAZAK IZ REŽIMA RADA BASIC-a U REŽIM RADA MONITOR-a+

Iz režima rada BASIC-a računar može preći u režim rada MONITOR-a+ naredbom

M+

Posle pritiska na dirku RETURN, računar na ekranu postavlja znak >, nakon čega se mogu unositi naredbe MONITOR-a +.

2.22. NAREDBA ZA PRELAZAK IZ REŽIMA RADA BASIC-a U REŽIM RADA EDITOR-a

Iz režima rada BASIC-a računar može preći u režim rada EDITOR-a naredbom

ED

Posle pritiska na dirku RETURN ekran se briše a u gornjem levom uglu se postavlja trepćući kursor, čime je računar prešao u režim rada EDITOR-a.

2.23. PRELAZAK IZ BASIC-a U RAD NA MAŠINSKOM JEZIKU

Prelazak iz BASIC-a na mašinski jezik vrši se komandom

PROB

Nakon unošenja PROB sa tastature i pritiska na dirku RETURN, na ekranu se pojavljuje znak >. On nas upozorava da se sada nalzimo u režimu rada sa mašinskim jezikom. Na primer, ako unesemo

```
: PROB ... pritisnimo dirku RETURN
... režim rada na mašinskom jeziku
```

Sada se nalzimo u novom režimu rada i na raspolaganju nam stoji 8 novih komandi: D,I,P,R,W,B,S,N.

P

Ova komanda služi za prikazivanje sadržaja memorije. Iza znaka D unosi se heksadecimalna adresa memorijske lokacije sa kojom se počinje prikazivanje sadržaja memorije. Nakon toga se unosi znak za blanko i heksadecimalni broj koji definiše koliko će se memorijskih lokacija prikazati. Na primer, komanda

```
D220 10
```

daje

```
220 FFFFFFFFFFFFFFFF  FFFFFFFFFFFFFFFF
```

što znači da je 16 memorijskih lokacija počev od adrese 0220 prazno. Kao što se iz primera zaključuje, memorijska lokacija ne mora da se adresira sa četiri heksadecimalne cifre. Adresa može da počne sa prvom cifrom najveće težine različite od nule.

I

Ova komanda služu za unošenje programa na mašinskom jeziku kao i za njegove korekcije prilikom testiranja. Takođe ona služi za izmenu sadržaja bilo koje memorijske lokacije. Tako na primer,

```
I300 F800AA
```

znači da se u memorijske lokacije sa heksadecimalnom adresom 0300, 0301 i 0302 upisuje sadržaj F800AA. Tri bajta na mašinskom jeziku upisuju konstantu 00 u niži bajt registra A.

P

Kada napišemo program na mašinskom jeziku i želimo da ga pozovemo na izvršenje to ćemo učiniti komandom P. Pre nego što izvršimo komandu P, poželjno je prekontrolisati program koristeći komandu D.

R

Kao što se naredbom PLOAD u BASIC-u čita program sa kasete, tako se u režimu rada sa mašinskim jezikom komandom R čita program sa kasete koji je prethodno upisan na nju komandom W.

W

Ovo je komanda za upis programa na mašinskom jeziku na kasetu.

B

Ovom komandom vrši se prelaz iz režima rada (na mašinskom jeziku) na BASIC.

S

Funkcija ove komande je da obezbedi programsku vezu sa štampačem (ukoliko je priključen). Posle ove komande, za štampanje sadržaja memorije na štampaču, koristi se komanda D.

N

Ovom komandom vrši se programski prekid veze sa štampačem. Pri prelasku iz režima rada sa mašinskim jezikom u režim rada u BASIC-u treba koristiti ovu komandu.

2.24. KORIŠĆENJE GRAFIKE SREDNJE REZOLUCIJE NA PECOM-u 64

Ako vaš PECOM nakon uključenja na ekranu ispisuje sledeću poruku

```
© Ei PECOM 64 VER. 4.0.
READY
:
```

tada je u njegovoj RAM memoriji (EVROM-u) smešten program koji omogućava korišćenje finije grafike od one koju pruža standardna verzija PECOM:a 64. U tu svrhu, korisniku na raspolaganju, stoji gornja polovina ekrana, dok se druga polovina može koristiti za prikazivanje određenog teksta, poruke ili tabela.

Grafika se realizuje pomoću tačaka, gde se svaka tačka može prikazati u 108 vrsta i 229 kolona.

Ova grafika je realizovana posebnim programom, koji vrši pregenerisanje karaktera, na način koji najbliže aproksimira trenutne pozicije tačaka. Ako uzmemo u obzir činjenicu da PECOM 64 poseduje skup od 128 karaktera, tada postoje izvesna ograničenja u upotrebi ove grafike. Ona se ogledaju u tome, da se kod crtanja složenijih slik moraju upotrebiti svi raspoloživi znaci. Ako dođe do takve situacije, računar će crtati sve dotle dok ne upotrebi i zadnji raspoloživi znak, a zatim će nastaviti sa izvršavanjem programa, s tim što neće ništa prikazivati na ekranu.

Za realizaciju ovakve grafike koristi se naredba PLOT, formata

PLOT (<brojni izraz 1>,<brojni izraz 2>)

gde je:
brojni izraz 1 — redni broj kolone i može uzimati vrednosti od 1 do 229
brojni izraz 2 — redni broj vrste i može uzimati vrednosti od 1 do 108

Za razliku od naredbe CPOS, kod naredbe PLOT je pozicija vrste sa rednim brojem 1 na sredini ekrana, a pozicija vrste sa rednim brojem 108 u krajnjem gornjem delu ekrana. Pozicije kolona sa najnižim i najvišim rednim brojem iste su kao i kod naredbe CPOS.

Na početku svakog programa treba upotrebiti naredbu

```
CALL(&CE70)
```

kojom se svi karakteri u računaru pregenerišu u oblik kakav imaju u trenutku kada se računar uključi. Ovo činimo iz sledećih razloga:

```
POKE(&7E30,#3C)
```

Posle brisanja ekrana može se pristupiti programu koji će definisati krivu koja se prikazuje na ekranu.

U nastavku teksta dajemo programe za crtanje nekuh elementarnih slika:

1. Za crtanje horizontalne linije koristimo program:

```
10 CALL (&CE70)
20 POKE (&7E30,#3C)
30 CLS
40 V=54
50 FOR K=1 TO 229
60 PLOT (K,V)
70 NEXT K
80 END
```

Ukoliko želimo da crtamo liniju sa proredom, potrebno je u naredbi FOR u programskom redu .50 uvesti priraštaj veći od 1. Na primer:

```
50 FOR K=1 TO 229 STEP 5
```

crtaće svaku petu tačku horizontalne linije.

2. Za crtanje vertikalne linije koristimo program:

```
10 CALL (&CE70)
20 POKE (&7E30,#3C)
30 CLS
40 K=100
50 FOR V=1 TO 104
60 PLOT (K,V)
70 NEXT V
80 END
```

3. Za crtanje kose linije možemo koristiti programe:

```
10 CALL (&CE70)
20 POKE (&7E30,#3C)
30 CLS
40 FOR V=1 TO 104
50 K=2*V
60 PLOT (K,V)
70 NEXT V
80 END
```

Pomoću datih programa možemo crtati i složenije figure.

3. Arhitektura mikroprocesora CDP 1802

Za pisanje programa na mašinskom jeziku i asembleru potrebno je detaljno poznavanje arhitekture mikroprocesora CDP 1802 i uloge njegovih pojedinih registara. Zato ćemo se detaljnije upoznati sa pojedinim delovima mikroprocesora.

Arhitektura mikroprocesora CDP 1802 prikazana je na sl.3.1. Ona obuhvata:

- registarsku matricu R od 16 16-bitnih registara opšte namene (Scratch Pad Registers)
- grupu 4-bitnih registara N,P,X,I.
- privremeni 8-bitni registar T
- akumulator D
- aritmetičko-logičku jedinicu ALU
- kolo za povećanje/smanjenje INCR/DECR
- upravljačku logiku
- adresni (privremeni) registar A
- flip-flopove: registar DF, flip-flop dozvole prekida IE, izlazni flip-flop Q
- 8-bitnu magistralu za adrese
- 8-bitnu magistralu za podatke.

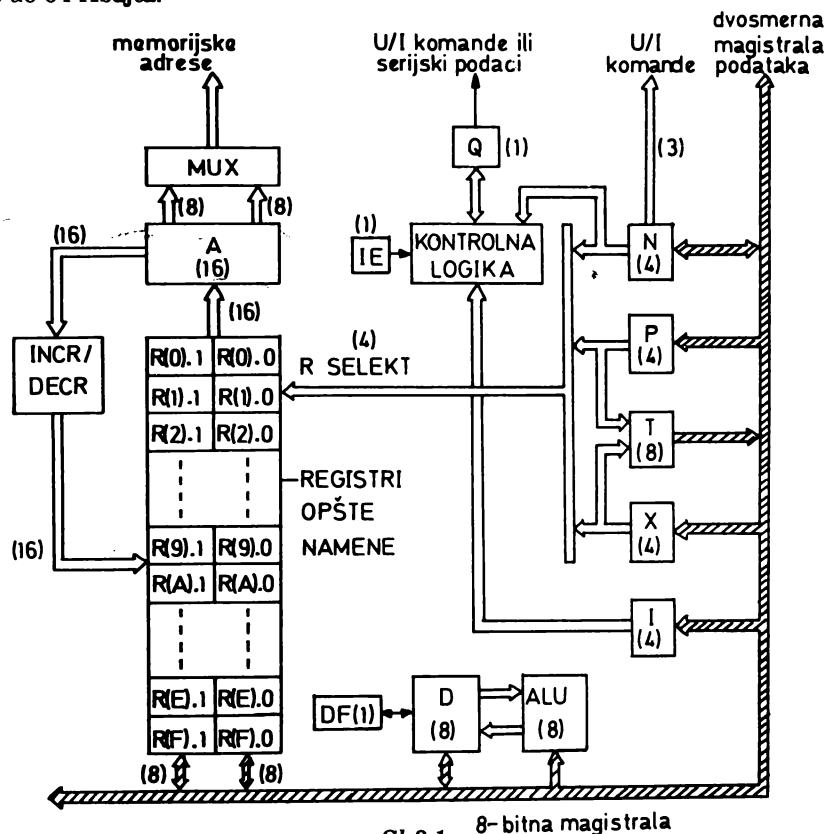
Registri opšte namene R mogu se koristiti:

- kao brojač naredbi PC (Program Counter), čiji je zadatak da adresira tekuću naredbu u programskoj memoriji. Po završetku ciklusa pripreme naredbe, vrednost brojača naredbi automatski se uvećava za 1,
- kao ukazatelj podataka (Data Pointer) da ukaže na memorijsku lokaciju gde je podatak smešten, ili gde ga treba smestiti,
- kao pomoćni registar za podatke SPR (Scratch Pad Register) čija je uloga da čuva 2 bajta podataka,
- kao memorija (SCRATCH PAD MEMORY), pošto se bitovima manje i veće težine može pristupiti posebno preko magistrale za podatke.

Sadržaj bilo kog R registra može biti upućen, preko registra A:

- na adresnu magistralu
- u akumulator (bilo koji od 2 bajta može se smestiti u 8-bitni akumulator D)
- u kolo za povećanje/smanjenje, gde se uvećava ili smanjuje za jedan i vraća unazad u izabrani registar R.

Pošto su brojač naredbi i ukazatelj podataka 16-bitni, moguće je adresiranje memorije do 64 Kbajta.



Sl.3.1

Četvorobitni registar P ukazuje na registar iz registarske matrice R koji ima ulogu brojača naredbi. Sadržaj registra može se menjati samo jednom naredbom. Promenom sadržaja registra P menja se brojač naredbi

Četvorobitni registar X ukazuje na adresni registar pri ulazno-izlaznim i nekim aritmetičko-logičkim operacijama. Sadržaj registra X određuje koji će registar iz registarske matrice R ukazivati na memorijske lokacije iz kojih se uzimaju podaci.

Četvorobitni registri I i N koriste se da prihvate pročitanu naredbu, i to tako da se 4 bita veće težine bajta naredbe smeštaju u registar I, a u registar N smeštaju 4 bitna manje težine. Zavisno od tipa naredbe, N može imati sledeće funkcije:

- da adresira jedan od 16 registara iz registarske matrice R koji se koristi kod operacija sa registrima
- da ukaže na smer prenosa kod ulazno/izlaznih naredbi i da definiše da se izlazni bajt interpretira kao kod za izbor ulazno/izlaznih uređaja, kontrolni kod ili bajt podataka
- da ukaže na određene operacije koje se izvršavaju kod ALU naredbi, na tipove testova kod naredbi grananja

— sadrži vrednost koja se smešta u registar P, kako bi se označio novi registar koji se koristi kao brojač naredbi R(P)
— sadrži vrednost koja se smešta u registar X, kako bi se označio novi registar koji se koristi kao ukazatelj podataka R(X)

Privremeni registar T je 8-bitni registar u koji se po prihvatanju zahteva za prekid, privremeno pamte sadržaji P i X registara tekućeg programa.

Registar za podatke D je 8-bitni registar koji se koristi pri prenosu podataka između registara R i magistrale za podatke. Registar D ima ulogu akumulatora.

Aritmetičko-logička jedinica — ALU je 8-bitna mreža koja nad dva 8-bitna operanda može da izvršava aritmetičke i logičke operacije. Jedan operand se uzima iz akumulatora, a drugi iz memorije.

Registar DF je 1-bitni registar koji se postavlja pri izvršenju naredbi sabiranja, oduzimanja i pomeranja i čiji se sadržaj može testirati naredbama grananja.

Flip-flop Q se može programski postavljati i resetovati. Sadržaj flip-flopa Q se može ispitivati nekim naredbama grananja. Stanje ovog flip-flopa se može koristiti kao serijski izlaz iz mikroprocesora.

Flip-flop dozvole prekida IE je 1-bitni registar koji se koristi kod prekida. Ako je IE postavljen u stanje logičke jedinice mikroprocesor prihvata zahteve za prekid, a ako je IE postavljen u stanje logičke nule zahtev za prekid se ignoriše. Prihvatanjem zahteva za prekid, IE se resetuje i novi zahtevi za prekid se ignorišu.

3.1. PRELAZAK IZ REŽIMA BASIC-a U MONITOR+, EKRANSKI EDITOR I ASEMBLER KOD PECOM-a 64

Deo memorijskog prostora ROM memorije od heksadecimalne adrese C000 do F400 iskorišćen je za smeštanje sledećih sistemskih programa:

— MONITOR+
— ekranski EDITOR-CREDIT
— ASEMBLER, simbolički programski jezik za mikroprocesor CDP 1802
— program za konverziju latinica-ćirilica
— program za podršku rada računara u računarskoj učionici
— karakter generator za ćirilicu

MONITOR+ sadrži programe za rad sa memorijom i programe za rad u mašinskom kodu. Ekranski EDITOR-CREDIT služi za unošenje programa u računar, a omogućava i prevođenje programa iz ASEMBLERA u mašinski jezik.

U režim rada MONITOR+ može se ući na dva načina. Prvi način je da se iz BASIC-a naredbom

```
PROB
```

i pritiskom na dirku RETURN pređe u režim rada u mašinskom kodu, a zatim pozivanjem programa koji počinje na adresi D000

```
P D000
```

i pritiskom na dirku RETURN, pređe u MONITOR+.

Drugi način je mnogo jednostavniji, jer postoji naredba

```
M+
```

kojom se direktno iz BASIC-a prelazi u MONITOR+.

Ako iz MONITOR-a+ želimo preći u EDITOR, to možemo postići naredbom

```
>E
```

Za direktan prelazak iz BASIC-a u EDITOR postoji naredba

ED

Posle pritiska na dirku RETURN briše se ekran, a trepćući kursor se pojavljuje u gornjem levom uglu ekrana. Prelazak u komandni način rada EDITOR-a vrši se pritiskom na dirku BREAK. Na ekranu se pojavljuje horizontalna crta sa trepćućim kursorom u gornjem levom uglu.

Iz komandnog načina rada EDITOR-a može se pozvati ASEMBLER naredbom

ASS

i pritiskom na dirku RETURN.

Povratak iz komandnog načina rada EDITOR-a u datoteku vrši se pritiskom na dirku LINE FEED.

Prelazak iz komandnog načina rada EDITOR-a u MONITOR+ vrši se istovremenim pritiskom na dirke CTRL i C.

Prelazak iz režima rada MONITOR-a u BASIC vrši se naredbom

```
>B
```

3.2. UPOTREBA MONITOR-a+

MONITOR+ obuhvata sve naredbe za rad sa memorijom, i to:

- startovanje mašinskih programa
- prikazivanje sadržaja memorijskih lokacija
- unošenje sadržaja memorijskih lokacija
- prenošenje blokova podataka iz memorije PECOM-a na kasetu
- učitavanje blokova podataka sa kasete u memoriju PECOM-a

3.2.1. Štampanje sadržaja memorije

Naredba

```
>D
```

služi za štampanje sadržaja memorijskih lokacija u heksadecimalnom kodu na ekranu ili štampaču. Na pitanje FROM korisnik unosi početnu adresu bloka koji se štampa. Potrebno je uneti najmanje jedan, a najviše četiri znaka. Ukoliko se unese više od četiri znaka, u obzir se uzimaju poslednja četiri otkucana znaka. Prikazivanje teče toliko dugo, koliko se drži pritisnuta dirka CTRL. Prekidanje prikaza obavlja se pritiskom na dirku BREAK. Na primer:

```
>D
  FROM 5500 (RETURN)
```

daje

```
  5500 00 00 00 00 00 00 00 00
```

što znači da je 8 memorijskih lokacija, počev od 5500, prazno.

3.2.2. Upisivanje sadržaja u memoriju

Naredbom

I

upisuje se sadržaj u memorijske lokacije. Na pitanje FROM unosi se adresa na kojoj se započinje upis. Vrednost sadržaja unosi se u heksadecimalni kod. Na primer:

```
>I
 FROM 5500
 5500 F8 01 B8 F8 00 A8
```

što znači da se u memorijske lokacije sa heksadecimalnom adresom, počev od 5500 do 5505, upisuje sadržaj F801B8F800A8. Ovih 6 upisanih bajtova na mašinskom jeziku upisuju konstantu 0100 u registar R8.

3.2.3. Pozivanje EDITOR-a

Naredbom

E

poziva se ekranski EDITOR-CREDIT.

3.2.4. Pretraživanje memorije

Naredba

S

je naredba za pretraživanje. Njome se u delu, definisanom početnom i poslednjom adresom, traže mesta na kojima se nalazi specificirani niz znakova. Početna adresa memorijskog bloka, u kome se obavlja pretraživanje, unosi se na pitanje FROM, a njegova konačna adresa na pitanje TO. Na pitanje WHAT unosi se sadržaj niza u heksadecimalnom kodu. MONITOR+ javlja adrese na kojima se nalazi definisani niz. Naredno prikazivanje reda vrši se pritiskom na dirku RETURN.

3.2.5. Punjenje dela memorije jednim podatkom

Naredbom

F

se sve lokacije nekog memorijskog bloka pune vrednošću datom u heksadecimalnom kodu. Na pitanje FROM unosi se početna, a na pitanje TO krajnja adresa bloka. Na pitanje WHAT unosi se jednobajtna vrednost, kojom se pune memorijske lokacije bloka. Na primer, ako je potrebno sve memorijske lokacije od 3000 do 300C napuniti podatkom 08, onda je postupak rada sledeći:

```
>F
 FROM 3000
 TO 300C
 WHAT 08
```

Da li je naredba F korektno izvršena može se proveriti naredbom D, koja je prethodno opisana.

```
>D
FROM 3000
3000 08 08 08 08 08 08 08 08
3008 08 08 08 08 08 00 00 00
```

3.2.6. Pomeranje memorijskog bloka

Naredbom

M

vrši se pomeranje memorijskog bloka u memoriji. Na pitanje FROM unosi se početna, na pitanje TO krajnja adresa bloka, a na pitanje WHERE adresa na koju se pomera blok.

Primer: Memorijski blok iz prethodnog primera od 3000 do 300C sa podatkom 08 pomeriti počev od memorijske lokacije 3500.

```
>M
FROM 3000
TO 300C
WHERE 3500
```

Da li je naredba M korektno izvršena, može se proveriti naredbom D.

```
D
FROM 3500
3500 08 08 08 08 08 08 08 08
350c 08 08 08 08 08 00 00 00
```

3.2.7. Snimanje bloka na kasetu

Naredbom

W

snima se blok na kasetu. Potrebno je uneti samo početnu i krajnju adresu bloka. Snimanje se može prekinuti pritiskom na dirku BREAK. Po završetku snimanja (ili posle prekida) kontrolu preuzima originalni MONITOR računara.

3.2.8. Učitavanje bloka sa kasete

Naredbom

CTRL R

u memoriju se učitava blok sa kasete i to na iste adrese sa kojih je bilo izvršeno snimanje. Prekidanje naredbe moguće je izvršiti jedino pritiskom na taster za uključivanje i isključivanje računara.

3.2.9. Startovanje programa

Naredbom

C

vrši se startovanje programa u mašinskom kodu. Na pitanje FROM unosi se početna adresa programa. Zavisno od toga kako je napisan, završetak programa može biti sledeći:

— po izvršenju, program se vraća u MONITOR+
— program skače u originalni monitor
— program skače u BASIC

3.2.10. Štampanje sadržaja dela memorije na štampaču

Naredbom

1

računar se priprema za štampanje sadržaja dela memorije na štampaču. Sve što bi se nakon ove naredbe prikazalo na ekranu, odštampalo bi se i na štampaču

```
>1
>D
```

FROM (adresa) i štampanje obavlja se sve dok je pritisnuta dirka CTRL.
Prekidanje štampanja na štampaču vrši se naredbom

0

3.2.11. Konverzija teksta u ćirilicu

Naredbom

K

vrši se konverzija teksta u ćirilicu.

3.2.12. Konverzija teksta u latinicu

Naredbom

L

vrši se prebacivanje teksta iz ćirilice u latinicu.

3.2.13. Povratak u režim BASIC-a

Naredbom

B

vrši se prelazak iz režima MONITOR-a+ u BASIC.

3.3. CREDIT-EKRANSKI EDITOR

Za unošenje nekog programa ili teksta u računar, potrebno je da se znakovi, otkucani na tastaturi, prikladno protumače i smeste u memoriju računara. Poželjno je korisniku obezbediti preglednost u toku rada i mogućnost unošenja ispravki i izmena u tekstu, brisanje i oblikovanje teksta. Ekranski EDITOR-CREDIT pruža i mnogo više od toga. On omogućava:

— kreiranje programa, proizvoljnih datoteka, podataka i tekstova
— pozivanje ASEMBLER-a za mikroprocesor CDP 1802.

U tu svrhu CREDIT koristi dve grupe naredbi:

1 — naredbe za obradu teksta (pomeranje kursora, brisanje i dodavanje znakova, definisanje, prebacivanje, brisanje i pretraživanje blokova itd.)
2 — naredbe za komandni način rada (zapis i čitanje na kaseti, štampanje, pozivanje ASEMBLER-a i prekidanje rada)

Ukoliko neka od naredbi ne može da se izvrši, EDITOR javlja grešku.

Većina naredbi za rad sa EDITOR-om odnosi se na datoteke. Da bismo što lakše shvatili problematiku definisaćemo datoteku. Radi se o skupu povezanih podataka koje smo objedinili u jednu celinu i smestili na zajednički memorijski medijum. Na engleskom, datoteka je file-fascikla. Naša reč datoteka potiče od teka(=sveska) sa podacima (=data). Osnovna datoteka prepoznaje se nazivom koji ima 8 alfanumeričkih znakova. Nazive datoteka kreira korisnik. Naziv datoteke sastoji se od jednog do 6 afanumeričkih znakova a proširenje se sastoji od 2 alfanumerička znaka. Prvi znak naziva i proširenja datoteke mora biti slovo.

3.3.1. Pozivanje EDITOR-a

EDITOR se iz MONITOR-a+ poziva naredbom

```
>E
```

a iz BASIC-a naredbom

```
ED (RETURN)
```

Ukoliko je pre pozivanja EDITOR-a u memorijskom delu računara, korišćenog za smeštanje teksta, ranije postojao neki tekst kreiran EDITOR-om, onda će se na ekranu pojaviti ona strana teksta sa koje je izvršen izlazak iz EDITOR-a. Ako se EDITOR poziva prvi put, pojavljuje se prazan ekran sa trepćućim kursorom u gornjem levom uglu ekrana. Tada se može otpočeti sa unošenjem teksta. Na raspolaganju su svi znakovi abecede, numerički znaci, razne oznake i neki posebni znaci.

3.3.2. Naredbe za obradu teksta

Ovaj skup naredbi služi za pomeranje kursora po datoteci, brisanje znakova i operacije sa blokovima. Nazivi i dejstva naredbi za obradu teksta dati su u tabeli 3.1.

TABELA 3.1.

NAREDBA	DEJSTVO
CTRL →	Pomeranje kursora udesno. Ukoliko se nalazi na kraju reda kursor se pomera na početak sledećeg reda.
CTRL ←	Pomeranje kursora ulevo. Ukoliko se nalazi na početku reda kursor se pomera na kraj prethodnog reda.
CTRL ↓	Pomeranje kursora na kraj tekućeg reda. Ukoliko se već nalazi na kraju reda kursor se pomera na kraj sledećeg reda.
CTRL ↑	Pomeranje kursora na kraj prethodnog reda.
CTRL E	Skok kursora na kraj datoteke.
CTRL S	Skok kursora na početak datoteke.
CTRL K	Skok kursora na početak prethodne reči.
CTRL L	Skok kursora na početak sledeće reči.
CTRL blanko	Skok kursora na kraj reči.
CTRL A	Na mestu na kojem se nalazi kursor ubaci se blanko znak.
DEl	Brisanje znaka levo od kursora.
CTRL V	Ova naredba se koristi kada se na mestu na kojem se nalazi kursor želi umetnuti neki tekst. Ulogu kursora preuzima trepćući znak *. Sve što se od tog trenutka otkuca na tastaturi, ostaje umetnuto u tekst. Upotrebom neke od naredbi za pomeranje završava se umetanje, vraća se stari kursor, izvršava se pomeranje i rad dalje teče na uobičajen način.
LINE FEED	Skok kursora na sledeću stranu.
ESC	Skok kursora na jednu stranu unazad.
CTRL Q	Za početak strane postavi se red u kojem je kursor.
CTRL RETURN	Na mestu na kojem se nalazi kursor ubaci se znak za novi red (CR) i skače se do prve tabulacione tačke.
BREAK	Prelazak u komandni način rada.
CTRL B	Prelazak iz komandnog načina rada EDITOR-a u BASIC.

3.3.3. Naredbe za rad sa blokovima

Blok može biti bilo koji deo datoteke, može da sadrži od jednog znaka pa do nekoliko strana. Svrha bloka je da omogući jednostavno prebacivanje, brisanje ili kopiranje dela programa, čime se štedi na nepotrebnom kucanju. Za ove operacije EDITOR obezbeđuje dodatni deo memorije za rad (bafer). Dužine bafera iznosi 1 Kbajt, koliko iznosi ograničenje za veličinu bloka.

Blok je definisan svojim početkom i krajem. EDITOR uzima za početak bloka mesto na kojem se nalazi (nevidljivi) znak CTRL Y, koji se naziva marker. Kraj bloka definisan je trenutinim položajem kursora (naravno, marker mora prethoditi kursoru). Početak bloka se predefiniše tako da se na željeno mesto otkuca CTRL Y.

Nazivi i dejstva naredbi za rad sa blokovima dati su u tabeli 3.2.

TABELA 3.2.

NAREDBA	DEJSTVO
CTRL Y	Postavljanje početka bloka
CTRL U	Sadržaj bloka od markera do kursora ubacuje se u bafer
CTRL SHIFT DEL	Sadržaj bloka briše se iz datoteke
CTRL C	Zamena pozicije kursora i markera bloka
CTRL I	Sadržaj bafera ubacuje se na mesto na kojem se trenutno nalazi kursor
CTRL R	Rolovanje kursora na liniji: trenutni položaj kursora, početak datoteke i početak bloka
CTRL P	Ako se u datoteci od kursora nadalje nalazi niz znakova koji je u potpunosti identičan nizu znakova smeštenom u baferu, onda kursor skače na početak prvog takvog niza. Ako ovakvog niza u baferu nema, kursor ostaje nepomeren.
CTRL O	Prilikom pretraživanja ne pravi razliku između malih i velikih slova.

3.3.4. Komandni način rada

Naredbe komandnog načina rada i njena dejstva data su u tabeli 3.3.

TABELA 3.3

LOAD xxxx	Ovom naredbom se sa kasete u memoriju računara učita prva po redu datoteka koja je bila zapisana pod tim imenom, gde je xxxx ime datoteke.
LOAD	Na ovaj način se, bez obzira na ime, učitava prva po redu datoteka koja se nađe na kaseti.
SAVE xxxx	Ovom naredbom se sadržaj editovane datoteke snimi na kasetu pod imenom definisanim nizom xxxx. Ime datoteke je neophodno navesti.
BSAVE xxxx	Pomoću ove naredbe se sadržaj bloka snimi na kasetu pod imenom određenim nizom xxxx.
ESC	Pritiskom na dirku ESC ponovo se otkuca poslednja upotrebljena naredba u komandnom načinu rada.

ASS	Pozivanje ASEMBLER-a
PRINT	Ovom naredbom se blok definisan markerom i kursorom odštampa na štampaču.
LINE FEED	Povratak u datoteku iz komandnog načina rada.
CTRL C	Povratak iz komandnog načina rada EDITOR-a u MONITOR+.
TAB	Podešavanje tabulacije, odnosno, omogućavanje skoka kursora od jegovog trenutnog položaja do najbliže, unapred, utvrđene tabulacione tačke, vrši se na sledeći način: kada se otkuca naredba TAB na ekranu se pojavi tabulaciona osa, tj. niz oznaka od 0 do 9, kojima se označavaju kolone. Pomeranje kursora do tabulacione tačke vrši se dirkama →i←, a postavljanje ili brisanje tabulacione tačke ostvaruje se dirkom↑. Skakanje na postavljene tabulacione tačke vrši se dirkom↓. Završetak postavljanja tabulacije vrši se pritiskom na dirku BREAK.

3.4. ASEMBLER ZA MIKROPROCESOR CDP 1802

Skup svih naredbi mikroprocesora, kodiranih brojevima, čini mašinski jezik tog mikroprocesora. Program pisan na mašinskom jeziku mikroprocesor izvršava direktno, odnosno čita naredbu po naredbu iz memorije. Mašinski program je jedini program koji računar razume.

Ako svakoj naredbi, koja je predstavljena brojem, pridružimo neki naziv dobijamo simbolički mašinski jezik ili **asembler**. Simbolička imena izmišlja proizvođač mikroprocesra tako da se što slikovitije opiše šta neka naredba radi. Ako je naš program napisan na simboličkom mašinskom jeziku, da bi ga procesor razumeo moramo ga prevesti u niz brojeva i slova, odnosno u niz mašinskih instrukcija. To možemo učiniti uz pomoć nekog drugog programa, koji nazivamo asembler. Njegova uloga je da pročita simboličku instrukciju, zameni je kodom odgovarajuće mašinske instrukcije i taj broj upiše u memoriju.

Pored originalnih naredbi za mikroprocesor CDP 1802 ovaj asembler na računaru sadrži i neke dodatne (korisne) naredbe, koje se inače pojavljuju kod nekih drugih mikroprocesora napredne arhitekture. To se, pre svega, odnosi na operacije punjenja registara i memorijskih lokacija. Korišćenjem ovakvih naredbi korisnik izbegava da za jednu istu operaciju stalno ponavlja niz originalnih naredbi mikroprocesora 1802.

Asembliranje izvornog programa, koji je trenutno pod kontrolom EDITOR-a, vrši se nakon pozivanja ASEMBLER-a naredbom ASS u komandnom načinu rada. Važno je znati da se asemblirani program smešta u memoriju počev od heksadecimalne adrese 0300.

ASEMBLER prelazi preko izvornog programa dvaput. U toku prvog prelaska kreira se tabela sa imenima labela i tabela sa vrednostima labela. U toku drugog prelaza vrši se dekodiranje naredbi i upis mašinskog koda u memoriju. U slučaju neke sintaksne greške u izvornom programu ASEMBLER će prekinuti dalji rad i vratiti se u EDITOR. Pri tome se kursor postavlja na mesto na kojem je detektovana greška.

Sve naredbe EDITOR-a možemo svrstati u nekoliko grupa:
— naredbe za rad sa registrima
— naredbe za rad sa memorijom
— naredbe kratkog grananja
— naredbe dugog grananja
— naredbe preskoka
— naredba za logičke operacije
— naredbe za aritmetičke operacije
— upravljačke naredbe
— posebne naredbe.

Osim specijalnih naredbi opšti oblik naredbe u ASEMBLER-u je:
MNEMONIC (operand 1),(operand 2)
pri čemu operandi mogu biti
— registar (ili deo registra)
— parametar.
Za obeležavanje registara koriste se sledeći simboli:
— R — oznaka za registar
— n,m — heksadecimalni brojevi koji mogu imati vrednosti 0,1,...,E ili F (na primer, oznaka RA govori da je operand deseti registar iz grupe registara opšte namene)
— q,p — promenljive koje mogu imati vrednosti H (high) ili vrednost L (low). Vrednost H se uzima onda kada se želi označiti bajt veće težine, a L za bajt manje težine.

Razlikujemo tri vrste parametara:
— konstante
— labele
— matematički izraz.
—

3.4.1. Konstante

Pod pojmom konstante podrazumeva se neka jednobajtna ili dvobajtna reč. Može se izraziti u binarnom, heksadecimalnom i decimalnom kodu. Način označavanja je sledeći:
a) binarni izraz: %xxxx
nakon znaka % sledi najviše 8 (za jednobajtni) ili 16 (za dvobajtni operand) znakova 1 ili 0.
b) heksadecimalni broj: &xxxx.
Nakon znaka & slede heksadecimalni brojevi 0,1,...,E ili F (1 ili 2 znaka jednobajtne ili 1 do 4 znaka za dvobajtne operande)
c) decimalni broj: xxxx.
Vrednost operanda može se izraziti pozitivnim ili negativnim celim brojem.
d) ASCII znak: "x"
Vrednost jednobajtnog operanda jednaka je kodu znaka navedenog između navodnika.

3.4.2. Labela

Labela je niz alfanumeričkih znakova kojima se u programu označava neka memorijska lokacija. Svakoj labeli pridružena je dvobajtna vrednost koju joj ASEMBLER dodeli prilikom prvog prolaska kroz program.

3.4.3. Matematički izraz

Matematički izraz predstavljaju dva ili više operanda (labele ili konstante) koji su međusobno povezani nekom aritmetičkom ili logičkom operacijom (+,—, AND,OR,XOR).

Konačna vrednost matematičkog izraza izračunava se prilikom drugog prolaska ASEMBLER-a kroz izvorni program. Operacije se izvode redom kojim se navode. Upotreba zagrada nije moguća.

3.4.4. Naredbe za rad sa registrima

Naredbe za rad sa registrima koriste se za izmenu sadržaja registra odbrojavanjem ili upisom novog podatka. Programi na mašinskom jeziku slobodno koriste registre R8,RA,RC,RD i RE. Ako treba da se upotrebe neki drugi registri, sadržaj tih registara treba prvo da se smesti u memoriju i obnovi pre vraćanja u BASIC. Standardne naredbe za rad sa registrima date su u tabeli 3.4.

One pored simboličkog koda sadrže i mašinski kod operacije koji se može koristiti za pisanje programa u mašinskom jeziku. U tabeli 3.5. date su dodatne naredbe za rad sa registrima, koje ne sadrže mašinski kod operacije i ne mogu se koristiti za pisanje programa u mašinskom jeziku.

TABELA 3.4.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
GHI Rn ili parametar	9N	Sadržaj višeg bajta registra n smešta se u akumulator
GLO Rn ili parametar	8N	Sadržaj nižeg bajta registra n smešta se u akumulator
PHI Rn ili parametar	BN	Sadržaj akumulatora smešta se u viši bajt registra n
PLO Rn ili parametar	AN	Sadržaj akumulatora smešta se u niži bajt registra n
INC Rn ili parametar	1N	Sadržaj registra n uveća se za 1
IRX	60	Sadržaj registra X uvećava se za 1
DEC Rn ili parametar	2N	Sadržaj registra n umanji se za 1
LDI parametar	F8	Akumulator se puni vrednošću parametra

TABELA 3.5.

SIMBOLIČKI KOD	DEJSTVO
LD Rn, Rm	Sadržaj registra m prepiše se u registar n. Prvobitni sadržaj akumulatora se menja.
LD Rn.q,Rm,p	Viši (niži) bajt registra n puni se vrednošću višeg (nižeg) bajta registra m.
LD Rn.q, A	Viši (niži) bajt registra n puni se vrednošću akumulatora.
LD Rn.q, parametar	Viši (niži) bajt registra n puni se vrednošću parametra. Prvobitni sadržaj akumulatora se menja.
LD Rn, parametar	Registar n puni se vrednošću parametra. Po završetku ope racije sadržaj akumulatora se menja.
LD A, Rn.q	Akumulator A puni se sadržajem višeg (nižeg) bajta registra n. Ova naredba ekvivalentna je naredbama PHI i PLO
LD A, parametar	Akumulator A puni se vrednošću parametra. Ova naredba je ekvivalentna naredbi LDI.
PUSH Rn	Sadržaj registra n smešta se u stek.
PUSH Rn.q	Sadržaj višeg (nižeg) bajta registra n smešta se u stek.
PUSH A	Sadržja akumulatora smešta se u stek.
POP Rn	Sadržaj registra n obnavlja se iz steka.
POP Rn.q	Sadržaj višeg (nižeg) bajta registra n obnavlja se iz steka.
POP A	Sadržaj akumulatora A obnavlja se iz steka.

3.4.5. Naredbe kratkog grananja

Ovim naredbama vrši se grananje programa na memorijske lokacije unutar tekuće strane. Sve naredbe iz ove grupe su standardne naredbe mikroprocesora CDP 1802. U tabeli 3.6. dati su simbolički i mašinski kodovi ovih naredbi.

TABELA 3.6.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
BR parametar	30	Vrši se bezuslovni skok na adresu na tekućoj strani pri čemu je sadržaj nižeg bajta adrese jednak vrednosti parametra.
BNZ parametar	3A	Kratak skok ako sadržaj akumulatora nije jednak nuli.
BZ parametar	32	Kratak skok ako je sadržaj akumulatora D jednak nuli.

BDF parametar	33	Kratak skok ako je DF registar na 0
BNF parametar	3B	Kratki skok ako je DF resetovan.
BQ parametar	31	Kratki skok ako je Q setovan.
BNQ parametar	39	Kratki skok ako je Q resetovan.
B1 parametar	34	Kratki skok ako je EF1 setovan.
BN1 parametar	3C	Kratki skok ako je EF1 resetovan.
B2 parametar	35	Kratki skok ako je EF2 setovan.
BN2 parametar	3D	Kratki skok ako je EF2 resetovan.
B3 parametar	36	Kratki skok ako je EF3 setovan.
BN3 parametar	3E	Kratki skok ako je EF3 resetovan.
B4 parametar	37	Kratki skok ako je EF4 setovan.
BN4 parametar	3F	Kratki skok ako je EF4 resetovan.

3.4.6. Naredbe dugog grananja

Kod naredbi dugog grananja operand je dvobajtan i predstavlja adresu lokacije u memoriji na koju se eventualno vrši skok. I ove naredbe su iz grupe standardnih naredbi mikroprocesora CDP i 802.

U tabeli 3.7. dati su simbolički i mašinski kodovi ovih naredbi.

TABELA 3.7.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
LBR parametar	C0	Bezuslovni skok na lokaciji čija je adresa jednaka vrednosti parametra.
LBZ parametar	C2	Dugi skok ukoliko je vrednost sadržaja akumulatora D jednaka nuli.
LBNZ parametar	CA	Dugi skok ukoliko sadržaj akumulatora D nije nula.
LBDF parametar	C3	Dugi skok ukoliko je DF setovan.
LBNF parametar	CB	Dugi skok ukoliko je DF na nuli.
LBQ parametar	C1	Dugi skok ukoliko je Q setovan.
LBNQ parametar	C9	Dugi skok ukoliko je Q resetovan.

3.4.7. Naredbe preskoka

Naredbe preskoka koriste se za izvođenje uslovnog ili bezuslovnog preskoka i mogu biti u kratkom i dugom formatu. Simbolički i mašinski kodovi naredbi preskoka dati su u tabeli 3.8.

TABELA 3.8.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
SKP	38	Naredba kratkog preskoka.
LSKP	C8	Naredba dugog preskoka.

LSZ	CE	Dugi preskok ukoliko je vrednost u akumulatoru D nula.
LSNZ	C6	Dugi preskok ukoliko vrednost u akumulatoru nije nula.
LSDF	CF	Dugi preskok ukoliko je DF na logičkoj jedinici.
LSNF	C7	Dugi preskok ukoliko je DF na logičkoj nuli.
LSNQ	CD	Dugi preskok ukoliko je Q setovan
LSNQ	C5	Dugi preskok ukoliko je Q resetovan.
LSIE	CC	Dugi preskok ukoliko je IE na logičkoj jedinici.

3.4.8. Naredbe za rad sa memorijom

Naredbe ove grupe koriste se za upis i čitanje podataka iz memorije. Simbolički i mašinski kodovi naredbi za rad sa memorijom dati su u tabeli 3.9.

TABELA 3.9.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
LDN Rn ili parametar	0N	Memorijski bajt adresiran sadržajem registra n prenosi se u akumulator.
LDA Rn ili parametar	4N	Memorijski bajt adresiran sadržajem registra n prenosi se u akumulator, a sadržaj registra n uvećava se za jedan.
STR Rn ili parametar	5N	Sadržaj akumulatora se prebacuje u memorijsku lokaciju adresiranu sadržajem registra n.
LDX	F0	Sadržaj memorijske lokacije adresirane sadržajem ukazatelja steka prebacuje se u akumulator.
LDXA	72	Sadržaj memorijske lokacije adresirane sadrža jem ukazatelja steka prebacuje se u akumulator, a ukazatelj steka se uveća za jedan.
STXD	73	Sadržaj akumulatora prebacuje se u memorij sku lokaciju adresiranu sadržajem ukazatelja steka, a ukazatelj steka se smanji za jedan.

3.4.9. Naredbe za logičke operacije

Naredbe ove grupe koriste se za izvođenje logičkih operacija nad dva operanda. Jedan operand se nalazi u akumulatoru, a drugi se eksplicitno navodi uz kod operacije ili se nalazi u memorijskoj lokaciji koja je adresirana sadržajem ukazatelja steka. Rezultat se smešta u akumulator. Simbolički i mašinski kodovi naredbi ove grupe dati su u tabeli 3.10.

TABELA 3.10.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
OR	F1	
ORI parametar	F9	Logička "ili" operacija
XOR	F3	
XRI parametar	FB	Operacija "ekskluzivno ili"
AND	F2	
ANI parametar	FA	Logička "i" operacija
SHR	F6	
SHRC	76	Sadržaj akumulatora se pomera za jedno mesto udesno (bez ili uz pomoć BIT-a za prenos)
SHL	FE	
SHLC	7E	Sadržaj akumulatora se pomera za jedno mesto ulevo (bez ili uz pomoć BIT-a za prenos)

3.4.10. Naredbe za aritmetičke operacije

Naredbe ove grupe se koriste za izvođenje aritmetičkih operacija nad dva operanda. Jedan operand stoji u akumulatoru, a drugi je kao parametar naveden uz kod operacije ili se nalazi u memorijskoj lokaciji na koju ukazuje ukazatelj steka. Simbolički i mašinski kodovi ovih naredbi dati su u tabeli 3.11.

TABELA 3.11.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
ADD	F4	
ADI parametar	FC	
ADC	74	
ADCI parametar	7C	Sabiranje (bez i sa prenosom)
SD	F5	
SDI parametar	FD	
SDB	75	
SDBI parametar	7D	Oduzimanje od akumulatora (bez i sa pozajm ljivanjem)
SM	F7	
SMI parametar	FF	
SMB	77	
SMBI parametar	7F	Oduzimanje od memorije (bez i sa pozajm ljivanjem)

3.4.11. Upravljačke naredbe

Upravljačke naredbe omogućavaju upravljanje prekidom, grananje i povezivanje potprograma i postavljanje stanja kontrolnog bita Q. Simbolički i mašinski kodovi ovih naredbi dati su u tabeli 3.12.

TABELA 3.12.

SIMBOLIČKI KOD	MAŠINSKI KOD	DEJSTVO
SEP Rn ili parametar	DN	Registar n postavlja se kao programski brojač.
SEX Rn ili parametar	EN	Registar n postavlja se kao ukazatelj memorijske lokacije M(R(x)).
IDLE	00	Čeka se DMA ili prekid.
SEQ	7B	Postavlja se Q flip-flop.
REQ	7A	Resetuje se Q flip-flop.
SAV	78	Sadržaj registra T upisuje se u memorijsku lokaciju adresiranu sa R(x).
MARK	79	Sadržaj registra X i P upisuje se u registar T i u stek.
RET	70	Memorijski bajt M(R(X)) prenosi se u registre X i P. Flip-flop. za dozvolu prekida IE se resetuje
DIS	71	Memorijski bajt M(R(X)) prenosi se u registre X i P. Flip-flop za dozvolu prekida se setuje.
NOP	C4	Naredba bez dejstva.

3.4.12. Specijalne naredbe

Naredbe iz ove grupe mogu se koristiti samo za rad u asembleru (ne spadaju u grupu standardnih naredbi za mikroprocesor 1802). Simbolički kodovi ovih naredbi dati su u tabeli 3.13.

TABELA 3.13.

SIMBOLIČKI KOD	DEJSTVO
CALL parametar	Ovom naredbom se poziva potprogram na adresi jednakoj vrednosti parametra.
RETS	Naredba za povratak iz potprograma u glavni program.
BYTE parametar 1,...,parametar	Kada ASEMBLER naiđe na ovu naredbu, on će vrednosti navedenih parametara ubaciti u memorijske lokacije koje su u programu. Pri tome broj parametara nije ograničen.
WORD parametar 1,.,parametar 2	Ova naredba je slična naredbi BYTE, s tom razlikom što parametri predstavljaju dvobajtne vrednosti koje se pakuju unutar programa.

TEXT "niz znakova"	Ova naredba služi za kreiranje tabela. Prilikom asembliranja u memoriji se pakuju kodovi znakova navedenih u nizu.
TEXTO "niz znakova"	Prilikom asembliranja na kraju niza postavi se znak END OF FILE (kod O0). Ova naredba se koristi sa rutinom (Potprogramom) za štampanje nizova znakova.

3.4.13. Posebne naredbe ASEMBLER-a

EQU

To je specijalna naredba koja služi za definisanje vrednosti labela. Format naredbe je:

```
Labela: EQU parametar
```

Ova naredba se koristi na početku programa. Na primer:

```
A900: EQU & 4800
```

Ograničenje za ime labele je da ima najviše šest znakova, s tim što prvi znak mora da bude alfanumerički.

ORG

Ova naredba obavezno dolazi na početak svakog programa, njome se definiše adresa memorijske lokacije na kojoj se asemblira program. Format naredbe je sledeći:

```
ORG parametar 1 (parametar 2)
```

gde je:

— parametar 1 — adresa na koju se asemblira
— parametar 2 — relocirana adresa na koju može ASEMBLER da asemblira program.

Na primer, naredba

ORG & 80BF, &5000

znači da se program koji sledi asemblira na adresu &80BF. Adresa &8-BF kod PECOM-a 64 je u stvari adresa EPROM-a. Na ovaj način korisnik može da pripremi sadržaj za eventualnu promenu sadržaja EPROM-a.

Program se istovremeno smešta od adrese &5000 u korisničku oblast (RAM).

OPT n

Ovom naredbom se bira jedan od načina ASEMBLER-a. Zato je to obavezno prva naredba programa. Broj n može da uzima vrednosti od 0 do 15. Predstavimo ga u binarnom kodu:

```
N=(D3) (D2) (D1) (D0)
```

Način rada zavisi od vrednosti pojedinih bitova:

D0=1 prevedeni kod se upisuje u memoriju
D1=1 listing datoteke se štampa na ekranu
D2=1 na kraju listinga datoteke napravi se tabela upotrebljenih labela
D3=1 listing datoteke se štampa na štampaču.

Na primer, ako se upotrebi naredba OPT 1, nakon poziva ASEMBLER-a, prevedeni kod će se upisati u memoriju, a ukoliko ima grešaka one će se pojaviti na ekranu. Ako se koristi naredba OPT 15, prevedeni kod se upisuje u memoriju, listing datoteke se prikazuje na ekranu sa tabelom upotrebljenih labela, a listing datoteke se štampa na štampaču.

SUBS

Ova naredba se koristi u slučaju da se u programu treba zameniti naziv neke labele. Format naredbe je

```
SUBSb:TEXT1:TEXT2
```

gde je:

```
SUBS - naziv naredbe
b - oznaka za blanko
TEXT1 - niz koji se zamenjuje
TEXT2 - niz kojim se vrši zamena.
```

3.5. KORIŠĆENJE ĆIRILICE

Ako se želi ispisivanje teksta na ekranu ćirilicom potrebno je da prva naredba programa, napisanog u BASIC-u, bude:

```
:1 CALL (&CE00)
```

Normalno, broj linije može biti bilo koji ali je bitno da naredba CALL bude pre prve PRINT naredbe u programu. Posle startovanja programa sa RUN, tekst koji se prikazuje na ekranu biće ispisan ćirilicom. Ako se dalje želi neka izmena u programu (na primer, dodavanje nekih naredbi ili slično) potrebno je preći u režim rada latinicom na sledeći način:

```
:CALL(&CE70)
```

3.6. PRIMERI PROGRAMA NA ASEMBLER-u

Primer: Napisati program za smeštanje konstante 1234 u registar RA. Sadržaj registra RA uvećan za jedan preneti u registar RB. Sadržaj registra RB preneti u memorijske lokacije sa adresama 6000 i 6001.

Adresu memorijske lokacije u koju se upisuje sadržaj registra RB smestićemo u registar R8.

Unošenje programa vršimo u režimu EDITOR-a, počev od osme kolone, ostavljajući 7 kolona za labele.

```
OPT 7
ORG &4000
START:
LD RA, & 1234
LD R8, & 6000
```

```
INC RA
LD RB, RA
GHI RB
STR R8
INC R8
GLO RB
STR R8
RETS
```

Ovako napisan program treba asemblirati. To činimo tako što prvo prelazimo u komandni način rada EDITOR-a pritiskom na dirku BREAK. Kao znak da smo prešli u komandni način rada EDITOR-a javlja se horizontalna crta sa trepćućim kursorom. Naredbom ASS vrši se pozivanje ASEMBLER-a.

Program koji se unosi u ASEMBLER-u smešta se u memoriju počev od adrese 0300. To treba imati na umu kada se bira početna izvršna adresa programa. Ako bismo uzeli za početnu izvršnu adresu programa 0300, tada bi on bio uništen.

Program koji se dobija posle asembliranja je sledeći:

```
ASS
0000                        OPT 7
4000                        ORG &4000
4000                        START
4000F812BAF834AA            LD RA, &1234
4006F860B8F800A8            LD R8, &6000
400C 1A                     INC RA
400D 9ABB8AAB               LD RB, RA
4011 9B                     GHI RB
4012 58                     STR R8
4013 18                     INC R8
4014 8B                     GLO RB
4015 58                     STR R8
4016 D5                     RETS
START                  4000
```

Izvršenje programa vrši se u režimu rada MONITOR-a, u koji se vraćamo istovremenim pritiskom na dirke CTRL i C. Za startovanje programa u mašinskom kodu koristi se naredba C. Na pitanje FROM unosi se početna adresa programa, 4000.

```
>C
FROM 4000
>
```

Kada se izvrši program u memorijske lokacije sa adresama 6000 i 6001 se upisuju podaci 12 i 35. Tačnost izvršenja programa možemo proveriti prikazivanjem sadržaja memorijskih lokacija sa adresama od 6000 nadalje. U tu svrhu koristimo naredbu D.

```
>D
FROM 6000
6000 12 35 00 00 00 00 00 00
```

Primer: Od adrese 5000 uneti sledeće podatke

adresa	podatak
5000	01
5001	23
5002	45
5003	AB
5004	CD
5005	EF

Napisati i istestirati program za prenošenje prva tri bajta u istom redosledu, a sledeća tri bajta u obrnutom redosledu u novo polje sa startnom adresom 500A. Za adresiranje polja koristiti registre RA i RB. Rezultat obrade je:

adresa	podatak
500A	01
500B	23
500C	45
500D	EF
500E	CD
500F	AB

Memorijske lokacije počev od 5000 treba napuniti podacima 01 23 45 AB CD EF. To činimo naredbom I iz režima MONITOR-a:

```
I
FROM 5000
5000 01 23 45 AB CD EF 00 00
```

Naredbom E prelazimo u režim EDITOR-a i vršimo unošenje programa:

```
OPT 7
ORG &4000
START
LD RA,&5000          ...u RA smeštamo adresu 5000
LD RB, &500A         ...u RB smeštamo adresu 500A
LDA RA               ... sadržaj memorijske lokacije čija je adresa sme
                     štena u RA prenosi se u akumulator D. Nadalje ko-
                     ristimo skraćenicu M(RA)→D, RA+1→RA
STR RB               ... 1. podatak u M(RB)
INC RB               ... RB+1→RB
LD RA                ... 2. podatak u D, RA+1→RA
STR RB               ... 2. podatak u M(RB)
INC RB
LDA RA               ... 3. podatak u D, RA+1→RA
STR RB               ... 3. podatak u M(RB)
INC RB
INC RB
INC RB
SEX RB               ...X=B
LDA RA               ... 4. podatak u D
STXD                 ... 4. podatak u M(RB), RB-1→RB
LDA RA
STXD                 ... 5. podatak u M(RB), RB-1→RB
```

```
LDN RA
STR RB                          ... 6. podatak u M(RB)
RETS
```

Posle asembliranja program je sledeći:

```
ASS
0000                            OPT 7
4000                            ORG &4000
4000                            START:
4000F850BAF800AA                LD RA, &5000
4006F850BBF80AAB                LD RB, &500A
400C 4A                         LDA RA
400D 5B                         STR RB
400E 1B                         INC RB
400F 4A                         LDA RA
4010 5B                         STR RB
4011 1B                         INC RB
4012 4A                         LDA RA
4013 5B                         STR RB
4014 1B                         INC RB
4015 1B                         INC RB
4016 1B                         INC RB
4017 EB                         SEX RB
4018 4A                         LDA RA
4019 73                         STXD
401A 4A                         LDA RA
401B 73                         STXD
401C 0A                         LDN RA
401D 5B                         STR RB
401E D5                         RETS
START                           4000
```

Program startujemo iz MONITOR-a+:

```
C.
FROM 4000
```

Za proveru ispravnosti programa koristimo naredbu D:

```
D
FROM 5000
5000 01 23 45 AB CD EF 00 00
5008 00 00 01 23 45 EF CD AB
```

4. Računarska učionica (RU)

Računarska učionica je nastala kao logična posledica usavršavanja nastavnih metoda iz predmeta programiranja u osnovnom i srednjoškolskom obrazovanju, gde se javila potreba za istovremenim radom više korisnika sa glavnim računarom i korišćenjem svih njegovih mogućnosti.

Na sl.4.1. prikazana je blok šema organizacije jedne računarske učionice. Takva jedna konfiguracija sadrži:

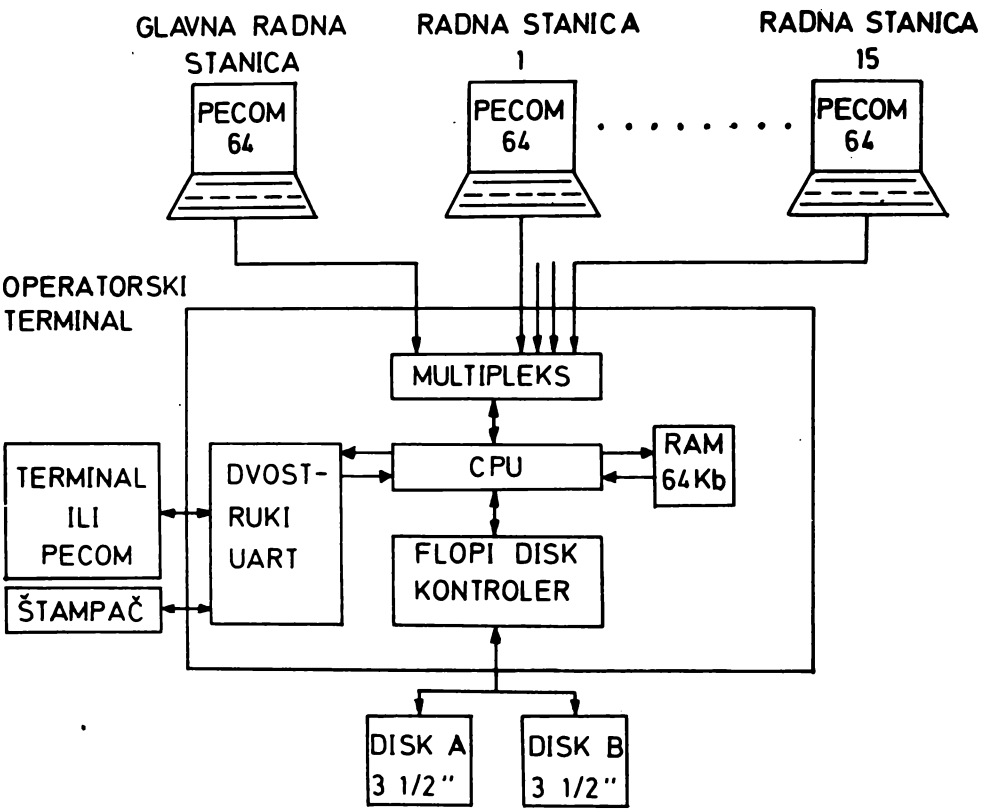
- modularni mikroračunarski sistem MMS 1800 RU kao glavni računar koji sadrži i dodatne module za povezivanje radnih stanica (MULTIPLEKS) i za spregu radnih stanica sa centralnom jedinicom glavnog računara
- maksimalno 16 radnih stanica (PECOM-a 64 sa monitorom ili TV prijemnikom) od kojih je jedna glavna (nastavnička) radna stanica
- operatorski terminal, koji može biti PECOM 64 sa monitorom ili asinhroni video terminal VIP 7251P sa standardnom tastaturom
- dvostruka disk jedinica, koja koristi IBM kompatibilne diskete veličine 3,5 inča kapaciteta 1 Mbajt.
- serijski štampač sa interfejsom RS 232C (sa 80 ili 132 znaka u koloni), gde se štampanje vrši preko nastavničke radne stanice. U autonomnom režimu rada svaka radna stanica može da koristi isti štampač.

4.1. RAD RAČUNARSKE UČIONICE

Ispitaćemo postupak uspostavljanja rada računarske učionice po određenim koracima aktivnosti:

— KORAK 1

Priključujemo raspoloživ broj radnih stanica (maksimalno 16) na glavni računar, operatorski terminal (PECOM 64) povezujemo sa MMS 1800 RU preko konektora sa oznakom TERMINAL i priključujemo na sistem serijski štampač i dvostruku flopi disk jedinicu.



4.1

— KORAK 2

Aktiviramo rad računarske učionice startovanjem programa za terminalski rad sa operatorskog terminala (PECOM 64). Program startujemo iz režima rada MONITOR-a+ unošenjem sa tastature naredbe

```
>C
```

Pojaviće se poruka

```
FROM CEE0
```

a posle pritiska dirke RETURN kursor će se naći u prvoj lokaciji na mestu karaktera F.

— KORAK 3

Vršimo inicijalizaciju sistema na taj način što se pritisnu tasteri RESET i RUN-u na prednjoj ploči centralne jedinice MMS 1800 RU.

— KORAK 4

Sada treba učitati operativni sistem sa diskete. Sistemsku disketu stavljamo u drajv sa oznakom 0, a sa tastature se unosi naredba $C. Ukoliko je uspostavljena komunikacija između operatorskog terminala i centralnog računara počinje učitavanje operativnog sistema (svetli dioda na disketnoj jedinici 0).

Ako komunikacija nije uspostavljena, na ekranu operatorskog terminala prikazuje se samo prvi karakter $. U tom slučaju treba proveriti da li je kablom dobro uspostavljena veza operatorskog terminala sa centralnim računarom.

Ako smo prethodnim postupkom uspostavili komunikaciju, onda smo startovali učitavanje operativnog sistema. Posle učitavanja na operatorskom terminalu se pojavljuje poruka:

```
CDOS VI.0
```

— KORAK 5

Sada pristupamo učitavanju programa za rad računarske učionice, koji se takođe nalazi na sistemskoj disketi. Program se poziva unošenjem naredbe RU i pritiskom na dirku RETURN.

Nakon učitavanja programa RU uspostavljamo automatsko startovanje i na ekranu se pojavljuje poruka:

```
CK.DRIVE
PRESS ANY KEY
TO START PROGRAM
```

Ova poruka znači da radnu disketu treba postaviti u drajv jedinicu 1, i pritiskom bilo kog tastera na operatorskom terminalu, sistem računarske učionice spreman je za rad. Ukoliko je radna disketa već bila postavljena u drajvu 1, poruka se ne pojavljuje, već je program za rad u računarskoj učionici automatski startovan.

— KORAK 6

U ovom koraku centralni računar selektuje pojedinačno radne stanice.

— KORAK 7

Centralna jedinica testira da li je selektovana radna stanica postavila zahtev za obradu. Ukoliko zahtev ne postoji selektuje se sledeća radna stanica, a posle selekcije poslednje počinje se od početka selekcijom nulte radne stanice. Ako je selektovana radna stanica postavila zahtev za obradu, centralni računar prelazi u režim prijema zahteva (korak 8).

KORAK 8

Kada se primi kompletni zahtev, vrši se testiranje ispravnosti prenetog zahteva. Ako je zahtev ispravan prelazi se na odgovarajući program, koji je naveden u zahte-

vu. Ukoliko zahtev nije ispravan, centralni računar vraća radnoj stanici, koja je uputila zahtev, informaciju da zahtev nije uspešno primljen.

Ovde treba istaći da sve radne stanice osim glavne imaju ravnopravnu ulogu u radu računarske učionice i jednak nivo prioriteta. Glavna radna stanica je namenjena nastavniku i ima dodatne mogućnosti i dodatni broj funkcija.

4.2. DATOTEKE I NAZIVI DATOTEKA

U daljem razmatranju naredbi za rad u računarskoj učionici stalno ćemo pominjati termin "datoteka". Da bismo što lakše shvatili problematiku definisaćemo datoteku. Radi se o skupu povezanih podataka koje smo objedinili u jednu celinu i smestili na zajednički memorijski medijum.

Osnovna datoteka koja se smešta na disketu identifikuje se nazivom koji ima 8 alfanumeričkih znakova. Nazive datoteka kreira sam korisnik. Naziv datoteke sastoji se od jednog do šest alfanumeričkih znakova a proširenje se sastoji od dva alfanumerička znaka. Prvi znak naziva i proširenja mora biti slovo. Standardni format za naziv datoteke je:

```
NAZIV.SUF
```

gde je:

```
NAZIV - naziv od 1 do 6 znakova
SUF - proširenje 1 do 2 znaka
```

Imenik (direktorijum) je datoteka smeštena na pisti 0 svih disketa. U direktorijumu se vodi računa o imenu i sufiksu datoteke. Svaka priključena radna stanica poseduje svoj imenik sa nazivima svih datoteka smeštenih na disketi. Primeri korektno napisanih naziva datoteka su:

```
PR1.SR, PRIM.H, P10.H1, DIODA.BA i sl.
```

4.3. NAREDBE ZA RAD U RAČUNARSKOJ UČIONICI

U ovom delu biće obrađene naredbe koje stoje na raspolaganju korisnicima za rad u računarskoj učionici. Obratimo pažnju na sl.1. gde je prikazana blok šema organizacije jedne kompletne računarske učionice. Zapažamo da PECOM 64 može vršiti ulogu operatorskog terminala, radne stanice (mesto učenika) ili glavne radne stanice (radne stanice nastavnika). Treba istaći da je PECOM 64 tako opremljen da može obavljati bilo koju od ovih uloga. U zavisnosti od toga, sa kojeg dela sistema računarske učionice sa naredbe generišu, može se izvršiti sledeća podela:

- naredbe koje aktivira operatorski terminal
- naredbe koje se aktiviraju sa bilo koje radne stanice (mesto učenika)
- naredbe koje se aktiviraju sa radne stanice nastavnika

4.3.1. Naredbe koje se aktiviraju sa operatorskog terminala

Diskete, koje nazivamo radnim, a koje služe za smeštanje programa koje sam korisnik formira, su neformatirane. Kao takve one su neupotrebljive. Zato ih, pre bilo kakve njihove upotrebe, treba formatirati i inicijalizovati.

Formatirati disketu znači formirati je tako da sadrži 77 krugova za smeštanje podataka (pisti) numerisanih od 0 do 76. Svaka pista se podeli na 32 jednaka dela (sektora) numerisanih od 1 do 32. Posle formatiranja svaki sektor sadrži podatke kojima se pronalazi pista i sam sektor.

Inicijalizovati disketu znači dodeliti joj datum njenog kreiranja u formatu koji se traži i naziv, kako bi je pronalazili u kasnijem radu.

Postupak formatiranja i inicijalizacije diskete vrši se preko operatorskog terminala.

Naredbom $C vršimo učitavanje operativnog sistema sa diskete smeštene u drajvu 0. Kao rezultat toga dobija se poruka na ekranu:

```
>CDOS V1.0
>
```

Disketu koju formatiramo stavljamo u drajv 1 a sa tastature unosimo naredbu za formatiranje:

```
>FORM (RETURN)
```

Posle učitavanja ovog programa štampa se poruka:

```
PRESS ANY KEY TO START FORMATTING IN DRIVE 1?
```

(pritisnite bilo koju dirku za startovanje formatiranja u drajvu 1).

Nakon pritiska na bilo koju dirku počinje formatiranje nove diskete. Posle formatiranja štampa se poruka:

```
>END
```

Sada se sa tastature može pozvati program za inicijalizaciju diskete:

```
>SYSGEN
```

Na ekranu će se pojaviti poruka:

```
INPUT USERID>
```

posle koje unosimo naziv diskete i pritisnemo dirku RETURN. Ispod prethodne pojaviće se nova poruka:

```
INPUT DATE (MM/DD/YY)
```

posle koje unosimo mesec, dan i godinu inicijalizacije. Nakon toga unosimo

```
S (RETURN)
C(RETURN)
IS IT OK TO COPY TO DRIVE 1?
```

Na postavljeno pitanje treba odgovoriti sa Y. Na ovaj način počinje inicijalizacija prethodno formatirane diskete. Po završetku inicijalizacije unosi se komanda QUIT:

```
Q (RETURN)
```

čime se kontrola vraća na operativni sistem.

Naredbom RU aktiviramo rad računarske učionice. Sa tastature unosimo:

```
>RU (RETURN)
```

Sada se najpre vrši učitavanje programa za rad u računarskoj učionici, posle čega je centralni računar MMS 1800 RU opremljen za rad. Priključenjem radnih stanica u mrežu i slanjem raznovrsnih zahteva sistem praktično počinje sa radom.

4.3.2. Naredbe koje mogu da se aktiviraju sa bilo koje radne stanice (sa učeničkog PECOM-a)

U prethodnim poglavljima analizirali smo režime rada PECOM-a 64 i zaključili smo da on može raditi u režimu: BASIC, MONITOR+, i EDITOR. Na osnovu ovog zaključka može se izvršiti podela naredbi za rad PECOM-a, kao radne stanice u računarskoj učionici, na:

— naredbe koje se pozivaju iz BASIC-a
— naredbe koje se pozivaju iz komandnog režima EDITOR-a
— naredbe koje se pozivaju iz MONITOR-a+

Naredbe koje se pozivaju iz BASIC-a

LOGIN

Ovom naredbom se radna stanica programski priključuje u sistem računarske učionice. To je naredba koja treba prva da se unese sa radne stanice učenika, pre bilo kakvog rada. Naredba LOGIN omogućava da radna stanica povremeno vrši testiranje ulazne linije i na taj način ispituje da li ima poruka sa drugih stanica učenika ili sa radne stanice nastavnika.

LOGOUT

Ovom naredbom se prekida programska veza radne stanice sa ostalim sistemom radne stanice. Kada se unese ova naredba sa radne stanice učenika, prekida se testiranje ulazne linije, čime se praktično prekida veza sa ostalim delom sistema.

PGREAD

Ovom naredbom se čita program napisan u BASIC-u, a koji se nalazi na disketi u datoteci čiji je naziv dat u samoj naredbi. Format ove naredbe je:

```
PGREAD"nbNAZIV.SUFIX"
```

gde je:

n — broj kanala kome pripada datoteka koja se čita (n=0 do 15)

b — oznaka za blanko

NAZIV.SUFIX — naziv datoteke određuje sam korisnik prema pravilima za formiranje imena datoteke

Ukoliko u naredbi nije određen broj kanala n, čita se datoteka kanala na kojem radi korisnik.

Primer: Pročitati datoteku pod nazivom MEMO.VI koja pripada sedmom kanalu.

Rešenje:

```
:PGREAD "7 MEMO.VI" (RETURN)
READY
:
```

DREAD

Ovom naredbom se sa diskete čita datoteka podataka za BASIC program pod nazivom definisanim u samoj naredbi. Format ove naredbe je:

```
DREAD"nbNAZIV.SUFIX"
```

Značenja promenljivih su ista kao za naredbu PGREAD.

Primer: Pročitati datoteku podataka smeštenu na disketi pod nazivom PECOM.AA. Datoteka se formira na 3.kanalu.

Rešenje:

```
:DREAD "3 PECOM.AA" (RETURN)
```

WRITE

Ovom naredbom se na disketu upisuje program u BASIC-u sa one radne stanice na kojoj je korisnik napravio taj program. Program se upisuje u datoteku pod nazivom koji je zadat u samoj naredbi. Format naredbe je:

```
WRITE"NAZIV.SUFIX"
```

gde je

NAZIV.SUFIX — ime datoteke koje utvrđuje sam korisnik po pravilima za formatiranje ime datoteke

DWRITE

Ovom naredbom se na disketi vrši, upis datoteke podataka za određeni BASIC program. Treba imati u vidu da se datoteka podataka odnosi na tačno određeni BASIC program, koji je prethodno, takođe, bio upisan na disketi pod određenim nazivom. Format naredbe je:

```
DWRITE"NAZIV.SUFIX"
```

gde je:

NAZIV.SUFIX — ime datoteke koju utvrđuje sam korisnik.

Primer: Datoteka BASIC programa pod imenom NAZIV, PR upisuje se na disketu. Upisati na disketu datoteku podataka pod nazivom NAZIV.PI, koja se odnosi na datoteku programa NAZIV.PR.

Rešenje:

```
:WRITE "NAZIV.PR" (RETURN)
:DWRITE "NAZIV.PI" (RETURN)
```

MSG

Ovom naredbom je omogućena komunikacija između radnih stanica slanjem određenih poruka. Dozvolu ili zabranu slanja ovakvih poruka daje radna stanica nastavnika. Radna stanica koja primi poruku daje informaciju o tome čija je poruka i prikazuje je na ekranu.

Format naredbe je:

```
MSG "nbTEXT"
```

gde je:

n — broj kanala kome se šalje poruka

TEXT — tekst sadržan u poruci. Dužina poruke je maksimalno 95 znakova.

Primer: Trećoj radnoj stanici poslati poruku tekstom JA SAM SPREMAN.

Rešenje:

```
:MSG "3 JA SAM SPREMAN" (RETURN)
```

LPRINT

Ovom naredbom se ispisuje tekst ili rezultat izračunavanja na ekranu i na štampaču. Ova naredba je identična naredbi LPRINT za rad samog PECOM-a sa štampa-

čem. Kada se naredba LPRINT koristi u režimu rada računarske učionice, radna stanica je povezana na centralni računar, na koji je priključen štampač. Pre unošenja naredbe LPRINT, sa radne stanice se izdaje naredba LOGIN. Format naredbe je:

LPRINT "TEXT"

gde je:

TEXT — izraz niza, numerička promenljiva ili numerički izraz.

LLIST

Ovom naredbom se listing BASIC programa prikazuje na ekranu i na štampaču. Za korišćenje naredbe LLIST u računarskoj učionici važe sva pravila kao i za naredbu LPRINT. Format naredbe je:

```
LLIST
```

Naredbe koje se pozivaju iz komandnog režima EDITOR-a

READ

Ova naredba na disketi omogućava čitanje datoteke koja je smeštena pod određenim nazivom. Slična je naredbi PGREAD koja se aktivira iz režima rada BASIC-a. Međutim, postoji jedna suštinska razlika. Pomoću naredbe READ mogu se čitati datoteke bilo kakvih tekstova ili asemblerskih listinga. Format naredbe je:

```
READbnbNAZIV.SUFIX
```

gde je:

n — broj kanala kome pripada datoteka koja se čita

b — oznaka za blanko.

NAZIV.SUFIX — naziv datoteke koji utvrđuje sam korisnik

Primer: Pročitati datoteku sa nazivom PROBA1.AS koja pripada trećoj radnoj stanici.

Rešenje:

```
READ 3 PROBA1.AS (RETURN)
```

WRITE

Ovom naredbom se datoteke editovane iz komandnog režima EDITOR-a smeštaju na disketu. To mogu biti: datoteke teksta, datoteke izvornog koda ili asemblirane datoteke. Format naredbe je:

```
WRITEbNAZIV.SUFIX
```

gde je:

NAZIV.SUFIX — naziv datoteke koji određuje korisnik

b — oznaka za blanko.

BWRITE

Ovom naredbom se jedan blok datoteke, kreiran u režimu rada EDITOR-a, smešta na disketu. Početak bloka se postavlja sa CTRL i Y, a kraj bloka je trenutna pozicija kursora. Format naredbe je:

```
BWRITEbNAZIV.SUFIX
```

gde je:

NAZIV.SUFIX — naziv bloka datoteke koji dodeljuje sam korisnik

b — oznaka za blanko

MSG

Ovom naredbom se šalje poruka nekoj od priključenih radnih stanica. Format naredbe je:

```
MSGbnbTEXT
```

gde je:

TEXT — tekst sadržan u poruci, maksimalne dužine 95 znakova
n — broj radne stanice kojoj se šalje poruka
b — oznaka za blanko.

Primer: Radnoj stanici 9 iz komandnog režima EDITOR-a poslati poruku ZDRAVO PERO.

Rešenje:

```
MSG 9 ZDRAVO PERO (RETURN)
```

DIR

Svaka radna stanica u sistemu računarske učionice ima vlastiti imenik sa nazivima svih datoteka smeštenih sa te radne stanice. Ovom naredbom se omogućuje prikazivanje naziva iz imenika određene radne stanice, čiji se broj definiše u samoj naredbi. Format naredbe je:

```
DIRbn
```

gde je:

b — oznaka za blanko
n — broj kanala čiji se imenik traži.

Ako je umesto broja n navedena zvezdica (*) prikazuju se imenici svih kanala, a format naredbe dobija oblik:

```
DIR *
```

DLOCK

Ovom naredbom se postavlja zabrana brisanja datoteke smeštene pod nazivom određenim u samoj naredbi. Zabrana od brisanja važi samo za one datoteke koje su generisane sa radne stanice koja zabranu izdaje. Format naredbe je:

```
DLOCKbNAZIV.SUFIX
```

DUNLOCK

Ovom naredbom se dozvoljava brisanje datoteke pod nazivom određenim u samoj naredbi. Dozvolu je moguće dati samo sa radne stanice koja je datoteku pod tim nazivom generisala. Format naredbe je

```
DUNLOCKbNAZIV.SUFIX
```

REN

Ova naredba omogućava promenu naziva datoteke one radne stanice koja je datoteku i generisala. Format naredbe je:

```
RENbNAZIV.SUFIXbNAZIV1.SUFIX
```

gde je:

NAZIV.SUFIX — stari naziv datoteke
NAZIV1.SUFIX — novi naziv datoteke
b — oznaka za blanko.

Primer: Naziv datoteke PROG.PR zameniti nazivom PECOM.H1
Rešenje:

```
REN PROG.PR PECOM.H1
```

DEL

Ovom naredbom se briše jedan ili više naziva datoteka iz imenika određene radne stanice. Na ovaj način se dozvoljava pristup svim delovima koji su pripadali obrisanoj datoteci ili obrisanim datotekama. Format naredbe je:

```
DELbNAZIV.SUFIX
```

gde je:
NAZIV.SUFIX — naziv datoteke koja se briše
b — oznaka za blanko.
Primer: Obrisati iz imenika datoteke pod nazivom PECOM.OK.
Rešenje:

```
DEL PECOM.OK
```

COPY

Ovom naredbom se ostvaruje funkcija kopiranja datoteke sa odabrane radne stanice na radnu stanicu koja izdaje naredbu COPY. Format naredbe je:

```
COPY  bnb NAZIV.SUFIXbNAZIV1.SUFIX
```

gde je:
NAZIV.SUFIX — naziv datoteke koja se kopira
NAZIV1.SUFIX — naziv datoteke koja se dobija kopiranjem
n — broj kanala radne stanice sa koje se vrši kopiranje
b — oznaka za blanko.
Primer: Sa radne stanice 3 kopirati datoteku PECOM.SR u datoteku sa nazivom PEKP.H
Rešenje:

```
COPY 3 PECOM.SR PEKP.H
```

PRINT

Ovom naredbom se štampa blok datoteke, koji je kreiran u EDITOR-u, na štampaču sa bilo koje radne stanice. Format naredbe je:

```
PRINT
```

Naredbe koje se pozivaju iz MONITOR-a+

Iz režima rada MONITOR+ mogu se pozivati naredbe za upis i čitanje programa na mašinskom jeziku.

Naredba Q

Ova naredba vrši upis određenog memorijskog sadržaja na disketi, pod određenim nazivom.
Postupak je sledeći: nakon unošenja sa tastature

```
> Q
```

pojavljuje se poruka

NAME

nakon čega korisnik unosi

NAME NAZIV.SUFIX (RETURN)

gde je NAZIV.SUFIX ime datoteke koje daje korisnik. Po unošenju naziva datoteke pojavljuje se poruka FROM

FROM XXXX (RETURN)

na šta korisnik odgovara unošenjem početne adrese memorijskog bloka (heksadecimalna vrednost adresa XXXX). Dalje računar postavlja pritanje kraja memorijskog bloka sa TO.

TO XXXX (RETURN)

na šta korisnik odgovara unošenjem poslednje adrese memorijskog bloka (heksadecimalna vrednost adrese XXXX).

Primer: Memorijski blok sa početnom adresom 10F2 i poslednjom adresom 2FB8 smestiti na disketu pod nazivom PRIMER.DO.

Rešenje:

```
Q
NAME PRIMER.DO (RETURN)
FROM 10F2 (RETURN)
TO 2FB8 (RETURN)
```

Naredba T

Ovom naredbom se čita datoteka sa diskete pod određenim nazivom i upisuje u korisnički memorijski prostor počev od proizvoljno odabrane memorijske adrese. Postupak je sledeći:

Po unošenju sa tasture

```
>T
```

javlja se poruka NAME

NAME NAZIV.SUFIX (RETURN)

nakon koje korisnik odgovara unošenjem naziva datoteke koja želi da se čita. Po unošenju naziva datoteke pojavljuje se poruka

FROM XXXX (RETURN)

na šta korisnik odgovara unošenjem početne adrese memorije (XXXX) koju proizvoljno bira.

Primer: Učitati sa diskete datoteku pod nazivom PRIMER.H i smestiti je u korisničku oblast memorije počev od heksadecimalne adrese 4000.

Rešenje:

```
T
NAME PRIMER.H (RETURN)
FROM 4000 (RETURN)
```

4.3.3. Naredbe koje mogu da se aktiviraju samo sa radne stanice nastavnika

U sistemu računarske učionice postoje naredbe koje se mogu generisati samo sa radne stanice nastavnika. Pokušaj da se neka od tih naredbi uputi sa bilo koje radne stanice završiće se izdavanjem poruke o greški. Postojanje ovih naredbi daje mo-

gućnost da radna stanica nastavnika bude mali kontrolor u sistemu. Kako to ona obavlja biće jasno kroz opis naredbi koji sledi.

Prema tome iz kog režima se aktiviraju naredbe ove grupe mogu se podeliti na:

— naredbe koje se pozivaju iz BASIC-a
— naredbe koje se pozivaju iz komandnog režima EDITOR-a.

Naredbe koje se pozivaju iz BASIC-a

SPY

Ovom naredbom se daje mogućnost nastavniku da ima uvid u to šta koji učenik u određenom trenutku radi, dokle je stigao u pisanju nekog programa, rešavanju nekog problema i slično. Format naredbe je:

```
SPY"n"
```

gde je:

n — broj radne stanice čiji se rad kontroliše

Radna stanica učenika, čiji se ekran uzima pod kontrolu, nema informaciju o tome da je pod kontrolom. S njegove strane gledišta ništa se ne događa.

SEND

Ova naredba daje mogućnost nastavniku da programsku datoteku pošalje svim radnim stanicama učenika ili samo odabranim radnim stanicama. Na ovaj način nastavnik može da deli zadatke prema mogućnostima i sklonostima svakog učenika. Isto tako, on može lako i brzo da obavi testiranje čitavog odeljenja iz određene oblasti.

Format naredbe je:

```
SEND"nbNAZIV.SUFIX"
```

gde je:

n — broj radne stanice (kanala) kome se šalje datoteka

b — oznaka za blanko.

Ako se umesto n napiše * datoteka se šalje svim radnim stanicama. U tom slučaju format naredbe je:

```
SEND"*NAZIV.SUFIX"
```

Primer: Datoteku pod nazivom PRIMER.SR poslati radnoj stanici 14.

Rešenje:

```
SEND "14 PRIMER.SR"
```

MSG

Naredbom MSG"D" omogućava se slanje poruka između radnih stanica učenika.

Naredbom MSG"Z" postavlja se zabrana slanja poruka između radnih stanica učenika.

Naredbe koje se pozivaju iz komandnog režima EDITOR-a

RLOCK

Radna stanica nastavnika korišćenjem ove naredbe postavlja zaštitu od čitanja za sve datoteke generisane u sistemu. Sa radne stanice učenika mogu se čitati samo da-

toteke koje gu generisane sa te radne stanice. Na ovaj način onemogućava se "prepisivanje" datoteka koje su generisale duge radne stanice.

RUNLOCK

Pomoću ove naredbe nastavnik sa glavne stanice stavlja sve datoteke generisane u sistemu na uvid svim radnim stanicama. To znači da bilo koja radna stanica može da pročita bilo koju datoteku, čiji naziv postoji u zajedničkom imeniku. Ovo može da bude korisno onda kada nastavnik želi da istakne način rešavanja pojedinih zadataka.

PLOCK

Radna stanica nastavnika korišćenjem ove naredbe stavlja zabranu upotrebe štampača za radne stanice učenika.

PUNLOCK

Sa radne stanice nastavnika korišćenjem ove naredbe izdaje se dozvola korišćenja štampača za radne stanice učenika.

PRILOG 1

TABELA ZA UGRAĐENE ZNAKE I SIMBOLE

Decimalni kod	Heksa kod	Znak i simbol	Pritisnuta dirka	Decimalni kod	Heksa kod	Znak i simbol	Pritisnuta dirka
0	00		NUL	64	40	@	
1	01	CA	CTRL A	65	41	A	
2	02	▯	CTRL B	66	42	B Б	
3	03	Cc	CTRL C	67	43	C Ц	
4	04			68	44	D Д	
5	05	⌟	CTRL E	69	45	E	
6	06	—	CTRL F	70	46	F Ф	
7	07	ı	CTRL G	71	47	G Г	
8	08		CTRL H	72	48	H Х	
9	09	—	CTRL I	73	49	I И	
10	0A		LF	74	4A	J	
11	0B	⸗	CTRL K	75	4B	K	
12	0C	ı	CTRL L	76	4C	L Л	
13	0D		CR	77	4D	M	
14	0E	⁼	CTRL N	78	4E	N Н	
15	0F		CTRL O	79	4F	O	
16	10	⊤	CTRL P	80	50	P П	
17	11	⌝	CTRL Q	81	51	Q Ъ	
18	12	⁄	CTRL R	82	52	R Р	
19	13	Cs	CTRL S	83	53	S С	
20	14	⌞	CTRL T	84	54	T	
21	15	⅀	CTRL U	85	55	U У	
22	16	⏐	CTRL V	86	56	V В	
23	17	⁄	CTRL W	87	57	W Ь	
24	18	ž Ж	CTRL X	88	58	X	
25	19	⌜	CTRL Y	89	59	Y Ц	
26	1A	ž Ж	CTRL Z	90	5A	Z З	
27	1B	Es	CTRL	91	5B	↑ Ш	
28	1C	←	CTRL	92	5C	← Ъ	
29	1D	→	CTRL	93	5D	→ Ћ	
30	1E	↑	CTRL	94	5E	↑ Ч	
31	1F			95	5F		
32	20		BLANKO	96	60		
33	21	!	SHIFT 1	97	61	a	CAPS A
34	22	"	SHIFT 2	98	62	b б	CAPS B
35	23	#	SHIFT 3	99	63	c ц	CAPS C
36	24	$	SHIFT 4	100	64	d д	CAPS D
37	25	%	SHIFT 5	101	65	e	CAPS E
38	26	&	SHIFT 6	102	66	f ф	CAPS F
39	27	'	SHIFT 7	103	67	g г	CAPS G
40	28	(	SHIFT 8	104	68	h х	CAPS H
41	29	)	SHIFT 9	105	69	i и	
42	2A	*	SHIFT :	106	6A	j	
43	2B	+	SHIFT ;	107	6B	k к	
44	2C	,		108	6C	l л	
45	2D	-	SHIFT =	109	6D	m м	
46	2E	.		110	6E	n н	
47	2F	/		111	6F	o	
48	30	0		112	70	p п	
49	31	1		113	71	q ь	
50	32	2		114	72	r р	

PRILOG 2.

Poruke za greške i njihovo značenje

00 — poruka o prekidu
01 — sintaksna greška u ASC ili LEN funkciji
02 — Polje van opsega ili nedimenzionisano polje
03 — DIM greška; sintaksna greška ili prekoračenje memorije
04 — DEFINT ima pogrešan završetak reda
05 — nepostojanje zagrada u funkciji sa argumentom
06 — argument van opsega
07 — nedozvoljeno izračunavanje zbog različitih režima
08 — greška zbog deljenja nulom ili log negativnog broja
09 — funkcija koja se ne može izvršiti
10 — EXIT naredba se koristi u GOSUB ili FOR/NEXT
11 — prekoračenje steka u FOR/NEXT, ili nemogućnost da se FOR izvrši direktno
12 — sintaksna greška u FOR
13 — prekoračenje steka u GOSUB
14 — neprihvatljiv znak u heksadecimalnom polju
15 — suviše veliki broj u pokretnom zarezu za prebacivanje u ceo broj
16 — nedozvoljen operator u uslovnoj naredbi
17 — INPUT ili FVAL ne mogu se izvršiti u direktnom režimu
18 — u READ naredbi mora da postoji naziv promenljive ili niza
19 — sintaksna greška u READ ili INPUT naredbi
20 — sintaksna greška u LEN funkciji
21 — sintaksna greška u LEN naredbi
22 — nepostojanje navodnika
23 — sintaksna greška u LIST
24 — nepostojeća reč u BASIC 3 biblioteci
25 — sintaksna greška u MID$ funkciji
26 — neprihvatljiv naziv promenljive u NEXT naredbi
27 — očekuje se broj ili slovo
28 — nedostaje aritmetička zagrada
29 — pogrešan broj argumenata u POKE naredbi
30 — nedozvoljeni zadani znak U PRINT naredbi
31 — sintaksna greška u DATA naredbi
32 — nema više podataka
33 — nepostojeći niz u INPUT naredbi
34 — nema znaka = u LET naredbi za nizove
35 — nema zagrada u polju niza pri INPUT
36 — suviše argumenata u USR ili CALL
37 — sintaksna greška u CHR$ funkciji
38 — neprihvatljiv znak u binarnom polju (1 ili 0)
39 — prekoračenje linijskog bafera
42 — nedozvoljeni završetak reda ili nepostojeća naredba
43 — prekoračenje steka u pokušaju poziva programa za prekid
44 — unet broj ima suviše cifara (10)
45 — neprihvatljiv znak u polju brojeva

46 — nepostojeći broj reda
47 — neprihvatljiv operator u IF naredbi za nizove
48 — prekoračenje memorije pri pokušaju unošenja niza
49 — pogrešan broj argumenata u MOD funkciji
50 — suviše dug program
51 — argument van opsega
52 — pogrešan broj argumenata
53 — pogrešan broj argumenata
54 — nedefinisana promenljiva niza
63 — nedostatak memorije za RUNUMBER tabelu
64 — nepronađeni broj za RENUMBER
65 — nedefinisana promenljiva
66 — pokušaj generisanja niza većeg od 127 znakova
68 — broj argumenata u SCR mora da bude 1
72 — broj argumenata u TONE mora biti 3

PRILOG 3.

GREŠKE KOMANDNOG NAČINA RADA EDITOR-A

1 : nepostojeća funkcija
2 : nepravilna upotreba funkcija
3 : kursor se nalazi ispred markera bloka
4 : greška prilikom učitavanja

PRILOG 4.

GREŠKE PRI ASEMBLIRANJU

11 : nepostojeći mnemonik
12 : dve labele sa istim imenom
13 : nepravilna upotreba operanda
14 : nedozvoljen završetak reda
15 : nepostojeći registar
16
17 : nepravilna upotreba mnemonika
18
22 : labela i BR nisu na istoj stranici
99 : greška u matematičkom izrazu

PRILOG 5.

GREŠKE KOJE MOGU DA SE JAVE U RADU SISTEMA RAČUNARSKE UČIONICE

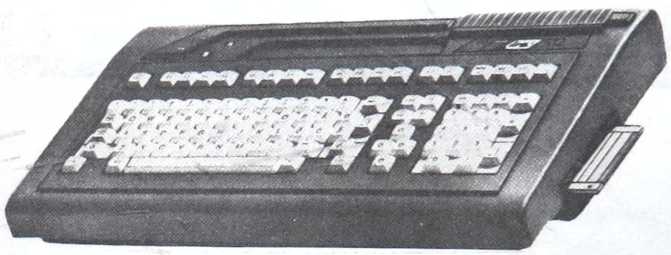
BASIC

ERR CODE 50	— nema dovoljno memorijskog prostora
ERR CODE 77	— sintaksna greška
ERR CODE 88	— nepostojeći naziv datoteke
	— pogrešan tip datoteke
	— zabrana čitanja
	— duplificirani naziv
	— greška pri upisu
	— zauzeće radne stanice
	— zabrana slanja poruka
	— nije dozvoljeno štampanje
	— privilegovana naredba
ERR CODE 99	— greška u prenosu

EDITOR

ERR CODE O	— sintaksna greška
ERR CODE 8	— nepostojeći naziv datoteke
	— pogrešan tip datoteke
	— zabrana čitanja datoteke
	— duplicirani naziv
	— greška pri upisu
	— zauzeće radne stanice
	— zabrana slanja poruka
	— postoji zaštita datoteke od brisanja
ERR CODE 9	— greška u prenosu
ERR CODE 5	— dužina datoteke je nula

Ei RO „Ei – RAČUNARI" OOUR FABRIKA RAČUNSKIH MAŠINA NIŠ



RAČUNARSKA UČIONICA LIRA RU

U Elektronskoj industriji OOUR Fabrika računskih mašina razvijeno je moćno nastavno sredstvo — Računarska učionica, u potpunosti u skladu sa kriterijumima za izbor računara za potrebe školstva.

KONFIGURACIJA RAČUNARSKE UČIONICE

Cnetralna jedinica • do 16 radnih stanica • komunikacioni softver

CENTRALNA JEDINICA PC, AT

16 bitni mikroprocesor 80286 • sistemski takt 6/10 MHz • ROM 16 – 128 KB • RAM 1 MB • jedna flopi disk jedinica od 360 KB – 1,2 MB • HARD DISC 50 MB • RS232 • CENTRONICS • kartica za komunikacije • XT/AT tastatura (102 tastera) • MS-DOS 3,21

ŠTAMPAČ

Matrični, serijski, 80/132 kolone

RADNA STANICA — LIČNI RAČUNAR LIRA

16-o bitni mikroprocesor 8088 • sistemski takt 4.77/10 MHz • RAM: 256, 512 ili 640 KB • monohromatski kontroler rezolucije 720 × 348 tačaka • CGA kontroler 640 × 200 tačaka • RF modulator za povezivanje TV prijemnika • kontroler za dve jedinice flopi diska • jedinica flopi diska od 3,5 incha • paralelni priključak za štampač • časovnik realnog vremena • serijska komunikacija RS232 • priključak za palicu (GAME port) • priključak za BAR kod čitač (LIGHT PEN — svetlosno pero) • opciono može biti postavljen aritmetički koprocesor • priključak za drugu jedinicu flopi diska

DISPLEJ

Monohromatski i kolor monitor • TV prijemnik (monohromatski — kolor)

KOMUNIKACIONI SOFTVER S-NET

omogućuje potpuno komuniciranje centralne jedinice i radnih stanica na nivou prenosa programa i podataka • omogućuje radnim stanicama korišćenje resursa centralne jedinice

PRIMENA

upravljanje vaspitno-obrazovnim procesima • informacioni sistem obrazovnih institucija • INDOK delatnost u obrazovanju • naučno istraživački rad u različitim naučnim disciplinama • učenje uz pomoć računara • nastava uz pomoć računara • učenje programskih jezika

ELEKTRONSKA INDUSTRIJA — Ei RO „Ei-RAČUNARI" OOUR Fabrika računskih mašina • Bul. Veljka Vlahovića 80-82, • Direktor 018/325-461, Direktor RJ marketing 018/55-583, Sektor plasman 018/54-779, 51-568, Sektor usluge korisnicima, Služba održavanja 018/54-867, Služba školski centar 018/54-090, Sektor inženjering informacionih sistema 018/52-782, 52-876, Sektor tehničke podrške 018/54-090 • TLX 16283 YU Ei FRM
POSLOVNA JEDINICA — BEOGRAD — Ul. Rudo 2, 11000 Beograd 011/488-8260, 488-3265
POSLOVNA JEDINICA — TITOGRAD — Ul. B. Bracanovića 58, 81000 Titograd 081/24739

ELEKTRONSKA INDUSTRIJA – Fabrika računskih mašina

MMS 1800 FiL

Računarska učionica

MMS 1800 RU je mikroračunarski sistem, namenjen osnovnom i srednjem obrazovanju. U procesu obrazovanja ovakav sistem se može koristiti za obavljanje edukativnih poslova: nastave uz pomoć računara, učenje uz pomoć računara, učenje programskih jezika i za obavljanje drugih poslova u obrazovnim institucijama (INDOK delatnost, upravljanje vaspitno – obrazovnim poslovima i administrativno – statistički poslovi).
Sistem je projektovan na bazi modularnog mikroračunarskog sistema MMS 1800 RU kao glavnog računara, na koji se priključuje maksimalno 16 radnih stanica. Ovakva konfiguracija omogućuje rad više korisnika uz sekvencijalan pristup glavnom računaru i korišćenje svih njegovih resursa.
Radna jedinica je realizovana sa mikroračunarom PECOM 64 i monitorom.
Režimi rada u kojima može da radi ova stanica su:
– terminalski režim rada i
– autonomni režim rada
Radne stanice komuniciraju sa centralnim procesorom preko asinhrone veze (RS232C) u terminalskom režimu rada.
U autonomnom režimu rada gube se terminalske funkcije i PECOM 64 radi kao autonomni mikroračunar.

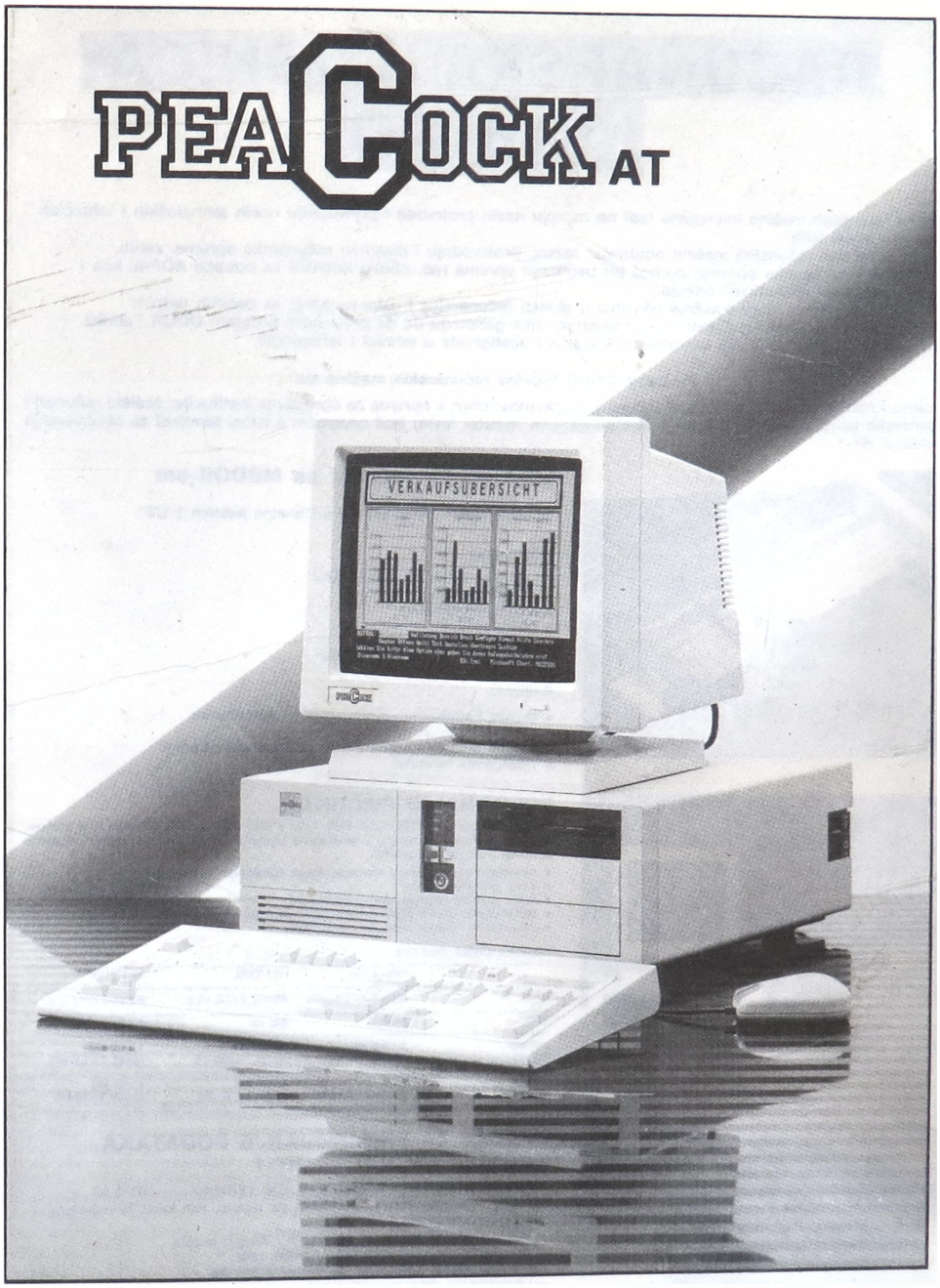
HARDVER SISTEMA

U osnovnoj konfiguraciji sistem računarske učionice (RU) sadrži:
– centralnu jedinicu MMS 1800 RU (sa multipleksiranim UART-om za povezivanje radnih stanica – maksimalno 16 PECOMA-a 64)
– radne stanice (maksimalno 16) koje su u stvari µR PECOM 64 sa firmverom za rad u računarskoj učionici,
– operatorski terminal koji može da bude PECOM 64 sa monitorom ili asinhroni video terminal VIP 7251 RU,
– serijski štampač (80-132 kolonski) sa interfejsom RS232C.
Komunikacioni deo omogućuje multipleksiranu vezu radnih stanica sa centralnim procesorom. Svakom korisniku je omogućeno korišćenje resursa MMS 1800 RU (čitanje, upis, kreiranje, kopiranje datoteka i slično).

SISTEMSKI SOFTVER CENTRALNE JEDINICE

Sistemski softver računarske učionice čine sistemski softver centralne jedinice i sistemski softver mikroračunara PECOM 64

Sistemski softver centralne jedinice MMS 1800 RU čine:
– uslužni program MS 10 (firmver) koji obezbeđuje:
  – komunikaciju MMS 1800 RU sa mehanizmom disk jedinica
  – interfejs MMS 1800 RU sa operatorskim terminalom
  – pristup svim memorijskim lokacijama i registrima opšte namene (SPR)
– centralne jedinice MMS 1800 RU
– MDOS operativni sistem na disketi
– program za podršku rada MMS 1800 RU u računarskoj učionici
– programski jezici (opcijski):
BASIC (kompilator – interpretator)
PLM 180
µ FORTH


PEACOCK AT
VERKAUFSUBERSICHT

RO „Ei-RAČUNARI" OOUR FABRIKA RAČUNSKIH MAŠINA NIŠ

18000 Niš, Bulevar Vojvode Vlahovića 80—82, telefon: 018/325-461; 018/55-583; 018/54-090; 018/54-779; 018/51-568; tlx 16283 YU EI FRM; žrn. 62500-601-3114

RAČUNARSKA UČIONICA LIRA RU

Fabrika računskih mašina intenzivno radi na razvoju novih proizvoda i prihvatanju novih tehnoloških i tehničkih rešenja u izradi istih.
Delatnost fabrika računskih mašina obuhvata: razvoj, proizvodnju i plasman računarske opreme, zatim, instaliranje i održavanje opreme, pomoć pri uvođenju opreme rad, obuku kadrova za potrebe AOP-a, kao i razvoj i uvođenje aplikativnih paketa.
Jaka kadrovska baza, dugogodišnje iskustvo u oblasti računarstva i uska saradnja sa ostalim radnim organizacijama u SOUR-u Elektronske industrije, daje garanciju da će proizvodni program OOUR Fabrika računarskih mašina biti u žiži savremenih kretanja i dostignuća u tehnici i tehnologiji.

Proizvodi OOUR Fabrika računarskih mašina su:

• mikro i mini računari • personalni računari IBM kompatibilan • oprema za obrazovne institucije: školski računari i računarske učionice • P.O.S sistemi (komunikacione registar kase) kod opremom • ručni terminal za obuhvatanje podataka RT-1

LIRA PC XT računar sa MSDOS-om

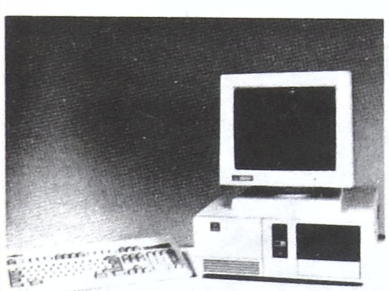
Procesor 8088 • RAM 512 KB • Disketna jedinica 3 1/2"

Priključci za:

- monohromatski i kolor monitor
- TV prijemnik
- štampač (paralelni)
- palicu za igru
- bar-kod čitač
- drugu disketnu jedinicu 5 1/4"

Primena:

- zabava i edukacija — računarstvo i informatika
- poslovni računar male privrede
- školski računar i radna stanica računarske učionice
- inteligentni terminal



PERSONALNI RAČUNAR

Personalni računari FRS 5100, FRS 7100 i FRS 9100 su PC računari IBM kompatibilni, vrhunske tehnologije i modularne konstrukcije sa širokim izborom dodatne opreme: monitora i adaptera

- monohromatski grafički monitor visoke rezolucije
- kolor grafički monitor
- kolor grafički monitor visoke rezolucije
- HERKULES, CGA i EGA kompatibilni grafički adapteri
- štampača i ostalih opcionih elementa

PREGLED KARAKTERISTIKA

	FRS 51XX	FRS 71XX	FRS 91XX
CPU	8088-2 4,77/8 MHz	8088-2 4,77/8 MHz	80286-2 6/10 Mhz
RAM	640 KB	640 KB	1 MB
jedinice diskete	2×360 KB	1×360 KB opcija 1×360 KB	1×1,2 MB opcija 1×1,2 MB
jedinice diska	/ opcija 2×20 MB	1×20 MB opcija k×20 MB	1×40 MB opcija 1×40 MB

UREĐAJ ZA PRIKUPLJANJE PODATAKA RUČNI TERMINAL — RT-1

Uređaj za prikupljanje podataka, RUČNI TERMINAL — RT-1 je namenjen za obuhvatanje podataka na terenu, van lokacije računara

— unošenjem preko tastature
  — učitavanje stanja s električnih i drugih brojila
  — popis i naručivanje robe i materijala
  — sve ostale aplikacije na zahtev korisnika
— automatski, čitanjem bar koda
  — popis, naručivanje, obuhvatanje za fakturisanje, praćenje robe kod izmena cena
— obradu obuhvaćenih podataka na računarskim sistemima

Rad sa RUČNIM TERMINALOM omogućava

— brzi prenos podataka s računara u ručni terminal
— brzo, lako i efikasno obuhvatanje podataka
— uštedu resursa računara u procesu obuhvatanja
— brzi prenos podatka računaru na dalju obradu

Za izdavača: Sava Radović, direktor
Tehnički urednik: Tomislav Radunić
Obrada teksta na računaru: Saša Stanojković
Naslovna strana: Zoran Branković
Lektor: Dušica Milanović
Izdavač: NIRO "Tehnička knjiga", Beograd, 7. jula 26
Štamparija: GRO "Kultura" OOUR "Slobodan Jović"
Tiraž:2000 primeraka I izdanje 1988.

Oslobođeno poreza na promet na osnovu mišljenja Republičkog komiteta za kulturu SRS

RAČUNARI I INFORMATIKA

PREPORUČUJEMO VAM NEKOLIKO KNJIGA IZ NAŠE BIBLIOTEKE

Grupa autora
ŠTA MOŽE COMMODORE 64
*

V. Spasić i D. Veljković
BASIC ZA MIKRORAČUNARE — COMMODORE 64
*

Armando Jorno
TURBO PASKAL SA GRAFIČKIM APLIKACIJAMA
*

A. Bennett
MAŠINSKE RUTINE ZA VAŠ COMMODORE 64
*

V. Petrović i Z. Mošorinski
COMMODORE 128 — PROGRAMIRANJE U BASIC-u
*

Dejan Stajić
INTERFEJSI I MODEMI ZA MIKRORAČUNARE
*

Dejan Ristanović
OBRADA TEKSTA NA RAČUNARU
*

Adem Jakupović
dBASE III PLUS

YUISBN 86-325-0135-6