veljko spasić
dušan veljković

basic

za mikroračunare

commodore 64

Mr VELJKO SPASIĆ, dipl. mat. i DUŠAN VELJKOVIĆ, dipl. mat.

BASIC ZA MIKRORAČUNARE (COMMODORE 64)

NIRO »TEHNIČKA KNJIGA« I ZAVOD ZA IZDAVANJE
UDŽBENIKA — OOUR STVARANJE I PROIZVODNJA
NASTAVNIH SREDSTAVA — BEOGRAD, 1985.

Recenzent:

prof. dr Nedeljko Parezanović

PREDGOVOR

Računari, njhova primena i programiranje, naglo izlaze iz zatvorenih profesionalnih krugova i brzo prodiru do velikog broja ljudi. Ova, nesumnjivo, pozitivna tendencija demokratizacije računarstva i informatike omogućena je tek kada su proizvedeni računari malih dimenzija, velikih mogućnosti i veoma niske cene.

Paralelno s povećanjem broja zainteresovanih za mikroračunare, kao i masovnosti njihovih sadašnjih i potencijalnih vlasnika, raste i potreba za odgovarajućom stručnom literaturom, koje na našem jeziku nema dovoljno. Zbog toga i smatramo da se ova knjiga pojavljuje u momentu kada može biti korisna.

Knjiga bi trebalo da doprinese savlađivanju programskog jezika BASIC, korišćenju periferijskih uređaja, kao i osnovama programiranja grafike i zvuka na mikroračunaru Commodore 64. Namenjena je svima koji su rešili da se upoznaju sa BASIC-jezikom, a posebno onima koji imaju ili će nabaviti računar Commodore 64.

Čitaocu nije potrebno predznanje iz računarstva, matematike ili elektronike.

Autori

1. UVOD

Mikroračunari nemaju u svom imenu reč »mikro« zbog malih dimenzija ili niske cene, već zbog toga što su bazirani na mikroprocesoru. Ovaj centralni element mikroračunara obavlja (između ostalog) i najvažniji zadatak — izvršavanje programa.

Programi, koje mikroprocesor može neposredno izvršavati, moraju biti zapisani u »mašinskom kôdu«, tj. isključivo pomoću nula i jedinica. Pisanje i korekcije programa na ovom nivou predstavljaju veoma veliki problem za čoveka. Zbog toga su stvoreni viši programski jezici, koji su tako konstruisani da im je forma prihvatljivija za čoveka. Time se problem programiranja olakšava, ali se javlja potreba za prevođenjem programa iz forme višeg programskog jezika u mašinski oblik spreman za izvršavanje. Ovaj posao se prepušta računaru.

Razvijen je veliki broj viših programskih jezika, a BASIC je jedan od najpoznatijih među njima. Njegovo ime je skraćenica nastala od Beginner's All Purpose Symbolic Instruction Code, što u prevodu znači »višenamenski jezik simboličkih instrukcija za početnike«. Prva verzija jezika je nastala 1965 (dakle pre pojave mikroračunara) na Dartmuntskom koledžu na bazi zahteva da se sastavi viši programski jezik za rešavanje zadataka u interaktivnom režimu dijaloga sa čovekom i pri uslovu da za rešavanje nije potreban veliki broj ulaznih podataka. Nakon 1965. BASIC se naglo širi, tako da već 1970. u rasprostranjenosti među programskim jezicima zauzima peto mesto u svetu. Paralelno sa rastom popularnosti BASIC doživljava i modifikacije. Ove promene najčešće

imaju za cilj da poboljšaju pojedine slabe tačke u okviru osnovnog BASIC-jezika, kao i da se BASIC prilagodi novim zahtevima koji su se javljali tokom njegovog postojanja. BASIC ovim promenama dobija na složenosti i postaje jezik kojim se može uraditi više, ali istovremeno gubi na svojoj prvobitnoj jednostavnosti.

Naznačimo samo neke osnovne pravce kojima je tekao razvoj BASIC-a. Kao prvo, to je uvođenje azbučnih promenljivih (pored numeričkih), koje omogućavaju lakše programiranje dijaloga sa računarom. Zatim, poboljšavale su se mogućnosti određivanja formata, tj. oblika u kome se izdaju izlazne veličine. Da bi se omogućilo lakše unošenje podataka u program, kao i razmena iste grupe podataka između više programa, u BASIC se uvode teke u koje se mogu trajno smeštati podaci i iz kojih se ti podaci mogu čitati. Tokom razvoja BASIC-jezika dodavani su i elementi koji omogućavaju segmentaciju programa, tj. razbijanje na delove koji su relativno nezavisni. U poslednje vreme se BASIC obogaćuje naredbama koje omogućavaju programiranje grafike, a u najnovije vreme i zvuka. Danas postoji veliki broj varijanti BASIC-jezika. Svi oni imaju zajedničko jezgro, a razlikuju se po formi nadgradnje (mada ne mnogo i prema njenoj suštini). Postoje standardi, ali još uvek se sa pojavom svakog novog mikroračunara, rađa i neka modifikacija jezika. Međutim, ove razlike među pojedinim »dijalektima« BASIC-jezika ni u kom slučaju nisu značajne ni velike. Kada se savlada jedna verzija BASIC-a, veoma se lako prelazi na neku drugu, jer se jednom usvojene osnovne ideje, lako prepoznaju u drugoj formi.

BASIC za računar Commodore 64 predstavlja jednu od verzija ovog jezika. On ima svoje dobre i loše strane. Na to se ukazuje na odgovarajućim mestima u tekstu. Ovde samo napomenimo da su osnovne mogućnosti BASIC-jezika za Commodore 64 (glava 2) gotovo iste sa verzijama ovog jezika na drugim mikroračunarima. Tek se u dodatnim mogućnostima BASIC-a (glava 3) pojavljuju razlike, mada ni one nisu velike.

Moguće je na veliki broj različitih načina izložiti materijal sadržan u ovoj knjizi. Može se ići od krajnje formalnog i sažetog prikaza, pa sve do opisa koji se bazira najvećim delom na primerima bez uopštavanja.

Ovde je učinjen pokušaj da se novi elementi najpre kratko i jednostavno objasne zajedno sa potrebom i logikom njihovog uvođenja, a da se zatim i preciznije uvedu. Tekst je praćen većim brojem primera, koji služe kao ilustracija i koje čitalac može neposredno unositi u svoj računar. Ovo je dobra praksa jer ništa nije tako uspešna vežba u programiranju kao direktan kontakt sa programom koji se izvršava ili modifikuje. Svakako da ovo nije neophodno. Već i samo analiziranje programa bez upotrebe računara, daje dovoljno informacije da se može nastaviti sa čitanjem. Ono što ovakav način ne pruža, to je slobodno eksperimentisanje na računaru i testiranje sopstvenih zamisli.

Materija se izlaže od jednostavnije ka složenijoj, a uređena je tako da se čitaocu postepeno proširuju mogućnosti programiranja. Pri ovome se vodilo računa da se najpre izloži ono što je potrebno za samostalno formiranje jednostavnijih programa, zatim osnovni pojmovi koji su neophodni za prelaz na složenije programe, iza čega sledi detaljnije izlaganje svih preostalih elemenata BASIC-jezika, koji su na raspolaganju. Radi toga se i preporučuje čitanje knjige redom, bez preskakanja (BASIC, periferija, zvuk, grafika). Ako se želi najpre pročitati glava koja izlaže programiranje zvuka ili muzike, onda se podrazumeva bar elementarno poznavanje BASIC programiranja.

U knjizi se redom izlažu: osnovne mogućnosti BASIC-a, dalje mogućnosti BASIC-a, periferijski uređaji, programiranje zvuka i muzike i programiranje grafike. Na kraju su dodaci, gde se između ostalog nalazi i kratak pregled svih BASIC naredbi, komandi i funkcija, što je korisno kada se već poznaje BASIC-jezik.

Cela knjiga je napisana na računaru Commodore 64 korišćenjem programa za obradu teksta.

2. OSNOVNE MOGUĆNOSTI BASIC-a

2.1. UVOD

Programiranje računara podrazumeva prethodno poznavanje osnovnih elemenata rukovanja ovim uređajem. Tu se pre svega misli na korišćenje tastature i povezivanje računara sa potrebnom pratećom opremom. Radi toga ćemo se najpre kratko zadržati na ovim pitanjima, a zatim preći na izlaganje elemenata BASIC-jezika.

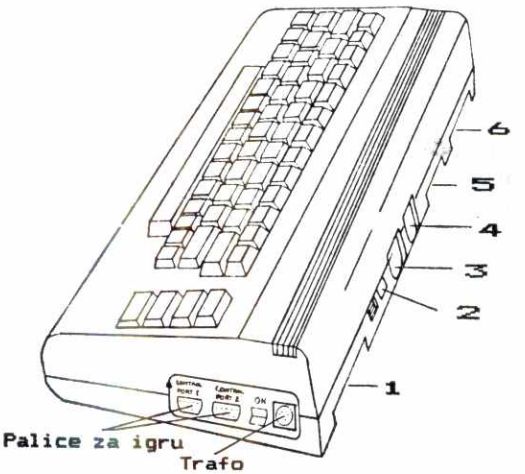
Na samom početku treba istaći osnovno pravilo: bez obzira na to koliko je uređaja priključeno na računar, on se **poslednji uključuje** i **prvi isključuje.** Nikako se ne sme priključivati kasetofon, a ni ostali uređaji, dok je uključena crvena signalna lampica računara.

Računar se uključuje prekidačem koji se nalazi sa desne strane. Odmah do prekidača nalaze se i dva priključka za palice za igru ili svetlosno pero.

Veza sa spoljašnjim svetom ostvaruje se preko priključaka na zadnjoj strani računara. Oni su međusobno različiti, pa je pogrešno priključivanje nemoguće. Tu se nalaze (sleva nadesno):

1) Priključak namenjen tehničkim proširenjima računara (novi operativni sistem, drugi jezici, dodatna memorija . . .). On je najveći od svih priključaka i pomoću njega je moguća kompletna kontrola računara u najrazličitijim primenama.

2) Priključak za vezu sa televizorom. Preko njega se na antenski ulaz televizora šalju signali kojima se generiše slika na ekranu. Neposredno pored njega je zavrtanj za fino podešavanje slike.



Sl. 2.1 — Priključci na računaru. 1 — proširenja, 2 — televizor, 3 — zvučni ulaz-izlaz i monitorski izlaz, 4 — disketna jedinica ili štampač, 5 — kasetofon, 6 — korisnički priključak

3) Zvučni ulaz/izlaz i monitorski izlaz. Preko ovog priključka računar se može povezati sa kolor-monitorom ili Hi-Fi uređajem, čime se postiže poboljšanje kvaliteta zvuka i slike. Posebno je zanimljiv zvučni ulaz na koji se može dovesti bilo kakav signal, obraditi (pojačati ili filtrirati) i priključiti zvučnom izlazu koji generiše računar. Tako se melodiji koju svira računar mogu dodati različiti drugi efekti, proizvedeni na drugim uređajima.

4) Priključak za serijsku vezu sa disketnom jedinicom ili štampačem. Mada postoji jedan ovakav priključak, moguće je priključiti do pet uređaja, jer se disketne jedinice mogu priključivati jedna na drugu i na taj način graditi lanac uređaja priključenih na isto mesto. Na kraju tog lanca može se nalaziti štampač.

5) Priključak za kasetofon. Na računar se priključuje specijalni kasetofon koji daje mogućnost čitanja, pisanja i kontrole motora pri ovim operacijama.

6) Korisnički priključak. Preko odgovarajućih elektronskih sklopova na ovo mesto mogu se priključiti najraznovrsniji uređaji: od modema za vezu sa drugim računarima, preko tvrdih diskova do igračaka ili zvona za buđenje.

2.1.1. Korišćenje tastature

Tasterima tastature mogu se obavljati različite funkcije. Razdvojićemo ih u 6 osnovnih grupa:

1 — unošenje teksta,
2 — promena boje slova kojima se piše,
3 — uključenje inverznog prikaza,
4 — prekidanje rada programa,
5 — korišćenje programabilnih tastera,
6 — vršenje izmena u programu — editovanje.

1. **Unošenje teksta.** Na prvi pogled zbunjuje mnoštvo simbola koji se nalaze na tasterima. Kao što se u svakodnevnom životu pri pisanju mogu koristiti simboli iz dve različite azbuke, ćirilične i latinične, tako i Commodore 64 ostavlja mogućnost izbora skupa simbola kojima će se ispisivati tekst na ekranu. Zajednički simboli u oba skupa su velika slova, cifre, specijalni znaci (+,*,=, ? . . .) i neki grafički simboli. Razlika je u tome što

— skup 1 sadrži mala slova,
— skup 2, umesto malih slova, ima nove grafičke simbole.

Promena skupa simbola za prikaz na ekranu vrši se istovremenim pritiskom na SHIFT i COMMODORE i izaziva promenu prikaza svih slova koja su već na ekranu.

Potrebno je zapamtiti sledeće: oba skupa sadrže sve simbole na gornjoj površini tastera, sve simbole koji su jedini na prednjoj strani i sve one koji su sa leve strane na onim tasterima koji na prednjoj strani imaju po dva simbola. Razlika je u tome što se u skupu sa malim slovima pritiskom na taster dobija malo slovo sa njegove gornje površine, a pritiskom na neki taster, dok je pritisnuto i SHIFT, odgovarajuće veliko slovo. U drugom skupu se istim postupkom dobijaju velika slova i novi grafički simboli.

2. **Promena boje slova.** Računar Commodore 64 raspolaže sa 16 boja kojima može prikazivati tekst. Osnovna boja kojom se piše po uključenju računara je svetloplava. Promena ove boje vrši se korišćenjem tastera sa ciframa od 1 do 8 i tastera »CTRL« i »COMMODORE«. Istovremenim

pritiskom na jedan od poslednja dva i neki od navedenih dobijaju se sledeće boje:

CTRL/1 . . . crna
CTRL/2 . . . bela
CTRL/3 . . . crvena
CTRL/4 . . . plavozelena
CTRL/5 . . . purpurna
CTRL/6 . . . zelena
CTRL/7 . . . plava
CTRL/8 . . . žuta
COMMODORE/1 . . . narandžasta
COMMODORE/2 . . . smeđa
COMMODORE/3 . . . svetlocrvena
COMMODORE/4 . . . siva 1
COMMODORE/5 . . . siva 2
COMMODORE/6 . . . svetlozelena
COMMODORE/7 . . . svetloplava
COMMODORE/8 . . . siva 3

Nakon promene boje svi simboli koji budu unošeni biće prikazani u novoj boji. Boja već prikazanih neće se promeniti.

3. **Uključenje inverznog prikaza.** Istovremeni pritisak na »CTRL« i 9 neće ostaviti traga na ekranu, ali će svi simboli koji nakon toga budu unošeni biti prikazani inverzno, bojom pozadine u kvadratu boje slova. Ovo daje mogućnost isticanja dela teksta. Isključenje se postiže istovremenim pritiskom na »CTRL« i 0.

4. **Prekid izvršavanja programa.** Za prekidanje izvršavanja programa koriste se tasteri sa natpisom RUN/STOP i RESTORE. Ako je u toku BASIC-program, on će gotovo uvek moći da se prekine pritiskom na RUN/STOP. Jedini izuzetak je mesto na kome se očekuje unošenje podataka sa tastature. Tada treba pritisnuti RUN/STOP a potom istovremeno i kratko RESTORE. Ovim postupkom obrisaće se ekran i vratiti svetloplava boja slova na tamnoplavoj podlozi. Program i podaci ostaju neoštećeni.

5. **Korišćenje programabilnih tastera.** Četiri tastera koji se nalaze sasvim desno na tastaturi nemaju za računar nikakvo značenje. Prepušteno je korisniku da im

dodeli značenja i koristi u skladu sa tim. Mogu se koristiti i zajedno sa SHIFT, pa mogu dati 8 različitih vrednosti. Na njima se može, na primer, predvideti upotreba nekih simbola koji ne postoje ni u jednoj od azbuka. To mogu biti slova č, ć, ž, š ili grčka i ćirilična slova itd.

6. **Vršenje ispravki u programu — editovanje.** Korišćenjem tastera sa strelicama i tastera INST/DEL i CLR/HOME omogućeno je lako ispravljanje grešaka u programu. Ovome je posvećen sledeći odeljak o ekranskom editoru.

7. **Osnovni elementi mikroračunara.** Među elektronskim komponentama na kojima se zasniva rad mikroračunara najvažnije su sledeće:

1 — mikroprocesor,
2 — memorija,
3 — generator slike,
4 — generator zvuka.

Mikroprocesor je zadužen za obavljanje svih obradnih i upravljačkih funkcija u računaru. Naredbe koje on izvršava nazivaju se **mašinske instrukcije,** a skup svih mašinskih instrukcija jednog mikroprocesora čini njegov **mašinski jezik.** Instrukcijama se izvršavaju radnje elementarnog nivoa (sabiranje, oduzimanje, poređenje...) i mogu imati najviše dva argumenta. Mašinski jezici različitih mikroprocesora su takođe različiti, pa se zato program napisan za jedan tip ne može direktno izvršiti na nekom drugom. Zbog toga je programiranje na mašinskom jeziku opravdano samo u posebnim slučajevima.

BASIC je jezik kojim se zahtevi saopštavaju na način blizak čoveku. Njegove se naredbe u mikroprocesoru ne mogu izvršiti, pa je zato u memoriji računara stalno prisutan program (BASIC-interpretator), koji svaku naredbu BASIC-jezika pretvara u niz mašinskih instrukcija koje joj odgovaraju. Tek izvršavanjem tog niza instrukcija dobija se traženi rezultat.

Mikroprocesor je direktno povezan sa memorijom i iz nje dobija instrukcije i podatke. Njena osnovna funkcija je čuvanje različitih podataka koji se koriste u radu mikroračunara. Postoje dva osnovna tipa memorije:

1 — Memorija koja sadrži programe na mašinskom jeziku, namenjene za upravljanje računarom i prihvatanje i izvršavanje naredbi zadatih na BASIC-jeziku. Osobina ove memorije je da joj se sadržaj ne može menjati, već služi samo za čitanje. Otuda i naziv ROM (Read Only Memory) — memorija samo za čitanje.

2 — Memorija u koju se sadržaj upisuje tek po uključenju računara. U ovoj memoriji se čuva korisnički BASIC-program, podaci koji se u njemu koriste, kao i oni podaci koji su potrebni za rad programa iz ROM-a, a mogu se menjati. U ovu memoriju može se upisivati ili iz nje čitati, pa otuda i naziv RAM (Random Access Memory) — memorija sa slobodnim pristupom. Za svaki od navedenih sadržaja koristi se posebna zona u memoriji, pa tako razlikujemo zonu programa, zonu podataka i zonu sistemskih promenljivih.

Osnovni memorijski element je registar (ili bajt) i u njega može biti upisan ceo broj između 0 i 255. Broj se zapisuje u binarnom brojnom sistemu, jedinom brojnom sistemu pogodnom za obradu zapisa u računaru. Registar ima 8 ćelija (ili bita) i svaki od njih može predstavljati nulu ili jedinicu. Zato se u registru može zapisati najviše osmocifreni binarni broj. Odatle potiče i pomenuti raspon brojeva koje registar može da sadrži.

Mesto registra u memoriji određeno je njegovom adresom. Skup svih različitih adresa kojima mikroprocesor može da manipuliše naziva se **adresni prostor** mikroprocesora.

Veličina memorije najčešće se izražava jedinicom »kilobajt« (kB), koja ima 1024 registra. Razlog za ovo nalazi se u činjenici da se i adresa, kao i podatak, u računaru posmatra kao binarni broj. Pri tome je 1024 (ili binarno 10000000000) broj najbliži 1000, a pogodan za ovu vrstu zapisa.

Dve specifične funkcije, generisanje slike i generisanje zvuka, obavljaju specijalno za to konstruisani elementi. Načinom njihovog rada upravlja mikroprocesor, postavljajući zahteve za dobijanje različitih efekata. Međutim, sam proces formiranja slike ili zvuka odvija se potpuno nezavisno od mikroprocesora i teče uporedo sa procesima koji se u njemu odvijaju. Tako se glavni posao mikroprocesora,

izvršavanje korisničkog zadatka, izvršava najvećom mogućom brzinom.

Na kraju ovog odeljka navedimo i nekoliko termina koji se često koriste:

čip — opšti naziv za elektronsku mikrokomponentu,

hardver — skup komponenti od kojih je sačinjen računar,

softver — skup programa za upravljanje radom računara.

2.2. EKRANSKI EDITOR

U memoriji se BASIC-program čuva u delu koji se naziva »zona programa«. Ispravljanje programa ili editovanje, ima zadatak da promeni sadržaj te zone, odnosno izvrši zamenu postojećih delova novim. Sve poslove u vezi sa editovanjem obavlja editor — poseban program iz ROM-a.

Startovanje editora kod većine računara se vrši posebnom komandom (EDIT ili slično), pri čemu se precizira deo programa, programski red, koji se želi ispravljati. Za upravljanje editovanjem se koriste posebne komande za tačno određivanje mesta na kome se vrše ispravke, brisanje i dodavanje novog sadržaja.

Ovakav tip editora naziva se »linijski editor«. Potreba da se poznaje poseban jezik editora i potreba za posebnim načinom pozivanja su njegovi osnovni nedostaci. Korak napred u postupku editovanja predstavlja tzv. »ekranski editor«, koji eliminiše spomenute nedostatke.

Ekranski editor radi na sledeći način: Sadržaj zone programa prikazuje se na ekranu (program se »lista«), a zatim se izmenama sadržaja ekrana vrše i izmene u zoni programa. Sve potrebne funkcije pri editovanju se ostvaruju pritiskom na neki od tastera tastature, pa je na taj način eliminisana potreba za jezikom editora. Za dovođenje kursora na željeno mesto koriste se tasteri sa strelicama na levo, desno, gore i dole. Za brisanje i ubacivanje simbola koriste se posebni tasteri. Nakon obavljanja svih željenih izmena u jednom programskom redu, zadaje se simbol za kraj reda. Nakon toga editor vrši zamenu starog

sadržaja iz zone programa novim. Na ovo mesto u postupku editovanja treba posebno upozoriti, jer se izostavljanjem unošenja kraja reda neće izvršiti izmena u zoni programa, iako je sadržaj ekrana promenjen.

Commodore 64 poseduje efikasan i jednostavan ekranski editor. Sve njegove komande dobijaju se sa tri tastera: LEVO/DESNO, GORE/DOLE i INST/DEL. Korišćenjem prva dva kursor se može dovesti na bilo koje mesto na ekranu. Kada je to urađeno, može se vršiti neka od sledećih izmena:

1 — Zamena starog sadržaja novim — preko starog se unosi novi sadržaj.

2 — Brisanje starog sadržaja — pritiskom na DEL briše se znak sa leve strane kursora.

3 — Ubacivanje novog sadržaja između dva postojeća znaka — treba onoliko puta pritisnuti INST (tj. kombinaciju INST/DEL i SHIFT) koliko se novih simbola želi uneti. Već postojeći sadržaj će se za toliko »razmaći«, ostavljajući slobodan prostor za upis novih simbola.

Pored navedenih tastera često se koristi i CLR/HOME. Pritiskom na njega kursor se dovodi u gornji levi ugao ekrana, dok se unošenjem CLR (SHIFT i CLR/HOME) ekran briše i kursor postavlja kao kod HOME.

2.3. BASIC-JEZIK I INTERPRETATOR

BASIC je jezik kojim se služimo pri programiranju računara. Slično »pravim«, živim jezicima, on ima svoje dozvoljene simbole, važeću gramatiku i značenje. Često se, međutim, rečju BASIC označava i BASIC-interpretator, što treba razlikovati. BASIC-interpretator je program, smešten u ROM memoriju mikroračunara, koji osposobljava računar da »govori«, odnosno »razume« BASIC-jezik. Tako, korisnik napiše program na BASIC-jeziku, a onda ga unese u računar gde ga preuzima BASIC-interpretator, koji je zadužen za kontrolu jezičke ispravnosti unetog programa, za njegovu interpretaciju i konačno izvršavanje.

Ako se ovo ima jasno u vidu, onda je pri svakom pominjanju reči BASIC jasno na šta se misli, i nema potrebe za stalnim razdvajanjem ova dva pojma.

2.4. NEPOSREDNI I PROGRAMSKI REŽIM RADA

BASIC omogućava dva režima rada. Jedan je neposredni, a drugi programski.

U neposrednom režimu računar izvršava komande odmah po njihovom unošenju sa tastature, tačnije rečeno, odmah po pritisku na taster RETURN nakon unošenja komande. Otuda i naziv neposredan režim. Za razliku od neposrednog, u programskom režimu se sve naredbe koje korisnik izdaje računaru, najpre samo smeštaju u memoriju, a računar ostaje pasivan. Tek kada korisnik završi sa unošenjem, a zatim izda računaru startni signal, započinje izvršavanje u skladu sa unetim naredbama koje sačinjavaju program. Otuda i naziv programski režim.

Na računaru Commodore 64, gotovo se sve operacije koje je on u stanju da izvršava, mogu zadati kako u neposrednom tako i u programskom režimu. Ovo nije slučaj kod svih mikroračunara, gde su često pojedine operacije rezervisane ili za neposredni režim, ili za programski režim.

Razne su situacije u kojima se može ili mora koristiti neposredni režim rada, i one će se javljati u daljem tekstu zajedno sa opisom odgovarajućih komandi. Na početku se grubo može reći da se ovaj režim koristi uglavnom za manipulaciju programom kao celinom.

Neka nam neposredni režim rada posluži za uvođenje prve naredbe BASIC-jezika sa kojom ćemo se upoznati.

2.5. NAREDBA PRINT U NEPOSREDNOM REŽIMU

Za izdavanje podataka služi naredba PRINT.

Unesite pomoću tastature:

```
PRINT 100
```

i zatim pritisnite RETURN. Računar vam odmah odgovara izdavanjem broja 100 na ekranu. Ako želite da prikažete neki tekst, unesite taj tekst pod navodnicima iza PRINT. Na primer:

```
PRINT "OVO JE TEKST"
```

a zatim pritisnite RETURN (RETURN zapravo predstavlja signal računaru da smo završili sa ukucavanjem onoga

što smo želeli. Ubuduće ćemo pritisak na RETURN podrazumevati posle svake komande u neposrednom režimu i nećemo ga više navoditi). Računar izdaje:

OVO JE TEKST

Znaci navoda su nestali. Oni su predstavljali signal računaru da je u pitanju tekst, a ne broj.

Sadržaj koji se navodi iza službene reči PRINT, čini listu koja može imati različite elemente. Videli smo da to mogu biti broj ili tekst. Ako se želi izdavanje više elemenata oni se moraju razdvajati separatorom. Unesite:

PRINT 1,"TEKST1";"TEKST2"

Računar izdaje:

1 TEKST1 TEKST 2

U ovom primeru su upotrebljeni simboli , i ; kao separatori. Drugih separatora i nema. Znak , označava da će se sledeći element iz liste izdati u sledećoj zoni (od ukupno četiri uzastopne na ekranu) dužine deset karaktera. Znak ; određuje izdavanje sledećeg elementa iz liste bez razmaka. Separatori se koriste za raspoređivanje, tj. formatiranje sadržaja koji se izdaje PRINT naredbom.

2.6. ARITMETIČKI OPERATORI I NUMERIČKE KONSTANTE

Upotrebom PRINT naredbe i neposrednog režima rada, računar se može koristiti kao prost kalkulator. BASIC sadrži u sebi deo koji se stara za vršenje osnovnih aritmetičkih operacija, sabiranja, oduzimanja, množenja, deljenja i stepenovanja. Za oznake tih operacija koriste se simboli:

\+ sabiranje,
— oduzimanje,
* množenje,
/ deljenje,
↑ stepenovanje.

Tako će naredba:

PRINT 3+9

dati rezultat 12, a naredba:

```
PRINT 3*4*2
```

rezultat 24.

Treba biti obazriv ako se koristi više aritmetičkih operacija u jednom redu, kao u prethodnom primeru, jer među operacijama postoji utvrđen prioritet.

Osim celih brojeva, računar poznaje i razlomljene brojeve. U zapisu ovih brojeva koristi se decimalna tačka (a ne decimalni zarez kao što je negde uobičajeno). Tako, na primer, naredba:

```
PRINT 3+0.14
```

daje broj 3.14 za rezultat, a:

```
PRINT 0.1/0.01
```

daje 10 kao rezultat.

Opišimo tačno, uz primere koje sve zapise brojeva, odnosno numeričkih konstanti računar prihvata.

Celobrojne konstante su celi brojevi koji se pišu bez decimalne tačke. Za pozitivne se može, a ne mora, zapisati znak + ispred broja, a za negativne se mora zapisati znak — ispred broja. Mada se istim simbolom + označava i predznak broja i aritmetička operacija sabiranja, jasno da ove dve stvari ne treba mešati. Slično je sa predznakom — i oduzimanjem. Nule ispred celobrojnih konstanti nemaju značenje i računar ih ne uzima u obzir, mada ne predstavljaju grešku u zapisu broja. Opseg celih brojeva koje računar prihvata je od —32768 do +32767.

Primeri celobrojnih konstanti:

Ispravno	Neispravno
—16	—34200
890	2.89
32767	32768
0	22,98

U slučaju celobrojnih konstanti tipično je to da ih računar izdaje u onom obliku u kome ih je primio. Ako unesemo:

```
PRINT —22780
```

računar će na ekranu izdati —22780. Ovo izgleda prilično razumljivo, ali, kao što ćemo videti, to nije slučaj sa drugom vrstom konstanti koje računar poznaje.

Razlomljene numeričke konstante (konstante u pokretnom zarezu kako se još nazivaju radi načina njihovog internog registrovanja u memoriji računara) predstavljaju razlomljene brojeve koji se pišu sa decimalnom tačkom. Ispred zapisa broja navodi se njegov znak. Ako je broj pozitivan znak se može izostaviti.

Računar prihvata dva načina zapisivanja ovih brojeva.

Prvi način zapisivanja nazvaćemo pozicioni zapis. To je uobičajeni oblik u kome decimale odvajamo decimalnom tačkom (ne zarezom). Ovaj zapis može imati najviše devet cifara. Ako unesete:

```
PRINT 123.456789
```

računar izdaje 123.456789 na ekranu. Ovako zapisan broj može biti iz opsega —999999999, +999999999. Ako se unese više od devet cifara, računar zaokružuje prema vrednosti desete cifre. Unesite:

```
PRINT 0.1234567892
```

i dobiće se 0.123456789, ali ako se unese:

```
PRINT 0.1234567876
```

računar izdaje 0.123456788.

Primeri pozicionog zapisa:

```
      1.67
      0.01
.123456789
—3456781.
     23.32
—999999999.
```

Međutim, ako unesete:

```
PRINT 0.000123456789
```

računar će izdati 1.23456789E—04 kao rezultat. Transformisao je uneti zapis (vrednost broja je ostala nepromenjena) u tzv. eksponencijalni oblik.

Svi pozitivni brojevi koji su veći od 999999999. ili manji od 0.01 biće izdati od strane računara u eksponencijalnom zapisu. Isto važi za negativne brojeve manje od —999999999. ili veće od —0.01.

Eksponencijalni zapis broja sastoji se iz tri dela:

— broja koji predstavlja mantisu,

— slova E,

— broja koji predstavlja eksponent.

Ovako zapisan broj ima vrednost:

mantisa * $10^{\text{eksponent}}$

Tako brojevi 1.23456789E—04 i 0.000123456789 imaju istu vrednost.

Mantisa se zapisuje u pozicionom obliku koji je već pomenut, a eksponent je uvek dvocifren ceo broj iz opsega [—39,+38]. Slovo E označava da je broj u eksponencijalnom obliku i razdvaja mantisu od eksponenta. I mantisa i eksponent imaju svoj predznak. Eksponent zapravo kazuje za koliko mesta bi trebalo pomeriti decimalnu tačku (ulevo za negativan eksponent, ili udesno za pozitivan eksponent) da bi se dobila vrednost broja.

Eksponencijalni oblik broja je uveden radi toga da bi se proširio opseg brojeva sa kojima računar može operisati. Za pozitivne brojeve zapisane u eksponencijalnom obliku opseg je [2.93873588E—39,1.70141183E+38]. Slično, za negativne brojeve opseg je [—1.70141183E+38,—2.93873588E—39].

Ovo se može predstaviti grafički:

```
... [**********] .. 0 .. [**********] ...
  —b          —a       a          b
```

gde su: a=2.93873588E—39,
b=1.70141183E+38.

Brojevi veći od b i manji od —b ne mogu se registrovati u računaru, a BASIC izdaje poruku o prekoračenju ako se susretne sa njima. Međutim, brojevi koji leže u intervalu [—a,a] registruju se kao nula, a BASIC ne daje nikakvu poruku.

Primeri eksponencijalnog zapisa:

Zapis	Odgovarajuća preračunata vrednost
54.876E—2	0. 54876
29E8	2900000000.
—3.76E—6	—0.00000376

Pomenimo još da korisnik može navoditi brojeve i u pozicionom i u eksponencijalnom obliku, a da će ih računar izdavati prema pomenutom pravilu: brojevi iz intervala [0.01,999999999.] i [—999999999.,—0.01] izdaju se u pozicionom, a svi ostali u eksponencijalnom obliku. Daćemo nekoliko primera koji ovo ilustruju.

PRINT 4000000000.	daje na ekranu 4E+09
PRINT 1000000001.	daje na ekranu 1E+09
PRINT 2E2	daje na ekranu 200
PRINT 0.123456789E5	daje na ekranu 12345.6789
PRINT —0.001	daje na ekranu 1E—03

Pre nego što pređemo na upotrebu programskog režima rada na računaru, izložićemo i drugi tip konstanti koje računar poznaje.

2.7. AZBUČNE KONSTANTE I NJIHOVO SPAJANJE

Azbučna konstanta je niz simbola koji je naveden u okviru znaka navoda. Kao na primer:

"OVO JE AZBUCNA KONSTANTA"

Simboli koji se mogu navoditi su slova, cifre i svi specijalni znaci osim znaka navoda ".

Smisao postojanja, pored numeričkih podataka, i ove vrste podataka u BASIC-jeziku, je pre svega u tome da se omogući laka manipulacija tekstom.

Primeri azbučnih promenljivih:

"NASLOV"
"IZNOS U DINARIMA"
"/ **** ... **** /"

Dužina azbučne konstante koju Commodore 64 prihvata, je maksimalno 80 simbola. Međutim, kada se azbučna konstanta nalazi u okviru reda koji sadrži i neku drugu informaciju, kao:

```
PRINT "AZBUCNI PODATAK"
```

onda joj je dužina ograničena na onaj broj slobodnih mesta koja ostaju kada se od 80 odbije utrošeni broj pozicija (na naredbu PRINT u ovom slučaju).

Kao što se među numeričke podatke uvode osnovne aritmetičke operacije, tako se i među azbučne podatke uvodi samo jedna osnovna operacija, koju nazivamo operacija spajanja. Ona se označava sa + i predstavlja jednostavno nastavljanje prve azbučne konstante na drugu. Tako:

```
PRINT "DOBAR"+" DAN"
```

daje DOBAR DAN na ekranu.

Ako se u okviru programa spajanjem više azbučnih konstanti formira nova, onda njena dužina može biti maksimalno 255 simbola.

2.8. PROGRAMSKI REŽIM RADA, BROJ REDA, RUN, LIST, NEW

Programski režim rada omogućuje da se od naredbi koje računar može izvršiti, sastavi program. Kao što je rečeno, korisnik tek nakon završetka unošenja svih naredbi, od kojih želi da sastavi program, daje komandu za početak izvršavanja. BASIC uzima jednu za drugom naredbe iz programa, interpretira ih, a zatim izvršava. U fazi interpretacije BASIC zapravo tumači naredbu, a zatim formira čitav niz mašinskih instrukcija namenjenih procesoru računara. Na taj način se izvršavaju naredbe BASIC-programa.

Svaki red u BASIC-programu mora imati svoj broj koji se navodi na početku reda. Ovaj broj programskog reda naziva se još i obeležje. Smisao njegovog uvođenja je utvrđivanje redosleda izvršavanja programskih redova, tj. naredbi, ali se koristi i kada želimo da između dva već

postojeća programska reda umetnemo treći. Tada ćemo upotrebiti broj reda koji je između dva već postojeća broja. Ako smo, na primer, uneli mali program:

```
10 PRINT "PRVI RED"
20 PRINT "DRUGI RED"
```

a želimo da umetnemo red među ova dva, unećemo:

```
15 PRINT "RED IZMEDJU"
```

i program će dobiti izgled:

```
10 PRINT "PRVI RED"
15 PRINT "RED IZMEDJU"
20 PRINT "DRUGI RED"
```

BASIC uvek sam uredi programske redove prema rastućem nizu obeležja.

Sada želimo da izvršimo program. Za to služi komanda RUN. Unesite RUN preko tastature (praćeno pritiskom na RETURN) i program će se izvršiti dajući:

```
PRVI RED
RED IZMEDJU
DRUGI RED
```

na ekranu.

Ali mogli smo uneti i:

```
RUN 15
```

i računar bi izdao:

```
RED IZMEDJU
DRUGI RED
```

dakle, izvršio bi se program počevši od reda sa obeležjem 15.

Na ovaj način smo ilustrovali dva moguća oblika komande RUN. Da bi se jednostavno mogao dati tačan opis neophodno je uvesti dogovore o označavanju koje ćemo ubuduće stalno koristiti.

Velikim slovima označavaćemo sve ono što se mora na identičan način zapisati.

Malim slovima uokvireno simbolima < i >, označavaćemo ono što treba zameniti konkretnim sadržajem. Pri ovom zamenjivanju simboli < i > će se uklanjati (ovo nisu simboli BASIC-jezika, već služe za opisivanje).

Sadržaje koji su opcioni, tj. koji se mogu izbaciti ili navesti, navodićemo u okviru uglastih zagrada [i]. Njih takođe ne treba navoditi kada se njima uokvireni sadržaji uključuju u zapis.

Koristeći dogovoreno, komandu RUN opisujemo na sledeći način:

RUN[<broj reda>]

Komentarišimo sa nekoliko reči ovaj opis. Prvo, RUN je zapisano velikim slovima što znači da ga tako moramo i uzeti. Kako je sve ostalo navedeno u okviru uglastih zagrada, možemo taj sadržaj izostaviti. U tom slučaju ostaje samo RUN što ima značenje: izvršiti ceo program, od početka do kraja. Međutim, ako uključimo i opcioni sadržaj, koji je u ovom slučaju <broj reda>, onda, s obzirom na upotrebljena mala slova, moramo upisati neki broj reda i izbaciti simbole < i >. Tako dolazimo do naredbe:

RUN 15

koja označava izvršavanje programa počevši od programskog reda sa brojem 15.

Sledeća komanda koju ćemo upoznati omogućava izdavanje teksta samog programa, tj. njegov prikaz na ekranu. Unesite LIST (praćeno sa RETURN što, kao što je rečeno, stalno podrazumevamo), i na ekranu ćete dobiti ispisan sadržaj programa.

Tačan oblik ove komande je:

LIST[[<prvi red>][—][<poslednji red>]]

Ako se navede samo LIST izdaje se ceo program. Ako se navedu <prvi red> i <poslednji red> izdaje se program od reda sa obeležjem <prvi red> do reda sa obeležjem <poslednji red>. Ako se izostavi <prvi red>, odnosno <poslednji red> izdaje se program od početka do obeležja <poslednji red>, odnosno od obeležja <prvi red> do kraja.

Primeri:

LIST	izdaje ceo program.
LIST 10—50	izdaje redove od 10 do 50.
LIST —100	izdaje redove od početka do 100.
LIST 20—	izdaje redove od 20 do kraja.
LIST 30	izdaje samo red 30.

Korišćenjem naredbe LIST i ekranskog editora lako se menjaju pojedini programski redovi, tako što se najpre pomoću LIST dovedu na ekran, zatim izmene korišćenjem ekranskog editora, a zatim pritiskom na RETURN upišu umesto starog sadržaja programskog reda. Ako se ponovo izda LIST komanda, prikazaće se izmenjen program.

Kada želimo da memoriju potpuno oslobodimo sadržaja da bismo mogli da slobodno upisujemo novi program, izdaćemo komandu NEW. Zapis ove komande je:

NEW

2.9. PROMENLJIVE — NUMERIČKE I AZBUČNE

Promenljiva je oznaka (ime) za određenu grupu registara u memoriji u koje se može upisivati sadržaj prema želji. Taj upisani sadržaj zovemo vrednost promenljive. Za razumevanje i korišćenje promenljivih može se smatrati da promenljiva direktno prima pojedine vrednosti (kao što se to radi u matematici).

U BASIC-jeziku za Commodore 64 postoje dve vrste promenljivih: numeričke i azbučne. Numeričke mogu biti celobrojne i realne, pri čemu celobrojne uzimaju cele brojeve kao svoje vrednosti, a realne uzimaju bilo koji broj kao vrednost.

Za vrednosti promenljivih, opsege, načine zapisa i sl. važi sve ono što je rečeno za konstante.

Imena promenljivih se sastoje od dva simbola, i to prvi može biti slovo od A—Z, a drugi može biti slovo od A—Z ili cifra 0—9. Tip promenljive se razlikuje prema tome što realne numeričke promenljive nemaju nikakav dodatak iza imena. Celobrojne imaju dodat znak % iza imena a

azbučne imaju dodat znak $ iza imena. Primeri imena promenljivih:

X — realna
XY — realna
Z1 — realna
X% — celobrojna
K5% — celobrojna
A$ — azbučna
TE$ — azbučna
A9$ — azbučna

Imena promenljivih se mogu zapisati i sa više od dva simbola (ne računajući %, odnosno $), ali BASIC će izdvojiti i prihvatiti samo prva dva. Tako će promenljive VISINA i VI biti identične, tj. ista promenljiva, sličnc IZNOS% i IZ%.

Promenljive su uvedene da bi se njima dodeljivale vrednosti. Kako se to radi? Postoji više načina dodeljivanja vrednosti koje ćemo sresti u daljem tekstu, ali jedan od najvažnijih je putem naredbe dodeljivanja.

2.10. DODELJIVANJE VREDNOSTI, ARITMETIČKI I AZBUČNI IZRAZ

Dodeljivanje vrednosti promenljivoj vrši se korišćenjem znaka jednakosti. Ako se zapiše:

```
A=30.5
```

promenljiva A dobija vrednost 30.5. Dodeljivanje se uvek vrši zdesna nalevo. Tako u malom programu:

```
10 A=30.5
20 B=A
```

promenljiva B dobija vrednost 30.5.

Tačan opis naredbe glasi:

[LET]<ime promenljive>=<izraz>

Službena reč LET je opciona i gotovo se nikad ne koristi, jer je i bez nje jasno da je u pitanju naredba dode

ljivanja. Na desnoj strani jednakosti stoji <izraz>. To je važan pojam koji moramo definisati. Prethodno samo napomenimo da su i konstante i promenljive specijalan slučaj izraza, pa se tako gornji primeri uklapaju u opštu definiciju.

Izraze ćemo podeliti na dve grupe: aritmetičke i azbučne.

Aritmetički izraz čine jedan ili više argumenata povezanih znacima aritmetičkih operacija. Argumenti aritmetičkog izraza su konstante i/ili promenljive. Aritmetički izraz je zapis postupka kojim se dobija jedna numerička vrednost. Kao što je već rečeno, aritmetičke operacije se označavaju sa + (sabiranje), — (oduzimanje), * (množenje), / (deljenje) i ↓ (stepenovanje). Izračunavanje vrednosti aritmetičkog izraza vrši se prema prioritetu operacija. Viši prioritet znači ranije izvršavanje.

Prioriteti operacija su:

1	najviši	↑
2	niži	— (promena znaka)
3	niži	*,/
4	najniži	+,—

U okviru istog prioriteta operacije se vrše sleva nadesno. Ako se želi promeniti redosled vršenja operacija u okviru aritmetičkog izraza, koriste se zagrade (isključivo male), koje se takođe mogu koristiti kao simboli u aritmetičkom izrazu. Njihovo značenje i upotreba su isti kao u matematici. Prioritet vršenja operacija u okviru zagrada je najviši. One se često koriste i kada nisu neophodne, da bi se aritmetički izraz napisao u formi jasnoj za onoga ko čita i eventualno dograđuje program. Zagrade se mogu uklapati jedna u drugu po dubini do (kod računara C—64) najviše 10 nivoa. U tom slučaju prioriteti opadaju od unutrašnjih ka spoljnim zagradama.

Treba još reći da vrednost aritmetičkog izraza zavisi od tipa promenljive kojoj se ta vrednost dodeljuje. Ako je promenljiva celobrojna, onda se vrednost transformiše u ceo broj (i ne sme izaći iz opsega —32768,+32767), a ako je realna onda će vrednost biti u obliku u kome se u računaru registruju vrednosti realnih promenljivih (pre

izračunavanja se uvek svaka celobrojna vrednost najpre transformiše u oblik u kome se interno registruju vrednosti realnih promenljivih). Dakle sve "međuvrednosti" su realne, a tek se "završna" vrednost eventualno transformiše u celobrojni oblik.

Primeri aritmetičkih izraza:

Izraz	Redosled izvršavanja operacija
A+B*C	*,+
(A+B)*C	+,*
X1*X2/X3+X4	*,/,+
(E—F)↑M%	—,↑
(A+B)/(C—D)	+,—,/
Q—W%*B+R	*,—,+

Unesite sledeći program:

```
10 A=8
20 B=2.5
30 C=0.5
40 D=A—B*C
50 PRINT "D=";D
```

a zatim (koristeći ekranski editor) menjajte programski red 40 i ispitujte razne aritmetičke izraze i redoslede izvršavanja operacija u okviru njih. Promenite D u D%. (u gornjem programu se u redu 50 u okviru naredbe PRINT navodi ime promenljive D. Vrednost koja se izdaje je vrednost te promenljive. Pored konstanti i promenljivih, u naredbi PRINT mogu figurisati i aritmetički izrazi).

Napomena. Često se u naredbi dodeljivanja koristi sledeći način:

```
A=A+1
```

koji može na prvi pogled izgledati čudan. Ali, ovo je regularna upotreba i znači da se vrednost promenljive A uvećava za jedan i dodeljuje kao vrednost opet istoj promenljivoj. Drugim rečima, simbol = ne znači jednakost, već dodeljivanje.

Azbučni izraz se sastoji od azbučnih argumenata i azbučne operacije spajanja. Vrednost azbučnog izraza se

može dodeliti jedino azbučnoj promenljivoj, kao što se vrednost aritmetičkog izraza može dodeliti samo numeričkoj promenljivoj. U suprotnom BASIC izdaje poruku o grešci.

Primeri azbučnih izraza:

```
"IME"+" PREZIME"
"IME"+P$+"ADRESA"
A$+B$+C$
```

Unesite program:

```
10 A$="AZBUCNI"
20 B$=" IZRAZ"
30 C$="OVO JE ———"+A$+B$
40 PRINT C$
```

2.11. UNOS PODATAKA SA TASTATURE — INPUT

Ni u jednom do sada navedenom primeru nije bilo moguće uneti neki podatak u program nakon što se on startuje komandom RUN. Drugim rečima nije bilo "dijaloga" između korisnika i programa. Da bi se ovo omogućilo, uvedena je naredba INPUT.

Unesite sledeći program (prethodno unesite NEW da bi se oslobodili prethodnog):

```
10 INPUT A
20 PRINT "UNELI STE BROJ";A
```

i startujte njegovo izvršavanje komandom RUN. Na ekranu se dobija znak pitanja ? i računar čeka. Na vas je red da preko tastature unesete vrednost za promenljivu A. Ako unesete 25.78 i nakon toga pritisnete RETURN, računar izdaje:

```
UNELI STE BROJ 25.78
```

Dakle, računar upozorava kada treba uneti potrebnu vrednost, zatim čeka, a kada završite unošenje (signal RETURN), računar unetu vrednost dodeljuje promenljivoj A, a zatim nastavlja sa izvršavanjem programa. Ako želite

da u jednom redu unesete dve vrednosti, razdvojićete imena promenljivih zarezom:

```
10 INPUT A,B
20 PRINT "UNELI STE BROJEVE";A;" I";B
```

Kada se izda znak pitanja pri izvršavanju reda 10, treba uneti u jednom redu odvojene zarezom dve vrednosti.

Tačan opis naredbe ulaza glasi:

INPUT["<tekst>";]<lista imena promenljivih>

Ako se navede opcioni "<tekst>"; iz opisa naredbe, taj tekst će se izdati kao komentar kada se računar zaustavi očekujući unos podataka. Simbol ; je tada obavezan kao separator. Obavezni deo naredbe je <lista imena promenljivih>. Ona predstavlja niz imena promenljivih međusobno odvojenih zarezima.

P r i m e r:

```
10 INPUT "UNESITE DVA BROJA";A,B
20 C=A+B
30 PRINT "ZBIR UNETIH BROJEVA JE";C
```

Program vas obaveštava šta i kada treba uneti.

P r i m e r:

```
10 INPUT "UNESITE VASE IME";I$
20 INPUT "UNESITE VASE PREZIME";P$
30 PRINT "VI STE ";I$+"        "+P$
```

Program uz dijalog prihvata dve azbučne vrednosti. U redu 30 koristi se azbučna konstanta, koja se sastoji od tri "blanko" simbola.

Naredba INPUT se ne može izdavati u neposrednom režimu rada, već se jedino može koristiti u programu.

O drugim načinima unosa podataka u program biće reči kasnije.

2.12. ZAUSTAVLJANJE I NASTAVAK IZVRŠAVANJA PROGRAMA, STOP, END, RUN/STOP, CONT

Postoje dve naredbe za završetak izvršavanja programa. To su STOP i END. One se najčešće pišu na kraju programa, mada se mogu nalaziti ma gde u programu.

Naredba STOP daje izveštaj BREAK IN LINE xxx što znači **prekid na redu** xxx, a zatim se štampa poruka READY. Naredba END ima identičnu funkciju prekidanja izvršavanja programa. Jedina je razlika što se izdaje samo READY poruka i ništa više.

Program se takođe može prekinuti i neposrednom intervencijom sa tastature. Treba pritisnuti taster na kome piše RUN/STOP. Efekat je potpuno isti kao kada program u toku izvršavanja naiđe na naredbu STOP.

Bez obzira na koji je od ova tri načina program zaustavljen, njegovo se izvršavanje može nastaviti (naravno ako je ostao još neki neizvršen deo programa) unošenjem komande CONT u neposrednom režimu.

Komanda CONT nastavlja izvršavanje programa na onom mestu gde je bio prekinut nekim od pomenuta tri načina.

Naredba END se najčešće piše na kraju programa i tako se program završava. Naredba STOP se može koristiti i kao naredba END, ali se ona često umeće u program na onim mestima na kojim želimo privremeno da prekinemo njegovo izvršavanje radi, npr., ispitivanja trenutnih vrednosti promenljivih. Kada se program prekine, korišćenjem naredbe PRINT u neposrednom režimu mogu se pregledati sve vrednosti promenljivih. Ovo je vrlo pogodno pri razvijanju i testiranju programa. Zatim se sa CONT nastavlja izvršavanje programa. Međutim, ako posle prekida programa izmenite neki programski red ili vrednost promenljive u neposrednom režimu, program se ne može nastaviti komandom CONT. Dobija se poruka CAN'T CONTINUE (ne može se nastaviti).

Primer:

```
10 A=10
20 B=20
30 STOP
40 PRINT "A=";A,"B=";B
```

Izdajte komandu RUN, a kada se program prekine na redu 30, u neposrednom režimu štampajte vrednosti za A i B, tj. unesite sa tastature:

PRINT A,B

i na ekranu dobijate vrednosti 10 i 20. Zatim nastavite izvršavanje programa pomoću CONT komande.

2.13. GRANANJE PROGRAMA — USLOVNI I BEZUSLOVNI PRELAZ, IF ... THEN, GOTO

U svim primerima do sada programi su se izvršavali naredba za naredbom tačnim redom kojim su navedene u programskim redovima sa rastućim obeležjima. Potrebno je imati na raspolaganju mogućnost da se na određenom mestu u programu krene jednim ili drugim putem u zavisnosti od toga da li je neki uslov ispunjen ili ne. Ovakve situacije su gotovo uvek prisutne kada razradimo postupak rešavanja problema. Naredba, koja vrši prelaz na osnovu ispunjenosti uslova, je osnovni faktor kojim se daje "inteligencija" programu.

Da bi se izložilo šta pomenuti uslov može biti, kao i da bi se objasnila naredba uslovnog prelaza, potrebno je najpre uvesti relacione i logičke operatore, kao i relacione i logičke izraze.

Relacioni operatori (operatori poređenja) predstavljeni su simbolima:

Operator	Značenje
=	jednako
>	veće
>=	veće ili jednako
<	manje
<=	manje ili jednako
<>	različito

Oni operišu aritmetičkim ili azbučnim izrazima, a rezultat njihove primene je "tačno" ili "netačno". Pomoću njih formiramo **relacione izraze**, koji predstavljaju dva aritmetička (ili azbučna) izraza povezana jednim od re-

lacionih operatora. Treba imati na umu da i same konstante i/ili promenljive predstavljaju aritmetički, odnosno azbučni izraz. Korišćenjem relacionih izraza utvrđujemo da li važi neki od odnosa koji je predstavljen upotrebljenim operatorom. Relacioni izrazi imaju vrednost "tačno" ili "netačno". Na primer:

Relacioni izraz	Vrednost
3>2	tačno
320=320	tačno
—9>=—30	netačno
A<B↑2	zavisi od veličina A i B
(X1/C)=A/B*C	zavisi od X1,C,A i B

Ako se relacioni operatori primenjuju na azbučne izraze onda se porede odgovarajući simboli polazeći sleva nadesno. Pri ispitivanju za >, < , >= i <= koristi se uređenje engleske abecede za slova A>B>C... (ali ako je korišćen i neki simbol koji nije slovo, onda je važeći redosled utvrđen ASCII kodom, koji se može naći u tabeli ASCII kodova). Primeri:

Relacioni izraz	Vrednost
"A"<"C"	tačno
"ABC"<"ABD"	tačno
"VRANA"="SVRAKA"	netačno
A$<>"IME"	zavisi od A$
A$+B$<>C$+"IZNOS"	zavisi od A$, B$ i C$

Treba još reći da računar Commodore 64 kodira "tačno" brojem —1, a "netačno" brojem 0, što nas u ovom momentu ne zanima.

Logički operatori se označavaju sledećim simbolima:

Operator	Značenje
AND	konjunkcija (operator "i")
OR	disjunkcija (operator "ili")
NOT	negacija (operator "ne")

operišu sa relacionim izrazima, dakle vrednostima "tačno" i "netačno" kao argumentima, a rezultat im je takođe vrednost "tačno" ili "netačno". Logički operatori AND i OR

imaju po dva argumenta, a NOT samo jedan argument, koji se piše desno od NOT. Operatori se definišu na sledeći način:

AND daje vrednost "tačno" ako oba argumenta imaju vrednost "tačno". U ostalim slučajevima ("tačno"AND"netačno", "netačno"AND"tačno" i "netačno"AND"netačno") daje vrednost "netačno".

OR daje vrednost »tačno« ako bar jedan od argumenata ima vrednost "tačno". Inače daje vrednost "netačno".

NOT daje vrednost "tačno" za argument sa vrednošću "netačno". Ako argument ima vrednost "tačno", NOT će dati vrednost "netačno".

Ovde su kao argumenti logičkih operatora namerno uzeti samo relacioni izrazi. O široj upotrebi logičkih operatora biće reči u narednim glavama.

Pomoću logičkih operatora formiramo **logičke izraze** koji su sastavljeni od relacionih izraza povezanih znacima za logičke operacije AND, OR i NOT. Vrednost logičkih izraza je "tačno" ili "netačno". Redosled izvršavanja logičkih operacija je sleva nadesno. Kao i u artimetičkim izrazima, tako se i u logičkim izrazima mogu koristiti zagrade za menjanje redosleda izvršavanja.

Tabela prioriteta, po opadajućem redosledu, među svim uvedenim operacijama (aritmetičkim, relacionim i logičkim) izgleda ovako:

Operator	Naziv
↑	stepenovanje
—	promena znaka
*,/	množenje, deljenje
+,—	sabiranje, oduzimanje
=,>,>=,<,<=,<>	relacioni operatori
NOT	logičko NE
AND	logičko I
OR	logičko ILI

Primer logičkih izraza:

Primer 1: A>5 AND A<10

U ovom primeru se najpre računa vrednost relacionog izraza A>5, zatim vrednost relacionog izraza A<10, a na

kraju se na bazi ta dva rezultata dobija vrednost "tačno" ili "netačno" celog logičkog izraza. Tačno će se dobiti ako je A broj koji leži između 5 i 10.

Primer 2: (X1>X2) OR (X3=X4 AND X5<>X6)

Redosled izračunavanja je: X1>X2, X3=X4, X5<>X6, AND, OR. Vrednost je ili "tačno" (što se kodira sa —1) ili "netačno" (kodira se sa 0), što zavisi od vrednosti X1,X2,X3,X4,X5,X6. Vrednost ovog logičkog izraza za neke vrednosti X1, ... X6 data je u tabeli (T označava "tačno", a N označava "netačno"):

X1	X2	X3	X4	X5	X6	Izraz
T	T	T	T	T	T	N
N	N	N	N	N	N	N
T	N	T	N	T	N	T

Primer 3: A$="IME" AND A>5

Ovde se najpre izračunava vrednost relacionog izraza A$="IME", zatim izraza A>5, a na kraju ceo logički izraz dobija svoju vrednost na bazi logičkog operatora AND. Neke vrednosti ovog logičkog izraza su:

A$	A	Izraz
IME	6	T
IME	4	N
PREZIME	10	N

Vratimo se sada naredbi uslovnog prelaza. Ona ima oblik:

IF<uslov>THEN<broj reda>

ili:

IF<uslov>GOTO<broj reda>

Iza službene reči IF navodi se <uslov> koji može biti logički ili relacioni izraz. Ako izraz ima vrednost "tačno",

kažemo da je u tom slučaju <uslov> ispunjen, prelazi se na izvršavanje naredbe čiji je <broj reda> naveden. Ako <uslov> ima vrednost "netačno", tj. ako uslov nije ispunjen, prelazi se na izvršavanje programskog reda koji neposredno sledi iza naredbe IF. Oba oblika (sa THEN i sa GOTO) imaju isto značenje.

Primeri:

```
IF A>5 THEN 100
IF X1>10 AND X2<20 GOTO 200
```

Napišimo program koji ispituje da li je uneti broj iz zadatog intervala, i daje izveštaj o tome.

```
10 INPUT "UNESITE LEVU I DESNU GRANICU
   INTERVALA";L,D
20 INPUT "UNESITE JEDAN BROJ";B
30 IF B>=L AND B<=D THEN 60
40 PRINT "BROJ JE IZVAN INTERVALA"
50 STOP
60 PRINT "BROJ PRIPADA INTERVALU"
70 STOP
```

Ako je uslov u redu 30 ispunjen, prelazi se na izvršavanje reda 60, a zatim 70. Ako uslov nije ispunjen, izvršava se red 40, a zatim 50.

Postoji još jedan oblik naredbe IF koji je dozvoljen. Da bismo njega objasnili moramo pomenuti mogućnost navođenja više naredbi u jednom programskom redu.

Do sada smo podrazumevali da se u jednom programskom redu nalazi jedna naredba. U BASIC-u za Commodore 64 može se navoditi i više naredbi u istom redu, ali se moraju razdvajati simbolom : (dve tačke). Tako se može napisati program:

```
10 INPUT A,B
20 Z=A+B:R=A—B:P=A*B:K=A/B
30 PRINT Z,R,P,K
40 END
```

koji u istom programskom redu 20 računa zbir, razliku, proizvod i količnik unetih brojeva, dakle izvršava četiri

naredbe dodeljivanja. Naredbe u okviru istog programskog reda izvršavaju se sleva nadesno, osim ako neka od naredbi sama ne određuje drugačije (recimo prelaz), a ako se izvrše sve naredbe u redu, prelazi se na sledeći red.

Alternativni oblik naredbe IF glasi:

IF<uslov>THEN<naredbe>

Ovaj oblik naredbe IF ima isto značenje kao i ranije navedeni, samo se u slučaju ispunjenog uslova redom izvršavaju naredbe koje su navedene iza THEN. Te naredbe moraju među sobom biti razdvojene simbolom : (dve tačke) kao što je već rečeno. Na primer, program:

```
10 X1=5:X2=0.5:Z=0:R=0
20 INPUT A$
30 IF A$="DA" THEN Z=X1+X2:R=X1—X2
40 PRINT Z,R
50 END
```

izdaje zbir i razliku dva broja X1 i X2 pod uslovom da je korisnik uneo DA. Ako je korisnik uneo bilo šta drugo, program izdaje dve nule kao vrednosti za Z i R.

Pored naredbe uslovnog prelaza postoji i naredba bezuslovnog prelaza. Ona se koristi kada u nekoj tački programa treba, obavezno, bezuslovno preći na neki određeni programski red. Oblik ove naredbe je:

GOTO<broj reda>

Ova naredba označava bezuslovan skok na red sa obeležjem <broj reda>.

P r i m e r:

```
10 PRINT "BESKONACNA PETLJA"
20 GOTO 10
```

Izvršavanje ovog programa prekinućete pritiskom na taster RUN/STOP.

Daćemo nekoliko primera programa koji ilustruju upotrebu naredbi koje su do sada izložene.

U prvom primeru treba sastaviti program koji u dijalogu sa korisnikom izračunava ukupnu površinu stana kao i ukupnu cenu parketa potrebnog da se prekrije pod stana:

```
10 P=0:U=0
20 INPUT "UNESITE CENU KVADRATNOG METRA
   PARKETA";C
30 INPUT "UNESITE DUZINU I SIRINU
   PROSTORIJE";D,S
40 P=P+D*S:U=U+D*S*C
50 INPUT "JOS PROSTORIJA (DA — NE)";A$
60 IF A$="DA" OR A$="D" THEN 30
70 PRINT "UKUPNA POVRSINA PROSTORIJA";P
80 PRINT "UKUPNA CENA PARKETA";U
90 END
```

U redu 10 se postavljaju nule za početne vrednosti ukupne površine P i ukupne cene U. Tim promenljivim se za svaku novu prostoriju dodaju vrednosti površine i cene parketa te prostorije — red 40. U redu 50 se prihvata, a u redu 60 analizira korisnikov odgovor tipa DA—NE. Prelazi se na novu prostoriju ako korisnik unese DA ili ako greškom unese samo D. U ostalim slučajevima program izdaje završne vrednosti i staje.

U sledećem primeru treba sastaviti program koji traži podatke o vašoj visini i visini vašeg oca, a zatim štampa izveštaj o tome ko je viši.

```
10 INPUT "UNESITE VASU VISINU U
   CENTIMETRIMA";V1
20 INPUT "UNESITE VISINU VASEG OCA (U CM)";V2
30 IF V1—V2 THEN PRINT "JEDNAKE STE
   VISINE":STOP
40 IF V1>V2 THEN PRINT "VI STE VISI OD
   OCA":STOP
50 PRINT "VI STE NIZI OD OCA":STOP
```

Očigledno je šta se događa na pojedinim programskim redovima, samo napomenimo da se u redovima 30, 40 i 50 može STOP zameniti sa END, da bi se izbeglo izdavanje poruke BREAK AT LINE xx.

Napišite sada program koji omogućuje korisniku da odabere i obavi jednu od sledeće dve operacije koje program može da izvrši: 1 — brisanje samog sebe iz memorije i 2 — izdavanje samog sebe na ekranu.

```
10 PRINT "BRISANJE OVOG PROGRAMA IZ
   MEMORIJE — 1"
20 PRINT "IZDAVANJE OVOG PROGRAMA NA
   EKRANU — 2"
30 INPUT A
40 IF A<>1 AND A<>2 THEN 10
50 IF A=1 THEN NEW
60 LIST
70 END
```

Napomenuto je ranije da se i komande BASIC-interpretatora računara C-64 mogu izvršavati u programskom režimu. To se koristi u ovom programu. Red 50 će ako je A=1 obrisati memoriju, tako da se program ne može ponovo startovati sa RUN, jer više neće biti prisutan u memoriji. Red 60 izdaje program na ekranu.

2.14. PROGRAMSKI CIKLUS — FOR . . . NEXT

Programski ciklus je skup naredbi čije izvršenje treba ponoviti tačno određeni broj puta. Programski ciklus se jednostavno formira umetanjem naredbe FOR na početak i naredbe NEXT na kraj segmenta koji treba izvršiti više puta.

Oblik naredbe kojom se započinje ciklus je:

FOR<ime promenljive>=<početna vrednost>
TO<završna vrednost>[STEP<priraštaj>]

<ime promenljive> označava numeričku promenljivu koja se naziva indeks ciklusa.

<početna vrednost> označava aritmetički izraz koji definiše početnu vrednost indeksa.

<završna vrednost> označava aritmetički izraz koji određuje završnu vrednost indeksa.

Opcioni <priraštaj> označava aritmetički izraz koji predstavlja priraštaj u uzastopnom povećavanju vrednosti indeksa od početne do završne vrednosti. Ako se ne navede podrazumeva se 1.

Kako su numeričke konstante i promenljive, kao što je već rečeno, specijalni slučajevi aritmetičkog izraza, jasno je da one mogu biti <ime promenljive>,<završna vrednost> ili <priraštaj>.

Iza svih naredbi koje se uključe u ciklus navodi se poslednja naredba programshog ciklusa koja ima oblik:

NEXT[<ime promenljive>][,<ime promenljive>]...

gde je <ime promenljive> isti indeks pomenut u FOR naredbi, koji se ovde može ali i ne mora navesti. Radi toga što se ciklusi mogu umetati jedan u drugog, u naredbi NEXT predviđeno je eventualno navođenje indeksa više od jednog ciklusa.

Programski ciklus se realizuje na sledeći način:

Najpre indeks dobije vrednost jednaku <početnoj vrednosti>, zatim se izvršavaju BASIC naredbe smeštene između FOR i NEXT, a kada se dođe do NEXT, indeks se uveća za <priraštaj> i uporedi sa <završnom vrednošću>. Ako <završna vrednost> nije dostignuta, ponovo se izvršavaju naredbe sadržane u okviru ciklusa, itd. Kada se <završna vrednost> prvi put dostigne (ili prvi put premaši) program nastavlja izvršavanjem prve naredbe koja sledi za naredbom NEXT. Napomenimo da se programski ciklus uvek izvršava bar jednom.

Ako se više programskih ciklusa umeće jedan u drugog, onda se svaki sledeći mora nalaziti unutar prethodnog, sa eventualno zajedničkim završetkom. Ciklusi se ne smeju »seći«.

Primeri ispravnih naredbi FOR:

```
FOR I=1 TO 100
FOR A=I TO I ↑2 STEP 0.1
FOR Z=—1 TO —10 STEP —0.2
```

Radi ilustrovanja upotrebe FOR...NEXT napišimo program koji sabira prvih N prirodnih brojeva.

```
10 INPUT "UNESITE BROJ SABIRAKA N";N
20 S=0
30 FOR I=1 TO N
40 S=S+I
50 NEXT I
60 PRINT "ZBIR PRVIH";N;" BROJEVA JE";S
70 END
```

Programski ciklus bi se mogao realizovati i bez upotrebe FOR . . . NEXT (korišćenjem dve naredbe dodeljivanja i jedne naredbe uslovnog prelaza. Kako?)

2.15. NUMERIČKE FUNKCIJE

Postoji određen broj matematičkih funkcija koje BASIC može izračunavati direktnim navođenjem njihovih imena. To su: ABS, INT, SGN, SQR, EXP, LOG, RND, SIN, COS, TAN i ATN.

Za numeričke funkcije SQR, EXP, LOG, SIN, COS, TAN i ATN važi da im se vrednost u najvećem broju slučajeva dobija sa približnom tačnošću (koja nije uzrokovana ograničenom tačnošću računara, već prirodom tih funkcija čije se vrednosti ne mogu tačno izračunati primenom konačnog broja operacija).

Ove funkcije su nazvane numeričkim (za razliku od azbučnih) radi toga što su im i argument i vrednost numerički.

Pored navedenih numeričkih funkcija spremnih za upotrebu, može se uvesti i proizvoljna funkcija koju definiše korisnik.

Opišimo redom sve numeričke funkcije:

ABS(<aritmetički izraz>) izračunava apsolutnu vrednost aritmetičkog izraza. Apsolutna vrednost nekog broja X definiše se kao:

$$\mathrm{ABS}(X)=\begin{cases} X \text{ za } X>0 \\ 0 \text{ za } X=0 \\ -X \text{ za } X<0 \end{cases}$$

Koristite PRINT i neposredni režim rada za primere koji slede.

P r i m e r: ABS(—10) daje vrednost 10

INT(<aritmetički izraz>) izračunava najveći ceo deo broja.

Primer:

INT(8.21) daje vrednost 8
INT(—4.3) daje vrednost —5
INT(1.99) daje vrednost 1

SGN(<aritmetički izraz>) daje vrednost prema znaku aritmetičkog izraza na sledeći način:

$$SGN(X)=\begin{cases} 1 \text{ za } X>0 \\ 0 \text{ za } X=0 \\ -1 \text{ za } X<0 \end{cases}$$

Primer:

SGN(—2) daje vrednost —1
SGN(5) daje vrednost 1

SQR(<aritmetički izraz>) izračunava vrednost kvadratnog korena. Vrednost <aritmetičkog izraza> ne sme biti negativna, jer za negativne vrednosti funkcija kvadratnog korena nije definisana.

Primer:

SQR(4) daje vrednost 2
SQR(20) daje vrednost 4.47213595

EXP(<aritmetički izraz>) izračunava vrednost eksponencije funkcije (osnova je e). Vrednost <aritmetičkog izraza> ne sme biti veća od 88.0296919), jer to dovodi do prekoračenja.

Primer:

EXP(0) daje vrednost 1
EXP(2) daje vrednost 7.3890561

LOG(<aritmetički izraz>) izračunava vrednost logaritma za osnovu e. Vrednost <aritmetičkog izraza> mora biti veća od nule, jer logaritamska funkcija nije definisana za negativne vrednosti i nulu.

Primer: LOG(5) daje vrednost 1.60943791

RND(<broj>) daje slučajan (nepredvidiv) broj iz intervala (0,1), pri čemu nula i jedinica nisu uključene. Argument <broj> je lažan argument, jer se od njega koristi samo znak pri generisanju slučajnog broja, i taj znak određuje na koji način će se formirati broj pri navođenju RND.

U računaru se zapravo generišu pseudoslučajni brojevi, koji se formiraju na osnovu određenog internog postupka koji na bazi jedne vrednosti generiše sledeću. Takav postupak zahteva početnu vrednost. Ako se krene od jedne početne vrednosti i proizvede npr. pedeset brojeva korišćenjem RND pedeset puta, dobiće se identičan niz sa nizom koji se dobija ako se ponovo krene od iste početne vrednosti. U računaru Commodore 64 početna vrednost se učitava sa ugrađenog sistemskog časovnika kada se računar uključi.

Ako se za <broj> uzme nula, sa sistemskog časovnika se učitava početna vrednost, koja se i izdaje kao vrednost za slučajni broj. Ovo se najčešće koristi za slučajno započinjanje niza slučajnih brojeva.

Ako se navede pozitivan <broj>, generisaće se slučajan broj (pomenutim utvrđenim internim postupkom), na bazi slučajnog broja koji je bio generisan u prethodnom pozivu RND. Ovo je najčešći način upotrebe, jer se time može dobiti niz slučajnih brojeva, što je gotovo uvek i potrebno u programima.

Ako se navede negativan <broj> dobija se za svaki broj uvek ista vrednost RND pri ponovljenom pozivu, jer se stalno postavlja ista početna vrednost. Ovaj, na prvi pogled nekoristan način, nalazi upotrebu u testiranju rada programa koji koriste RND.

Primer:

```
10 X=RND(0)
20 FOR I=1 TO 100
30 X=RND(1)
40 PRINT X
50 NEXT I
60 END
```

U navedenom programu red 10 postavlja slučajan početak za niz slučajnih brojeva koji se generišu u redu 30. Ako uključite računar, unesete ovaj program i izvršite ga, dobićete niz slučajnih brojeva koji će biti različit od niza koji ćete dobiti ako računar isključite, a zatim ponovo uključite, unesete program i izvršite ga. Kada bi se izbacio red 10, program bi pri svakom ponovnom unošenju i izvršavanju generisao isti niz slučajnih brojeva, jer bi početna vrednost bila uvek ista (učitana sa sistemskog časovnika koji se pri uključenju računara uvek startuje od nule).

N a p o m e n a: Slučajni brojevi iz nekog šireg intervala (A,B) mogu se dobiti na osnovu sledećeg izraza:

A+(B—A)*RND(1)

Slučajan izbor jednog od datih celih brojeva, npr. 1,2,3,4,5,6, može se realizovati korišćenjem izraza:

INT(RND(1)*6)+1

SIN(<aritmetički izraz>) izračunava vrednost trigonometrijske funkcije sin(x). Argument treba izraziti u radijanima.

P r i m e r: SIN(0) daje vrednost 0.

COS(<aritmetički izraz>) izračunava vrednost trigonometrijske funkcije cos(x). Argument treba izraziti u radijanima.

P r i m e r: COS(0) daje vrednost 1.

TAN(<aritmetički izraz>) izračunava vrednost trigonometrijske funkcije tg(x). Argument treba izraziti u radijanima. Voditi računa o tačkama prekida tangensne funkcije.

P r i m e r: TAN(0) daje vrednost 0.

ATN(<aritmetički izraz>) izračunava vrednost inverzne trigonometrijske funkcije arctg(x). Rezultat je u radijanima.

P r i m e r: ANT(0) daje vrednost 0.

N a p o m e n a: Vrednosti ostalih trigonometrijskih funkcija moraju se računati na bazi navedenih trigonometrijskih funkcija korišćenjem odgovarajućih identičnosti.

2.16. AZBUČNE FUNKCIJE

Pored numeričkih funkcija, u okviru BASIC-a postoje i azbučne funkcije. To su funkcije kod kojih su argument ili/i vrednost azbučnog tipa. U BASIC za Commodore 64 ugrađene su sledeće azbučne funkcije: ASC, LEN, VAL, CHR$, LEFT$, MID$, RIGHT$ i STR$.

Azbučne funkcije koje na kraju imena nemaju znak $ daju numeričke vrednosti (argumenti su im azbučni). Funkcije koje na kraju imena imaju znak $ daju azbučnu vrednost.

ASC(<azbučni izraz>). Vrednost ove funkcije je numerička, a argument azbučni. Daje ASCII kôd prvog simbola iz <azbučnog izraza>. ASCII je skraćenica za Američki standardni kôd kojim su kodirani svi simboli. Kôd je prihvaćen u celom svetu. Tabela ASCII kodova se nalazi u dodatku, na kraju knjige.

P r i m e r: ASC("DAN") daje vrednost 68, što je ASCII kôd za D.

LEN(<azbučni izraz>). Vrednost ove funkcije je numerička, a argument azbučni. Daje broj simbola <azbučnog izraza>. Broje se i razmaci.

P r i m e r:

LEN("DAN") daje vrednost 3
LEN("DAN"+" JUTRO") daje vrednost 9

VAL (<azbučni izraz>). Vrednost ove funkcije je numerička, a argument azbučni. Vrši "prevođenje", tj. konverziju azbučnog izraza u numeričku formu. (Podatak 123, na primer, može se posmatrati kao broj sa vrednošću sto dvadeset i tri koji može učestvovati u aritmetičkim operacijama, ili kao niz od tri simbola 1, 2 i 3, koji predstavljaju azbučni podatak koji ne može učestvovati u aritmetičkim operacijama, a u računaru se interno registruje na drugačiji način od broja 123. Ovo daje smisao konverziji).

Ako prvi simbol azbučnog podatka nije +,— ili cifra, VAL daje nulu kao vrednost. Ako je prvi simbol +,— ili cifra, prevodi se azbučni podatak sve do poslednjeg simbola ili do prvog simbola koji nije numerički (osim decimalne tačke i slova E).

Primer:

VAL("123") daje kao vrednost broj 123
VAL("DAN") daje kao vrednost broj 0
VAL("15DANA") daje kao vrednost broj 15

STR$(<aritmetički izraz>). Vrednost ove funkcije je azbučna, a argument numerički. Vrši prevođenje tj. konverziju numeričkog argumenta u azbučnu formu, što predstavlja obrnutu operaciju od one koju vrši VAL.

Primer:

STR$(356) daje azbučnu vrednost 356
PRINT STR$(A*B) daje azbučnu vrednost 50
ako je, recimo, A=5 i B=10

CHR$(<aritmetički izraz>). Vrednost ove funkcije je azbučna (uvek samo jedan simbol), argument je numerički. Daje simbol koji je ASCII ekvivalent vrednosti <aritmetičkog izraza>. Broj koji predstavlja vrednost <aritmetičkog izraza> mora biti između 0 i 255.

Primer: CHR$(65) daje simbol A

LEFT$(<azbučni izraz>,<aritmetički izraz>). Vrednost ove funkcije je azbučna. Ona daje prvih <aritmetički izraz> simbola sleva iz <azbučnog izraza>.

Primer:

LEFT$("JUTRO I PODNE",7) daje JUTRO I
LEFT$("JUTRO"+"PODNE",7) daje JUTROPO

MID$(<azbučni izraz>,<aritmetički izraz—1>[,<aritmetički izraz—2>]). Vrednost ove funkcije je azbučna. Daje <aritmetički izraz—2> uzastopnih simbola <azbučnog izraza> i to počevši od simbola koji je na poziciji određenoj vrednošću <aritmetički izraz—1>.

Primer: MID$("OTAC SIN UNUK",6,3) daje SIN

RIGHT$(<azbučni izraz>,<aritmetički izraz>). Vrednost ove funkcije je azbučna. Daje poslednji <aritmetički izraz> simbola iz <azbučnog izraza>.

Primer: RIGHT$("LEVO DESNO",5) daje DESNO

Navodimo sada dva kratka programa, prvi za ispitivanje parnosti unetog celog broja, i drugi za dobijanje

vrednosti azbučnih funkcija za uneti azbučni podatak. Da bi se završio prvi program treba uneti nulu.

```
10 INPUT "UNESITE JEDAN CEO BROJ";A
20 IF A=0 THEN END
30 IF INT(A/2)=A/2 THEN PRINT "BROJ JE
   PARAN":GOTO10
40 PRINT "BROJ JE NEPARAN"
50 GOTO 10
```

Da bi se završio drugi program treba uneti KRAJ.

```
10 INPUT "UNESITE AZBUCNI PODATAK";A$
20 IF A$="KRAJ" THEN END
30 PRINT "ASC(PODATAK)=";ASC(A$)
40 PRINT "LEN(PODATAK)=";LEN(A$)
50 PRINT "VAL(PODATAK)=";VAL(A$)
60 PRINT "LEFT$(PODATAK,2)=";LEFT$(A$,2)
70 PRINT "MID$(PODATAK,2,1)=";MID$(A$,2,1)
80 PRINT "RIGHT$(PODATAK,2)=";RIGHT$(A$,2)
90 GOTO 10
```

2.17. KOMENTARI U OKVIRU PROGRAMA — REM

Da bi programi postali čitljiviji za čoveka (ne za računar), u BASIC je uvedena i naredba REM. Ova naredba omogućava da se u okviru samog programa navode i komentari.

Oblik je sledeći:

REM[<komentar>]

gde opcioni <komentar> predstavlja bilo kakav tekst. Kada računar u toku izvršavanja programa naiđe na naredbu REM, on je ne izvršava, već prelazi na sledeću naredbu. Naredba REM nije izvršna. Navodi se bilo sama u jednom programskom redu, bilo kao poslednja naredba u višenaredbenom redu.

Radi ilustracije navodimo komentarisanu verziju prethodnog programa.

```
5 REM *** ILUSTRACIJA AZBUCNIH FUNKCIJA ***
7 REM ****************************************
```

```
10 INPUT "UNESITE AZBUCNI PODATAK";A$:REM
   ULAZ
20 IF A$="KRAJ" THEN END:REM EVENTUALNI
   ZAVRSETAK
23 REM
25 REM *** IZDAVANJE VREDNOSTI FUNKCIJA ***
27 REM
30 PRINT "ASC(PODATAK)=";ASC(A$)
40 PRINT "LEN(PODATAK)=";LEN(A$)
50 PRINT "VAL(PODATAK)=";VAL(A$)
60 PRINT "LEFT$(PODATAK,2)=";LEFT$(A$,2)
70 PRINT "MID$(PODATAK,2,1)=";MID$(A$,2,1)
80 PRINT "RIGHT$(PODATAK,2)=";RIGHT$(A$,2)
90 GOTO 10:REM POVRATAK NA UNOS NOVOG
   PODATKA
```

2.18. SMEŠTANJE PROGRAMA NA SPOLJNU MEMORIJU — SAVE, I NJIHOVO UČITAVANJE — LOAD

Za čuvanje programa i podataka koristi se spoljna memorija, bilo kaseta bilo disketa, u zavisnosti od toga koji je uređaj priključen na računar. Detaljan opis kasetne spoljne memorije, kao i disketne jedinice koja koristi diskete, dat je kasnije u posebnim poglavljima knjige. Detaljno je opisan i skup komandi kojima se obezbeđuje komunikacija sa ovim uređajima.

Ovde ćemo samo u nekoliko reči opisati komande SAVE i LOAD.

Komanda za smeštanje (pamćenje) programa na kasetu ili disketu ima opšti oblik:

SAVE["<ime programa>"][,<tip uređaja>][,<tip naredbe>]

Komanda za učitavanje programa sa kasete ili diskete ima sledeći sličan oblik:

LOAD["<ime programa>"][,<tip uređaja>][,<tip naredbe>]

U oba slučaja "<ime programa>" označava neko ime koje odaberete i pod kojim se program čuva na spoljnoj memoriji.

U oba slučaja ako se za <tip uređaja> navede 8, komanda će se odnositi na disketnu jedinicu, a ako se ne navede ništa, ili se navede 1, komanda će se odnositi na kasetni magnetofon.

Poslednji opcioni <tip naredbe> nećemo za sada koristiti.

Pretpostavimo da smo neki program pomoću tastature uneli u memoriju računara, i da ga želimo smestiti na disketu pod imenom PROGRAM1. Da bismo to postigli unećemo u neposrednom režimu sledeće:

```
SAVE"PROGRAM1",8
```

Ako isti program želimo smestiti na kasetu, izdaćemo komandu:

```
SAVE"PROGRAM1"
```

Da bi taj isti program učitali u unutrašnju memoriju sa diskete, unećemo u neposrednom režimu:

```
LOAD"PROGRAM1",8
```

i program će se učitati. Ako ga želimo učitati sa kasete, učinićemo to komandom:

```
LOAD"PROGRAM1"
```

Napomenimo samo da se i komanda SAVE i komanda LOAD mogu smestiti i u program, i izvršavati u programskom režimu rada.

Za detaljnije upoznavanje periferijskih uređaja i rada sa njima, treba pogledati odgovarajuća poglavlja u knjizi.

3. DALJE MOGUĆNOSTI BASIC-a

3.1. NIZOVI, DIM

Elementi BASIC-jezika koji su obrađeni u prethodnoj glavi nisu pogodni za obradu velikog broja podataka. Podsetimo se da je broj različitih imena za promenljive ograničen. Čak i da nije tako, rad sa mnoštvom različitih imena bio bi praktično nemoguć.

Da bi se omogućila obrada velikog broja podataka koji se javljaju kao ulazni podaci ili kao rezultati obrade u BASIC-jezik je uključen još jedan matematički pojam, NIZ. Svojstvo niza je da zajedničkim imenom objedinjuje celu grupu elemenata. Svaki element ima svoj broj (indeks) pomoću koga mu se pristupa. Niz je skup numerisanih podataka koji su označeni istim imenom, a međusobno se razlikuju po indeksima.

U matematici se 20. element niza X označava sa X_{20}. Ovakav zapis na računaru ne postoji, pa se indeks niza navodi između zagrada, posle imena niza. Elementi nizova su:

X(10) NIZ(100) CENA(500)

Niz može biti jednog od tri tipa: realni, celobrojni ili azbučni. Pri tome se tip niza, kao i tip promenljive, označava specijalnim dodatnim znakom na kraju imena. Elementi iz prethodnog primera pripadaju realnim nizovima X, NIZ i CENA.

IME$(10)

je deseti član niza azbučnih podataka IME$

CELI%(20)

je dvadeseti član celobrojnog niza CELI%.

Svi podaci u nizu moraju biti istog tipa kao i niz. Elementi niza imaju iste osobine i primenjuju se na isti način kao promenljive.

Svi do sada navedeni primeri odnosili su se na obične, jednodimenzionalne nizove. U opštem slučaju niz može imati do 255 dimenzija, tj. element niza tačno se određuje sa najviše 255 indeksa. U praksi se retko koriste nizovi sa više od tri dimenzije. Nizovi sa više od jedne dimenzije nazivaju se matrice.

Prostor za čuvanje niza u memoriji rezerviše se posebnom naredbom za dimenzionisanje niza:

DIM <lista imena nizova>

<lista imena nizova> sadrži imena nizova koji se uvode, dimenziju i maksimalne vrednosti indeksa za svaki od nizova. Tako naredba:

DIM X(100),Y%(20),Z$(30)

rezerviše prostor za tri niza i to realni niz X, celobrojni niz Y% i azbučni niz Z$. Vrednosti navedene u zagradi određuju maksimalnu vrednost indeksa. Naredbom:

DIM A(10,10,50)

dimenzioniše se trodimenzionalni realni niz A.

Svaki element numeričkog niza postavlja se pri dimenzionisanju na vrednost 0. Elementi azbučnih nizova dobijaju vrednost "prazan podatak".

Indeks niza je ceo broj između 0 i 32767, pa se i dimenzije mogu kretati u tim granicama. Ukoliko indeks nije ceo broj, decimalni deo se odbacuje. Uočite da je indeks 0 takođe dozvoljen, pa prema tome dimenzionisanje niza definiše broj elemenata za jedan veći od maksimalne vrednosti indeksa navedene pri dimenzionisanju. Na primer, sa DIM X(100) rezerviše se prostor za 101 element niza X.

Za jednodimenzionalne i dvodimenzionalne nizove sa manje od deset elemenata po dimenziji naredba za dimenzionisanje se obavlja automatski od strane BASIC-interpretatora odmah nakon prve upotrebe nekog od elemenata. Pri tome se smatra da je najveći dozvoljeni indeks 10, pa će navođenje veće vrednosti kao indeksa izazvati grešku u programu.

Višestruko dimenzionisanje nije dozvoljeno, već se niz u programu može definisati samo jedanput. Zato se to najčešće vrši na početku programa.

Prema prethodnom, i sledeća upotreba niza izaziva grešku:

```
10 A(1)=100
20 DIM A(15)
```

jer je već u redu 10 izvršeno automatsko dimenzionisanje niza.

U praksi se nizovi najčešće koriste u kombinaciji sa programskim ciklusom. Na taj način obično se vrši unošenje i izdavanje niza, a u slučaju istih pravila obrade za sve članove, i obrada.

Primer: Za niz X treba uneti dimenziju i sve njegove članove. Zatim se od niza X formira drugi niz Y koji sadrži samo pozitivne elemente niza X. Prikazati sadržaj niza Y na ekranu.

```
10 INPUT "DIMENZIJA NIZA";D
20 DIM X(D),Y(D)
30 :::REM UNOSENJE NIZA
40 FOR I=0 TO D
50 PRINT "ELEMENT";I;
60 INPUT X(I)
70 IF X(I)<0 THEN 100
80 Y(J)=X(I)
90 J=J+1
100 NEXT I
105 :::REM IZDAVANJE NIZA
108 IF J=0 THEN PRINT "NIZ Y JE PRAZAN": STOP
110 FOR I=0 TO J—1
120 PRINT "Y(";I;")=";Y(I)
130 NEXT I
```

Kako pokazuje linija 20, u dimenzionisanju se kao maksimalna vrednost indeksa može pojaviti promenljiva, pa i aritmetički izraz. U redu 60 učitava se element niza, u redu 80 vrši formiranje niza Y, a redovi od 110 do 130 sadrže naredbe za prikazivanje niza Y.

3.2. PROGRAMSKA DATOTEKA, DATA, READ, RESTORE

Unošenje podataka u program obično se vrši naredbom INPUT. Međutim, u slučajevima u kojima se zahteva često korišćenje istih podataka, oni se mogu zadati i u samom programu, u okviru nekih programskih redova. Ovakav skup podataka naziva se **programska datoteka** i za njeno korišćenje služe sledeće BASIC naredbe: DATA, READ i RESTORE.

Naredba DATA ima oblik:

DATA <lista konstanti>

Ovom naredbom vrši se definisanje podataka u datoteci. Podaci se navode u obliku brojnih ili azbučnih konstanti, međusobno razdvojenih zarezom. U jednom programu može se nalaziti proizvoljan broj DATA naredbi i mesto njihovog navođenja u programu nije od značaja (obično se navode na početku ili na kraju programa).

Na primer, u redu 10:

10 DATA 10,MAJ,1985

navedena su dva brojna i jedan azbučni podatak.

Umesto naredbom INPUT, podaci iz programske datoteke čitaju se naredbom READ:

READ <lista imena promenljivih>

Promenljiva i njoj odgovarajući podatak moraju se slagati po tipu. U suprotnom se izdaje izveštaj o grešci BAD DATA. U slučaju da se pokuša čitanje više podataka nego što je navedeno u datoteci, dobija se izveštaj OUT OF DATA.

Prethodno navedena DATA naredba mogla bi se čitati:

READ D,M$,G

Čitanje datoteke počinje od prvog podatka navedenog u DATA naredbi sa najmanjim brojem reda. U programu se automatski definiše pokazivač koji ukazuje na sledeći element za čitanje, a njegova vrednost se uvećava za svaki pročitani podatak. Povratak unazad nije moguć, izuzev u slučaju da se počinje čitanje cele datoteke od početka. Naredbom:

RESTORE

se tada pokazivač vraća na prvi element datoteke.

Primer: Izračunavanje srednje godišnje temperature nekog mesta može se vršiti sledećim programom:

```
10 DATA JANUAR, FEBRUAR, MART, APRIL, MAJ
20 DATA JUN, JUL, AVGUST, SEPTEMBAR
30 DATA OKTOBAR,NOVEMBAR,DECEMBAR
40 :
50 INPUT "MESTO:";M$
60 PRINT "MESEC","TEMPERATURA"
70 S=0
80 FOR I=1 TO 12
90 READ X$
100 PRINT X$," ";
110 INPUT T
120 S=S+T
130 NEXT I
140 :
150 PRINT "SREDNJA GODISNJA TEMPERATURA"
160 PRINT "U MESTU ";M$;" JE ";S/12;" STEPENI"
170 RESTORE
180 GOTO 50
```

Redovi 10 do 30 sadrže podatke koji definišu programsku datoteku. Nakon unošenja imena mesta (red 50) vrši se unošenje srednje temperature za svaki od meseca. Naziv meseca uzima se iz programske datoteke (red 90). Nakon završetka unošenja vrednosti za svih 12 meseci, izdaje se završni izveštaj, obnavlja programska datoteka (red 170) i ponavlja postupak za neko drugo mesto.

3.3. KORISNIČKI DEFINISANE FUNKCIJE, DEF FN, FN

Pored standardnih funkcija prisutnih u BASIC-u često se pojavljuje potreba za novim, korisničkim funkcijama. Tu mogućnost pruža naredba za definisanje korisničke funkcije i nakon njene upotrebe uvedena funkcija je potpuno ravnopravna sa ostalim.

Proizvoljan aritmetički izraz može se smatrati funkcijom nekog argumenta i može mu se dodeliti ime. Ime

ovako uvedene funkcije može se koristiti kao argument nekog drugog aritmetičkog izraza.

Oblik naredbe za definisanje korisničke funkcije je:

DEF FN<ime funkcije>(<ime promenljive>)=<aritmetički izraz)

<ime funkcije> određuje puno ime korisničke funkcije.

Format imena je istovetan kao i format imena promenljive — značajna su dva simbola, prvi mora biti slovo, drugi slovo ili cifra. Međutim, ime se ne može završavati sa % ili $, tj. može se definisati samo funkcija čija će vrednost biti realan broj.

<ime promenljive> predstavlja fiktivni argument funkcije.

Svako njegovo pojavljivanje u aritmetičkom izrazu na desnoj strani znaka jednakosti u pozivu funkcije će se automatski zameniti vrednošću navedenom u pozivu. Ovaj element mora biti naveden.

<aritmetički izraz> je uobičajenog tipa, i u sebi može sadržati i imena drugih korisničkih funkcija.

Korisnički definisane funkcije navode se na početku programa.

Na primer, naredbom:

DEF FNA5(Z)=Z ↑ 2+10*Z+6

definisana je korisnička funkcija sa imenom A5 i fiktivnom promenljivom označenom sa Z. Vrednost funkcije predstavlja vrednost navedenog polinoma za neku zadatu vrednost Z.

Poziv korisnički definisane funkcije vrši se navođenjem njenog imena i stvarnog argumenta u sklopu aritmetičkog izraza. Svako pojavljivanje fiktivnog argumenta u izrazu kojim se funkcija definiše zamenjuje se vrednošću stvarnog argumenta i izračunata vrednost dodeljuje kao vrednost uvedene funkcije.

Ranije definisana funkcija A5 može se pozvati na sledeći način:

X=SQR(FNA5(10))+1

Naredba DEF FN može se koristiti samo u programskom režimu, dok se poziv definisane funkcije može vršiti i u neposrednom režimu.

P r i m e r: Definisanje korisničke funkcije koja će imati vrednost funkcije sekans:

SEC(X)=1/COS(X)

može se izvršiti sledećim programom:

```
10 DEF FNSC (A)=1/COS(A)
20 PRINT "X","SEC(X)"
30 INPUT X
40 PRINT X,FNSC(X)
50 GOTO 30
```

3.4. POTPROGRAMI, GOSUB, RETURN

Kao što je moguće definisati novu funkciju i koristiti je u okviru aritmetičkog izraza, tako je moguće izdvojiti neke linije programa u potprogramski segment, ili **potprogram** i zahtevati njihovo izvršavanje iz bilo kog dela programa.

Poziv potprograma vrši se naredbom:

GOSUB <broj reda>

Efekat naredbe je sličan naredbi GOTO po tome što se vrši skok na red sa navedenim brojem, ali je razlika u tome što se (na posebnom mestu u memoriji) pamti mesto u programu sa koga je skok izvršen. Time se omogućuje da se na izlazu iz potprograma naredbom:

RETURN

nastavi rad programa od onog mesta sa koga je poziv došao.

Neka, na primer, program treba da odredi zbir kvadratnih korena dva cela, pozitivna broja uneta sa ulaza.

Program tada može biti:

```
10 INPUT X
20 IF X<0 THEN PRINT "POGRESNO":STOP
30 IF X—INT(X)<>0 THEN PRINT "POGRESNO":STOP
40 INPUT Y
50 IF Y<0 THEN PRINT "POGRESNO":STOP
60 IF Y—INT(Y)<>0 THEN PRINT "POGRESNO":STOP
```

```
 70 PRINT SQR(X)+SQR(Y)
 80 STOP
```

Upotrebom naredba GOSUB program dobija pregledniji oblik:

```
 10 INPUT X
 20 A=X : GOSUB 100
 30 INPUT Y
 40 A=Y : GOSUB 100
 50 PRINT SQR(X)+SQR(Y)
 60 STOP
 70 :
100 IF A<0 THEN PRINT "POGRESNO":STOP
110 IF A—INT(A)<>0 THEN PRINT "POGRESNO":STOP
120 RETURN
```

Potprogram nema nikakvo posebno ime, već predstavlja logičku celinu koju je zamislio sam programer. Iz jednog potprograma može postojati više izlaza naredbom RETURN i sve one imaju isti efekat. Može postojati i više ulaza u potprogram, što je takođe stvar programera.

Za potprogram ne postoje posebne ulazne i izlazne veličine, već su u njemu prisutne i mogu se koristiti sve promenljive iz programa. Sve one promenljive koje se definišu unutar potprograma prisutne su po povratku i u programu.

Potprogrami se mogu pozivati jedan iz drugog, po dubini, a moguće je i "pozivanje samog sebe" (tzv. rekurzivni poziv). Broj ovakvih poziva po dubini ograničen je veličinom memorije u kojoj se čuvaju adrese povratka za svaki poziv potprograma. Koliki je tačno ovaj broj može se dobiti izvršavanjem ovog kratkog programa:

```
 10 I=0
 20 GOSUB 100
100 I=I+1
110 PRINT "ZA ";I;"POZIVA PO DUBINI RADI"
120 GOSUB100
```

Poslednji izveštaj pre prekida programa zbog nedostatka memorije daje najveći broj poziva po dubini.

3.5. VIŠESTRUKO GRANANJE, ON

Često se u programu pojavljuje potreba za skokom na red određen trenutnom vrednošću neke promenljive. Problem se može rešiti koristeći nekoliko uzastopnih IF naredbi, za svaku od vrednosti po jednu. Daleko elegantnije rešenje nudi naredba ON:

ON <aritmetički izraz> GOTO <lista brojeva redova>

Za izračunatu vrednost aritmetičkog izraza određuje se celi deo i vrši skok na ono obeležje čiji je to redni broj u listi. Na primer, naredbom:

ON A GOTO 100,300,500

vrši se skok na 100 ako je A=1, na 300 ako je A=2 ili na 500 ako je A=3.

Ukoliko je vrednost izraza 0 ili za nju ne postoji odgovarajući član liste (veća je od broja elemenata liste) prelazi se na sledeću navedenu naredbu posle ON. Ukoliko je vrednost izraza negativna, javlja se izveštaj o grešci.

Sledeći kratak primer ilustruje primenu naredbe ON ... GOTO za konverziju unete brojčane ocene u njen opis.

```
 10 INPUT A : REM UNOSENJE OCENE
 20 ON A GOTO 100,110,120,130,140
 30 PRINT "UNETA OCENA NE POSTOJI": GOTO 10
100 PRINT "NEDOVOLJAN":GOTO 10
110 PRINT "DOVOLJAN":GOTO 10
120 PRINT "DOBAR":GOTO 10
130 PRINT "VRLO DOBAR":GOTO 10
140 PRINT "ODLICAN":GOTO 10
```

Drugi oblik naredbe ON je prelaz na potprogram u zavisnosti od vrednosti aritmetičkog izraza. Oblik naredbe je:

ON <aritmetički izraz> GOSUB <lista brojeva redova>

Mehanizam prelaza je isti kao i za prethodni oblik, ali se umesto skoka vrši poziv potprograma sa navedenog obeležja. Izlazom iz potprograma sa RETURN izvršavanje se nastavlja od prve sledeće naredbe posle ON.

3.6. DODATNE MOGUĆNOSTI PRI UNOŠENJU I IZDAVANJU PODATAKA

3.6.1. Naredba GET

Pri unošenju podataka naredbom INPUT program se zaustavi i čeka na kompletiranje ulaznog podatka. Unošenje jednog znaka bez zaustavljanja programa postiže se naredbom GET. Oblik naredbe je:

GET <lista imena promenljivih>

U memoriji računara postoji poseban deo, takozvani bafer tastature, u koji se upisuje simbol koji odgovara pritisnutom tasteru. Naredba INPUT čeka da se u bafer unese podatak, uzima ga, prikazuje na ekranu i formira vrednosti za promenljive navedene u listi. Kada se pritisne taster za kraj reda, njegov kôd dolazi, kao i kod svakog drugog simbola, u bafer tastature. Čitanjem ovog kôda u naredbi INPUT njeno se izvršavanje završava i program nastavlja rad.

Izvršavanje naredbe GET ne sadrži sve ove korake, već samo onaj u kome se čita bafer tastature. Čitanje se vrši bez čekanja i samo jedanput za svaku promenljivu navedenu u listi. Odmah nakon toga program nastavlja rad. Zato se kao pročitane vrednosti mogu pojaviti:

1) "Prazan znak", ako je bafer bio prazan. Njegova vrednost je 0, a odgovarajući azbučni podatak "".
2) ASCII kôd unetog znaka, ako bafer nije bio prazan. Pročitani znak se ne prikazuje na ekranu.

Svakoj promenljivoj u listi može se dodeliti samo po jedan znak. Kako je veličina bafera 10 znakova, navođenje više od 10 promenljivih u listi nema smisla. Ukoliko je promenljiva brojnog tipa, a pročitani simbol nije cifra, javlja se izveštaj o grešci. Zato se obično u listi navode azbučne promenljive, a zatim vrši njihovo pretvaranje u brojne vrednosti.

Prethodno objašnjenje demonstriraćemo jednostavnim primerom:

```
10 GET A$
20 PRINT ASC(A$+CHR$(0)),A$
30 GOTO 10
```

Na ekranu će se nakon startovanja pojaviti niz nula. One će se ponavljati sve dok se ne pritisne neki taster, kada se dobija njegov ASCII kôd i simbol koji mu odgovara (ukoliko je vidljiv). Zatim se nule dalje ponavljaju, sve do narednog unetog znaka. Isprobajte rad ovog programa za promenljivu numeričkog tipa, unoseći najpre cifre, a zatim neki drugi znak.

P r i m e r 1: Činjenica da se uneti znak ne prikazuje na ekranu i da nema čekanja često se koristi za realizovanje pauze u programu sve do unošenja bilo kog znaka. Na primer, programski red 20 u sledećem programu:

```
10 PRINT "PROGRAM CE NASTAVITI RAD"
15 PRINT "KADA SE PRITISNE BILO KOJI TASTER"
20 GET A$ : IF A$="" THEN 20
30 PRINT "PROGRAM JE NASTAVIO RAD"
```

ne dopušta nastavak programa sve dok se ne pritisne neki taster.

P r i m e r 2: Na osnovu primera 1 može se napraviti programski segment, koji očekuje unošenje zadate šifre, proverava je i odlučuje o daljem toku programa.

```
10 SFRA$="BASIC"
15 ULAZ$=""
20 GET A$ : IF A$="" THEN 20
30 ULAZ$=ULAZ$+A$
40 IF LEN(ULAZ$)<LEN(SFRA$) THEN 20
50 IF ULAZ$<>SIFRA$ THEN PRINT "NEISPRAVNO":
   GOTO 15
60 PRINT "ISPRAVNA SIFRA"
70 PRINT "DOZVOLJAVA SE DALJI RAD PROGRAMA"
```

Program čeka da se unese onoliko simbola koliko ih ima u šifri (redovi 30 i 40), a nakon toga proverava njenu ispravnost (red 50). Šifra se može menjati promenom azbučnog podatka u redu 10.

P r i m e r 3: Iz opisa naredbi INPUT i GET sledi da se INPUT može programski realizovati pomoću GET. Sledeći primer ima tu funkciju:

```
 5 PRINT "?";
10 ULAZ $=""
20 GET A$:IF A$="" THEN 20
30 PRINT A$;
40 IF A$=CHR$(13) THEN 70
50 ULAZ $=ULAZ$+A$
60 GOTO 20
70 PRINT "ULAZNI PODATAK JE";ULAZ$
```

Program čeka na unošenje simbola sa tastature (red 20), prikazuje simbol na ekranu (red 30), i ukoliko je to bio kraj reda (CHR$(13)), završava sa unošenjem (red 40). Ukoliko je to bio neki drugi simbol, nastavlja se formiranje ulaznog podatka (redovi 50 i 60).

3.6.2. **Dodatne mogućnosti naredbe PRINT**

Funkcije SPC i TAB. Formiranje izlaznog izveštaja u naredbi PRINT može se, osim poznatim simbolima "," i ";", vršiti i posebnim funkcijama SPC i TAB.

Funkcija SPC je oblika:

SPC(<aritmetički izraz>)

i vrši izdavanje onoliko blanko simbola kolika je vrednost aritmetičkog izraza. Sledeće naredbe zato imaju isto dejstvo:

PRINT "A";SPC(10);"B" i PRINT "A B"

Funkcija TAB ima sličan format:

TAB(<aritmetički izraz>)

ali joj je značenje nešto drugačije. Vrednost aritmetičkog izraza određuje poziciju mesta u tekućem redu od koga će početi izdavanje sledećeg izveštaja. Ukoliko je trenutna pozicija kursora veća od pozicije koju zahteva funkcija TAB, ništa se neće promeniti. "Povratak unazad" nije moguć i izdavanje će se izvršiti u sledećem slobodnom polju.

Naredba:

PRINT "MESTO";TAB(20);"20"

izdaje 20 počev od mesta 20 u tekućem redu, dok:

```
PRINT " MESTO";TAB(1);"1"
```

daje na ekranu MESTO1, a ne 1MESTO, kako bi se to očekivalo.

Funkcija POS. Funkcija POS daje poziciju kursora u odnosu na početak tekuće linije. To je prava numerička funkcija, mada se koristi isključivo u postupku formatiranja izlaznog izveštaja. Njen oblik je:

```
POS(<ime promenljive>)
```

pri čemu navedeno ime promenljive nema značaja.

Vrednost ove funkcije kreće se između 0 i 79, jer se smatra da je 80 najveća dužina linije. U slučajevima kada je stvarna dužina veća, vrednost se dobija umanjenjem dobijenog rezultata za 80.

Tako se sa:

```
FOR I=1 TO 50 :PRINT "*"; : NEXT : PRINT POS(A)
```

dobija rezultat 50. Isti rezultat se dobija i za:

```
FOR I=1 TO 130 :PRINT "*"; : NEXT : PRINT POS(A)
```

iako je stvarna dužina za 80 veća.

P r i m e r: Funkcija POS se često koristi za smanjenje dužine linije na ekranu sa 40 na neku manju vrednost. Na primer, sledeći program pri izdavanju naredbom PRINT koristi samo 20 mesta u redu:

```
10 FOR I=1 TO 200
20 PRINT "*";
30 IF POS(A)>20 THEN PRINT
40 NEXT I
```

Posle svakog simbola izdatog naredbom PRINT u redu 20 ispituje se pozicija kursora. Ako je prešla 20, prelazi se u novi red.

3.6.3. **Rad u režimu navodnika**

Unošenjem znaka navoda (") sa tastature ulazi se u takozvani "režim navodnika". Specifičnost ovog oblika rada je da sve funkcije tastature vezane za prikazivanje teksta

(navedene u uvodu, glava 1), kao i sve funkcije ekranskog editora gube izvršni karakter. To znači da umesto da se ostvari dejstvo funkcije, na ekranu se pojavljuje specijalni simbol kojim se ta funkcija označava. Izlaz iz režima navodnika vrši se unošenjem novog znaka navoda ili kra-jem reda.

Mada deluje zbunjujuće, zbog neobičnih znakova koji se dobijaju, režim navodnika je veoma značajan. On omo-gućuje da se u okviru azbučne konstante nađu i komande tastature, pa se tako one mogu koristiti i iz programa. Na primer, postavljanje kursora na neko mesto na ekranu može se izvršiti jedino na ovaj način. Pored toga, moguća su i brisanja teksta na ekranu iz programa, promena boje slova i slično.

Teškoće pri radu javljaju se onda kada treba prepo-znati funkciju znaka navedenog među navodnicima. Radi toga ćemo se u ovoj knjizi poslužiti posebnim zapisom, u obliku:

{<tekst>}

pri čemu će navedeni tekst opisivati šta treba da se uradi. Na primer:

PRINT"P {dole} R {dole} IM { gore } E {gore} R"

označava da posle P i R treba uneti strelicu nadole, a posle M i E strelicu nagore.

Koristiće se sledeći zapisi:

Pomeranje kursora: { **levo** }, {desno}, {gore}, {dole }
Inverzni prikaz: {rvs on}, { rvs off}
Promena boje: {ctrl/1 }, { ctrl/2} , . . . { ctrl/8}
{comm/1 }, { comm/2} , . . . { comm/8}
Brisanje ekrana sa CLR: { clr }
Premeštanje kursora u gornji ugao sa HOME: {home}
Brisanje poslednjeg simbola sa DEL: {del}
Ubacivanje praznog mesta sa INST: { inst}
Unošenje blanko-znaka sa SPACE: { prazno }

Grafički simboli zapisivaće se u sličnoj formi, kao kombinacija specijalnih znakova SHIFT i COMMODORE i slova na kojima se nalaze. Na primer:

{ shift/j } ili {comm/r}

Kada bude potrebno koristiće se i nešto slobodniji opisi:

Deset puta levo: { 10 levo }
Osam grafičkih simbola: { 8 shift/d}

Sledećih nekoliko jednostavnih primera prikazaće mo guće načine primene:

Primer 1:

```
10 PRINT "OVAJ TEKST CE BITI OBRISAN";
20 FOR I=1 TO 26
30 PRINT "{INST}{DEL}";
40 FOR J=1 TO 700:NEXT J
50 NEXT I
60 PRINT
```

Izdavanje zahteva za brisanje poslednjeg simbola (red 30) vrši se 26 puta, kolika je i dužina izdate rečenice (redovi 20 do 50). Proces se usporava radi boljeg uočavanja efekta (red 40).

Primer 2: Deo teksta može biti prikazan inverznim simbolima:

```
PRINT "INVERZNI { rvs on } TEKST { rvs off }"
```

Primer 3:

```
10 PRINT "GRAD";
20 FOR I=1 TO 700: NEXT I
30 PRINT "{ 4 levo }{3 inst}";
40 FOR I=1 TO 700: NEXT I
50 PRINT "BEO"
```

Na početku reči GRAD oslobađaju se tri prazna mesta i upisuje nova reč BEO.

3.7. DIREKTAN UPIS I ČITANJE MEMORIJE

3.7.1. **Raspored ROM-a i RAM-a**

Pre nego što pređemo na naredbe koje se odnose na direktan rad sa memorijom, razmotrimo najpre kako je memorija organizovana.

Podaci o veličini memorije kod čitaoca mogu dovesti do zabune, jer su na prvi pogled kontradiktorni. Karakteristična mesta su sledeća:

— Uključivanjem se u vrhu ekrana dobija podatak o 38 kB RAM za BASIC. Računar je dobio ime po 64 kB RAM. Šta se dogodilo sa preostalih 24 kB?

— Među karakteristikama računara nalazi se podatak o 64 kB RAM i 20 kB ROM. Mikroprocesor 6510 ugrađen u COMMODORE 64 ima adresni prostor od 64 kB. Kako onda komunicira sa ukupno 84 kB memorije?

Odgovor na ova pitanja nalazi se u posebnom načinu organizacije memorije. U računaru zaista ima 64 kB RAM i 20 kB ROM i u ROM-u se nalaze:

1 — BASIC-interpretator (8 kB),
2 — KERNAL operativni sistem (8 kB),
3 — karakter-generator (4 kB).

O BASIC-interpretatoru već je bilo govora. KERNAL operativni sistem zadužen je za komunikaciju sa spoljašnjim uređajima i kontrolu rada računara. U karakter-generatoru nalaze se opisi svih simbola iz obe azbuke.

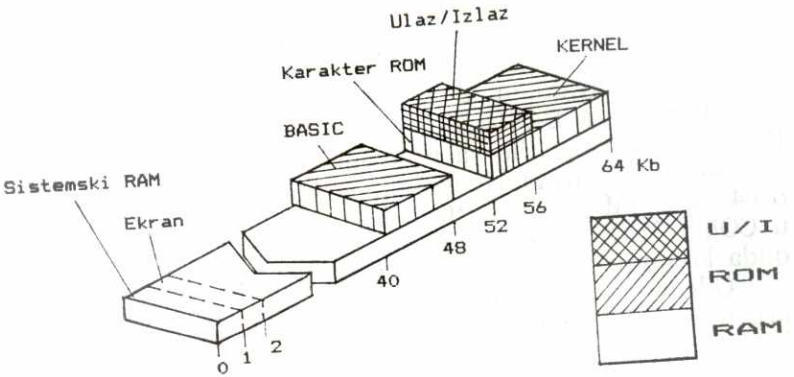
Kao poseban memorijski blok izdvaja se i prostor za komunikaciju mikroprocesora i generatora slike i zvuka. Ovaj prostor je ukupne veličine 4 kB, pa ukupni memorijski zahtevi narastaju na 88 kB memorijskih adresa.

Ograničenje na 64 kB koje nameće adresni prostor je nesavladivo: računar može da pristupa samo tolikom broju adresa. Međutim, ti se pristupi ne događaju istovremeno, pa se u različitim trenucima na nekim memorijskim adresama može naći različit sadržaj. Upravo na ovome se zasniva i ideja organizacije memorije kod COMMODORE 64: Neke adresne zone imaju višestruku namenu i ona zavisi od stanja jednog upravljačkog registra. Raspodela ovih zona data je na sl. 3.1.

Uočljivo je da se RAM nalazi najniže, odnosno da je sakriven drugim delovima. Ako izračunate ukupnu veličinu vidljivog RAM-a, dobićete 44 kB. Od toga prva dva kilobajta koriste programi iz ROM-a, dok je deo posle BASIC ROM-a (od 4 kB) namenjen mašinskim programima. Tako se dobija spomenutih 38 kB za korišćenje iz BASIC-a.

Gornji elementi sa slike prisutni su pri uključenju računara. Ovaj se raspored može menjati, zavisno od kon-

5*



Sl. 3.1 — Organizacija memorije

kretne primene, promenom sadržaja na adresi 1. Za programiranje na mašinskom jeziku, BASIC-interpretator je suvišan, pa se obično isključuje i na to mesto dolazi RAM. Ovo se može postići postavljanjem bita 0 na adresi 0 na 0.

Kako je ova knjiga posvećena programiranju na BASIC-u, za šta su potrebni i interpretator i operativni sistem, njihovo mesto se neće menjati. U odeljku o definisanju novih karaktera detaljno će se opisati postupak pristupanja skrivenom karakter-generatoru i njegovo korišćenje.

3.7.2. **Korišćenje logičkih operacija**

U direktnom radu sa memorijom logičke operacije AND, OR i NOT dobijaju nešto drugačije značenje. Do sada smo smatrali da se njima određuje logička vrednost TAČNO ili NETAČNO. Postoji i drugačija interpretacija.

Jedan registar sadrži 8 bita. Sadržaj svakog od njih može biti 1 ili 0. Logičkom operacijom između vrednosti dva registra dobija se takođe rezultat na 8 bita, pri čemu je svaki bit rezultata dobijen primenom te logičke operacije nad odgovarajućim mestima argumenata. Na primer:

01010101 OR 10101010 daje 11111111
01010101 AND 10101010 daje 00000000
NOT 01010101 daje 10101010

Imajući ovo u vidu, ima smisla govoriti o BROJNOJ vrednosti rezultata logičke operacije. U prethodnim primerima rezultati se tada mogu shvatiti kao dekadni brojevi 255,0 i 170.

U praksi se ovo koristi u sledećim slučajevima:

1) OR postavlja vrednost nekog mesta prvog argumenta na 1, bez obzira na prethodni sadržaj, ukoliko je na odgovarajućem mestu drugog argumenta bila jedinica.

REG OR 192

postavlja mesta 6 i 7 u registru REG na 1, ne menjajući preostali deo registra.

2) AND postavlja vrednost nekih mesta prvog argumenta na 0, ukoliko je drugi argument na tim mestima imao 0, a na ostalima 1.

REG AND 192

postavlja nule na prvih šest mesta u registru REG, ne menjajući mesta 6 i 7.

Navedeni postupci čine osnovu pri programiranju grafike i zvuka, gde će se bolje upoznati njihova primena.

3.7.3. Naredba POKE

Funkcija naredbe POKE je direktan upis sadržaja na neku memorijsku adresu, bez obzira šta se tu ranije nalazilo. Oblik naredbe je:

POKE <aritmetički izraz>,<aritmetički izraz>

Vrednost prvog navedenog aritmetičkog izraza određuje memorijsku adresu na koju će biti upisana vrednost drugog aritmetičkog izraza. Kako se u jedan registar može upisati samo broj između 0 i 255, to se vrednost drugog aritmetičkog izraza najpre pretvara u ceo broj, a zatim se proverava da li pripada navedenom intervalu. U slučaju da uzima neku vrednost van ovog intervala, javlja se izveštaj o grešci (ILLEGAL QUANTITY ERROR).

Greška se javlja i u slučaju da vrednost koja određuje adresu izađe iz intervala 0 do 65535.

U kombinaciji sa funkcijom PEEK, ova naredba se vrlo često koristi pri radu sa mašinskim programima, radu sa grafikom ili zvukom ili čuvanju podataka u memoriji. U

narednim glavama ove knjige ovo će biti jedna od najčešće korišćenih naredbi. Zato ćemo ovde navesti samo nekoliko jednostavnih primera.

Primer 1: Promena boje slova kojima se piše dobija se postavljanjem registra 646.

```
POKE 646,1
```

određuje belu boju za pisanje slova.

Primer 2: Rad tastature u REPEAT modu (simbol se ponavlja dok se taster drži pritisnut) uključuje se sa:

```
POKE 650,128
```

a isključuje sa:

```
POKE 650,0
```

3.7.4. **Funkcija PEEK**

Čitanje sadržaja memorijskog registra postiže se navođenjem njegove adrese kao argumenta numeričke funkcije PEEK. Oblik funkcije je:

```
PEEK(<aritmetički izraz>)
```

Vrednost funkcije je ceo broj između 0 i 255, zapisan na navedenoj adresi.

Vrednost aritmetičkog izraza mora biti između 0 i 65535. U protivnom se javlja izveštaj o grešci.

Primer ove funkcije je:

```
PRINT PEEK(53280)
```

čime se dobija kôd boje za okvir ekrana.

3.7.5. **Naredba WAIT**

Ovom naredbom omogućeno je da izvršavanje BASIC-programa bude uslovljeno nekim spoljašnjim događajima. Ima ulogu programskog "semafora" i njome se može potpuno zaustaviti rad programa sve dok se ne pojavi signal, koji označava da se u hardverskom delu računara nešto dogodilo. Ovaj signal može doći na primer od tastature, nekog spoljašnjeg uređaja ili od časovnika ugrađenog

u računar. Tako je ostvaren jedan oblik komunikacije BASIC-programa i pojedinih delova hardvera.

Oblik naredbe je:

WAIT <adresa>,<maska>[,<nivo za čekanje>]

<adresa> može biti bilo koja memorijska adresa, ali ima smisla koristiti samo one koje su tesno vezane za hardver i čije se vrednosti menjaju nezavisno od toka BASIC-programa. Neke od takvih adresa su:

162 uvećava se za 1 svakih 1/60 sekundi
161 uvećava se za 1 svakih 4.2 sekunde
160 uvećava se za 1 svakih 18.2 minuta
198 broj simbola u baferu tastature (0 do 9) (uvećava se pritiskom bilo kog tastera)
197 kod pritisnutog tastera (vrednost je 64 ako nijedan nije pritisnut)

<maska> određuje koja se binarna mesta na navedenoj adresi smatraju značajnim za određivanje rezultata. Na primer, vrednost 255 uključuje svih osam mesta, vrednost 5 određuje nulto i drugo mesto, dok vrednost 0 nema smisla jer definiše "večno čekanje".

<nivo za čekanje> zadaje vrednost svakog od značajnih mesta za koju će program čekati (da li za 0 ili za 1). Ukoliko se ne navede, smatra se da je 0.

Dejstvo naredbe je sledeće: Testira se vrednost svih binarnih mesta sa navedene adrese za koja su odgovarajuća binarna mesta u masci postavljena na 1. Ostala mesta nemaju značaja. Testiranje se, zavisno od zadatog oblika naredbe, može vršiti na dva načina:

1) Ako <nivo za čekanje> nije naveden.

Rezultat testa je "nastavi" ako je bar jedno od označenih mesta postavljeno na 1. Ukoliko su sva 0, rezultat je "čekaj".

Matematički gledano, rezultat testa se može predstaviti kao rezultat logičke operacije AND, izvršene nad sadržajem navedene adrese i sadržajem maske. Ako je rezultat 0 program stoji, dok za svaku drugu vrednost nastavlja rad.

Smisao naredbe je, dakle, "čekaj sve dok se na nekom od maskom označenih mesta na zadatoj adresi ne pojavi 1"

Ekvivalent ovog oblika naredbe WAIT u BASIC-jeziku bi bio:

<broj reda> IF PEEK(<adresa>)AND<maska>=0 then <broj reda>

2) Ako je naveden <nivo za čekanje>.

Sadržaj navedene adrese se najpre transformiše, tako da se sve nule u njemu zamene jedinicama, a sve jedinice nulama (ovo je logička XOR operacija). Sa dobijenim sadržajem tada se radi kao i u slučaju 1.

Smisao ovog oblika naredbe je: "Čekaj sve dok se na nekom od maskom označenih mesta na zadatoj adresi ne pojavi vrednost suprotna od one koju za to mesto predviđa <nivo za čekanje>". Dakle, ako se samo jedna od njih promeni, program se nastavlja.

BASIC-ekvivalent ovog oblika bio bi, na primer:

```
10 IF ((PEEK(A) AND NOT(N)) OR (NOT(PEEK
(A) AND N)) AND M=0 THEN 10
```

Najčešća primena ove naredbe vezana je za tastaturu.

P r i m e r 1: Može se čekati na unošenje bilo kojeg simbola:

```
10 PRINT "PRIMER1"
20 WAIT 197,64,64 : REM CEKA SE NA SIMBOL
30 PRINT "PROGRAM JE NASTAVLJEN"
40 POKE 198,0: PONISTAVANJE UNETOG SIMBOLA
```

Sve dok se na mestu 6 na adresi 197, nalazi jedinica, znači da nije pritisnut ni jedan taster. Maska ukazuje na samo jedno mesto, i to baš mesto 6. Nivo za čekanje je na tom mestu postavljen na jedan, pa će se program nastaviti tek kada se tu pojavi 0, tj. kada se neki taster pritisne.

P r i m e r 2: Isti efekat može se postići i sledećim programom:

```
10 PRINT "PRIMER2"
20 WAIT 198,1
30 PRINT "PRITISNUT JE TASTER"
```

Broj simbola u baferu je 0 sve dok se neki taster ne pritisne. Broj jedan se tada upisuje u 197, tj. postavlja se bit 0 na 1 i čekanje prestaje.

P r i m e r 3: Moguće je čekati da se otpusti pritisnuti taster. Za startovanje ovog primera otkucajte RUN i <kraj reda>, ali tako da <kraj reda> zadržite pritisnut.

```
10 PRINT "PRIMER3"
20 WAIT 197,64: REM CEKANJE DA SE TASTER
   OTPUSTI
30 PRINT "TASTATURA JE NEAKTIVNA"
```

Na mestu 6 postavlja se 1 samo kada ništa na tastaturi nije pritisnuto. Kada se to dogodi, čekanje prestaje.

3.7.6. Naredba CLR i funkcija FRE

Naredba CLR i funkcija FRE nisu vezane za direktan upis i čitanje memorije, ali su vezane za njeno posredno korišćenje (pri radu u BASIC-u o raspodeli memorije vodi računa interpretator). Opisaćemo ukratko njihovu funkciju.

Upotrebom komande NEW brišu se sve zone u memoriji, koje su bile dodeljene korisniku. Interpretator se dovodi u početno stanje. Delimično brisanje memorije postiže se naredbom:

CLR

Njenom upotrebom biće izbrisane: promenljive, nizovi, veza sa tekama podataka na periferijskim uređajima i korisnički definisane funkcije. Interpretator se dovodi u početno stanje, a program ostaje nepromenjen. U slučajevima kada se program može podeliti u dva dela u kojima se koriste uzajamno nezavisne promenljive, između delova je dobro koristiti naredbu CLR. Time se maksimalno povećava prostor za čuvanje podataka u sledećem delu.

Funkcija FRE ima oblik:

FRE(<ime promenljive>)

pri čemu navedeno ime nema značaja.

Funkcija je numeričkog tipa i daje veličinu slobodne memorije za upotrebu u BASIC-programu. Kao rezultat se može dobiti i negativan broj, što je posledica načina prikazivanja celih brojeva na ekranu. Kako je raspon celih

brojeva između —32768 i 32767, a veličina slobodne memorije može preći 32787, pri izdavanju će se veličina smatrati negativnim brojem i tako i prikazati. Prava veličina dobija se u tom slučaju oduzimanjem apsolutne vrednosti dobijenog rezultata od 65536.

3.8. KORIŠĆENJE MAŠINSKIH POTPROGRAMA IZ BASIC-a

Potreba za direktnim programiranjem mikroprocesora ugrađenog u računar javlja se onda kada je brzina izvršavanja programa na BASIC-u nedovoljna ili kada se žele proširiti standardne mogućnosti računara. U oba slučaja od korisnika se zahteva poznavanje mašinskog jezika, pa ćemo se zadržati samo na opštem prikazu postojećih mogućnosti.

Korišćenje mašinskih potprograma moguće je na dva nivoa: nivou naredbe i nivou funkcije. Potprogram će se koristiti kao funkcija onda kada se njime vrši obrada neke zadate veličine, obično realnog broja, i rezultat te obrade vraća nazad u program. Kao naredba, potprogram se koristi onda kada ima izvršni smisao, tj. proizvodi neku akciju na računaru.

3.8.1. **Naredba SYS**

Izvršavanjem naredbe SYS mikroprocesor se dodeljuje mašinskom programu koji počinje na navedenoj adresi. Oblik naredbe je:

SYS <memorijska lokacija>

pri čemu <memorijska lokacija> može biti i vrednost aritmetičkog izraza.

Način rada ove naredbe sličan je naredbi GOSUB. Mesto u programu sa koga je poziv upućen čuva se na poseban način u memoriji računara i na kraju rada mašinskog potprograma, izvršavanjem mašinske instrukcije za povratak iz potprograma (RTS), kontrolu nad procesorom ponovo preuzima BASIC-interpretator. BASIC-program

se tada nastavlja izvršavanjem naredbe koja sledi posle SYS.

Na ovaj način mogu se pozivati novi korisnički programi smešteni u RAM, ili već postojeći sistemski programi iz ROM-a. U slučaju da se pojavi potreba za prenosom argumenata, on se obavlja posredstvom neke zone u memoriji koju korisnik sam definiše i kojom se služe i mašinski i BASIC-program.

Na primer, naredba:

SYS 64738

vrši potpunu inicijalizaciju računara, dovodeći ga u isto stanje kao da je tek uključen. Na navedenoj adresi nalazi se početak programa u ROM-u, kojim se vrši postavljanje svih tekućih vrednosti značajnih za rad računara na početne vrednosti.

3.8.2. **Funkcija USR**

Kao i svaka druga funkcija i funkcija USR se navodi u okviru aritmetičkog izraza. Zapis funkcije je sledeći:

USR(<aritmeticki izraz>)

Značenje funkcije je sledeće: Nad vrednošću navedenog argumenta izvršava se postupak obrade zadat mašinskim potprogramom, čija se početna adresa nalazi na za to predviđenom mestu u memoriji (na adresama 785 i 786). Dobijeni rezultat se vraća kao vrednost funkcije i sa njim se nastavlja računanje vrednosti izraza. Prenos podataka (argumenta i rezultata) vrši se preko memorijske zone poznate kao akumulator broj 1, na adresama 97 do 101. Pre samog početka rada korisničkog mašinskog programa, BASIC-interpretator tamo postavlja vrednost argumenta. Mašinski potprogram je dužan da rezultat svoga rada takođe upiše na to mesto, jer se po povratku uzima vrednost funkcije USR.

Postavljanje početne adrese mašinskog potprograma, koji izvršava funkciju USR, vrši se pre njene upotrebe, naredbama POKE. Kraj rada mašinskog potprograma zadaje se kao i kod SYS, mašinskom instrukcijom za povratak iz potprograma (RTS).

Neka se, na primer, mašinskim potprogramom izračunava vrednost neke funkcije f(x) i neka on počinje na adresi ADR. Postupak izračunavanja za $y=f(x)+\sin(f(x))$ bio bi sledeći:

```
10 POKE 785,<ADR niži deo>
20 POKE 786,<ADR viši deo>
30 INPUT X : REM UNOSENJE VREDNOSTI
   ARGUMENTA
40 Y=USR(X)+SIN(USR(X))
50 PRINT Y
```

4. PERIFERIJSKI UREĐAJI

4.1. UVOD

Ni jedan računar, bez obzira na cenu, veličinu ili karakteristike, ne može se ozbiljno upotrebljavati bez nekih dodatnih uređaja, tzv. **periferijske opreme** računara. Zato proizvođači mikroračunara konstruišu i te uređaje, koji su po ceni pristupačni, a po funkciji slični profesionalnoj opremi za velike računske centre. Umesto širokih i dugih magnetskih traka na koturovima upotrebljava se standardna kaseta, umesto paketa magnetskih diskova sposobnih da prime preko 100 miliona znakova upotrebljava se savitljiva plastična disketa daleko manjeg kapaciteta, ali i neuporedivo niže cene. Uređaji za štampanje su malih dimenzija, relativno spori, ali mogu za nekoliko minuta da preslikaju tekst ili sliku sa ekrana na papir.

U ovom poglavlju biće opisani neki od uređaja iz porodice **Commodore**, kojima se računar proširuje u mali računarski sistem, sposoban da omogući brzu i jednostavnu manipulaciju programima i podacima.

4.2. KOMUNIKACIJA SA PERIFERIJSKIM UREĐAJIMA

U nastojanju da se komunikacija sa periferijskim uređajima učini što jednostavnijom, razrađen je opšti mehanizam za komunikaciju sa bilo kojim tipom uređaja. Ona se obavlja pomoću nekoliko standardnih komandi i omogućuje rad na dva nivoa: nivou programa i nivou po-

dataka. Pri tome se tipovi uređaja međusobno razlikuju po broju kojim se označavaju:

1 — kasetofon,
4 ili 5 — štampač,
8 do 11 — disketna jedinica.

Svakom od tipova uređaja odgovara skup dodatnih operacija koje se mogu izvršiti samo na njemu. Ovo se na poseban način zahteva u okviru standardnih naredbi.

Rad sa uređajima baziran je na postojanju posebne, imenovane ili neimenovane grupe podataka koju je moguće formirati i koja se uobičajeno naziva TEKA. Postoje dva osnovna tipa teke:

— teka koja sadrži program (programska teka, programoteka),
— teka koja sadrži podatke (teka podataka, datoteka).

Rad sa programima zasniva se na komandama za prenos programa iz memorije u teku (SAVE), iz teke u memoriju (LOAD) i za proveru zapisa u teci (VERIFY):

SAVE [<ime programa>][,<tip uređaja>][,<tip naredbe>]
LOAD [<ime programa>][,<tip uređaja>][,<tip naredbe>]
VERIFY <ime programa>,[<tip uređaja>]

Pojam "teka" se u radu sa programima ne pojavljuje direktno, već se o tome brine sam računar. Korisniku je time dopušteno da razmišlja na nivou programa, jer je u navedenim naredbama program jedini predmet rada. Zbog toga se često pojam "teka" poistovećuje sa pojmom "datoteka", što je pogrešno, jer je "teka" najopštiji naziv za bilo kakav organizovan skup informacija, bio to program ili podaci.

Rad sa podacima zasnovan je, slično programima, na nekoliko osnovnih naredbi. To su:

Otvaranje teke:
OPEN <logički broj teke>,<tip uređaja>[,<dopunski parametri>]
Upis u teku:
PRINT# <logički broj teke>,<lista>
Čitanje iz teke:
INPUT# <logički broj teke>,<lista>
GET# <logički broj teke>,<lista>

Zatvaranje teke:

CLOSE <logički broj teke>

Uočljivo je da se <tip uređaja> pojavljuje samo prilikom otvaranja teke, dok se prava komunikacija sa tekom vrši preko njoj dodeljenog logičkog broja. Ovo omogućuje spomenuti opšti mehanizam za rad sa uređajima. Preko uvedenih logičkih brojeva on određuje sve ostale potrebne podatke, naznačene u naredbi OPEN, i u skladu sa njima izvršava zahtevanu operaciju.

<logički broj teke> može biti bilo koji broj između 0 i 255. Treba znati da postoji razlika u radu sa tekama otvorenim pod brojevima 0 do 127 i onim od 128 do 255. Kod teka iz prve grupe, posle svake naredbe PRINT# šalje se u teku i specijalni znak za kraj reda, dok se kod teka iz druge grupe pored njega šalje još i kontrolni znak za prelaz u sledeći red (LINE FEED). Ovim je omogućena upotreba nekih nestandardnih štampača koji za prelazak na početak novog reda zahtevaju, jedan za drugim, oba ova simbola. Za normalan rad sa, na primer, kasetofonom ili disketnom jedinicom preporučljivo je zato birati brojeve između 0 i 127.

<tip uređaja> označava se brojem uređaja.

<dopunski parametri> variraju po složenosti od vrlo jednostavnih (kod štampača se ne mora navesti, ili je to jedan od dva broja) do grupe od nekoliko podataka različitog značenja (npr. kod formiranja relativne teke podataka na disketi).

Ulazno/izlazne naredbe PRINT#, INPUT# i GET# po formatu su istovetne sa običnim naredbama PRINT, INPUT i GET. Razlika je u tome što se komunikacija ne obavlja sa ekranom i tastaturom, već sa uređajem na kome se nalazi teka sa navedenim logičkim brojem. Vrlo korisno može poslužiti i naredba:

CMD <logički broj teke>[,<azbučni izraz>]

kojom se omogućuje da se svako izdavanje na ekran (sa PRINT ili LIST) "skrene" u nekom drugom pravcu, tj. da se podaci pošalju u teku umesto da se prikažu na ekranu. Ovo dolazi do punog izražaja kada se teka nalazi na štampaču, jer omogućuje da se dobije kompletan pisani dokument o izvršavanju programa koji je, inače, predviđen

za rad sa ekranom. Ukoliko je drugi argument naredbe naveden, vrednost azbučnog izraza se upisuje u navedenu teku.

Važnost naredbe CMD prestaje izvršavanjem neke od naredbi GET ili PRINT#, ili pojavom greške u programu.

Kraj rada sa tekom (zatvaranje teke) postiže se naredbom CLOSE, čime se teka kompletira i njoj dodeljeni logički broj oslobađa. Izostavljanje ove naredbe ne izaziva grešku u programu, ali na nekim uređajima (kasetofon, disketna jedinica) može dovesti do gubljenja jednog dela podataka (kasetofon) ili čak cele teke (disketna jedinica). Zato se od početka treba navići na njeno disciplinovano korišćenje.

U daljem tekstu ove glave detaljnije se izlaže korišćenje opisanih naredbi u radu sa svakim od periferijskih uređaja.

4.3. KASETOFON

Teško je naći mikroračunar koji nema mogućnost priključenja kasetofona. Bilo da se radi o običnom ili specijalnom tipu kasetofona, zajedničke karakteristike su veliki kapacitet, vrlo niska cena, velika pouzdanost i jednostavnost upotrebe. Nedostaci ovog uređaja, kao što su mala brzina prenosa ili sekvencijalni (tj. neadresivi) oblik zapisa na traci, još dugo neće potisnuti njegovu upotrebu.

4.3.1. **Organizacija podataka na kaseti**

Princip zapisivanja podataka na traku predstavlja pojednostavljenje snimanja muzike ili glasa, jer se zapisuju samo dve vrste podataka: nule i jedinice. Podaci koji se upisuju organizovani su u teku i njen zapis podeljen je u tri dela:

1 — sinhro-signal — označava početak teke,
2 — zaglavlje teke,
3 — podaci.

Osnovna jedinica teke na kaseti naziva se BLOK, fiksne je dužine i obično sadrži između 100 i 256 simbola. Zaglavlje i podaci zapisuju se u obliku blokova, dok se

početak teke označava tonom neke fiksne frekvencije. Uz svaki blok upisuju se i specijalni simboli kojima se pri čitanju proverava ispravnost pročitanog sadržaja.

Zaglavlje teke sledi neposredno iza sinhro-signala i sadrži informacije o sadržaju teke. To su, na primer, ime teke, tip teke (program ili podaci), početna i krajnja adresa memorijske zone u koju se pri punjenju prebacuje sadržaj teke i slično.

Zapis na kaseti je primer **sekvencijalnog zapisa,** jer se podaci upisuju u jednom smeru, redom jedan za drugim i jedino se istim redom mogu i čitati. Ako se želi pristupiti nekom podatku iz teke, moraju se pročitati svi prethodni. Direktne izmene u sadržaju nisu moguće, već se mora čitava teka preneti u memoriju, izvršiti željena izmena a zatim ponoviti snimanje cele teke. Iz ovoga proizlazi da je kaseta pogodnija za čuvanje programa, nego podataka. Za podatke je ima smisla koristiti u slučajevima kada se radi o velikom skupu informacija koje se ne menjaju i pri obradi se uvek ponavljaju istim redom.

Teke ovakvog oblika nazivaju se **sekvencijalne teke.** Sve teke na kaseti, i programske i teke podataka, su sekvencijalnog tipa.

4.3.2. Kasetofon DATASSETTE 1530

Za računar Commodore 64 predviđen je poseban tip kasetofona. Prednost ovakvog rešenja je u mogućnosti programskog pokretanja i zaustavljanja motora i velikoj pouzdanosti zapisa. Operacije kao što su premotavanje trake i izbor snimanja ili učitavanja i dalje se moraju vršiti ručno.

Na kasetofonu se nalazi brojač okretaja kojim je omogućeno odmeravanje pozicioniranja trake. Korisno je za svaku kasetu voditi njen sadržaj, pomoću imena teka i pozicije njihovih početaka u odnosu na početak trake. Tako se može olakšati kasnije pretraživanje kasete i nalaženje potrebnog sadržaja.

Za vreme prenosa podataka između kasetofona i računara gubi se slika na ekranu.

Dve opšte napomene u radu sa ovim uređajem su vrlo važne:

1) Dok je računar uključen **ne sme se vršiti ni priključivanje ni isključivanje kasetofona.**

2) Kasete koje se koriste treba da imaju dobar mehanizam za transport trake i ne treba da budu duže od 2×30 minuta.

Broj kasetofona kao periferijskog uređaja je 1.

Rad sa programima. Programi se na kasetu upisuju komandom SAVE:

SAVE [<ime programa>] [,1] [,<tip upotrebe>]

<ime programa> može biti sastavljeno od najviše 16 simbola. Ukoliko se ne navede, program se pamti bez imena.

<tip upotrebe> označava sadržaj teke koja se formira. Ako se ne navede, u pitanju je upis BASIC-programa i pri kasnijem učitavanju sadržaj teke će se preneti u standardnu zonu za BASIC-program (počev od adrese 2048). Ostale vrednosti su:

1 — snima se mašinski program. U zaglavlje se prepisuje sadržaj sistemskih adresa za početak i kraj programa. Program će se učitavanjem uneti u tu memorijsku zonu.

2 — isto kao da je izostavljen, ali se nakon kraja teke dopisuje oznaka za kraj trake.

3 — isto kao 1, ali se, kao pod 2, upisuje dodatna oznaka za kraj trake.

Oznaka za kraj trake onemogućava traženje teke na preostalom delu.

Program se unosi komandom:

LOAD [<ime programa>] [,1] [,<tip upotrebe>]

Ako se ime izostavi, unosi se prvi program na koji se naiđe.

<tip upotrebe> može biti izostavljen ili 1. Ukoliko je izostavljen, smatra se da se učitava BASIC-program i učitavanje se vrši u zonu od adrese 2048 nadalje. Ukoliko je 1, učitava se mašinski program i početna adresa uzima se iz zaglavlja.

Provera zapisa na kaseti vrši se komandom VERIFY:

VERIFY [<ime programa>]

Sadržaj teke poredi se sa sadržajem memorije, nakon čega se javlja izveštaj o rezultatu. Ova komanda primenjuje se neposredno po snimanju programa naredbom SAVE, jer je to jedini trenutak kada ima smisla porediti zapis na traci sa sadržajem memorije.

Rad sa podacima. Kako je već rečeno, na kaseti se mogu formirati jedino sekvencijalne teke podataka. Oblik naredbe za otvaranje teke je:

OPEN <logički broj teke>,1[,<namena>] [,"<ime>"]

<logički broj teke> je ceo broj između 1 i 127.

<namena> se označava brojem od 0 do 2 a značenje je:

0 — otvaranje teke za čitanje,

1 — otvaranje teke za upis,

2 — otvaranje teke za upis. Po zatvaranju se upisuje oznaka za kraj trake.

Ukoliko je izostavljen, smatra se da je 0.

Prenos podataka između računara i kasetofona vrši se posredstvom posebnog bafera koji se nalazi u memoriji računara, na adresama od 828 do 1029. Veličina bafera jednaka je veličini jednog bloka teke (192 simbola).

Pri otvaranju teke za čitanje, bafer se puni podacima iz prvog bloka teke. Nakon toga, motor se zaustavlja. Poseban pokazivač postavlja se na početak bafera i njegova vrednost uvećava se naredbama INPUT# i GET# za broj učitanih simbola. U trenutku kada pokazivač izađe iz bafera (tj. pročitan je ceo blok) uključuje se motor i u bafer se unosi sadržaj sledećeg bloka. Pokazivač se postavlja na početak.

Pri upisu se naredbom PRINT# vrši upis u bafer. Pokazivač se automatski pomera za broj upisanih simbola. U trenutku kada pokazivač izađe van bafera (tj. bafer je pun), motor se uključuje i sadržaj bafera se upisuje na traku. Motor se zatim zaustavlja i pokazivač postavlja na prvo mesto.

Kraj teke ili pojava greške može se ustanoviti ispitivanjem vrednosti sistemske promenljive STATUS (ST), i tada je ona različita od 0 (jednaka 64 za kraj teke).

Posle svakog upisanog elementa treba da postoji simbol koji će ga pri čitanju odvajati od narednog. Ovaj

simbol naziva se **separator** i može biti predstavljen oznakom za <kraj reda>, ili simbolom ",". Nakon svake izvršene naredbe PRINT# automatski se upisuje i <kraj reda>. Separator "," se ne može upisivati automatski, već se mora eksplicitno navesti u okviru naredbe PRINT#. Dakle, naredbe:

```
PRINT#1,A,B$,"TEKST",5.8
PRINT#1,A$;B;C
PRINT#1,X,Y;
```

ne upisuju separatore među elemente iz liste. Simboli "," i ";" imaju isključivo formatirajući karakter (određuju prazan prostor između dva elementa). Kod prva dva primera nakon poslednjeg elementa upisuje se <kraj reda>, dok se u trećem ne upisuje ni to. Ovako upisani podaci **ne mogu** se pojedinačno učitati naredbom INPUT#. Zato se u slučaju navođenja više elemenata u listi za izdavanje oni moraju razdvojiti upisivanjem separatora. Prethodni primeri tada mogu imati oblik:

```
PRINT#1,A,",";B$,",";"TEKST",",";5.8
PRINT#1,A$;",";B;",";C
PRINT#1,X,",";Y;",";
```

Formatiranje se i dalje vrši na isti način kao i ranije.

Nakon završenog upisa teka se obavezno mora zatvoriti. U suprotnom sadržaj bafera neće biti upisan u teku i ti će podaci biti izgubljeni. Pri zatvaranju teke, neiskorišćeni deo bafera se popunjava bajtovima čija je vrednost 0 i sadržaj bafera upisuje u teku.

Primer 1: SAVE

BASIC-program koji je u memoriji upisuje se na kasetu bez imena.

Primer 2: LOAD "IME",1,1

Sa kasete se učitava mašinski program iz teke IME i smešta u memoriju na mesto naznačeno u zaglavlju teke.

Primer 3: Uočite razliku u radu ova dva programa:

```
10 OPEN 1,1,1,"PRIMER"
20 FOR I=1 TO 10
```

```
30 PRINT#1,I;
40 NEXT I
50 PRINT#1
60 CLOSE 1

10 OPEN 1,1,1,"PRIMERI"
20 FOR I=1 TO 10
30 PRINT#1,I;CHE$(13);
40 NEXT I
50 PRINT#1
60 CLOSE 1
```

Proverite upisani sadržaj obe teke programom:

```
10 INPUT "IME TEKE";I$
20 OPEN 1,1,0,I$
30 IF ST=64 THEN STOP
40 INPUT#1,A$
50 PRINT A$
60 GOTO 30
```

4.4. ŠTAMPAČ

Svaka ozbiljnija primena računara brzo dovodi do potrebe za korišćenjem štampača. Nije lako tražiti grešku u programu od nekoliko stotina redova ako se na ekranu prikazuje svega dvadesetak. Obrada teksta bez štampača je nezamisliva.

Vrlo je teško precizno podeliti štampače prema nekom usvojenom kriterijumu. Proizvode se mnogobrojni tipovi sa vrlo različitim principom rada, od klasičnih, poput pisaće mašine, do neobičnih (tzv. ink-džet) štampača, koji pišu tankim mlazevima mastila. Zato će u ovom poglavlju biti reči samo o najrasprostranjenijima, tzv. **matričnim štampačima.**

4.4.1. Matrični štampači

Naziv ovog tipa potiče od načina formiranja slova. Nasuprot pisaćoj mašini čiji mehanizam za štampanje ima onoliko elemenata koliko i simbola za prikaz, kod matrič-

nih štampača svi se simboli dobijaju iz jednog elementa, tzv. matrice tačaka. Simbol se formira kombinacijom nekih tačaka iz matrice i otiskivanjem na papir preko masti-ljave trake. Ovakav princip mnogostruko redukuje mehaničke delove štampača, ali dodaje neke elemente koji se u ovom uređaju ne bi očekivali:

1 — Mikroprocesor specifične konstrukcije.

2 — ROM za čuvanje programa za upravljanje uređajem i opisa svih simbola koji se mogu prikazati.

3 — RAM kao prostor za brzo prihvatanje velikog broja podataka iz računara i njihovo znatno sporije štampanje. Zahvaljujući ovoj memoriji brze procese u računaru ne koči spori proces štampanja.

Broj tačaka u matrici je različit i kreće se od 7×5, kao donje granice za prihvatljiv izgled teksta do 17×15, čime razlika u odnosu na klasični prikaz punim slovima praktično nestaje.

Slovo "E" se, na primer, u matrici 7×5 predstavlja na sledeći način:

```
*****
*....
*....
*****
*....
*....
*****
```

Sam mehanizam za štampanje čini tzv. "glava za štampanje". U njoj se nalazi jedan red vertikalno raspoređenih iglica. Njihovom kombinacijom dobija se jedna kolona u matrici za prikaz simbola i matrica od, npr. 7×5 tačaka štampa se u 5 odvojenih vertikalnih kolona sa po 7 tačaka, jer glava sadrži 7 tačaka.

Pored broja iglica u glavi važan faktor je i finoća pomeranja glave, jer se time može dobiti različita gustina zapisa.

Velika prednost matričnih štampača nad ostalima je mogućnost da se raspored tačaka u glavi, odnosno matrici, neposredno definiše. Tako se mogu dobiti različiti nestandardni simboli ili formirati slika. Na taj način svaki

sadržaj ekrana može biti prenet na papir. Tako štampač može zameniti specijalizovane uređaje za crtanje.

Dakle, mogućnosti matričnih štampača su vrlo velike. Zato se pojavljuje potreba da se njime upravlja iz računara. Iz skupa podataka koji se mogu uputiti štampaču (obično od 0 do 255) izdvajaju se neki koji imaju komandni karakter. Nakon njihovog prijema u štampaču, način štampanja se menja sve do prijema nekog drugog takvog simbola. Ovi simboli nazivaju se **kontrolni simboli.** Funkcije koje se njima mogu postići su, na primer, prelaz na direktno opisivanje otiska glave (tzv. grafički režim), zadavanje veličine slova, zadavanje veličine proreda i slično.

Papir koji se koristi za štampanje dugo je bio specijalnog tipa, sa listovima spojenim u beskonačnu traku i izbušenim rupama po ivici (perforacijom) za vođenje. Ovakav način transporta papira kroz štampač naziva se **traktorski prenos.** Danas je sve više u upotrebi i tzv. **frikcioni prenos** kod koga se, slično pisaćoj mašini, koriste obični listovi papira, ili papir u rolni.

Prenos podataka od računara ka štampaču može se obavljati serijski (jedan podatak prenosi se u osam delova, bit po bit, jednom linijom) ili paralelno (svih osam bita odjedanput, svaki svojom linijom). Standardizovani su, i najrasprostranjeniji, RS232 (serijski) i CENTRONICS (paralelni).

4.4.2. Štampač MPS-801

Štampač MPS-801 spada u donju klasu matričnih štampača. Matrica kojom formira simbol je 7×5 tačaka, a veza sa računarom ostvaruje se posebnom, u osnovi serijskom komunikacijom. Zato se može koristiti isključivo na Commodore računarima. Niska cena i jednostavno priključivanje i upotreba doprineli su da se mnogi korisnici računara opredele za ovaj model.

Za štampanje se koristi standardni perforirani papir u obliku beskonačne trake. U jednom redu štampa se 80 simbola, a finoća pomeranja glave se ne može određivati.

Mogućnosti uređaja su sledeće:

1 — Pozicioniranje glave (na početak novog reda ili neko mesto u redu).

2 — Izbor jedne od dve azbuke (istovetne azbukama u računaru).

3 — Štampanje uvećanih slova (40 u redu).

4 — Inverzno štampanje (belo na crnom).

5 — Programiranje otiska glave (nestandardni simboli, grafika . . .)

Korišćenje. Otvaranje teke na štampaču je primer najprostijeg oblika naredbe OPEN:

OPEN <logički broj teke>,<broj štampača>[,<azbuka>]

<logički broj teke> je ceo broj između 1 i 127.

<broj štampača> može biti 4 ili 5, što se definiše položajem preklopnika na samom uređaju.

<azbuka> određuje skup simbola koji će se prikazivati. Može biti izostavljen (ili 0) kada je odgovarajući skup velika slova/grafički znaci, ili 7 kada je skup mala/velika slova.

Teka se na štampaču otvara samo za upis. Podaci se izdaju naredbom PRINT#, u istom obliku kao za izdavanje na ekran.

U radu sa štampačem komanda CMD dolazi do punog izražaja. Nakon testiranja rada programa na ekranu izdavanje se može skrenuti na štampač i tako dobiti kompletna dokumentacija.

Kombinacijom CMD i LIST dobija se i listing programa na štampaču.

Ne treba zaboraviti da dejstvo komande CMD prestaje pojavom greške ili izvršavanjem neke od naredbi GET ili PRINT#.

Kontrolni simboli upućuju se štampaču korišćenjem funkcije CHR$. Argumenti funkcije mogu biti:

8 — uključuje direktno programiranje glave,
10 — prelaz u sledeći red,
12 — prelaz na početak sledećeg reda,
14 — uvodi režim uvećanih slova (40 u redu),
15 — uvodi režim standardnog prikaza (80 u redu),
16 — pozicionira mesto simbola u redu,
17 — uvodi skup simbola mala/velika slova,
18 — uključuje inverzni prikaz,

26 — ponavlja programirani oblik,
27 — pozicionira adresu tačke u redu,
145 — uvodi skup simbola velika slova/grafički znaci,
146 — isključuje inverzni prikaz.

Zatvaranje teke na štampaču vrši se uobičajeno, naredbom CLOSE.

Isprobajmo efekat kontrolnih simbola na sledećem primeru:

Neka je teka na štampaču otvorena sa:

```
OPEN 4,4
```

i neka je računar u režimu mala/velika slova.

1. **Pozicioniranje glave.** Glava se može pozicionirati na dva načina: na početak novog reda i na neko mesto u tekućem redu:

```
print#4,chr$(12);chr$(10);chr$(12)
```

Ovom naredbom preskaču se tri reda.

Nakon kontrolnog simbola za pozicioniranje glave 16 treba poslati još dva dodatna, takođe pomoću CHR$ funkcije. To su cifre dvocifrenog broja čija vrednost predstavlja novu poziciju glave. Na primer:

```
print#4,chr$(16);chr$(4);chr$(0);"40. pozicija"
```

dovodi glavu na četrdesetu poziciju i štampa navedeni tekst.

Ukoliko je trenutna pozicija glave veća od tražene, nema efekta (povratak glave nije moguć).

2. **Izbor azbuke.** Početno definisanje azbuke iz naredbe OPEN može se menjati iz programa kontrolnim simbolima 17 i 145:

```
a$="VELIKA/mala slova"
print#4,chr$(17);a$
print#4,chr$(145);a$
```

Sada je štampač ponovo u režimu mala/velika slova.

3. **Štampanje uvećanih slova.** Svi simboli navedeni nakon CHR$(14) biće prikazani dvostruko šire nego obični. To je naročito pogodno ako se žele dobiti listinzi ili izveštaji koji imaju iste formate kao i na ekranu (koji takođe ima 40 mesta).

Nakon CHR$(15) štampaju se simboli obične širine.

```
print#4,chr$(14);"UVECANI TEKST"
print#4,chr$(15);"OBICAN TEKST"
```

4. **Inverzno štampanje.** Kontrolni simboli za uključenje i kraj inverznog prikaza teksta imaju iste vrednosti na štampaču kao i na ekranu. Tako je očuvana jednoobraznost prikaza u oba slučaja.

```
print#4,chr$(18);"inverzno";chr$(146);" obično"
```

Isti efekat dobija se sa:

```
print#4," { RVS ON} inverzno { RVS OFF} obično"
```

5. **Programiranje otiska glave.** Ekvivalentna ekranska komanda ne postoji. Nakon štampanja CHR$(8) svi sledeći podaci predstavljaju zadavanje direktnog opisa jedne vertikalne kolone od 7 tačaka. Zato se kaže da se štampač nalazi u "režimu za grafički prikaz".

Programiranje se vrši na sledeći način: svakoj od 7 tačaka u glavi odgovara jedan bit podatka koji se šalje.

□ bit 6
□ bit 5
□ bit 4
□ bit 3
□ bit 2
□ bit 1
□ bit 0

Bit 7 mora obavezno biti postavljen na 1, a za svaku tačku za prikaz postavlja se vrednost odgovarajućeg bita 1. Prazna mesta postavljaju se na 0. Na primer, podatak 255 definiše punu vertikalnu crtu, a podatak 227 sledeće tačke:

■ bit 6=1
■ bit 5=1
□ bit 4=0
□ bit 3=0
□ bit 2=0
■ bit 1=1
■ bit 0=1

jer je 227=128+64+32+2+1.

Definisanje novih simbola vrši se prevođenjem svake od kolona iz njihovih matrica, u dekadni broj i njihovim izdavanjem u grafičkom režimu. Na primer, program:

```
10 OPEN 4,4
20 PRINT#4,CHR$(8)
30 FOR I=1 TO 5
40 READ A
50 PRINT#4,CHR$(A);
60 NEXT I
70 PRINT#4,CHR$(15)
80 STOP
100 DATA 255,129,255,129,255
```

štampa ćirilično slovo Š.

Prelaz iz grafičkog režima na tekst vrši se komandnim simbolom 15.

Napomenimo još i to da prelaz u novi red izvršen u grafičkom režimu ne ostavlja prored između redova, što je vrlo značajno za formiranje grafičkih likova na većoj površini papira. U sledećem primeru ova činjenica se koristi za dobijanje punog, crnog četvorougla:

```
10 OPEN 4,4:PRINT#4, CHR$(8)
20 FOR I=1 TO 50
30 A$=A$+CHR$(255)
40 NEXT I
50 FOR I=1 TO 7
60 PRINT#4,A$
70 NEXT I
80 PRINT#4,CHR$(15):CLOSE 4
```

U redu 10 otvara se teka na štampaču i uključuje grafički režim. Štampanje više grafičkih podataka odjedanput često se izvodi njihovim upisom u neku azbučnu promenljivu, kao u redovima 20 do 40. Vrednost promenljive se zatim štampa jednom naredbom print (red 60).

Upravo u ovakvim slučajevima može se koristiti kontrolni simbol za ponavljanje programiranih opisa. Posle kôda CHR$(16) navode se dva podatka: broj ponavljanja

i opis. Navođenje se vrši korišćenjem funkcije CHR$. Tako se prethodni program može napisati kao:

```
10 OPEN 4,4:PRINT#4,CHR$(8)
20 FOR I=1 TO 7
30 PRINT#4,CHR$(26);CHR$(50);CHR$(255)
40 NEXT I
50 CLOSE 4
```

Pozicioniranje tačke u redu na koju će doći glava za štampanje vrši se kontrolnim simbolom 27. Argumenti koji se navode iza njega, predstavljaju niži i viši deo adrese tačke (ND i VD). Računaju se na sledeći način:

```
VD=INT(PT/256)
ND=PT—VD*256
```

Isprobajte dejstvo ove naredbe dodavanjem reda:

```
25 PRINT#4,CHR$(27);CHR$(50);CHR$(0);
```

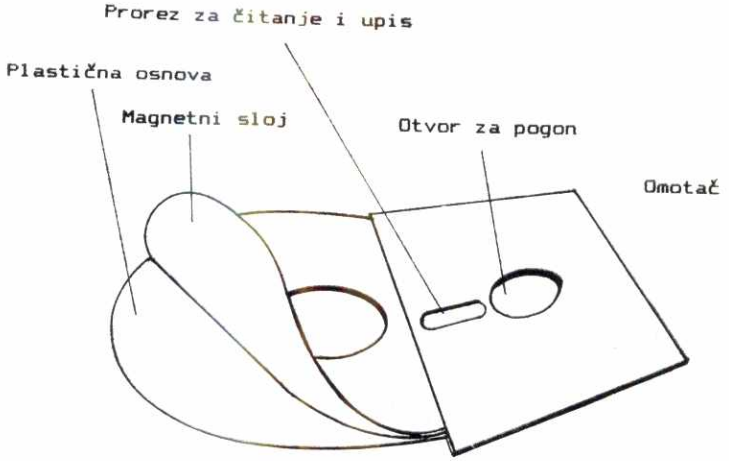
u prethodni program.

4.5. DISKETNA JEDINICA

4.5.1. **Organizacija podataka na disketi**

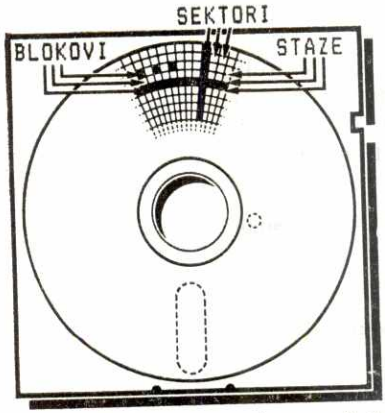
Kao što je kaseta u kućnoj upotrebi ekvivalent profesionalnoj magnetskoj traci, tako disketa odgovara magnetskom disku velikih računara. Uređaj koji služi za čitanje i upis podataka na disketu naziva se disketna jedinica. Princip čuvanja informacija u osnovi je isti: u oba slučaja radi se o površini kružnog oblika prevučenoj feromagnetskim slojem sposobnim da trajno zabeleži namagnetisanje. Sadržaju diska, odnosno diskete, pristupa se pomoću glave za čitanje i pisanje, koja se i sama može pomerati. Dok se disk kreće rotirajući oko centra, glava se pomera linijski, po poluprečniku diska. Na taj način je, kombinacijom ova dva kretanja, omogućeno da glava može pristupiti bilo kom delu na površini diska, pročitati njegov sadržaj ili upisati novi.

Radi zaštite osetljive površine i lakše manipulacije disketa se pakuje u kvadratni plastični ili kartonski omotač. Na njemu se ostavljaju dva otvora: centralni otvor za kontakt između same diskete i osovine motora koji je pokreće i poprečni otvor kroz koji glava dolazi u dodir sa površinom diskete.



Sl. 4.1 — Disketa

Površina diskete izdeljena je u koncentrične kružne trake koje se obično nazivaju **staze.** Broj staza se kreće od 35 do 80, zavisno od preciznosti pomeranja i postavljanja glave u disketnoj jedinici. Svaka od staza podeljena je na segmente, **sektore,** i u ovaj prostor obično može da se upiše 128 ili 256 bajtova. Sadržaj jednog sektora naziva se **blok** i to je najmanji skup podataka na disketi koji ima svoju adresu: sektor i stazu. Broj sektora po stazi zavisi od gustine kojom glava može da pakuje sadržaj. Dve gustine postoje kao standardne: obična (jednostruka, SINGLE DENSITY) i dvostruka gustina (DOUBLE DENSITY). U disketnoj jedinici se mogu nalaziti samo jedna ili dve glave — po jedna za svaku stranu diska. Otuda i podela na jednostrane (SINGLE SIDED) i dvostrane (DOUBLE SIDED).



Sl. 4.2 — Struktura zapisa na disketi

Shodno navedenim standardima diskete se proizvode u četiri osnovna tipa:

— jednostrana/obična gustina (SS/SD),
— jednostrana/dvostruka gustina (SS/DD),
— dvostrana/obična gustina (DS/SD),
— dvostrana/dvostruka gustina (DS/DD).

Diskete dolaze u prodaju sa naznačenim tipom disketne jedinice kojoj su namenjene. Razumljivo je da se običnom gustinom može pisati i na disketama namenjenim za dvostruku.

Na samoj disketi izdvaja se jedna ili više staza koje neće biti korišćene za zapisivanje programa i podataka, već će čuvati informacije vezane za sadržaj i raspored tih podataka na disketi. Među njima se izdvajaju dva skupa informacija:

1. **Katalog** (Directory) sadrži listu svih programa i datoteka sa jedne diskete, uključujući njihova imena, opis sadržaja, adresu (stazu i sektor) prvog bloka u kome se nalaze podaci itd.

2. **Mapa dodeljenih blokova** (Block Allocation Map — BAM) predstavlja opis trenutnog stanja svih blokova na disketi. Svakom bloku dodeljen je po jedan bit u mapi i ako je njegova vrednost 1, blok je slobodan. U suprotnom je zauzet i ne može biti korišćen.

Da bi se jedna disketna jedinica mogla priključiti na postojeći računarski sistem, moraju postojati sledeći elementi:

— skup elektronskih komponenti za kontrolu uređaja,
— skup programa koji omogućuju manipulaciju sa podacima na disketi,
— elektronski sklop (interfejs) preko koga će se povezati računar i disketna jedinica.

Celina koju sačinjavaju sva tri navedena dela u jednom kućištu u poslednje vreme se često naziva "disketni sistem". "Commodore 1541" je primer jednog takvog disketnog sistema.

4.5.2. Opšte napomene o rukovanju disketama

Jednostavna konstrukcija diskete čini je osetljivom na mnoge uticaje. Zato se pri radu treba pridržavati nekih opštih pravila:

1 — ne savijati,
2 — ne ostavljati u blizini peći ili na suncu,
3 — nalepnice koje se stavljaju na disketu ispuniti pre lepljenja,
4 — ne držati u blizini telefona ili na televizoru,
5 — vratiti u omot odmah posle upotrebe,
6 — ne dodirivati površinu diskete ili pokušavati da se očisti,
7 — izvući disketu iz disketne jedinice pri uključenju ili isključenju.

4.5.3. Disk-jedinica Commodore-1541 — karakteristike

Prema ranije navedenoj podeli Commodore 1541 spada u jednostrane disketne jedinice obične gustine (SINGLE SIDED/SINGLE DENSITY). Dakle, svaka se disketa, bez obzira na oznaku, može koristiti u ovom uređaju. Na disketi se koristi 35 staza, označenih od 1 do 35, pri čemu je staza 1 najširа, krajnja spoljašnja staza. Svaka staza deli se na sektore u koje staje po jedan blok od 256 bajtova. Broj sektora po stazi je promenljiv, pa šire, spoljašnje

staze imaju po 21, središne po 20 ili 18 a krajnje unutrašnje, najuže staze po 17 sektora. Ukupan kapacitet diskete je oko 170 kB.

Staza	Broj sektora
1 do 17	21
18 do 24	19
25 do 30	18
31 do 35	17

Staza 18 rezervisana je za čuvanje kataloga (sektori 1 do 19) i mape dodeljenih blokova (sektor 0).

Kako je već rečeno, Commodore 1541 je primer disketnog sistema. U svojoj unutrašnjosti sadrži elektronske i mehaničke komponente koje mu dopuštaju punu autonomnost. Mehanički deo omogućuje da se pri zatvorenim vratima disketa okreće brzinom od oko 300 obrtaja u minutu. Specijalni motor zadužen je za pomeranje glave, njeno podizanje i spuštanje na površinu diska.

Prema elektronskim komponentama koje sadrži, disketna jedinica pre podseća na računar nego na perifernu jedinicu:

— Mikroprocesor 6502 obezbeđuje sve upravljačke i kontrolne operacije.

— ROM od 16 kB sadrži sve programe neophodne za rad uređaja i čini ga dovoljno "inteligentnim" da može da radi nezavisno od računara. Kontroliše sve mehaničke funkcije i održava vezu sa računarom. Zahvaljujući njemu, od računara se očekuje jedino da saopšti komandu, a zatim se ona izvršava od strane ovih programa. Računar je slobodan za obavljanje drugih radnji.

— RAM od 2 kB predstavlja prostor za razmenu podataka između računara i disketne jedinice. Podeljena je u grupe od 256 bajtova.

Komunikacija računara i disketne jedinice uspostavlja se preko jednog od 16 posebnih kanala preko kojih se prenose podaci ili program. Kanali su označeni brojevima od 0 do 15. Ukoliko se radi o upisu ili čitanju programa, naredbama LOAD i SAVE, koriste se kanali 0 i 1. Njihovo otvaranje i zatvaranje se u tom slučaju vrši automatski od strane BASIC-interpretatora (na početku i na kraju

izvršavanja naredbe). Za prenos podataka kanal se mora otvoriti korišćenjem naredbe OPEN i ostaje otvoren sve do zatvaranja naredbom CLOSE. Pri tome se mogu, po izboru, koristiti kanali sa brojevima 2 do 14.

Kanal 15 je specijalni komandni kanal preko koga se upućuju komande disketnoj jedinici i čitaju izveštaji o greškama.

4.5.4. **Rad sa programima**

U odnosu na kasetofon, naredbe za čitanje, upis i proveru programa koriste se u radu sa disketnom jedinicom u nešto izmenjenom obliku. Broj uređaja je obavezno navoditi. BASIC-programi se, dakle, učitavaju sa:

```
LOAD "<ime>",8
```

a upisuju i proveravaju sa:

```
SAVE "<ime>",8
VERIFY"<ime>",8
```

Za učitavanje mašinskih programa koji moraju doći tačno na ono mesto na kome treba da se izvršavaju, predviđena je varijanta naredbe LOAD:

```
LOAD "<ime>", 8,1
```

Na istu disketu ne mogu biti snimljena dva programa sa istim imenom, pa je omogućen postupak zamene starog programa novim. Ovo se označava navođenjem simbola "@:" ispred imena programa u naredbi SAVE:

```
SAVE "@:<ime>",8
```

Ovakav način korišćenja lako može izazvati i neprijatnosti, jer se može dogoditi da, ukoliko je novi program duži od starog, dođe do oštećenja nekih drugih podataka na disketi. Zato je preporučljivo praviti stalno nove verzije, sa različitim imenima, a povremeno vršiti brisanje starih, nepotrebnih programa.

Primer 1: `SAVE "PRIMER",8`

Tekući BASIC-program se pamti na disketi pod imenom PRIMER.

Primer 2: `LOAD "PRIMER",8`

BASIC-program pod imenom PRIMER učitava se sa diskete u memoriju. Ako je neki program već bio u memoriji, njegov sadržaj se gubi.

Primer 3: LOAD "PRIMER",8,1

Mašinski program PRIMER unosi se sa diskete u memoriju počev od adrese koja je na poseban način uključena u program.

Primer 4: SAVE "@:PRIMER",8

Tekući BASIC-program upisuje se na disketu na mesto starog programa pod istim imenom PRIMER.

Čitanje kataloga diskete. Katalog se na disketi smatra posebnim programom za koji je rezervisano ime "$". Zahvaljujući tome, sa katalogom se manipuliše kao sa svakim drugim programom.

Učitavanje kataloga u memoriju vrši se naredbom:

LOAD "$",8

Sadržaj kataloga dobija se komandom LIST.

Osnovni nedostatak ovakvog postupka je to što svako korišćenje naredbe LOAD povlači za sobom i gubljenje tekućeg programa iz memorije. Zato se katalogu treba obraćati tek pošto je tekući program zapamćen.

4.5.5. Korišćenje komandnog kanala

Kako je disketna jedinica Commodore 1541 inteligentan uređaj, od računara se očekuje jedino da zada komandu za obavljanje određene operacije. Samo izvršavanje obavljaju programi iz ROM-a disketne jedinice, koristeći sopstveni procesor. Postupak zadavanja komande organizovan je tako da se pri tome koriste isti mehanizmi kao i za prenos podataka. Da bi se napravila razlika, izdvojen je poseban kanal (kanal 15), preko koga se prenose isključivo komande.

Sa gledišta korisnika, komandnim kanalom se manipuliše na isti način kao i tekom. Mora se otvoriti naredbom:

OPEN <logički broj teke>,8,15

a potom se u njega može upisivati naredbom PRINT# ili čitati naredbama INPUT# i GET#. Upisivati se mogu

isključivo komande, a pri čitanju se dobijaju izveštaji o greškama. Na kraju rada komandni kanal treba zatvoriti naredbom:

CLOSE <logički broj teke>

Komande koje se zadaju disketnoj jedinici mogu se generalno podeliti u tri grupe:

— komande na nivou podatka,
— komande na nivou teke,
— komande na nivou diskete ili disketne jedinice.

Komande za rad sa podacima omogućuju pristupanje nekom od zapisanih podataka i vezane su za rad sa tekama podataka. One će biti razmatrane pojedinačno i detaljno u sledećim poglavljima.

Komande iz druge dve grupe omogućuju različite operacije sa pojedinim tekama, celom disketom ili disketnom jedinicom. Mogu se zadavati na dva načina:

OPEN <logički broj teke>,8,15,"<komanda>"

ili upisom na već otvoren kanal 15:

OPEN <logički broj teke>,8,15
PRINT# <logički broj teke>,"<komanda>"

Za tu namenu postoje sledeće komande:

1. **Formatiranje nove diskete (NEW ili skraćeno N).** Tek kupljena disketa ne može se koristiti pre nego što se obavi njeno formatiranje. Pod tim se podrazumeva postupak upisivanja na disketu svih onih (za korisnika nevidljivih) podataka koji su neophodni za njeno kasnije korišćenje (oznake za početak svakog bloka, katalog, mapa dodeljenih blokova). Ovom naredbom se istovremeno zadaje ime diskete i poseban kôd za identifikaciju, po kome će se lako razlikovati od drugih disketa.

Formatiranje se obavlja komandom:

NEW:<ime diskete>,<identifikator>

<ime diskete> sastoji se od najviše 16 simbola i upisuje se kao naslov kataloga.

<identifikator> je sastavljen od dva proizvoljna simbola i omogućuje da se otkrije eventualna zamena diskete u toku rada (onda kada to nije dozvoljeno, na primer kod otvorene teke).

Proces formatiranja je odličan primer za paralelan rad računara i disketne jedinice. Računar je slobodan za dalji rad odmah po zadavanju komande, dok će disk jedinica ovim poslom biti zauzeta narednih 80 sekundi.

Ukoliko se ovaj postupak obavi sa starom, već formatizovanom disketom, njen je sadržaj u potpunosti izgubljen. Ukoliko se, međutim, navede samo ime diskete bez identifikatora:

NEW:<ime diskete>

biće obrisan katalog i oslobođena sva mesta za podatke u mapi dodeljenih blokova. Efekat je isti kao i da je sadržaj fizički izbrisan, ali je postupak daleko brži.

2. **Uređenje sadržaja diskete (VALIDATE ili V).** Dugotrajna upotreba jedne diskete, brisanje starih teka i sve greške koje nastaju pri upisivanju čine da se na disketi pojavljuju neki delovi koje nije moguće koristiti. Na primer, svaka nezatvorena teka na disketi zauzima prostor kao da je zatvorena, ali se njenom sadržaju ne može pristupiti. Uređenjem se ovaj prostor oslobađa za ponovno korišćenje.

Oblik ove komande je:

VALIDATE ili samo V

3. **Inicijalizacija disketne jedinice (INITIALIZE ili I).** Pojavu greške u radu sa disketom ne treba rešavati isključivanjem disketne jedinice. Dovođenje u početno stanje omogućeno je komandom koja ima isti efekat kao i pritisak na prekidač, ali ne izaziva, ponekad kobno, prekidanje kontakta računara i disketne jedinice.

Oblik naredbe je:

INITIALIZE ili I

4. **Promena imena teke (RENAME ili R).** Jednom dodeljeno ime upisuje se u katalog i može se promeniti jedino komandom RENAME. Nakon toka staro ime se gubi, a na njegovo mesto upisuje novo:

RENAME:<novo ime>=<staro ime>

5. **Brisanje teke (SCRATCH ili S).** Nepotrebne teke mogu biti isključene iz kataloga i njihov prostor oslobođen za ponovnu upotrebu korišćenjem komande:

SCRATCH:<ime teke> ili S:<ime teke>

Razume sa da pri korišćenju ove komande treba biti posebno obazriv.

6. **Kopiranje teke (COPY ili C).** Sadržaj teke može se kopirati u novu teku (duplirati) ili se nekoliko teka (najviše četiri) može spojiti u jednu. Izvorne teke se ni u čemu ne menjaju, već se od njihovog sadržaja formira nova. Oblik naredbe je:

COPY:<nova teka>=<teka1>[,<teka2> . . .]

Obrada greške. Pojava greške u obradi zadatog zahteva takođe uključuje upotrebu komandnog kanala. Kako se obrada izvršava u samoj disketnoj jedinici, ne postoji način da se izveštaj o greškama prikaže na ekranu. To može da se označi jedino treptanjem crvene signalne lampe. Kompletan izveštaj o grešci dobija se od disk jedinice učitavanjem sledećih elemenata sa komandnog kanala:

1 — broj greške,
2 — tekst izveštaja (opis greške),
3 — staza i sektor gde je greška nastala.

Ovim elementima u svakom trenutku je opisan STATUS disketne jedinice.

Učitavanje statusa obavlja se naredbom INPUT# sa najviše 4 elementa. Na primer, sa:

INPUT#1,A,B$,C,D

dobija se broj greške u A, tekst u B$ staza u C i sektor u D.

Primer 1: OPEN 15,8,15,"N:DISK1,D1"

Disketa se formatizuje, dodeljuje joj se ime DISK1 i identifikator D1.

Primer 2: OPEN 15,8,15 : PRINT#15,"V"

Uređuje se sadržaj diskete, tako da se oslobađa sav nepotrebno zauzet prostor.

Primer 3:

```
OPEN 15,8,15
PRINT#15,"I"
```

Postupak koji je ekvivalentan ponovnom uključenju disketne jedinice. Uređaj se dovodi u početno stanje. Ako se pročita komandni kanal sa:

```
INPUT#15,A,B$,C,D : PRINT A;B$;C;D
```

dobija se:

```
0 OK 0 0
```

što znači da je status uređaja "greška ne postoji".

Primer 4: OPEN 8,8,15,"S:PROGRAM1"

Teka sa imenom PROGRAM1 briše se iz kataloga.

Primer 5: OPEN 5,8,15,"R:PRIMER5=PR5"

Ranija teka PR5 dobija novo ime PRIMER5.

Primer 6: OPEN 1,8,15, "C:PRIMER6=PR6A,PR6B"

Sadržaj dve postojeće teke PR6A i PR6B prepisuje se u novu teku pod imenom PRIMER6.

Primer 7: Ako se komanda zada u obliku:

OPEN 2,8,15,"N P7,12"

zatreptaće kontrolna signalna lampa. Čitanjem komandnog kanala sa:

```
INPUT#2,A,B$,C,D : PRINT A;B$;C;D
```

dobija se izveštaj:

```
34 SYNTAX ERROR 0 0
```

jer je posle "N" u komandi izostavljen znak ":".

4.5.6. **Rad sa podacima**

Prave prednosti disketne jedinice nad kasetofonom uočavaju se tek kod rada sa tekama podataka. Mogućnost da se po izboru pristupa različitim delovima diskete prevazilazi ograničenje trake i omogućuje da se podaci organizuju u nekoliko različitih tipova teka:

1 — Sekvencijalne — (Sequential Filas) podacima se manipuliše slično kao na kaseti, pa i ograničenja ostaju: da bi se došlo do nekog podatka, moraju se pročitati svi prethodni iz datoteke i mogućnost "koraka unazad" ne postoji.

2 — Relativne — (Relative Files) programeru se omogućuje da svoju teku "konstruiše" tako da se svakom podatku ili grupi podataka može pristupiti direktno i nezavisno od ostalog sadržaja teke.

3 — Direktne — (Random Files) dopuštaju programeru da u svoju teku uključi blokove sa proizvoljnih staza i sektora, po slobodnom izboru.

Rad sa sekvencijalnim tekama. U obliku sekvencijalnih teka na disketi se čuvaju ne samo sekvencijalne teke podataka, već i sve programske teke i korisničke teke. Kako je već rečeno, programske teke mogu biti unete u računar korišćenjem LOAD naredbe, ali se mogu koristiti i kao obične sekvencijalne teke.

Opšti oblik otvaranja sekvencijalne teke može se prikazati na sledeći način:

OPEN <logički broj>,8,<broj kanala>,"<ime>,<tip>[,<upotreba>]"

<logički broj> određuje sam korisnik i treba ga uzimati između 1 i 127 (videti napomenu kod opšteg oblika naredbe OPEN).

<broj kanala> se može zadati između 2 i 14, jer su kanali 0, 1 i 15 rezervisani za posebne namene. Ukoliko se želi koristiti više sekvencijalnih teka istovremeno, one moraju biti otvorene na različitim kanalima. Ukoliko se dogodi da se na već dodeljenom kanalu otvori nova teka, prethodno otvorena teka se automatski zatvara. Istovremeno mogu biti otvorene, na različitim kanalima, najviše tri sekvencijalne teke.

<ime teke> je istog formata kao i ime programa.

<tip teke> označava o kakvom se sadržaju radi. Može biti:

S — sekvencijalna teka podataka,

P — programska teka,

U — korisnička teka.

<upotreba> označava operaciju koja će se primeniti nad tekom. Pored čitanja i pisanja koje je postojalo i kod kasetofona, ovde se zahvaljujući organizaciji diskete pojavljuju još dve moguće upotrebe: dopisivanje podataka u već formiranu teku i čitanje loše zatvorene teke. Indikator operacije je:

R — čitanje,

W — formiranje teke,

A — dopisivanje,

M — čitanje loše zatvorene teke.

Ukoliko se <upotreba> izostavi, smatra se da je u pitanju čitanje (R). Zatvaranje sekvencijalne teke vrši se naredbom:

CLOSE <logički broj>

Za razliku od kasetofona kod koga se u slučaju nezatvaranja teke pri upisu gubi samo deo podataka koji još nije prenet na traku, na disketi se u tom slučaju gubi mogućnost čitanja i prethodno upisanog sadržaja. Ovo objašnjava potrebu za postojanjem tipa upotrebe M, bez koga bi se postupak formiranja teke (koji traje i po nekoliko sati) morao ponoviti.

Formiranje nove teke i upis podataka. Format naredbe je:

OPEN <logički broj>,8,<broj kanala>, "<ime>, <tip>,W"

Izvršavanje ove naredbe sprovodi se u nekoliko faza:

1. U katalogu se traži navedeno ime teke i ako se pronađe, preko komandnog kanala javlja se izveštaj o grešci FILE EXIST. Ne dopušta se formiranje nove teke.
2. Ako ime nije nađeno, ono se upisuje u katalog. Uz njega se upisuje i tip teke i oznaka da teka nije zatvorena, što se u listingu kasnije može videti kao zvezdica.
3. Traži se slobodan blok na disketi i ako se pronađe, njegova se adresa upisuje u katalog (dodeljuje se za upis podataka).
4. Na mesto koje u katalogu označava broj blokova u teci postavlja se nula.

Razumljivo je da se, slično naredbi SAVE, pri formiranju teke u imenu ne smeju navoditi simboli "*" i "?", što je inače moguće pri svim ostalim tipovima upotrebe. Ukoliko se želi formirati teka sa istim imenom, kao što je neka već postojeća, može se ispred imena navesti "@:". Stara teka se tada gubi i na njeno mesto upisuje nova.

Upisivanje podataka vrši se na uobičajen način naredbom:

PRINT# <logički broj>,<lista>

Treba voditi računa da su, kao i na kaseti, podaci međusobno razdvojeni nekim od simbola <kraj reda> ili "," (ASCII kôdovi 13, 44). Automatski se zapisuje jedino separator <kraj reda>, dok se "," mora upi-

sivati kao i podaci. Dakle, ukoliko u listi ima više od jednog elementa, oni neće moći da se učitaju pojedinačno, jer simboli "," i ";" u PRINT naredbi imaju samo formatirajući karakter, i **ne upisuju** separator u teku. Da bi se podaci razdvojili koristi se isti postupak kao i kod kasetofona.

Čitanje sekvencijalne teke. Format otvaranja za čitanje je:

OPEN <logički broj>,8,<broj kanala>,"<ime>,<tip>[,R]"

Ukoliko navedena teka postoji, omogućeno je prihvatanje njenog sadržaja korišćenjem naredbi:

INPUT# <logički broj>,<lista>

GET# <logički broj>,<lista>

Prihvatanje podataka vrši se počev od prvog upisanog podatka i već pročitani podaci ne mogu se više čitati (sve dok se teka ne zatvori, ponovo otvori i ponovi čitanje od početka).

Naredbom INPUT# unosi se sadržaj između dva separatora, dok se naredbom GET# dobija sadržaj sledećeg bajta u zapisu, bez obzira je li to deo podatka ili separator. Ukoliko broj elemenata u teci nije poznat (što je najčešći slučaj), otkrivanje kraja teke može se vršiti ispitivanjem vrednosti sistemske promenljive STATUS (ST). Ukoliko je njena vrednost 64, pročitan je i poslednji podatak iz teke.

Proširivanje teke. Format naredbe OPEN je isti kao i kod formiranja nove teke, ali se kao tip upotrebe umesto W navodi A. Nakon toga se podaci zadaju naredbom PRINT#. Smeštanje podataka vrši se na kraj stare teke.

Čitanje loše zatvorene teke. Format naredbe OPEN je isti kao i za čitanje, ali se umesto tipa R navodi M. Da bi se sadržaj loše zatvorene teke sačuvao, treba otvoriti još jednu sekvencijalnu teku i u nju upisivati podatke pročitane iz loše zatvorene teke. Nakon što je i poslednji podatak pročitan iz stare i upisan u novu teku, novu treba zatvoriti, a staru izbrisati. Čitanje loše zatvorene teke ne razlikuje se od čitanja korektno zatvorene.

Primer 1: U ovom primeru formira se sekvencijalna teka podataka od tri podatka, a zatim se ti podaci čitaju iz teke.

```
10 OPEN 8,8,8,"TEKA1,S,W"
20 PRINT "UNESITE TRI AZBUCNA PODATKA"
30 INPUT A$,B$,C$
40 PRINT#8,A$
50 PRINT#8,B$
60 PRINT#8,C$
70 CLOSE8
80 PRINT "PRITISNITE 'P' ZA POCETAK CITANJA"
90 GET A$:IF A$<>"P" THEN 90
100 OPEN 8,8,8,"TEKA1,S,R"
110 INPUT #8,A$,B$,C$
120 PRINT A$
130 PRINT B$
140 PRINT C$
150 CLOSE 8
160 STOP
```

Na početku se teka otvara za upis (red 10) i u nju se upisuju tri azbučna podatka uneta sa tastature (40 do 60).

Upis se vrši pojedinačno, da bi se time na kraj svakog podatka upisao separator <kraj reda>. Nakon toga upis je završen i teka se zatvara (red 70). Po unošenju simbola "P" za početak učitavanja, teka se otvara za čitanje (red 100), podaci čitaju jednom INPUT# naredbom (red 110), a zatim prikazuju na ekranu. Otvorena teka se zatvara (red 150).

P r i m e r 2: Teku formiranu u prethodnom primeru možemo proširiti na sledeći način:

```
10 OPEN 8,8,8,"TEKA1,S,A"
20 PRINT#8,"DOPUNSKI PODATAK"
30 CLOSE 8
40 OPEN 8,8,8,"TEKA1,S,R"
50 FOR I=1 TO 4
60 INPUT#8,A$
70 PRINT A$
80 NEXT I
90 CLOSE 8
```

Posle startovanja programa u teku se dopisuje novi podatak. Teka se mora zatvoriti da bi se omogućilo njeno

otvaranje za čitanje. Čitanjem se dobijaju sva tri upisana podatka.

Relativne teke podataka. Obrada jednog podatka iz sekvencijalne teke može u slučaju velikog broja podataka biti dug i složen postupak. Zbog toga se pojavljuje potreba za direktnim pristupom pojedinim podacima. Adresiranje fizičkih segmenata diskete (staza i sektora) je, kao što će u sledećem delu biti detaljno opisano, složen postupak koji je opravdan samo u izuzetnim slučajevima i za potrebe običnog korisnika nije zadovoljavajuće rešenje.

Slično nizovima koji omogućuju racionalno i jednostavno zapisivanje, čitanje i izmenu podataka u unutrašnjoj memoriji, na disketi se kao spoljašnjoj memoriji organizuje teka podataka kojima se pristupa na sličan način, preko indeksa. Ovakve teke nazivaju se **relativne teke,** jer se pozicije pojedinih podataka zadaju u odnosu na početak teke, dakle relativno (početak teke za korisnika je nepoznata veličina). Elementi relativne teke nazivaju se **slogovi,** a njihovi delovi **polja.** Polje je najmanji element koji se može adresirati, pa, u terminologiji nizova, relativna teka ima najviše dve dimenzije — slog ili slog i polje unutar njega.

Slog može imati najviše 254 simbola i svi slogovi jedne teke moraju biti iste dužine. Dužina sloga zadaje se pri formiranju teke.

Broj slogova u jednoj teci ograničen je na 720. Uzevši u obzir najveću dužinu sloga, teka bi teoretski mogla zauzimati veći prostor od kapaciteta diskete.

Podaci su u slogu organizovani sekvencijalno, tako da broj podataka zavisi samo od njihove dužine.

Veze između relativnih adresa unutar teke i stvarnih adresa na disketi čuvaju se, na poseban način, unutar teke. Njima manipulišu programi iz ROM-a disk-jedinice i za korisnika su nevidljivi.

Za vreme rada sa relativnom tekom (od otvaranja do zatvaranja) slog sa kojim se radi nalazi se u baferu disk-jedinice. Transfer podataka između bafera i diskete vrši se zadavanjem komande POSITION preko komandnog kanala. Time se postavlja i interni pokazivač na mesto u slogu na kome će otpočeti sledeća ulazno/izlazna operacija.

Korišćenje podataka se iz BASIC-programa vrši uobičajenim naredbama INPUT#, GET# i PRINT#. Zato su za rad sa ovim tipom teka neophodna dva kanala: kanal za podatke i komandni kanal 15.

Otvaranje relativne teke. Naredba OPEN u slučaju relativnih teka ima dve funkcije:

1 — formiranje nove teke,

2 — otvaranje stare teke radi izmena ili čitanja.

Oblik naredbe za formiranje teke je:

OPEN <logički broj>,8,<broj kanala>,"<ime>,L,"+CHR$(<dužina sloga>)

<dužina sloga> je ceo broj između 1 i 254. Njime se zadaje broj simbola u jednom slogu teke. Da bi se omogućio rad naredbi PRINT# i INPUT#, dužina sloga treba da predstavlja najveći broj simbola koji se u njega mogu upisati. Ovaj broj dobija se kao zbir dužina svih podataka u slogu uvećan za broj podataka. "Višak" simbola je neophodan, jer je za međusobno razdvajanje podataka unutar sloga potrebno još jedno mesto za separator (<kraj reda> ili ",") pa je neophodno rezervisati prostor i za to.

Napomenimo na ovom mestu da se podela sloga na polja nigde eksplicitno ne zadaje, već se o tome brine sam korisnik. Dužina polja mora biti izračunata na isti način kao i dužina sloga i zbir dužina polja mora odgovarati dužini sloga. Broj polja u slogu i njihova dužina **nigde se posebno ne zadaju,** pa korisnik mora oformiti svoj programski mehanizam pomoću koga će im pristupati. Redni broj prvog simbola polja u celom slogu naziva se ADRESA POLJA. Na primer, ako slog ima 50 mesta, a polja po 10, adrese svakog od njih su 1,11,21,31 i 41.

Parametar L i zarez nakon njega su obavezni i označavaju da se radi o formiranju nove relativne teke.

Pri izvršavanju naredbe OPEN najpre se proverava da li teka sa tim imenom već postoji. Ako je tako, otvara se već postojeća teka i njen sadržaj je nepromenjen. Opcija "@:" ovde nema efekta i jedini način da se relativna teka izbriše je upotreba komande SCRATCH.

Otvaranje teke radi čitanja ili unošenja izmena je sledećeg oblika:

OPEN <logički broj>,8,<broj kanala>,"<ime teke>"

Sve karakteristike teke već se nalaze u katalogu i određuju se pomoću navedenog imena.

U oba navedena slučaja teka se otvara i za čitanje i za upis. Operacije koje se mogu vršiti nakon otvaranja su čitanje, promena sadržaja i proširivanje teke.

Istovremeno se u programu ne smeju naći dve otvorene relativne teke.

Pristupanje slogu. Broj sloga kome se želi pristupiti saopštava se preko komandnog kanala komandom POSITION ili P. Ovo se mora izvršiti pre bilo koje ulazno/izlazne operacije. Oblik komande je:

"P"CHR$(<broj kanala>)CHR$(<NBS>)CHR$
(<VBS>)CHR$(<simbol>)
(<NBS>=Niži deo Broja Sloga)
(<VBS>=Viši deo Broja Sloga)

Kako argument funkcije CHR$ mora biti između 0 i 255, broj sloga se mora zadati u dva dela (jer može biti do 720). Određivanje vrednosti nižeg i višeg dela vrši se po formulama:

VBS=INT(BS/256)
NBS=BS—VBS*256

Parametar <simbol> precizira mesto simbola u slogu od koga će početi obavljanje sledeće ulazno/izlazne operacije, tj. postavlja početnu vrednost pokazivača. Najčešće se koristi za pozicioniranje pokazivača na početak nekog od polja u slogu.

Zadavanjem broja sloga većeg od ukupnog broja slogova u teci vrši se formiranje novog sloga na disketi, tj. proširivanje teke. Pri tome se formiraju i svi slogovi između poslednjeg sloga u teci i navedenog. Zato je dobro nakon naredbe OPEN kojom se formira nova teka uputiti komandu za formiranje poslednjeg sloga u teci. Ovom operacijom rezerviše se prostor za čitavu teku, čime se dobija na brzini kasnije obrade. Na primer, segmentom:

```
10 OPEN 15,8,15
20 OPEN 5,8,5,"TEKA1,L,"+CHR$(50)
30 PRINT#15,"P"CHR$(5)CHR$(0)CHR$(100)CHR$(1)
```

vrši se formiranje podataka o teci TEKA1 u katalogu, postavljanje dužine teke na 50 simbola i rezervisanje prostora na disketi za 100 slogova.

Pri pozicioniranju na slog koji ne postoji, crvena kontrolna sijalica na disk-jedinici će zatreptati. Čitanjem komandnog kanala dobija se izveštaj RECORD NOT PRESENT. U ovom slučaju nije u pitanju greška, već upozorenje. Ukoliko se radi o upisu, ono se može ignorisati, dok čitanje treba na neki način programski sprečiti.

Formirani, ali još nekorišćeni slogovi imaju na prvom mestu bajt čija je vrednost 255, dok se na svim ostalim mestima u slogu nalazi bajt čija je vrednost 0. Ova osobina se može koristiti za prepoznavanje praznih slogova i na taj način ubrzati obrada.

Čitanje i upis. Nakon upotrebe komande POSITION čitanje sloga ili polja iz sloga vrši se na uobičajeni način naredbama INPUT# i GET#. Svaka od njih uvećava broj mesta u slogu na koje ukazuje pokazivač. GET# vrši uvećanje za 1, a INPUT# za dužinu pročitanog podatka uvećanu za 1. Zato se posle provere da li slog postoji, naredbom GET#, mora ponoviti pozicioniranje pokazivača na početak.

Naredbom PRINT# vrši se upisivanje podataka u slog počev od onog mesta na koje ukazuje pokazivač. Nakon toga, pokazivač je automatski uvećan za broj upisanih simbola.

Primer 1: Postupak rada sa relativnom tekom prikazaćemo na primeru teke TELEFONI. Teka će sadržati slogove sa sledećom strukturom:

ime i prezime . . . 25 mesta
broj telefona . . . 10 mesta

Svaki podatak zahteva i posebno mesto za separator, pa je ukupna dužina sloga 25+10+2=37.

Formiraćemo relativnu teku od 100 slogova dužine 37 simbola.

```
10 OPEN 15,8,15
20 OPEN 8,8,8,"TELEFONI,L,"+CHR$(37)
30 PRINT#15,"P"CHR$(8)CHR$(100)CHR$(0)
40 INPUT#15,A,B$,C,D
50 PRINT "GRESKA";A;B$
60 PRINT#8,"KRAJ TEKE"
70 CLOSE 8:CLOSE 15
```

Posle otvaranja komandnog kanala (red 10) i relativne teke podataka (red 20), vrši se postavljanje pokazivača na stoti slog. Pri tome se formira prostor na disketi za sve slogove od 1 do 100 (red 30). Kako taj slog još ne postoji, doći će do pojave upozorenja da čitanje nema smisla. Upozorenje se učitava sa komandnog kanala i izdaje na ekran (redovi 40 i 50). Upisom teksta "KRAJ TEKE" završava se formiranje teke i završava rad programa.

Upišimo neke podatke u teku:

```
 10 OPEN 15,8,15
 20 OPEN 8,8,8,"TELEFONI"
 30 INPUT "BROJ SLOGA";S
 40 INPUT "IME I PREZIME";I$
 50 INPUT "BROJ TELEFONA";T$
 60 PRINT#15,"P"CHR$(8)CHR$(S)CHR$(0)CHR$(1)
 70 PRINT#8,I$
 80 PRINT#15,"P"CHR$(8)CHR$(S)CHR$(0)CHR$(27)
 90 PRINT#8,T$
100 INPUT "KRAJ";A$
110 IF A$<>"DA" THEN 30
120 CLOSE 8:CLOSE 15
130 STOP
```

Nakon otvaranja već postojeće relativne teke TELEFONI (red 20) unosi se broj sloga koji se želi popuniti i sadržaj za oba polja u slogu (redovi 30 do 50). Komandom "P" pokazivač se najpre postavlja na prvo polje u slogu i upisuje ime (redovi 60 i 70), a zatim na drugo polje, vodeći računa o njegovom početku. U drugo polje se upisuje broj telefona.

Ukoliko se želi kraj rada, završetak programa postiže se sa "DA", kao odgovorom na "KRAJ?".

Čitanje slogova koji nisu prazni vrši se programom:

```
 10 OPEN 15,8,15
 20 OPEN 8,8,8,"TELEFONI"
 30 FOR I=1 TO 99
 40 PRINT#15,"P"CHR$(8)CHR$(I)CHR$(0)CHR$(1)
 50 GET#8,A$
 60 IF ASC(A$)=255 THEN 120
```

```
 70 PRINT#15,"P"CHR$(8)CHR$(I)CHR$(0)CHR$(1)
 80 INPUT#8,IME$
 90 PRINT#15,"P"CHR$(8)CHR$(I)CHR$(0)CHR$(27)
100 INPUT#8,TEL$
110 PRINT I,IME$,TEL$
120 NEXT I
130 CLOSE 8:CLOSE 15
```

Nakon otvaranja teke i komandnog kanala počinje se sa pretraživanjem teke i traganjem za popunjenim slogovima. Pokazivač "P" se postavlja na prvo mesto svakog sloga i odatle naredbom GET# učitava sadržaj (redovi 40 i 50). Ukoliko je upisana vrednost 255, taj slog je neiskorišćen i traganje se nastavlja (red 60). Ako je zauzet, učitava se ime (redovi 70 i 80) i telefon (redovi 90 i 100). Po okončanju celog ciklusa, teke se zatvaraju.

Teke podataka sa direktnim pristupom. Kod prethodna dva tipa teka fizički raspored podataka na disketi određivali su sistemski programi iz ROM-a disk-jedinice. Operacije kao što su "izaberi slobodan blok" ili "označi dodelu bloka u mapi slobodnih blokova" bile su van domena korisnika i za njega nevažne. Da bi se omogućila potpuna sloboda u organizovanju teke na disketi (tj. formiranju proizvoljne strukture podataka koji se čuvaju), uveden je treći tip teke podataka — teka sa direktnim pristupom. Ovaj tip korisniku dozvoljava punu slobodu u izboru i načinu korišćenja blokova sa diskete. Naravno, time se i sva odgovornost prenosi na korisnika. Zato treba prethodno dobro upoznati strukturu zapisa na disketi i ispravno zasnovati koncepciju svoje teke.

Kako se čitava disketa može smatrati jednom tekom sa direktnim pristupom, moguće je vršiti i izmene u sistemskim delovima diskete, katalogu i mapi dodeljenih blokova. Na taj način mogu se obnoviti izbrisane teke ili zatvoriti dobro nezatvorene teke, promeniti ime diskete ili identifikacija.

Posledica direktnog pristupa je da teke ovog tipa nemaju ime i ne ulaze u sadržaj kataloga. Tako se postojanje ovakve jedne teke može utvrditi jedino na osnovu činjenice da je broj blokova koji su korišćeni od strane

ostalih teka manji od ukupnog broja zauzetih blokova. Ovi podaci mogu se dobiti čitanjem kataloga diskete.

Diskete na kojima se čuvaju teke sa direktnim pristupom ne smeju se uređivati komandom VALIDATE. Adrese blokova koje ona sadrži ne ulaze u sadržaj kataloga, pa će se pri ovoj operaciji smatrati nepotrebno zauzetim i označiti kao slobodne.

Mehanizam rada zasniva se na sledećem:

Podaci sa kojima se trenutno radi nalaze se u prihvatnoj memoriji (baferu) disk-jedinice. U nju mogu dospeti čitanjem nekog bloka sa diskete (komandom preko kanala 15) ili upisom iz računara (naredbom PRINT#). Čitanje sadržaja bafera vrši se naredbama INPUT# i GET#. Pored čitanja i upisa bloka, pojavljuje se još i dodeljivanje i oslobađanje bloka, čime se sprečava, odnosno dozvoljava njegovo korišćenje od strane neke druge teke. Za bafer postoji poseban pokazivač koji ukazuje na mesto u baferu na kome će se obaviti sledeća operacija (upis ili čitanje). Ovaj pokazivač naziva se bafer-pokazivač i njegova se vrednost automatski menja naredbama INPUT#,GET# i PRINT#. Postoji i posebna komanda za postavljanje pokazivača na proizvoljno mesto u baferu. Trenutna vrednost pokazivača se pri upisu sadržaja bafera na disketu upisuje kao prvi bajt tog bloka.

Operacije čitanja i upisa bloka, dodeljivanje bloka, oslobađanja bloka i postavljanja bafer-pokazivača obavljaju se zadavanjem posebnih komandi preko komandnog kanala 15. Zato za rad sa tekama sa direktnim pristupom moraju biti otvorene bar dve teke: teka podataka i komandni kanal.

Otvaranje teke. Oblik naredbe OPEN sada je sledeći:

OPEN <logički broj teke>,8,<broj kanala>, "#"

<logički broj teke> je ceo broj između 0 i 127.

<broj kanala> je ceo broj između 2 i 14.

Ovim je označeno da će se preko navedenog kanala vršiti direktno korišćenje diskete. Za korišćenje se dodeljuje jedan od slobodnih bafera u disk-jedinici (veličina bafera je 256 simbola).

Bafer-pokazivač se postavlja na prvo mesto u baferu.

Upis u teku. Podaci se upisuju naredbom PRINT#, čime se automatski uvećava i bafer-pokazivač. Struktui-

ranje podataka omogućeno je pomeranjem pokazivača u okviru bafera. To se ostvaruje upućivanjem komande BUFFER-POINTER preko komandnog kanala. Oblik komande je:

"B—P:"<broj kanala>;<nova pozicija>

Time se pokazivač u baferu za navedeni kanal postavlja na novu poziciju.

Pre nego se zahteva prenošenje sadržaja bafera na disketu, treba proveriti da li je za to predviđeno mesto slobodno. Koristeći komandu BLOCK—ALLOCATE:

"B—A:"0;<staza>;<sektor>

i podatke dobijene čitanjem komandnog kanala nakon toga, određuje se da li je navedeno mesto slobodno. Ukoliko je dobijen kôd greške A=65, blok je zauzet i tada se kao C i D može učitati adresa (staza i sektor) prvog sledećeg slobodnog bloka.

Kada je sadržaj bafera kompletiran, on se upisuje na disketu komandom BLOCK-WRITE:

"B-W"<broj kanala>;0;<staza>;<sektor>

na navedenu stazu i sektor.

Tekuća vrednost bafer-pokazivača upisuje se na prvo mesto u bloku.

Čitanje teke. Čitanje nekog bloka sa diskete vrši se komandom BLOCK-READ:

"B-R:"<broj kanala>;<staza>;<sektor>

Pri izvršavanju ove komande najpre se pročita zapamćena vrednost bafer-pokazivača, a zatim se izvrši prenos još toliko simbola iz bloka. Posle poslednjeg prenetog simbola u baferu se nalazi specijalni simbol za kraj teke. Pošto su podaci preneti u prihvatnu memoriju, bafer-pokazivač se postavlja na prvi simbol i podaci se mogu čitati naredbama INPUT# i GET#. Kraj podataka može se odrediti pomoću sistemske promenljive STATUS (ST), čija je vrednost u tom slučaju različita od 0.

Slično kao i kod upisa, i ovde je moguće postavljanje pozicije bafer-pokazivača, tj. neposredno čitanje bilo kog podatka iz bafera.

Disketa kao teka sa direktnim pristupom. Ukoliko se disketa smatra tekom sa direktnim pristupom, pojavljuje

se potreba za učitavanjem blokova koji nisu upisani kao delovi neke teke sa direktnim pristupom, komandom B-W (na primer blokovi koji pripadaju katalogu). Kod takvih blokova prvi bajt nema funkciju pokazivača i učitavanje sa B-R može dovesti do nepredviđenih situacija. U tom slučaju, umesto B-R se koristi komanda USER1, sličnog oblika:

"U1:"<broj kanala>;0;<staza>;<sektor>

U prihvatnu memoriju se pri tome prenosi čitav sadržaj bloka, tj. svih 256 simbola.

Za upis na disketu se u navedenom slučaju koristi naredba USER2:

"U2:"<broj kanala>;0;<staza>;<sektor>

Njome se čitav sadržaj bafera prenosi na navedeno mesto na disketi bez promene prethodno zapisane vrednosti na mestu za bafer-pokazivač.

P r i m e r 1: Traženje slobodnog bloka na disketi može se vršiti na sledeći način:

```
10 OPEN 15,8,15
20 T=18 : S=1
30 PRINT#15,"B—A:"0,T,S
40 INPUT#15,A,B$,C,D
50 IF A<>65 THEN 80
60 PRINT "STAZA";T;"SEKTOR";S;" — ZAUZET"
70 T=C : S=D
80 PRINT "STAZA";T;"SEKTOR";S;" — SLOBODAN"
90 CLOSE 15
```

Upućivanjem komande B-A preko komandnog kanala (red 30) i čitanjem greške (red 40) dobija se podatak o zauzetosti traženog bloka. Ukoliko je kod greške A=65, blok je zauzet (redovi 60 i 70) i daje se izveštaj o sledećem slobodnom bloku (red 80). Ukoliko je A<>65, blok je slobodan.

P r i m e r 2: Ime diskete može se pročitati iz kataloga, a potom zameniti novim imenom. Ovo se obavlja programom:

```
10 OPEN 15,8,15
20 OPEN 8,8,8,"#"
30 PRINT#15,"U1:"8;0;18;0
```

```
 40 PRINT#15,"B—P:"8,144
 50 FOR I=1 TO 16
 60 GET#8,A$
 70 IF A$=CHR$(160) THEN 100
 80 PRINT A$;
 90 NEXT I
100 PRINT
110 INPUT "NOVO IME";I$
120 IF LEN(I$)>=16 THEN 110
130 IF LEN(I$)<16 THEN I$=I$+CHR$(160):GOTO 130
140 PRINT#15,"B—P:"8;144
150 PRINT#8,I$;
160 PRINT#15,"U2:"8;0;18;0
170 CLOSE 8
180 PRINT#15,"I"
```

Blok sa staze 18, sektora 0, učitava se u neki od bafera dodeljen teci sa direktnim pristupom (red 30). Pokazivač se postavlja na početak zapisa imena diskete, mesto 144 u bloku (red 40). Staro ime se čita i izdaje na ekranu. Kraj imena označen je bajtom 160 (redovi 50 do 90). Zatim se unosi novo ime sa tastature i dopunjuje simbolima CHR$(160) do ukupne dužine imena od 16 znakova (redovi 110 do 130). Pokazivač se namešta na mesto na kome počinje ime i upisuje u bafer (redovi 140 i 150), a zatim se sadržaj bafera upisuje nazad na stazu 18, sektor 0 (red 160). Učitavanjem kataloga sa diskete može se videti da je ime promenjeno.

5. ZVUK I MUZIKA

5.1. UVOD

Gotovo svi mikroračunari koji se pojavljuju na tržištu od početka osamdesetih godina, poseduju sposobnost generisanja zvučnih signala. U računare se obično ugrađuju jedan do tri nezavisna generatora zvuka, odnosno hardverskih sklopova koji proizvode zvuk. Kompleksnost tih uređaja varira od računara do računara, a samim tim i mogućnosti koje pružaju.

Računari se takođe razlikuju i po obimu programske podrške koja je na raspolaganju pri operacijama sa zvukom. Negde se u sistemskom ROM-u nalazi veći broj rutina koje podržavaju rad generatora zvuka i koje korisnik upotrebljava kao i sve druge naredbe BASIC-jezika, a negde te rutine nisu prisutne, pa se generatorom mora neposredno upravljati.

Računar Commodore 64 u hardverskom pogledu spada u mikroračunare sa izuzetno dobrim generatorom zvuka, ali je u pogledu pomenute softverske podrške veoma skroman. Programiranje zvuka se ovde svodi na korišćenje POKE naredbe BASIC-jezika, kojom se ostvaruje direktno postavljanje potrebnih vrednosti u pojedine registre generatora zvuka. Pored naredbe POKE, koriste se i sve ostale naredbe BASIC-a, ali ne postoje specijalizovane koje bi izvršavale određene tipske, složenije zadatke u vezi sa programiranjem zvuka.

Ne upuštajući se u razloge zbog kojih je Commodore 64 ostavljen "siromašnim" u ovoj softverskoj nadgradnji,

treba istaći da pisanje programa koji su vezani za zvuk podrazumeva poznavanje osnovnih funkcija koje ima generator zvuka u C-64.

5.2. OSNOVNI POJMOVI U VEZI SA ZVUKOM

Zvučni talasi se predstavljaju periodičnim funkcijama. Za to su najpogodnije trigonometrijske funkcije. Razvijen je matematički aparat (Furijeova analiza) koji omogućuje da se veoma složene periodične pojave predstave preko zbira prostoperiodičnih funkcija kao što su sin(t) ili cos(t).

Pomenimo neke osnovne pojmove u vezi sa zvukom.

Prost ton nastaje kada zvučni izvor proizvodi proste oscilacije koje se mogu predstaviti jednom sinusnom ili jednom kosinusnom funkcijom.

Muzički ton nastaje kod složenijeg periodičnog kretanja zvučnog izvora. Muzički instrumenti daju ovakve tonove. Kao što je pomenuto, muzički ton se može predstaviti preko zbira prostoperiodičnih funkcija, koje predstavljaju HARMONIKE toga tona. Dakle, složeniji se tonovi mogu zamisliti kao zbir (superpozicija) prostih.

Šum je takođe složen zvuk, ali ovde ne postoji pravilnost složenog periodičnog kretanja, već se osnovni elementi zvuka (frekvencija, amplituda i faza) stalno menjaju.

Visina tona zavisi od brzine oscilovanja zvučnog izvora, ili drugim rečima od frekvencije (broja treptaja zvučnog izvora u sekundi). Viša frekvencija daje viši ton, a manja niži.

Akustička frekvencija predstavlja onaj opseg frekvencije koje je ljudsko uho u stanju da registruje. Uzima se da je to opseg od 16 Hz do 20000 Hz.

Jačina zvuka zavisi od amplitude izvora. Amplituda je najveće udaljenje sa obe strane od ravnotežnog položaja zvučnog izvora. Ako se prost ton predstavi sinusoidom, amplituda je rastojanje maksimuma sinusoide od horizontalne ose koordinatnog sistema.

Boja tona zavisi od broja harmonijskih tonova i njihove relativne jačine. Muzički instrumenti se razlikuju po boji tona.

5.3. GENERATOR ZVUKA

Generator zvuka u Commodore 64 smešten je u jedan specijalizovani čip koji nosi oznaku 6581 Sound Interface Device, ili kraće SID. Za razliku od mnogih mikroračunara koji poseduju mali zvučnik ugrađen u samom kućištu, kod C-64 zvuk proizveden u SID čipu šalje se zajedno sa video-signalom u TV i tu se reprodukuje, sa mogućnošću pojačavanja i utišavanja. Kvalitet tako dobijenog zvuka je sasvim zadovoljavajući. Postoji takođe i poseban audio, tj. zvučni izlaz iz SID čipa, koji je izveden na zadnjoj ploči računara, i koji se može priključiti na kućni muzički sistem. Time se dobija dobar kvalitet zvuka na zvučnim kutijama.

Programiranje rada generatora zvuka vrši se pomoću komandnih registara. SID čip ima 29 komandnih registara, svaki dužine 8 bitova tj. jedan bajt, preko kojih se postavljanjem odgovarajućih vrednosti upravlja svim funkcijama ovog čipa.

Ovi registri se mogu podeliti na pet grupa:

1. Grupa od sedam registara za kontrolu prvog kanala.
2. Grupa od sedam registara za kontrolu drugog kanala.
3. Grupa od sedam registara za kontrolu trećeg kanala.
4. Grupa od četiri registra za upravljanje filtrom i regulaciju jačine zvuka.
5. Grupa od četiri registra koji omogućuju programsku modulaciju signala, kao i komunikaciju sa dva potenciometra koji se mogu priključiti na SID čip.

Registri se nalaze na memorijskim adresama 54272 do 54300 i njihov tačan raspored je dat u tabeli.

Registar	Adresa	Funkcija
0	54272	K1 Frekvencija NB
1	54273	K1 Frekvencija VB
2	54274	K1 Širina pravougaonog talasa NB
3	54275	K1 Širina pravougaonog talasa VB
4	54276	K1 Kontrolni registar
5	54277	K1 Uzlazno/Silazni segment
6	54278	K1 Segment Održanje/Završetak

7	54279	K2 Frekvencija NB
8	54280	K2 Frekvencija VB
9	54281	K2 Širina pravougaonog talasa NB
10	54282	K2 Širina pravougaonog talasa VB
11	54283	K2 Kontrolni registar
12	54284	K2 Uzlazno/Silazni segment
13	54285	K2 Segment Održanje/Završetak
14	54286	K3 Frekvencija NB
15	54287	K3 Frekvencija VB
16	54288	K3 Širina pravougaonog talasa NB
17	54289	K3 Širina pravougaonog talasa VB
18	54290	K3 Kontrolni registar
19	54291	K3 Uzlazno/Silazni segment
20	54292	K3 Segment Održanje/Završetak
21	54293	Frekvencija filtra NB
22	54294	Frekvencija filtra VB
23	54295	Kontrolni registar filtra
24	54296	Jačina zvuka i tip filtra
25	54297	Potenciometar X
26	54298	Potenciometar Y
27	54299	Izlazni signal sa K3
28	54300	Izlaz obvojnica K3

U tabeli K1, K2 i K3 označavaju kanal 1,2 i 3, NB je oznaka za niži bajt, a VB za viši bajt.

Upoznaćemo se sa detaljima i načinom programiranja zvuka koristeći tipičan redosled postavljanja vrednosti u registre generatora zvuka. Ovaj redosled je sledeći:

— inicijalizacija čipa,
— postavljanje jačine zvuka,
— postavljanje frekvencije,
— određivanje tipa zvučnog signala,
— definisanje obvojnice,
— uključivanje zvuka.

Poštujući ovaj redosled i ilustrujući svaki korak odgovarajućim naredbama, izgradićemo deo po deo jedan mali program, koji će predstavljati upotrebu svih bitnih elemenata generatora zvuka.

5.4. INICIJALIZACIJA

Na početku svakog programa koji koristi SID čip treba "očistiti" sve njegove registre od eventualnih zaostalih vrednosti koje ne želimo. Radi toga je dobra praksa da se u ciklusu postave nule u sve registre čipa.

```
10 REM *** INICIJALIZACIJA ***
20 R=54272
30 FOR I=R TO R+24
40 POKE I,0
50 NEXT I
```

Ovde je ubačena početna adresa R=54272 radi toga što je pogodnije registre pamtiti po njihovom rednom broju nego direktno po adresama. Nule su postavljene samo u prvih 25 registara radi toga što se u njih mogu upisivati sadržaji, dok su preostala četiri registra predviđena samo za čitanje, i u njih se sadržaji ne mogu upisivati. Ovo ne treba da zbunjuje jer je funkcija ova poslednja četiri registra takva da je potrebno iz njih iščitavati podatke.

5.5. POSTAVLJANJE JAČINE ZVUKA

Jačina zvuka se reguliše istovremeno za sva tri kanala generatora. Registar 24 kontroliše ovu veličinu. Pri tome se koriste samo prva četiri bita (bit 0,1,2,3) registra koji je označen brojem 24, dakle registra sa adresom 54296. Preostala četiri bita ovog registra koriste se za određivanje tipa filtra i za sada nas ne interesuju.

Obzirom da se za regulisanje jačine koriste četiri bita, jasno je da postoji 16 mogućih nivoa, od 0 do 15. Nivo nula je određen sa 0000, a najviši nivo 15 sa 1111. Ako, na primer želimo da jačinu postavimo na vrednost J (između 0 i 15), i da pri tome viša četiri bita postavimo na nulu, to možemo učiniti na sledeći način:

```
55 REM *** POSTAVLJANJE JACINE ZVUKA ***
60 INPUT "UNESITE JACINU ZVUKA (0—15)";J
70 POKE R+24,J
```

5.6. POSTAVLJANJE FREKVENCIJE

Generator zvuka ima tri nezavisna kanala. Iz tabele u kojoj su prikazani svi njegovi registri, vidi se da su prve tri grupe od po sedam registara međusobno identične po funkciji, ali se razlikuju prema tome na koji se kanal odnose. U našem primeru koristićemo kanal 1, a ostala dva nećemo, jer se upravljanje kanalima 2 i 3 vrši na isti način kao i kanalom 1. Naravno, treba odmah napomenuti da je jedna od najvrednijih karakteristika SID čipa, upravo postojanje tri kanala koji omogućavaju paralelno, istovremeno i nezavisno troglasno generisanje zvuka, odnosno muzike.

Frekvencije koje muzički čip u C-64 može proizvesti leže u opsegu 0 Hz—4000 Hz. Za postavljanje frekvencije kanala 1 služe dva registra 0 i 1. Njih treba posmatrati kao prostor za jedan 16-bitni broj koji zapravo određuje frekvenciju kanala 1. Kako 16-bitni broj ima opseg od 0—65535, a čip može proizvoditi frekvencije od 0 Hz—4000 Hz, to se lako može uspostaviti veza između ova dva podatka. Ako sa F označimo frekvenciju u Hz, a sa N broj postavljen u 16-bitni registar (izražen decimalno), onda:

$$F=0.06097*N$$

Ako je, na primer, u 16-bitni registar postavljen broj sa decimalnom vrednošću 1000, onda će se dobiti frekvencija od 60.97 Hz.

Ako treba odrediti koji broj postaviti u registar da bi se dobila željena frekvencija, opet se koristi ista veza:

$$N=F/0.06097$$

Da bi se broj N koji odgovara željenoj frekvenciji postavio u registre 0 i 1 neophodno ga je raščlaniti na niži i viši bajt. Da bi se izdvojio deo koji ide u viši bajt treba broj N podeliti sa 256 i odbacujući razlomljeni deo zadržati samo ceo deo. Dakle:

$$VB=INT(N/256)$$

Da bi se izdvojio deo koji treba smestiti u niži bajt, možemo jednostavno od broja N oduzeti upravo izračunati viši bajt VB, koji najpre pomnožimo sa 256 da bi

ga pravilno "potpisali" za oduzimanje. Time se dobija vrednost koju treba upisati u niži bajt:

NB=N—VB*256

NB upisujemo u registar 0, VB u registar 1.

Dodajmo sada programu deo za postavljanje frekvencije i to unete sa tastature izražene u hercima:

```
 80 REM *** POSTAVLJANJE FREKVENCIJE ***
 90 INPUT ,,UNESITE FREKVENCIJU TONA (U HZ)";F
 95 N=F/0.06097
100 VB=INT(N/256)
110 NB=N—VB*256
120 POKE R,NB
130 POKE R+1,VB
```

5.7. ODREĐIVANJE TIPA ZVUČNOG SIGNALA

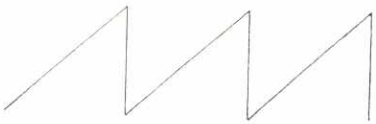
Već je rečeno da je prost ton zapravo prostoperiodično oscilovanje tipa sinusoide, a da je muzički ton složenija periodična pojava koja ima i više harmonike u svom sastavu. U SID čip ugrađena su tri oscilatora (za svaki kanal po jedan) od kojih svaki može proizvoditi tri različita osnovna periodična signala i šum kao četvrti tip zvučnog signala.

Ovi osnovni tipovi su:

— trouglasti (sl. 5.1):

Sl. 5.1 — Trouglasti signal

— testerasti (sl. 5.2):



Sl. 5.2 — Testerasti signal

— pravougaoni (sl. 5.3):

Sl. 5.3 — Pravougaoni signal

Razlikuju se prema svom harmonijskom sadržaju što im i daje različitu boju. Tako na primer, trouglasti signal koji ima relativno siromašan harmonijski sadržaj, zvuči prazno, dok testerasti signal kod koga su prisutni svi harmonici, daje puniji, bogatiji, rezak zvuk.

Kada se koristi pravougaoni signal, neophodno je zadati i njegovu širinu. To je širina onog dela u jednom ciklusu kada je signal na gornjem nivou. Širina može ići od nule do celog ciklusa, a povećanje širine, pri istoj frekvenciji, dovodi do smanjenja dela signala u kome je on na niskom nivou. Menjanjem širine pravougaonog signala, a zadržavanjem iste frekvencije, menja se harmonijski sadržaj, tj. boja muzičkog toga date frekvencije. Ovo pruža mogućnosti eksperimentisanja koje nisu na raspolaganju sa trouglastim i testerastim tipom signala.

Generisanje šuma predstavlja menjanje amplitude zvučnog signala ne prema nekom periodičnom zakonu, već prema nizu slučajnih vrednosti (beli šum). Šum se može koristiti za razne zvučne efekte.

Za određivanje tipa signala koji će SID čip proizvoditi "nadležna" su četiri viša bita kontrolnog registra svakog od nezavisnih kanala i to na sledeći način:

jedinica postavljena u bit 4 — trouglasti signal,
jedinica postavljena u bit 5 — testerasti signal,
jedinica postavljena u bit 6 — pravougaoni signal,
jedinica postavljena u bit 7 — šum.

Podrazumeva se da je jedinica postavljena u samo jedan od ova četiri bita, a da je u ostalima nula. Može se postaviti jedinica i u dva ili više navedenih bitova, ali to ne daje superponiranje, tj. zbir signala kao što bi se moglo pretpostaviti, već rezultujući signal predstavlja logičku konjunkciju (AND operaciju) signala za koje su bitovi postavljeni na jedan. Ovo daje neočekivane rezultate.

Primera radi uzmimo da treba odrediti trouglasti signal za prvi kanal (a da ostale bitove kontrolnog registra ovog kanala treba postaviti na nulu, uključujući i niža četiri bita čija se uloga opisuje kasnije). Tada treba koristiti naredbu POKE R+4,16 radi toga što se 16 zapisuje u binarnom obliku kao 00010000. Slično, za šum treba upotrebiti POKE R+4,128.

Za postavljanje širine kod pravougaonog signala na prvom kanalu koriste se registri 2 i 3. Broj koji predstavlja širinu signala je 12-bitni, pa se viša četiri bita registra 3 ne koriste. Dakle širina signala može biti zadata brojem iz opsega 0 do 4093. Na primer, za jednaku dužinu "radnog" i "neradnog" perioda treba postaviti širinu na 2048. Ovo bi se postiglo naredbama POKE R+2,0 a zatim POKE R+3,8 (za R podrazumevamo vrednost 54272).

Dodaćemo sada programu koji razvijamo deo za postavljanje tipa zvučnog signala i to pomoću dijaloga i unošenja vrednosti preko tastature.

```
140 REM *** POSTAVLJANJE TIPA ZVUCNOG
    SIGNALA ***
150 PRINT "UNESITE TIP TALASA"
160 PRINT "TROUGLASTI — 1"
170 PRINT "TESTERASTI — 2"
180 PRINT "PRAVOUGAONI — 3"
190 PRINT "SUM — 4"
200 INPUT T
210 IF T<1 OR T>4 GOTO 150
220 POKE R+4,16*2 ↑ (T—1)
230 IF T<>3 THEN 280
240 INPUT "SIRINA PRAVOUGAONOG TALASA
    (0—4093)";S
250 M=INT(S/256)
260 POKE R+3,M
270 POKE R+2,S—256*M
```

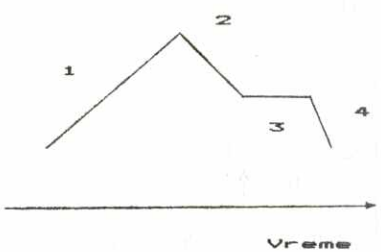
U redu 220 koristi se formula koja za unete 1,2,3 ili 4 daje brojeve 16,32,64 ili 128 redom, što postavlja bitove 4,5,6 ili 7 kontrolnog registra kanala 1. U redovima 250—270

postavlja se širina pravougaonog signala (neophodna jedino kada je tražen tip broj 3), korišćenjem razdvajanja unetog broja na niži i viši deo.

5.8. DEFINISANJE OBVOJNICE

Svaki zvuk traje određeno vreme. Za to vreme, njegovi osnovni elementi amplituda i frekvencija mogu biti konstantni. To je na primer slučaj kod tona proizvedenog na klasičnom elektronskom instrumentu organi, kada se ne koriste nikakvi dodatni efekti. Ali, što je najčešće slučaj kod muzičkih instrumenata, elementi tona variraju za vreme njegovog trajanja. Nas ovde zanima promena amplitude (ili drugim rečima jačine zvuka) u toku vremena.

Način, odnosno kriva po kojoj se menja amplituda tona za vreme njegovog trajanja može biti veoma različita. Ta kriva se naziva obvojnica. U radu na elektronskim instrumentima pokazano je da sledeći tip obvojnice dovoljno dobro reprezentuje obvojnice muzičkih tonova koje srećemo, pa čak daje i široke mogućnosti za formiranje novih originalnih tonova. Ta obvojnica je predstavljena na sl. 5.4.



Sl. 5.4 — Obvojnica

Ona se sastoji od četiri dela:

— uzlaznog dela (1), kada raste jačina zvuka u toku vremena sve dok ne dostigne maksimalni nivo (vršnu jačinu),

— silaznog dela (2), koji opisuje pad jačine zvuka sve do nekog nivoa održanja,

— dela održanja nivoa (3), koji opisuje održanje jačine zvuka tokom izvesnog vremena i

— završnog dela (4), koji dolazi na kraju opisujući postepeno nestajanje zvuka.

Muzički SID čip može nezavisno određivati svaki od pomenuta četiri elementa i to na sledeći način:

Uzlazni deo se definiše tako što se zada vreme koje treba da protekne od momenta nulte amplitude do momenta kada amplituda dostigne vršnu jačinu. Za kanal 1, rezervisana su četiri viša bita registra 5 u koje se postavlja broj koji određuje trajanje uzlaznog dela. Može se postaviti jedan od mogućih 16 različitih brojeva kojima odgovaraju trajanja data u tabeli.

Broj za uzlazni deo	Trajanje (milisekunde)
0	2
1	8
2	16
3	24
4	38
5	56
6	68
7	80
8	100
9	250
10	500
11	800
12	1000
13	3000
14	5000
15	8000

Silazni deo se definiše pomoću vremena koje treba da protekne od momenta dostizanja vršne jačine do momenta kada se prelazi na deo održanja nivoa. Broj kojim se određuje trajanje silaznog dela za kanal 1 smešta se u niža četiri bita registra 5. Odgovarajuća trajanja se vide u sledećoj tabeli.

Deo održanja nivoa se ne zadaje pomoću vremena trajanja, već pomoću određivanja nivoa, relativno u odnosu na vršnu jačinu.

Nivo održanja može biti jedna od 16 vrednosti koje ravnomerno dele interval od jačine nula do postavljene vršne jačine. Ako se, na primer, za nivo održanja uzme broj 8, dobija se jačina jednaka polovini vršne jačine. Vrednost za nivo održanja smešta se (za kanal 1) u viša četiri bita registra 6. Deo održanja nivoa započinje odmah po završetku silaznog dela, a traje sve dok se ne isključi postavljanjem nule u bit za uključivanje/isključivanje zvuka, dakle proizvoljno dugo. Taj bit se za kanal jedan nalazi na nultom (najnižem) bitu registra 4. Njegova funkcija je da postavljanjem na 1 uključi sekvencu uzlaz-silaz-održanje, koja traje sve dok se ovaj bit ne postavi na nulu i time startuje završni deo koji zapravo gasi zvuk.

Završni deo se zadaje trajanjem, a počinje, kada se postavi 0 u bit za isključivanje/uključivanje. Broj koji određuje trajanje ovog završnog dela postavlja se u niža četiri bita registra 6 (za kanal 1), može imati jednu od 16 vrednosti, a trajanja koja se time definišu data su u tabeli (koja služi i za očitavanje trajanja silaznog dela koji se postavlja nezavisno).

Broj za silazni/završni deo	Trajanje (milisekunde)
0	6
1	24
2	48
3	72
4	114
5	168
6	204
7	240
8	300
9	750
10	1500
11	2400
12	3000
13	9000
14	15000
15	24000

Nastavimo sada dograđivanje programa delom koji postavlja parametre obvojnice:

```
280 REM *** POSTAVLJANJE PARAMETARA
    OBVOJNICE ***
290 INPUT "VREDNOST ZA UZLAZNI DEO (0—15)";P1
300 INPUT "VREDNOST ZA SILAZNI DEO (0—15)";P2
310 INPUT "VREDNOST ZA NIVO ODRZANJA
    (0—15)";P3
320 INPUT "VREDNOST ZA ZAVRSNI DEO (0—15)";P4
330 POKE R+5,16*P1+P2
340 POKE R+6,16*P3+P4
```

U naredbi 330, P1 se najpre množi sa 16 da bi se vrednost pomerila na viša četiri bita, a zatim se dodaje vrednost za P2 koja će tako biti smeštena na niža četiri bita. Slično je i u naredbi 340.

Primera radi navedimo vrednosti parametara koje daju obvojnicu sličnu onoj koja je tipična za neke muzičke instrumente. Naravno da sam tip obvojnice nije dovoljan da potpuno oponaša boju tona instrumenata.

Violina:			Činele (bubanj):		
Uzlazni deo:	10	500 ms	Uzlazni deo:	0	2 ms
Silazni deo:	8	300 ms	Silazni deo:	9	750 ms
Nivo održanja:	10		Nivo održanja:	0	
Završni deo:	9	750 ms	Završni deo:	9	750 ms

Klavir:			Organa:		
Uzlazni deo:	0	2 ms	Uzlazni deo:	0	2 ms
Silazni deo:	9	750 ms	Silazni deo:	0	6 ms
Nivo održanja:	0		Nivo održanja:	15	
Završni deo:	0	6 ms	Završni deo:	0	6 ms

5.9. UKLJUČIVANJE ZVUKA

Preostalo je još samo uključivanje. Tačnije rečeno, treba pokrenuti sekvencu uzlaz-silaz-održanje. Kada (za kanal 1) postavimo jedinicu na mesto najnižeg bita, zvuk se uključuje (red 360), a kada tu postavimo nulu (red 390), nastavlja se završni deo, tj. zvuk se gubi.

Dovršimo sada program imajući na umu da se ne sme poremetiti već postavljeni sadržaj u registru 4.

```
350 REM *** UKLJUCIVANJE/ISKLJUCIVANJE
    ZVUKA ***
360 POKE R+4,16*2 ↑ (T—1)+1
370 FOR K=1 TO 400
380 NEXT K
390 POKE R+4,16*2 ↑ (T—1)
400 INPUT "ISTI TON(1), NOVI(2), KRAJ(3)";W
410 IF W=1 THEN 360
420 IF W=2 THEN 30
430 END
```

Pored toga što je poslužio kao ilustracija, ovaj program omogućuje da se variranjem parametara čuje veliki broj različitih tonova koje računar C-64 može proizvesti.

5.10. FILTAR I OSTALE MOGUĆNOSTI

Daćemo kratak opis mogućnosti SID-a u domenu filtriranja zvuka.

Filtriranje je postupak izbacivanja određenih frekvencija iz zvučnog signala.

U SID čipu postoji jedan filtar za sva tri kanala, ali se može regulisati koji će kanal ići kroz filtar, a koji direktno na izlaz.

Za upravljanje filtrom nadležni su registri 21,22,23 i 24.

U registre 21 i 22 postavlja se frekvencija odsecanja filtra. Koriste se samo niža tri bita registra 21 i ceo registar 22.

Registar 23 ima bitove sa sledećim funkcijama:

Bit 0 — kanal 1 kroz filtar ili ne
Bit 1 — kanal 2 kroz filtar ili ne
Bit 2 — kanal 3 kroz filtar ili ne
Bit 3 — spoljni audio-ulaz izlazi kroz filtar ili ne
Bit 4,5,6,7 — postavljanje intenziteta rezonanse (isticanje frekventnih komponenti na frekvenciji odsecanja).

Registar 24 ima sledeće funkcije:

Bit 0,1,2,3 — jačina zvuka (videti raniji tekst)

Bit 4 — postavljanje niskopropusnog filtra
Bit 5 — postavljanje pojasnog filtra
Bit 6 — postavljanje visokopropusnog filtra
Bit 7 — isključuje kanal 3 (korisno pri modulaciji)

Nabrojimo i preostale mogućnosti SID-a. To su kružna modulacija kanala 1 i 3, sinhronizacija kanala 1 i 3, prisutnost signala kanala 3 i obvojnice kanala 3 u posebnim registrima 27 i 28 koji se mogu čitati i tako upotrebiti bilo za amplitudnu, bilo za frekventnu modulaciju kanala 1 ili 2 (pogledati sledeći primer), zatim mogućnost da se spoljni audio-signal uvede u čip, kao i postojanje dva registra (25 i 26) u koja se smeštaju vrednosti koje direktno predstavljaju položaj dva potenciometra eventualno vezana za čip.

Navešćemo dva primera kratkih programa. Prvi ilustruje tipičan način formiranja melodije pomoću DATA naredbe, a istovremeno pokazuje kako se mogu proizvesti vibracije tona korišćenjem programskog vezivanja kanala 3 i 1.

Drugi program koristi sva tri kanala za dobijanje ritma. Upotrebljava se generator šuma kojim se oponašaju činele.

```
 10 REM *** MELODIJA ***
 20 R=54272
 30 FOR I=R TO R+24
 40 POKE I,0
 50 NEXT I
 60 REM *** KANALI 1 I 3, ISKLJUCIVANJE 3 ***
 70 POKE R+3,8:POKE R+5,41:POKE R+6,89
 80 POKE R+14,130:POKE R+18,16
 90 POKE R+24,128+15
100 REM *** UCITAVANJE I SVIRANJE ***
110 READ F,TR
120 IF F=—1 THEN END
130 POKE R+4,65
140 FOR T=1 TO TR*1.3
150 REM *** VARIRANJE FREKVENCIJE ***
160 VF=F+PEEK(R+27)*0.6
170 GF=INT(VF/256):DF=VF—GF*256
180 POKE R,DF: POKE R+1,GF
```

```
190 NEXT T
200 POKE R+4,64
210 GOTO 110
220 DATA 4817,4,5103,4,5407,4,8583,8,5407,4
230 DATA 8583,8,5407,8,8583,24,9634,4,10207,4
240 DATA 10814,4,8583,4,9634,8,10814,4,8583,4
250 DATA 9634,8,8583,24,—1,—1
```

U programskim redovima 70 i 80 postavljaju se parametri kanala 1 i 3, dok se u redu 90 isključuje izlaz sa trećeg kanala i istovremeno određuje jačina. Melodija se proizvodi na osnovu frekvencije F i trajanja TR. Frekvencija se u redu 160 varira sabiranjem sa vrednostima periodičnog trouglastog signala iz kanala 3 koji se očitava iz registra 27. Broj 0.6 predstavlja težinski faktor koji određuje koliko će ton vibrirati (menjajte vrednosti ovog faktora).

```
 10 REM *** RITAM ***
 20 R=54272
 30 FOR I= R TO R+24
 35 POKE I,0
 40 NEXT I
 45 POKE R+24,15
 50 REM *** PRVI KANAL ***
 60 POKE R,8:POKE R+1,70
 70 POKE R+5,16*0+9:POKE R+6,16*0+9
 80 REM *** DRUGI KANAL ***
 90 POKE R+7,8:POKE R+8,100
100 POKE R+12,16*0+9:POKE R+13,16*0+9
110 REM *** TRECI KANAL ***
120 POKE R+14,8:POKE R+15,220
130 POKE R+19,16*0+9:POKE R+20,16*0+9
140 REM *** UKLJUCIVANJE/ISKLJUCIVANJE ***
150 FOR I=1 TO 7
160 READ B,T
170 POKE R+4+(B—1)*7,129
180 FOR J=1 TO T:NEXT J
190 POKE R+4+(B—1)*7,128
200 NEXT I
```

```
210 RESTORE
220 GO TO 150
230 DATA 2,50,2,60,3,60,2,60,2,60,1,60,1,60
```

U ovom programu se najpre određuju parametri sva tri kanala, tako da proizvode zvuk sličan činelama različitih veličina. Ritam se formira čitanjem DATA naredbe. Učitava se B — redni broj kanala (1,2 ili 3) i T — trajanje perioda do sledećeg tona. Red 190 računa broj registra i uključuje odgovarajući kanal. Naredbe u 210 i 220 započinju ritam iz početka. Mogu se dobiti zanimljivi ritmovi promenom DATA naredbe 230.

5.11. COMMODORE 64 KAO MUZIČKI INSTRUMENT

Umesto posrednog upravljanja generatorom zvuka pomoću DATA naredbi, zanimljivo je sastaviti mali program koji omogućava da se računar pretvori u muzički instrument sa klavijaturom (tastaturom), na kome se mogu neposredno svirati melodije. Takođe se i parametri zvuka mogu staviti pod neposrednu kontrolu na bazi pritiskanja tastera. Da bi se izložio program koji ovo ilustruje, potrebno je prethodno navesti nekoliko činjenica.

Svaka muzička oktava je podeljena na 12 intervala, što daje 13 nota, odnosno 13 određenih frekvencija. Note su raspoređene tako da se frekvencija svake sledeće dobija množenjem frekvencije prethodne sa 12-tim korenom broja 2, tj. približno sa 1.06. Note oktave se označavaju redom: C,C#,D,D#E,F,F#,G,G#,A,A#,H,C. Note C,D,E,F,G,A,H označavaju osnovne tonove (bele dirke na klaviru), a C#, D#,F#,G#,A# polutonove (crne dirke na klaviru).

Svaka nota u susednoj višoj oktavi ima dvostruko veću frekvenciju od odgovarajuće note u nižoj.

Da bi se u registre SID čipa lako postavile frekvencije, navodimo tabelu sa potrebnim podacima. Množenjem sa dva datih brojeva prelazi se u susednu gornju oktavu, što se može ponoviti za sledeću, itd. Deljenjem sa dva prelazi se u susednu donju oktavu.

Nota	Frekvencija oscilatora
C	1072
C#	1136
D	1204
D#	1275
E	1351
F	1432
F#	1517
G	1607
G#	1703
A	1804
A#	1911
H	2025
C	2145

Da bi se tastatura računara pretvorila u klavijaturu, neophodno je detektovati kada je određeni taster pritisnut, ali isto tako i kada je on otpušten, da bi se ton proizvodio sve vreme dok je taster pritisnut. Radi toga ćemo, umesto GET naredbe koja ne odgovara zahtevima, programski čitati direktno memorijsku lokaciju 197. Procesor računara tokom rada pretražuje tastaturu i ispituje da li je pritisnut neki od tastera. Ako nije, u lokaciju 197 postavlja broj 64, a ako jeste u lokaciju 197 postavlja kôd pritisnutog tastera. Ti kodovi su navedeni u sledećoj tabeli.

Taster	Kôd	Taster	Kôd
A	10	W	9
B	28	X	23
C	20	Y	25
D	18	Z	12
E	14	1	56
F	21	2	59
G	26	3	8
H	29	4	11
I	33	5	16
J	34	6	19
K	37	7	24
L	42	8	27
M	36	9	32

N	39	0	35
O	38	@	46
P	41	*	49
Q	62	=	53
R	17	/	55
S	13	+	40
T	22	-	43
U	30	;	50
V	31	:	45

U programu treba tasterima dodeliti uloge dirki klavijature, i uključivati, odnosno isključivati tonove na bazi sadržaja lokacije 197.

Za tonove C,C#,D,D#,E,F,F#,G,G#,A,A#,H,C odredićemo redom tastere Z,S,X,D,C,V,G,B,H,N,J,M,<. Pored toga, može se za vreme izvršavanja programa u bilo kom momentu, pritiskom na odgovarajući taster, menjati tip zvuka i oktava. Za ovo se koriste sledeći tasteri:

1 — trougaoni tip zvučnog signala,
2 — testerasti tip zvučnog signala,
3 — pravougaoni tip zvučnog signala,
4 — šum,
5 — najniža oktava,
6 — sledeća viša oktava,
7 — sledeća viša oktava,
8 — najviša oktava.

Svi podaci potrebni programu navedeni su u DATA naredbama (kodovi tastera i frekvencije osnovne oktave).

```
 10 REM *** COMMODORE KAO MUZICKI
    INSTRUMENT ***
 20 DIM DI(13),DP(13),DO(13),VZ(4),VT(4),VP(4)
 30 REM UCITAVANJE KODOVA DIRKI KLAVIJATURE
 40 FOR I=1 TO 13
 50 READ DI(I)
 60 NEXT I
 70 REM UCITAVANJE PODATAKA ZA FREKVENCIJE
    NAJNIZE OKTAVE
 80 FOR I=1 TO 13
 90 READ DO(I)
100 NEXT I
```

```
110 REM UCITAVANJE KODOVA TASTERA ZA VRSTU
    ZVUKA
120 FOR I=1 TO 4
130 READ VZ(I)
140 NEXT I
150 REM UCITAVANJE KODOVA TASTERA ZA
    BIRANJE OKTAVE
160 FOR I=1 TO 4
170 READ VT(1)
180 NEXT I
190 REM UCITAVANJE PODATAKA ZA VRSTU
    ZVUKA
200 FOR I=1 TO 4
210 READ VP(I)
220 NEXT I
230 REM POSTAVLJANJE NAJNIZE OKTAVE ZA
    POCETNU
240 FOR I=1 TO 13
250 DP(I)=DO(I)
260 NEXT I
270 REM INICIJALIZACIJA MUZICKOG CIPA
280 ZR=54272
290 FOR I=ZR TO ZR+24
300 POKE I,0
310 NEXT I
320 REM POSTAVLJANJE SIRINE PRAVOUGAONOG
    TALASA,
325 REM OVOJNICE I JACINE ZVUKA
330 V=65
340 POKE ZR+2,10
350 POKE ZR+3,15
360 POKE ZR+5,16*4+5
370 POKE ZR+6,16*15+9
380 POKE ZR+24,15
390 REM NEKI TASTER PRITISNUT? AKO NE
    ISKLJUCIVANJE ZVUKA
400 P=PEEK(197)
410 IF P<> 64 THEN 450
420 POKE ZR+4,V-1
430 GOTO 400
440 REM DA LI JE PRITISNUTA DIRKA NA
    KLAVIJATURI?
```

```
450 FOR I=1 TO 13
460 IF P=DI(I) THEN 580
470 NEXT I
480 REM DA LI JE PRITISNUT TASTER ZA VRSTU
    ZVUKA?
490 FOR I=1 TO 4
500 IF P=VZ(I) THEN 650
510 NEXT I
520 REM DA LI JE PRITISNUT TASTER ZA BIRANJE
    OKTAVA?
530 FOR I=1 TO 4
540 IF P=VT(I) THEN 680
550 NEXT I
560 GOTO 400
570 REM POSTAVLJANJE FREKVENCIJA I VRSTE
    ZVUKA
580 A=INT(DP(I)/256)
590 B=DP(I)—A*256
600 POKE ZR,B
610 POKE ZR+1,A
620 POKE ZR+4,V
630 GOTO 400
640 REM POSTAVLJANJE VRSTE ZVUKA PREMA
    PRITISNUTOM TASTERU
650 V=VP(I)
660 GOTO 400
670 REM POSTAVLJANJE OKTAVE PREMA
    PRITISNUTOM TASTERU
680 OS=2 ↑ (I—1)
690 FOR I=1 TO 13
700 DP(I)=OS*DO(I)
710 NEXT I
720 GO TO 400
730 REM PODACI KOJI SU REDOM UCITANI NA
    POCETKU PROGRAMA
740 DATA 12,13,23,18,20,31,26,28,29,39,34,36,47
750 DATA 1072,1136,1204,1275,1351,1432
760 DATA 1517,1607,1703,1804,1911,2025,2145
770 DATA 56,59,8,11
780 DATA 16,19,24,27
790 DATA 17,33,65,129
```

Red 330 — postavljanje pravougaonog tipa signala za početni (promenljiva V određuje tip zvuka).
Red 380 — uključivanje zvuka.
Red 400 — ispitivanje lokacije 197.
Red 420 — isključivanje zvuka.
Red 460 — ispituje se da li je pritisnut taster za ton.
Red 500 — ispituje se da li je pritisnut taster za vrstu zvuka.
Red 540 — ispituje se da li je pritisnut taster za biranje oktava.
Redovi 580—610 — postavljanje frekvencije tona prema tasteru koji je pritisnut.
Red 650 — postavljanje vrste zvuka prema tasteru koji je pritisnut.
Redovi 680—710 — postavljanje oktave prema tasteru koji je pritisnut (frekvencije osnovne oktave se množe sa 2,4 ili 8).

5.12. SAŽETI PRIKAZ KARAKTERISTIKA GENERATORA ZVUKA

SID 6581 je trokanalni generator zvuka i zvučnih efekata kompatibilan sa familijom procesora 65xx. Omogućava veoma precizno upravljanje frekvencijom, bojom i dinamikom zvuka. Karakteristike čipa su sledeće:

— Tri nezavisna oscilatora
(opseg: 0 Hz—4000 Hz)
— Četiri tipa signala za svaki oscilator
(trouglasti, testerasti, pravougaoni i šum)
— Tri modulatora amplitude
(opseg: 48 dB)
— Tri generatora obvojnice
(Eksponencijalni odgovor
Trajanje uzlaznog segmenta: 2 ms—8 s
Trajanje silaznog segmenta: 6 ms—24 s
Nivo održanja: od 0 do vršne jačine
Trajanje završnog segmenta 6 ms—24 s)
— Sinhronizacija oscilatora
— Kružna modulacija
— Programabilni filtar
(Opseg frekvencije odsecanja: 30 Hz—12 kHz

Niskopropusni filtar
Visokopropusni filtar
Pojasni filtar
Promenljiva rezonansa

— Kontrola jačine tona
— Dva analogno/digitalna potenciometra
— Generator modulacije na bazi slučajnih brojeva

6. GRAFIKA

6.1. UVOD I OSNOVNI POJMOVI

Može se slobodno reći da je ono što običnog čoveka danas magnetski privlači računaru — slika na njegovom ekranu. Prethodni računari nisu bili mnogo slabiji po snazi za računanje i obradu različitih podataka od današnjih. Oni su jednostavno bili "loše obučeni" po pitanju estetike i "siromašni" u izražavanju. Slova na ekranu, magnetske kartice, bušene trake nisu bili dovoljni da opčine posmatrača, kao što ni cirkuska šatra bez bleštavog svetla i šarenih kostima nema ništa od one magične atmosfere koja je tu dok predstava teče.

Znajući sve ovo, konstruktori Commodore-64 računara opremili su ga čitavim "arsenalom oružja za atak na čulo vida". Sve to povereno je jednom nadzorniku, "video-čipu", koji oslobađa procesor svih briga oko proizvodnje slike. Slično kao procesor, ovaj čip može da pristupa pojedinim delovima memorije i iz njih uzima podatke koji opisuju sliku. Takođe, neki registri memorije direktno su vezani na njega i postavljanjem njihovih sadržaja daju mu se uputstva za rad. Pojednostavljeno, zadatak video-čipa je da neprekidno na ekranu ispisuje jednu osnovnu veliku sliku sačinjenu od 200 redova po 320 tačaka i da vodi računa o još osam malih slika (sprajtova) od 21 red po 24 tačke, nezavisnih od osnovne slike.

U radu sa ekranom postoje dva osnovna načina formiranja slike (moda), a svaki od njih se može koristiti u više varijanti. To su:

1. **Tekst-mod.** U ovom modu ekran je podeljen na 25 redova po 40 karaktera od kojih svaki sadrži po 8 puta

8=64 tačke. Svako polje može se naći u nekom od 256 različitih stanja, a svako od stanja u stvari je slika jednog simbola iz azbuke (slova, cifre, grafičkog znaka . . .). Proizvođač je u računar ugradio posebnu memoriju, karakter-ROM, koja sadrži opise dve azbuke po 256 simbola (izbor one koja će se koristiti vrši se sa COMMODORE-SHIFT), ali je ostavljena i mogućnost da korisnik sam definiše svoju azbuku i tada će računar svako polje na ekranu formirati po tom obrascu. Jasno je da je 256 ipak broj daleko manji od broja svih kombinacija upaljenih i ugašenih tačaka u jednom polju, pa tekst-mod nije u stanju da predstavi bilo kakvu sliku.

Ovaj mod ima sledeće varijante:

— standardni tekst-mod: pozadina polja je u istoj boji za sva polja, a tačke u okviru istog polja moraju biti u istoj boji (nekoj od 16);

— višebojni tekst-mod: pozadina svih polja je u istoj boji, ali u svakom polju mogu postojati tačke u tri boje, dve koje su unapred odabrane od 16 boja i zajedničke su za sva polja i jednom od 8 boja koja se dodeljuje baš tom polju. Ova mogućnost dobija se na račun dvostruko manjeg broja tačaka u svakom redu karakter-polja (sada su slova formirana od 8 redova po 4 tačke);

— tekst-mod sa proširenom bojom pozadine: ono što u prethodna dva moda nije bilo moguće, to je da se pozadina svakog karakter-polja boji posebnom bojom. Sada se to omogućuje, ali se nešto i gubi. Ovoga puta to je broj karaktera azbuke. Naime, broj karaktera se svodi na 64, a azbuka se deli na četiri grupe po 64 jednaka znaka pri čemu svaka grupa ima svoju boju pozadine. Boja svakog slova i dalje se može birati između 16 mogućih.

2. **Visokorezolucijski mod.** Mogućnost da se svaka od 64000 tačaka na ekranu upali ili ugasi nije postojala u tekst-modu. Za to je predviđen poseban, tzv. visokorezolucijski mod. U njemu se nijedna od azbuka ne može koristiti, pa ako se želi prikazati slovo ili broj, taj se simbol mora nacrtati, tačku po tačku, kao i sve ostalo. Ovaj mod ima dve varijante:

— standardni visokorezolucijski mod: broj boja u svakom nekadašnjem karakter-polju sada je sveden na dve: boju pozadine polja i boju tačaka u polju. Za svako polje moguće su sve kombinacije boja;

— višebojni visokorezolucijski mod: slično višebojnom tekst-modu, ovde se na račun broja tačaka po horizontali dobija više boja u karakter-polju. Broj tačaka je sada 160 puta 200, sva polja imaju zajedničku boju pozadine, a svako se karakter-polje može puniti tačkama u tri bilo koje boje.

Osam malih slika (sprajtova) omogućuju da ekran oživi: mogu se po njemu kretati, sudarati, nestajati i pojavljivati se, menjati boju ili oblik. One su glavno oružje u akcionim igrama i, zahvaljujući video-čipu, jednostavan alat kojim se lako rukuje. Mogu se javiti u dve varijante:

— obični sprajtovi: jednobojne slike od 21 puta 24 tačke. Svaki sprajt ima svoju boju.

— višebojni sprajtovi: slike od 21 puta 24 tačke, sačinjene iz tri boje. Dve su zajedničke za sve sprajtove, a po jedna se može dodeliti svakom pojedinačno.

Način definisanja sprajtova i manipulacije sa njima ni u čemu ne zavisi od moda i njegove varijante u kojoj se nalazi ekran. Stoga je moguće njihovo uzajamno kombinovanje.

6.2. OSNOVNI ELEMENTI ZA FORMIRANJE SLIKE

Da bi obavio svoj zadatak, video-čip mora voditi računa o velikom broju elemenata za formiranje slike, čitavih 64000 tačaka u visokorezolucijskom modu ili 1000 karaktera i njihovih opisa u tekst-modu. Uz to treba dodati i 8 sprajtova o kojima se posebno brine. Od njega se zahteva da sa svim ovim elementima manipuliše što je moguće brže, pa je stoga neophodno da ima direktan pristup memoriji u kojoj se ovi elementi nalaze. Sa druge strane, korisnik određuje kako će slika izgledati, pa je važno da ima lak pristup u te zone. Kao rešenje ovog problema nameće se uvođenje memorije kojoj će, istovremeno i nezavisno, jedan od drugog moći da pristupaju i računar i korisnik. Video-čip je u stanju da adresira ukupno 16 kB memorije, pa se stoga ukupnih 64 kB RAM može podeliti na četiri odvojene grupe po 16 kB. Bilo koja od njih može biti odabrana kao zajednička. Postupak za to je sledeći:

Na adresi 56578 postaviti dva najniža bita na 11:

POKE 56578,PEEK(56578)OR3

Ovim je omogućen sledeći korak, određivanje zajedničke memorije, o čemu brinu dva najniža bita iz registra 56576:

POKE 56576,(PEEK(56576)AND252)ORA

pri čemu je A neka od vrednosti:

A dekadno	A binarno	Zajednička memorija je:
0	0 0	od 49152 do 65535
1	0 1	od 32768 do 49151
2	1 0	od 16384 do 32767
3	1 1	od 0 do 16383

Ovaj zajednički deo memorije naziva se VIDEO-BLOK. U njemu se obavezno moraju nalaziti sledeći elementi:

— ekranska memorija,
— karakter-memorija,
— opisi svih korišćenih sprajtova.

Nakon uključenja računara video-blok se u memoriju postavlja na adrese od 0 do 16383 i da bi se izlaganje pojednostavilo, dalje će se uvek smatrati da se radi o ovom memorijskom delu. U ovom delu prave memorijske adrese (kojima korisnik jedino i može da pristupi) jednake su sa adresama unutar video-bloka. Treba voditi računa da ako se menja mesto ovog bloka, obavezno treba preračunati adrese navedenih elemenata. Njihove stvarne adrese dobijaju se tako što se na adrese unutar video-bloka dodaje njegova početna adresa.

Za rad video-čipa neposredno su vezane sledeće grupe elemenata:

— kontrolni elementi,
— elementi za opis slike.

1) **Kontrolni elementi** služe da se pomoću njih zadaju opšti podaci za rad video-čipa. Pomoću njih se bira način formiranja slike (tekst ili visokorezolucijski mod), ukazuje se u kom se delu video-bloka nalaze elementi za opis slike, određuje boja okvira slike i boje zajedničke za ceo ekran, vrši kompletna kontrola položaja, veličine, boje i uzajamnih odnosa sprajtova i sl. Ovi elementi nalaze se u tzv. ulazno-izlaznoj memorijskoj zoni i imaju

fiksne adrese. Uz svaki grafički mod biće date i adresa i funkcija korišćenih kontrolnih registara.

2) **Elementi za opis slike — ekranska memorija** — U tekst-modu svakom od 1000 slovnih mesta na ekranu odgovara jedan registar u memoriji. U njega se smešta redni broj (kôd) slova iz azbuke koje će se na odgovarajućem mestu prikazati. Skup svih tih registara čini ekransku memoriju od 1000 uzastopnih registara. Ekran se u memoriju preslikava po redovima, sleva nadesno, i odozgo nadole. Na prvo mesto u ekranskoj memoriji dolazi prvi levi karakter gornjeg reda, na drugo drugi u istom redu, na četrdeset prvo prvi karakter sleva u drugom redu itd.

Mesto ekranske memorije u video-bloku može se menjati izmenom kontrolnog registra na adresi 53272. Za to su zadužena gornja četiri bita ovog registra, dok se donja četiri ne smeju menjati jer sadrže početnu adresu posebne karakter-memorije. Zato se određivanje početne adrese ekranske memorije u video-bloku obavlja na sledeći način:

```
POKE 53272, (PEEK(53272) AND 15) OR A*16
```

Vrednost za A može biti između 0 i 15. Funkcija je sledeća:

A	Sadržaj registra 53272	Početna adresa ekranske mem. u video-bloku
0	0000xxxx	0
1	0001xxxx	1024
2	0010xxxx	2048
3	0011xxxx	3072
4	0100xxxx	4096
5	0101xxxx	5120
6	0110xxxx	6144
7	0111xxxx	7168
8	1000xxxx	8192
9	1001xxxx	9216
10	1010xxxx	10240
11	1011xxxx	11264
12	1100xxxx	12288
13	1101xxxx	13312
14	1110xxxx	14336
15	1111xxxx	15360

Na osnovu ovoga video-čip će uzimati podatke o izgledu ekrana sa novog mesta u memoriji. Međutim, operativni sistem, tačnije ekranski editor, koji zadaje sadržaj ekranske memorije (prikazuje sve što se otkuca na tastaturi, pomera kursor...) o ovoj promeni još nije obavešten i još uvek barata sa starim mestom. Ako se uradi prethodna POKE naredba za bilo koje A veće od 2, dobiće se na ekranu neki sadržaj koji se neće menjati ma šta se kucalo na tastaturi. Zato se mora izvršiti još jedna operacija: u registar 648 treba upisati broj memorijske strane na kojoj počinje ekranska memorija. Ovaj broj dobija se tako što se stvarna adresa početka ekranske memorije (dobijena sabiranjem početne adrese video-bloka i adrese ekranske memorije unutar njega) podeli sa 256 (svaka strana ima 256 registara). Tek tada se komunikacija sa računarom obavlja preko nove ekranske memorije.

U visokorezolucijskom modu ekranska memorija se koristi za zapis boje tačke i boje pozadine u svakom od karakter-polja.

Nakon uključenja računara, ekranska memorija se nalazi na adresama od 1024 do 2023.

— **Kolor-memorija** — služi za zapis podataka vezanih za prikaz boja na ekranu. Sastoji se od 1000 registara koji su organizovani na isti način kao i ekranska memorija, ali imaju stalno mesto u memoriji, na adresama od 55296 do 56295. U svakom registru koriste se samo donja četiri bita, jer je toliko dovoljno za zapis bilo koje od 16 mogućih boja. U različitim grafičkim modovima koristi se na različite načine.

— **Karakter-memorija** — Ova memorija sadrži opis svakog simbola iz azbuke od 256 različitih znakova. Kako je već rečeno, u posebnoj ROM (karakter-ROM) računara upisane su dve azbuke od po 256 znakova. Kako je svaki od simbola sačinjen od 8 redova po 8 tačaka, za njegov zapis u memoriji potrebno je 8 registara. Pri tome svaki registar opisuje jedan horizontalni red.

Računajući ukupnu memoriju potrebnu za zapis cele azbuke dobija se 256 znakova * 8 registara = 2048 registara (2 kB). Dakle, za zapis jedne azbuke potrebna su 2 kilobajta, pa karakter-ROM ima ukupno 4 kilobajta. Za

korisnika je u toku rada ova memorija nevidljiva, jer se njome koristi jedino video-čip.

Važno je napomenuti da je korišćenje karakter-ROM-a moguće jedino kada se video-blok nalazi između 0 i 16383 ili između 32768 i 49151. Tada video-čip smatra da se na lokacijama video-bloka od 4096 do 8191 umesto stvarne RAM nalazi karakter-ROM. Obraćajući se tim adresama video-bloka, video-čip čita opise karaktera iz ROM, tako da ovaj proces nema nikakve veze sa sadržajem koji se za nas nalazi na tim adresama. To može biti BASIC-program, kao i bilo koji drugi sadržaj.

U preostala dva dela memorije, između 16384 i 32767 i između 49152 i 65535, video-čip nije u stanju da koristi karakter-ROM i ako se njima želi koristiti, korisnik mora sam definisati opise simbola svoje azbuke. Definisanje nove azbuke moguće je bez obzira na mesto video-bloka u memoriji, ali ako to mesto dopušta korišćenje karakter--ROM-a, nova azbuka se ne može opisati na mestu koje se smatra ROM-memorijom. Tom opisu video-čip ne bi mogao pristupiti, pa ni ti novi simboli ne bi mogli biti prikazani. Umesto njih uporno bi se prikazivali simboli iz ROM-a.

Kako video-blok ima 16 kB, u njemu postoji 8 grupa po 2 kB, pa za smeštaj opisa jedne azbuke ima 8 mogućnosti. U kojoj se grupi on nalazi određuju tri bita (jedan do tri) kontrolnog registra 53272. Mogu se postaviti sledećom naredbom:

POKE 53272,(PEEK(53272)AND241)OR A*2

gde su vrednosti za A između 0 i 7. Značenje je sledeće:

Vrednost A	Vrednost 53272	Karakter-memorija
0	xxxx000x	0 — 2047
1	xxxx001x	2048 — 4095
2	xxxx010x	4096 — 6143
3	xxxx011x	6144 — 8191
4	xxxx100x	8192 — 10239
5	xxxx101x	10240 — 12287
6	xxxx110x	12288 — 14335
7	xxxx111x	14336 — 16383

Ukoliko korisnik ne koristi svoju azbuku, vrednost za A može biti 2 (ako koristi azbuku velika slova/grafički znaci) ili 3 (ako koristi azbuku velika/mala slova).

Mada je rečeno da je karakter ROM za korisnika praktično nevidljiva, može se, ako se to želi, omogućiti njeno čitanje ukoliko se isključi grupa ulazno-izlaznih registara koji se nalazi u RAM i umesto nje uključi karakter-ROM. Tada se počev od adrese 53248 do 57353 nalaze opisi karaktera obe azbuke, ali je svaka komunikacija računara sa spoljašnjom sredinom onemogućena. Zato treba potrebne delove prepisati u neki drugi deo memorije, ponovo postaviti ulazno/izlazne registre na njihovo mesto i omogućiti dalji rad. Na ovome se zasniva postupak zamene simbola iz postojeće azbuke nekim drugim, uvedenim simbolima.

6.3. PRIKAZIVANJE TEKSTA

Odmah po uključenju računar automatski dolazi u tzv. standardni tekst-mod. To znači da je ekran pripremljen za prikaz teksta korišćenjem 25 redova sa po 40 mesta u svakom redu. Formiranje slike obavlja se na sledeći način:

Za svako od 1000 karakter-polja, iz ekranske memorije se uzima kôd karaktera koji će u njemu biti prikazan. Na osnovu njega određuje se mesto u karakter-ROM-u sa koga će se uzeti opis tog simbola. Za jedan karakter, opis zahteva 8 registara, pri čemu svaki registar predstavlja jedan horizontalni red a svaka binarna jedinica u njemu svetlu tačku na odgovarajućem mestu karakter-polja.

Na primer, slovo A opisano je na sledeći način:

Binarni zapis		Dekadni zapis		Izgled na ekranu
00011000		24		...**...
00111100		60		..****..
01100110		102		.**..**.
01111110		126		.******.
01100110	<=>	102	<=>	.**..**.
01100110		102		.**..**.
01100110		102		.**..**.
00000000		0		....

Tačke koje su opisane sa 0 biće prikazane u boji koja je zapisana u registru za boju pozadine, na adresi 53281, a tačke opisane sa 1 bojom zapisanom u registru kolor-memorije koji odgovara tom karakter-polju. Dakle, boja pozadine je zajednička za ceo ekran, a boju tačaka u svakom polju možete odrediti sami. Boja okvira ekrana određena je sadržajem registra na adresi 53280.

Na primer, da bi se u gornjem levom uglu ekrana dobilo slovo "A" napisano belom bojom, treba izvršiti sledeće naredbe:

```
POKE 1024,1 : POKE 55296,1
```

Da bi se ekran i njegov okvir slili u površinu iste boje, treba upisati u registre 53280 i 53281 isti kôd boje (na primer 0 za crnu boju):

```
POKE 53280,0 : POKE 53281,0
```

Ekranski kodovi kao i kodovi za boje dati su u prilogu svakog priručnika koji se dobija uz C-64. **Ne treba mešati ekranske i ASCII kodove simbola! ASCII** kodovi nemaju nikakve direktne veze sa prikazom teksta na ekranu.

U standardnom tekst-modu za izgled ekrana važni su sledeći registri:

— pozadinska boja	registar 53281
— boja okvira	registar 53280
— ekranska memorija	1024 — 2023
— kolor-memorija	55296 — 56295

P r i m e r 1: Brisanje ekrana sa { clr } ekvivalentno je brisanju svih registara ekranske memorije.

```
10 FOR I=1024 TO 2023
20 POKE I,32
30 NEXT I
```

U svih 1000 registara ekranske memorije postavljaju se na ekranski kôd za prazno polje.

P r i m e r 2: Obojimo slova u prvih šesnaest redova ekrana različitim bojama.

```
 5 PRINT {clr}
10 FOR I=0 TO 15
15 PRINT "BOJA";I
```

```
20 FOR J=0 TO 39
30 POKE 40*I+J+55296,I
40 NEXT J
50 NEXT I
```

Nakon brisanja ekrana (red 5) počinje ciklus po I (red 10). Svakom od 16 prolaza odgovara jedna boja simbola koja se nalaze u tom redu. Upis sadržaja reda vrši se u redu 15. Ciklus po J (red 30) upisuje boju određenu sa I na sva mesta u redu.

Višebojni tekst-mod. Ovaj mod omogućuje prikazivanje karaktera koji u sebi sadrže do četiri boje, od kojih je jedna boja pozadine ekrana, a dve boje su unapred određene za korišćenje u svim karakter-poljima. Dakle, za jedno polje individualna je i dalje samo jedna boja.

Ispisivanje ekrana vrši se sada nešto drugačije. Iz ekranske memorije uzima se kôd karaktera, na osnovu njega nalazi se blok od 8 registara koji ga opisuje. Međutim, sadržaj ovih registara sada ima drugačije značenje. Svaki od njih deli se na četiri grupe po dva bita. Svaka grupa određuje kojom će bojom na ekranu biti prikazana jedna tačka koja po širini obuhvata dve tačke iz standardnog moda (ili za svaku ovu grupu pišu se dve tačke u boji koju određuje vrednost te grupe). Mogućim kombinacijama odgovaraju sledeće boje:

Sadržaj grupe	Adresa boje u memoriji
0 0	53281
0 1	53282
1 0	53283
1 1	kolor-memorija

Za razliku od drugih grafičkih modova koji rade za ceo ekran, ovaj mod dozvoljava da na opisani način budu prikazani samo neki simboli, dok se ostali mogu prikazivati kao u standardnom modu (8 puta 8 samostalnih tačaka opisuju simbol, nema grupa niti više od dve boje). Kao indikator za način prikaza za pojedina mesta na ekranu služi treći bit njemu odgovarajućeg registra kolor-memorije. Ukoliko je njegova vrednost 0, biće prikazan

u standardnom modu, a ukoliko mu je vrednost 1 biće prikazan u višebojnom modu. Na taj način skup od 16 boja podeljen je na dve grupe od po 8, jednu koja se koristi u standardnom modu (boje sa kôdom 0 do 7) i drugu koja se koristi u višebojnom modu (boje sa kôdom od 8 do 15).

Slovo "A" će sada biti prikazano na sledeći način:

Binarni zapis	Izgled na ekranu
00011000	. . XXYY . .
00111100	. . ZZZZ . .
01100110	XXYYXXYY
01111110	XXZZZZYY
01100110	XXYYXXYY
01100110	XXYYXXYY
01100110	XXYYXXYY
00000000	. . . . .

(. = boja iz registra 53281)
(X = boja iz registra 53282)
(Y = boja iz registra 53283)
(Z = boja iz kolor-memorije)

Kao prekidač za uključenje ovog moda koristi se bit 4 u registru 53270.

Postupak korišćenja ovog moda može se podeliti u dve faze:

1) Priprema zajedničkih boja
POKE 53281,B00
POKE 53282,B01
POKE 53283,B10

(B00 — boja za grupu 00)
(B01 — boja za grupu 01)
(B10 — boja za grupu 10)

2) Generalno uključenje višebojnog karakter-moda
POKE 53270,PEEK(53270)OR16
3) Formiranje slike
— prikaz karaktera u višebojnom modu:
POKE AEM,EKK
POKE AKM,B11
(AEM — adresa ekranske memorije)
(EKK — ekranski kôd karaktera)

(AKM — adresa kolor-memorije)
B11 — boja za grupu 11, mora biti između 8 i 15)
— prikaz karaktera u standardnom modu:
POKE AEM,EKK
POKE AKM,BOJA
(BOJA — boja karaktera, mora biti između 0 i 7)

Važni registri su:

Uključenje moda	registar 53270 bit 4
Ekranska memorija	1024 do 2023
Kolor-memorija	55296 do 56295
Pozadinska boja 00	registar 53281
Pozadinska boja 01	registar 53282
Pozadinska boja 10	registar 53283

Primer za višebojni tekst-mod:

U prvom redu ekrana biće ispisan niz simbola "1234567890ASDFG", belom bojom. Nakon toga, uključuje se višebojni tekst-mod i prvih deset simbola se definišu kao višebojni.

```
 10 PRINT {CLR};
 20 PRINT "(CTR1/2)1234567890ASDFG"
 30 POKE 53282,0 : REM BOJA ZA KOD 01
 40 POKE 53283,7 : REM BOJA ZA KOD 10
 50 POKE 53270,PEEK(53270) OR 16
 60 PRINT "VISEBOJNI TEKST-MOD JE UKLJUCEN."
 70 PRINT "KOD BELE BOJE JE 1, PA SE TEKST"
 80 PRINT "I DALJE PRIKAZUJE OBICNO"
 90 PRINT "UNESITE KRAJ REDA": INPUT A$
100 FOR I=0 TO 9
110 POKE 55296+I,9 : REM BOJA ZA KOD 11
120 NEXT I
130 PRINT "PRVIH DESET SIMBOLA SU VISEBOJNI"
```

Nakon brisanja ekrana i upisa teksta belom bojom (redovi 10,20), postavljaju se boje za kodove 01 i 10 u za to predviđene registre (redovi 30,40). Nakon uključivanja višebojnog moda izgled teksta na ekranu se ne menja, jer odgovarajuća mesta u kolor-memoriji sadrže kôd bele boje 1, pa je četvrti bit svakog od njih 0. Zato višebojni mod za njih ne važi. Postavljanjem boje sa kodom 9 na prvih deset mesta kolor-memorije (redovi 100 do 120) tih

deset simbola se označavaju kao višebojni i prikazuju na način opisan u tekstu.

Tekst-mod sa proširenom bojom pozadine. Ukoliko je potrebno da pozadine karaktera budu u boji različitoj od pozadinske boje ekrana, koriste se mogućnosti ovog moda. On dopušta da se unapred odrede četiri boje koje će biti moguće koristiti kao pozadine za svako karakter-polje. Na primer, može se na plavom ekranu prikazati žuto polje sa crnim simbolom u njemu. Pozadina ekrana dobija četiri boje, a po njoj se piše sa svih 16 boja. Međutim, mora se uvek misliti i na to šta se gubi. Pogledajmo zato kako se formira slika u ovom modu:

Za svako polje se iz ekranske memorije uzima kôd karaktera koji će se prikazati. **Iz ovog koda izdvajaju se dva najviša bita i na osnovu njih određuje boja pozadine tog polja.** Grupa od šest nižih mesta određuje simbol koji će se prikazivati i to onom bojom koja je zapisana u odgovarajućem registru kolor-memorije.

Dakle, simbol iz azbuke se više ne kodira sa 8 već sa 6 bita, pa azbuka nema 256 već 64 karaktera. Ovo može da se kaže i drugačije: Od jedne azbuke čijih svih 256 simbola imaju istu boju pozadine, dobijaju se četiri azbuke sa istim simbolima, ali različitom bojom pozadine. Svaka od njih sadrži samo 64 prva simbola standardne azbuke.

Boje pozadine se određuju na sledeći način:

Registar ekranske memorije	Boja pozadine
00xxxxxx	registar 53281
01xxxxxx	registar 53282
10xxxxxx	registar 53283
11xxxxxx	registar 53284

Višebojni tekst-mod se uključuje postavljanjem bita 6 u registru 53265 na 1:

```
POKE 53265,PEEK(53265)OR64
```

Isključuje se sa:

```
POKE 53265,PEEK(53265)AND191
```

Važni registri su:

Uključivanje moda	registar 53265 bit 6
Ekranska memorija	1024 do 2023
Kolor-memorija	55296 do 56295

Pozadinska boja 00	registar 53281
Pozadinska boja 01	registar 53282
Pozadinska boja 10	registar 53283
Pozadinska boja 11	registar 53284

Primer za tekst-mod sa proširenom bojom pozadine: U ovom primeru redovi će na ekranu biti obojeni u četiri različite boje.

```
 10 PRINT{clr}
 20 POKE 53282,5 :REM ZELENA
 30 POKE 53283,7 :REM ZUTA
 40 POKE 53284,0 :REM CRNA
 50 :
 60 POKE 53265,PEEK(53265) OR 64
 70 FOR I=0 TO 5
 80 FOR J=0 TO 3
 90 FOR K=0 TO 39
100 POKE 1024+(I*4+J)*40+K,32+J*64
110 NEXT K
120 NEXT J
130 NEXT I
140 WAIT 198,1
150 POKE 53265,PEEK(53265) AND 191
```

Nakon brisanja ekrana (red 10) postavljaju se boje za pozadinu polja. Osnovna boja ekrana se ne menja (redovi 20 do 40). Uključenje tekst-moda sa proširenom bojom pozadine vrši naredba POKE u redu 60. Naredna tri ciklusa uložena jedan u drugi (redovi 70 do 130) postavljaju u ekransku memoriju blanko-simbole sa različito postavljenom kombinacijom bita 6 i 7. Prvi, peti, deveti ... red će imati 00. Drugi, šesti, deseti ... će imati 01 i slično za ostale. Nakon unetog bilo kog simbola sa tastature (red 140) vrši se povratak u standardni tekst-mod (red 150). Zapazite razliku koja je pri tome nastala.

6.4. KORISNIČKI DEFINISANI KARAKTERI

Često se pri radu ukazuje potreba za simbolima koji ne postoje u standardnim azbukama koje C-64 poseduje, na primer grčkim ili ćiriličnim slovima, indeksima ili po-

sebnim grafičkim znacima. Taj se problem može rešiti tako da neka slova azbuke, a ako želite i celu azbuku, predefinišete u simbole koje želite, potpuno slobodno birajući kombinacije svih 64 tačaka kojima se simbol opisuje. Simboli koje na taj način napravite mogu se koristiti u svim varijantama text-moda.

* * *

Prvo što treba uraditi to je — oformiti na papiru izgled novih simbola. U polju od 8 puta 8 tačaka treba pojačati one koje će biti vidljive i tako formirati oblik simbola. Neka simbol ima sledeći oblik:

```
********
*......*
*.****.*
*.*..*.*
*.*..*.*
*.****.*
*......*
********
```

Potom sledi postupak suprotan onom koji obavlja video-čip: treba od gotovog simbola napraviti njegov opis. Svaki red slike smatra se jednim registrom, pojačane tačke u njemu jedinicama a ostale nulama. Pretvaranjem binarnih vrednosti zapisanih u ovim registrima u dekadne brojeve dobija se:

```
11111111 = 255
10000001 = 129
10111101 = 189
10100101 = 165
10100101 = 165
10111101 = 189
10000001 = 129
11111111 = 255
```

Ovih osam brojeva predstavljaće opis novog simbola.

Sledeći korak je smestiti ovaj opis negde u memoriju, tako da bude ravnopravan sa opisima iz karakter-ROM-a.

Ovde nastaju problemi. Novi opis se ne može upisati u ROM, jer ova memorija služi samo za čitanje. Sa druge strane, video-čip nije u stanju da uzima opise sa dva različita mesta. Jasno je da sve dok video-čip uzima opise simbola iz ROM, problem nema rešenja. Zato treba uraditi sledeće:

Treba omogućiti da na neki način direktno pristupite karakter-ROM i da iz nje prepišete u RAM opise onih koji će biti zadržani u novoj azbuci. Umesto opisa onih koji se neće koristiti, upisaće se opisi novih simbola, a potom proglasiti za zonu RAM za karakter-memoriju. Tako će video-čip prihvatiti nove simbole kao deo azbuke i ispisivati ih kad god se to bude zahtevalo.

Postupak se u praksi sprovodi na sledeći način:

Prva faza: Ova faza sastoji se iz dva dela: pripreme za prepisivanje i samog prepisivanja. Zašto je potrebna bilo kakva priprema? Kao što je poznato, C-64 je dobio ime po 64 kB RAM koju sadrži u sebi. Pored toga, on sadrži još i 20kB ROM. Kako je procesor u stanju da u jednom trenutku komunicira sa memorijom od najviše 64kB (»adresni prostor« mu je toliki), on ne može biti direktno fizički povezan i sa ROM i sa RAM. Stoga se za neke adresne zone (od ukupnih 64kB) uvode preklopnici (skretnice) kojima se određuje šta će biti »prikačeno« na te adrese, ROM ili RAM. Dakle, na istim adresama može se naći različit sadržaj, zavisno od položaja preklopnika. Istovremeno se sa oba ne može raditi. Procesor ne koristi karakter-ROM u svom radu, jer je oslobođen svih briga oko prikazivanja slike (njih je prebacio na video-čip). Da bi se omogućio pristup sadržaju ROM-a, treba isključiti RAM-zonu koja se nalazi na adresama 53248 do 57343 i sadrži ulazno/izlazne (U/I) registre, čime će se na tom mestu uključiti karakter-ROM. Od tog časa, procesor nije u stanju da ostvari bilo kakvu vezu sa spoljašnjim svetom, na primer tastaturom. Ovo je zato vrlo opasan trenutak i ako se neko uspostavljanje veze od njega bude zahtevalo, procesor će se na uobičajeni način obratiti U/I registrima. A ako se to dogodi, obično se mora isključiti računar. Zato je za pripremu čitanja karakter-ROM-a neophodno obaviti dve operacije:

— Onemogućiti da se spolja procesoru postavljaju bilo kakvi zahtevi. Od tog časa on izvršava samo zadati program

iz memorije i ništa ga ne može prekidati (tzv. »sistem prekida« je isključen).

— Zameniti U/I registre karakter-ROM-om.

Nakon toga treba obaviti prepisivanje iz memorijske zone od 53248 do 57343 u neki deo RAM koji ćete kasnije proglasiti za karakter-memoriju. Početna adresa ovog dela mora biti umnožak od 2kB (2048 bajta), jer se video-čipu može ukazati jedino na takve adrese. U opštem slučaju, može se izabrati bilo koji deo RAM-memorije koji se ne koristi od strane operativnog sistema, ali je to najjednostavnije učiniti u prvih 16kB (proces obaveštavanja video-čipa o nastaloj promeni je ovde najprostiji): Kako se u ovoj zoni nalazi i BASIC-program, treba izabrati njene najviše delove, počev od 12kB=12288 ako se žele koristiti obe azbuke, ili od 14kB=14336 ako se želi koristiti samo jedna. Da ne bi došlo do preklapanja, sistemske lokacije koje ukazuju na kraj zone za BASIC-program treba postaviti ispod izabrane zone. Kada je prepisivanje završeno, ROM se zamenjuje U/I registrima i dozvoljava se obraćanje procesoru.

Prilikom prepisivanja korisno je znati tabelu sadržaja ROM-a:

Adrese	Sadržaj
53248 — 53759	velika slova
53760 — 54271	grafički simboli
54272 — 54783	inverzna velika slova
54784 — 55295	inverzni grafički simboli
55296 — 55807	mala slova
55808 — 56319	velika slova i graf. simboli
56320 — 56831	inverzna mala slova
56382 — 57343	inverzna velika slova i graf. sim.

Druga faza: U RAM se upisuju opisi novih simbola. Početna adresa od koje će se vršiti upisivanje računa se na sledeći način:

PADR = 8*EKS + PAKM

(PADR — tražena početna adresa za upis novog simbola)

(EKS — ekranski kôd koji smo dodelili novom simbolu)

(PAKM — početna adresa karakter-memorije)

Kada su na ovaj način upisani opisi svih novih simbola, ostaje da se ta zona proglasi za karakter-memoriju. To se postiže tako što se u registar 53272, u deo od prvog do trećeg bita upiše broj izabrane memorijske grupe od po 2kB. Na tri mesta može se upisati najviše 8 različitih brojeva, pa se time pokriva samo prvih 16kB. Za korišćenje drugih delova memorije morale bi se učiniti još neke predradnje.

P r i m e r: Pretpostavimo da želimo da sačuvamo obe azbuke iz ROM-a, ali da umesto znaka »@« želimo da koristimo simbol koji je opisan na početku. Njegov opis daćemo u okviru DATA naredbe:

```
5 DATA 255,129,189,165,165,189,129,255
```

Pripremimo sada prepisivanje. Da bi onemogućili prekide, postavimo nulti bit registra 56334 na 0:

```
10 POKE 56334,PEEK(56334)AND254
```

Potom ćemo izvršiti zamenu U/I registara karakter-ROM-om. Za to treba postaviti drugi bit u registru 1 na 0:

```
20 POKE 1,PEEK(1)AND251
```

Prepišimo sada ROM u RAM počev od adrese 12288:

```
30 FOR I=0 TO 4095
40 POKE 12288+I,PEEK(53248+I)
50 NEXT I
```

Vratimo U/I registre:

```
60 POKE 1,PEEK(1)OR4
```

Uključimo komunikaciju sa procesorom:

```
70 POKE 56334,PEEK(56334)OR1
```

Kako je ekranski kôd za »@« jednak 0, na osam mesta počev od 12288 i 14336 (prva mesta u opisu prvog simbola svake azbuke) treba upisati opis novog simbola:

```
 80 FOR I=0 TO 7
 90 READ X
100 POKE 12288+I,X
110 POKE 14336+I,X
120 NEXT I
```

Postavimo pokazivač za BASIC-memoriju:

```
130 POKE 52,48 : POKE 56,48
```

Postavimo adresu karakter-memorije na 12288:

```
140 POKE 53272, (PEEK(53272)AND241)OR12
```

Nakon što izvršite ovaj program, video-čip će dalje uzimati opise simbola iz RAM i kada pritisnete taster za znak »@« na ekranu će se umesto njega pojaviti novi simbol. Ovo važi sve dok ne pritisnete RUNstop/RESTORE, nakon čega se prikaz ponovo vrši na osnovu karakter--ROM-a.

6.5. GRAFIKA VISOKE REZOLUCIJE

U »ličnoj karti« bilo kog računara uvek se nalaze podaci o rezoluciji. Oni govore o tome koliko se najviše tačaka može prikazati po širini i po visini ekrana. Pomnožena, ova dva broja daju broj tačaka koje se mogu prikazati na ekranu. COMMODORE-64 ima 320 tačaka po širini i 200 tačaka po visini, pa je, dakle, 320 * 200 = 64000 broj tačaka kojima se formira slika.

U tekst-modu kontrola osvetljenosti i boje ovih tačaka obavlja se u skladu sa oblikom i bojom slova u tekstu koji se želi prikazati. Pri tome ne postoji mogućnost da se po izboru pale i gase pojedine tačke. Ova mogućnost se dobija tek kada se uključi mod za visoku rezoluciju. »Pre-

kidač« za uključenje ovog moda je peti bit u registru 53265. Ova operacija, na žalost, nije dovoljna da se mogu koristiti sve tačke ekrana.

Iz prethodnog poglavlja o tekst-modu vidi se kako teče proces ispisivanja ekrana: iz ekranske memorije uzima se kôd karaktera, iz kolor-memorije uzima se boja, a iz karakter-memorije oblik simbola koji će se prikazati. Kao što se vidi, nigde u memoriji ne nalazi se tačan opis trenutnog rasporeda tačaka na ekranu, već se on neprekidno oblikuje, uzimajući pri tom neophodne podatke iz različitih delova karakter-memorije. U visokoj rezoluciji primenjuje se drugačija tehnika rada:

1) Najpre se odredi deo memorije veličine 64000 bita (ili 8000 registara) u kome će svaki bit predstavljati jednu tačku na ekranu. Ako je njegova vrednost 1, tačka će biti prikazana, u protivnom neće (biće u boji pozadine). Ovaj deo memorije zove se **BIT-MAPA ekrana;**

2) Ekranska memorija koristi se sada u druge svrhe: sadržaj svakog njenog registra odnosi se na boju pozadine i boju tačke u odgovarajućem polju od 8 puta 8 tačaka, koje je ranije bilo predviđeno za prikaz jednog karaktera. Gornja četiri bita odnose se na boju tačke, donja četiri na boju pozadine. U visokoj rezoluciji moguće je odrediti osvetljenost svake od 64000 tačaka ponaosob, ali se obojenost tačaka i pozadine na kojoj se nalaze određuje za grupe od 8*8=64 tačke, sve koje se nalaze na mestu za prikaz jednog karaktera. (Bilo bi potrebno mnogo više memorije za čuvanje boje svake tačke, jer je za zapis boje potrebno četiri bita, tj. polovina registra!).

Sada će se prilikom formiranja slike na ekranu jednostavno preslikati stanje 8000 registara iz bit-mape ekrana u odgovarajuće tačke. Karakter-memorija se sada ne može koristiti, pa je onemogućeno i svako pisanje teksta (sem ako korisnik sam ne nacrta pojedine simbole, tačku po tačku).

Boja tačke i pozadine se za svaku grupu od 8*8 tačaka uzima iz ekranske memorije.

Ukoliko se nije menjalo mesto video-bloka u memoriji, uobičajeni postupak za rezervisanje memorije za bit-mapu je postavljanje bita 3 u registru 53272 na 1. Ovim se postiže da se početna adresa bit-mape ekrana postavi na

8192. Ukoliko je menjano mesto video-bloka, vrednost 0 za ovaj bit označava da je za bit-mapu izabrana njegova niža polovina (0 do 8191), a vrednost jedan viša polovina (8192 do 16389).

Ekran je i dalje organizovan po karakterima i na svakom mestu za karakter nalazi se 64 tačke u osam uzastopnih registara čiji se sadržaj direktno opisuje. Sa slike se vidi na koji način registri iz bit-mape obrazuju sliku:

	8192		8200		8504
	8193		8201		8505
	8194		8202 ...		8506
Registri za	8195	Drugi	8203 ...	40-ti	8507
prvi karakter	8196	karakter	8204 ...	karakter	8508
prvog reda	8197		8205		8509
	8198		8206		8510
	8199		8207		8511
	8512		8520		
	8513		8521		
	8514		.		
Registri za	8515	Drugi	.		
prvi karakter	8516	karakter	.		
drugog reda	8517				
	8518				
	8519				
	.				
	.				
	.				

Ovakav postupak omogućuje da se neposrednim menjanjem bit-mape menja i slika na ekranu. Dalje se navodi postupak za postavljanje jednog bita u mapi, tj. jedne tačke na ekranu:

Ako su X i Y koordinate tačke (X je između 0 i 319 a Y između 0 i 199), onda se registar u kome se nalazi bit za prikaz te tačke nalazi na sledeći način:

— Najpre se odredi koordinata karakter-polja, tj. polja na ekranu u kome se nalazi tačka (X, Y). Kako svaki karak-

ter zauzima 8 tačaka po širini i osam tačaka po visini ekrana biće:

RED=INT(Y/8)
KOLONA=INT(X/8)

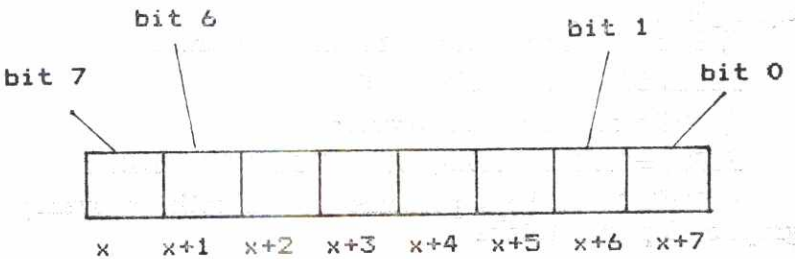
Koji od 8 registara u karakter-polju sadrži tačku (X,Y), biće određeno kao ostatak celobrojnog deljenja Y sa 8:

LINIJA=Y—8*INT(Y/8)=—8*RED
(ili LINIJA=Y AND 7)

Konačno, koji tačno bit odgovara traženoj tački, odrediće se na sledeći način:

BIT = 7 — (X — 8*INT(X/8))
(ili BIT = 7 — X AND 7)

Na prvi pogled, zbunjuje oduzimanje ostatka od celobrojnog deljenja X sa 8 od 7. Bitovi u okviru registra označavaju se zdesna nalevo, a ne sleva udesno, kako se ponašaju tačke na ekranu, pa je jasno zašto je to neophodno.



Sl. 6.1 — Veza X-koordinate ekrana i prikaza u bit-mapi

Konačno, izvodimo formulu za određivanje adrese registra i bita u njemu koji treba postaviti na 1 da bi na ekranu zasvetlela tačka. Potrebno je uočiti sledeće:

1. Za svaki karakter-red pre onoga u kome se nalazi tačka treba preskočiti 40 karakter-polja. Pošto se svako polje prikazuje pomoću 8 registara, treba od početne adrese bit-mape preskočiti 8*40=320 registara za svaki takav red.

2. U okviru istog karakter-reda treba preskočiti po 8 registara za svako karakter-polje pre onoga u kome se nalazi tačka.

3. U okviru istog karakter-polja treba preskočiti onoliko registara kolika je vrednost koordinate LINIJA unutar karakter-polja.

Na osnovu ovoga izvodimo obrazac:

```
REG = 320 * RED + 8 * KOLONA + LINIJA +
+ 8192
```

Postavljanje bita u ovom registru dobija se logičkom operacijom OR između prethodnog stanja registra i 2 ∧ BIT. (2 ∧ BIT ima vrednost 1 samo na mestu BIT, pa u rezultatu na tom mestu sigurno stoji 1, bez obzira na prethodno stanje. Preostali deo registra se ne menja.)

Naredba za postavljanje tačke na ekran ima sledeći oblik:

```
POKE REG, PEEK(REG) OR 2 ↑ BIT
```

Brisanje tačke izvršilo bi se slično upisu:

```
POKE REG, PEEK(REG) AND (255—2 ↑ BIT)
```

Kompletan postupak formiranja slike u visokoj rezoluciji izgleda ovako:

1) Rezerviše se prostor za bit-mapu ekrana sa:

```
POKE 53272,PEEK(53272) ORB
```

2) Briše se rezervisani prostor:

```
FOR I=8192 TO 8192+7999 : POKE I,0 : NEXT I
```

3) Definiše se boja pozadine i boja tačke u svakoj grupi od 8 puta 8 tačaka:

```
POKE AEM,BP+16*BT
```

(AEM=Adresa Ekranske Memorije)
(BP=Boja Pozadine)
(BT=Boja Tačke)

4) Uključuje se visokorezolucijski mod:

```
POKE 53265,PEEK(53265)OR32
```

5) Transformacija koordinata svih tačaka i prikaz na ekranu:

```
RED=INT(Y/8)
KOL=INT(X/8)
```

```
LIN=Y AND 7
BIT=7—(X AND 7)
REG=8192+RED*320+KOL*8+LIN
POKE REG,PEEK(REG) OR 2↑BIT
```

6) Isključuje se visokorezolucijski mod:

```
POKE 53265,PEEK(53265) AND 223
```

7) Oslobađa se rezervisana memorija:

```
POKE 53272,PEEK(53272) AND 247
```

Važne memorijske lokacije za visokorezolucijski mod su:

Uključenje moda	53265 bit 5
Rezervisanje memorije	53272 bit 3
Bit-mapa ekrana	8192 do 16191
Boje tačaka i pozadine	1024 do 2023

P r i m e r: Rad u visokorezolucijskom modu biće prikazan na programu za crtanje kruga. Pri tome će biti korišćen prethodni kompletno opisani postupak.

```
 10 POKE 53272,PEEK(53272) OR 8
 20 POKE 53265,PEEK(53265) OR 32
 30 FOR I=8192 TO 8192+7999
 40 POKE I,0
 50 NEXT I
 60 FOR I=1024 TO 2023
 70 POKE I,1
 80 NEXT I
 90 FOR I=0.001 TO 2*3.14159 STEP 3.14159/300
100 X=80*COS(I)+160
110 Y=80*SIN(I)+100
120 GOSUB 200
130 NEXT I
140 WAIT 198,1
150 POKE 53265,PEEK(53265) AND (255—32)
160 POKE 53272,PEEK(53272) AND (255—8)
170 PRINT "{CLR}"
180 STOP
190:
```

```
200 REM UPIS TACKE NA EKRAN
210 RED=INT(Y/8)
220 KOL=INT(X/8)
230 LIN=Y AND 7
240 BIT=7—(X AND 7)
250 REG=8192+RED*320+KOL*8+LIN
260 POKE REG,PEEK(REG) OR 2 ↑ BIT
270 RETURN
```

Nakon rezervisanja memorije (red 10) i uključenja visokorezolucijskog moda (red 20) vrši se brisanje memorije za bit-mapu (redovi 30 do 50) i postavljanje boja za polja i tačke u njima (redovi 60 do 80). Postavljanjem vrednosti u sve registre ekranske memorije određuje se da pozadina polja bude bela (jer niža četiri bita postavljena na 0001 daju vrednost 1, što je kôd za belu boju), a tačke u njima crne (jer viša četiri bita postavljena na 0000 daju vrednost 0, što je kôd za crnu boju). Brisanje memorije za bit-mapu i postavljanje boje obično se vrši pre uključenja visokorezolucijskog moda, ali se na ovaj način može jasno videti tok izvršavanja ovih operacija.

Određivanje koordinata tačaka koje čine krug sa centrom u središtu ekrana vrši se opisom kruga u polarnim koordinatama (redovi 90 do 130). Za izračunate koordinate tačke vrši se njen prikaz na ekranu pozivanjem potprograma na liniji 200 (red 120). Po završenom crtanju čeka se na unošenje bilo kog znaka da bi se prešlo u tekst-mod (red 140).

Upis tačke na ekran vrši se određivanjem onog bita u bit-mapi koji odgovaraju toj tački. Postupak se vrši prema već opisanim pravilima.

N a p o m e n a: Zamenom redova 90 do 130 mogu se dobiti i drugi geometrijski likovi. Na primer:

```
 90 Y=100
100 FOR I=100 TO 200
110 X=I
120 GOSUB 200
130 NEXT I
```

daje duž između tačaka (100,100) i (100,200).

Zamenom, na primer, reda 110 prvog primera sa:

```
110 Y=40*SIN(I)+100
```

dobija se elipsa, i slično.

Višebojni visokorezolucijski mod. Ako je za rešenje nekog problema neophodno da i tačke unutar jednog karakter-polja budu u više boja, može se žrtvovati polovina tačaka po širini ekrana i dobiće se ova mogućnost. Na žalost, potpuna sloboda za bojenje tačaka ovim se neće dobiti, već se broj boja koje se mogu koristiti u okviru jednog karaktera povećava na 3.

Slično kao i kod višebojnog tekst-moda, jedna tačka se ovde u memoriji označava sa dva bita, čija kombinacija određuje boju tačke. Na račun toga, rezolucija se po širini smanjuje sa 320 na 160 tačaka.

Da bi se uključio ovaj mod, pored prethodno opisanog generalnog uključenja visokorezolucijskog moda postavljanjem bita 5 u registru 53265 na 1 treba na 1 postaviti i bit 4 na lokaciji 53271. Time je označeno da se koristi višebojna varijanta visokorezolucijskog moda.

Boja kojom će se tačka prikazati sada se određuje na sledeći način:

Kombinacija bita u bit-mapi ekrana	Boja
0 0	boja pozadine iz 53281
0 1	boja u gornja četiri bita ekranske memorije
1 0	boja u donja četiri bita ekranske memorije
1 1	boja iz kolor-memorije

Postavljanje tačke na ekran sada se vrši nešto drugačije, jer je rezolucija po širini dvostruko manja, a pored toga pojavljuje se i faktor "boja", koji se sada može postaviti po želji.

Određivanje registara i para bitova u njemu vrši se sledećim transformacijama:

```
RED=INT(Y/8)
KOL=INT(X/4)
```

```
LIN=Y AND 7
BIT=3—(X AND 3)
REG=320*RED+8*KOL+LIN+8192
```

Postavljanje tačke u boji BOJA vrši se na sledeći način:
— Najpre se obriše prethodna boja:

```
POKE REG,PEEK(REG) AND (255—4↑ BIT*3)
```

— Upisuje se nova boja:

```
POKE REG,PEEK(REG) OR (4 ↑ BIT*BOJA)
```

Pri tome je BOJA između 0 i 3.
Kompletan postupak je sledeći:
1) Rezerviše se memorija za bit-mapu:

```
POKE 53272,PEEK(53272) OR 8
```

2) Briše se rezervisani prostor:

```
FOR I=8192 TO 8192+7999 : POKE I,0 : NEXT I
```

3) Definiše se boja pozadine i boja tačaka u svakom polju od 8*8 tačaka

```
POKE 53281,B00
                          (B00=Boja čiji će kôd biti 00)
POKE AEM,B10+16*B01
                  (AEM=Adresa Ekranske Memorije)
                  (B10=Boja čiji će kôd biti 10)
                  (B01=Boja čiji će kôd biti 01)
POKE AKM,B11
                  (AKM=Adresa Kolor-Memorije)
                  (B11=Boja čiji će kôd biti 11)
```

4) Uključuje se visokorezolucijski višebojni mod:

```
POKE 53265,PEEK(53265)OR32
POKE 53271,PEEK(53271)OR16
```

5) Transformacija koordinata svih tačaka i prikaz na ekranu:

```
RED=INT(Y/8)
KOL=INT(X/4)
LIN=Y AND 7
BIT=3—(X AND 3)
REG=8192+RED*320+KOL*8+LIN
POKE REG,PEEK(REG) AND (255—4 ↑ BIT*3)
POKE REG,PEEK(REG) OR (4 ↑ BIT*BOJA)
```

6) Isključuje se višebojni visokorezolucijski mod:

```
POKE 53265,PEEK(53265) AND 223
POKE 53271,PEEK(53271) AND 239
```

7) Oslobađa se rezervisana memorija:
POKE 53272,PEEK(53272) AND 247

Važne memorijske lokacije za visokorezolucijski višebojni mod su:

Uključenje moda	53265 bit 5
	53271 bit 4
Rezervisanje memorije	53272 bit 3
Bit-mapa ekrana	8192 do 16191
Boje tačaka za kôd 01 i 10	1024 do 2023
Boja tačaka za kôd 11	55296 do 56295

P r i m e r: Nacrtajmo tri raznobojne duži jednu do druge.

```
 10 POKE 53272,PEEK(53272) OR 8
 20 POKE 53265,PEEK(53265) OR 32
 25 POKE 53271,PEEK(53271) OR 16
 26 POKE 53281,2 : REM BOJA 00 JE CRVENA
 30 FOR I=8192 TO 8192+7999
 40 POKE I,0
 50 NEXT I
 60 FOR I=0 TO 999
 70 POKE 1024+I,1 : REM BOJA 01 JE CRNA,
    10 JE BELA
 75 POKE 55296+I,5 : REM BOJA 11 JE ZELENA
 80 NEXT I
 90 FOR BOJA=1 TO 3
100 FOR X=50 TO 100
105 Y=100+BOJA
110 GOSUB 200
120 NEXT X
130 NEXT BOJA
140 WAIT 198,1
145 POKE 53271,PEEK(53271) AND (255—16)
150 POKE 53265,PEEK(53265) AND (255—32)
160 POKE 53272,PEEK(53272) AND (255—8)
170 PRINT "{CLR}"
180 STOP
190:
200 REM UPIS TACKE NA EKRAN
```

```
210 RED=INT(Y/8)
220 KOL=INT(X/4)
230 LIN=Y AND 7
240 BIT=3—(X AND 3)
250 REG=8192+RED*320+KOL*8+LIN
255 POKE REG,PEEK(REG) AND (255—4↑BIT*3)
260 POKE REG,PEEK(REG) OR (4↑BIT*BOJA)
270 RETURN
```

Uočimo razlike između upotrebe standardnog i višebojnog visokorezolucijskog moda: Treba uključiti višebojni mod (red 25) i postaviti boje za tačke 00,01,10 i 11. Tačke 00 boje se bojom za pozadinu ekrana (red 26), za boje 01 i 10 to je određeno ekranskom a za boje 11 kolor-memorijom (redovi 70 i 72). Raspon koordinata po X-osi sada je 160, pa će duži zauzimati sredinu ekrana (redovi 100 do 120). Pišu se tri duži, za svaku boju po jedna.

Pri isključenju se vrši i isključenje višebojnog moda (red 145).

Postupak upisa tačke u bit-mapu sada se obavlja po postupku za višebojni visokorezolucijski mod.

6.6. SPRAJTOVI

Pokretanje objekata na ekranu i njihovi uzajamni odnosi predstavljaju veliki izazov za programera. Ako je skupom tačaka prikazan neki lik koji treba pomeriti, onda treba obaviti sledeće programske operacije:

1. Izračunati nove koordinate svake od tačaka u odnosu na pravac kretanja i stare koordinate.
2. Izbrisati stari prikaz.
3. Upisati sve tačke na njihova nova mesta.

Pri tome mora biti postignuta i dovoljno velika brzina izvođenja ovih operacija, tolika da se ne primećuje promena pozicija pojedinih tačaka, već samo celog lika.

BASIC-jezik navedene operacije nije u stanju da izvrši dovoljno brzo. Za programiranje u mašinskom jeziku treba uložiti mnogo znanja i truda, a efekat će i pored toga često biti skroman.

Rešenje ovog problema ostvareno je tehničkim putem. Projektovane su elektronske komponente pomoću kojih

se istovremeno na ekranu može prikazati nekoliko slika. Pri tome se za svaku od njih zadaje samo minimalni broj podataka o izgledu i mestu prikazivanja. Podaci o njihovim uzajamnim odnosima mogu se dobiti na jednostavan način.

Video-čip koji u sebi sadrži Commodore 64 je primer jedne takve komponente. On omogućuje da se istovremeno na ekranu prikaže jedna glavna slika od 320 puta 200 tačaka (koja prikazuje tekst ili sliku) i još 8 malih slika od 24 puta 21 tačke kojima se prikazuju različiti likovi. Ovih 8 slika se prikazuje potpuno nezavisno od glavne slike.

Oblik svake od njih opisuje sam korisnik, i ne prikazuje se dok se to ne zatraži. Kada se to učini, trenutno se pojavljuje na prethodno određenom mestu i tamo ostaje sve dok se ne zahteva njeno pomeranje na drugo mesto ili isključenje. One mogu biti prikazane tako da zaklanjaju sadržaj glavnog ekrana ili da njime budu zaklonjene, a mogu i jedna drugu zaklanjati. Boje tačaka kojima se formiraju su nezavisne. Moguća je jednostavna detekcija međusobnog dodira.

Uobičajeni naziv za ovakve male slike je SPRAJT (SPRITE) i ovaj termin koristiće se i dalje. Pošto ih ima 8, neka njihove oznake budu brojevi od 0 do 7.

Da bi se sprajt pojavio na ekranu treba obaviti sledeće opracije:

1 — definisati izgled sprajta,

2 — postaviti njegov opis na za to određeno mesto u memoriji,

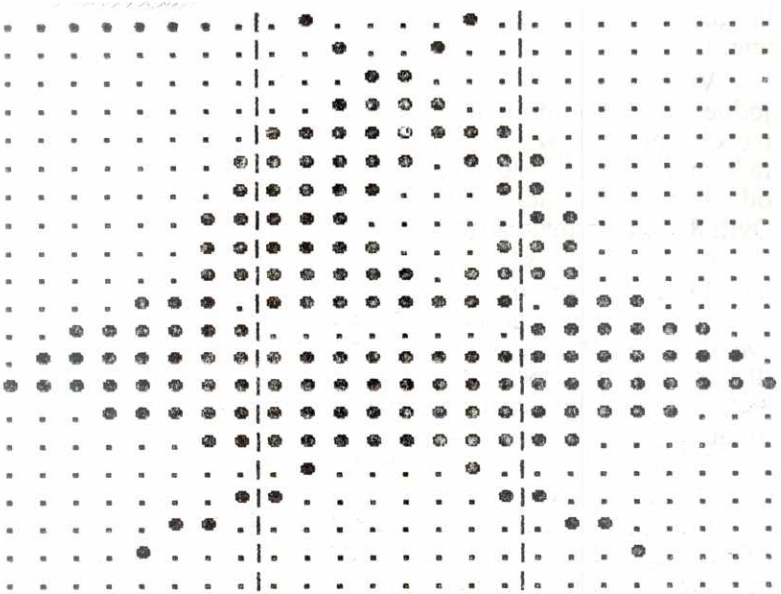
3 — odrediti njegovu boju,

4 — postaviti koordinate mesta od koga počinje prikaz,

5 — uključiti prikazivanje.

1. **Definisanje izgleda sprajta.** U ovoj etapi računar ne može pomoći, već su mesta i smisao za grafički prikaz ovde glavno oruđe. Treba uraditi sledeće: nacrtati na papiru polje od 21 reda po 24 tačke i u njemu pojačati svaku tačku koja treba da bude deo slike. U prikazu sprajta samo će se ove tačke videti, dok će sve ostale biti prikazane u boji pozadine (boji osnove ekrana). Polje potom podeliti na tri uspravna dela širine 8 tačaka.

Neka sprajt ima oblik letećeg tanjira, kakav je prikazan na sl. 6.2.



Sl. 6.2 — Opis sprajta

Ako se sve pojačane tačke zamisle kao jedinice, a ostale kao nule, za svaku od 21 linije mogu se izračunati po tri dekadna broja koji predstavljaju zapis svake od 3 grupe od po 8 nula i jedinica iz te linije. Na primer, za drugu grupu tačaka u prvoj liniji biće:

$01000010 = 2 \wedge 6 + 2 \wedge 1 = 64 + 2 = 66$

Polje tačaka sa slike sada je predstavljeno brojevima na sledeći način:

```
0   66    0
0   36    0
0   24    0
0   60    0
0  255    0
1  251  128
1  241  128
```

3	224	192
3	241	192
3	251	192
14	255	112
63	0	252
127	255	254
255	255	255
31	255	248
3	255	192
0	66	0
1	129	128
6	0	96
8	0	16
0	0	0

Sa ova 63 broja sprajt je potpuno definisan i može se preći na sledeću etapu.

2. **Postavljanje opisa u memoriju.** Da bi ovaj postupak bio što jasniji, definišimo na početku sledeće pojmove:

1) **Adresa sprajta** — ova adresa nije prava memorijska adresa, već adresa grupe od 64 registra u koje se u memoriji zapisuje opis strajta (na primer ako je adresa sprajta 13, grupa registara koji sadrže njegov opis je između 13*64=832 i 895). Kako je za zapis potrebno 63 mesta, 64-to se ne koristi. Adresa sprajta može biti između 0 i 255.

2) **Adresa prvog registra u opisu sprajta** — ova adresa je prava memorijska adresa i ukazuje na mesto gde je zapisan prvi broj u opisu sprajta. Dobija se tako što se adresa sprajta pomnoži sa 64. (Ako je menjano mesto video-bloka ovoj vrednosti treba dodati i početnu adresu video-bloka).

Kako sprajt izgleda biće potpuno definisano tek onda kada se saopšti gde se u memoriji nalazi njegov opis. To se vrši za svaki sprajt koji se koristi, postavljanjem adrese sprajta u za to predviđeni registar:

za sprajt 0	registar 2040
za sprajt 1	registar 2041
*	
*	
za sprajt 7	registar 2047

Nakon toga, počev od adrese prvog broja u opisu sprajta, upisuju se jedan za drugim 63 broja iz opisa. Brojevi se uzimaju redom, vrstu po vrstu. Time je završen postupak postavljanja opisa sprajta u memoriju.

Treba napomenuti da se adrese sprajta od 0 do 31 (izuzev 13,14 i 15) ne mogu koristiti, jer se na tim mestima nalaze važne sistemske promenljive koje bi tako bile uništene i program prekinut. Takođe, ako se koristi adresa sprajta 32 ili nekoliko narednih, doći će do preklapanja opisa sprajta i zapisa BASIC-programa u memoriji, pri čemu će program biti uništen. Zato je dobro kao ove adrese koristiti brojeve veće od 128, u slučaju da se koristi tekst-mod ili nešto niže od 128 ako se koristi visokorezolucijski mod.

Prethodna napomena važi samo ukoliko se video-blok nalazi na adresama od 0 do 16383, tj. mesto mu po uključenju računara nije menjano.

3. **Postavljanje boje sprajta.** Za svaki sprajt postoji njegov registar boje. Upisom kôda boje (od 0 do 15) u ovaj registar, određuje se kojom bojom će sprajt biti prikazan:

za sprajt 0	registar 53287
za sprajt 1	registar 53288
*	
*	
za sprajt 7	registar 53294

4. **Postavljanje koordinata sprajta.** Pod koordinatom sprajta podrazumeva se koordinata tačke u gornjem levom uglu sprajta na celom ekranu. Koordinate sprajta zato nisu isto što i koordinate vidljivih tačaka (0 do 320 za X i 0 do 200 za Y). Naime, koordinate tačaka obuhvataju samo vidljivi deo ekrana, dok koordinate sprajta mogu biti i van tog dela, u ramu polja slike. Tada se sprajt ne vidi. Na taj način omogućeno je da sprajt može da ulazi i izlazi iz vidljivog dela, pri čemu se vidi samo onaj njegov deo koji je u vidljivom delu.

Veza između koordinata sprajtova (koordinata celog ekrana) i koordinata vidljivog dela može se predstaviti na sledeći način:

Koordinate tačke u vidljivom delu		Koordinate tačke na celom ekranu	
X	Y	X	Y
0	0	24	50
319	0	343	50
0	199	24	249
319	199	343	249

Iz ovoga se može zaključiti (imajući u vidu da sprajt ima 24 * 21 tačku) da je sprajt delimično vidljiv ako su obe njegove koordinate u sledećim granicama:

$1 =< X =< 343$

$30 =< Y =< 249$

Dakle, transformacija vidljivih koordinata (VX,VY) u ekranske (X,Y) vrši se po sledećim formulama:

$X=VX+24 \qquad Y=VY+50$

Kao što je poznato, u jedan registar od 8 bita može biti upisan jedino broj između 0 i 255. Za upis Y koordinate to je dovoljno, ali za X nije. Zato se zapis X-koordinate rastavlja u dva dela: glavni deo (nižih 8 bita) i dodatni, deveti bit. Za svaki sprajt određen je po jedan registar za zapis glavnog dela a za zapis dodatnog bita predviđen je registar u kome bit 0 služi za upis dodatnog bita X-koordinate sprajta 0, bit 1 za dodatni bit X-koordinate sprajta 1 i slično za ostale.

Registri za koordinate u memoriji su raspoređeni na sledeći način:

X-koo sprajta 0	53248
Y-koo sprajta 0	53249
X-koo sprajta 0	53250
Y-koo sprajta 0	53251
*	
*	
*	
X-koo sprajta 0	53262
Y-koo sprajta 0	53263
registar za dopunski bit	53264

Ovakav zapis zahteva da se određivanje X-koordinate obavi u nekoliko faza:

1 — celobrojno deljenje vrednosti X—a sa 256,

2 — upis ostatka od deljenja u registar za glavni deo vrednosti X-koordinate,

3 — upis rezultata deljenja u odgovarajući bit registra za dodatne bitove.

Postupak se izvodi na sledeći način:

```
DBIT=INT(X/256)
POKE XSPR,X—256*DBIT
IF DBIT=0 THEN POKE 53264,PEEK(53264)AND
(255—2 ↑ BRSPR)
IF DBIT=1 THEN POKE 53264,PEEK(53264)OR
(2 ↑ BRSPR)
POKE YSPR,Y
```

(DBIT= dodatni dit za X)
(BRSPR= broj sprajta)
(XSPR= X-koo sprajta)
(YSPR= Y-koo sprajta)

Ovim je potpuno određena pozicija sprajta na ekranu. On se neće videti sve dok se ne obavi sledeća etapa, uključenje prikaza.

5. **Uključenje prikaza sprajta.** Uključivanju i isključivanju svih osam sprajtova namenjen je registar na adresi 53269. Bit 0 odgovara sprajtu 0, bit 1 sprajtu 1... Postavljanjem nekog bita na 1 vrši se uključivanje prikazivanja odgovarajućeg sprajta, na onom mestu koje je zapisano u registrima za koordinate sprajta. Postavljanjem bita na 0, prikazivanje se prekida.

6. **Kretanje sprajta.** Kada je sprajt jednom definisan, postupak njegovog pokretanja svodi se na promenu vrednosti registara za koordinate sprajta. Potrebno je registre za X-koordinatu (glavni i dodatni) i Y-koordinatu postaviti na nove vrednosti i sprajt će trenutno automatski biti premešten sa starog na novo mesto.

7. **Odnos sprajta i pozadine.** Kako je već rečeno, sprajtovi se mogu koristiti u isto vreme kad i glavni ekran. Pri tome se mogu postići sledeći efekti:

— da bi sprajt zaklanjao tačke na ekranu, treba postaviti njemu odgovarajući bit u registru 53275 na 0. Tada sprajt ima viši prioritet prikaza u odnosu na pozadinu;

— da sprajt prolazi ispod vidljivih tačaka ekrana treba isti bit postaviti na 1 (prioritet pozadine);

— da li sprajt i ekran imaju bar jednu zajedničku tačku, može se saznati čitajući sadržaj registra 53279. U slučaju dodira nekog sprajta i osvetljene tačke sa ekrana, bit koji odgovara broju sprajta tehnički se postavlja na 1 i ostaje tako sve dok se ne pročita. To znači da će jednom otkriveni sudar biti zapamćen i ako se oni potom udalje, ukoliko se ne pročita sadržaj ovog registra. Tada se svi bitovi automatski postavljaju na 0 i omogućuje detektovanje sledećeg sudara.

Postupak provere sudara vrši se naredbom:

IF PEEK(53279)<>0 THEN... obrada za slučaj sudara.

8. **Uzajamni odnos sprajtova.** Sprajt sa nižim brojem uvek ima viši prioritet od sprajta sa višim brojem. To znači da će se u slučaju preklapanja dva sprajta prikazati onaj sa nižim brojem.

Da li su se dva sprajta dotakla može se ustanoviti slično kao sa pozadinom. U registru 53278 automatski će biti postavljeni oni bitovi čiji brojevi odgovaraju brojevima sprajtova koji su učestvovali u sudaru. I ovaj se registar briše tek nakon čitanja. Ispitivanje da li se sudar dogodio može se vršiti naredbom:

IF PEEK(53278)<>0 THEN... obrada za slučaj sudara

9. **Uvećanje sprajtova.** Kako je 24*21 dosta mala slika, predviđena je mogućnost uvećanja sprajtova. Ovo se može obaviti na tri načina:

1 — uvećanjem širine dva puta (na 48*21 tačku),

2 — uvećanjem visine dva puta (na 24*42 tačke),

3 — uvećanjem i širine i visine po dva puta (na 48 puta 42 tačke)

U svim ovim slučajevima važi sledeće: definisanje sprajta je nepromenjeno, tj. njegov opis je isti kao i ranije, ali se svaka njegova tačka prikazuje sa po dve tačke po širini ako se vrši proširivanje, po visini ako mu se uvećava visina i kvadratom od 2*2 tačke ako se uvećanje vrši po obe dimenzije.

Kao indikatori za uvećan prikaz sprajtova služe registri:

— za horizontalno uvećanje 53271,
— za vertikalno uvećanje 53277.

Ako u nekom bitu stoji 1, odgovarajući sprajt će biti prikazan prošireno. Dakle, kombinacijom vrednosti u ova dva registra dobija se:

Bit u 53271	Bit u 53277	Efekat
0	0	običan prikaz
1	0	horizontalno proširenje
0	1	vertikalno proširenje
1	1	proširenje u oba pravca

10. **Višebojni sprajtovi.** Žrtvujući polovinu tačaka po horizontali, tj. svodeći dimenzije sprajta na 12*21 tačku, moguće je u okviru jednog sprajta dobiti do četiri boje. Opis sprajta u ovom slučaju i dalje sadrži 24*21 tačku, ali se ne razmatra tačku po tačku, već par po par tačaka. Svaki par određuje kojom bojom će prikaz biti izvršen. Jedna od tih boja je boja ekrana, druga je boja određena za taj sprajt i smeštena u ranije opisani registar boje a preostale dve boje se unapred definišu za svih osam sprajtova. One se smeštaju u posebne za to predviđene registre na adresama 53285 i 53286.

Prilikom prikazivanja sprajt neće postati dva puta uži, već će ostati normalnih dimenzija (24*21). Ograničenje je u tome što su sada njegove tačke (po širini) za korisnika dostupne **samo u parovima,** što je isto kao da ih je dva puta manje. A za svaku takvu tačku na ekranu se prikazuju dve, u istoj boji, jedna do druge.

Boja tačke višebojnog sprajta određuje se na sledeći način:

Kombinacija bitova u opisu	Adresa boje
0 0	boja pozadine 53281
0 1	53285
1 0	registar za boju sprajta
1 1	53286

Sprajt će se prikazivati jednobojno, bojom iz njegovog registra za boju, sve dok se ne označi kao višebojni sprajt.

To se postiže postavljanjem odgovarajućeg bita u registru 53276.

Primer korišćenja sprajtova. Kao primer za demonstraciju rada sa sprajtovima poslužiće opis letećeg tanjira, sa početka ovog poglavlja. Neka njegov broj bude 0 i neka je za adresu sprajta 0 uzeto 13. Adresa prvog bajta u opisu biće tada 13*64=832. Neka boja sprajta bude crvena i neka se kreće po ekranu koji sadrži samo dve linije nastale kombinacijom grafičkih simbola. Osnovni program koji će obaviti sve navedene operacije je sledeći:

```
 5 REM OPIS SPRAJTA
10 DATA 0,66,0,0,36,0,0,24,0,0,60,0,0,255,0,1,251,
   128,1,241,128
20 DATA 3,224,192,3,241,192,3,251,192,14,255,112,63,
   0,252
30 DATA 127,255,254,255,255,255,31,255,248,3,255,
   192,0,66
40 DATA 0,1,129,128,6,0,96,8,0,16,0,0,0
45 :
50 REM SMESTANJE OPISA U MEMORIJU
55 :
60 REM------ADRESA SPRAJTA JE 13
70 REM------PRVI BAJT OPISA JE 13*64=832
75 :
80 POKE 2040,13
85 :
90 FOR I=0 TO 62
100 READ X
110 POKE 13*64+I,X
120 NEXT I
121 :
125 REM BOJA SPRAJTA JE CRVENA
129 :
130 POKE 53287,2
131 :
135 REM CRTANJE POZADINE
139 :
145 REM CEO EKRAN U BELU BOJU
150 POKE 53281,1 : POKE 53280,1
151 :
```

```
155 PRINT "{clr}" :REM BRISANJE EKRANA
160 PRINT "{ctrl/1}" :REM PISE SE CRNOM BOJOM
161 :
170 PRINT "{15 dole}"
180 FOR I=1 TO 20
19 PRINT "{rvs on} {prazno)(rvs off} {prazno}" ;
195 NEXT I
196 PRINT
200 PRINT "{dole}"
210 FOR I=1 TO 40
211 PRINT "{rvs on} {prazno)(rvs off}" ;
212 NEXT I
213 :
214 REM PARAMETRI SPRAJTA
215 POKE 53269,1: REM UKLJUCENJE PRIKAZA
216 POKE 53271,0: REM NEPROSIREN PO Y
217 POKE 53277,0: REM NEPROSIREN PO X
218 POKE 53275,0: REM VISI PRIORITET OD
    POZADINE
219 :
220 REM KRETANJE SPRAJTA
221 :
230 FOR X=0 TO 255
235 Y=X
240 POKE 53248,X
250 POKE 53249,Y
260 NEXT X
265 :
266 REM BESKONACNA PETLJA, KRAJ SA RUN/STOP
270 GOTO 230
```

Program treba uneti u računar i snimiti na traku ili disketu, jer će kasnije biti potreban u prvobitnom obliku.

Različitim modifikacijama ovog programa biće prikazano šta se događa kada se neki parametri promene.

Kada se program startuje treba obratiti pažnju na veličinu i boju sprajta, kao i na pravac njegovog kretanja i prepreke na putu.

Sada nastaje modifikacija:

1) Proširenje po horizontali:

```
216 POKE 53271,0
217 POKE 53277,1
```

2) Proširenje po vertikali:

```
216 POKE 53271,1
217 POKE 53277,0
```

3) Proširenje po horizontali i vertikali:

```
216 POKE 53271,1
217 POKE 53277,1
```

4) Povratak na običan prikaz:

```
216 POKE 53271,0
217 POKE 53277,0
```

5) Viši prioritet pozadine od sprajta:

```
218 POKE 53275,1
```

6) Sudar sprajta i pozadine:

```
255 IF PEEK(53279)<>0 THEN 230
```

ili:

```
255 IF PEEK(53279)<>0 THEN WAIT 198,1
```

Ovo demonstrira i naredbu WAIT. Ako je broj simbola koji je otkucan na tastaturi dok program radi neparan, program će se zaustaviti i čekati kad god je uslov ispunjen, tj. kada sprajt dotakne liniju. Ako je taj broj paran, WAIT nema efekta i program se odvija kao da linija 255 ne postoji.

7) Sudar dva sprajta:

```
132 REM DEFINISANJE SPRAJTA 1
133 POKE 2041,13 : REM IZGLEDA ISTO KAO SPRAJT 0
134 POKE 53288,7 : REM ZUTA BOJA
```

```
215 POKE 53269,3 : REM UKLJUCENJE PRIKAZA
251 POKE 53250,255—X : REM X—KOORDINATA
252 POKE 53251,Y : REM Y—KOORDINATA
255 IF PEEK(53278)<>0 THEN 230
ILI
255 IF PEEK(53278)<>0 THEN STOP
```

8) Rad sa dodatnim bitom za X:

```
132
133
134
251
252
255
215 POKE 53269,1 : REM UKLJUCENJE SPRAJTA 0
230 POKE 53249,100 : REM Y—KOORDINATA
235 BRSPR=0
240 FOR X=0 TO 360
245 DBIT=INT(X/256)
250 POKE 53248,X—DBIT*256
255 IF DBIT=0 THEN POKE 53264,PEEK(53264)AND
    (255—2 ↑ BRSPR)
260 IF DBIT=1 THEN POKE 53264,PEEK(53264)OR
    (2 ↑ BRSPR)
263 NEXT X
```

Sprajt se sada kreće po horizontali, preko celog ekrana.

9) Višebojni sprajtovi.

Zapamćeni program sada treba uneti sa trake ili diskete u memoriju i izvršiti sledeće izmene:

```
10 DATA 0,66,0,0,36,0,0,24,0,0,40,0,0,170,0,2,170,128,2,
   160,128
20 DATA 2,160,128,2,160,128,2,170,128,12,170,
   48,63,0,252
30 DATA 255,255,255,255,255,255,63,255,252,
   3,255,192,0,66
40 DATA 0,1,129,128,6,0,96,8,0,16,0,0,0
```

```
131 POKE 53285,2 : REM CRVENA BOJA ZA 10
132 POKE 53286,6 : REM TAMNO PLAVA ZA 11
133 POKE 53276,1 : REM SPRAJT 0 JE VISEBOJNI
134:
```

Ako se sada od opisa iz DATA naredbe napravi skica na papiru, mogu se uočiti promene nastale u prvobitnom opisu i njihov uticaj na konačni izgled sprajta.

7. DODACI

7.1. BASIC ZA COMMODORE 64
(pregled komandi, naredbi i funkcija)

U ovom prikazu BASIC-jezika za Commodore-64, izložene su sve naredbe, komande i funkcije abecednim redom.

Svaka naredba, komanda ili funkcija prikazana je navođenjem najpre **oblika,** zatim **tipa, objašnjenja** i na kraju **primera.**

Oblik definiše dozvoljene konstrukcije BASIC-a.

Tip može biti: komanda, naredba, numerička funkcija, azbučna funkcija, logički operator.

Objašnjenje je kratko, jedna do dve rečenice i ne zalazi u detalje.

Primer služi samo kao ilustracija mogućeg oblika i načina upotrebe.

Pri definisanju oblika sve ključne reči BASIC-a date su velikim slovima.

Malim slovima uokvireno simbolima < >, označeno je sve ono što pri korišćenju treba zameniti konkretnim sadržajima (bio to aritmetički izraz, broj uređaja, komentar, memorijska lokacija i sl.). Pri ovom zamenjivanju simboli <i> se uklanjaju (oni nisu simboli BASIC-a).

Svi sadržaji koji su opcioni, tj. mogu se izbaciti ili ostaviti, navedeni su u okviru uglastih zagrada [ ], koje nisu simboli BASIC-a već služe samo za opis. Njih ne treba navoditi kada se njima uokvireni sadržaji uključuju u naredbu, komandu ili funkciju.

Treba još napomenuti da male zagrade ( ) predstavljaju simbole BASIC-jezika i ostaju svuda gde su navedene.

Oblik: ABS(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunava apsolutnu vrednost.
Primer: 10 Y=ABS(X1+X2)

Oblik: <operand>AND<operand>
Tip: Logički operator
Objašnjenje: Logička konjunkcija
Primer: 10 IF X=10 AND X2=5 THEN GOTO 20

Oblik: ASC(<azbučni izraz>)
Tip: Numerička funkcija
Objašnjenje: Daje ASCII kôd prvog simbola u izrazu.
Primer: 10 X=ASC(Y$)

Oblik: ATN(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunava arctan u radijanima.
Primer: 10 Y=ATN(X1*X2)

Oblik: CHR$(<aritmetički izraz>)
Tip: Azbučna funkcija
Objašnjenje: Daje karakter čiji je ASCII kôd jednak vrednosti aritmetičkog izraza.
Primer: 10 A$=CHR$(X)

Oblik: CLOSE<logički broj teke>
Tip: Naredba
Objašnjenje: Zatvaranje teke.
Primer: 10 CLOSE 5

Oblik: CLR
Tip: Naredba
Objašnjenje: Oslobađanje memorije (program ostaje).
Primer: 10 CLR

Oblik: CMD<logički broj teke>[,azbučni izraz]
Tip: Naredba
Objašnjenje: Izraz se preusmerava sa TV na drugi uređaj. Azbučni podatak se odmah šalje (ako je prisutan).
Primer: 10 CMD 4,"ZAGLAVLJE"

Oblik: CONT
Tip: Komanda
Objašnjenje: Nastavak izvršavanja zaustavljenog programa.
Primer: CONT

Oblik: COS(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunavanje cos funkcije argumenta zadatog u radijanima.
Primer: 10 Y=COS(X*3.14/180)

Oblik: DATA<lista konstanti>
Tip: Naredba
Objašnjenje: Čuvanje podataka u okviru programa, tj. u listi.
Primer: 10 DATA 80,12,"DUNAV, SAVA"

Oblik: DEF FN<ime funkcije>(<ime promenljive>)= <aritmetički izraz>
Tip: Naredba
Objašnjenje: Definisanje funkcije koja se kasnije može koristiti u programu.
Primer: 10 DEF FN Y(X)=50*X↑2+25*X+5

Oblik: DIM<lista imena nizova>
Tip: Naredba
Objašnjenje: Rezervisanje memorijskog prostora za nizove.
Primer: 10 DIM A(100),B(300),C(10,50)

Oblik: END
Tip: Naredba
Objašnjenje: Završava program i daje poruku READY.
Primer: 1000 END

Oblik: EXP(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunavanje eksponencijalne funkcije.
Primer: 10 Y=EXP(X1*X2—X3)

Oblik: FN<ime funkcije>(<aritmetički izraz>)
Tip: Numerička funkcija

Objašnjenje: Izračunavanje vrednosti funkcije prethodno definisane sa DEF FN<ime funkcije>.
Primer: 10 PRINT FN Y(15.5)

Oblik: FOR <ime promenljive>=<početna vrednost>TO <završna vrednost>[STEP<priraštaj>]
Tip: Naredba
Objašnjenje: Ciklus koji se izvršava pod kontrolom vrednosti promenljive počevši od početne vrednosti do završne vrednosti sa korakom<priraštaj>.
Primer: 10 FOR I=1 TO 2*K STEP 5

Oblik: FRE(<ime promenljive>)
Tip: Numerička funkcija
Objašnjenje: Daje veličinu RAM prostora koji je na raspolaganju. Promenljiva se ne koristi.
Primer: 10 S=FRE(0)

Oblik: GET<lista imena promenljivih>
Tip: Naredba
Objašnjenje: Naredba ulaza koja čita tastaturu i prihvata unete simbole koji redom odgovaraju pritisnutim tasterima.
Primer: 10 GET X,X$

Oblik: GET#<logički broj teke>,<lista imena promenljivih>
Tip: Naredba
Objašnjenje: Obavlja zadatak sličan sa GET, ali uzima karaktere iz teke umesto sa tastature.
Primer: 10 GET#1,K$

Oblik: GOSUB<broj reda>
Tip: Naredba
Objašnjenje: Prelazak na potprogram.
Primer: 10 GOSUB 1000

Oblik: GOTO<broj reda>
Tip: Naredba
Objašnjenje: Bezuslovni skok na programski red sa navedenim brojem.
Primer: 10 GOTO 130

Oblik: IF<uslov>THEN<broj reda>
IF<uslov>GOTO<broj reda>
IF<uslov>THEN<naredbe>
Tip: Naredba
Objašnjenje: Naredba uslovnog prelaska. Ako je uslov ispunjen, prelazi se na naredbu sa brojem reda datim iza THEN, odnosno GOTO, ili se izvršavaju naredbe koje su eventualno navedene iza THEN (ako ove naredbe ne određuju dalje prelaze, nakon njihovog izvršenja izvršava se naredba koja neposredno sledi iza naredbe IF ... THEN).

Međutim, ako uslov nije ispunjen, odmah se prelazi na izvršavanje naredbe koja neposredno sledi iza naredbe IF ... THEN.

Primer:
```
10 IF I>=0 THEN 200
20 IF A<1 OR B=3 THEN A=0:B=0
```

Oblik: INPUT["<tekst>";]<lista imena promenljivih>
Tip: Naredba
Objašnjenje: Prihvatanje ulaznih podataka iz liste. Najpre se (ako je naveden) izdaje tekst, a zatim čeka da korisnik unese podatke.

Primer:
```
10 INPUT A,B,C$
20 INPUT "UNESITE VREDNOST ZA X";X
```

Oblik: INPUT#<logički broj teke>,<lista imena promenljivih>
Tip: Naredba
Objašnjenje: Unos podataka iz teke na isti način kao što INPUT unosi podatke sa tastature.

Primer:
```
10 INPUT#1,X1,X2
```

Oblik: INT(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunavanje celog dela vrednosti izraza.

Primer:
```
10 C=INT(A/2)
```

Oblik: LEFT$(<azbučni izraz>,<aritmetički izraz>)
Tip: Azbučna funkcija
Objašnjenje: Daje prvih <aritmetički izraz> karaktera sa leva iz azbučnog izraza.

Primer:
```
10 PRINT LEFT$(A$+B$,1)
```

Oblik: LEN(<azbučni izraz>)
Tip: Numerička funkcija
Objašnjenje: Daje broj karaktera u azbučnom izrazu.
Primer: 10 PRINT LEN(A$)

Oblik: [LET]<ime promenljive>=<izraz>
Tip: Naredba
Objašnjenje: Dodeljivanje vrednosti promenljivoj.
Primer: 10 A=15
20 LET B=10*A
30 C$="TEKST"+D$

Oblik: LIST[[<prvi red>][—][<poslednji red>]]
Tip: Komanda
Objašnjenje: Prikaz programa na ekranu.
Primer: LIST
LIST 1—80

Oblik: LOAD["<ime programa>"][,<tip uređaja>][,<tip naredbe>]
Tip: Komanda
Objašnjenje: Učitava program sa trake ili diskete u memoriju.
Primer: LOAD"PROGRAM"
LOAD"PROGRAM",8,1

Oblik: LOG(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunavanje prirodnog logaritma.
Primer: 10 PRINT LOG(A/B)

Oblik: MID$(<azbučni izraz>,<aritmetički izraz—1>[,<aritmetički izraz—2>])
Tip: Azbučna funkcija
Objašnjenje: Daje deo azbučnog izraza koji počinje od pozicije koju daje aritmetički izraz—1, a sadrži <aritmetički izraz—2> uzastopnih simbola.
Primer: 10 PRINT MID$(A$,2,5)

Oblik: NEW
Tip: Komanda

Objašnjenje: Briše program i sve promenljive iz memorije.
Primer: NEW

Oblik: NEXT[<ime promenljive>][,<ime promenljive>]...
Tip: Naredba
Objašnjenje: Završetak ciklusa započetog sa FOR.
Primer: 10 NEXT I
20 NEXT I,J,K

Oblik: NOT<operand>
Tip: Logički operator
Objašnjenje: Logička negacija.
Primer: 10 IF NOT A=A1 THEN 20

Oblik: ON<aritmetički izraz>GOTO<lista brojeva redova>
Tip: Naredba
Objašnjenje: Bezuslovni skok prema vrednosti aritmetičkog izraza.
Primer: 10 ON S GOTO 10,20,30

Oblik: ON<aritmetički izraz>GOSUB<lista brojeva redova>
Tip: Naredba
Objašnjenje: Prelaz na potprogram prema vrednosti aritmetičkog izraza.
Primer: 10 ON S GOSUB 50,100,150

Oblik: OPEN<logički broj teke>,<tip uređaja>[,<dopunski parametri>]
Tip: Naredba
Objašnjenje: Otvara kanal za ulaz/izlaz sa periferijskih uređaja.
Primer: 10 OPEN 1,4
10 OPEN 1,1,1,"IME"
10 OPEN 5,8,5,"DISKETA,S,W"

Oblik: <operand>OR<operand>
Tip: Logički operator
Objašnjenje: Logička disjunkcija.
Primer: 10 IF A>2 OR A<=2 THEN 100

Oblik: PEEK(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Daje sadržaj memorijskog registra sa adresom <aritmetički izraz>.
Primer: 10 PEEK(197)

Oblik: POKE<aritmetički izraz>,<aritmetički izraz>
Tip: Naredba
Objašnjenje: Upisuje vrednost desnog aritmetičkog izraza u memorijski registar sa adresom koju određuje levi aritmetički izraz.
Primer: 10 POKE 197,A
20 POKE A,B+C

Oblik: POS(<ime promenljive>)
Tip: Numerička funkcija
Objašnjenje: Daje trenutni položaj kursora. Promenljiva se ne koristi.
Primer: 10 PRINT POS(X)

Oblik: PRINT<lista>
Tip: Naredba
Objašnjenje: Izdavanje podataka.
Primer: 10 PRINT X,Y,Z ↑ 2—F;A$;"KRAJ"

Oblik: PRINT#<logički broj teke>,<lista>
Tip: Naredba
Objašnjenje: Slanje informacija periferijskim uređajima.
Primer: 10 PRINT#2,A$;B$

Oblik: READ<lista imena promenljivih>
Tip: Naredba
Objašnjenje: Promenljivima iz liste dodeljuje redom vrednosti navedene u DATA naredbi.
Primer: 10 READ X,Y$

Oblik: REM[<komentar>]
Objašnjenje: Omogućava pisanje komentara u okviru programa. Ne izvršava se.
Primer: 10 REM *** NASLOV ***

Oblik: RESTORE
Tip: Naredba

Objašnjenje: Vraća interni pokazivač na prvi podatak u DATA naredbi.
Primer: 100 RESTORE

Oblik: RETURN
Tip: Naredba
Objašnjenje: Izlaz iz potprograma.
Primer: 100 RETURN

Oblik: RIGHT$(<azbučni izraz>,<aritmetički izraz>)
Tip: Azbučna funkcija
Objašnjenje: Daje poslednjih <aritmetički izraz> simbola iz azbučnog izraza.
Primer: 10 PRINT RIGHT$(A$,2)

Oblik: RND(<broj>)
Tip: Numerička funkcija
Objašnjenje: Daje pseudoslučajni broj iz intervala (0,1). Od argumenta <broj> koristi se samo znak
Primer: 10 PRINT RND(—1)

Oblik: RUN[<broj reda>]
Tip: Komanda
Objašnjenje: Komanda za početak izvršavanja programa koji se nalazi u memoriji.
Primer: RUN
RUN 100

Oblik: SAVE["<ime programa>"][,<tip uređaja>]
[,<tip naredbe>]
Tip: Komanda
Objašnjenje: Smeštanje programa iz memorije na traku ili disketu u vidu teke.
Primer: SAVE"PROGRAM",8

Oblik: SGN(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Daje 1 ako je aritmetički izraz pozitivan, nulu ako je vrednost aritmetičkog izraza nula, a —1 ako je aritmetički izraz negativan.
Primer: 10 PRINT SGN(X1*X2)

Oblik: SIN(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunava vrednost sin funkcije.
Primer: 10 Y=SIN(X ↑ 2)

Oblik: SPC(<aritmetički izraz>)
Tip: Azbučna funkcija
Objašnjenje: Izdavanje <aritmetički izraz> blanko karaktera.
Primer: 10 SPC(10)"POMERENO"

Oblik: SQR(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunava kvadratni koren.
Primer: 10 Y=SQR(X1*X2)

Oblik: STATUS
Tip: Numerička funkcija
Objašnjenje: Daje status reč poslednje ulazno/izlazne operacije.
Primer: 100 IF STATUS>0 THEN 1000

Oblik: STEP<aritmetički izraz>
Tip: Naredba
Objašnjenje: Definisanje koraka u FOR naredbi.
Primer: 10 FOR I=10 TO 0 STEP—1

Oblik: STOP
Tip: Naredba
Objašnjenje: Zaustavljanje izvršavanja programa.
Primer: STOP

Oblik: STR$(<aritmetički izraz>)
Tip: Azbučna funkcija
Objašnjenje: Pretvaranje broja (koji predstavlja vrednost aritmetičkog izraza) u oblik azbučne konstante.
Primer: 10 PRINT STR$(X)

Oblik: SYS<memorijska lokacija>
Tip: Naredba
Objašnjenje: Prelazak (iz BASIC-a) na izvršavanje mašin-

skog programa koji počinje na datoj memorijskoj lokaciji.
Primer: 10 SYS 64738

Oblik: TAB(<aritmetički izraz>)
Tip: Naredba
Objašnjenje: Naredbom TAB se određuje položaj prvog sledećeg simbola koji će se izdavati. Ovaj položaj je određen vrednošću <aritmetičkog izraza> računato od početka tekućeg reda.
Primer: 10 PRINT TAB(20)"POLOŽAJ"

Oblik: TAN(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Izračunavanje vrednosti tan funkcije.
Primer: 10 Y=TAN(X)

Oblik: TI
Tip: Numerička funkcija
Objašnjenje: Daje vreme (izraženo u šesdesetinkama sekunde) od momenta uključenja računara.
Primer: 10 PRINT TI/60

Oblik: TI$
Tip: Azbučna funkcija
Objašnjenje: Daje vreme (časovi, minuti, sekunde) od momenta uključenja računara.
Primer: 10 PRINT TI$

Oblik: USR(<aritmetički izraz>)
Tip: Numerička funkcija
Objašnjenje: Prelaz na korisnički mašinski program koji računa vrednost na bazi datog aritmetičkog izraza kao argumenta.
Primer: 10 Y=USR(X↑5)

Oblik: VAL(<azbučni izraz>)
Tip: Numerička funkcija
Objašnjenje: Pretvaranje azbučne vrednosti u numeričku.
Primer: 10 Y=VAL(X$)

Oblik: VERIFY["<ime programa>"] [,<tip uređaja>]
Tip: Komanda
Objašnjenje: Poredi program u memoriji sa programom na traci ili disketi.

Primer: VERIFY"PROGRAM",8

Oblik: WAIT<adresa>,<maska>[,<nivo za čekanje>]
Tip: Naredba
Objašnjenje: Naredba koja prekida izvršavanje programa sve dok se na datoj memorijskoj lokaciji ne prepozna željeni oblik bajta.

Primer: 10 WAIT 53273,6,6

7.2. SPISAK IZVEŠTAJA O GREŠKAMA

Abecednim redom su navedeni svi izveštaji o BASIC-greškama koje Commodore-64 može izdati. Istovremeno su ukratko opisani i uzroci koji dovode do izdavanja ovih izveštaja.

BAD DATA (pogrešni podaci). Primljen je azbučni podatak iz otvorene teke, a očekivan je numerički.

BAD SUBSCRIPT (pogrešan indeks). Indeks niza u programu je uzeo vrednost van opsega određenog DIM naredbom.

BREAK (prekid). Prekinuto je izvršavanje programa radi toga što je pritisnut RUN/STOP taster na tastaturi.

CAN'T CONTINUE (nemoguće nastaviti). Ovaj izveštaj može uslediti nakon komande CONT. Uzrok je to što je neki programski red bio izmenjen neposredno pre naredbe CONT, ili što se pojavila greška u programu, ili što program nije ni bio startovan.

DEVICE NOT PRESENT (uređaj odsutan). Traženi ulazno/izlazni uređaj nije na raspolaganju za OPEN, CLOSE, CMD, PRINT#, INPUT# ili GET# naredbe.

DIVISION BY ZERO (deljenje nulom). U programu se pojavilo deljenje nulom što je zabranjeno jer nije matematički definisano.

EXTRA IGNORED (ostatak odbačen). Previše podataka je uneto kao odgovor na INPUT naredbu. Prihvaćeno je

samo onoliko prvih koliko je zahtevala lista INPUT naredbe, a ostali su odbačeni.

FILE NOT FOUND (teka nije nađena). Ne postoji teka pod traženim imenom na disketi, ili ako je u pitanju traka, došlo se do oznake za kraj trake a da teka nije pronađena.

FILE NOT OPEN (teka nije otvorena). Nije otvorena teka koja je navedena u CLOSE, CMD, PRINT#, INPUT# ili GET# naredbi.

FILE OPEN (teka otvorena). Pokušaj da se otvori već otvorena teka.

FORMULA TOO COMPLEX (formula previše složena). Formula ima previše zagrada, ili se azbučni izraz mora rastaviti na najmanje dva.

ILLEGAL DIRECT (nedozvoljen neposredni režim). Pokušaj da se INPUT naredba upotrebi u neposrednom režimu rada.

ILLEGAL QUANTITY (nedozvoljena veličina). Prekoračenje dozvoljenog opsega za brojeve.

LOAD (učitavanje). Problem sa programom na kaseti.

NEXT WITHOUT FOR (next bez for). Promenljiva koja se pojavila u NEXT naredbi nije prethodno uvedena odgovarajućom FOR naredbom. Drugi mogući uzrok je nekorektno ukrštanje više FOR... NEXT ciklusa.

NOT INPUT FILE (nije teka za učitavanje). Pokušaj da se učitaju podaci sa teke koja je deklarisana kao teka u koju se samo upisuje sadržaj.

NOT OUTPUT FILE (nije teka za upisivanje). Pokušaj da se upišu podaci na teku koja je deklarisana kao teka iz koje se samo učitava sadržaj.

OUT OF DATA (nema više podataka). Pokušaj da se naredbom READ pročita podatak kada je već poslednji postojeći podatak iz DATA naredbe bio pročitan.

OUT OF MEMORY (nema slobodne memorije). Nema više slobodne RAM za program ili za promenljive. Takođe se može javiti u slučaju da je previše potprograma pozvano po dubini naredbama GOSUB, ili je previše FOR... NEXT ciklusa umetnuto jedan u drugog.

OVERFLOW (prekoračenje). Neka od vrednosti u programu je postala veća od najveće dozvoljene 1.70141884E++38.

REDIM'D ARRAY (ponovo dimenzionisan niz). Pokušaj da se naredbom DIM dimenzioniše niz koji je već dimenzionisan bilo ranijom DIM naredbom bilo implicitnim dimenzionisanjem na dimenziju 10 koje se vrši u slučaju korišćenja imena novog niza (koji nije bio prethodno dimenzionisan).

REDO FROM START (ponovite od početka). Ovaj izveštaj se javlja kada program očekuje numerički podatak a unese se neka druga nenumerička vrsta simbola. Treba ponoviti unošenjem ispravnog podatka.

RETURN WITHOUT GOSUB (return bez gosub). Program je u izvršavanju naišao na naredbu RETURN a da prethodno nije izvršena naredba GOSUB.

STRING TOO LONG (azbučni podatak predugačak). U programu se pojavio azbučni podatak sa više od 255 simbola što predstavlja maksimalnu dužinu.

SYNTAX ERROR (sintaksna greška). Računar nije prepoznao komandu, naredbu ili funkciju radi toga što je zapisana u nekorektnom (nedozvoljenom) obliku.

TYPE MISMATCH (ne slaže se tip). Upotrebljen je azbučni podatak umesto numeričkog ili obrnuto.

UNDEF'D FUNCTION (nedefinisana funkcija). Pokušaj da se sa FN upotrebi funkcija koja prethodno nije definisana naredbom DEF FN.

UNDEF'D STATEMENT (nedefinisana naredba). Pokušaj da se sa RUN, GOTO ili GOSUB pređe na broj reda koji ne postoji.

VERIFY (verifikovati). Program na disketi ili traci nije identičan programu koji se nalazi u memoriji.

7.3. TABELA ASCII KODOVA

SIMBOL	KOD	SIMBOL	KOD	SIMBOL	KOD	SIMBOL	KOD
	0		24	0	48	H	72
	1		25	1	49	I	73
	2		26	2	50	J	74
	3		27	3	51	K	75
	4	{crvena}	28	4	52	L	76
{bela}	5	{desno}	29	5	53	M	77
	6	{zelena}	30	6	54	N	78
	7	{plava}	31	7	55	O	79
ONEMOG shift ⇄	8	{blanko}	32	8	56	P	80
OMOGUC shift ⇄	9	!	33	9	57	Q	81
	10	"	34	:	58	R	82
	11	#	35	;	59	S	83
	12	$	36	<	60	T	84
{return}	13	%	37	=	61	U	85
{mala slova}	14	&	38	>	62	V	86
	15	'	39	?	63	W	87
	16	(	40	@	64	X	88
{dole}	17	)	41	A	65	Y	89
{rvs on}	18	*	42	B	66	Z	90
{home}	19	+	43	C	67	[	91
{del}	20	,	44	D	68	£	92
	21	-	45	E	69	]	93
	22	.	46	F	70	↑	94
	23	/	47	G	71	←	95

SIMBOL	KOD	SIMBOL	KOD	SIMBOL	KOD	SIMBOL	KOD
	96		120	(crna)	144		168
	97		121	(gore)	145		169
	98		122	(rvs off)	146		170
	99		123	(clr)	147		171
	100		124	(inst)	148		172
	101		125	(smedja)	149		173
	102		126	(sv.crvena)	150		174
	103		127	(siva 1)	151		175
	104		128	(siva 2)	152		176
	105	(orandz)	129	(sv.zelena)	153		177
	106		130	(sv.plava)	154		178
	107		131	(siva 3)	155		179
	108		132	(purpurna)	156		180
	109	f1	133	(levo)	157		181
	110	f3	134	(zuta)	158		182
	111	f5	135	(plavozel.)	159		183
	112	f7	136	(blanko)	160		184
	113	f2	137		161		185
	114	f4	138		162		186
	115	f6	139		163		187
	116	f8	140		164		188
	117	(shift/ret)	141		165		189
	118	(vel.slova)	142		166		190
	119		143		167		191

KODOVI	192-233	ISTI KAO	96-127
KODOVI	224-254	ISTI KAO	160-190
KODOVI	255	ISTI KAO	126

LITERATURA

1. Angerhausen M. i dr.: The Anatomy of the Commodore 64, Abacus Software, USA 1984.

2. Beggining Self-Teaching Computer Lessons, P. C. Publications.

3. Boris Allan: Graphic Art on the Commodore 64, Sunshine Books, London, 1984.

4. CBM Professional Computer Guide, Osborne/McGraw-Hill.

5. Clive Prigmore: 30 Hour BASIC, BBC Publications.

6. Commodore 1541, Disc Drive, Commodore Business Machines, 1983.

7. Commodore 64, Programmers Reference Guide, Commodore Business Machines, 1983.

8. Commodore 64, User Manual, Commodore Business Machines, 1983.

9. David E. Simon: BASIC From the Ground Up, Hayden Book Co, 1984.

10. David Lawrence: The Working Commodore 64, Sunshine Books, London, 1984.

11. Dwyer, Critchfield: Basic and the Personal Computer, Addison-Wesley.

12. Gary W. Orwig, William S. Hodges: The Computer Tutor: Learning Activities for Homes and Schoos Little, Brown & Co.

13. Jonathan Inglis: The Friendly Computer Book, BBC Publications, London, 1983.

14. Koffman E. B., Friedman F. L.: Problem Solving and Structured Programming in BASIC, Addison-Wesley Publishing Company, 1979.

15. Lothar English, Norbert Szczepanowski: The Anatomy of the 1541 Disc Drive Abacus Software, USA, 1984.

16. Morton J. B.: Introduction to BASIC, Pitman Publisher.

17. Mladenović N., Grbović R., Petrović V.: Kućni kompjuteri, algoritmi i programi, Tehnička knjiga, Beograd, 1985.

18. Datey M. J., Payne C.: BASIC: A Short Self-Instructional Course, Pitman Publisher, 1982.

19. Operating Instructions for Your 1530 Datasette, Commodore Electronics Ltd., 1983.

20. Parezanović N., Janković B.: Programski jezik BASIC, Privredno-finansijski vodič, Beograd, 1978.

21. Peter Bishop: Computer Programming in BASIC, Thomas Nelson and Sons Ltd., London, 1981.

22. Peter Falconer: Commodore 64 Sound ang Graphics, Melbourne House, 1984.

23. Richard Freeman: Structured Basic, BBC Publications, London 1983.

24. Stojković V., Tošić D.: Zbirka zadataka iz programiranja, programski jezik BASIC, Naučna knjiga, Beograd, 1982.

25. T. A. Dwyer, M. Critchfield: BASIC and the Personal Computer, Telmas Courseware Ratings.

26. Thom Hogan: Osborne CP/M User Guide, Osborne/McGraw-Hill, London, 1983.

27. W. Amsbury: Structured BASIC and Beyond, Pitman Publisher.

SADRŽAJ

1. UVOD — — — — — — — — — — — — — 7

2. OSNOVNE MOGUĆNOSTI BASIC-a — — — — — 10
2.1. Uvod — — — — — — — — — — — — — 10
2.2. Ekranski editor — — — — — — — — — — — 16
2.3. BASIC-jezik i interpretator — — — — — — — 17
2.4. Neposredni i programski režim rada — — — — — 18
2.5. Naredba PRINT u neposrednom režimu — — — — 18
2.6. Aritmetički operatori i numeričke konstante — — — 19
2.7. Azbučne konstante i njihovo spajanje — — — — — 23
2.8. Programski režim rada, broj reda, RUN, LIST, NEW — 24
2.9. Promenljive — numeričke i azbučne — — — — — 27
2.10. Dodeljivanje vrednosti, aritmetički i azbučni izraz — — 28
2.11. Unos podataka sa tastature — INPUT — — — — — 31
2.12. Zaustavljanje i nastavak izvršavanja programa, STOP, END, RUN/STOP, CONT — — — — — — — — 33
2.13. Grananje programa — uslovni i bezuslovni prelaz, IF ... THEN, GO TO — — — — — — — — — — 34
2.14. Programski ciklus — FOR ... NEXT — — — — — 41
2.15. Numeričke funkcije ABS, INT, SGN, SQR, EXP, LOG, RND, SIN, COS, TAN, ATN — — — — — — — 43
2.16. Azbučne funkcije ASC, LEN, VAL, STR$, CHR$, LEFT$, MID$, RIGHT$ — — — — — — — — — — 47
2.17. Komentari u okviru programa — REM — — — — 49
2.18. Smeštanje programa na spoljnu memoriju — SAVE, i njihovo učitavanje — LOAD — — — — — — — 50

3. DALJE MOGUĆNOSTI BASIC-a — — — — — — 52
3.1. Nizovi, DIM — — — — — — — — — — — 52
3.2. Programska datoteka, DATA, READ, RESTORE — — 55
3.3. Korisnički definisane funkcije, DEF FN, FN — — — 56
3.4. Potprogrami, GO SUB, RETURN — — — — — — 58
3.5. Višestruko grananje — — — — — — — — — — 60
3.6. Dodatne mogućnosti pri unošenju i izdavanju podataka, GET, PRINT, POS, SPC, TAB — — — — — — — 61
3.7. Direktan upis i čitanje memorije, raspored ROM-a i RAM-a, PEEK, POKE, FRE, CLR, WAIT — — — — 66

3.8. Korišćenje mašinskih potprograma iz BASIC-a, SYS, USR 74

4. PERIFERIJSKI UREĐAJI — — — — — — — — — 77
4.1. Uvod — — — — — — — — — — — — — — 77
4.3. Kasetofon — — — — — — — — — — — — 80
4.3.1. Organizacija podataka na kaseti — — — — — — 80
4.3.2. Kasetofon DATASSETTE 1530 — — — — — — 81
4.4. Štampač — — — — — — — — — — — — 85
4.4.1. Matrični štampači — — — — — — — — — — 85
4.4.2. Štampač MPS 801 — — — — — — — — — — 87
4.5. Disketna jedinica — — — — — — — — — — 92
4.5.1. Organizacija podataka na disketi — — — — — — 92
4.5.2. Opšte napomene o rukovanju disketama — — — — 95
4.5.3. Commodore 1541 — karakteristike — — — — — — 95
4.5.4. Rad sa programima — — — — — — — — — 97
4.5.5. Korišćenje komandnog kanala — — — — — — — 98
4.5.6. Rad sa podacima — — — — — — — — — — 102

5. ZVUK I MUZIKA — — — — — — — — — — 117
5.1. Uvod — — — — — — — — — — — — — — 117
5.2. Osnovni pojmovi u vezi sa zvukom — — — — — — 118
5.3. Generator zvuka — — — — — — — — — — — 119
5.4. Inicijalizacija — — — — — — — — — — — 121
5.5. Postavljanje jačine zvuka — — — — — — — — — 121
5.6. Postavljanje frekvencije — — — — — — — — — 122
5.7. Određivanje tipa zvučnog signala — — — — — — 123
5.8. Definisanje obvojnice — — — — — — — — — — 126
5.9. Uključivanje zvuka — — — — — — — — — — 129
5.10. Filtar i preostale mogućnosti — — — — — — — — 130
5.11. Commodore 64 kao muzički instrument — — — — 133
5.12. Sažeti prikaz karakteristika generatora zvuka — — — 138

6. GRAFIKA — — — — — — — — — — — — — 140
6.1. Uvod i osnovni pojmovi — — — — — — — — — 140
6.2. Osnovni elementi za formiranje slike — — — — — 142
6.3. Prikazivanje teksta — — — — — — — — — — 147
6.4. Korisnički definisani karakteri — — — — — — — — 153
6.5. Grafika visoke rezolucije — — — — — — — — — 158
6.6. Sprajtovi — — — — — — — — — — — — 168

7. DODACI — — — — — — — — — — — — — 182
7.1. BASIC za Commodore 64 (pregled komandi, naredbi i funkcija) — — — — — — — — — — — — — 182
7.2. Spisak izveštaja o greškama — — — — — — — — 193
7.3. Tabela ASCII kodova — — — — — — — — — — 196

LITERATURA — — — — — — — — — — — — 198

Mr Veljko Spasić, dipl. mat. i Dušan Veljković, dipl. mat.: BASIC za mikroračunare (Commodore 64) — Recenzent: prof. dr Nedeljko Parezanović — Za izdavače: Branko Nikolić, direktor i glavni urednik i prof. mr Velimir Branković, direktor i glavni urednik — Urednici: Dušica Lučić i Svetislav Todić — Grafički urednik: Jugoslav Bogdanović — Korektor: Mirjana Aćimović — Izdavači: NIRO "Tehnička knjiga", Beograd, 7. juli 26 i Zavod za izdavanje udžbenika — OOUR Stvaranje i proizvodnja nastavnih sredstava, Beograd, Obilićev venac 5 — Štamparija: BIGZ, Beograd, Bulevar vojvode Mišića 17 — Tiraž: 5.000 primeraka — Oslobođeno poreza na promet na osnovu mišljenja Republičkog komiteta za kulturu SRS

YU ISBN 86-325-0010-4

Računari, njihova primena i programiranje, naglo izlaze iz zatvorenih profesionalnih krugova i brzo prodiru do velikog broja ljudi. Ova, nesumnjivo, pozitivna tendencija demokratizacije računarstva i informatike omogućena je tek kada su proizvedeni računari malih dimenzija, velikih mogućnosti i veoma niske cene.

Paralelno s povećanjem broja zainteresovanih za mikroračunare, kao i masovnosti njihovih sadašnjih i potencijalnih vlasnika, raste i potreba za odgovarajućom stručnom literaturom, koje na našem jeziku nema dovoljno. Zbog toga i smatramo da se ova knjiga pojavljuje u momentu kada može biti korisna.

Knjiga bi trebalo da doprinese savlađivanju programskog jezika BASIC, korišćenju periferijskih uređaja, kao i osnovama programiranja grafike i zvuka na mikroračunaru Commodore 64. Namenjena je svima koji su rešili da se upoznaju sa BASIC-jezikom, a posebno onima koji imaju ili će nabaviti računar Commodore 64.

Citaocu nije potrebno predznanje iz računarstva, matematike ili elektronike.

YU ISBN 86-325-0010-4