Izdaje BIGZ

OOUR „Duga"

računari 7

Specijalno izdanje
avgust/septembar '85.

časopisa „Galaksija"
cena 250 dinara

naš test
štampač „kanon"

računari u izlogu
komodor 128

ekskluzivno
velikani na rasprodaji

periferijska oprema
svetlosna pera

umetnost programiranja
hakeri na usijanom limenom krovu

„komodor"
periferije iz mašinca

računari u akciji
baze podataka

umetak
32 strane

SVE SPEK TRUMOVE RUTINE

računari

Izdaje BIGZ

OOUR „Duga"

časopisa „Galaksija"
cena 250 dinara

Specijalno izdanje
avgust/septembar '85.

Izdaje
Beogradski izdavačko-grafički zavod
OOUR Novinska delatnost „Duga"
11000 Beograd
Bulevar vojvode Mišića 17

Telefoni
650-161 (redakcija)
650-528 (prodaja)
651-793 (propaganda)

Generalni direktor
Dobrosav Petrović

Direktor OOUR „Duga"
Bratoljub Babić
Glavni i odgovorni urednik
Gavrilo Vučković
Urednik izdanja
Jova Regasek

Likovna i grafička oprema:
Raša Golubović

Redakcija časopisa „Galaksija"
Tanasije Gavranović, pomoćnik glavnog i odgovornog urednika
Esad Jakupović, zamenik glavnog i odgovornog urednika
Aleksandar Milinković, urednik
Jova Regasek, urednik
Zorka Simović, sekretar redakcije
Srđan Stojančev, novinar
Gavrilo Vučković, glavni i odgovorni urednik

Stručna saradnja
Dejan Ristanović
Nevenka Spalević
Anđelko Zgorelec
Mihajlo Tešević

Autori tekstova
Vladimir Krstanović
Mihajlo Karapandžić
Zvonimir Vistrička
Branko Đaković
Vladimir Kostić
Ivan Nador, dipl. ing.
Borivoj Perzić
Bogdan Petrović
Dejan Ristanović
Jelena Rupnik
Duško Savić
Jovan Skuljan
Zoran Životić

Crteži
Miša Marković

Tehnička saradnja
Ljubiša Milovanović
Ljev Rjadčenko

Prevodioci
Esad Jakupović
Ksenija Pješčić-Lebedinski
Domagoj Bačić

Izdavački savet „Galaksije"

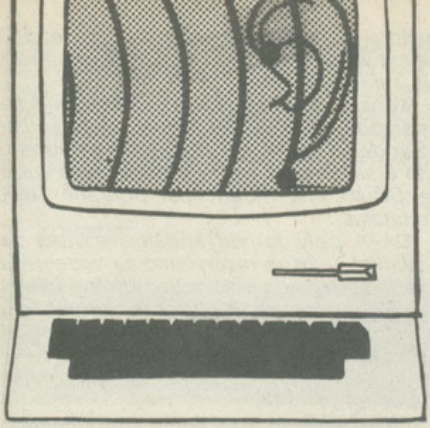
Dr Rudi Debijađi, prof. dr Branislav Dimitrijević (predsednik), Radovan Drašković, Tanasije Gavranović, Živorad Glišić, Esad Jakupović, Velizar Maslać, Nikola Pajić, Željka Perunović,, prof. dr Momčilo Ristić, Vlada Ristić, dr inž. Milorad Teofilović, Vidojko Veličković, Velimir Vesović, Miloje Vuković

Štampa
Beogradsko izdavačko-grafički zavod
11000 Beograd, Bulevar vojvode Mišića 17

Žiro-račun kod SDK 60802-833-2463
Devizni račun kod Beobanke
60811-620-6-82701-999-01066
Za inostranstvo cena dvostruka (400 D, 2,50 US$, 6.50 DM, 45 Sch, 5,50 Sfrs, 20 FFRS)

Na osnovu mišljenja Republičkog sekretarijata za kulturu broj 413-77/72-03 i „Službenog glasnika" broj 26/72, ovo izdanje oslobođeno je poreza na promet

Sadržaj



šta ima novo

„Galaksija plus"

Godinu i po dana nakon pojave računara „galaksija", u prostorijama Zavoda za učila i nastavna sredstva u Beogradu održana je demonstracija „galaksije plus", koja je takođe nastala kao proizvod saradnje Zavoda, Elektronike inženjering iz Zemuna i časopisa „Galaksija". Autori hardverski i softverski proširene „galaksije plus" su Milan Tadić i Nenad Dunjić, koji su čitaocima poznati već po tome što su „galaksiju" obogatili grafikom visoke rezolucije.

Već i na prvi pogled, nova „galaksija" razlikuje se od stare: kutija u koju je računar smešten nešto je veća, i na njoj se nalaze izvodi za priključak ispravljača, kasetofona, televizora, monitora i audio pojačivača, a ugrađen je i konektor za priključenje perifernih jedinica, odnosno jedinica za priključenje novih jezičkih procesora ili aplikativnog softvera u ROM-u. Profesionalna tastatura, inače proizvod IEVT iz Ljubljane, sadrži i posebne tastere za č, ć, ž i š, što takođe predstavlja novinu.

„Galaksija plus" ima ugrađena dva ROM-a. U prvom ROM-u, kapaciteta 8 K smešten je bejzik interpreter i standardni Z80 asembler, dok se u drugom (kapaciteta 2 K) nalazi ekranski editor. Osim toga, „galaksija plus" sadrži 48 K dinamičkog RAM-a. Pored proširenja memorije i generatora zvuka, tu je i fina grafika (256–208 tačaka) sa mogućnošću prikazivanja više slika koje se mogu smenjivati u svakom „frame" intervalu, ili jedna virtuelna slika koja se prikazuje sa mekim pomeranjem, kao i poaljenje, gašenje i testiranje svake tačke.

Treba spomenuti i softverski izmenljiv set karaktera (mala slova, matematički simboli, grčki alfabet, itd.), kao i mogućnost prikazivanja više setova simultano na ekranu. Od priključaka novine predstavljaju poboljšani kasetni ulaz/izlaz, audio izlaz i dva paralelna osmobitna input/output porta.

Nema sumnje da nova verzija „galaksije" predstavlja znatno poboljšanje u odnosu na prethodnu. Za sada postoji samo nekoliko primeraka ovog računara, ali bi trebalo očekivati da će i „galaksija plus" uskoro naći svoju primenu, pre svega u osnovnim i srednjim školama. U sledećem broju objavljujemo detaljan prikaz nove „galaksije".

Istočne jabuke

Prema „Mikroszamitogep magazin"-u (Mikroračunarski magazin), časopisu za mađarske ljubitelje računara, u Mađarskoj se 1984. tek stidljivo razmišljalo o razvoju računara koji bi bio kompatibilan sa „eplom II."

Međutim, u Bugarskoj su se za taj pravac odlučili mnogo ranije, pa su već 1982. ponuđena tržištu dva mikroračunara — IMKO-2 i Pravec-82. Ova dva računara su bila namenjena, pre svega, školama, a proizvedeno je više hiljada komada. Prošle godine svetlo dana su ugledala dva nova i snažnija modela — Pravec-8B i Pravec-8M. Ova dva računara uz svoj imidž imaju i etiketu „za profesionalnu namenu".

U SSSR proizvodi se mikroračunar AGAT, takođe kompatibilan sa „eplom II", ali baziran na sovjetskoj tehnologiji, tj. na komponentama koje su proizvedene u Sovjetskom Savezu.

U Mađarskoj pak, pod imenom AX-II, u 1984. pojavljuje se mikroračunar koji je napravilo Omladinsko istraživačko društvo. Ipak glavni događaj, da ne kažemo događaj sezone, biće pojava prvih 100 komada mikroračunara (takođe kompatibilnih sa „eplom IIe") po imenu IVEL Z-3.

Ovaj mikroračunar u srcu ima jedan Z80 i dva Rockwell 6502 mikroprocesora. Opremljen je „epl" DOS-om 3.2/3.3 i uz sve to i CP/M i USCD. U sklopu mikroračunara je disk jedinica sa dva drajva.

Ipak glavno „osveženje" ostavljeno je za kraj. Ovaj mikroračunar se proizvodi u kooperaciji, pri čemu Videoton Rt radi terminalni deo, a računarski deo jugoslovenska firma IVASIM ! ! !

Računaru se prognozira veoma svetla budućnost, pri čemu mu se najavljuje cena od oko pola miliona forinti, što, prema mađarskim kursnim listama, iznosi 1.9 miliona dinara.

I. Nador

Računari i žene

Stara i veoma raširena rasprava o tome da li računari i računarski sistemi utiču negativno na čovekovo zdravlje obogaćena je ovih dana sa dva nova priloga — iz Japana i Velike Britanije.

Prema saopštenju Glavnog veća japanskih Trejd Uniona, više od jedne trećine žena koje rade za računarskim terminalima je imalo problema za vreme porođaja. Ispitivanjem je bilo obuhvaćeno 250 žena koje su zatrudnile ili se porodile za vreme ili nakon rada na terminalima. Od trudnih žena koje su radile za terminalom 6 časova ili duže, dve trećine je imalo problema sa porođajem. Od onih koje su radile tri do četiri časa, probleme je imalo 46%, a od onih koje su radile manje od sata dnevno probleme je imalo 25%.

I pored obimnog istraživanja, nisu pronađeni konkretni faktori koji utiču na ovakve rezultate. Glavno veće ipak sugeriše da ni muški ni ženski računarski operatori ne bi trebalo da provode više od četiri časa dnevno za video terminalima.

Gotovo istovremeno Britanska vlada je u specijalnoj izjavi, koja se tiče tog istog problema, saopštila da ne postoje konkretni dokazi da video terminali negativno utiču na tok trudnoće i porođaj. U specijalizovanom izveštaju vladine agencije koja se bavi tim problemom takođe stoji da su nagomilavanja broja pobačaja među ženama koje rade na terminalima verovatno rezultat statističkih anomalija.

Mada u ovome, izgleda, ima mnogo više dima nego vatre, Bilo bi interesantno videti da li se i kod nas neko brine za problem trudnih žena koje rade za terminalom. Istina, kod nas nema toliko računara da bi to bila raširena pojava, ali zdravlje je zdravlje, tim pre što se zna da kod nas na unošenju podataka preko terminala rade gotovo isključivo žene.

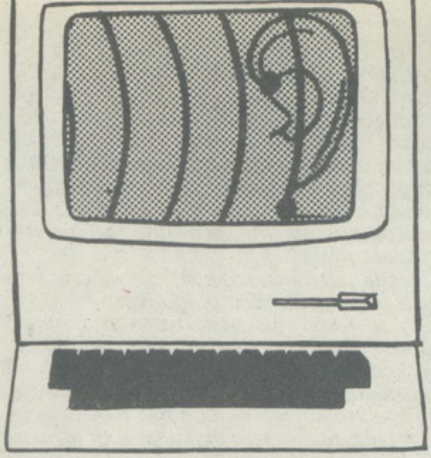
B. Đaković

„Komodorovo novo ruho

Oni koji su bar jednom videli velike računarske sisteme kako rade sa hard diskovima znaju kakva je razlika između njih i običnih disk drajvova. Sada vlasnici „Komodora" imaju to zadovoljstvo da kažu da i njihov računar ima neke odlike velikog računara. Mala računarska firma COMPUTER SPECIALTIES napravila je prvi hard disk za „komodor 64" sa oznakom CSI ST 10C. Disk je kapaciteta 10 megabajta, što predstavlja oko 50.000 strana gusto kucanog teksta, a brzina prenosa podataka je prosto fantastična — za nekoliko stotih delova sekunde popunjava se čitava memorija od 64K. Disk koristi serijski ili IEEE bas za prenos podataka. Jedina, ali za nas sasvim dovoljna mana je — cena. Disk košta tačno — 1.595 dolara ili oko 478.000 dinara. Ako vas ovo nije ohladilo, za detaljnije informacije možete pisati na adresu: CSI, P.O. 1718, Melbourne, Florida 32902.

Interfejs za povezivanje „komodora 64" sa „eplom II+" omogućava vlasnicima „komodora" da koriste sav postojeći softver i hardver za „epl II+", a toga ima dosta. Mogu se koristiti „eplovi" diskovi i sve one silne kartice (slotovi) koji su od „epla" napravili najraširenij računar u svetu. Ni ovaj interfejs nije bez ozbiljnih mana — košta oko 600 dolara, što je, pri trenutnom kursu dinara, zaista prilično mnogo. Za sve informacije obratite se na: MIMIC SYSTEMS INC. 1112 Fort ST, Victoria B C, CANADA V 8 V 4 V 2.

Treća, za nas možda i najinteresantnija novost je program koji na „komodoru 64" simulira „ZX spektrum". Pokazuje se da je „komodor 64" dovoljno brz računar da može da simulira rad nekog manjeg računara, što može biti vrlo interesantno. Ljudi koji su napravili ovaj program tvrde da će svi bejzik programi raditi sasvim normalno, čak i istom brzinom. Uz to, pošto i „komodor" i „spektrum" imaju RS-232, možete koristiti i neke „spektrumove" dodatke, kao, na primer, ZX interfejs II. Jedino treba probati kako ovaj program radi sa nekim ozbiljnijim bejzik programima, koji se oslanjaju na grafiku. Ako ste zainteresovani za ovaj interesantan program, obratite se bilo kom beogradskom piratu. Program već kruži Beogradom. Živeli MALI pirati!



šta ima novo

Novi „amstrad"

Pojavom prvog modela tvrtke Amstrad — šarenog računala CPC 464 — i njegove ozbiljnije obojene zapadno-njemačke kopije tvrtke Schneider počela je dugo očekivana prekretnica u razvoju integriranih sistema kućnih računala. Za svega 1400 DM moguće je nabaviti računalo s ugrađenim kazetofonom te frlo kvalitetnim RGB monitorom, a za 900 DM istu konfiguraciju s zelenim monitorom. Na tržištu je ovo računalo manje od godine dana, a već ima široki krug korisnika. Tvrtka Amstrad, svjesna svog poslovnog uspjeha, na tržište izbacuje novi model CPC 664 koji, za razliku od prethodnog modela, ima ugrađenu diskjedinicu umjesto kazetofona. Istovremeno konstruktori, „amstrada" najavljuju nove modele koji će biti softverski kompatibilni s prethodnim modelima, a zna se što to znači za plasman i korištenje proizvoda ove tvrtke te buduće korisnike — obilje višestruko iskoristivog softvera na izvanrednom hardveru.

Cijena novog računala ostaje u stilu „svega 400 funti", što je jeftinije od ranijeg CPC 464 i disk jedinice. Ugrađeni CP/M operativni sistem osigurava veliku softversku podršku, pa je i to prilog rentabilnosti ovog računala.

Računalo CPC 664 radi s procesorom Z 80A (4MHz). Memorija je i dalje 64K od kojih je 41K upotrebljiv za korisnika. Računalo koristi i izvjesne mogućnosti „preklapanja" ROM memorije koja ima veličinu 32K, u kojoj je sadržan poboljšani Locomotiv bejzik i operativni sistem. Takva konfiguracija memorije omogućuje vrlo složeno programiranje.

Tastatura je ukusnije obojena sivo — svijetlo plava (neke tipke). Sadrži 74 tipke standardnog QWERTY rasporeda, u svemu ista kao i kod ranijeg Amstradovog modela; 32 tipke mogu biti posebno definirane od korisnika.

Zvuk sa računala se generira zvučnim čipom AY-3-8192. Na raspolaganju stoje tri kanala s rasponom od 8 oktava. Svaki kanal može biti nezavisno namješten na određeni ton i amplitudu, odnosno na lijevi, desni i centralni signal, što se može koristiti kao stereo ton preko posebnog izlaza. Jačina zvuka iz ugrađenog zvučnika je promjenljiva, a predstavlja miksani monoauralni izlaz.

Uz računalo je moguće koristiti mnogo dodatnog hardvera. Oni naviknuti na kazetofon mogu ga priključiti na poseban port. Amstradov matrični printer DMP-1, novi džojstik JY-2, te druga disk jedinica FD-1 predstavljaju osnovnu dodatnu konfiguraciju. Naravno, moguće je koristiti i proizvode drugih tvrtki. Oni koji ne žele nabavljati monitor i dalje imaju na raspolaganju modulator MP-2 preko kojeg mogu računalo priključiti na običan kolor TV-prijemnik. Inače, računalo ima izvedeni RGB i kompozitni video izlaz.

Locomotiv bejzik je poboljšan, ali kompatibilan s bejzikom na CPC 464. Glavna prednost poboljšanog bejzika su vrlo brze „area fill" naredbe, odnosno poboljšanje se uglavnom odnose na komande za rad s diskom. Također postoji i nova interapt sabrutina vezana uz instrukcije AFTER i EVERY.

Grafički modovi su ostali isti kao i na ranijem modelu.

Priključnica ima ukupno 8: za printer (paralelni Centronics), za dodatnu disk jedinicu, joystick, RGB i kompozitni video signal, kazetofon, stereo zvučni izlaz, te priključnice za napajanja (5V i 12V). Napajanje se dobija sa monitora, jer je u njemu ugrađen ispravljač.

Već letimičnim pregledom tehničkih karakteristika se vidi da je jedina bitna razlika između CPC-a 664 i CPC-a 464 samo ugrađena disk jedinica.

Ona odgovara Hitachi/Panasonic standardu od 3 inča (1 inč=2,54 cm). Cijeli sistem računala je koncipiran da može podržavati maksimalno dvije disk jedinice.

Operativni sistem na Amstradu je AMSDOS ali i CP/M. Moguće je koristiti i jezik LOGO. AMSDOS je operativni sistem koji proširuje Locomotive bejzik, a omogućuje spremanje podataka slično kao kod spremanja na traku. Upotreba CP/M-a omogućuje korištenje izuzetno velike programske podrške. Organizacija diska je takva da AMSDOS i CP/M podržavaju tri različita formata diskova : SYSTEM, DATA only te IBM format. Selekcija formata je automatska na zahtjev diska. Sva tri formata koriste isti „framework", ali različitu konfiguraciju sektora. Zajedničko za sve formate je: 512 bajtna veličina sektora, 40 STAZA te umetanje sektora u omjeru 2:1.

Konstruktori „amstrada" su se opredelili za standardni SYSTEM format: 2K se koristi za vođenje, a 9K je rezervirano za sistem. Ima 9 sektora po stazi i 2 rezervirane staze za CP/M. Ima mogućnost pohrane 169K podataka.

DATA only format koristi sve staze za podatke — 2K je rezervirano za vođenje, a ima 9 sektora po stazi i kapacitet 178K.

IBM format je sličan kao format koji koristi CP/M na IBM PC-u: 2K se koristi za vođenje, a 4K je rezervirano. Postoji 8 sektora po stazi, a 1 staza je rezervirana. Kapacitet je 154K.

Mikro diskete koje se koriste imaju kapacitet 360K (2×180K), a puno su skuplje od npr. 5,25 inčnih.

Dakle, osnovna razlika starog i novog „amstrada" je ugrađena diskjedinica, a može se priključiti i dotatna, pa čak i ona od 5,25 inča. „Amstrad CPC 664" predstavlja do sada najsretnije koncipirano računalo, a ako se u obzir uzme njegova cijena, boljeg rješenja za korisnike računala u našim uvjetima nema.

Z. Vistrička

Deset megabajta na jednoj pločici

Na nedavno održanom sajmu za IBM PC računare u Njujorku jedan od najzapaženijih proizvoda bio je tvrdi disk (hard disk) sa kapacitetom od 10 megabajta, koji je napravljen tako da izgleda kao periferna kartica, koja se direktno priključuje na jedan od slotova na IBM PC i njemu kompatibilnim računarima. Dimenzije su mu: dužina 33 cm, širina 10, a debljina svega 2,5 cm.

Ovaj proizvod, koji nosi ime HARD-CARD, predstavlja rezultat 20-tomesečnog rada kalifornijske kompanije Plus Development i japanske firme Matsushita. HARD-CARD će biti pušten u prodaju u Americi u oktobru ove godine po ceni od 1,095 dolara i proriče mu se sjajna budućnost. Višemesečno testiranje ovog tvrdog diska na računarima Compaq (PC kompatibilnim) izvršeno u poznatoj fabrici džinsa Levi Strauss u San Francisku dalo je odlične rezultate. Ideju za ovaj proizvod dao je pionir kućnih računara i video igara legendarni Nolan Busnel (Bushnell), osnivač firme Atari. Kada smo već kod Nolana, da spomenemo da

je on posle optimističkog starta digao ruke od proizvodnje personalnih robota, zbog slabe prodaje. Ali, Nolan ne posustaje — upravo je sklopio ugovor sa američkim automobilskim gigantom General Motors za isporuku specijalnih računara za kontrolu motora i orijentaciju vozila.

A. Zgorelec

Ejkorn nije umoran

I pored svih nevolja kroz koje je Acorn prolazio i prolazi, njegova „zvanična" softverska firma Acornsoft, po svemu sudeći, dobro posluje. Očekuje se da se na ovogodišnjem letnjem sajmu časopisa Acorn User Acornsoft predstavi nekoliko novih i veoma interesantnih naslova.

Radi se, pre svega, o novoj verziji njihovog tekst procesora View koja nosi oznaku 3.0. U prošlim računarima smo predstavili verziju 1.4 ovog programa i napomenuli da verziju 2.1 još nismo imali prilike da vidimo. U međuvremenu je View 2.1 pristigao u Jugoslaviju ali se, kao što to obično biva, odmah zatim pojavila nova verzija.

Acornsoft će svakako reći da View 3.0 rezultat firmine želje da stalno unapređuje svoje produkte, ali izgleda da nije baš tako: View 2.1 je pokazao nekoliko veoma ozbiljnih bagova koji se odnose na konfiguracije periferije. Da bi bio kompatibilan sa američkim verzijama BBC B, koji imaju nekoliko redova teksta na ekranu manje, View 2.1 po inicijalizaciji proverava sistemsku promenljivu HIMEM u kojoj je upisana adresa prvog bajta video memorije; ako ona nije na nekoj od vrednosti predviđenih za evropski BBC, računar radi sa manje redova na ekranu. Na nesreću, neki vlasnici BBC B su se opremili RAM tablom uz pomoć koje im sva 32 Kb osnovne memorije ostaju slobodne bez obzira na grafički mod. Kod njih je HIMEM na &8000 (kraj RAM-a) pa se View pogrešno inicijalizuje!

Osim ispravke ovoga baga, autori View 3.1 su napravili jedan odličan potez: vlasnici 6502 dodatnog procesora su do sada morali da kupuju takozvani Hi View, jer se običan View u memoriju dodatnog procesora prepisivao na adrese &8000-&BFFF, što znači da je za tekst ostajalo memorije kao u slučaju kad dodatnog procesora nema, dok je prostor od &8000-&FFFF zvrjao prazan. View 3.0 je dovoljno inteligentan da se sam relocira, prepravljajući svoje apsolutne skokove i apsolutna adresiranja. Na taj način se tekst koji pišete može prostirati na preko 40 K u bilo kom grafičkom modu.

Samo za vlasnike 6502 dodatnog procesora Acornsoft je pripremio i Elitu 2. Nova verzija je slična staroj, ali je u koloru i daleko brža, a dodata su i neka nova oružja i vrste opreme kojima treba opremiti Cobru. Ako imate BBC B, ova igra je odličan razlog da dokupite drugi procesor!

Drugi procesor, kada ga već kupite, može da posluži i za ozbiljnije stvari: Acornsoft je pripremio disketu sa Fortranom 77 i Paskalom koji su potpuno implementirani po važećim standardima (računajući kompleksne promenljive i funkcije u fortranu). Cena diskete je prava sitnica — „svega" 300 funti! View 3.0 košta samo 70 funti, a Elite, naravno, 17.95.

Dejan Ristanović

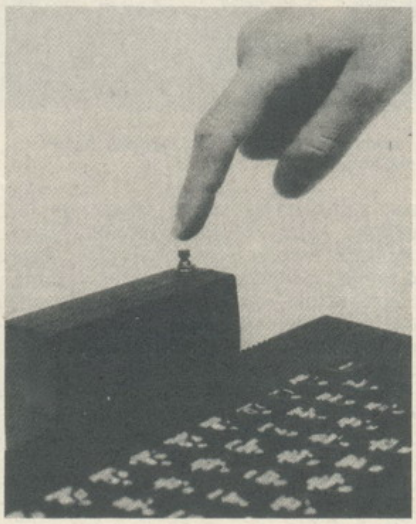
Crne vesti za pirate

Piratima je ovih dana i lakše a i sve teže da rade.

Činjenica je da pirati sve lakše kopiraju tuđe programe zahvaljujući, pored ostalog, i sve savršenijim tehničkim sredstvima koja im pomažu. Jedno od takvih sredstava je i Ivešam Mirkos Interfejs III (Evesham Micros Interface III). Upotreba je sasvim jednostavna. IMI III se priključi na računar, u računar se unese neka igra ili nešto slično, pritisne se dugme na vrhu IMI III i čitav sadržaj memorije računara odlazi na mikrodrajv, izbegavajući bilo kakvu zaštitu koja se može staviti na traku ili disketu. To je, dakle, lepa vest za naše pirate.

Drugu činjenicu, da je piratima sve teže da rade, možda bi trebalo preformulisati u: piratima će uskoro (u Velikoj Britaniji) biti jako, jako teško da rade. Predviđa se da će ove godine biti usvojen takozvani „Vilijem Pauel kompjuter kopirajt" predlog zakona, kojim se zabranjuje izrada bilo kakvih kopija izdatih programa osim za istraživanja i proučavanje. Tako će se i u softverskim izdanjima prihvatiti pojam kopirajta. Kazne za kršenje ovog zakona će biti jako oštre, što bi trebalo da sasvim određeno utiče na broj pirata u Engleskoj. Znači, crne vesti za pirate u Britaniji, ali bi mogle biti loše i za domaće pirate ako se naši zakonodavci budu ugledali na svoje britanske kolege.

B. Đaković



IBM napušta Epsona

Izgleda da su američki proizvođači kućnih računara sa zakašnjenjem shvatili da se na štampačima može razraditi dosta novaca i da je to jedna od najpopularnijih perferijskih jedinica. Dok su oni pre 5—6 godina dominirali tržištem (Centronics itd), u poslednjih nekoliko godina su dozvolili da ih Japanci preteknu i da zauzmu čak 80% američkog tržišta, koje, što se tiče štampača, godišnje vredi 7 milijardi dolara.

Sada, međutim, i IBM i Herox najavljuju ulazak na to tržište sa vlastitim štampačima!

IBM planira jeftiniji matrični štampač, nazvan „proprint", dizajniran za tri različite brzine od 40, 100 i 200 slova u sekundi. Cena u britaniji će biti 499 funti. Xerox uskoro pušta u prodaju laserski štampač velike brzine, jer se takvi štampači zbog kvaliteta otiska i brzog štampanja sve više traže.

IBM takođe, napušta prodaju Epsonovih stampača za PC računar pod svojim imenom, zbog slabe prodaje. Oni su, naime, bili nešto skuplji od indentičnih Epsonovih štampača, a samo tri magična slova na kutiji nisu bila dovoljna. Ovaj korak IBM-a neće mnogo pogoditi Epsona, jer se njegovi računari odlično prodaju i sada zauzimaju čak 30% američkog tržišta.

A. Zgorelec

Koga ne boli glava

Izgleda da ipak nije sve tako crno u svetu mikroračunara, pogotovu za proizvođače i izdavače softvera za IBM PC i njemu kompatibilne računare. Na nedavno održanom PC sajmu u Njujorku moglo je da se vidi nekoliko veoma interesantnih proizvoda, pogotovo sa područja softvera.

I londonski PC sajam bio je veoma uspešan — preko 200 izlagača i oko 15 hiljada posetioca. Jedan od trendova uočljiv sa tih sajmova je da gotovo svaki veći proizvođač nastoji da proizvede mali prenosni računar sa ravnim ekranom od rečnog kristala (liquid crystal).

Za sada se takvi računari, međutim, još uvek ne traže.

Na području softvera, mnogi izdavači su se trudili da pokažu da su njihovi integrisani programi korisniji od poznatih jednonamenskih. Ali, bar za sada, kupci izgleda više veruju jednonamenskim programima, koji su jednostavniji i rade odlično, nego komplikovanim integrisanim, koji samo zadovoljavaju.

Smatra se da će odličnu budućnost imati programi specijalno pisani za pojedine struke advokate, inženjere, farmere itd.

Na ovim izložbama bilo je i više izlagača koji smatraju da je sledeći trend da upotreba računara bude što jednostavnija, pa se mnogi utrkuju da naprave računar što sličniji „mekintošu" sa „mišom" i prozorima.

Izgleda, međutim, da je glavni razlog slabe prođe kućnih računara to što potencijalni kupci ne vide kako im oni mogu koristiti i učiniti život lakšim.

A. Zgorelec

Razglednica iz Londona

velikani na rasprodaji

Sreća u industriji kućnih računara varljivija je, reklo bi se, čak nego i u samim igrama na sreću. Vrtoglavi usponi i još strmoglaviji padovi ovde se dešavaju bukvalno preko noći. Perspektivnim preduzećima koja su naprečac otišla pod led ni broja se ne zna, a žrvnja u „točku sreće" nisu poštedena čak ni najveća imena. Za nepunih pola godine na rasprodaji su se našle i dve najveće žive legende iz domena kućnog računarstva. Pre nekoliko meseci sa svog trona u „Apple"-u svrgnut je Stiven Džobs, a sada je slična sudbina zadesila i legendarnog Klajva Sinklera, oca evropske mikroelektronske revolucije. Prodajom svoje kompanije, on je i poslednje veliko uporište biznisa u računarskoj industriji predao u ruke profesionalaca.

Kao što se i očekivalo, nije trebalo čekati dugo da se spasi najsvetlije ime britanskog mikroračunarstva. U akciju spasavanja uključile su se i poznate banke, i financijske institucije, i uticajni političari. Spasilac, međutim, nije došao iz elektronske ili kompjuterske industrije, već iz sveta biznisa.

Vitez na belom konju

Sinklerov „beli vitez", kako ga nazivaju ovdašnji listovi, je Robert Maksvel (Makswell), 60-godišnji veoma uspešni, ali često i kontroverzni biznismen. Maksvel, koji je po poreklu Čeh, došao je posle rata u Britaniju i uspeo je da za ciglih četrdeset godina postane jedan od najistaknutijih poslovnih ljudi na Ostrvu. On je vlasnik jedne od najvećih britanskih izdavačkih kuća za naučno-tehničku literaturu — Pergamon Press. Pre nekoliko godina uspeo je da spase od bankrotstva najveće britansko štamparsko preduzeće — BPPC — i pretvori ga u profitabilnu kompaniju. Prošle godine postao je i vlasnik jednog od najtiražnijih ovdašnjih dnevnika „Dejli Miror" (Daily Miror), jedinog dnevnika koji podržava laburističku stranku (Maksvel je i sam bio više godina laburistički poslanik parlamenta). On je, takođe, vlasnik i novog engleskog prvoligaša, tima Oxford city.

Kada je u maju Maksvel saznao da je Sinkler u finansijskim poteškoćama i kada je vidio da nema sigurnog kupca, odlučio je da stupi u akciju, jer, kako on kaže, „bila je u pitanju čast nacije". Za ulog od 12 miliona funti i uz preuzimanje nekih kreditnih obaveza, on je kupio 80% akcija i kontrolu nad Sinkler Riserčom. Finansijske institucije zadržavaju 10%, a samom Sinkleru takođe ostaje 10% njegove bivše kompanije, uz ulogu savetnika i počasne titule doživotnog predsednika.

Ova dobra vest za Sinklera izazvala je oduševljenje među vlasnicima Sinklerovih računara u Britaniji, kao i u drugim zemljama. Robert Maksvel se slavi kao spasitelj kućnog računarstva. No, Maksvel ne sedi na lovorikama — on smatra da Sinkler mora da poveća izvoz po svaku cenu! U

Greškom u svetu velikog biznisa: Klajv Sinkler

svom stilu, da je vreme novac, on je za nekoliko dana obišao nekoliko zemalja Istočne Evrope u trgovačkoj misiji. Poznat od ranije po odličnim poslovnim vezama sa socijalističkim zemljama, njega su primile i najviše ličnosti u nekim zemljama. Tako je, prema pisanju londonskih listova, on razgovarao i sa poljskim premijerom Jaruzelskim i, kako se ovde tvrdi, Poljska bi uskoro trebalo da uveze 60,000 „spektruma+", koji će se u početku prodavati u radnjama Pewex, gde se zapadni proizvodi mogu kupiti samo za devize. Navodno, i Bugarska i Sovjetski Savez pokazuju interes za Sinklerove računare i pregovori su u toku. Sinkler je nedavno pokazao i najnoviju verziju „spektruma+" sa ćiriličnom tastaturom. Izgleda da su Sovjeti dosta zainteresovani i za QI, pogotovo pošto je početkom jula britanska vlada skinula zabranu sa izvoza savršenijih računara u zemlje Istočne Evrope.

Quo vadis, Sinkler?

Izgledi da ovi finansijski problemi i nisu mnogo uzbudili ser Klajva. On se i dalje pojavljuje na sajmovima, seminarima i televiziji. I dalje priča o svojim idejama o revolucionarnijoj petoj generaciji računara, na kojima sada radi. Samo je pitanje da li će ga finansijske institucije i banke i dalje podržavati novčano. Jer, Sinkler je nekima već postao dosadan — ponaša se kao da igra ulogu rastresenog naučnika koji je greškom ušao u svet velikog biznisa i velikog novca. To njegovo tapkanje u svetu biznisa stajalo je banke u poslednjih desetak godina i dosta novaca. Bilo je tu poslovnih grešaka sa kalkulatorima i digitalnim časovnicima, malim prenosnim televizorima, električnim vozilom — da nabrojimo samo nekoliko od njih, mada su sve to u svoje vreme bili revolucionarni proizvodi, koji bi, možda, uz bolji marketing i bolje poslovno vođenje imali drugu budućnost.

Računarski Kolčicki

I pored tmurnih vesti, izgleda, ipak, da industrija mikroračunara polako izlazi iz krize. Najgore kao da je prošlo. Poznati komentator sa toga područja Rej Hemond (Ray Hammond) daje sledeću prognozu za šest meseci:

***Sinkler** — ove godine neće biti novog kućnog računara. Početkom sledeće biće lansirana prenosna verzija sa ugrađenim ravnim ekranom i mikrodrajvima. Spektrumu+ se neće sniziti cena, već će neki prodavci davati besplatni softver. Spektrum + će i u nekoliko sledećih godina biti najvažniji kućni računar.*

***Ejkorn** — ništa novo, osim malo sniženje cene za BBC računar.*

***Amstrad** — još jedan novi model ove godine, 16-bitni računar programski kompatibilan sa prethodnim modelima.*

***Komodor** — nade se polažu u model C128, koji je u velikoj meri kompatibilan sa popularnim modelom C64.*

***Atari** — previše se obećava i biće dobro ako se ispuni i pola. Veliko sniženje cena sadašnjim modelima.*

Sinkler je nedavno dao i intervju nedeljniku MicroScope gde tvrdi da on nikada nije ni želeo da bude biznismen (a zašto to nije rekao ranije?), već da je najsrećniji kada radi u labaratoriji. Sinkler tvrdi da je sretan što je došlo do sporazuma sa Maksvelom i da se nada da će Sinkler Riserč stati na zdrave noge, pošto će ga voditi pravi poslovni ljudi. On će biti plaćen za svoj posao kao savetnik u kompaniji, a zadržao je i pravo da samostalno radi na petoj generaciji računara, mada Maksvel želi da se to radi u okviru Sinkler Riserča i podružnice Metlab u Kembridžu. Ukoliko Sinkler ne želi da se pridruži radu na inteligentnim računarima, Maksvel namerava da okupi ekipu svojih stručnjaka. Sinkler, bar za sada, ne želi da otkrije svoje planove — samo kaže da se nada da neće doći do spora sa Bobom Maksvelom.

U ovoj zbrci oko spasavanja Sinklera, malo je nezapaženo prošla vest, koja je došla iz Sinklerove podružnice za naučna istraživanja Metlab, da je uspešno konstruisan visokointegrisani čip, koji može da preuzme ulogu svih čipova u nekom računaru. Probna proizvodnja bi trebalo da počne već sledeće godine.

Oliveti diže ruke?

Drugi najistaknutiji predstavnik britanske mikroračunarske industrije, firma Ejkorn (Acorn), i dalje ima puno problema. Kao što smo javili pre nekoliko meseci, italijanski gigant Oliveti je kupio pola deonica Ejkorna i time, u stvari, preuzeo efektivnu kontrolu. Ali i pored investicije od preko 10 miliona funti, koliko je plaćeno za te deonice, koja se sada koristi za otplatu dugova, finansijski problemi se i dalje gomilaju. Zastareli i preskupi model BBC se slabo prodaje, a najnovija verzija BBC+ nije doživela komercijalni uspeh, tako da veoma malo novaca ulazi u kasu Ejkorna. To je primoralo ovu firmu da potraži kupce za svoje podružnice, među kojima je i najvažnija softverska kompanija Ejkornsoft.

Najveći interesent za Ejkornsoft je British Telecom, kompanija koja je vlasnik telefonske mreže u Britaniji i koja ima vlastitu uspešnu softversku firmu Firebird. Za oko 1 milion funti British Telecom bi trebalo da postane vlasnik preko 1000 odličnih programa, kao i gotove robe čija je prodajna vrednost blizu 2 miliona funti. No, ta svota je tek kap u moru Ejkornovih finansijskih poteškoća. Izgleda da Oliveti ne želi da neograničeno otvori kesu i zato je došlo do ponovne suspenzije Ejkornovih deonica na londonskoj berzi, a banke traže nove mogućnosti za otplatu dugova. Prema vestima iz londonskog Sitija, postoji čak i mogućnost da Oliveti potpuno otpiše svoju investiciju od 10 miliona funti i digne, ruke od Ejkorna. Drugo rešenje za Olivetija je da i dalje sipa novac u vreću bez dna ili da proda Ejkorn nekom trećem investitoru. Ali, da li postoji još neki Bob Maksvel, koji je zainteresovan za pune magacine robe koja se sve slabije traži? Sudbina Ejkorna sada najviše zavisi od sposobnosti bankara da osiguraju dodatne kredite.

Anđelko Zgorelec

Naš test

„kanon" protiv „epsona"

KAGA TAXAN KP810/CANON PW1080

Prikazujući poznate Epsonove modele RX80 i FX80 u „Računarima 2", zaključili smo da se radi o najboljim štampačima u svojoj klasi, koji predstavljaju dominirajući svetski standard. U međuvremenu, ostale japanske firme nisu sedele skrštenih ruku — na tržištu se pojavilo nekoliko štampača koji su, i pored znatno niže cene, nudili potpunu kompatibilnost sa Epsonom FX80. Štampači „Kaga" (Kaga) KP810 i „Kanon" (Canon) PW1080A, uklapajući se u ovaj trend, nude i nešto novo: NLQ (near letter quality) mod koji se po kvalitetu otiska opasno približavaju štampačima sa lepezom. Obzirom da se „Kaga" i „Kanon" razlikuju jedino po ceni (drugi je dvadesetak funti skuplji), odlučili smo da im posvetimo zajednički prostor u „Računarima".

Kada prebrodite sve carinske probleme i donesete paket težak osam kilograma do kuće, očekuje vas uzbudljiv zadatak: računar je obično lako otpakovati i priključiti, disk nešto teže, a štampač predstavlja pravi problem: osiguran je gomilom šrafova, gumica, lepljivih traka i drugih sprava koje testiraju vaše strpljenje. „Kaga" je dobra potvrda ovog pravila: kada se uverite da vam se u paketu nalazi štampač, kabl, ručica za okretanje papira, traka i, naravno, uputstvo za upotrebu, moraćete najpre da odšrafljujete dno printera od dna kutije, zatim da polomite nokte odlepljujući selotejpe, oslobodite glavu snažne gumice koja joj ne dozvoljava da se pomeri, montirate utikač na kabl (ako vam štampač stigne u subotu popodne, a u kući nemate ni jedan šuko priključak, sačekaćete do ponedeljka!) i montirate traku i papir.

Poslednje dve operacije su najodgovornije: traka se nalazi u kaseti koja je drukčija od standarda koji je nametnuo „epson" (skuplja je dvadesetak penija) i ne treba je provući između limenog zaštitnika i papira već između glave i limenog zaštitnika; ako, poput autora ovoga teksta, uradite prvo, sve će lepo raditi ali će traka izlaziti iz ležišta kada god naiđe na perforaciju. Pre umetanja papira pažljivo proverite da li su oba preklopnika na mestima koja odgovaraju papiru koji koristite: „kaga" i „kanon" mogu da pišu kako na standardnoj perforiranoj hartiji tako i na rolnama papira ili pojedinačnim listovima. Posebno je teško umetati obične listove: glavni preklopnik će, najpre, morati da bude u položaju „Paper Set", zatim u položaju „Pin Feed" a onda će zauzeti svoj konačni položaj „Friction". Za eksperimentisanje sa printerom ćete, jasno, odabrati perforirani papir ili, ako možete da je nabavite, rolnu.

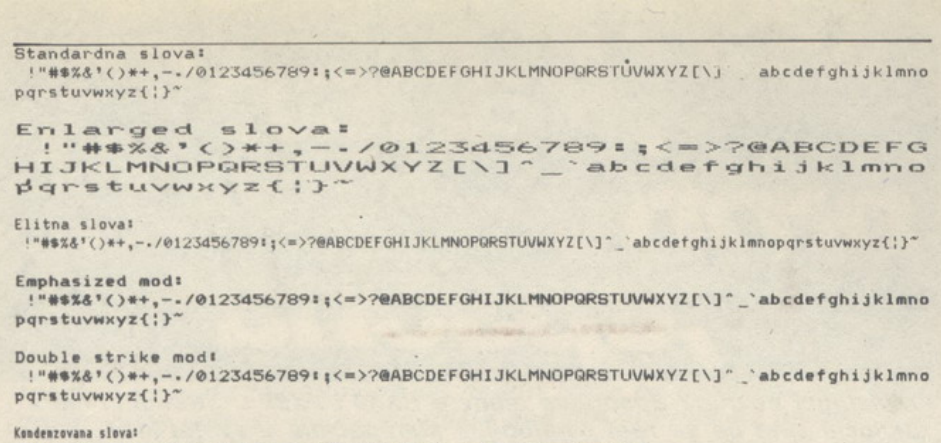
Pošto ste sve pripremili, printer će testirati samoga sebe: pritisnite senzor LF (da, dobro ste pročitali: umesto „epsonova" tri prekidača, „kaga" i „kanon" imaju isto toliko senzora koji, nekim slučajem, imaju potpuno iste funkcije) i, držeći ga pritisnut, uključite štampač. Ukoliko je sve u redu, trebao bi da se začuje kratak bip, a zatim započne ispisivanje teksta; za nas je to bila prilika da prikažemo standardni set karaktera bez potrebe da pišemo bejzik program!

Diskretna veza

„Kaga" i „kanon" su, u osnovnoj verziji, opremljeni Centronics interfejsom, što znači da ih možete povezati sa amstradom, BBC-jem, „eplom", IBM-om i mnogim drugim računarima. Što se „komodora" i „spektruma" tiče, možete da birate između kupovine RS 232 interfejsa, koji se ugrađuje u štampač, ili Centronics interfejsa koji se povezuje sa računarom (ukoliko imate „spektrum" pa se odlučite za prvo rešenje, moraćete da kupite i „interfejs i"). Da spasemo redakcijski telefon, nismo nigde videli oglas u kome bi se bilo koji od ovih štampača nudio sa interfejsom RS 232, a bez centronicsa. Razlog za to je verovatno činjenica da Centronics opcija i ne zahteva neke posebne komponente; to je prirodni način na koji štampač prima podatke. Ako želite serijsku komunikaciju, doplatite 50 funti za interfejs i dobićete sa njim printer bafer od 3 kilobajta.

Kada, najzad, povežete računar sa štampačem (pazite, kabl nije uračunat u cenu!), možete da testirate sve tipove slova koje „kaga" („kanon") nude: za to vam je potreban program poput onoga sa slike 2, ispod koga su prikazani rezultati njegovog izvršavanja. Pažljiv posmatrač će primetiti da su korišćeni modovi (Standard, Enlarged, Condensed, Condensed enlarged, Elite, i Elite enlarged) po imenu i po kontrolnim kodovima koji ih aktiviraju identični sa „epsonovim". Vrlo pažljiv posmatrač će, međutim, primetiti da slova nisu potpuno identična „epsonovim", što se najbolje vidi kod standardnog seta: karakteri su za nijansu niži i oblici nekih su promenjeni (pogledajte, na primer, nulu!). Da li razloge za promene treba tražiti u zakonu o zaštiti autorskih prava ili u hiru „kanonovih" dizajnera teško je reći; u svakom slučaju, stvar je ukusa koja će vam se slova više dopasti, pri čemu je nesumljivo da je njihov kvalitet jednak: printer ih crta na matrici 11*9 tačaka (kao i kod „epsona", dve susedne tačkice po horizontali ne smeju da budu ispisivane).

→



Brzi Gonzales

Ono što ne možete da primetite, ma koliko pažljivo posmatrali sliku, je fascinantna brzina rada štampača: u standardnom modu specificirana brzina pisanja iznosi 180 karaktera u sekundi, ali su naša merenja pokazala da, pri potpuno popunjenim redovima, ona dostiže preko 200 znakova. Razlika u odnosu na 100 znakova kod „epsona RX80" i 150 znakova kod FX80 je, dakle, lako primetna. I pored znatnog ubrzanja, glava štampača se praktično ne greje (probajte da pipnete glavu „epsona" pošto ispiše desetak strana) što bi trebalo da joj produži vek koji, prema specifikacijama, iznosi 100 miliona znakova (oko 67000 strana). Da nastavimo sa poboljšanjima: „kaga" je u normalnom modu primetno tiša od „epsona", a oba štampača, osim toga, poseduju mod u kome rade znatno sporije i znatno tiše.

Glavna prednost „kage" i „kanona" nad „epsonom" se ipak zove malo drugačije: NLQ mod. U engleskoj literaturi se, naime, često pominje termin „Letter Quality" koji označava kvalitet otiska štampača sa lepezom i električnih pisaćih mašina, kvalitet koji je, po njihovim standardima, jedini prihvatljiv za ozbiljna poslovna pisma. NLQ je skraćenica od „Near Letter Quality", što bi trebalo da označi otisak sličan pomenutom. U kvalitet tog otiska ćete se lako uveriti kada bacite pogled na sliku 3: ma koliko se trudili, nećete primetiti tačkastu strukturu slova. Da stvar bude lepša, konstruktori ovih štampača su se pobrinuli da u ovom modu predizajniraju karaktere tako da liče na slova IBM-ove pisaće mašine. Da bi se taj kvalitet postigao, bila je potrebna matrica 23*16 koja je dobijena uz pomoć trikova: glava dva puta prelazi preko reda, pri čemu je u drugom prolazu pomerena za pola tačke (ili možda za celu tačku?) naniže. Obzirom da se u takvim prilikama ne može štampati u oba pravca i da preko svakog reda treba prelaziti dva puta, brzina rada je smanjena na 30 znakova; to je samo petnaestak procenata od uobičajenih 200 znakova, ali je i dalje dvostruko brže od štampača sa lepezom!

Pošto je za definiciju jednog NLQ slova potrebno 23*16=368 bita ili 46 bajtova, konstruktori „kage" i „kanona" su morali u njih da ugrade veliki ROM — umesto „epsonovih" 8 K, u „kanon" su ugrađena tri eproma 2764, što predstavlja ROM od 24 K. Na štampi je ostavljeno i jedno prazno podnožje u koje možete da ugradite bilo dodatni osmokilobajtni eprom sa definicijama karaktera (u okviru uputstva za upotrebu nije, na žalost, opisan tačan format tih eproma, verovatno zato što proizvođači nisu preveliki ljubitelji konkurencije), bilo RAM 6264 (8 K) koji, zavisno od položaja jednog od 20 prekidača, može da služi kao bafer ili kao prostor za definicije karaktera.

Ćevapčići u Čačku

Kao što ste i očekivali, štampači omogućavaju definisanje znakova, što je operacija neobično interesantna za one koji se ne usuđuju da prčkaju po ROM-u sledeći naše savete iz „Računara 3". Crtanje slova je spor posao koji zahteva pomalo umetničkog dara, a pretvaranje crteža u brojke koje bi štampač razumeo mučenje čak i za najstrpljivijeg hakera. Ukoliko vam se čini da je tako jednostavnije, možete da sastavite nekakav editor za definicije karaktera na bejziku ili da se poslužite našim programom ispisanom na 6502 asembleru; koji posebno objavljujemo u „Biblioteci programa" (str. 37); interesovaće vas, jasno, jedino DATA liste. Predefinisali smo srednje i velike zagrade, oznaku za funtu, naopaku kosu crtu (backslash), vertikalnu liniju i tildu kao manje-više nepotrebne znakove koji se obično nalaze na tastaturama boljih kućnih računara; u principu bi bilo jednostavno predefinisati neka od nemačkih, švedskih, francuskih, danskih ili sličnih slova, ali bi tada trebalo pisati printer drajver za tekst procesor koji vrši transformaciju koda.

Ukoliko ste do sada koristili FX80, bilo kakav program za definisanje znakova koji posedujete će raditi i na novom štampaču, zahvaljujući potpunoj kompatibilnosti kodova. Moraćete jedino da se namučite ako želite da definišete NLQ karaktere: za svako slovo treba pretvoriti 2*23=46 binarnih brojeva u dekadne, pa otkucati rezultate. Želeli smo da vam pomognemo i u ovome, ali bezuspešno; u uputstvu za upotrebu očigledno postoji neka greška zahvaljujući kojoj štampač ignoriše pokušaje da se definišu NLQ slova; jedan jedini primer koji je dat nije od naročite pomoći pošto ne radi ono što bi trebao da radi (u uputstvu za upotrebu ćete, uzgred budi rečeno, pronaći i mnogo drugih grešaka, ali ni jedna nije kritična kao ova, jer možete da se poslužite uputstvom za „epson" FX80 koje je takođe „prošarano" greškama ali se one, srećom, nalaze na različitim mestima).

Kontrolni kodovi

Osim raznih vrsta slova, „kaga" i „kanon" poseduju i gomilu kontrolnih kodova koji pružaju usluge odlično opisane engleskim terminom „miscellaneous". Tu spadaju horizontalna i vertikalna tabulacija (koliko god smo razmišljali, nismo mogli da smislimo za šta bi se ove tabulacije koristile ukoliko je računar opremljen bilo kakvim tekst procesorom), poništavanje zadnjeg poslatog slova, pražnjenje čitavog bafera, izbor veličine strane i aktiviranje automatskog preskakanja perforacije, postavljanje leve i desne margine, zabrana dvosmernog štampanja (za one koji vole da čekaju),

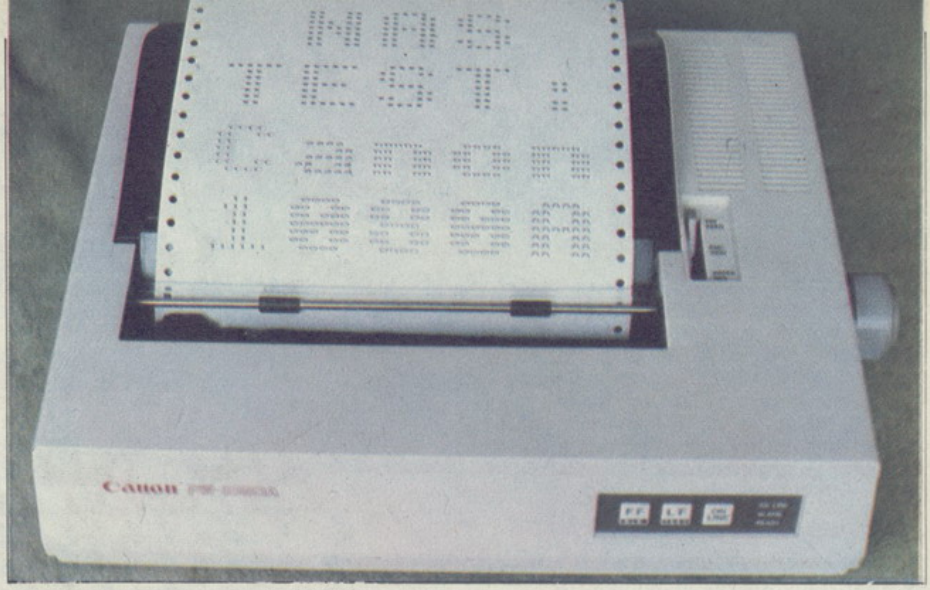
Doslo je vreme da se posvetimo glavnoj prednosti Kage i Canona nad Epsonom: NLQ modu. U engleskoj literaturi se, naime, cesto pominje termin Letter Quality koji oznacava kvalitet otiska Daisy Wheel stampaca i elektricnih pisacih masina, kvalitet koji je, po njihovim standardima, jedini prihvatljiv za ozbiljna poslovna pisme. NLQ je skracenica od Near Letter Quality sto bi trebalo da oznaci otisak slican pomenutom. U kvalitet tog otiska cete se lako uverite kada bacite pogled na sliku 3: ma koliko se trudili, necete primetiti tackastu strukturu slova. Da stvar bude lepsa, konstruktori ovih stampaca su se pobrinuli da u ovom modu predizajniraju karaktere tako da lice na slova IBM-ove pisace masine. Da bi se taj kvalitet postigao bila je potrebna matrica 23*16 koja je

isključivanje i uključivanje detektora nestanka papira i, naravno, CHR$(7) za biper. Neke od ovih karakteristika možete da birate tako što ćete promeniti položaj jednog od mikroprekidača na štampanom kolu štampača, mada pre toga treba rasklopiti čitav uređaj, što se pokazuje kao umereno komplikovan zahvat.

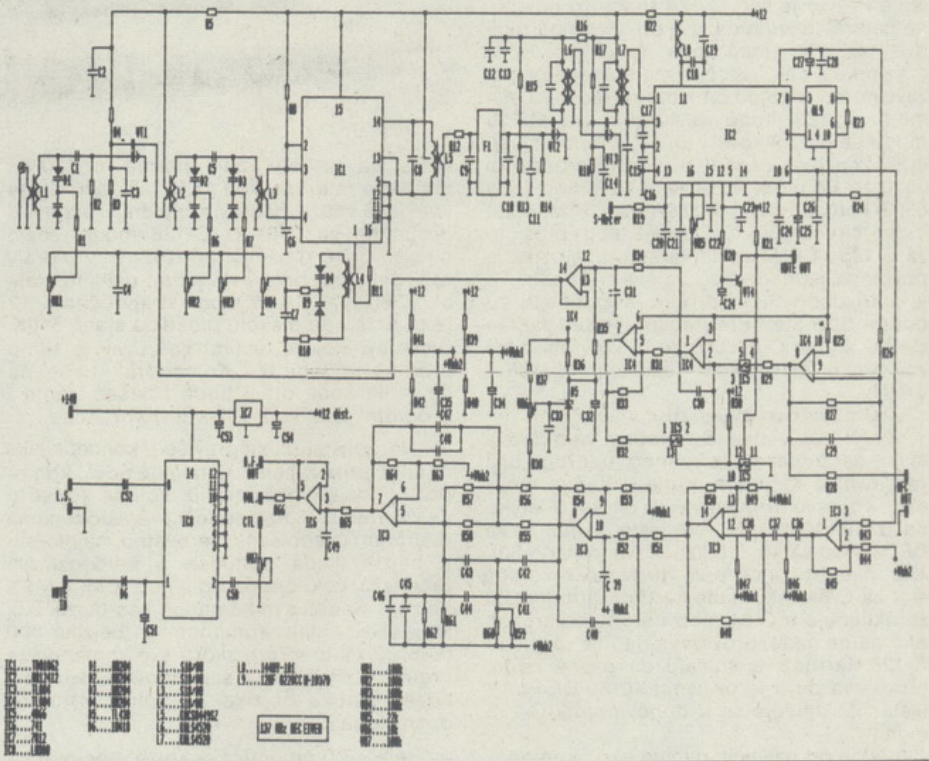
Grafičke mogućnosti „kage" i „kanona" su jednake „epsonovim": u normalnom modu može da se upravlja sa 60 tačaka po inču (=2,54 cm), u dvostrukoj gustina sa 120, a u četvorostrukoj gustini čak sa 240! Razmak između linija može da se bira i iznosi n/72 ili n/216 inča (n sami birate). Svi programi za crtanje koji su radili na „epsonu" lepo će poslužiti i na „kagi" pa je tako nastao i naš crtež: radi se o demonstraciji rada programa „Diagram" koji je pisan za BBC B; onoga ko pogodi šta radi iscrtano kolo pozivamo da nam se javi.

Pre nego što završimo ovaj kratki prikaz, treba da pomenemo i „sitnice" koje, verovatno, nestrpljivo očekujete od samog početka: cene „Kaga KP810" se u Engleskoj može nabaviti za 290 funti a „Kanon PW1080A" za 320, što je i dalje manje od 360 funti koje treba izdvojiti za „epson FX80". Isplati li se onda kome da kupuje epsona? Možda, jer se kod nas ovaj štampač mnogo koristi u raznim firmama, pa nije teško (besplatno) doći do trake. Van ovih razloga, „Kaga KP810" i „Kanon PW1080A" predstavljaju izvanredne printere, koje vam bez rezerve preporučujemo.

Dejan Ristanović



Novi standard u ekonomskoj klasi: Kaga KP 810



Časopis GALAKSIJA i ZAVOD ZA UDŽBENIKE I NASTAVNA SREDSTVA — Beograd raspisali su prošle godine STALNI OTVOREN KONKURS za izradu softverske i hardverske podrške za računare GALAKSIJA, SPEKTRUM i KOMODORE. Tokom protekle godine Zavod je od autora otkupio 13 kaseta sa više od 50 programa. Realizovan je i novi model računara „GALAKSIJA+", sa visokom rezolucijom, trokanalnim zvukom i ekranskim editorom. Uz mnoge knjige koje su prihvaćene i štampane, prihvaćena je i ideja za proizvodnju robota u kitu. Zbog nesumnjivog uspeha koji je KONKURS imao i velikog interesa kod mladih stvaralaca iz cele zemlje, obaveštavamo Vas da je i dalje:

OTVOREN
STALNI KONKURS
za izradu softverske podrške za računare

1. **GALAKSIJA** (4×6 i 8×6)
2. **GALAKSIJA+**
3. **SPEKTRUM** i
4. **KOMODORE 64**

Konkurs se odnosi na izradu:
1. Sistemskih i uslužnih programa
2. Obrazovnih programa
3. Dalje usavršavanje hardvera i softvera za GALAKSIJU
4. Didaktičke igre
5. Elektronika u kitu (računari i roboti) i
6. Priručnici i knjige o računarima

Sve radove treba dostaviti na adresu Zavoda za udžbenike i nastavna sredstva, Beograd, Obilićev Venac 5/I, telefon (011) 636-971 i (011) 638-405. Posle ocene Zavod sa autorima potpisuje ugovor o saradnji.

o „komodoru' sve najlepše

Komodor 128

Pre nego što je „komodor" 128 zvanično i prikazan, pričalo se da je C-128 samo prošireni C-64,. Prvi utisci posle prikazivanja bili su da je to CP/M računar sa C-64 modom. Istina je da je „komodor 128" zaista kutija sa tri različita računara u sebi, pri čemu je, izgleda, ipak najinteresantniji onaj — treći.

Tri u jednom

Spolja, sva tri računara dele istu, odlično dizajniranu, plastičnu kutiju sa 92 tastera. Iznutra, dele jedino RAM čipove i napajanje i — to je sve. C-128 sadrži tri odvojene centralne jedinice, dva odvojena bejzika, dva različita video čipa, odvojene ROM memorije i, čak, različite memorijske mape, zavisno od toga u kome modu radi — C-64 modu, C-128 modu sa 40 znakova, C-128 modu sa 80 znakova i u CP/M modu sa 40 ili 80 znakova u redu. Tu su i tri procesora 6510 sa 64 mod, 8052 za C-128 mod (oba 6502 kompatibilna) i Z80A za CP/M mod. Video čip sadrži VIC II za 64 mod, plus čip za C-128 i CP/M sa 80 karaktera. Muzika je poverena jednom čipu za sva tri moda. To je stari dobri SID čip, čije mogućnosti su dobro poznate. Programski jezici sadrže BASIC 2.0 za C-64, 7.0 za C-128 i monitor za rad u mašinskom jeziku za C-128 i CP/M.

Da li sve ovo zvuči zbunjujuce?

Zvuči, ali — nije. Svi modovi vam dopuštaju da prelazite iz jednog u drugi bez resetovanja računara. Kada uključite računar, on prvo proverava da li se u disku nalazi CP/M disketa. Ako jeste, uključuje se CP/M mod sa 40 ili 80 znakova (obično 80). Ako nije, računar prvo proverava kartidž slot za C-64. Ako nađe kartridž, automatski se uključuje u C-64 mod i startuje kartridž. Ako ga ne nađe, proverava da li je uključen C-128 kartridž. U slučaju da ga ne nađe, proverava da li je pritisnut 40/80 DISPLAY taster i uključuje se u dogovarajući C-128 mod.

Jedna od najvećih dilema oko „komodora 128" bilo je pitanje kompatibilnosti sa C-64. Zbog gorkih iskustava sa nekompatibilnom plus četvrtkom, „Komodorovi" ljudi su morali da se pobrinu da C-128 bude 100% kompatibilan sa C-64. Testovi pokazuju da C-128 radi sa svim postojećim bejzik i mašinskim programima za C-64. Samo jedan program nije prošao test. To je Turbo Disk — program koji ubrzava drajv 1541 tri puta. Program je lepo radio sa C-128, povezanim sa starim drajvom 1541, ali nije radio sa novim drajvom 1571. To i nije neko iznenađenje, jer TurboDisk radi i sa memorijom disk-drajva i sa memorijom računara. Treba uzeti u obzir da i novi drajv 1571 radi u tri moda: kao 1571, kao stari drajv 1541, a može da čita i original Osborne i Kaypro CP/M diskete. Kada radi kao 1541, na disketu staje normalno 170K i brzina je ista kao kod 1541, ali kažu da se vrlo lako može napraviti da drajv učitava i programe za C-64 iz C-128 moda, znači mnogo brže. U osnovnom 1571 modu drajv radi pet puta brže i na jednu disketu staje oko 360K. U CP/M modu drajv učitava 12 puta brže i na disketu može da stane 510K. Diskdrajv koji je testiran još uvek je samo prototip i ljudi iz „Komodora" tvrde da će raditi kada drajv bude završen, raditi i programi kao TurboDisk ili HypraLoad.

Ne upuštajući se previše u koncepcijske detalje, prikazaćemo detaljnije novi „komodorov" bejzik nazvan 7.0, koji se koristi u C-128 modu. To je, sudeći prema ocenama u stranim časopisima, verovatno, najmoćniji bejzik ikada ponuđen u personalnom računaru, čak mnogo bolji i od IBM ili MSX bejzika. Sadrži sve komande kao bejzik 2.0, sve disk i fajl komande iz bejzika 4.0 (C-8032 i superPet), skoro sve komande za grafiku i muziku iz sajmonovog bejzika i super grafike bejzika 3.5 (plus /4) i još dosta toga.

Bejzik 7.0 omogućava korišćenje grafike visoke rezolucije, sprajtova i muzike bez starih PEEK i POKE komandi ili mašinskog jezika. Pokretanje sprajtova se odvija preko interapta, tako da nekoliko bejzik komandi mogu istovremeno da pokreće osam sprajtova za vreme dok program radi nešto drugo, pa čak i kada je program prekinut. Za definisanje sprajtova možete nacrtati nešto u visokoj rezoluciji pa to prebaciti u sprajt, ili koristiti ugrađeni sprajt editor. Korišćenje SID muzičkog čipa je takođe veoma pojednostavljeno, tako da možete svirati ili praviti zvučne efekte sa svega nekoliko bejzik komandi.

Zbog vrlo obimnog i razrađenog bejzika , opisaćemo samo važnije i interesantnije naredbe iz onih oblasti sa kojima je „komodor" do sada uvek oskudevao.

Računar za sve: Zahvaljujući kompatibilnosti sa „šezdesetčetvorkom", na „komodoru 128" se mogu igrati izuzetno atraktivne igre, dok CP/M mod otvara vrata u „akademsku" biblioteku poslovnog softvera sa preko 10000 ozbiljnih naslova

Disk

Disk i fajl komande sadrže: DLOAD i DSAVE (za snimanje na disk i sa diska bez dodavanja ,8 na ime programa); DVERIFY (za verifikaciju programa na disku); CATALOG i DIRECTORY (za prikazivanje spiska fajlova na disketi bez brisanja bejzik programa); COPY (kopiranje programa korišćenjem duplog diska); BACKUP (kopiranje cele diskete korišćenjem duplog diska); COLLECT (reorganizacija blokova na disketi); CONCAR (za kombinovanje dva programa); HEADER (za formatiranje diskete); RENAME (promena imena programa); SCRATCH (brisanje programa); DOPEN i DCLOSE (otvaranje i zatvaranje kanala); DCLEAR (zatvaranje svih disk kanala); DS i DS$ (za čitanje kanala grešaka); BLOAD i BSAVE (za snimanje mašinskog programa ili dela memorije); i BOOT (za automatsko učitavanje i startovanje programa).

Neke komande su vrlo slične kao za C-64 i 1541 drajv: na primer, COLLECT je isto što i OPEN 1, 8, 15, „VO":CLOSE1. Komande se takođe mogu koristiti kao D-SHIFT-L za DLOAD ili čak SHIFT-RUN-STOP za automatsko učitavanje i startovanje programa sa diskete. Takođe, neki funkcionalni tasteri su programirani tako da izvršavaju određene komande kao, na primer, DIRECTORY.

Sprajtovi

Komande za rad sa sprajtovima ne zamenjuju samo stare POKE komande već pružaju i mnogo veće mogucnosti. U demo programu koji ide uz računar, bejzik 7.0 pokreće sprajtove dovoljno brzo da možete praviti odlične akcione igre i bez mašinskog jezika.

sprdef

U direktnom režimu komanda aktivira ugrađeni sprajt editor. Komande uz sprajt editor omogućavaju vam da sprajtove brišete, ispra-

Jedan od najmodernijih trendova u današnjoj softverskoj industriji je „integrisani softver" — softverski paket koji sadrži dva, tri ili više različitih programa u jednom, kao na primer, LOTUS 1-2-3. Sada nam „Komodor" prvi put predstavlja integrisani hardver — tri računara u jednoj kutiji i to sve za manje od 300 dolara! Ova kutija sadrži standardni „komodor" sa 64 K memorije, novi C-128 mod sa novim bejzikom 7.0 i CP/M mod baziran na procesoru Z80 sa svojih 10000 profesionalnih programa. Korišćenjem RAM diskova, memorija se može proširiti do 512 K. Kada se tome doda grafika visoke rezolucije 640•200, osam sprajtova, trokanalni generator tona i, neki kažu, najmoćniji bejzik ikada ponuđen u nekom kućnom računaru, postaje jasno da oni koji vole „komodor" ovoga puta dobijaju premiju.

Rečnik bejzika 7.0

AUTO automatska numeracija linija
APPEND otvara sekvencijalnu datoteku
BACKUP kopira celu disketu
BANK poziva blokove memorije za PEEK, POKE i SYS
BEGIN . . .BEND postavlja više bejzik linija u jedan blok
BOOT učitava i startuje CP/M sa diskete
BOX crta pravougaonike
BSAVE snima deo memorije
BUMP određuje tip dodira sprajtova
CATALOG lista sadržaj diskete
CHAR prebacuje tekst u visoku rezoluciju
CIRCLE crta krugove, elipse i sve poligone
COLLECT briše otvorene datoteke i reorganizuje disk
COLLISION određuje dodire sprajtova
COLOR boja teksta i grafike
CONCAT povezuje dve sekvencijalne datoteke u jednu
COPY kopira fajl sa diska
DCLEAR zatvara sve kanale prema disku
DCLOSE zatvara pojedinačan kanal prema disku
DEC pretvara heksadecimalne brojeve u decimalne
DELETE briše određeni deo programa
DIRECTORY izdaje spisak programa sa diska
DLOAD učitava sa diska
DOPEN otvara kanal prema disku
DO . . . LOOP struktuirana „petlja"
DRAW crta tačke i linije
DSAVE snima na disk
DS čita kanal za greške sa diska
DS$ daje tekst greške sa diska
DVERIFY proverava program sa diska
EL prijavljuje programsku liniju u kojoj je greška
ELSE deo IF . . .THEN naredbe
ENVELOPE definiše krivu za sintesajzer
ER daje kod greške
ER$ daje tekst greške
EXIT izlazak iz DO . . .LOOP petlje
FAST prelazak na rad sa taktom od 2MHz
FETCH uzima podatke iz memorije RAM-FLOPPY
FILTER postavlja parametre za SID cip
GETKEY čeka na pritisak nekog tastera
GO 64 uključuje C-64 mod
GRAPHIC bira grafički mod
GSHAPE pretvara string u visoku rezoluciju
HEADER formatira disketu
HELP lista greške
HEX$ pretvara decimalne brojeve u heksadecimalne
INSTR određuje poziciju stringa u drugom stringu
JOY ispituje džojstik portove
KEY definiše funkcionalne tastere
LOCATE pozicionira grafički kursor
MONITOR poziva ugrađeni mašinski monitor
MOVESPR pomera sprajt
PAINT ispunjava prostor u visokoj rezoluciji
PEN ispituje svetlosno pero
PLAY svira tonove iz nekog stringa
POINTER daje adresu varijable u memoriji
POT ispituje pedl kontrolere
PRINT USING omogućava definisanje karaktera
PUDEF definiše karaktere za PRINT USING
RCLR ispituje boju za tekst i grafiku
RECORD formira relativne datoteke
RENAME menja ime fajlu na disku
RENUMBER vrši prenumeraciju programskih linija
RESTORE postavlja brojač za DATA na određenu liniju
RGR daje broj grafičkog moda
RREG daje vrednosti A, X, Y i status flega
RSPRCOLOR daje kod multikolor moda za sprajtove
RSPPOS daje poziciju i brzinu sprajta
RSPRITE daje sve sprajt parametre
RWINDOW daje sve prozor parametre
SCNCLR briše grafički ili ekran za tekst
SCRATCH briše fajl sa diska
SSHAPE pretvara deo visoke rezolucije u string
SLEEP zadržava izvršavanje programa za zadato vreme
SLOW prelazi na 1MHz takt
SOUND određuje frekfrenciju i trajanje tona
SPRCOLOR boja sprajta
SPRDEF poziva ugrađeni sprajt editor
SPRITE postavlja parametre za sprajtove
SPRSAV pretvara sprajt u string i obrnuto
STASH prevodi DATA podatke u memoriju
SWAP vrši razmenu podataka između dva bloka od 64 K
TEMPO određuje brzinu za PLAY
TROF isključuje TRACE
TRON uključuje TRACE
UNTIL postavlja uslove za Do . . .(UNTIL) LOOP petlje
VOL jačina zvuka za SOUND
WHILE (DO (WHILE) LOOP
WIDTH određuje debljinu linija za crtanje
WINDOW određuje veličinu prozora
XOR ekskluzivno ILI

U spisku se ne nalaze komande bejzika 2.0 za C-64 i VIC-20, već samo nove komande bejzika 7.0.

vljajte, ili im menjate boju i to u običnom ili multi-kolor modu. Kada završite definisanje sprajtova, vraćate se u bejzik. U Bejziku 7.0 ne morate da brinete gde ćete smestiti sprajtove u memoriji. Raunar za njih odvaja memoriju počev od lokacije 3584 (E00 hex). U ostalom, sa 128K RAM memorije sigurno nećete imati problema sa nekoliko sprajtova.

SPRITE ,on/off, foreground, priority, x, y, mode

Postavlja određene parametre kao što su uključivanje i isključivanje sprajtova, prioritet, postavlja sprajt na određeno mesto i određuje singl ili multi-kolor mod.

SPRSAV sprite , string
SPRISAV string, sprite

Pretvara sprajt u string i obrnuto. Na primer, možete prvo nacrtati nešto u visokoj rezoluciji, zatim to pretvoriti u string komandom SSHAPE, i na krju string pretvoriti u sprajt. Možete, takođe, šetati sprajt po ekranu i kada poželite da vam stalno stoji na jednom mestu, jednostavno ga pretvoriti u grafiku komandama SPRSAV i GSHAPE. Tako možete postići da vam cela slika visoke rezolucije dođe iz raznih pravaca i sastavi se na ekranu.

MOVSPR sprite#X, Y

Ova naredba se zove apsolutno pomeranje sprajtova i ona pomera određeni sprajt na koordinate određene sa X, Y. Slična komanda kao standardno POKE kod C-64.

MOVSPR sprite#,+/−X,+/−Y

Ova naredba se zove relativno pomeranje sprajtova i koristiti se kada ne znate gde vam se sprajt nalazi na ekranu. Određeni sprajt se pomera za X i Y u odnosu na stare koordinate. Na primer, ako želite da pomerite sprajt pet za sedam mesta ulevo i deset mesta nadole, kucaćete MOVSPR 5,−7,+10.

MOVSPR sprite=, angle# speed

Pomera sprajt u određenom pravcu određenom brzinom. Ovo je urađeno preko mašinskog interapta, tako da se sprajtvo kreću po ekranu za vreme dok se bejzik program i dalje izvršava. Kada sprajt izađe sa ekrana, pojavljuje se automatski na drugoj strani ekrana. Na primer, možete sa par bejzik komandi napraviti da vam se sprajtovi kreću po ekranu u nekoj igri tipa osvajača, dok vam program ispituje džojstik i pomera vaš brod po ekranu. Parametar angle određuje pravac kretanja sprajtova: 0=gore, 90=desno, 180=dole, speed brzinu pomeranja.

COLLISION type, line#

Ova komanda registruje dodir sprajta sa drugim sprajtom, dodir sa pozadinom ili sa lajtpenom (ovo zavisi od parametra type). Parametar

→

line# označava broj programske linije podprograma na koji program treba da pređe ako dođe do nekog dodira. Potprogram mora da se završava sa RETURN kao i svaki potprogram.

BUMP (type)

Ispituje dodire sprajtova i vraća ih kao vrednost. Pomoću njega možemo detektovati i dodire koji su se dogodili izvan dela vidljivog na ekranu. BUMP(0) registruje dodire između sprajtova, a BUMP (1) između sprajtova i pozadine

Grafika

GRAPHIC mode#, clear, window

Prebacuje ekran u mod određen parametrom MODE#. Mod 0 je mod za tekst sa 40 znakova u redu. 1 je visoka rezolucija; 2 je visoka rezolucija sa prozorima; 3 je multi-kolor mod; 4 je višebojni mod sa prozorima i 5 je mod za tekst 80 znakova u redu (samo RGB). Prozori su isti kao kod „epl" i „atori" računara i predstavljaju nekoliko linija teksta na ekranu visoke rezolucije. Počinju od linije 19, ali se mogu pomerati pomoću parametra window. Parametar clear određuje da li će ekran biti obrisan: 0 = da, 1 = ne.

COLOR source#, color#

Postavlja boje na ekranu. Color# parametar označava boju od 0 do 16, a parametar source određuje ekran: 0 za 40 znakova pozadinu; 1 za pozadinu u visokoj rezoluciji; 3 za boju okvira (bordera); 5 za boju karaktera i 6 za boju pozadine kod ekrana sa 80 znakova.

BOX source#, Xcentre, Ycentre, Xradius, Yradius, arc angle1, arc angle 2, angle, increment

Crta na ekranu visoke rezolucije krugove, elipse, petouglove, osmouglove i sve ostale poligone. Source# je boja tačaka, X i Y centre su koordinate centra, X i Y radius su X i Y koordinata radiusa, arc angle 1 početni ugao u stepenima (najbolje 0), arc angle2 je krajnji ugao u stepenima (360), angle je ugao rotacije (0), a increment određuje ugao između segmenata (2). Na primer: CIRCLE 1, 160, 100, 65, 50 crta zeleni krug; CIRCLE 1, 160, 100, 65, 10 zelenu elipsu; CIRCLE , 60, 40, 20, 18,,,, 45 osmougao, a CIRCLE , 260, 40, 20,,,,, 90 crta dijamant.

DRAWN source#, X1, Y1 to X2, Y2 to . . .

Crta tačke ili linije na ekranu. Source označava boju tačaka, X1 i Y1 su početne koordinate linija, X2 i Y2 su krajnje tačke. Na primer, DRAW 1, 150, 100 TO 250, 100 crta horizontalnu liniju, a DRAW 1, 10, 10 TO 150, 100 TO 250, 200 TO 10, 10 crta trougao. Ako vam tačke izađu izvan ekrana, odnosno ako su vam koordinate recimo negativne, ne brinite, C-128 će povući liniju do neke zamišljene tačke izvan ekrana i neće vam prijaviti nikakvu grešku.

LOCATE X, Y

Postavlja nevidljivi grafički kursor na koordinate X, Y kao početni položaj za dalje crtanje (DRAW TO A, B).

PAINT source#, Xstart, Ystart, mode

Popunjava deo ekrana bojom određenom parametrom source#. X i Y označavaju početne tačke, a mode određuje kako da se izvrši popunjavanje (0 = popuni deo označen sa source#, 1 = popuni deo označen tačkuma na ekranu).

SSHAPE string, corner 1, corner 2

Pretvara određeni pravougaonik nacrtan na ekranu visoke rezolucije u string. Površina ovog dela može biti do 255 bajtova, koliki je jedan bejzik string. SSHAPE je vrlo slična komanda kao GET kod IBM bejzika.

GSHAPE string, corenr 1, corner 2, mode

Pretvara podatke iz određenog stringa u tačke na ekranu. Corner 1 i 2 određuju koordinate na ekranu vioske rezolucije, a mode označava način kako će tačke biti postavljene na ekran: 0 pravougaonik će biti smešten onako kakav i jeste, 1 invertuje pravougaonik, 2 radi logičko OR (ILI) između tačaka na ekranu i tačaka iz stringa, 3 radi logičko AND (I), 4 EDR (ekskluzivno ILI). Komanda slična kao PUT kod IBM bejzika.

Zvuk

SOUND voice, freq, dur, sweep, min, step, wave, width

Postavlja zvučne parametre. Voice označava glas (1—3), freq frekfenciju 0—65535, a dur 0—32767 jiffys (jiffy = 1/60 sec.). Ostali parametri podešavaju zvuk do poslednjih sitnica.

PLAY O oct, T tune, U vol, V voice, X filter, notes

Svira jednu ili više nota koristeći odabranu oktavu, anvelopu, glasnoću, glas i filter. Oct označava oktavu 1—8; tune označava anvelopu 0—9; vol jačinu 1—9; voice glas 1—3; filter (0 = uključen 1 = isključen). Posebno je interesantan tyne parametar koji označava instrumente: 0 piano; 1 akorde; 2 kaliope; 3 bubanj; 4 flauta; 5 gitara; 6 hapsikord; 7 organa; 8 truba; 9 ksilofon.

ENVELOPE#, attack, decay, sus, rel, wave, width

Ovom komandom može se predefinisati neki instrument određen paramterom tune.

FILTER freq, lopass, bandapass, hipass, res

Ova komanda uključuje i isključuje filtere zadate filter parametrom kod komandie PLAY. Postoje četiri vrste filtera koje možete koristiti za pisanje muzike ili pravljenje različitih zvučnih efekata.

Pored komandi za sprajtove, grafiku i muziku, „komodore 128" sadrži još dosta drugih komandi, kao što su komande za prozore, za uključivanje i isključivanje blokova od 64K, brisanja delova bejzik programa, postavljanje funkcionalnih tastera, pronalaženje grešaka u bejzik programima . . . Takođe, postoje i komande za uključivanje mašinskog monitora, zatim za rad sa petljama bez komandi FOR-NEXT i komande za struktuirano programiranje.

Bejzik 7.0 predstavlja, izuzetan domet u razvoju programskog jezika za programiranje na visokom nivou. Potpuno isključuje potrebu za najnižim komandama tipa POKE i PEEK. Pored ovog obimnog bejzika, „Komodore 128" ima ugrađeni i odličan mašinski monitor za sve one kojima ni ovoliki bejzik nije dovoljan i koji žele da iz svojih mašina izvuku — još više.

Vladimir Krstonošić

Za one koji znaju malo više

Koristeći posebne memorijske kartridže, „komodor 128" se može proširiti do 512K RAM-a. Ova memorija se ne može koristiti iz programa, već se koristi kao RAM — DISK. Prenos podataka je brži nego čak i kod hard diskova. Memorija se, na žalost, briše gašenjem računara iz struje, pa bi taj deo memorije morao da se snima na disk drajv pre isključivanja računara iz struje.

Posebna memorijska jedinica MMU (Memory Management Unit), koja se nalazi od lokacije FF00, kontroliše komplikovanu memorijsku mapu C-64. MMU proverava memorijske adrese pre nego što ih procesor i vidi. To omogućava da se koristi više blokova po 64 K, bez razmišljanja o tome u kom se od njih nalaze nulta stranica i stek. Osim toga, MMU dozvoljava i korišćenje više banaka podataka po 64K, sve do 1 megabajta.

MMU kontroliše i rad VIC čipa, kao i video čipa za 80 znakova u redu. Programer može sam da izabere sa kojim bankama će raditi i direktno se obratiti MMU. Adresa VIC čipa može se smestiti u bilo koji deo u prvih 256K radne memorije.

MMU kontroliše i serijski I/0 port za disk drajv 1571 (i sve ostale periferne uređaje). On podešava klok na kojem radi 8502 i kontroliše aktivnost sva tri procesora (8502, 6510, Z80).

Novi „komodor" je opremljen ne samo izvanrednim bejzikom nego i lepim pogodnostima za rad na mašinskom jeziku. Procesor 8502 je potpuno kompatibilan sa 6502. Korisniku je na raspolaganju mašinski monitor koji je ugrađen u memoriju računara. Sličan je kao 64 SUPERMON i sadrži sve što jedan dobar monitor treba da ima. Ima komande za asembliranje, disasembliranje, popunjavanje memorije, startovanje programa, za pretraživanje memorije, za pomeranje memorije, učitavanje snimanje i verifikaciju programa i još mnogo toga. Naravno, može se raditi u binarnom decimalnom i heksadekadnom kodu i u svakom trenutku mogu se dobiti svi podaci o računaru.

Bejzik komande za rad u mašinskom jeziku sadrže BLOAD i BSAVE, za binarno snimanje deloa memorija, kao i BOOT za učitavanje i automatsko startovanje mašinskih programa. Familijarne komande, kao što su USR, WAIT, PEEK, POKE i SYS sada mogu koristiti i ostale blokove po 64K pomoću komande BANK. SYS komanda može biti praćena sa četiri parametra, koji će biti smešteni u akumulator, X registar, Y registar i u status flag registar. Posle SYS, komandom PREG vrednosti ovih registara se mogu vratiti u četiri varijabile. Ovo u mnogome olakšava kombinovanje bejzik i mašinskih programa.

Procesor 8502 je kompatibilan sa 6502, ali radi na 2 MHz — znači dva puta brže nego 6510 kod C-64 ili 6502 kod VIC-20. Sve VIC/64 Kernal rutine su iste, što olakšava prevođenje programa.

RESET taster je ugrađen u računar sa desne strane, kod prekidača za uključivanje, i može izvršiti tzv. hidani start mašine. Osim reseta sa RUN-STOP-RESTORE, može se izvršiti i jedan još delotvorniji oporavak računara. Pritisak na RUN-STOP i RESET prebacuje u mašinski monitor iz koga se (lako) izazi (jednom komandom). Bejzik ili bilo koji drugi program ostaju u memoriji nepromenjeni i neoštećeni.

stiven džobs — programer koji je hteo da promeni svet

Nigde, valjda, toliko često ne padaju krunisane glave kao u industriji kućnih računara. Pre nekoliko meseci na spisku penzionisanih kraljeva našao se i Stiven Džobs, projektant čuvene „jabuke" i osnivač kompanije „Apple". Dok je boravio u Japanu u „misiji dobre volje", Džobsa je u dinastija-stilu svrgao sa prestola Džon Skuli, čovek koji zna sve o tržištu i gotovo ništa o računarima. Dodeljena mu je utešna kruna, utešna kancelarija i utešna sekretarica. Ovaj razgovor sa Džobsom, objavljen u časopisu „Playboy", vođen je pre prevrata, krajem prošle godine — u trenutku kada se mislilo da je „Apple" definitivno isplivao iz teškoća u koje su ga uvalili računari „liza" i „epl III". Govoreći o sadašnjosti i, još više, o budućnosti kućnih računara, Džobs otkriva suptilne zakonitosti i mehanizme u ovoj mladoj industriji, u kojoj su jabuke, izgleda, znatno kiselije nego što to na pravi pogled izgleda.

Ako se i za koga može reći da predstavlja duh poduzetničke generacije, onda je čovjek kojeg treba tražiti karizmatički suosnivač i predsjednik Apple Computer, Inc., Stiven Džobs. On je transformisao malo preduzeće, koje je nastalo u garaži u Los Altosu, Kalifornija, u revolucionarnu kompaniju od milijardu dolara — onu koja je dostigla nivo Fortune 500 (lista od 500 najvećih kompanija koju objavljuje časopis Fortune) za samo pet godina, brže nego i jedna druga kompanija u istoriji. I ono što najviše pogađa u cijeloj stvari je to da je ta osoba stara samo 29 godina.

Džobsova kompanija je uvela personalne računare u američke kuće i radne prostorije. Prije osnivanja Apple-a 1976, slika koju su ljudi imali o računarima je ona o mašinama iz naučnofantastičnih filmova koje su davale svjetlosne i zvučne signale ili, pak, o onim velikim mašinama koje u tišini zloslutno čame iza vrata velikih korporacija ili vladinih agencija. Ali sa razvojem tranzistora, a zatim i mikroprocesorskog čipa, postalo je moguće minijaturizirati tehnologiju računara i učiniti ih dostupnim pojedinačnim korisnicima. Do sredine sedamdesetih godina, početni računarski kit, uglavnom interesantan samo hobistima, mogao se nabaviti za oko 375 dolara, plus dodatni pribor.

U dolini južno od San Franciska, već poznatoj po koncentraciji elektronskih firmi i mladih kompanija koje su tek startale sa radom, dva prijatelja koja su dijelila sklonost za neprilike i elektroniku dala su se na posao da stvore vlastiti računar. Džobs, tada star 21 godinu, usvojeni sin mašiniste, preuzeo je posao dizajniranja video igara kod Atarija nakon što je napustio REED koledž, dok je Vozniak (Stephen Wozniak), 26, radio kao inženjer kod Hewlett-Packard-a, jedne od najvećih kompanija u regiji zvanoj Silicijanska dolina. U svoje slobodno vrijeme prijatelji su projektovali i izgradili improvizirani računar, u stvari samo ploču sa elektronskim krugovima, koju su hirovito nazvali „eple I". On nije radio mnogo stvari, ali kada su se suočili sa listom porudžbina od 50 komada Džobsu je svanulo da se, možda, stvara rastuće tržište personalnih računara.

Vazniakov interes je bio primarno tehnički; Džobs se bacio na to da učini računar pristupačnim ljudima. Zajedno su dodali tastaturu i memoriju (na „eple I", a Vozniak je razvio disk drajv) i dodao video terminal. Džobs je angažirao eksperte da osiguraju sigurno napajanje i lijepo kućište i tako je rođen „epl II" — zajedno sa cijelom industrijom.

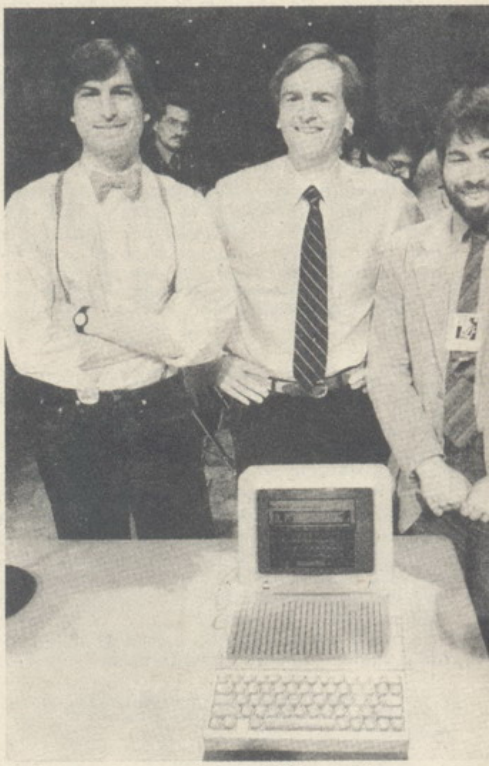
Uspon Apple-a je bio meteorski. Od prodaje od 200.000 dolara te prve godine u Džobsovoj garaži (verzija Linkolnovog uspjeha ali sad u Silicijumskoj dolini) kompanija je narasla u gigantsku firmu sa prihodom od 1,4 milijarde dolara u 1984. godini. Njeni osnivači su postali multimilioneri i narodni heroji. Vozniak koji se stvarno povukao iz Apple-a 1979. da bi se vratio na koledž i posvetio sponzorstvu nad muzičkim festivalima, imao je malo posla sem svog kreativnog doprinosa tehnologiji. Džobs je ostao da vodi kompaniju, da vidi da 70% kućnih i školskih računara nose znak Apple-a, da se bori protiv snaga u Apple-u koje žele da ga detroniziraju, i, najviše od svega, da se bori sa IBM-om, divom od 40 milijardi dolara, kako ga inače zovu, od kada je odlučio da uđe u posao sa personalnim računarima.

Sa procijenjenim čistim imetkom od 450.000.000 dolara, uglavnom u akcijama Applea. Džobs je daleko najmlađa osoba na Forbesovoj listi najbogatijih Amerikanaca već nekoliko posljednjih godina. (Vrijedno je napomenuti da je od 100 Amerikanaca koje imenuje Forbes, Džobs jedan od sedmorice koji su svoj imetak zaradili sami.) Nedavno, sa padom vrijednosti akcija Apple-a, za vrijeme teškoća u 1983. on je izgubio skoro četvrt milijarde dolara, na papiru, tako da mu se čist imetak danas procjenjuje na 200.000.000 dolara.

Ali treba vjerovati Džobsu da mu novac ne znači mnogo, pogotovo jer ga ne troši pretjereno i zaista nema mnogo vremena za društveni život. On ima misiju — propovjeda spasenje uz pomoć personalnog računara, po mogućnosti Apple-ove proizvodnje. On je vrlo angažirani prodavac i nikad ne propušta priliku da proda svoj proizvod, rječito opisujući vrijeme kada će računari biti u tako širokoj upotrebi kao kuhinjski aparati i tako revolucionarni po svom uticaju kao telefon ili motor sa unutrašnjim sagorjevanjem. Činjenica je da trenutno ima više od 2 000 000 „epl" računara i, procjenjuje se, 16000 programa — u učionicama, dnevnim sobama u predgrađu, farmama, stanicama za lansiranje raketa, te malim i velikim preduzećima diljem Amerike.

Stvarajući široko tržište računara, Apple je stvorio i uslove za konkurenciju i mnoštvo kompanija je ušlo u borbu za osvajanje tržišta na kome je Apple dominirao do 1977. do 1982. Ali ni jedan drugi proizvod nije bio tako uspješan kao IBM-ov PC, koji je ubrzo zauzeo 28% tržišta, uspostavljajući novi standard. U uslovima smanjivanja svog tržišta, Apple je predstavio dva nova računara, „lisu" i „eple III", ali prijem nije bio oduševljen. Sredinom 1983. analitičari su se glasno pitali da li će Apple preživjeti.

Tokom trvenja unutar kompanije, Džobs je preuzeo odjeljenje Apple-a, koje je gradilo kompletno nov računar, za koji je on mislio da je Apple-ova najbolja i posljednja šansa. To nije bila uskogrudnost, kaže on, jer da su propali, „IBM bi ostao da dominira i upropastio bi industriju". Nakon tri godine, lansiran je „mekintoš" (Macintosh) uz reklamnu kampanju od 20 000 000 dolara. Objavljen kao „računar za nas ostale", razglašen je kao gigantski korak u



(Ne) složna braca: Stiven Džobs, novi predsednik „Apple"-a Džon Skuli i Stiven

proizvodnji računara koji su laki za korištenje. Sa ekranom bijelim kao papir, malim sličicama koje predstavljaju programske opcije i „mišem" „mek" je, sigurno, najmanje zastrašujući računar koji je ikad napravljen. Bio je, također, kritikovan da je isuviše igračka i nepodesan za ozbiljnu poslovnu upotrebu. Uprkos žestini tih argumenata, Apple je vrijedno proizvodio 40 000 „mekintoša" mjesečno, a ove godine ima namjeru da proizvodnju udvostruči.

Zavisno od toga s kim razgovarate, za jedne je Džobs vizionar koji je promijenio svijet na bolje, a za druge oportunista čije su vještine u marketingu dovele do nevjerovatnog komercijalnog uspjeha. U farmerkama i iznošenim tenisicama, on vodi kompaniju koja se ponosi time da ima u sebi entuzijazam šezdesetih i poslovnost osamdesetih, tako da mu se dive i da ga se boje. „On je taj zbog koga radim 20 sati dnevno", kaže jedan inženjer. Ili, kako Michael Moritz izvještava u „Malom kraljevstvu", Džobsova tvrdoglavost — da kuje hvalospjeve, jedan dan, a

→

izvrgava ruglu drugi dan — umalo je dovela do raspada tima koji je radio na „mekintošu". On je nepostojanog predsjednika Pepsi Cole, Džona Skulija (John Sculley) pitao da preuzme administrativnu upravu Apple-a riječima: „Hoćeš li da nastaviš da prodaješ djeci zaslađenu vodu, kad možeš da mijenjaš svijet?". Skuli je prihvatio ponudu.

Ovaj „intervju" je bio jedan od malo onih u mom životu kada mi se stalno događalo da budem pretjerano obučen. Čuo sam o neformalnosti koja vlada u Apple-u, ali ipak ja sam intervjuisao glavu kompanije od milijardu dolara, pa sam na prvi sastanak došao sa kravatom. Naravno, kad sam sreo Džobsa u njegovoj kancelariji u Kapertinu, Kalifornija, on je nosio flanelsku košulju i farmerke. Još se uvijek nisam osjećao deplasirano — dok nisam sreo Džona Skulija, novog predsjednika Apple-a: On je nosio majicu sa kratkim rukavima.

Kancelarije u Apple-u definitivno ne nalikuju ostalim radnim mjestima. Svuda naokolo video igre, stolovi za stoni tenis su u stalnoj upotrebi, zvučnici trešte od muzike Rolling Stones-a do Windham Hill džeza. Sale za sastanke se zovu po Da Vinčiju i Pikasu, a frižideri u snek prostorijama su pretrpani svježim sokovima od mrkve, jabuke i naranče. (Samo „mek" tim potroši 100 000 dolara godišnje na sokove).

Nadugo sam razgovarao sa Džobsom i na poslu i na njegova jedina dva odmora tokom godine u Aspenu i banji Sonoma, gdje je trebalo da se opusti. Nesposoban da uspori svoju misiju da širi riječ Apple-a, sa izuzetnom oštrinom je govorio o svom ratu sa IBM-om, da bi onda podcrtao svoj entuzijazam za neku ideju riječima „Čisto!" ili „Nevjerovatno grandiozno!"

• Preživjeli smo 1984, a računari nisu osvojili svijet, mada neki ljudi u to teško vjeruju. Ako postoji pojedinac koga treba bilo optužiti bilo hvaliti za proliferaciju računara, vi, 29-togodišnji otac kompjuterske revolucije, ste prvooptuženi. To vas je, također, učinilo bogatim preko svih snova — vaše akcije su u jednom momentu vrijedile skoro pola milijarde dolara, zar ne?

— Ja sam, u stvari, izgubio 250 000 000 dolara u jednoj godini kad su akcije pale (smije se).

• Možete da se na to smijete?

— Neću dozvoliti da mi to upropasti život. Zar to nije na neki način i smješno? Znate, moja osnovna reakcija na to pitanje novca je da je to smiješno, sva ta pažnja koja se posvećuje tome, jer teško da je to najveće saznanje ili najvrijednija stvar koja mi se desila u posljednjih deset godina. Osećam se ponekad starim, kad govorim na univerzitetu i uvidim da najviše strahopoštovanja studentima uliva činjenica da sam milioner.

• Interesantno je da je kompjutersko područje učinilo milionere od . . .

— Mladih manijaka, znam.

• Htjeli smo da kažemo, od ljudi kao što ste vi i Stiv Voznijak koji su radili u garaži prije tek deset godina. Šta je, u stvari, ta revolucija koju ste započeli vas dvojica?

— Mi živimo nad opelom petrohemijskoj revoluciji od prije 100 godina. Petrohemijska revolucija nam je dala jeftinu energiju — jeftinu mehaničku energiju u ovom slučaju. To je promijenilo teksturu društva u raznim vidovima. **Ova revolucija, informatička revolucija, je revolucija jeftine energije također, ali energije druge vrste: jeftine intelektualne energije.** Ona je još vrlo sirova danas, a ipak računar „mekintoš" troši, da bi radio, manje energije nego sijalica od 100 vata, a može da vam uštedi sate dnevno. Šta će biti u stanju da uradi za deset ili dvadeset godina od danas? Ova revolucija će zasjeniti petrohemijsku revoluciju. Mi smo u prvoj liniji fronta.

14) Stiven Džobs

*• Kako bi bilo da nam date neki **konkretan** razlog zašto kupiti računar danas? Jedan rukovodilac iz vaše branše je nedavno rekao: „Dali smo ljudima računare, ali im nismo pokazali šta da rade sa njima. Ja mogu bilansirati svoju čekovnu knjižicu ručno brže nego pomoću računara. „Zašto bi neka osoba trebala kupiti računar?*

— Postoje različiti odgovori za razne osobe. U poslu, na to pitanje je lako odgovoriti: vi stvarno možete dokumente pripremiti mnogo brže i sa većim nivoom kvaliteta i možete uraditi mnogo stvari da povećate produktivnost biroa. Računar oslobađa ljude od mnogo prostog posla. Sem toga, dajete im alat koji ih ohrabruje da budu kreativni. Sjetite se, računari su alatke. Alati nam pomažu da bolje obavimo svoj posao. U obrazovanju, računari su prva stvar koja se pojavila poslije knjige, koja ostaje sa vama u interakciji do u beskraj. Sokratovo podučavanje se više ne može obezbijediti, a računari imaju stvarni potencijal da predstavljaju prvi proboj u obrazovnom procesu kada se koriste u vezi sa prosvijećenim učiteljima. Mi smo već u mnogim školama.

• To su argumenti za uvođenje računara u posao i škole, ali šta kažete za kuće?

— Za sada, to je tek tržište u začetku. Primarni razlog da kupite računar za svoju kuću je taj da želite da radite neki posao kod kuće ili želite da koristite neki obrazovni softver za sebe ili svoju djecu. Ako kupovinu računara ne možete opravdati jednim od ova dva razloga, jedini drugi razlog je taj da želite da budete računarski pismeni. Znate da se nešto dešava, ali ne znate tačno šta je to i stoga želite da naučite. Ovo će se izmijeniti. Računari će biti neophodni u većini kuća.

„Novac nije najveće saznanje niti najvrednija stvar koja mi se desila u poslednjih deset godina"

• Šta će se promijeniti?

— Najizazovniji razlog za većinu ljudi, da kupe računar za kuću, će biti veza na nacionalnu komunikacijsku mrežu. Upravo se nalazimo u početnom stadiju onoga što će biti istinski zavidan napredak za većinu ljudi — isto toliki koliki je bio telefon.

• O kakvoj, specifično, vrsti napretka govorite?

— Mogu samo početi da spekulišem. Dosta toga vidimo u našoj industriji. Ne znate tačno čime će to rezultirati, ali znate da je to nešto veoma veliko i veoma dobro.

• Dakle, zar vi, za sad, ne tražite od kupaca kućnih računara da investiraju 3000 dolara u nešto što je, u biti, akt vjerovanja?

— U budućnosti to neće biti akt vjerovanja. Teškoće sa kojima se sada srećemo su te da ljudi traže da im govorimo o detaljima o kojima ne možemo govoriti. Da je prije sto godina neko upitao Bela: „Šta ćete moći uraditi sa telefonom?", on ne bi bio u stanju da im objasni u kojim sve vidovima će telefon uticati na svijet. On nije znao da će ljudi koristiti telefon da saznaju koji film igra večeras, da kupe neku robu ili da porazgovaraju sa rodbinom na drugoj strani globusa. Ali, prisjetite se da je prvi javni telegraf bio inaugurisan 1844. To je bio začudujući proboj u komunikacijama. Mogli ste tada poslati poruku iz Njujorka u San Francisko u toku jednog popodneva. Ljudi su govorili o tome da treba staviti po jedan telegraf na svaki radni stol u Americi, da bi se povećala produktivnost. Ali to nije radilo. Bilo je potrebno da ljudi nauče cijeli niz čudnih znakova, Morzeovu abecedu, tačkica i crtica, da bi koristili telegraf. Trebalo je oko 40 sati da se to nauči. Većina ljudi ne bi nikad to naučila. Ali, srećom, 1870 Bel, je patentirao telefon. On je, u osnovi, radio isto što i telegraf, ali su ljudi već znali kako da ga koriste. Također, sem što je dozvolio komunikaciju i riječima, dozvolio je i mogućnost da pjevate.

• Šta sad to znači?

— Omogućio je da date riječima intonaciju sa značenjem koje je iznad čisto lingvističkog. Mi smo u istoj situaciji danas. Neki ljudi misle da na svaki radni stol u Americi treba staviti IBM-PC da bi se povećala produktivnost. To neće funkcionirati. Specijalne čarolije koje ovog puta treba da naučite su „crtica q-zs" i slične stvari. Uputstvo za WordStar, najpopularniji program za obradu teksta, debelo je 400 stranica. Da biste napisali roman morate pročitati roman — i to roman koji je kao misterija za mnoge ljude. Oni neće željeti da nauče crtica q-sz nimalo više nego što su željeli da nauče Morzeovu abecedu. To je ono što čini „mekintoša". To je prvi „telefon" naše industrije. A osim toga, za mene je najvažnije, da „mekintoš" dozvoljava da pjevate na isti način na koji je to činio telefon. Ne komunicirate jednostavno riječima — imate specijalne stilove štampanja i mogućnost da crtate i dodate slike da biste se bolje izrazili.

• Većina kompjutera koristi tastaturu za unošenje instrukcija, a „mekintoš" je zamjenjuje nečim što se naziva miš — malom kutijom koja se kotrlja po radnom stolu i time vodi kursor po ekranu. To je velika promjena za ljude koji su navikli na korištenje tastature. Zašto miš?

— Ako želim da vam kažem da imate mrlju na košulji, neću to reći riječima: „Ima jedna mrža na vašoj košulji 14 cm niže od kragne i tri centimetra lijevo od dugmeta." Ako ima mrlje — ovde (on pokazuje) — ja ću vam to pokazati prstom. Pokazivanje prstom je metafora koju svi mi znamo. Izradili smo mnogo studija i ispitivanja o tome i ustanovljeno je da se sve vrste funkcija izvode ne samo lakše nego i mnogo brže uz pomoć miša.

• Koliko je trebalo vremena da se razvije „mekintoš"?

— Za izradu samog računara trebalo nam je više od dvije godine. Na tehnologijama koje su primjenjene radili smo godinama prije toga. Rad na „mekintošu" bio je najljepše iskustvo mog života. Skoro svako ko je na njemu radio reći će to isto. Na kraju, niko od nas nije htio da ga pusti. Kao da smo znali da, kad ga jednom ispustimo iz ruku, više nikad neće biti naš. **Kad smo ga, konačno, prikazali na sastanku akcionara, svi u auditorijumu su ustali i pozdravili ga petominutnim ovacijama. Svi su počeli da plaču.**

• Bili smo upozoreni na vas: prije nego što je ovaj intervju započeo, neko je rekao da „ćemo biti zasuti lavinom superlativa".

— Mi smo samo zaneseni onim što radimo.

• Ali baš imajući u vidu taj entuzijazam, reklamnu kampanju od više miliona dolara, i vašu vlastitu sposobnost da pridobijete publicitet štampe, kako da potrošač zna šta se krije iza sve te buke?

— Reklamne kampanje su neophodne radi konkurencije — reklame IBM-a su svuda. Ali dobro propaganda obrazuje ljude i to je sve. U ovom poslu ne možete zavarati ljude. Proizvodi govore sami za sebe.

• Pored nekih ponovljenih kritika — da je miš neefikasan, da je ekran „mekintoša" samo crno-bijeli — najozbiljnija optužba je da „eple" precjenjuje svoje proizvode. Bi li vam smetalo da odgovorite na neku ili sve ove optužbe?

„Ova revolucija, informatička revolucija, jeste revolucija jeftine energije također, ali energije druge vrste: jeftine intelektualne energije"

Kralj u prinudnoj penziji: Stiven Džobs

— Obavili smo studije koje dokazuju da je miš brži nego tradicionalni načini prolaza kroz podatke ili razne primjene. Jednog dana ćemo, možda, biti u stanju da ugradimo kolor ekran uz razumnu cijenu. Što se tiče previsoke cijene, start novog proizvoda ga čini skupljim nego što će biti kasnije. Što više budemo proizvodili, imaćemo nižu cijenu.

• To i jeste ono za što vas optužuju kritičari: pecate zanesenjake — vrhunskim cijenama, a onda se okrecete i snižavanjem cijena hvatate ostatak tržišta.

— To jednostavno nije istina. Čim **to možemo**, mi snižavamo cijene. Istina je da su naši računari jeftiniji danas nego prije nekoliko godina, ili čak lani. Ali to je također istina i za IBM PC-a. Naš cilj je da dovedemo računare do desetine miliona ljudi, i što ih jeftinije proizvodimo, lakše nam je da taj cilj ostvarimo. Ja bih volio kad bi „mekintoš" mogao da košta 1000 dolara.

• Šta je sa ljudima koji su kupili „lizu" „epl III" — dva računara koja ste izbacili na tržište prije „mekintoša"? Ostavili ste ih sa nekompatibilnim i zastarjelim proizvodima.

— Ako želite tako da govorite, onda dodajte na tu listu i ljude koji su kupili IBM PC-a ili PC juniora. Što se tiče računara „liza", pošto je dio iste tehnologije korišten i za „mekintoša", on može da koristi softver „mekintoša" i mi na njega gledamo kao na „mekintoševog" starijeg brata. Iako je u početku bila neuspješna, prodaja „lize" je probila plafon. Također, još uvek prodajemo preko 2000 „epla III" mjesečno — više od pola starim kupcima. Konačni zaključak je da nova tehnologija ne mora značiti zamjenu stare, nego joj se pridružuje. Po definiciji. Ponekad, ona je zamjenjuje. Ali to je onda slučaj kao sa ljudima koji su imali crno-bijeli televizor kada se pojavio kolor. Oni su tada zaključili da li im se investicija u novu tehnologiju isplati ili ne.

• Po brzini kojom se stvari mijenjaju, zar nece i sam „mek" biti zastario za nekoliko godina?

— Prije „mekintoša" postojala su dva standarda — „epl II" i IBM PC. Ta dva standarda su kao dvije rijeke koje su se urezale u stijenu kanjona. Trebale su godine da se one usijeku u taj kamen — sedam godina za „epl II" a četiri za IBM. Ono što smo uradili sa „mekintošem" je da smo u manje od godinu dana, silom revolucionarnog aspekta samog proizvoda i svom raspoloživom energijom marketinga koju smo imali kao kompanija, bili u stanju da probijemo treći kanal kroz tu stijenu i stvorimo treću rijeku, treći standard. Po mom mišljenju, danas postoje samo dvije kompanije koje su u stanju da to urade: Apple i IBM. Možda je to, zaista, loše, ali da se to sad ponovi, predstavlja monumentalan napor i ja ne mislim da će to Apple ili IBM uraditi u narednih tri ili četiri godine. Pred kraj osamdesetih cemo možda vidjeti neke nove stvari.

• A u međuvremenu?

— Razvoj će biti u tome da se proizvodi naprave sve više i više prenosivim, da se povežu u mreže, da se ostvare laserski pisači, da se dobiju razdijeljene baze podataka, da se dobiju veće mogućnosti komuniciranja, možda da se spoje telefon i lični računar.

• Još je dug put pred njim. Neki ljudi su rekli da ce „mekintoš" stvoriti ili slomiti Apple. Nakon „lize" i „epla III", akcije Apple-a su strmoglavo pale i industrija je spekulirala da Apple nece preživjeti.

— Da, osjetili smo težinu svijeta na našim plecima. Znali smo da moramo izvući zeca iz šešira sa „mekintošem", ili da nikad nećemo ostvariti snove vezane za sam taj proizvod i za samu kompaniju.

• Koliko je to bilo ozbiljno? Da li je Apple bio blizu bankrotstva?

— Ne, ne, ne. U stvari, 1983, kada su sva ova predviđanja nastala, bila je fenomenalno uspješna godina za Apple. U 1983. smo gotovo udvostručili našu veličinu. Od 583 000 000 dolara u 1982. prodaju smo u 1983. povećali na oko 980 000 000 dolara. To je skoro sve bilo povezano sa „eplom II". Samo se nisu ostvarila naša očekivanja. Da „mekintoš" nije bio uspješan, vjerovatno bi ostali na prodaji od oko milijardu dolara godišnje, prodajući „epl II" i njegove verzije.

• Šta je onda bilo pozadina priča da je Apple gotov?

— IBM je ušao veoma silovito i teg je prešao na njegovu stranu. Proizvođači softvera su se okrenuli IBM-u. Preprodavci su sve više i više govorili o IBM-u. Postalo je jasno svima nama koji smo radili na „mekintošu" da će to rasturiti industriju, da će to redefinisati industriju. A upravo je to i želio da uradi. Da „mekintoš" nije bio uspješan, ja bih morao baciti peškir (u znak predaje), jer bi to značilo da moja vizija industrije nije ispravna.

• „Epl III" je trebao biti vaš poboljšani „epl II", ali je bio promašaj od kad je lansiran prije četiri godine. Povukli ste prvih 14000 komada, pa i revidirani „epl III" nije nikad uzletio. Koliki je bio gubitak na „eplu III"?

— Beskonačan, neizračunljiv iznos. Mislim da bi IBM, da je ovaj računar bio uspješniji, imao daleko teži posao da uđe na tržište. Ali, to je život. Mislim da smo isplivali iz ovog iskustva mnogo jači.

• Još kad se pojavila „liza", koja je također bila relativan neuspjeh na tržištu! Šta se to desilo?

— Kao prvo, bila je previše skupa — oko deset hiljada dolara. Uhvatili smo se kompanija sa liste 500 najvećih časopisa FORTUNE, pokušavajući da prodajemo njima, dok su naši korjeni da prodajemo narodu. Bilo je i drugih problema: kašnjenje u isporukama, softver nije ispao tako dobar kako smo se nadali i . . . izgubili smo zalet. A IBM je ušao veoma snažno. Mi smo odlučili da unajmimo ljude za koje smo mislili da su eksperti u marketingu i menadžmentu. Nije to bila loša ideja, ali, na nesreću, to je bio tako nov posao da se stvari koje su takozvani profesionalci znali išle samo na uštrb njihovom uspjehu u ovom novom načinu gledanja na posao.

• Da li je to bio odraz nesigurnosti kod vas samih — „Stvari su postale krupne i sada se odigrava ključna lopta: bilo bi bolje da uvedemo u igru prave profesionalce"?

— Prisjetite se, mi smo imali po 23, 24 i 25 godina. Ništa od toga nismo ranije radili, tako da se činilo da bi to bila dobra stvar.

• Da li je većina tih odluka, dobrih i loših, bila vaša?

— Trudili smo se da nikad jedna osoba ne donosi sve odluke. U to doba su tri osobe vodile kompaniju: Majk Skot, (Mike Scott), Majk Marakula (Mike Marakkula) i ja. Sada smo to Džon Skuli (John Sculley), predsjednik Apple-a, i ja. U ranim danima, ako je bilo neslaganja, ja bih obično podčinjavao svoj sud onome koji su imali ljudi sa većim iskustvom od mog. U mnogim slučajevima oni su bili u pravu. U nekim važnim slučajevima, da smo išli mojim putem, uradili bi bolje.

• Zašto na polju računara dominiraju tako mladi ljudi? Srednji vijek zaposlenih u Apple-u je 29 godina.

— Kako postaju stariji, ljudi se sve teže mijenjaju. Naši mozgovi su neka vrsta elektrohemijskih računara. Vaše misli konstruišu oblike u mozgu nalik na skele. U stvari, stvarate trajne hemijske odlike. U većini slučajeva, ljudi bivaju uhvaćeni u te oblike, nalik na zapis na gramofonskoj ploči, i nikad ne uspijevaju da iz njih pobjegnu. Rijetke su osobe koje stvaraju taj zapis različit od uobičajenog pogleda na stvari, uobičajenog načina ispitivanja stvari. Rijetkost je da sretnete nekog umjetnika u njegovim 30-setim ili 40-desetim sposobnog da doprinese nešto izvanredno. Naravno, postoje ljudi koji su po prirodi znatiželjni, zauvijek mala djeca cijelog svog života, ali oni su rijetki.

• Mnoge osobe u 40-desetim godinama će biti zaista oduševljeni vama. No, pređimo na drugu stvar o kojoj ljudi govore kada se spomene Apple. Vi imate sličan misionarski poriv i u pogledu načina na koji se radi u Apple-u, zar ne?

— Ja zaista osjećam da postoji i drugi uticaj koji mi imamo na društvo sem onog koji čine naši računari. Mislim da Apple ima šansu da bude model Fortune 500 kompanije u kasnim osamdesetim i ranim devedesetim godinama. Prije deset do petnaest godina, da ste pitali ljude da vam sastave listu od pet „najuzbudljivijih" kompanija u Americi, Xerox i Polaroid bi bili na svakoj listi. Gdje su oni sad? Danas ne bi bili ni na jednoj listi! Šta se to dogodilo? Kompanije, **kako rastu da bi postale entiteti od više milijardi dolara, na neki način gube svoje vizije.** One umeću mnogo slojeva srednjeg upravljačkog kadra između ljudi koji vode kompaniju i ljudi koji obavljaju posao. **One više nemaju usađen osjećaj pasije za proizvod koji rade.** Kreativni ljudi, a to su oni koji sa žarom brinu o stvarima, treba da ubijede pet slojeva menadžera da urade ono za što oni znaju da se treba uraditi. Ono što se dešava u većini kompanija je da se pojedinačno postignuće ne podržava nego baš obeshrabruje. Kreativni kadrovi odlaze i vi završavate sa mediokritetima. Ja to znam, jer je tako nastao

→

15) programer koji je hteo da izmeni svet

Apple. **Apple je Ostrvo izbjeglica. Apple je sastavljen od izbjeglica iz drugih kompanija. To su krajnje nadareni individualisti koji doprinose uspjehu, a koji su stvarali probleme u drugim kompanijama.** U svakom slučaju, jedan od najvećih izazova po kome treba ocjenjivati Džona Skulija i mene za pet do deset godina je da napravimo od Apple-a kompaniju od nevjerovatno velikih deset ili dvadeset milijardi dolara. Da li će ona i tada imati duh koji ima danas? Mi iskolčavamo novu teritoriju. Nema modela na koje se možemo ugledati za neke nove koncepte upravljanja koje imamo da bi omogućili tako velik rast. Stoga moramo naći svoj vlastiti put.

• Ako je Apple stvarno takva kompanija, zašto onda projicirati dvadesetostruki rast? Zašto ne ostanete relativno mali?

— Način na koji to može funkcionirati u našem poslu, da bismo nastavili da budemo jedan od glavnih nosilaca napretka, jeste da postanemo kompanija od deset milijardi dolara. Taj rast nam je potreban da bismo izdržali konkurenciju. Nas više brine kako da to postignemo, nego sama zarada, koja je za nas beznačajna. U Apple-u ljudi rade po 18 časova dnevno. Mi privlačimo drugačiji tip osoba — osobe koje ne žele da čekaju pet do deset godina da bi se našao neko ko će preuzeti gigantski rizik za njih. Neko ko stvarno želi da se propne malo iznad svoga stasa i zagrize malo u univerzum. **Mi smo svjesni da radimo nešto značajno.** Mi smo tu na njegovom početku i u stanju smo da mu damo oblik daljeg razvoja. Svako je ovde uvjeren da je upravo sada moment kada utičemo na budućnost. **Veći dio vremena mi samo uzimamo stvari.**

„Apple je sastavljen od izbjeglica — krajnje nadarenih individualista koji su stvarali probleme u drugim kompanijama"

Ni vi ni ja nismo napravili odijela koja nosimo; mi ne pravimo hranu niti uzgajamo biljke koje jedemo; koristimo jezik koji su razvili drugi; koristimo matematiku drugih društava. **Veoma rijetko imamo šansu da stavimo nešto u taj bazen iz koga uzimamo.** Mislim da sad imamo tu priliku. Ali, ne, mi ne znamo gdje će to dovesti. Mi samo znamo da je to nešto mnogo veće od bilo koga od nas ovde.

• Da bi ova industrija stvarno procvjetala i da bi to stvarno koristilo potrošačima, rekao je jedan ekspert, treba da prevlada jedan standard.

— To, jednostavno, nije istina. Insistirati na tome da nam treba jedan standard je isto kao i kazati da je u automobilskoj industriji 1920. godine trebao jedan standard. Da su oni u to tada vjerovali, ne bi bilo nikakvih inovacija, kao što su automatska transmisija, neovisno vješenje i sl. Posljednje što želimo je da zamrznemo tehnologiju. **Među računarima „mekintoš" je revolucionaran. Uopšte se ne postavlja pitanje oko toga da je tehnologija „mekintoša" superiorna nad IBM-ovom. Postoji jasna potreba za alternativom IBM-u.**

• Zar vaša odluka da ne budete kompatibilni sa IBM-om ni u jednom momentu nije bazirala na činjenici da niste željeli da se pokorite IBM-u? Jedan kritičar kaže da je razlog zašto „mek" nije IBM kompatibilan čista arogancija — i da je „Stiven Džobs rekao IBM-u 'Da se . . . .'"

— Nismo dokazivali svoju muškost željom da budemo različiti.

• Pa zašto ste onda drugačiji?

— Glavni razlog je, jednostavno, taj da je tehnologija koju smo razvili superiornija. Ona,

„Japanci uzmu nešto što je već izmišljeno i studiraju ga dok ga u potpunosti ne shvate. Iz tog razumjevanja oni ga ponovo izmišljaju u mnogo rafiniranijoj verziji druge generacije"

možda, ne bi bila tako dobra da smo trebali biti kompatibilni sa IBM-om. Naravno, tačno je da mi ne želimo da IBM dominira ovom industrijom. **Mnogo ljudi misle da smo ludi što nismo IBM kompatibilni i što ne živimo pod kišobranom IBM-a.** Postoje dva glavna razloga što to nismo uradili: prvi je što smo mislili i ja mislim da će budućnost pokazati da smo bili u pravu — da IBM može zatvoriti taj kišobran nad kompanijama koje proizvode kompatibilne računare i tako ih uništiti. Drugi i mnogo važniji — mi nismo išli putem kompatibilnosti sa IBM-om zbog vizije proizvoda koja je pokretačka snaga ove kompanije. Mi mislimo da su računari najizvanrednije alatke sa kojima se čovječanstvo srelo i mi mislimo da su ljudi, u osnovi, korisnici alata. Tako, ako budemo u stanju da mnogo računara, damo velikom broju ljudi, to će dovesti do kvalitetivne razlike u svijetu. Ono što mi želimo je da od računara stvorimo dio kućnog namještaja i da ga dovedemo do desetine miliona ljudi. Jednostavno, to želimo da uradimo. A to ne bismo mogli uraditi sa tekućom generacijom IBM tehnologije. Stoga smo morali da izradimo nešto drugo. Zato smo proizveli „mekintoša".

• Kao i kompjuterska industrija, automobilska industrija je bila američka industrija koju smo skoro izgubili — preuzeli su je Japanci. Ima mnogo priče oko toga da američke kompanije, proizvođači poluprovodnika, gube tlo pod nogama zbog Japanaca. Kako ćete vi izdržati?

— Japan je veoma interesantan. **Neki ljudi misle da oni kopiraju stvari.** Ja to više ne mislim. **Ja mislim da oni ponovo izmišljaju stvari. Oni uzmu nešto što je već izmišljeno i studiraju to dok ga u potpunosti ne shvate.** U nekim slučajevima ga shvate bolje nego originalni pronalazač. **Iz tog razumijevanja oni ga ponovo izmišljaju u mnogo rafiniranijoj verziji druge generacije.** Ta strategija funkcioniše samo onda kada se ono na čemu oni rade ne mijenja brzo — industrija stereo uređaja i automobilska industrija su dva primjera za to. Kada se cilj brzo kreće, to im je jako teško, jer ciklus reinvencije traje nekoliko godina. Dok god se definicija ličnog računara nastavlja mijenjati brzinom kojom se trenutno mijenja, oni će imati teška vremena. Jednom kada se brzina promjena uspori, Japanci će baciti sve svoje snage da se probiju na ovo tržište. Oni apsolutno žele da dominiraju ovim poslom, o tome nema sumnje. Oni na to gledaju kao na nacionalni prioritet. Mi mislimo da će za četiri do pet godina Japanci, napokon, otkriti kako da naprave čestit računar. Ako želimo da zadržimo primat u ovoj industriji, imamo četiri godine da postanemo svjetski proizvođači.

• Kako planirate da to ostvarite?

— U vrijeme kad smo projektovali „mekintoša", mi smo, također, projektovali i mašinu koja pravi mašine. Potrošili smo 20 000 000 dolara da izgradimo najautomatiziraniju tvornicu industrije računara. Ali to nije dovoljno. Umjesto da amortizujemo našu tvornicu za sedam godina, što bi uradila većina kompanija, mi ćemo je amortizirati za dvije godine. Bacit ćemo je do konca 1985 i sagraditi ćemo drugu, a i nju ćemo otpisati za dvije godine i baciti je i tako ćemo za tri godine od danas raditi na trećoj automatiziranoj tvornici. To je jedini način da učimo dovoljno brzo.

• Razgovarajmo o softveru. Koje su revolucionarne promjene u razvoju softvera u posljednjih nekoliko godina?

— Sigurno da unošenje programskog jezika u mikroprocesorski čip predstavlja napredak. VisiCalc je bio napredak, jer je to bilo prvo stvarno korištenje računara za poslovne svrhe, gdje su poslovni ljudi vidjeli opipljive koristi. Prije toga ste trebali programirati svoje vlastite aplikacije, a broj ljudi koji je voljan da programira je mali — svega jedan posto. Onda je došao Lotus — kombinacija dobrog spreadsheet-a (matrične tabele) i grafičkog programa. Procesor teksta i baza podataka Lotus-a sigurno nisu ono najbolje što čovjek može kupiti. Stvarna vrijednost Lotus-a je kombinacija tabele i grafike u jednom programu i mogućnost brzog skakanja sa jednog na drugo. Slijedeći prodor se dešava sada, zahvaljujući „mekintošu", koji je donio tehnologiju „lize" uz razumnu cijenu. Tu ima, a biće još više, revolucionarnog softvera. Pravi prodori se mogu cijeniti nekoliko godina nakon što su se dogodili.

• U kom pravcu vi vidite razvoj računara i softvera u bliskoj budućnosti?

— Za sada mi prilično koristimo računare kao dobre sluge. Tražimo od njih da nam nešto urade — tražimo da urade neku operaciju kao što je obračun tabele (spreadsheet-a), tražimo da prihvate naše kucanje po tastaturi i od toga naprave pismo, i oni to rade prilično dobro. Dakle, vidjećete, sve više i više perfekcije u domenu — računar kao sluga. Ali, slijedeća stvar će biti računar kao vodič ili agent. A to znači da će sve više biti u stanju da anticipira ono što želimo. Bićemo u stanju da od računara tražimo da prati događaje umjesto nas i, kad se određeni uslov ispuni, da povuče okidač — da obavi neke akcije i da nas obavijesti o tome.

• Hoćemo li biti u stanju da sve to obavimo na mašinama koje danas imamo? Ili ćete nas opteretiti novim računarima?

— Sve? To bi bila opasna izjava. To ne znam. „Mekintoš" je, u svakom slučaju, dizajniran tako da su se ti koncepti imali u vidu.

• Govorili smo o tome šta vi vidite u bliskoj budućnosti, a šta je sa dalekom budućnosti? Ako počnete da zamišljate neke od načina na koje će računari izmjeniti naše živote, šta onda vidite?

„IBM je ušao veoma silovito i teg je prešao na njihovu stranu. Svima nama koji smo radili na 'mekintošu' postalo je jasno da će to rasturiti industriju"

— Kada sam se vratio iz Indije, pitao sam se šta je najvažnija stvar koju sam naučio? Mislim da je to da zapadno racionalno razmišljanje nije urođena ljudska osobina, nego da je to naučena sposobnost. Prije mi nije padalo na pamet da mi ne bi ni razmišljali na određeni način da nas neko nije naučio da tako razmišljamo. A ipak, to je tako. Očigledno, jedan od velikih izazova obrazovanja je da nas nauči kako da mislimo. Ono što otkrivamo je to da će računari uticati na kvalitet razmišljanja tim više što više naše djece bude imalo te sprave na raspolaganju. Ljudi su korisnici alata. Ono što je zaista nevjerovatno za knjige je to da možete čitati šta je Aristotel napisao. Ne treba vam profesorova interpretacija Aristotela. Naravno, možete i to dobiti, ali možete pročitati tačno ono što je Aristotel pisao. Taj direktni prenos misli i ideja je ključni kamen koji zgradu društva čini onakvom kakva ona jeste. Ali problem sa knjigom je taj da ne možete da Aristotelu postavite pitanje. Mislim da je potencijalna mogućnost računara da na neki način . . . . uhvati osnovni, bazni princip nekog iskustva.

Priredio: Domagaj Bačić

Periferijska oprema

svetlosna pera

Pokazivanje prstom predstavlja najjednostavniji način da nekome ko nas ne razume pokažemo ono što želimo. Za dobar broj korisnika računar će do kraja života ostati „crna kutija" s kojom će se sporazumevati isključivo preko tuđih programa. Pošto računari još uvek nisu dovoljno savršeni da bi zapazili i shvatili dodir nečijeg prsta, izmišljena je svetlosna olovka — neka vrsta „elektronskog prsta" — koja komunikaciju sa računarom pojednostavljuje do krajnjih mogućih granica. Svetlosna pera su, osim toga, i nezamenjiva alatka za ljubitelje crtanja.

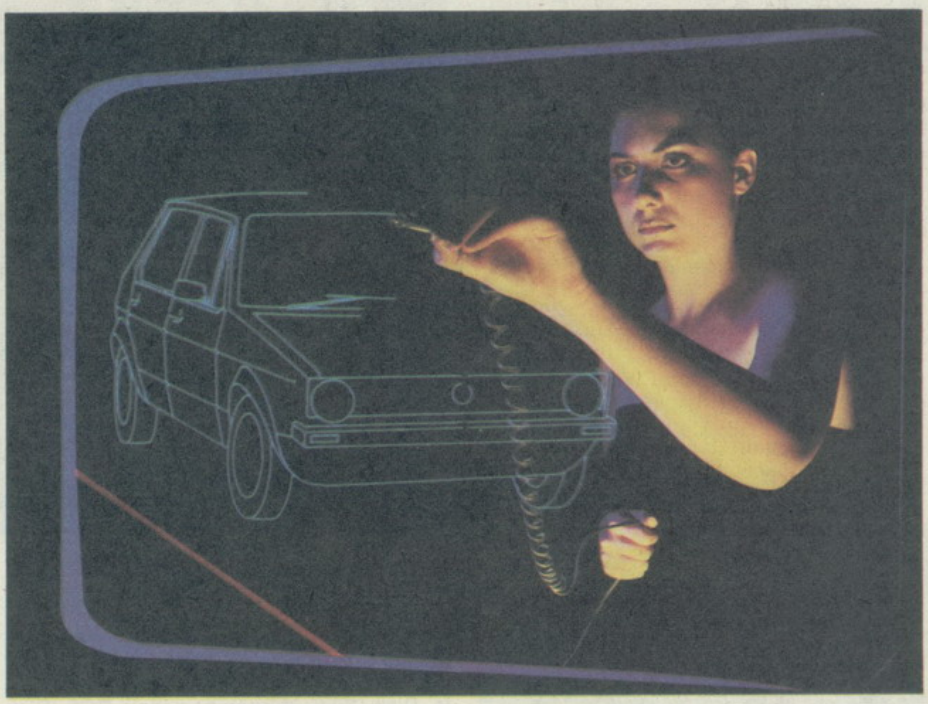
Prema rezultatima poslednje ankete američkog časopisa COMPUTE, samo 6 odsto ljubitelja računara poseduje svetlosno pero. To je nesrećna okolnost, jer ono pojednostavljuje korišćenje programa i najviše odgovara upravo korisnicima računara koje programiranje ne interesuje već koriste opcije iz gotovih programa. Svetlosno pero, osim toga, vezuje pažnju kod dece. Ako ste roditelj sa minimalnim ili osrednjim programerskim umećem, možete nabaviti gotove programe za obučavanje dece koji se koriste uz svetlosno pero. Za korišćenje ovakvih edukativnih programa deca ne moraju da znaju čak ni da čitaju, jer izbor iz menija koji je prikazan u sličicama vrše jednostavnim dodirivanjem željene sličice.

Zašto nam treba . . .

Svetlosno pero je nezamenjivo oruđe kada treba brzo locirati slučajnu poziciju na ekranu. To može biti unošenje koordinata, stavljanje oznaka, crtanje linija, izbor iz menija i slično. Upotreba kursora ili, još gore, unošenje koordinata sa tastature, sporo je i zamorno, a svetlosna pera su brza, jednostavna za upotrebu, vrlo precizna i posebno pogodna za grafiku. Njima možete crtati ili pokazati na ekranu odgovarajuću opciju iz grafičkog programa. Danas se mogu nabaviti ozbiljni aplikacioni programi — tekst procesori i programi za obradu tabela koji umesto ukucavanja instrukcija omogućavaju njihovo unošenje pokazivanjem svetlosnim perom. Jedni programi omogućavaju da se svetlosno pero koristi za kreiranje linija, krugova i kvadrata i popunjavanje tih geometrijskih slika raznim bojama, a drugi da svetlosno pero koristite za definisanje grafičkih simbola ili karaktera ili konstruisanja komplikovanih geometrijskih slika. Mada je za igre pogodniji džojstik, u zadnje vreme sve je više igara na tržištu koje se igraju uz pomoć svetlosnog pera — to su, pre svega, igre koje zahtevaju pomeranje objekata po ekranu, na primer, šah. Postoji i nov proizvod u domenu igara za svetlosno pero — svetlosna puška. Ona je već raspoloživa za „spektrum" i „komodor" računare.

. . . i kako radi

Svetlosno pero „čita" poziciju na ekranu monitora ili televizora. Pri dodiru ekrana, elektronika u peru i računar zajedno izračunavaju koordinate te tačke. To je moguće zbog načina na koji se kreira displej na katodnoj cevi (tipu koji je u televizorima i monitorima). Tanak snop svetlosti kreće se ekranom s leva na desno i odozgo na dole. Kod evropskih televizora on pređe preko 625 linija displeja 50 puta u sekundi. To „jurenje" svetlosti je prebrzo da bi ljudsko oko registrovalo pomeranje, pa imamo iluziju da gledamo stabilnu sliku. Svetlosna pera sadrže svetlosni senzor pri vrhu. Taj senzor, obično svetlosno osetljiv tranzistor ili dioda, registruje svetlosni trag u momentu prolaska, proizvodi elektronski signal, pojačava ga i šalje u računar. Pošto računar kontroliše displej na ekranu, on „zna" gde se u tom momentu svetlost nalazi, pa može da izračuna x i y koordinate tačne pozicije vrha svetlosnog pera.



Crtanje bez muke: Optičko pero značajno olakšava život ljubiteljima crtanja na računaru

Modeli svetlosnih pera se razlikuju po mnogim karakteristikama — po tipu svetlosnog senzora, položaju senzora, mestu na kome se nalazi pridružena elektronika, da li daju različite režime rada, da li i kako pokazuju kada su registrovali signal i, najzad, za koji tip računara su primenjivi. Za korisnika je, uz cenu, najznačajnije da zna osetljivost i rezoluciju svetlosnog pera.

Osetljivost . . .

Pod osetljivošću svetlosnog pera podrazumevamo intenzitet svetlosti ekrana koju pero može da registruje. Najbolja su ona koja mogu da registruju većinu, ako ne i sve boje sa dobro podešenog kolor televizora, dok svetlosna pera slabijeg kvaliteta registruju samo jaču svetlost. U dosta slučajeva je i to dovoljno, ali dovodi do ozbiljnih ograničenja u grafičkim aplikacijama kod kojih pozadina može biti i tamna.

Jedan od problema sa svetlosnim perima je neispravna detekcija. Do nje dolazi kada je osetljivost uvećana velikim pojačanjem signala sa svetlosnog senzora. Preterana osetljivost može dovesti do toga da i vanekranski izbori svetlosti, pa čak i dnevna svetlost, deluju na pero. Skuplja pera koriste razne metode osiguranja od tih izvora svetlosti, recimo kombinacije svetlosnih senzora i svetlosnih filtera. Mogu se upotrebiti i elektronski filteri. Svetlost sa

→

ekrana dolazi 50 puta u sekundi. Jednostavno elektronsko kolo propušta ka računaru samo svetlosti sa tom frekvencijom.

. . . i finoća crtanja

Rezolucija se odnosi na zonu ekrana koja se detektuje. Ona može da varira od jedne tačke (piksela) do grupe karaktera kod najlošijih pera. Rezolucija svetlosnog pera zavisi od tipa senzora, brzine reagovanja i metoda kolimacije (sakupljanja svetlosti). Visoka rezolucija je važnija za crtanje nego za registrovanje. Pero sa visokom rezolucijom može registrovati manju zonu, pa je, stoga, preciznije.

Razna pera imaju različitu dokumentaciju i softver koji se uz njih dobija. Da bi svetlosno pero uopšte radilo, potrebna mu je softverska podrška. Neki proizvođači obezbeđuju kasete sa demonstracionim programima, dok drugi daju samo listinge da bi ilustrovali način korišćenja svetlosnog pera.

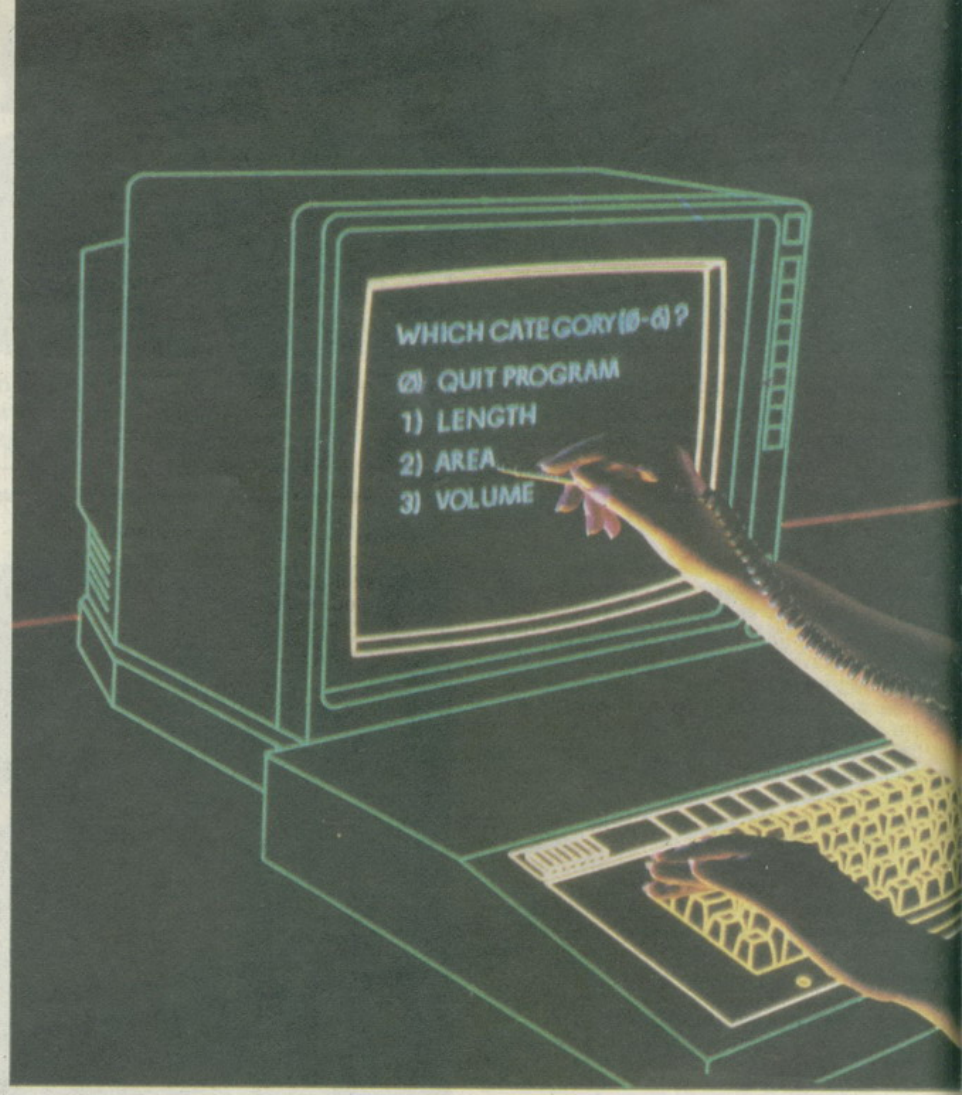
U „mekintoš" stilu . . .

Kao i kod drugih vrsta „specijalnog" hardvera, i svetlosna pera sa odgovarajućom programskom podrškom prvo su bila raspoloživa za skuplje i za nas nepristupačne računare, kao što su „epl II", IBM PC i PCjr., ali danas ima sve više tipova za „komodor 64" i druge kod nas popularne računare. Predstavićemo vam softverske pakete koji se mogu koristiti uz Edumate Light Pen i Gibson Light Pen.

Edumate Light Pen, koji nudi Futurehouse, ima sve neophodne veze i spreman je da se utakne u džojstik port „komodora". Tu su i brošura sa instrukcijama i softverski paket sa raznim programima za svetlosno pero. Softver sadrži program za crtanje sa visokom rezolucijom, disk utility, muzički program i igru 3D Tic Tac Toe. Rutina za crtanje omogućava da se crta u grafičkom režimu visoke rezolucije. Nudi se 16 boja sa opcijama brisanja ekrana ili uklanjanja linija. Disk Utility je opštenamenski DOS (Disk Operating System) program koji nudi opciju displeja direktorijuma i mogućnost biranja, punjenja i startovanja programa pomoću svetlosnog pera. Uz to, može se inicijalizovati, validizovati ili formatirati disketa, prikazati direktorijum i izbrisati bilo koja datoteka ili program.

Da vam život bude lakši

- *Svetlosna pera moraju biti savršeno čista. Svaka nečistoća utiče na svetlost koju pero treba da sakuplja. Postoji dosta četkica za čišćenje pera i ili ekrana baziranih na alkoholu.*
- *Pero zamračuje bar deo ekrana kada ga koristite. To je hendikep jedino ako program ima veoma mnogo detalja na ekranu.*
- *Pre kupovine se uverite da za vaše potrebe nije dovoljan džojstik. Ne zaboravite da za crtanje po vertikalnoj površini ekrana treba mnogo više truda nego po horizontalnoj površini hartije. Ako vam cena nije problem, izaberite grafičku tablu.*
- *Ako se vaše pero ponaša nepravilno ili je netačno, možda skuplja previše svetlosti iz sobe. Pokušajte da spustite zavese ili ugasite neku sijalicu.*
- *Možda treba da podesite osetljivost ili kontrast na televizoru da bi pero radilo korektno.*
- *Pri biranju opcije iz menija ne šetajte pero okolo, već ga usmerite pravo na željenu opciju — u suprotnom možete dobiti pogrešnu tačku i pogrešan meni.*
- *Neka novija pera otkrivaju koja je boja na ekranu. Ako imate takvo pero, proverite da li vaš softver „kaže" peru koju boju da registruje. Možda ćete morati i da podešavate boju na ekranu sve dok ne bude potpuno odgovarala peru.*



Pero umesto miša: optičkom olovkom može se najbrže doseći proizvoljno odabrana pozicija na ekranu

. . . i nešto malo siromašnije

Za spektrum ćemo predstaviti tri vrste svetlosnih pera, čije se cene kreću između 15 i 20 funti.

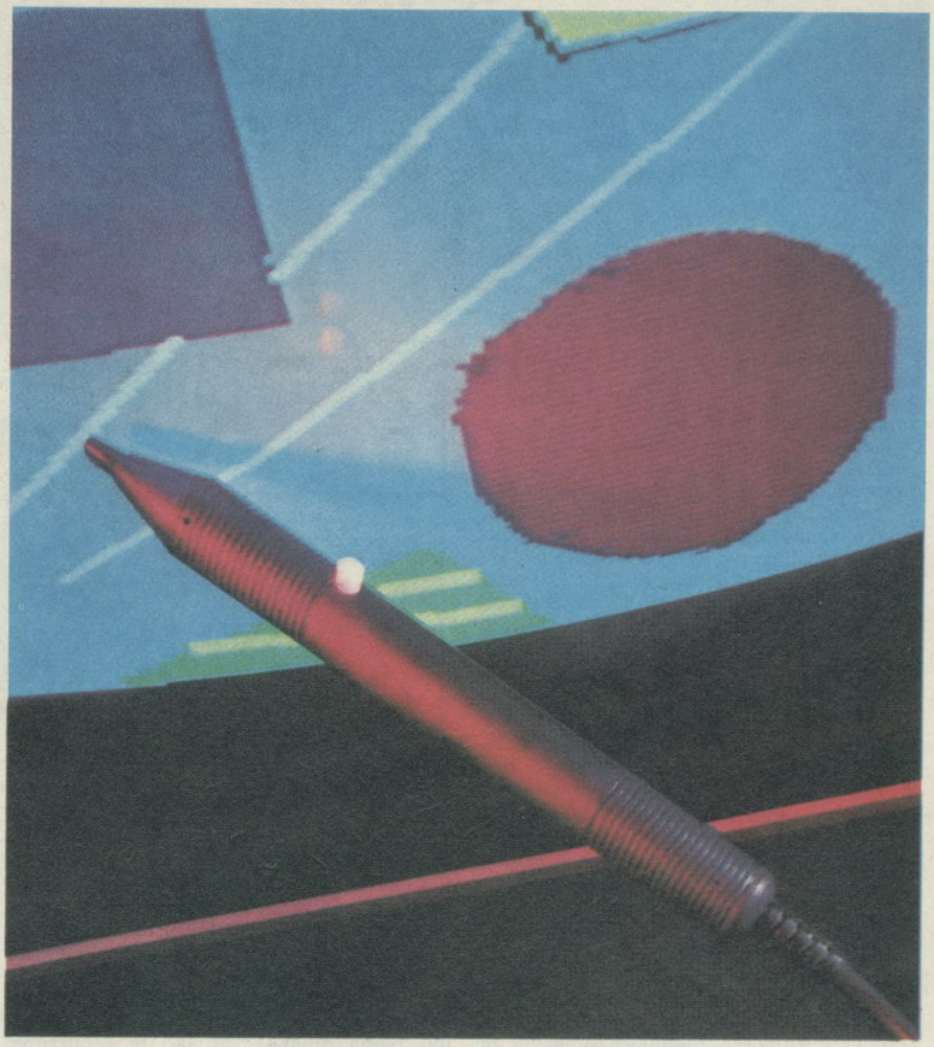
Svetlosno pero koje je na tržištu izbacio Custom Cabels International najjeftinije je i sa najmanjim mogućnostima, ali korektno radi sa svakom opcijom. Opcije uključuju crtanje svih geometrijskih oblika i, što nemaju ostali proizvodi, crtanje poligona, za koji sami birate broj strana. Ovo pero nema meni opciju ni mogućnost slobodnog crtanja, ali, ako vam ove dve stvari nisu bitne, bićete sasvim zadovoljni njegovim kvalitetom.

Svetlosno pero TROJAN nema najbolju softversku podršku. Kalibraciona rutina koja proverava da li se slaže ono što vidite sa onim što se šalje u računar je neprecizna i čini se, bar u demonstraciji menija, da se pozicija pera čita prebrzo. Rutina slobodnog crtanja radi tako sporo da je praktično neupotrebljiva. Možete crtati krugove, linije i ostale programirane figure, ali je pero suviše neprecizno da bi to učinilo lakim. Kad se sve sabere, više se isplati kupiti jeftinije pero čije sve opcije korektno rade.

DK' troničko MK.4 pero je sasvim druga (najskuplja) priča. Broj nije bez značaja, MK.3 verzija je samo neznatno bolja od TROJAN SP, ali MK.4 dozvoljava razumno crtanje slobodnom rukom. Jedino je problem mirno držati ruku na gotovo vertikalnoj površini ekrana. Osim kreiranja linija pravougaonika i krugova, možete sačuvati do pet ekrana i staviti ih u rotacije od veoma brzih (pseudoanimacije) do izuzetno sporih. Rutina slobodnog crtanja zahteva malo napora, ali on se isplati.

Pen Music omogućava sviranje dodirom svetlosnog pera po raznim tačkama ekrana. Prikazuje se zona od 12 nota — kompletna muzička skala u bilo kojoj od 8 oktava.

Mada su svi programi upotrebljivi, moramo napomenuti da imaju i nekoliko ograničenja. Korišćenjem Disk Utility mogu se puniti samo bejzik programi. Opcija formatizovanja diskete ne omogućava izbor imena diskete. Muzički program ne omogućava biranje filtera i jačine zvuka. Crteže kreirane ovom rutinom za crtanje ne možete da sačuvate na traci ili disku . . . i da ne nabrajamo dalje — i najboljim paketima mogu se naći mane.



Optika u akciji: Čak i kod najskupljih modela optičkih olovaka crteži su još uvek daleko od savršenstva

Jedan od vodećih dizajnera mikroračunarskog grafičkog softvera, Stiven Gibson, pripremio je paket Gibson Light Pen za „komodor" računare. U ovom paketu nalaze se programi Pen Painter, Pen Animator i Pen Musician. Prvi od njih je kolor grafički program koji omogućava slobodno crtanje, crtanje pravilnih geometrijskih likova i daje obrasce za punjenje bojom. Ovaj softver omogućava da se izbor vrši iz menija u „mekintoš" stilu — biranjem sličica. Drugi program predstavlja uvod u animiranu kompjutersku grafiku i omogućava da se kreira do 20 delova animacije koji mogu da jure po ekranu dovoljno brzo da bi se stvorio utisak kretanja. Treći program uvodi u muzičku kompoziciju i dozvoljava da postavite, editujete i izvodite note na muzičkoj skali pomoću svetlosnog pera.

Izbor — uvek vaš

Ima više faktora koje treba imati u vidu pri izboru svetlosnog pera. Mada neka svetlosna pera teoretski imaju visoku rezoluciju, u praksi nisu dovoljno precizna. Neka su, čak, toliko gruba da im se linije razbijaju u mnoštvo piksela po ekranu. Princip je i ovde — dobijaš koliko platiš. Cene variraju od cene za tipično memorijsko proširenje do cene jeftinijih računara. Naravno, kao i kod ostalih prizvoda (ne kod nas), cene stalno padaju.

Neka, obično skuplja, pera imaju još dve korisne osobine — proizvode signal obaveštenja da je pero registrovalo svetlost i signal da je svetlost predata u računar. Svetlosna pera obično koriste „svetlosno emitujuću diodu" za obaveštenja da je registrovana ispravna svetlost. To nije od značaja ako nam pero treba samo za crtanje, ali pomaže za čitanje pozicija sa ekrana ili izlaz iz menija.

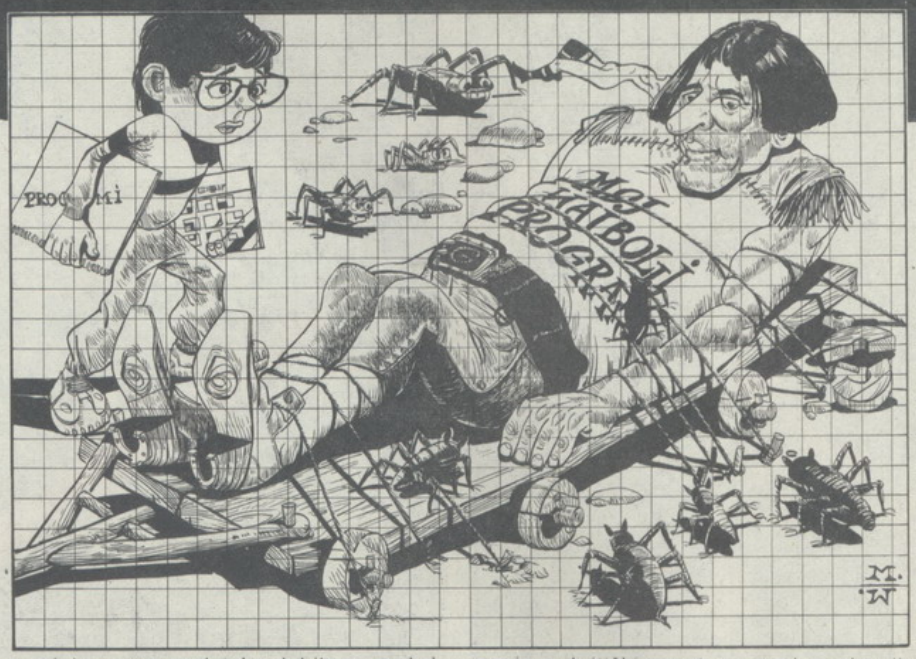
Svetlosna pera često imaju prekidač koji obezbeđuje da se informacije šalju u računar samo kada želimo, što je korisno jer smanjuje mogućnost slučajnog ili pogrešnog čitanja (pogrešan izlaz iz menija npr.). Sem toga, imamo više kontrole nad čitanjima koje vrši pero. Postoje razni tipovi prekidača. Neka od boljih pera imaju minijaturne prekidače pri vrhu koja reaguju tek ako pero dodirne ekran. Međutim, bez obzira šta se piše o mogućnostima pera, preporučujemo da pri kupovini obavezno isprobate svetlosno pero i proverite da li radi onako kako želite.

Najvažnije je da proverite da li svetlosno pero radi s vašim računarom. „Spektrum" i ZX 81 sigurno zahtevaju interfejs koji se, po pravilu ugrađuje i u sama pera. Ne zaboravite ni softver. Ako kupite svetlosno pero koje nije dizajnirano za vaš računar, nije isključeno da ćete morati sami da ga pišete ili kupujete novi softver da bi uopšte proradilo.

S obzirom na veliku primenljivost za igre i ozbiljne aplikacije, brzinu i jednostavnost upotrebe i nisku cenu, popularnost svetlosnih pera raste iz dana u dan. Cene će i dalje padati i biće više raspoloživog softvera. Za korisnike koji žele da unaprede igranje kompjuterskih igara, ali i one koji hoće da udobnije koriste ozbiljne programe, svetlosno pero je idealan dodatak računaru.

Vladimir Ilijevski
Nevenka Spalević

haker Umetnost programiranja
na usijanom limenom krovu



Osnovni problem u testiranju je pogrešna definicija, koja se obično izražava na jedan od sledećih načina: „Testiranje je proces demonstriranja da nema grešaka u programu", „Smisao testiranja je da se pokaže da program radi korektno", ili: „Testiranje je proces sticanja uverenja da program radi ono što treba da radi". Sve ove definicije imaju jednu jedinu manu: totalno su pogrešne! Pozabavimo se analognom situacijom u kojoj bolestan pacijent dolazi na laboratorijsku pretragu. Ako analize ne otkriju poreklo problema, sigurno nećemo reći da je postupak bio uspešan: vreme je izgubljeno, pacijent je i dalje bolestan, a, uz to, polako počinje da sumnja u kompetentnost dijagnostičara. Obratno, ako pretrage pronađu čir na želucu, može se pristupiti lečenju. (Mogućno je, dakako, da pacijent boluje i od drugih bolesti.)

Ova analogija nas navodi na ispravnu definiciju testiranja, koja glasi: „Testiranje je proces izvršavanja programa sa ciljem da se pronađe greška!", pri čemu se podrazumeva da je program „bolestan", tj. da sadrži greške. Sa izuzetkom retkih hardverskih grešaka, sve ostale greške u programu prave ljudi — programeri, pa uspešno testiranje zavisi od prilaza problemu. Ako je cilj pokazati da u programu nema grešaka, onda ćete verovatno uspeti da pokažete da ih ima malo; ako je, suprotno tome, cilj pokazati da su greške prisutne, onda ćete verovatno naći njihov dobar deo! Na žalost, kod iole većih programa nikada nećemo biti sigurni da smo pronašli sve postojeće greške.

Crne i bele kutije

Pristup crne kutije: „Baš me briga kako je program napisan; propustiću sve mogućne ulazne podatke, i ako program dobro radi — sve je u redu!". Pristup bele kutije je totalno suprotan: „Napraviću taman toliko testova da se svaka grana programa izvrši barem jednom. Najvažnije je proveriti logiku programa!" Nijedan od ova dva pristupa nije mogućno striktno provesti u praksi. Ulaznih podataka ima u svakom, ma kako trivijalnom programu, praktično beskonačno. Kako, recimo, testirati jednu akcionu igru u kojoj se sve dešava u realnom vremenu? U njoj bi trebalo testirati ne samo sve mogućne kombinacije ulaznih podataka nego i sve mogućne situacije u toku igre — a broj takvih situacija je astronomski.

Jasno je da sve ulazne podatke nikad ne možemo proveriti u praksi — ekonomičnost je osnovni problem u procesu testiranja programa. Svaki test treba da predstavlja čitavu klasu ulaznih podataka, tako da sa minimumom podataka dobijemo maksimum informacija o ponašanju programa. Ni pristup bele kutije nije ništa bolji u svom ekstremnom vidu. Broj mogućih puteva u iole složenijem programu raste sa faktorijelom, to jest, strašno brzo. Štaviše, i kada bi smo bili u stanju da izvedemo potrebne bilione testova u neko dogledno vreme, čak ni onda ne bismo mogli tvrditi da u programu nema greške. Šta ako je programer napisao program za računanje kvadratnog korena — a tražio se program za kubni koren? A tu je i problem osetljivosti podataka: neke grane programa izvršavaju se za datu tačnost ulaznih podataka, ali ako podaci izađu iz predviđenih granica — onda se izvršavaju sasvim drugi delovi programa. Najkraće rečeno, testiranje je praktično nerešiv problem: ma kako se trudili u procesu pisanja programa, greške će biti prisutne; a kada su već jednom tu — možemo ih testiranjem uklanjati, ali nikad nećemo biti sigurni da su zaista sve uništene. Zato je važno da se pridržavamo sledećih fundamentalnih principa testiranja.

Principi . . .

• Dobar je onaj test za koji je vrlo verovatno da će otkriti grešku u programu, dok je test koji demonstrira pravilan rad programa neuspešan. Cela aktivnost testiranja je usmerena na poboljšanje kvaliteta programa, tj. na poboljšanje pouzdanosti programa u toku eksploatacije. Povećanje pouzdanosti postiže se nalaženjem i uklanjanjem grešaka iz programa. Stoga i ne treba praviti test-primere koji pokazuju da program dobro radi, već suprotno, da program radi pogrešno!

• Programer koji je pisao program ne bi trebalo i da ga testira! Projektovanje i pisanje programa je vrlo kreativna ljudska delatnost, a testiranje je rušilački proces, u kome sve napisano treba podvrći sumnji i kritici. (Da li ste ikada čuli za pesnika koji je napisao kritički osvrt na svoju najnoviju pesmu?) Ako je programiranje umetnost pisanja eseja u izvršnom obliku, onda je testiranje pisanje kritičkog prikaza — a kritike se objavljuju dva-tri dana posle premijere, kada se duhovi smire. Ako želite zaista kvalitetan program, zamolite svog prijatelja da demolira vaše potencijalno remek-delo. Ako ipak hoćete (ili morate) sami da testirate svoj program, nemojte to raditi istog dana kada ste napisali i ukucali program u računar. Namerno se odvojite od mašine na dva-tri dana, ohladite se, naljutite se na svoj program i tek onda počnite sa testiranjem.

• Svaki test mora sadržati i opis očekivanih izlaznih podataka ili rezultata. Jedna od najčešćih grešaka u testiranju upravo potiče od ignorisanja ovog pravila. Očekivane rezultate moramo izračunati ranije, inače dovodimo sebe u situaciju da vidimo ono što hoćemo da vidimo, a ne ono što jeste. Izvršavanje programa bez unapred poznatih rezultata nije testiranje već — eksperimentisanje.

• Kućni i lični računari, baš kao i rad sa terminalima u sistemu tajm-šeringa, su kontraproduktivni testiranju. Isuviše je lako upasti u začarani krug u kojem vi samo sedite i pritiskate delove tastature, izvršavajući program opet i opet, a sve u strahu da „računar ne stoji". Nikad ne stižete da napravite valjani test-primer, a najveći problem je što se testiranje obavlja u letu: testovi iščezavaju bez traga, pa posle svake nove izmene u programu treba ponovo izmišljati već stvorene no izgubljene testove . . .

• Testove treba dokumentovati i čuvati u obliku datoteke na spoljnoj memoriji. Time štedimo trud na kreiranju testova, i istovremeno osiguravamo da program posle svake izmene bude testiran na isti način.

Faza testiranja zauzima više od 50 odsto vremena u svakom programerskom projektu. Godišnji obrt računarske industrije meri se desetinama milijardi dolara, pa bi se moglo pomisliti da je testiranje dobro izučena aktivnost — ali nije tako. Problem je kako u samoj prirodi procesa testiranja, tako i u nedovoljnoj obučenosti programera za ovaj deo svog posla. Većina testova sročena je tako da po svaku cenu dokaže da program radi, dakle po principu „hajde da se pravimo blesavi" — „maločas je odlično radio, ne znam šta mu sad bi". Cilj svakog pravog testa je, međutim, da pokaže da program ne radi kako treba.

• Testirajte program kako pomoću tačnih, tako i pomoću netačnih i neprihvatljivih ulaznih podataka. Ovo je najbolji način da zaštitite program od naivnih korisnika, pa i od samog sebe: kroz šest meseci i vi ćete biti samo korisnik sopstvenog programa!

• Detaljno izučite sve rezultate svakog testa. I najbolji testovi ne vrede ako ne interpretirate njihove rezultate. Ovo pravilo je toliko očigledno da ga ne bi trebalo posebno izdvajati — samo da se tako često ne prenebregava u praksi!

• Nikad ne menjajte program da biste sebi olakšali testiranje! Recimo da neki ciklus od 100 prolaza smanjite na 10 — da bi ste brže dobili rezultate testa. U tom slučaju zaista vršite testiranje — ali ne svog, nego nekog drugog programa!

• Nije dovoljno uveriti se da program radi sve što treba da radi; treba se uveriti i da program ne radi nešto što ne bi trebalo da radi! Primer takvog neželjenog bočnog efekta je procesor reči koji od vas obavezno zahteva da tekst snimite na traku pre štampanja.

• Greške nailaze u grupama! Ovaj fenomen se stalno iznova pojavljuje u praksi, mada još niko nije dao zadovoljavajuće objašnjenje zbog čega je to tako. Jednostavno, neki delovi programa podložniji su greškama od ostalih. Poznavanje ovog principa daje nam važno praktično uputstvo za testiranje: kad naiđete na jednu grešku, potražite u njenoj okolini bar još jednu ili više grešaka.

. . . i vrste testiranja

• Testiranje modula. Modul je nezavisna celina unutar programa, sa jasno određenim ulaznim i izlaznim veličinama. Testiranje modula počinje upravo proučavanjem ulazno/izlaznih podataka, a nastavlja se generisanjem testova samo na osnovu specifikacija modula. Zatim proveravamo tekst modula i ubeđujemo se bez efektivnog izvršavanja programa da odabrani testovi pokrivaju sve interesantne grane modula. Na primer, za svaki ciklus mora postojati bar po jedan test u kojem se ciklus izvršava maksimalan broj puta. Na kraju, proveravamo da li postoje putevi u modulu čije izvršenje zavisi od posebnih vrednosti ulaznih podataka, i dodajemo nove testove u tom smislu ako je potrebno.

• Funkcionalno testiranje je proces nalaženja razlika između programa i njegove specifikacije, tj. onog kako je program zamišljen da radi. Najbolje je pustiti krajnje korisnike da isprobavaju program neko vreme: oni koriste program na krajnje originalne načine, njih ne interesuje kako je program napisan — ako radi bez problema.

• Sistemsko testiranje ima za cilj da uporedi program (ili sistem programa) sa početnim ciljem pisanja programa. Sistemsko testiranje programa pokušava da prikaže kako sve program ne postiže svoje zamišljene ciljeve. Jasno, sistemsko testiranje je nemoguće ako projektant sistema nije zapisao prvobitne ciljeve pravljenja programa. Najprostiji i najsigurniji postupak za sistemsko testiranje je poređenje osobina programa sa priručnikom za korisnika; često je mogućno i bez izvršavanja programa uočiti neslaganja u ovom domenu.

• Količinski test ispituje kako program radi sa maksimalnim mogućim brojem podataka, tj. cilj ove vrste testiranja je da pokaže da program ne može da se izbori sa prevelikim količinama podataka. Na primer, šta se dešava sa procesorom reči kad ispunimo celu memoriju tekstom, ili kad u nekoj avanturi posetimo sve lokacije?

• Test načina korišćenja pokušava da otkrije greške u komunikaciji program/čovek. Na primer, da li su poruke programa prilagođene inteligenciji, uzrastu i obrazovanju krajnjeg korisnika? Da li korisnik mora stalno da prelazi iz malih u velika slova? Da li postoji previše opcija, ili naprotiv, da li postoje opcije koje se ne koriste? Da li program javlja greške na razumljiv način? Itd.

• Test konfiguracije. Da li vaš program radi i sa džojstikom? Da li procesor reči radi sa raznim vrstama štampača, raznim prečnicima diskova, raznim disk-formatima? itd.

• Testiranje oporavljanja sistema. Da li procesor reči briše sav tekst ako je taster Reset bio pritisnut? Da li računar briše celu memoriju posle pritiska na Reset?

• Testiranje dokumentacije. Da li program radi sve što piše u priručniku? Da li priručnik opisuje kako se koriste sve mogućnosti softvera i/ili hardvera?

• Test instaliranja sistema. Mnogi komercijalni programi se prilagođavaju svojoj okolini, što je uslov da se prodaju na različitim računarima. Cilj ovog testa je da otkrije na koje se sve sisteme dati program ne može instalirati.

Proverama nikad kraja

Na ovo pitanje teško je dati precizan odgovor. Videli smo da je nemogućno istestirati program na silu, tj. efektivnim izvršavanjem programa za moguće ulazne podatke ili sve mogućne grane unutar programa. Ma kako se trudili, senka sumnje uvek ostaje, jer nema načina da se dokaže da je greška koju smo tek otkrili baš poslednja u celom programu. Programeri obično prekidaju testiranje kada program izvrši sve test-primere bez greške. Ovakav pristup potiče od pogrešnih definicija testiranja, ali na sreću, postoje bolji (mada ne i savršeni) kriterijumi.

Jedan takav kriterijum može da bude sledeće: testiranje modula je završeno kada su neuspešni svi test-primeri koji su nastali pokrivanjem logičkih uslova u modulu, kao i analizom graničnih vrednosti ulazno/izlaznih podataka. Ovakav kriterijum je daleko od savršenog, jer zavisi od subjektivne procene i iskustva osobe koja vrši testiranje. Najveći nedostatak je tu ipak odsustvo merljivog cilja. Kako je cilj testiranja nalaženje grešaka, logično je uzeti broj pronađenih grešaka u datom vremenu kao „izlazni" kriterijum iz procesa testiranja. Na primer, test modula nije završen sve dok ne pronađemo 3 greške ili dok ne prođe dva dana — koje god od ta dva se dogodi kasnije.

Sad je logično postaviti sledeća pitanja: (1) Kako proceniti ukupan broj grešaka u programu? (2) Koji deo postojećih grešaka se realno može otkriti kroz testiranje? (3) Koliko grešaka je napravljeno u fazi projektovanja programa, a koliko grešaka otpada na kodiranje? Na prvo pitanje odgovara nam uglavnom iskustvo stečeno kroz protekla testiranja. Uzima se da je „industrijski" prosek 4—8 grešaka na 100 linija programa. Odgovor na pitanje (2) je ponešto proizvoljan i zavisi od prirode programa kojeg treba testirati. Odgovor na pitanje (3) je, ipak, najteži. Može se uzeti da u velikim programima samo oko 40% grešaka nastaju u procesu kodiranja, a sve ostale nastaju u ranijoj fazi projektovanja. Gotovih formula za kraj testiranja nema, pa je najbolje postati umetnik u pravljenju kompromisa: treba koristiti prvo kriterijum završetka testa modula, a zatim neku razumnu kobinaciju gornja tri kriterijuma.

Sve u svemu, testiranje je izuzetno kreativan i intelektualno izazovan zadatak. Da li to znači da samo geniji mogu da testiraju programe na kvalitetan način? — Sigurno ne! — jer ipak postoje metodi za generisanje testova. U okviru pristupa crne kutije to su: razbijanje po klasama ekvivalencije, analiza graničnih uslova, graf uzroka i posledice, kao i prosto pogađanje teških test-primera. U okviru pristupa bele kutije razlikujemo metode: izvršenje svake naredbe, izvršenje svake odluke, pokrivanje svakog uslova, pokrivanje svake odluke i uslova, i na kraju, višestruko pokrivanje uslova. U praktičnoj primeni trebalo bi koristiti skoro sve ili baš sve spomenute metode generisanja test-primera, jer nijedan od njih sam za sebe ne obezbeđuje potpun uspeh. Idealno bi bilo prvo napraviti testove po metodama pristupa crne kutije, a zatim dodavati testove koristeći metode bele kutije. Ali, o tome u sledećem broju!

Duško Savić

Smešna strana računara

računarenje i računsko razgovaranje

Očito je da su računari ušli u svakodnevni život — ako ne kao predmet ono bar kao pojam. Ali to više nije ona reč koja je označavala misterioznu i opskurnu mašinu za koju se samo znalo da čini čuda. Računari su danas čvrsto ugrađeni u trend i kao takvi predstavljaju obaveznu lektiru za sve one koji pretenduju da budu „u toku". Kako steći računarski manirizam i u svakom razgovoru zvučati kao da si proveo pet godina na Masačusetsu specijalizujući Tjuringove nauke? Evo ovako.

Najomiljenija tema računarskih priučenih erudita su, naravno, tipovi računara. Svaki od modela, tipova ili vrsta ima obavezne prefikse i sufikse, zavisno od toga da li je čovek koji govori navijač ili protivnik tog računara. „Sinkler ZX 81" je, tako, „smešan", „igračka" i „minijatura". To bi još i imalo smisla da te reči masovno ne koriste ljudi koji ni jednom računaru, pa ni „80-kecu", nisu prišli bliže od dva metra. O „spektrumu" i „komodoru 64" se „sve zna". „Spektrum" je isto „igračka" ali i „početnički idealan", „simpatičan", „jeftin" i „dobar za vežbu". On ima „nedovoljan ROM" i „odvratnu tastaturu", a predstavlja „smrt za mašinac".

Comodore ima dosta čvrstu poziciju u Jugoslaviji, pa nije ni čudo što se o njemu dosta priča. „Pet" je „davno prošla stvar", „Vic-20" je „zaboravljen", ali je zato „šezdesetčetvroka" „hit". „Komodor 64" je „izvanredan za te pare", „ozbiljniji" (verovatno od spektruma), „spor" u radu sa periferijskom opremom i „netačan" u računu. On još ima „solidnu tastaturu" i „odvratan" dizajn. O novim „komodorima" se manje priča jer još nisu doprli do Jugoslavije ali se već zna da su „moćni", „skupi" i „zanimljivi", otprilike ono što se zna za svaku novu seriju bilo kog proizvođača.

I ostali računari nisu zaboravljeni. O „BBC-ju" se neko vreme dosta pisalo, ali je to zamuklo. Ostala je samo priča da je „BBC B" „jako dobar". „Epl II" je „legenda" iako je „prevaziđen". „Mekintoš" je „super stvar" i predmet snova tipa „. . . ja bih voleo Meka . . .". „IBM PC" je „ozbiljna mašina" i „klasa za sebe". O Atariju se skoro nije ni pričalo ali otkad je Atari „kidnapovao" Džeka Tramijela i njegovu svitu, na našem tržištu, koje nije videlo više od tri „atarija", smatra se da je nova serija Atarija „zanimljiva" i sasvim „nov kvalitet za Atari".

O domaćim računarima se uopšte i ne priča, jer čak i oni koji u računar ne umeju da unesu ni igru sa snobovskim prezirom smatraju da su to kompjuteri za „prve korake u bejziku", nešto kao priručni materijal za bejzik.

Dosta je česta pojava privrženosti i računaru koji niko nema. Zašto da ne? Tako je „dobar ali prosečan" ali i „jeftin" „amstrad" postao maltene legenda pre nego što je bio i stigao do Jugoslavije. Neki zaljubljenici u „Hewllet Packard" i dalje tvrde da njegovi računari imaju „izvesne dodatne kvalitete", malobrojni vlasnici TRS 80 tvrde da je on „ipak klasika", poklonici „elektrona" tvrde da je on „nepravedno zanemarivan", i tako dalje i tome slično.

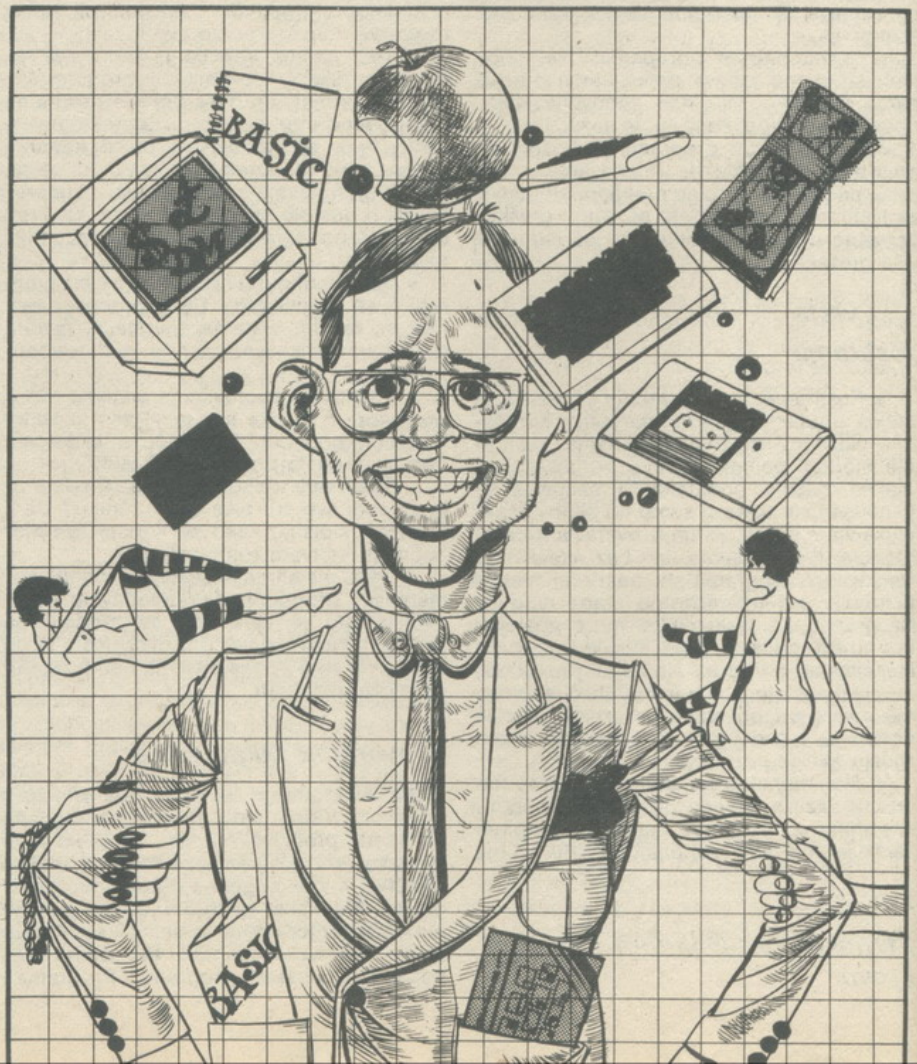
Najzahvalniji za blic ocene i eseje iz malog mozga su veoma poznati a veoma nedostupni računari koje ćemo teško videti negde u Jugoslaviji. Na njihov račun se može reći bilo šta, a što je najlepše, svi prisutni hakeri i parahakeri će se odmah složiti sa izrečenim, jer ni oni ne znaju o čemu se radi, a ne smeju dopustiti da izostanu iz uskog kruga posvećenih koji su „upućeni".

Nisu samo računari predmet tako opširnog „poznavanja". I sve ostalo vezano za računare podleže istom procesu.

Bejzik je „spor" i „primitivan" ali i „jednostavan" i „lak". Paskal je „kvalitetan" i „funkcionalan", a logo je ona stvar koju „treba više primenjivati", mada malo ko zna kako on uopšte i izgleda. I, najzad, programiranje na mašincu je „prava stvar" ali i „komplikovana stvar".

ROM je skoro uvek „skučen", „nedovoljan" i „loše organizovan". Mikrodrajv je „spor", „nepraktičan" i „promašen". Diskete su „praktične" i „pouzdane", dok je disk drajva „najbolje imati dva". Zašto? Za „ozbiljnu primenu".

Matrični štampač je „praktičan", štampač sa lepezom je „bučan" ali ima „kvalitetan otisak" dok je injekt štampač „još uvek skup".



Ilustracija: Miša Marković

Specijalni monitor, monohromatski ili ne, je za „određene" primene „neophodan". LCD je „nečitljiv" iako „praktičan" (!?). Grafika visoke rezolucije je „obavezna".

Video igre su svima „dosadne", „prevaziđene", „suviše jednostavne" ali i „suviše komplikovane", sa „kvalitetnom grafikom", „lošom animacijom" i „izvanrednim efektima". Sve je više tvrdnji da su video igre „truba" i „glupost", ali ih igra isti toliki broj ljudi kao i pre.

Obavezna lektira za „znalce" su raznorazni časopisi koji pišu o računarima. Jedna od poslastica koje se u njima mogu naći su tabele sa brzinama računara. Na osnovu tih tabela se lako tvrdi da je neki računar „spor" ili „brz", iako bi trebalo reći da bi računari za primene koje ih snalaze kod dobrog broja jugovlasnika mogli slobodno da budu i desetak puta sporiji nego što jesu.

Grafičke table, ploteri, svetlosna pera i slične „stvarčice" se ne pominju previše, osim u snobovskim krugovima onih koji smatraju da činjenica što imaju dovoljno para za takve delikatese stoji kao razlog da konverziraju samo o takvim stvarima.

Posebna stvar su priče o računarskim časopisima. Postoje računarski časopisi kojima fali samo još jedan korak do časopisa za domaćicu i njenu porodicu. Takvim časopisima se smeju i njihovi sopstveni saradnici, pa nije ni čudo da se za njih priča da su „jednostavni" i „razumljivi". Drugi tip časopisa je suviše zanet tehnikom računarskog bitisanja, pa tako naočigled svojih čitalaca evoluira u tehničku enciklopediju. Za takve se kaže da su „suviše kvalitetni" (!?). Treća vrsta su oni časopisi koji dobrano filuju svoje strane tekstovima preuzetim iz mora svetskih računarskih časopisa. Oni važe za „prevodilačke" časopise. Za jedne se priča da su opasni, a za druge da „postaju sve bolji". Lako je primetiti da sam izostavio časopise za koje bi neko mogao da smatra da nešto i vrede. To je normalno: o takvim časopisima se malo priča — oni se kupuju.

Postoji čitava jedna oblast nebuloznih preganjanja i rasprava koja je vezana za tako „apstraktne" stvari kao što su operativni sistemi, čipovi, kompatibilnost svega sa svačim i slično. Rasprave o takvim temama imaju tendenciju da budu tako zamršene da polako teže filozofsko-religioznim diskusijama, punim reči kao što su „modulitet", „apsolutno", „apriorno", „sublimacija", „rekognitivno" i sličnih.

Na svu sreću, to je više zabavno nego štetno. Tako će i biti sve dok bude više onih koji su čuli za računar nego onih koji o računaru nešto i znaju.

Moglo bi lako da se dogodi da računari uskoro iskliznu iz neke trenutne „mode" pa da čujemo o njima da su „dosadni", „šta je to?", „stara fora" i obična — „glupost"!

Branko Đaković

Psiho test

pirati i obožavaoci

Po ugledu na nagradne igre, testove i kvizove sa dидaktički opravdanim pitanjima i poučnim odgovorima, rešili smo da uradimo i više od toga: ovo nije samo test kojim ćete proveriti ispravnost svojih stavova, već i prilika da se u tom smislu korigujete i nadogradite. Sve je zasnovano na strogo naučnim principima i sve što budete izjavili moći će da se iskoristi protiv Vas. Osim toga, možete da učestvujete u nagradnoj i kaznenoj igri.

Pažljivo pročitajte pitanja i zaokružite samo jedan od predloženih odgovora. Strogo je zagranjeno korišćenje literature (na primer, ranijih brojeva „Računara"), kao i drugih nedozvoljenih sredstava. Na raspolaganju imate 45 minuta.

1. Zašto se domaći kućni računari zovu mišolovke?

a) Sa indignacijom odbijam insinuaciju.

b) Ne znam, tako skupe mišolovke još nisam video.

c) Zato što umesto procesora imaju miša.

d) Zato što predstavljaju stupicu za naivne.

2. Da li se pomoću domaćeg kućnog računara mogu loviti miševi?

a) Da, mi sve možemo.

b) Moraju se jasno precizirati ciljevi i zadaci subjektivnih snaga, pa i šire.

c) Ne isplati se.

d) Ne, ni miševi više nisu tako naivni.

3. Yugo-računare treba izvoziti Amerikancima:

a) Bio bi to posao stoleća.

b) Ima da izvozimo, pa šta košta da košta.

c) Neka ljudi vide šta mi imamo.

d) Neka se i oni malo nasmeju.

4. Ako Vam neko traži neki ozbiljniji program, recimo tekst procesor, za džabe, šta ćete uraditi?

a) Pošto ne možete tek tako da ga odbijete, reći ćete da program nije predviđen za kasetofon nego samo za disk.

b) Uvalićete mu program sa greškom da on ne bi mogao da koristi program koji vi ne znate da koristite.

c) Odbićete ga ladno zato što ne možete da podnesete da još neko ima isto što i Vi.

d) Daćete mu traženi program i sve ostale koje imate.

5. Imate priliku da presnimite program. Vi ćete:

a) Tražiti pismeno odobrenje od stranog proizvođača.

b) Sa gnušanjem ćete odbiti i pomisao na tako nešto, ali ćete program ipak presnimiti da zlobnici ne bi mogli da kažu da Vi to ne radite zato što ne umete.

c) Zažmuricete i uraditi to u mraku da niko, pa ni Vi, ne vidi šta ste to uradili.

d) Pitaćete da li ima još.

6. Ako Vam neko ponudi da Vi sebi presnimite program koji on ima, Vi ćete:

a) Privesti ga u najbližu stanicu milicije, a program ćete da prekopirate da biste imali dokazni materijal.

b) Išamaraćete ga i oduzeti mu sve programe zato što pokušava da Vas navede na piratiju, a i pokušaj se kažnjava.

c) Objasnićete mu da Vi sebi ne možete dozvoliti takvu niskost, nego da Vam on to presnimi.

d) Ponudićete mu svoje programe.

7. Kako ćete objasniti ako Vas Komisija za ispitivanje porekla imovine pita odakle Vam toliki programi?

a) Programe ste kupili od stranog proizvođača za devize koje ste pozajmili od MMF; deca će da vrate.

b) Reći ćete da ste programe pozajmili i da ćete ih uredno vratiti.

c) Vi ih samo čuvate da ih neko drugi ne presnimi.

d) Pitaćete Komisiju da li njima treba neki program.

8. Koji je najprikladniji jezik za školske računare?

a) Pseudojezik koji će predavati pseudoprofesori.

b) Trebalo bi ravnopravno upotrebljavati sve jezike naroda i narodnosti.

c) Starogrčki.

d) Šatrovački.

9. Zašto ženama treba zabraniti pristup računarima, a naročito računarskim časopisima?

a) Zato što zagađuju okolinu.

b) Zato što to nije uobičajeno među narodima jugoseverne Afrike.

c) Zato što se, po pravilu, Marica ne razume u kriv džojstik.

d) Zato što je čitalac iz Novog Beograda to tražio.

10. Vaša omiljena igra je:

a) Piratbusters (istrebljivači pirata)

b) Brankovo kolo (igra avanture sa jednim životom)

c) Manic Writer (prepisivanje knjiga)

d) Krekeri — razbijači zaštite

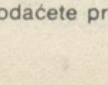
11. Kako ćete na najbolji način na brzaka da zaradite lovu na račun kompjutera?

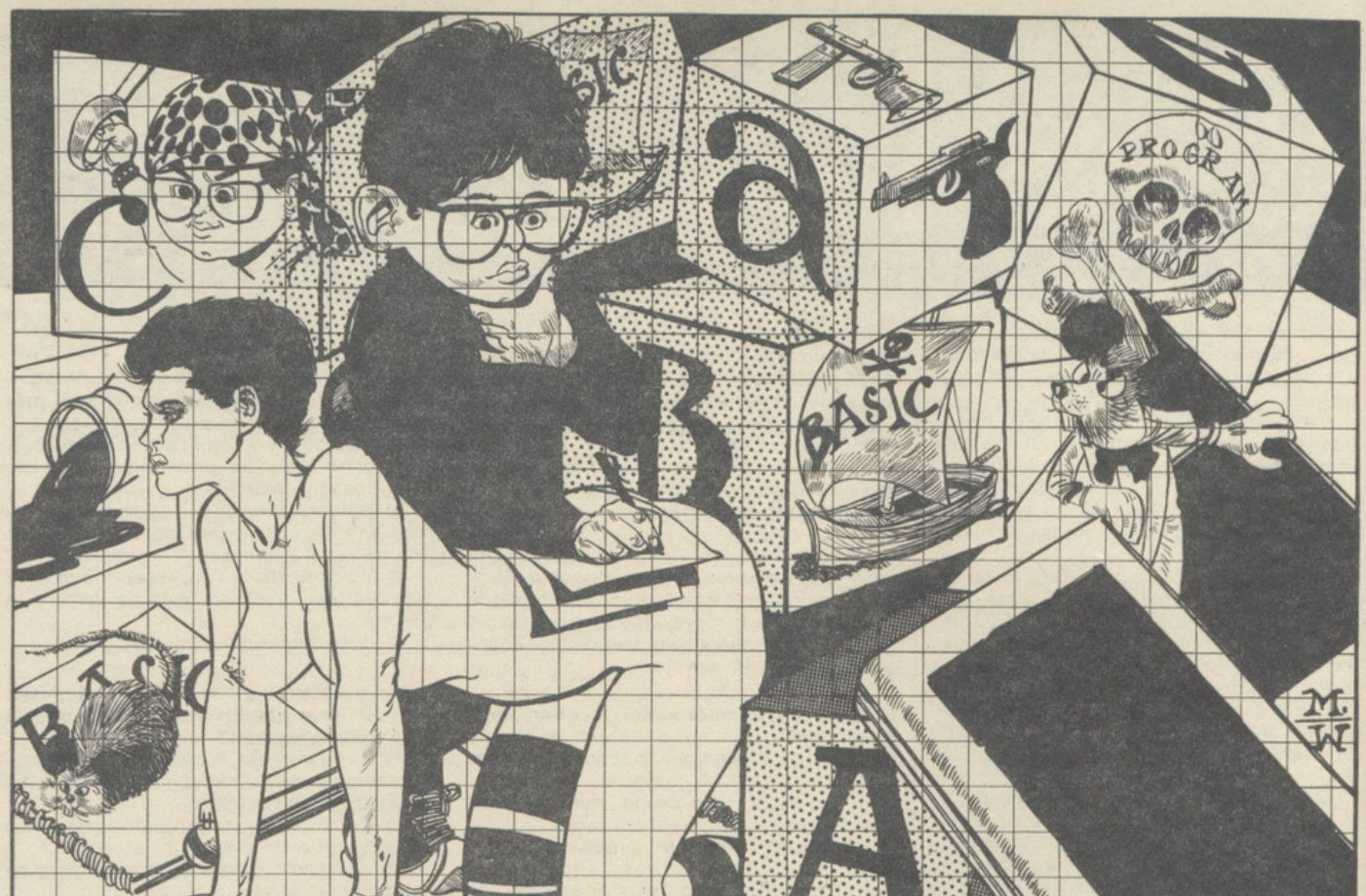
a) Izdaćete jednu knjigu i dva prijatelja.

b) Objavićete kompjuterski poster sa pogrešnim spiskom naredbi.

c) Kumovima, pašenozima, šuracima i ostaloj mnogobrojnoj rodbini držaćete kurseve bejzika i prodavati programe 13,5% jeftinije.

d) Prodaćete program Englezima.





Ilustracija: Miša Marković

12. Naslov Vaše najnovije knjige iz oblasti računarstva biće:

a) „Kompjuter kao velika nužda" (razmatra probleme društva koje iz informacijske predistorije zakoračuje u takođe informacijski srednji vek).

b) „Kako bez muke napisati knjigu o računarima"

c) „Majstorije na džojstiku".

d) „Bitak i užitak".

13. Gde biste najradije držali kurseve bejzika?

a) U mesnoj zajednici.

b) U Engleskoj.

c) U liftu, između prizemlja i osmog sprata.

d) U fioci.

Ako su Vam svi odgovori pod a), Vi ste zdrava, zrela i uravnotežena osoba sa visokom moralno-stručnom podobnošću, naša radost, dika i uzdanica. Samo nastavite tako!

Ako ste na sva pitanja odgovorili sa a) i b), Vaši stavovi nisu uvek precizno formulisani, ali u Vaše opredeljenje ne bi trebalo sumnjati. Treba još da radite na svom usavršavanju i distanciranju od etički štetnih ideja.

Ako u odgovorima imate a), b) i c), Vaši su protivrečni i nedosledni, ali ste u biti poštena osoba koja luta i kojoj treba pomoći da nađe pravi put.

Ako pored ostalih odgovora imate i poneko d), Vi ste već zatrovani i očito je da ste pod uticajem lošeg društva. Promenite sredinu i odrecite se prijatelja, jer je očigledno da Vas vuku u ponor od svih veći.

Ako Vaši odgovori sadrže veći broj d), to je već indikativno. U pitanju je teška indokrinacija društveno neprihvatljivim idejama. Ipak, još ima nade za Vas. Potrebno je kompletno ispiranje mozga; posle toga nećete morati da se odreknete prijatelja — oni će se odreći Vas.

Ako ste na sva pitanja odgovorili sa d), Vi ste neizlečiv slučaj moralnog raspada i najnižih asocijalnih nagona. Ne samo da niste dovoljno licemerni, nego još pokušavate to da kompenzujete svojim cinizmom. Vaše pogubno delovanje je tim opasnije zbog Vašeg nastojanja da i druge navučete. Zbog Vas i sličnih nosilaca pojave, zdrave subjektivne snage moraju povesti energičnu akciju razotkrivanja i razobličavanja.

Nagradna igra za one koji su na sva pitanja, odgovorili sa a)a)a)a)a)a)a)a)a)a):

Onaj ko pohvata najveći broj onih koji su na sva pitanja odgovorili sa d) dobiće glavnu nagradu: Velikog zlatnog Kerbera, značajno priznanje koje će mu biti dodeljeno uz sve počasti koje dolikuju jednom A-pozitivnom članu naše društvene zajednice.

D-negativni nosioci negativne pojave koji budu otkriveni i uhvaćeni, ili koji se sami prijave, moraće javno i pred svima da priznaju svoja nedela.

Jelena Rupnik

Računari u akciji

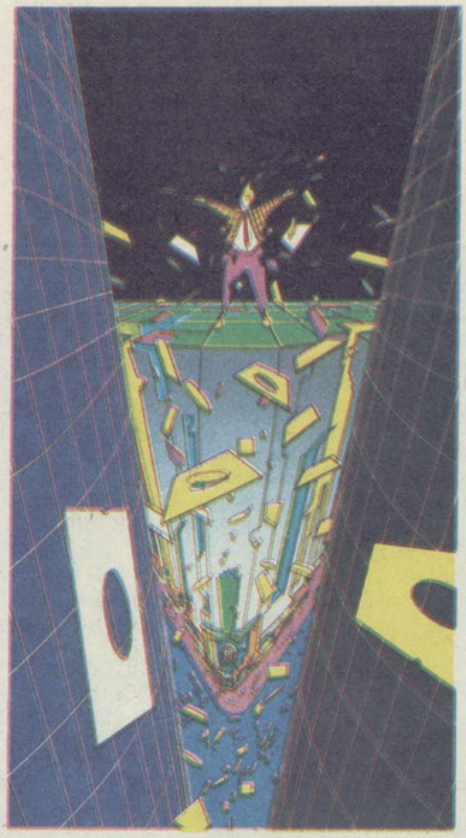
baze podataka

Posle obrade teksta, rad sa bazama podataka privlači najširi krug ozbiljno nastrojenih korisnika kućnih računara. Programi ovoga tipa omogućuju da u memoriji kompjutera i na disku držimo velike skupove raznih podataka koje je moguće brzo i efikasno menjati, poretraživati i štampati. Ako je baza podataka integrisana sa tekst procesorom, moći ćemo da je koristimo za pisanje službenih pisama i štampanje spiskova članova nekog kluba ili pretplatnika na neku knjigu i tako bitno pojednostavimo i pojeftinimo organizaciju posla. Za rad sa komercijalnim programima za baze podataka je, jasno, potrebno daleko manje znanje nego za sastavljanje bejzik ili fortran programa koji će obrađivati podatke, ali je ipak neophodno upoznati osnovne pojmove i — pažljivo proučiti uputstvo za upotrebu!

Pre nego što se pozabavimo terminima od značaja za rad sa bazama podataka, posvetićemo malo vremena jezičkim nedoumicama. Činjenica je da su se kod nas računari (pri tom prvenstveno mislimo na velike kompjuterske sisteme) najpre koristili za obradu podataka, što znači da postoje manje ili više dobri prevodi za osnovne engleske termine. S druge strane, program za rad sa bazama podataka koji ćete koristiti će, po svemu sudeći biti uvezen iz inostranstva, što znači da će njegovo uputstvo biti pisano na engleskom; ako koristimo jedne termine u tekstu, a vi pročitate druge termine u uputstvu, ovaj napis nema mnogo šanse da bude koristan. Zato smo odlučili da neke od izraza koji su se veoma odomaćili prevedemo, a da za druge zadržimo originalna fonetski ispisana imena. Na kraju, za one koji imaju problema sa jezikom, dajemo mali uporedni rečnik.

Izbor . . .

Datoteku (fajl) možemo da zamislimo kao veliki metalni orman sa mnogo fioka kakve gledamo u starim kriminalističkim filmovima ili u novoj još nekompjuterizovanoj „Dinastiji". Jedan orman (ili nekoliko ormana koji se nalaze u istoj prostoriji) sadrži fascikle sa podacima o raznim artiklima (u našem primeru svaki prestupnik, odnosno svaka naftna bušotina ima po jedan dosije). U kompjuterskoj terminologiji svaki dosije nazivamo **slogom** datoteke (record). Struktura svakog sloga je, u većini slučajeva, jednaka: on, na primer, sadrži naziv bušotine, njenu lokaciju, datum početka radova na njoj, informacije o do tog momenta postignutim rezultatima i troškovima, ime čoveka koji je zadužen za radove i tako dalje. Svaka od ovih informacija se naziva poljem (field) i sastoji se od slova ili brojeva, tako da razlikujemo numerička i alfanumerička polja. Vidimo da je struktura baze podataka hijerarhijska: na najvišem hijerarhijskom nivou je baza podataka (svi podaci koje poseduje firma Denver Karington), ispod nje su pojedine datoteke (npr. datoteka bušotina) koje se sastoje od slogova (podaci o svakoj bušotini), dok se slogovi sastoje od polja, a polja od slova i brojeva. Veliki kompjuterski sistemi, a u



poslednje vreme i složeniji programi za kućne računare, podržavaju i dodatne hijerarhijske nivoe kao što su grupna polja, pa čak i razlike u suštini obrade podataka — povećava se jedino njen komfor.

Kapacitet datoteke

Najvažnija karakteristika programa za obradu podataka je koliko kilo/mega/giga bajta informacija može da „sažvaće". Programi ovoga tipa pisani za kućne računare se dele na dečije igračke, koje rade isključivo sa podacima u memoriji, i profesionalne proizvode koji ograničavaju količinu podataka jedino slobodnim prostorom na flopi (ili hard) disku. Programi koji rade isključivo sa podacima u memoriji se karakterišu brzim pretraživanjima ili sortiranjima i pogodni su za kućnu upotrebu: adresar, telefonski imenik, kućni troškovi, spisak softvera i, na samoj granici njihovih mogućnosti, sređivanje manje kućne biblioteke. Za sve veće poslove program podatke treba da drži na disku, dok se u memoriju upisuje samo jedan njihov manji deo, na primer slog koji se trenutno prikazuje ili edituje. Time se, jasno, znatno gubi na brzini, ali je rad i dalje neuporedivo brži od ručnog pretraživanja kartoteke. Da bi se gubici u brzini smanjili, primenjuju se razni principi rada sa indeksima i u lančanim listama koji mogu da daju fascinantne rezultate, tako da program za rad sa bazama podataka možemo proceniti jedino ako ga probamo: treba tražiti od prodavca disketu sa podacima koja je skoro puna i proveriti koliko je vremena potrebno za dodavanje novog sloga, pronalaženje nekog starog ili sortiranje datoteke.

Brzina sortiranja

Brzina i fleksibilnost sortiranja su aspekt po kome većina korisnika sa razlogom rangira programe za baze podataka. Smisao sortiranja kompjuterske baze podataka možda, na prvi pogled, nije jasan: dosijei prestupnika u policijskom ormanu moraju da budu sortirani po prezimenu, jer bi za čoveka bilo praktično nemoguće da ih lista jedan po jedan. Kompjuteru je, reći će neko, svejedno da li će pretraživati sortiranu ili nesortiranu datoteku: čitaće je slog po slog dok ne pronađe onaj koji mu je potreban. Nije, međutimm, baš tako: najčešće se od računara traži da slog pronađe po nekom fiksnom polju, na primer po imenu i prezimenu. Ako je datoteka sortirana po ovom polju, računar će moći da primeni binarno pretraživanje i daleko brže pronađe ono što mu je potrebno. Ukoliko se datoteka pretražuje po nekim drugim kriterijumima, od čitanja svakog posebnog sloga se teško može pobeći.

Loši algoritmi za sortiranje imaju red složenosti n , što znači da će za sortiranje 200 slogova biti potrebno četiri puta više vremena nego za sortiranje 100 slogova. Dobri algoritmi za sortiranje imaju red složenosti n*log n, što znači da je za sortiranje 200 slogova potrebno 2.3 puta više vremena nego za sortiranje 100 slogova; razlika između ovih veličina postaje daleko značajnija kada broj slogova raste. Osim toga, daleko će brže sortirati program koji na disku drži sadržaje slogova, a u memoriji

→

tablicu sa njihovim redosledom — slogovi se, tada, fizički ne pomeraju, već se samo zamenjuju redovi u tablici. Sve u svemu, razlike u brzini rada programa za sortiranje mogu da budu neverovatno velike i njihovo poređenje je opravdan kriterijum za rangiranje sistema za baze podataka.

Tehnike pretraživanja

Pretraživanje je sledeća radnja koju program za rad sa bazama podataka uvek podržava. Brzina pretraživanja obično nije kritična: program kome se zadaju neki složeni uslovi po kojima treba da pronađe slog će morati da čita datoteku od početka prema kraju. Kod pretraživanja se, međutim, ceni fleksibilnost: treba omogućiti korisniku da zadaje složene uslove po kojima će program vršiti pretraživanje. Možemo, na primer, da zahtevamo da se pronađu svi prestupnici koji žive na Zvezdari i koji su osuđivani za sitne krađe, a nisu se nalazili u zatvoru na dan 1. marta 1985. godine. Od programa za pretraživanje kome je postavljen ovakav zahtev očekujemo određenu dozu inteligencije: on treba da zna da neko ko je bio u zatvoru od 1. januara do 15. aprila nije mogao da bude na Zvezdari prvog marta i da je, u nekom još komplikovanijem slučaju, Vranjska ulica na Zvezdari, a Čolak Antina nije. Poslednji primer pokazuje da, na velikim sistemima pa i na manjim računarima koji su povezani sa hard diskom ili kompjuterskom mrežom, program za bazu podataka može da konsultuje još neke banke podataka (u našem slučaju, kompjuterski plan grada) osim datoteke koju pretražuje. Ukoliko želite da znate kako će sve to izgledati jednoga dana, preporučujemo vam naučno-fantastični roman Frederika Pola „Kapija 2"!.

Želeći da ubrzaju pretraživanje, autori programa za baze podataka često se odlučuju da oslabe njegove karakteristike: nije malo čak i vrlo skupih programa koji pri pretraživanju uzimaju u obzir samo prvih šest ili deset slova (brojeva) u nekom polju. Ovakva solucija može da bude dobra u nekim primenama, dok u drugim predstavlja veliko ograničenje. Međutim, neki programi ovoga tipa omogućavaju da se za svaki slog definiše primarni ključ, to jest polje od četiri ili pet slova koje će računar držati u memoriji zajedno sa tabelom sa indeksima slogova. Ukoliko je u to polje upisana skraćenica koja karakteriše (ne)dela nekog prestupnika, računar će praktično trenutno moći da pronađe sve registrovane prodavce narkotika za nekoliko sekundi dok bi učitavanje svih slogova sa diska odnelo nekoliko časova.

Editovanje slogova

Četvrta karakteristika programa za baze podataka koju smatramo bitnom je mogućnost editovanja postojećih slogova. Jasno je da svaki program ovoga tipa omogućava da se podaci u nekom slogu izmene, ali pravi problem nastaje tek onda kada editovanjem slog postane bitno duži nego što je bio. Kako su slogovi gusto pakovani na disketi, za ovo produženje neće biti mesta, pa će slog biti obrisan i prepisan na neko drugo slobodno mesto, čime će se u bazi podataka pojaviti rupa. Dobar program će, kada disketa bude puna, izvršiti takozvani „trash colecting" i kompaktovati postojeće slogove, čime će se na kraju datoteke stvoriti veliki slobodan prostor. Osrednji program će ponuditi korisniku bejzik rutinu koja prepisuje koristan sadržaj diska sa praznim prostorima na tek formatirani disk, dok će loš program jednostavno ignorisati ovaj problem.

Organizacija programa

Može se dosta raspravljati o tome da li su bolji oni tekst procesori organizovani po menijima ili tekst procesori u kojima treba

Lozinke

Vlasnicima kućnih računara ne treba mnogo govoriti o zaštićenim programima — dobro im je poznato da su softverske firme smislile mali milion metoda koji kupcu omogućavaju da nesmetano koristi program, ali ga sprečavaju da ga kopira i dalje distribuira. Poznato je, isto tako, da ni jedan od tih metoda nije stoprocentno siguran.

Zaštita podataka je sasvim drukčija stvar: njen cilj je sprečavanje neovlašćenih lica da čitaju poverljive informacije. Disketu sa zaštićenim podacima biste, dakle, nesmetano mogli da poverite bilo kome na čuvanje; on bi bio u stanju da napravi neograničen broj kopija, ali i dalje ne bi bio u stanju da pročita podatke koji su upisani na njoj. Da stvar bude još lepša, neovlašćeno lice (ili ‚bad guy', kako ga slikovito zovu u američkoj literaturi) nema načina da na smislen način promeni podatke na disketi; on, jasno, može da ih uništi, ali ne i da modifikuje neke od njih i tako sebi pribavi bilo kakvu korist.

Zanimljivo je da se ne može postići stoprocentna zaštita podataka; štaviše, čak i početnik može bez mnogo problema da sastavi bejzik program koji će zaštititi podatke u bilo kojoj datoteci.

Kako taj program radi? Najpre treba da sastavimo proceduru koja, na osnovu broja koji zadamo, generiše sekvencu pseudoslučajnih brojeva između 0 i 255, pri čemu je vrlo bitno da isti početni broj uvek generiše istu sekvencu. Jedna od najjednostavnijih procedura koje ispunjavaju taj uslov je:

```
1000 DEFPROCs1br
1010 B=EXP (SEED+3.1415927)
1020 SEED=B-INT (B)
1030 SLBR=INT (B*256)
1040 ENDPROC
```

Program koji šifruje datoteku glasi otprilike ovako:

```
10 INPUT „Početna lozinka";SEED
20 INPUT „Ime izvorne datoteke";DI$
30 INPUT „Ime objektne datoteke";D2$
40 X=OPENIN D1$
50 Y=OPENOUT D2$
60 REPEAT
70 A=BGET#X
80 PROCs1br
90 A=A+SLBR:IF A>255 THEN A=A-256
100 BPUT#Y,A
110 UNTIL EOF#X
120 CLOSE#X:CLOSE#Y
130 DELETE D1$
```

Princip rada programa je sasvim jednostavan: čita se prvi bajt ulazne datoteke, uvećava za slučajan broj, a zatim se rezultat upisuje u izlaznu datoteku. Kada, na kraju, ulazna datoteka bude obrisana, sadržaj izlazne datoteke će biti besmislen za svakoga osim onoga ko zna lozinku. On će, kada bude želeo da koristi podatke, izvršiti program koji se od gornjeg razlikuje samo u jednoj liniji:

```
90 A=A-SLBR:IF A>0 THEN A=A+256
```

Startovanjem ovoga programa i unošenjem lozinke (istog decimalnog broja koji je unesen kada je datoteka šifrovana) dobiće se početna datoteka, dok će „zaključana" verzija biti obrisana.

Programi za baze podataka obično sadrže generatore pseudoslučajnih brojeva koji su daleko bolji od onoga koji smo mi dali i koji omogućavaju da početna lozinka bude reč, a ne broj, ali je princip njihovog delovanja potpuno isti: lozinka nije zapisana nigde na disku i ne postoji nikakav način da neko ko je ne zna u nekom zamislivom vremenskom intervalu od par hiljada godina pročita „zaključane" podatke.

Nevolja nastaje kada vlasnik podataka zaboravi lozinku — njegov višemesečni rad se začas pretvara u gomilu besmislenih podataka. Osim toga, ukoliko koristite neki komercijalni program i zaštitite svoje podatke lozinkom, verovatno nećete biti u stanju da ih čitate i obrađujete primenom nekih bejzik rutina koje ćete, uz pomoć ‚Utility diska', samostalno napisati. Čini se da je, zbog svega ovoga, daleko lakše i bezopasnije zaključavati diskete nego podatke na njima. Ne recite da vas nismo upozorili!

kucati komande, ali je takva dilema kod baza podataka rešena — svi vredni programi ovoga tipa pisani za kućne računare su organizovani po menijima, jer se pokazalo da samo tako korisnik može da iskoristi sve potencijale programa. Važno je da, osim menija, program ima i neku vrstu HELP moda u kome će korisnik dobiti obaveštenja o funkcijama koje su pridružene „soft" tasterima i eventualnom značenju slova kada je zajedno sa nekim od njih pritisnuta dirka CTRL. Ovakva organizacija programa može, međutim, da dovede do drugog problema: za obavljanje neke sasvim obične funkcije potrebno je proći kroz nekoliko nivoa menija i, prema tome, pritisnuti desetak tastera. Samo se u praksi može pokazati da li su autori nekog programa napravili dobar kompromis između brzine rada sa programom i jednostavnosti njegove upotrebe.

Fleksibilnost programa je poslednja karakteristika o kojoj imamo nameru da govorimo. Jasno je da autori programa za rad sa bazama podataka ne mogu da predvide sve sitnice koje će nam zatrebati. Tim pre, oni ne mogu da predvide mogućnosti tekst procesora koji koristimo, karakteristike našeg printera i tipove podataka koje smo na neki drugi način upisali na disk. Zato je vrlo zgodno ako se korisniku programa za baze podataka koji poznaje bejzik omogući da uz, umeren trud, sastavi programe koji će pristupati bazi podataka i vaditi podatke iz nje, ili programe koji će pripremati podatke na način koji baza podataka može da prihvati. Zato je zgodno kupiti program za rad sa bazama podataka uz koji se dobija i „utility disk" sa skupom bejzik i mašinskih potprograma.

. . . i kreiranje datoteke

Pošto, sledeći savete iz prethodnog poglavlja i proučivši prikaze programa za baze podataka iz stranih časopisa i sledećeg broja „računara", odaberemo program koji nam odgovara, treba da uložimo malo vremena u privikavanje na njega. Bez obzira na činjenicu da ste računar možda kupili baš zbog rada sa bazama podataka i da vam je važno da što brže počnete da koristite njegove usluge, savladajte nestrpljenje i sastavite jednu vrlo jednostavnu bazu podataka koja će vam omogućiti da vežbate. Nema, naravno, prepreka da počnete sa kreiranjem baze podataka koja vam je zaista potrebna i da se učite na njoj, ali treba da se pomirite sa tim da ćete ovu bazu podataka vrlo brzo morati da uništite i da započnete posao od početka — ona, jednostavno, neće biti zasnovana dovoljno fleksibilno. Zbog toga vam predlažemo da zajedno sa nama pokušate da kreirate datoteku sa imenima, adresama i drugim podacima o vašim poznanicima u koju ćete moći da upisujete i napomene o tome šta kome dugujete i šta vam ko duguje.

Započinjanje baze podataka je, bez sumnje najneprijatniji deo posla: treba da se odlučite za veličinu sloga i za polja koja će ga sačinjavati. Značaj ove operacije verovatno ne treba posebno objašnjavati — slog treba da bude dovoljno velik da omogući smeštanje svih podataka koji su vam potrebni i svih podataka koji će vam biti potrebni u budućnosti, a da, istovremeno, bude dovoljno mali da ne zauzme preveliki deo diska. Za našu datoteku će nam biti potrebna polja čije su dimenzije prikazane na slici 1.

```
Prezime:                15 (A)
Ime:                    10 (A)
Telefon:                10 (N)
Ulica:                  15 (A)
Broj:                    3 (N)
Grad:                   15 (A)
Poštanski broj:          5 (N)
Status:                  2 (A)
Duguje:                 45 (A)
Potražuje               45 (A)

A: alfanumeričko polje
N: numeričko polje
```

Pošto na papiru isplaniramo slog, startovaćemo program koji će nam predstaviti glavni meni, u kome je jedna od opcija, svakako, kreiranje nove datoteke. Izabraćemo tu opciju (verovatno će biti potrebno da takvu odluku potvrdimo jednim „Yes" obzirom da njeno izvršavanje možda uništava deo sadržaja diskete). Računar će proveriti da li je disketa koju smo pripremili korektno formatirana i dovoljno prazna (neki programi ovoga tipa zahtevaju da za svaku bazu podataka odvojimo po jednu disketu) i, ako je sve u redu, zahtevati da otkucamo ime baze podataka. Ovo ime nije naročito bitno sa funkcionalnog aspekta, ali će nam pomoći da docnije pronađemo disketu sa podacima koji su nam potrebni. Sledе pitanja o datumu kreiranja i nekim drugim podacima među koje, kod iole boljih programa, spada i pitanje o potrebi za štićenjem baze podataka lozinkom koja će onemogućiti ostalim ukućanima da zaviruju u imena vaših prijatelja (i prijateljica).

Definisanje sloga

Sledeća faza je startovanje ekranskog editora koji treba da vam omogući da definišete izgled sloga (u siromašnijoj verziji će vam jednostavno biti postavljena pitanja o svakom polju). Konstruisanje sloga možete da zamislite kao pripremanje matrice u koju ćete docnije ubacivati podatke: kada god bude trebalo da unesete sadržaj nekog sloga, računar će vam iscrtati formular koji upravo kreirate i od vas zahtevati da ga popunite. Formular koji odgovara našoj bazi podataka je dat na slici 2.

Na slici vidimo da je svako polje dobilo po neko ime ispred koga je upisana skraćenica. Ime je isključivo estetske prirode i biće prikazivano zajedno sa formularom — računar ga neće upisivati na disk zajedno sa podacima, niti ćemo moći da ga referenciramo pri sortiranju ili pretraživanju datoteke. Skraćenica u zagradi, takozvani „tag name" (osobina koju imaju samo bolji programi), upisuje se zajedno sa podacima na disk i može da se koristi za referenciranje polja. „Tag name" je, na prvi pogled, nepotreban luksuz kada već postoje obična imena. Ipak, na taj se način značajno štedi memorija, a istovremeno povećava komfor rada, jer pri unošenju podataka ne morate da razmišljate o tome da li je GR polje za naziv grada ili oznaka grupe u koju ste svrstali vaseg prijatelja. Dobar program za rad sa datotekama će omogućiti definisanje primarnog ključa za sortiranje; u našem slučaju to je polje STATUS, u koje možete da upisujete dvoslovnu skraćenicu koja će bliže odredivati osobu na koju se slog odnosi (npr. PR za prijatelje, RO za rođake, KL za kolege, CO za vlasnike kompjutera iste firme, PI za entuzijaste od kojih kupujete programe i tako dalje). Ovo polje će, ako ga dobro upotrebite, omogućiti da višestruko ubrzate buduća pretraživanja.

Popunjavanje formulara

Pošto je prekrivač (overlay) sloga definisan, računar će nas vratiti u glavni meni — datoteka je kreirana i možemo da počnemo da je popunjavamo. Izabraćemo opciju ‚Append' ili ‚Insert' i na ekranu će se pojaviti prazan formular. Ekranski editor koji će automatski biti startovan će nam omogućiti da strelicama pokrećemo kurzor i u polja upisujemo podatke. Kada unesemo

```
                                                    Status:__ (ST)

Prezime: _____________ (PR)
Ime:     _____________ (IM)

Telefon: ___ (PB) _____ (TL)

Ulica:   _____________ (UL)      Broj:   ___ (BR)
Grad:    _____________ (GR)      Pošta:  _____ (NO)


Duguje:  _____________ (D1)      Potražuje: _____________ (P1)
         _____________ (D2)                 _____________ (P2)
         _____________ (D3)                 _____________ (P3)
         _____________ (D4)                 _____________ (P4)
```

REČNIK ŽARGONA

ACCESS — način pristupa datoteci.
ALPHABETIC FIELD — polje u koje se smeštaju isključivo slova.
ALPHANUMERIC FIELD — polje u koje se smeštaju slova i (eventualno) brojevi.
AMEND RECORD — menjati neka polja u slogu.
APPEND — dodavanje slogova na kraj datoteke.
ASSOCIATE VARIABLE — promenljiva (ponekad interna) koja ukazuje na sledeći slog datoteke pri sekvencijalnom pristupu.
BACKUP — formiranje kopije datoteke koja će biti korišćena u slučaju kvara na mediju na kome se datoteka čuva.
BLOCK — skup slogova koji se, zbog bržeg rada, zajedno učitavaju u bafer.
BROWSE — pregledanje datoteke ili sabseta slog po slog
CARRIDGE CONTROL — da li će se na kraju sloga nalaziti CR, CR i LF ili blanko.
CHARACTER — slovo, broj ili specijalni znak.
CHARACTER SET — sva slova i znakovi kojima računar ili štampač raspoloaže.
CLOSE — zatvaranje datoteke.
DATA BASE — baza podataka ili program za rad sa bazama podataka.
DEFAULT — vrednost ili opcija koja se podrazumeva ako se ništa ne navede.
DELETE — brisanje
DIRECT ACCESS — način pristupa kod koga se može direktno čitati bilo koji slog.
EDITING — promena nekih podataka.
ESCAPE SEKVENCA — sekvenca kodova koja se šalje printeru. Počinje sa 27.
EXTEND SIZE — uvećanje prostora na disku koji je dodeljen datoteci.
FIELD — polje, deo sloga.
FILE — datoteka, skup slogova.
FINISH — regularan završetak — promene se upisuju u datoteku.
FIXED RECORD — svi slogovi imaju jednaku dužinu.
FORMATTED FILE — datoteka koja sadrži ASCII tekst.
HELP — naredba kojom se od računara traže podaci o drugim naredbama.
INDEXED FILE — datoteka u kojoj se slogovi pronalaze prema sadržaju jednog polja — ključa.
INPUT FORM — matrica prema kojoj se unose podaci.
KEYED ACCESS — način pristupa indeksnim datotekama.
MAIL-MERGING — pripremanje pisama i etiketa za koverte na osnovu sadržaja datoteke.
MENUS — meni pomoću koga korisnik kontroliše rad programa.
MERGE — mešanje datoteka tj. formiranje nove datoteke od više postojećih.
NUMERIC FIELD — polje koje sadrži brojni podatak (obično celobrojan).
OPEN — otvaranje datoteke.
PASSWORD — lozinka koja sprečava neovlašćeni pristup podacima.
PRIMARY KEY — polje po kome se datoteka sortira.
PRINTER DRIVER — program pomoću koga se podaci prenose na štampač.
RANDOM FILE — datoteka u kojoj u svakom trenutku može direktno da se pristupa bilo kom slogu.
READ ONLY FILE — datoteka koja sme samo da se čita.
RECORD — slog, skup podataka o nekoj osobi/predmetu.
REVIEW — brzo pregledanje sadržaja datoteke.
SECUNDARY KEY — polje po kome se sortiraju slogovi sa istim primarnim ključem.
SEQUENTIAL ACCESS — čitanje (ili upis) datoteke slog po slog.
SEQUENTIAL FILE — datoteka u kojoj se slogovima mora pristupati redom.
SORT —ređanje slogova u rastući ili opadajući niz.
STATUS LINE — prva ili poslednja linija ekrana u kojoj vidimo obaveštenje o onome što program u nekom trenutku radi.
SUBSET — određen broj slogova izdvojen iz datoteke.
TAG NAME — stvarno ime polja.
TITLE — naslov ili ime datoteke.
UNFORMATTED FILE — datoteka koja sadrži brojeve u internom formatu.
UPDATE FILE — ažurirati datoteku.
USERNAME — ime korisnika, obično se navodi pre lozinke.
VARIABLE RECORD — slogovi su promenljive dužine.
SHARED FILE — datoteka kojoj (na velikom sistemu) istovremeno pristupa više korisnika.

čitav slog, pritisnućemo neki od „soft" tastera, dobiti novi prazan formular i ponoviti čitavu operaciju. Pri unošenju podataka treba biti veoma pažljiv, jer ćemo svaku grešku docnije skupno platiti: ako umesto PERA PERIĆ otkucamo PERA PERUĆ, program za pretraživanje neće moći da nađe podatke o ovom našem prijatelju, iako smo sigurni da smo te podatke otkucali. Za ovu grešku smo, na neki način, sami krivi, pa nemamo mnogo razloga da se ljutimo na program koji koristimo. Neki programi za rad sa bazama podataka, međutim, prave razliku između velikih i malih slova, pa nećemo moći da pronađemo slog koji se odnosi na PERU PERIĆA ako je u njega upisano Pera Perić. Stvar će biti još lepša ako podatke o polovini poznanika kucamo danas, pri čemu pišemo prvo prezime pa ime, a ostatak dopišemo sutra i to, naravno, obrnutim redom!

Sortiranje i pretraživanje

Pošto unesemo sve podatke, vratićemo se u glavni meni i zahtevati sortiranje. Računar će tražiti da navedemo ime polja koje se zove primarni ključ, a zatim imena polja sa sekundarnim ključevima (primarno polje za sortiranje bi po svoj prilici, trebalo da bude prezime, sekundarno ime, a sledeće mesto stanovanja). Datoteka će, zatim, biti sortirana po primarnom ključu (ako se radi o numeričkom polju, manji brojevi će se nalaziti ispred većih, a ako se radi o alfanumeričkom polju, sortiranje će se obaviti po abecedi) a zatim će svi slogovi koji imaju jednaki primarni ključ biti sortirani po sekundarnom ključu, a oni sa istim primarnim i sekundarnim ključem, po sledećem specificiranom polju. Zavisno od broja slogova, sortiranje je operacija koja najčešće ume da potraje.

Kada nam docnije budu zatrebali podaci o nekom od prijatelja, izabraćemo opciju Search i, na način koji je opisan u okviru uputstva za upotrebu programa koji koristimo, zadati uslove prema kojima se pretraživanje vrši. Pri tom će svakako biti omogućeno korišćenje džoker znakova o kojima smo govorili u napisu „disketna veza" u „Računarima 5". Ukratko,povisilice (number sign, „taraba", #) zamenjuje bilo koje slovo, a zvezdica bilo koju grupu slova tako da će SEARCH MI#IĆ pronaći sve MIIĆE (ako takvo prezime postoji), MIŠIĆE, MILIĆE i MIRIĆE, dok će SARCH MI*IĆ uz to pronaći i MILJKOVIĆE, MILANOVIĆE i slične. Dobri programi za baze podataka omogućavaju daleko fleksibilniju (i samim tim složeniju) upotrebu džoker znakova, koja vam verovatno neće nikada biti potrebna.

Umesto jednostavnog pretraživanja, kvalitetni programi omogućavaju kreiranje takozvanih subseta (subset). Ako očekujemo da će računar pronaći mnogo slogova koji ispunjavaju date uslove, verovatno nam neće biti zgodno da ih samo na kratko pogledamo — ukoliko propustimo slog koji smo tražili, moraćemo da ponovimo čitavu operaciju i tako izgubimo dosta vremena. Zato od računara možemo da zahtevamo da kreira odvojenu datoteku (subset), koja sadrži samo slogove osnovne datoteke koji zadovoljavaju neke uslove. Docnije ćemo moći brzo da pregledamo te slogove i pronađemo onaj koji nam je potreban. Korist od subseta je i subjektivna — ako očekujemo da neko pretraživanje dugo traje, možemo da ga startujemo, pa da se onda bavimo nekim drugim poslom. Ukoliko računar prikazuje slog čim ga pronađe, taj posao ćemo iz časa u čas morati da prekidamo, dok za kreiranje čitavog subseta nije potrebna nikakva intervencija — posvetićemo pažnju računaru tek kada bude u stanju da nam brzo prikaže podatke koje smo tražili.

Programski paketi

Program za rad sa bazama podataka postaje zaista koristan tek kada ga povežemo sa drugim softverom. U taj softver spada, pre svega, tekst procesor: treba da proučimo mogućnosti prenošenja sadržaja nekih slogova u fajl sa tekstom koji ćemo docnije ispisati. Neki programi su opremljena printer drajverima koji omogućavaju direktno prenošenje podataka na papir, ali je fleksibilnost takvih drajvera retko dovoljna čak i za najjednostavnije poslovne primene. Zato ćete na tržištu naći razne programe koji se zovu „Mail list" (ispisivanje etiketa za pisma uz korišćenje imena i adresa iz baze podataka), FieldCalc (izračunavanje srednjih ili prosečnih vrednosti sadržaja nekih polja), Consbase i slično. Ovakve programe ćete, na način koji smo već opisali, s vremena na vreme morati da dopunjavate softverom iz kućne radinosti.

Dejan Ristanović

Programi koje treba imati

ne daj se, ines!

Među programskom opremom koja ličnim računarima daje pravu vrednost, pored programskih jezika u prvi red se ubrajaju tekst procesori, programi za obradu podataka, vođenje knjigovodstva i uopšte, uslužni programi. Neki od njih su odigrali pionirsku ulogu u razvoju softvera, recimo VISICALC, drugi su postali veliki hitovi — WORDSTAR, dBASE II — a trenutno su na ceni kompletni programski paketi kao LOTUS 1-2-3, SYMPHONY, ili FRAMEWORK. Većina tih programa je namenjena „eplu", IBM-u, ili sličnim računarima, dok se za jeftine mašine poput „spektruma" uglavnom prodaju samo igre. Ipak, i za običnog, smrtnog YU hakera kome je IBM san, a „spektrum" realnost, od skora postoji rešenje: INES — program za obradu teksta i podataka, delo jednog našeg programera iz Slovenije, Primoža Jakopina.

Tekst procesor je, sigurno, najvažniji deo INES-a — od pedeset naredbi koliko poznaje ovaj program, skoro četrdeset je namenjeno upravo pisanju na računaru. Da bi se obezbedio konforan rad, na ekranu se prikazuje tekst u formatu 22 reda puta 64 znaka. Naravno, to „spektrumova" rezolucija vrlo teško podnosi, ali drugog rešenja sigurno nije bilo. Takođe, u cilju prijatnijeg rada, slova su bela na tamnoj pozadini, a tekst se na ekranu polako pomera, bez naglih skokova, od kojih posle dužeg rada zaboli glava.

Lako oblikovanje teksta, brzo ispravljanje grešaka, mogućnost da se vrše razne izmene i, jednom rečju, elastičnost u radu je ono što se očekuje od svakog tekst procesora, i INES sigurno neće izneveriti očekivanja. Međutim, INES pruža i neke pogodnosti koje slični inostrani programi, recimo slavni TASWORD II, ne pružaju.

Pre svega, tu su za nas vrlo važna slova ć, č, š, ž i đ, bez kojih obrada teksta gubi svaki smisao. Šta više, autor se pobrinuo da omogući INES-u da ta slova i odštampa, a ne samo prikaže na ekranu. Pogodnost važi samo za vlasnike štampača iz serije EPSON FX, DELTA i STAR GEMINI, ali to su, po svojoj prilici, i najrasprostranjeniji štampači kod nas, ne računajući Sinklerovu termičku

Inesov rečnik

		i	insert character	s	substitute
		I		t	setup
			Insert line	T	multiple records
a	adjust to right margin	J	Jump	u	unadjust
A	Append	k	kill to end-of-line, insert	U	Upper to lower case
b	beginning screen	K	rank	v	verify
B	Bytes left	l	last screen	V	dissolve
c	change	m	mark	w	delete word
C	Copy	M	Move	W	Where
d	delete character	n	next screen	x	exchange
D	Delete line	N	mailing list	X	display pixels
e	ending screen	o	order	y	center line
E	Exit	O	turn around	z	zero and insert
f	find	p	pick	Z	set end-of-file
F		P	Print	/	change places
g	Frequencies	q	sequence numbers		upper to lower case
G	lengths	Q	Query		duplicate file FILE
h	help	r	restore		join files
H	Help	R	Remove		

Zatim, postoji dosta korisna mogućnost štampanja cirkularnih pisama. Naime, kod ovakvih pisama, glavni sadržaj uvek ostaje isti, ali se zaglavlje menja od primerka do primerka. INES ceo posao obavlja automatski.

Ono što je, međutim, sigurno najinteresantnije i daleko najkorisnije je mogućnost da se razne slike i tabele uključe u tekst. Za slike se morate sami pobrinuti, pomoću programa MELBOURNE DRAW na primer, ali zato INES može da stvori sve tabele koje mogu da zatrebaju. Program za obradu podataka je, naime, sastavni deo INES-a, a rezultati njegovog rada se lako i brzo mogu preneti u tekst procesor. Naredbe koje manipulišu podacima nisu naročito brojne, ali su zato izuzetno moćne. Za tren oka može se ispremeštati cela datoteka, po određenom kriterijumu odstraniti sve što ne treba, sortirati ostatak, dodati još ponešto, formirati preglednu tabelu i rezultat odštampati. Ovakve mogućnosti, kombinovane sa tekst procesorom, širom otvaraju vrata INES-u za sve primene.

Veliki problem kod ovakve vrste programa je raspoloživa memorija. INES nije kratak program, ali je ipak za korisnika ostalo prohvatljivih 21K, što iznosi dobrih deset kucanih strana. Vrlo važna je i kompatibilnost sa Sinklerovim mikrodrajvom. To otvara nove horizonte i olakšava rad. Obrada podataka sa običnim kasetofonom je teško zamisliva i vrlo mukotrpna.

Zajedno sa INES-om dobijaju se još tri posebna programa — DENIS, SORTI i UTE.

DENIS služi za unošenje podataka u skladu sa određenim oblikom formulara i za istovremenu kontrolu ispravnosti unetih podataka.

SORTI je namenjen sortiranju vrlo velikih datoteka (do 37K) koje ne mogu da stanu u INES.

UTE je najzanimljiviji od sva tri dodatna programa. Omogućava da se u INES prenesu datoteke formirane pomoću nekog drugog programa. To mogu biti izvorni programi iz devpaka ili preskola, programi u bejziku, tekstovi napisani pomoću TASWORD-a, delovi RAM-a, ROM-a, ili slike.

Uputstvo za upotrebu od 68 strana koje prati INES je vrlo opširno i pruža mnoštvo primera. Inače, INES se po ceni od 1500 dinara može naručiti preko Mladinske Knjige iz Ljubljane.

Vladimir Kostić

Kako to rade Englezi

TASWORD I (TASMAN SOFTWARE) Tekst procesor sa mogućnošću obrade samo 32 karaktera u redu. Samo to ograničenje ga čini potpuno beskorisnim za neku ozbiljniju primenu.

TASWORD II (TASMAN SOFTWARE) Prvi, i pored INES-a, verovatno jedini ozbiljan program za obradu teksta namenjen „spektrumu". Može da radi sa 32 ili 64 karaktera u redu. Raspolaže mnoštvom vrlo moćnih naredbi i podržava EPSON-ove štampače iz serije FX. Vrlo je popularan i na zapadu i kod nas, mada, naravno, ne raspolaže našim slovima ć, č, š i ž. Može se vrlo lako prilagoditi za rad sa mikrodrajvom.

VU FILE (PSION) Jedan od prvih programa za obradu podataka namenjen „spektrumu" sa 16 ili 48K RAM-a. Ograničenih mogućnosti. Ne predstavlja naročito dobar izbor. Nije kompatibilan sa mikrodrajvom.

SMALL BUSINESS ACCOUNTS (WILLDEN) Program namenjen malim firmama maloj privredi?) za lakše vođenje knjigovodstva. Okrenut je vrlo uskom području primene. Nije stekao naročitu popularnost. Nije kompatibilan sa mikrodrajvom.

MASTER FILE (CAMPBELL SYSTEMS) Jedan od snažnijih programa za obradu podataka. Ima mogućnost da prikaže 51 znak u redu, što predstavlje lep kompromis između velikog broja slova u redu i čitljivosti. Lako se koristi zahvaljujući tome što se upravljanje vrši preko menija. Raspolaže dobrim i vrlo brzim komandama. Može da obrađuje do 32k podataka. Kompatibilan je sa mikrodrajvom.

VU CALC (PSION) Omogućava razna izračunavanja sa podacima sređenim u obliku matrice. Može da radi i na „spektrumu" sa 16K RAM-a. Nije nešto posebno proslavio. Nije kompatibilan sa mikrodrajvom.

Računari u poslovnoj primeni
mali računari u velikoj privredi

Nešto toliko mali kao što izgleda: Čak i za najmanji kućni računar u svakoj radnoj organizaciji može se naći čitav niz praktičnih poslova

Veliki računari, namenjeni obradi poslovnih podataka, već dvadesetak godina postoje u našim radnim organizacijama. Njihova centralna memorija se meri megabajtima a kapaciteti diskova gigabajtima, magnetne trake imaju velike brzine prenosa, kao i gustine upisa, a modemi omogućuju da se korišćenjem telefonskih linija formira čitava mreža računara. Zar u takvim uslovima naš kućni ljubimac stvarno ima šta da traži? I te kako! Zašto? Postoje uglavnom tri razloga:

- Veliki broj računskih centara nije kadrovski osposobljen da se bavi svim poslovima koji su od interesa za radnu organizaciju. Pri tome se, uglavnom misli na nedovoljnu usklađenost kadrovskih i tehničkih resursa u računskom centru, a mnogo manje na stručnu osposobljenost radnika. Većina radnih organizacija koristi svoje računare za obavljanje poslova vezanih za praćenje finansijskih promena (kupci, dobavljači, fakturisanje, lični dohodci, osnovna sredstva i slično). Kako je ova oblast vezana za praćenje zakonskih propisa koji se veoma često menjaju (kontni plan, stope revalorizacije, pravilnici o raspodeli), a skopčani su i sa propisanim rokovima, radnici u računskom centru prilično, ako ne i čitav, deo radnog vremena koriste za takozvano održavanje programa. Na taj način im ostaje malo slobodnog vremena da ga posvete nekim drugim oblastima primene, koje su, možda, i mnogo korisnije za radnu organizaciju, ali nisu bile prenete na računar, i nisu bile razvijene, jer su zakonski propisani rokovi nametali prioritet na prethodno navedene poslove.
- Izvestan broj radnih organizacija, s obzirom na ne male teškoće koje prate zamenu kompjuterskih sistema (uključujući tu zakonske propise, finansijske probleme, pa, konačno, i probleme prenosa razvijenih aplikacija na novi sistem) do te mere je opteretio svoju opremu da je praktično nemoguće postaviti na sistem neku aplikaciju.
- Uvek postoje poslovi koji su od interesa samo za neki manji deo radne organizacije. Ovakvih poslova ima jako mnogo — mogu biti čak i vezani za sam proces proizvodnje (kontrola alata, mernih instrumenata, preventivno održavanje opreme, i slično), ili van procesa proizvodnje (kadrovska evidencija, normativna akta, biblioteka, itd) — ali verovatno nikada neće ni biti postavljeni na veliki računski sitem.

Veliki i mali

Da bismo na neki način uporedili veliki računar za obradu podataka sa kućnim računarom, analizirajmo pojedine delove velikog sistema — centralni procesor, centralnu memoriju, jedinice sa direktnim pristupom, magnetne trake, štampač i komunikacione veze — kao i tri posebna parametra
— programske jezike i pomoćne programe, brzinu rada i tačnost. Pri tom nikako ne treba zaboraviti i odnos cena koji postoji između ovih sistema (primera radi, samo terminali za veliki sistem danas košta preko 100 miliona starih dinara).

Centralni procesor

Ako izuzmemo sisteme nekih proizvođača računara za obradu podataka koji su danas prisutni na našem tržištu (IBM, UNIVAC, ICL, kao i neki modeli NCR, DELTA, HONEYWELL, ERA), većina računara koji su instalinrani poslednjih godina kod nas su izgrađeni oko somobitnih mikroprocesora (NCR serija 90 HONEYWELL sistem 6, ERA 20 i 60, manji sistem DELTA) — znači oko istih ili sličnih mikroprocesora kao i kućni računar. Repertoar mašinskih instrukcija je praktično isti, što ima veoma velikih posledica na samu organizaciju čitavog računara.

Centralna memorija

Kapaciteti centralne memorije računara instaliranih u radnim organizacijama mere se, uglavnom megabajtovima. Računari izgrađeni oko osmobitnih mikroprocesora, a kojih je daleko najviše, omogućavaju programeru korišćenje 64 kilobajta po jednom radnom mestu (terminalu) — isto toliko koliko i mi imamo na stolu kod većine kućnih računara. Šta više, sedamdesetih godina, kada je jedan bajt centralne memorije koštao približno 1 dolar, samo je mali broj nas koji smo radili na sistemima za obradu podataka imao sreću da radi sa 64 kilobajta. Da se ne vraćamo u kasne šezdesete, kada je pojava sistema UNIVAC 1004 sa 1 kilobajt centralne memorije za nas bila pravi događaj.

Jedinice sa direktnim pristupom

Ukupni kapaciteti jedinica sa direktnim pristupom kod velikih sistema iznose neko-

Od ovog broja dobili smo novog saradnika koji će voditi rubriku za poslovnu primenu kućnih računara. To je MIHAILO KARAPANDŽIĆ, dipl. ing. elektrotehnike, koji na poslovima AOP-a radi već više od 20 godina. Odrastao je na velikim sistemima. Radio je prvo u Zavodu za primenu elektroničnih računara iz Zagreba, koja je zastupala japansku firmu FUJITSU. Tamo je učestvovao u uvođenju više sistema i obrazovanju kadrova. Pored ostalog, projektovao je Zajednički informacioni sistem SR Srbije u socijalno-zdravstvenoj zaštiti i Informacioni sistem budžeta Srbije. Sada radi u Zavodu za ekonomske ekspertize iz Beograda na poslovima projektovanja AOP sistema.

Iako se u poslednje vreme za kućne računre nalazi i u kući poneki ozbiljniji posao, ogromna računarska menažerija — koja se, poput bujice, sliva preko severnih granicau naše domove — uglavnom čami nelskorišćena, a novopečeni programeri nisu ni svesni kakvo možno oruđe imaju u ruci. Kućni računari su, naprosto, idealni za malu privredu, a mogu, na određenim poslovima da nadu svoje mesto čak i u najvećim radnim organizacijama — bez obzira da li u njima već postoji računski sistem ili ne. To, možda, najbolje znaju oni dečaci, za sada istina veoma retki, koji na sovjim skromnim kućnim mašinama obrću milione vršeći obradu podataka za radne organizacije. Ovim tekstom hoćemo da odgovorimo na pitanje „zašto da", a za sledeće brojeve „Računara" pripremamo seriju napisa koji će kroz niz praktičnih, konkretnih primera pokušati da odgovore i na pitanje „kako".

liko stotina megabajtova, pa i više gigabajtova. Kod kućnih računara možemo danas, u najboljem slučaju da imamo 10 megabajta na vinčester disku. Vidimo da je odnos veoma nepovoljan za kućne računare. Ali, ne treba zaboraviti da mi ne želimo da koristimo svoj računar da radi sve poslove, već mu namenjujemo samo neke.

Magnetne trake

Magnetna traka je, verovatno, jedinica koju nikada nećemo imati uz svoj kućni računar. Tazlog za ovo je, svakako, enormna cena jedinice magnetne trake, ali i činjenica da su danas u velikim računskim centrima magnetne trake degradirane u svojoj funkciji, pa služe uglavnom za čuvanje arhivskih datoteka. Ovu funkciju može na kućnom računaru u potpunosti da preuzme na sebe kasetna jedinica. Odnos brzina prenosa podataka između disk-jedinice i magnetne trake kod velikih sistema i brzine prenosa podataka između disketne i kasetne jedinice je približno isti. Međutim, čuvanje podataka kod kućnih računara je mnogo bolje vršiti kopiranjem disketa, tako da nam kasetna jedinica nije ni potrebna za poslove koje nameravamao da radimo.

Štampač

Svi poslovni računari su opremljeni linijskim štampačima koji mogu da štampaju između 300 i 2000 redova u jednoj minuti, pri čemu u jednom, redu može biti obično 132 znaka. Na žalost, o ovakvim brzinama štampanja na kućnim računarima možemo samo da čeznemo. Međutim ne treba zaboraviti da je skromnih nekoliko desetina znakova u sekundi, koliko imamo na našem mališi, u potpunosti poredljivo sa brzinama koje imaju matični štampači koji se danas nude na velikim sistemima kao „hard-copy" uz udaljene terminale.

Komunikacione veze

Po pravilu, svi savremeni računari za obradu podataka su komunikaciono orijentisani. To znači da mogu komunicirati bilo između sebe, bilo sa udaljenim terminalima, korišćenjem javnih ili iznajmljenih veza za prenos podataka. Međutim, danas je takođe moguće formirati i mrežu uz korišćenje kućnih računara. Povezivanje kućnog računara sa velikim sistemom je u potpunosti ostvarljivo. Na taj način smo u mogućnosti da za naše potrebe koristimo velike datoteke smeštene na jedinicama sa direktnim pristupom u okviru velikog sistema, tj. da kućni računar koristimo kao inteligentni terminal velikog sistema. Mada ovaj način korišćenja nije neinteresantan (naprotiv!) — on ipak ne predstavlja onu koju želimo da dodelimo kućnom računaru.

Programski jezici

Na velikim sistemima se uglavnom koriste kobol, PL/I i fortran, a u mnogo manjoj meri Algol, bejzik i paskar. Autoru nije poznat ni jedan slučaj da je na velikim sistemima kod nas u zemlji instaliran najnoviji programski jezik ADA, koji je nedavno u Sjedinjenim Američkim Državama zbog svojih kvaliteta, na mnogim mestima prihvaćen kao jedini programski jezik. Kućni računari, koji rade pod operativnim sistemom CP/M, imaju na raspolaganju sve programske jezike koji su razvijeni za velike sisteme, uključujući i programski jezik ADA. Korišćenjem kućnog kompjutera nismo nimalo hendikepirani u pogledu raspoloživih programskih jezika.

Što se tiče pomoćnih programa, i kod kućnog kompjutera nam je na raspolaganju osnovni — SORT/MERGE program. Takođe nam je na raspolaganju i veliki broj drugih programa za prepis datoteka, izmenu imena datoteka, testiranje programa i mnogi drugi.

Brzina rada

Brzina rada velikih sistema je, nesumnjivo, znatno veća od one koju imamo na kućnim računarima. Ali za poslove koje želimo da radimo na malim računarima rad u realnom vremenu nije imperativ (uostalom, kao i kod velikog broja poslova koji se rade na velikim sistemima). Jedino što želimo je da se obrada vrši dovoljno brzo — mnogo brže nego ako bismo radili bez računara. ako neko od čitalaca ima mogućnost da izvrši standardni benč-mark test za kućne računare na velikom sistemu (autor je to uradio na sistemu NCR 90/20, pri čemu je, da bi rezultati, bili što verodostojniji, u sistemu radio samo testirani program), verovatno će doći do rezultata koji mogu samo da posluže kao reklama za kućne računare.

Tačnost

Pod ovim pojmom, svakako, ne podrazumevamo da li će i kućni računar sabrati 2 i 2 i kao rezultat dati 4. Sigurno da hoće. Ali broj značajnih cifara sa kojima on radi je svakako veoma važan. Za veliki broj primena, devet cifara, koliko prikazuje, na primer, „amstrad" u bejziku je sasvim dovoljno. Što se tiče prikazivanja preko mantise i eksponenta, 10 na 38 stepen će takođe zadovoljiti veliki broj naših potreba.

Mesto pod suncem

Iz svega ovoga moguće je dosta kritički odrediti poslove koje u poslovnoj primeni možemo poveriti kućnom računaru. Pri tome uopšte ne treba razgovarati o njegovoj primeni pri projektovanju. Njegovo mesto je, svakako, na radnom stolu inženjera. Ali šta je sa ostalim delovima radne organizacije? Pri tome ne treba zaboraviti da nisu sve radne organizacije veliki sistemi sa više hiljada zaposlenih, već da ih ima daleko više sa nekoliko stotina radnika, relativno malim brojem osnovnih sredstava, nekoliko stotina kupaca i dobavljača. Postoje dva osnovna kriterijuma koji određuju da li se neki posao može postaviti na kućni kompjuter:

- veličine datoteka koje će se koristiti
- obim štampe na izlazu nakon završene obrade.

Od ova dva kriterijuma, drugi zaslužuje posebnu pažnju, ali i prvi ima svoju težinu. Pogodnom organizacijom podataka na različitim disketama (na primer, kupci su podeljeni po azbučnom redosledu) ovaj problem je mbguće, uz smanjenje komfora, prevazići. Ovo će za mnoge moje kolege izgledati kao jeres, ali neka se sete samo knjigovodstvenih mašina sa magnetnim konto karticama koje i danas neprikosnoveno vladaju u mnogim radnim organizacijama. Rešenje sa kućnim računarom je ipak mnogo bolje.

Međutim, problem obima štampe je nemoguće prevazići. Skromna brzina matričnih štampača je, u mnogome, limitirajući faktor primene kućnih računara u radnim organizacijama. Stoga je, pre nego što se pristupi izradi pojedinih programskih rešenja, najpre potrebno ustanoviti obim štampe koji je potrebno izvršiti u nekoj jedinici vremena, dovoljno kratkoj da zadovolji naše potrebe.

Što se tiče same konfiguracije kućnog računara, za primenu u radnoj organizaciji, dilema nema mnogo. Pored samog računara, nužno je raspolagati bar sa jednom jedinicom disketa i jednim matričnim štampačem. Druga disketna jedinica će u mnogome olakšati život kako onome ko bude razvijao samu aplikaciju, tako i osobama koje će koristiti tako razvijeni sistem.

Cilj ovog članka nije da pokaže kako je postojanje velikih sistema bezpredmetno kada su nam na raspolaganju kućni računari. Na protiv, danas u našoj zemlji nema ni iz daleka onoliko velikih i modernih sistema koliko bi nam bilo potrebno. Utoliko pre u svakoj organizaciji kućni računari imaju svoje mesto pod suncem. Uostalom, krajem sedamdesetih godina firma Teksas Instruments, sa svojim modelom 900, napravila je izvanredan uspeh, namenjujući ga upravo onome o čemu govori i ovaj članak.

Ukoliko ovaj tekst navede bar jednog čitaoca da razmišlja o mogućnostima primene kućnih računara u radnoj organizaciji i ako, kao posledica toga, bar jedan računar zaživi negde, autor će smatrati da je njegov trud u potpunosti nagrađen, a da je članak ispunio svoju namenu.

Mihailo Karapandžić

Spektrum Programator eproma

Pre ravno godinu dana, u „Računarima 2", objavljen je projekat programatora eproma za „spektrum" i „galaksiju" sa odgovarajućim softverom za prvi i najavom softvera za drugi računar. Čitaoci koji su pokušali da unesu onih nekoliko sotina bajtova mašinca našli su se, bez sumnje, u velikom čudu — objavljeni program je predstavljao prosti zbir besmislenih bajtova kao da je pre ulaska u štampu propušten kroz mašinu za šifriranje. Naši mladi tehnički saradnici, koji su listali ovaj program na štampaču, uzimali su malo od izvornog a malo od objektnog koda dok program nije izobličen do neupotrebljivosti.

Detaljna uputstva za pisanje programa, koja smo dali u računarima 3, malo su ublažila čitavu stvar, ali je prema graditeljima programatora eproma ipak ostao veliki dug, koji isunjavamo ovom prilikom. Pored integralnog programiranja tipa „čitav eprom odjednom", nova verzija programa omogućuje i parcijalno programiranje tipa „bajt po bajt". Ova verzija radi besprekorno više od devet meseci — sa njom je isprogramirano oko 2000 eproma tipa 2732 i najmanje po stotinu od svih ostalih tipova. Nijedan amaterski projekat, sa izuzetkom, razume se, računara „galaksija", nije podvrgnut tako rigoroznim testovima kao programator eproma.

Iako u komercijalnom softveru, naročito kod računara najnovije generacije, polako preovlađuje princip „pokaži prstom ono što hoćeš", prilikom razvoja programa uložen je poseban trud upravo da se izbegne sistem menija. Sa dosta razloga. Šanse da se, izborom opcije iz menija, unese pogrešna naredba ili pogrešan podatak znatno su veće nego u komandnom režimu, kada se oni ne biraju nego se unose svojeručno. Izvestan broj grešaka u radu sa programatorom eproma može zaista da bude „štetan po zdravlje".

Zato je „spektrumov" operativni sistem proširen specijalnim setom naredbi za rad sa programatorom eproma. Naredbe se poznaju po zvezdici ispred imena, unose se slovo po slovo i mogu, ali i ne moraju, da imaju određene parametre. Ako se parametri izostave, program ih dodeljuje sam —

EPROMER

Najpre otkucajte hex-loader i startujte ga a onda, red po red, unosite masinski program. ENTER pritiskatě tek na kraju svakoga reda tj. posto otkucate devet bajtova. Osam od njih ce biti upisani u memoriju dok je deveti zaduzen za proveru korektnosti kucanja.

CTRL code hex list:

```
25500: C3DA633A3A5CFE0BD9
25508: 2834FDCB017E2008CB
25516: 2AB25C23E5C3B712CC
25524: CD0313FD3600FF2A3F
25532: 595CCDA7112AB25C72
25540: 23E5C3B412FD3600C4
25548: 0B2A5D5C225F5C18E3
25556: D1225D5C18EF2A5D3A
25564: 5C2B7EFE2A20C32232
25572: 5D5CFD3600FFFD361E
25580: 2600060711BA682A90
25588: 5D5C23131A3C28107D
25596: 3DBE28F61A133C20A2
25604: FB3D28CD051B18E74C
25612: 7832045B225D5CFEE2
25620: 0228547EFE0D280433
25628: FE3A200CFD3600FF96
25636: FDCB017E205C18330E
2564d: CD7A1CF53A045BFEEF
25652: 053843F1FE2CC2C926
25660: 63E7CD821CFDCB007D
25668: 7ECAC963FE0D2809B0
25676: FD36000BFE3AC2C901
25684: 63FD3600FFFDCB015E
25692: 7E202A2AB25C23E508
25700: 21B712E5C3761BCDF0
25708: 821CFE41280B210738
25716: 5BCBBEC34164F11855
25724: C421075BCBFEE7C3BA
25732: 4164EFA03821075BEF
25740: CBB68A045BFE01CAE3
25748: FE64FE02286CFE03F7
25756: CAFE64F53E02CD012F
25764: 163E01111869CD0ABE
25772: 0CF13D113F6ACD0ACB
25780: 0CCD991E78B1CA64E7
25788: 65C53E0E32105B3E51
25796: 04CDB365E1220A5B51
2580d: CD991EC53E0C3210D5
25812: 5B3E03CDB365E12284
25820: 025B220E5B3A045B81
25828: FE04CA0C66CD991EC2
25836: C53E0A32105B3E02EA
2584d: CDB365E122005BC306
25852: 0C662AB25C23E5C375
25860: 761BCD991EC53E0A22
25868: B828063E69B8C2D5DC
25876: 63C5E106057DDD218F
25884: E168DDBE002808DDF1
25892: 230520F6C3D56378B1
25900: FE04200821075BCB78
25908: 7E28013D32065BC138
25916: F53E0632105BAFCD52
25924: B36521075BCB7EC4A8
25932: 7D68F13DCBC621E6AB
25940: 6816005F195E21EB60
25948: 681922085BC3FE642B
25956: 3A045BFE04281221F6
25964: 307522005BE5C13E06
25972: 0A32105B3E02CDB367
25980: 65210000220525B2227
25988: 0E5BE5C13E0C32109B
25996: 5B3E03CDB365DD2A88
26004: 085BDD6E07DD660800
26012: 220A5BE5C13E0E32AB
26020: 105B3E04CDB36521B3
26028: 075BCBF6C30C66C51D
26036: C5F53E02CD0116F1CF
```

```
10 REM CTRL CODE HEX LOADER

20 BORDER 1: PAPER 1: INK 7: P
OKE 23609,20
30 CLEAR 25499: POKE 23658,8:
PRINT AT 20,0
40 INPUT "START ADDR:";A
50 INPUT (A);": "; LINE A$: IF
A$="" THEN STOP
60 LET C=0: IF LEN A$<>18 THEN
GO TO 90
70 FOR I=1 TO 17 STEP 2: LET X
=16*CODE A$(I)-768-112*(A$(I)>"9
")+CODE A$(I+1)-48-7*(A$(I+1)>"9
"): IF I<17 THEN POKE A,X: LET A
=A+1: LET C=C+X
80 NEXT I: LET A=A-8: IF C-256
*INT (C/256)=X THEN PRINT A;": "
;A$: LET A=A+8: GO TO 50
90 PRINT #1; FLASH 1;"ERROR!":
BEEP 1,1: GO TO 50
```

```
26044: 111869CD0A0CC11E54
26052: 0521F0D80938161D62
26060: 2118FC09380F1D21C3
26068: 9CFF0938081D21F618
26076: FF0938011D3E20934F
26084: 21075BCB7E2012F5F3
26092: 3E16D73A105BD7F198
26100: D7C1CD2B2DCDE32D9A
26108: C9F53A045BFE02287F
26116: 03F118E3F13D18DF14
26124: 3A045BFE07CA1A67E9
26132: FE05286CCD9967CD31
26140: A567CDAF67CD8167A4
26148: CDBA67DD7E01D33F5C
26156: DB1FFEFFC2CB66CDB7
26164: 6567CD706720F1DD5E
26172: 7E00D33F3A045BFE27
26180: 04CADA66CD9967CDA8
26188: AA67CDAF67CDBA67E2
26196: DD7E02D33F7EFEFFEA
26204: CA5F67D31FDD7E03E0
26212: D33FCD1268F3C53E4F
26220: 150EDB0D20FD3D2085
26228: F8C1FBDD7E02D33F23
26236: CD6567CD706720D532
2624d: ED5B025BED530E5B4E
26252: 21075BCBAECD9967C9
26260: CDA567CDAF673A04FA
26268: 5BFE052003CD816736
26276: CDBA67DD7E01D33F5C
26284: DB1FBE2038CD6567A9
26292: CD7067CD756720F05D
26300: 110E6BCDD567CDF050
26308: 67CDFA67C3FE6411CB
26316: DC6ACDD567CDF06773
26324: CDFA67C3FE6411B91D
26332: 6ACDD567CDF067CD64
26340: FA67C3FE64E5210793
26348: 5BCB6E201DE11126E9
26356: 6BCDD567CD8F68CD05
26364: 8168CD3068CDE16763
26372: FE0D2809CDFA67C32D
26380: FE64E118EAE5210752
26388: 5BCBEEE11897CD990A
26396: 67CDA567CDAF67CDF0
26404: 8167CDBA67DD7E0132
26412: D33FAF320C5BDB1F54
26420: 773A0C5B86320C5B37
26428: CD6567CD706720EE4B
26436: DD7E00D33F3E1232EF
2644d: 105B06003A0C5B4F61
26452: 3E06CDB365CDFA6757
26460: C3FE64CD7567C37C0D
26468: 66DD7E06D35DDD7E52
26476: 05D35DC9230B78B155
26484: C9C5ED4B0E5B03ED1F
26492: 430E5BC1C9DD7E0091
26500: D33F11F46ACDD5678A
26508: 11646BCDD567CDE197
26516: 67CD8168C92A005B6B
26524: ED4B0A5BDD2A085B07
26532: C93E90D37FC93E8070
26540: D37FC9DD7E04D35DAA
26548: DD7E05D35DC9E5215F
26556: 075BCB76CCC567E17C
26564: C9ED5B025B7AB3C863
```

```
26572: CD65671B7AB320F8F9
26580: C9AF323C5C1AFE0B65
26588: C8D71318F876FDCB00
26596: 016E28F9FDCB01AE07
26604: 3A085CC9CD8F68012C
26612: 9600CD3D1FC9DD7EE3
26620: 00D33F11466BCDD576
26628: 6711646BCDD567CD1D
26636: E167CD8168C9E5C571
2664d: DDE53E1032105B3EEB
26652: 05ED4B0E5BC5CDB3EB
26660: 65C103ED430E5BDD9F
26668: E1C1E1C9E5C5DDE5B8
26676: E5E53E07111869CD6E
26684: 0A0CC1CDF665E17E5E
26692: CD5B683E20D7ED4BFD
26700: 0E5BCDF665DB1FCD58
26708: 5B68DDE1C1E1C9F5E1
26716: 3E20D7F1F5E6F01F10
26724: 1F1F1FC630FE3A38C3
26732: 02C607D7F1E60FC652
26740: 30FE3A3802C607D746
26748: C93E41D7C9E5C5DD6F
26756: E50604CD440EDDE1CC
26764: C1E1C9E5D5DDE506ED
26772: 0AC5210000116400 65
26780: E5CDB503011400 1190
26788: 6400E1C600ED4AE527
26796: C600ED52E138E6C1C5
26804: 10DFDDE1C1E1C94C64
26812: 4F4144FF5341564502
26820: FF564552494659FFD3
26828: 54455354FF4C495327
26836: 54FF545950 45FF50E4
26844: 4F4B45FF009CACACD2
26852: CCF8241B120900415F
26860: 81074784040500 0864
26868: 50905E1E84040500E9
26876: 1050905C1C8404 05F5
26884: 001070B03414840400
26892: 05002070B03414820F
26900: 02030040801603 06E4
26908: 50524F4752414D4159
26916: 54 4F522020455052 1C
26924: 4F4D410D0D0D5449A1
26932: 5020455052 4F4D4134
26940: 2E2E2E2E2E2E2E2E70
26948: 2E2E2E2E2E2E2E2E70
26956: 2EAE16080050524FEB
26964: 43454552412E2E10
26972: 2E2E2E2E2E2E2E2E70
26980: 2E2E2E2E2E2E2EAEF0
26988: 160A00504F4345549B
26996: 414B20552052414D01
27004: 552E2E2E2E2E2E2E97
27012: 2E2E2E2E2E2E2E2E70
27020: 2EAE160C00504F43E0
27028: 4554414B20552045FF
27036: 50524F4D552E2E2E1D
27044: 2E2E2E2E2E2E2E2E70
27052: 2E2E2EAE160E00 44A0
27060: 555A494E4120424C35
27068: 4F4B412E2E2E2E2EC1
27076: 2E2E2E2E2E2E2E2E70
27084: 2E2E2E2E2EAE1610BA
27092: 00544548554341 20DD
```

smatra da se radi o čitavom epromu i adresama 0 u epromu i 30000 u RAM-u. Program poznaje sledeće komande:

*** TYPE nnnn**

* TYPE 2732A

Naredba određuje tip eproma i, izuzetno, mora biti praćena jednim parametrom. Pošto spada u red opasnih naredbi, u program je ugrađena zaštita od pogrešnih parametara — on prihvata samo tipove 2716, 2732, 2732A, 2764 i 27128. Naredba se unosi pre svih ostalih, jer program na osnovu nje određuje sve svoje interne parametre — napon programiranja i sve kontrolne logičke nivoe. Kada se jednom unese, program smatra da radi sa određenim tipom sve dok se on ne promeni.

```
PROGRAMATOR EPROMA

TIP EPROMA.....2732
PROCEDURA.....KOPIRANJE U RAM
POCETAK U RAMU.....30000
POCETAK U EPROMU.....0
DUZINA BLOKA.....4096

ZBIR BAJTOVA.....62

IZVADI EPROM IZ PODNOZJA
I PRITISNI BILO KOJI TASTER
```

*** LOAD/*LOAD xxxx, nnn, mmmm**

LOAD/*LOAD 45000, 200, 1000

Kopira master eprom u RAM integralno (od adrese 0 u epromu na adresu 30000 u RAM-u) ili samo jedan blok (od adrese nnnn u epromu na adresu xxxx u RAM-u) čija je dužina određena parametrom mmmm. Uneta bez parametara, naredba *LOAD ima isto dejstvo kao i da je uneto *LOAD 30000, 0,4096. Preslikavanje mastera u RAM se završava porukom o zbiru bajitova u masteru. Zbir se izračunava u toku kopiranja po modulu 256. Kopiranje mastera u RAM, razume se, ne predstavlja uslov da bi se obavilo programiranje — u eprom se može preneti sadržaj sa bilo koje lokacije u memoriji računara između 0 i 65535.

*** SAVE/*SAVE xxxx, nnnn, mmmm**

* SAVE/*SAVE 45000, 200, 1000

Upisuje sadržaj RAM-a u eprom integralno (sa adrese 30000 u RAM-u na adresu 0 u epromu) ili samo jedan blok (sa adrese u RAM-u na adresu nnnn u epromu) čija je dužina određena parametrom mmmm. Uneta bez parametara, naredba *SAVE ima isto dejstvo kao i da je uneto *SAVE 30000, 0, 4096. Pre ulaska u proceduru programiranja, program vrši automatsku proveru da li je eprom prazan, a po završenom programiranju automatsku verifikaciju. Direktno programiranje, bez ovih verifikacija, programski je potpuno zabranjeno. Od programiranja ćelije u kojoj se već nalazi neki sadržaj može da bude samo štete. Programiranje je linearno — svaka lokacija se programira 50 ms. Preskaču se samo one lokacije u koje treba upisati FF, jer je to početni sadržaj svake prazne ćelije u epromu. Operacija programiranja se završava zvučnim upozorenjem i porukom

PROGRAMIRANJE USPELO

ili

PROGRAMIRANJE NIJE USPELO

*** TEST/*TEST nnnn, mmmm**

*** TEST/*TEST 1000, 2000**

Proverava da li je prazan čitav eprom ili samo jedan određeni segment (od adrese nnnn, do adrese nnnn + mmmm). Ova opcija je posebno pogodna za one koji često brišu svoje eprome. Operacija se, završava porukom

EPROM ISPRAVAN

ili

EPROM NIJE PRAZAN

*** VERIFY/*VERIFY xxxx, nnnn, mmmm**

* VERIFY/*VERIFY 45000, 200

Upoređuje sadržaj u memoriji računara (master) sa sadržajem u epromu koji se trenutno nalazi u programatoru. Ako u epromu pronađe lokaciju čiji se sadržaj ne slaže sa sadržajem odgovarajuće lokacije u RAM-u, program ispisuje, heksadecimalno, obe lokacije i njihove sadržaje i zaustavlja verifikaciju. Nastavak se obezbeđuje tasterom ENTER, a potpuni prekid i izlazak iz procedure provere pritiskom na bilo koji drugi taster.

Nove naredbe su gotovo savršeno uklopljene u operativni sistem i bejzik interpreter računara — „spektrum" ih ni po čemu ne razlikuje od svojih sopstvenih. Naredbe se mogu izdavati direktno sa tastature ili iz programa. Dozvoljeno je slobodno mešanje originalnih i novih naredbi u jednoj jedinoj komandnoj liniji ili velikom bejzik programu. To praktično, znači da se programator eproma može pomoći i svim potencijalima „spektrumovog" bejzika. Prilikom programiranja ROM-ova za računar „galaksija" obično je korišćen program poput ovog:

```
10 RONDOMIZE USR 25713
20 *TYPE 2732
30 *LOAD
40 FOR A =1 TO 20
50 *SAVE
60 NEXT A
```

Ovaj mali program kopira master eprom u memoriju računara počev od adrese 30000 i programira dvadeset kopija. Programer treba samo da menja eprome.

Za uvođenje novih naredbi iskorišćena je tehnika preseretanja rutine za obradu grešaka, na čiju adresu pokazuje promenljivi ERR-SP. Nakon inicijalizacije programa, na adresu na koju pokazuje ova promenljiva ubacuje se njegova sopstvena adresa. Pre nego što se vrati u bejzik interpreter računara, program svaki put obnavlja ovu adresu, i, zahvaljujući tome, ostaje stalno aktivan, tačnije sve dok se ne prede na upravljanje iz programa u bejziku. Pre poziva prve nove naredbe u programu mora obavezno da se nađe jedno RANDOMIZE USR 25713 bez obzira što je program jednom već inicijalizovan. To je relativno skromna cena za jednu veoma efektnu tehniku uvođenja novih naredbi.

Rad bez menija ima, međutim, i svojih nedostataka — programer nema uvek potpuni uvid u tok programa. Stoga je uveden statusni ekran (vidi sliku) iz koga programer u svakom trenutku može da vidi sa kojim tipom eproma radi, koji je master u računaru, koliki je njegov zbir bajtova, koja je operacija u toku, pa čak i koja se adresa trenutno programira. Nakon unošenja svih podataka, program zahteva od korisnika da postavi eprom u podnožje i, kao znak da je to učinio, pritisne bilo koji taster — poslednja prilika da se predomisli i ispravi eventualne greške.

Skladnim uklapanjem u interpreter bejzika i opcijama za parcijalno programiranje ovaj program pretvara programator eproma iz sprave za kopiranje (tuđeg truda) u spravu za razvoj sopstvenog sistemskog softvera.

Jova Regasek

```
27100: 414452455341 2E2E0C
27108: 2E2E2E2E2E2E2E2E70
27116: 2E2E2E2E2E2E2EAEF0
27124: 1612005A424952207F
27132: 42414A544F56412E35
27140: 2E2E2E2E2E2E2E2E70
27148: 2E2E2E2E2E2E2E2E70
27156: 2EAE1614071200 4E6D
27164: 454953505241564E68
27172: 41204144524553 4111
27180: 2020202020202020 00
27188: 2020202020202020 A080
27196: 1615078016080F4524
27204: 4449544F56414E4A5F
27212: 4520455052 4F4DC1A9
27220: 801608114C495354EB
27228: 414E4A4520455052 25
27236: 4F4DC1160811250522F
27244: 4F5645524120455032
27252: 524F4DC116080B5028
27260: 524F5645524120503F
27268: 524F4752414D495263
27276: 414E4AC116080C5014
27284: 524F4752414D495263
27292: 414E4A4520455052 25
27300: 4F4DC1160811 4B4F26
27308: 504952414E4A452029
27316: 55205241CD16140605
27324: 4550524F4D20495 33F
27332: 505241564 14E0B16E9
27340: 090950524F475241DD
27348: 4D4952414E4A450B11
27356: 161406120145505 22A
27364: 4F4D204E494A452002
27372: 5052415A414E210BF8
27380: 161404535441564 9B5
27388: 204550524F4D205518
27396: 20504F444E4F5A4A4D
27404: 450B161407505 24F72
27412: 4752414D495241 4E51
27420: 4A4520555350454C38
27428: 4F0B161403120150EA
27436: 524F4752414D495263
27444: 414E4A4520 4E494A1F
27452: 45205553504 54C4F3D
27460: 210B161404120049B5
27468: 5A5641444920455033
27476: 524F4D204 95A205021
27484: 4F444E4F5A4A410B20
27492: 161502492050524981
27500: 5449534E4920424932
27508: 4C4F204B4F4A492008
27516: 5441535441 5520BC19F
27524: 219B6322B25C363EC3
27532: 2BF9219C63E5ED7389
27540: 3D5C216A5CCBDECDF6
27548: 6B0DC3FE6400000009D
```

Korisni zbirovi

eprom za „galaksiju"
ROM 1 verzija 28 50
ROM 1 verzija 29 223
ili 222
ROM 2 62
generator znakova 105

Lična karta

Početak 25500
Dužina 2053
Poziv 27523
Učitavanje
LOAD CODE

33) programatat eproma

„Spektrum"

SPRAJTOVI

Sigurno vam je već dosadilo da svaki „komodorovac" kojeg sretnete na ulici sruši ugled vašem miljeniku onom poznatom „Ali komodor ima sprajtove!". No, ne očajavajte, jer ovaj program ispravlja taj „spektrumov" nedostatak. Ukucajte ga, otipkajte RUN i, kad „spektrum" kaže OK, sprajtovi čekaju na vas. Program možete spremiti sa SAVE „sprajtovi" CODE 64032,284.

Program omogućava korištenje osam sprajtova, obilježenih brojevima 0–7 i veličine 16×16 točkica. Najveći prioritet ima sprajt 0, a najmanji sprajt broj 7 — pri preklapanju sprajtova, sprajt obilježen malim brojem bit će nacrtan preko sprajta obilježenog većim.

Upravljanje svakim sprajtom postiže se, kao i kod „komodora", pokiranjem na četiri posebne memorijske lokacije. Za sprajt 0 to su lokacije 64000, 64001, 64002 i 64003, za sprajt 1 lokacije od 64004 do 64007 itd. do sprajta 7, čije su lokacije od 64028 do 64031. Prva od te četiri lokacije indicira je li sprajt uključen (u tom slučaju mora sadržavati broj 1) ili isključen (kada sadrži broj 0). Druga lokacija sadrži X, a treća Y koordinatu gornjeg lijevog kuta sprajta. Četvrta lokacija sadrži atribut boje za sprajt.

Da biste definirali sprajt, prvo ga nacrtajte na milimetarskom papiru, u veličini 16 sa 16 mm. Zatim ga podijelite u dvije kolone široke 8 mm. Prvu kolonu ukucajte u grafičke karaktere A i B, a drugu u C i D. Na lokaciju 64316 unesite broj sprajta koji definirate i otipkajte RANDOMIZE USR 64240. Sad je taj sprajt spreman za upotrebu, a sadržaj grafičkih karaktera više nije važan.

Za iscrtavanja svih uključenih sprajtova dovoljno je otipkati RANDOMIZE USR 64032. No, ne zaboravite za svaki sprajt unijeti odgovarajuće podatke u njegove 4 lokacije.

Evo jednog malog ali korisnog savjeta: uz rub sprajta ostavite bar 1 red točkica razmaka kako sprajt ne bi prilikom pomicanja ostavljao trag.

S obzirom da upravljanje sprajtovima pomoću naredbe POKE nije baš elegantno, evo zadatka za prave hakere: obogatite „spektrumov" bejzik naredbom za upravljanje sprajtovima.

Igor Fišer

```
  10 FOR i=64032 TO 64315
  20 READ a: POKE i,a
  30 NEXT i
1000 DATA 62,7,50,60,251,38,250,
7
1010 DATA 7,111,126,61,250,232,2
50,35
1020 DATA 78,35,70,120,254,176,2
10,232
1030 DATA 250,63,31,55,31,183,31
,168
1040 DATA 230,248,168,87,121,254
,240 210
1050 DATA 232,250,197,7,7,7,168,
230
1060 DATA 199,168,7,7,95,35,126,
8
1070 DATA 121,230,7,245,38,0,15,
15
1080 DATA 15,111,15,79,68,9,229,
58
1090 DATA 80,251,103,46,0,77,31,
203
1100 DATA 25,71,9,1,0,238,9,193
1110 DATA 9,193,4,62,255,183,31,
16
1120 DATA 252,23,79,6,16,213,121
,47
1130 DATA 79,26,161,182,18,35,28
,126
1140 DATA 18,35,28,121,47,79,26,
161
1150 DATA 182,18,35,209,20,122,2
30,7
1160 DATA 194,175,250,123,198,32
,95,230
1170 DATA 224,40,4,122,214,8,87,
16
1180 DATA 212,193,121,230,7,95,1
21,7
1190 DATA 7,7,168,230,199,168,7,
7
1200 DATA 111,120,7,7,230,3,246,
88
1210 DATA 103,8,87,14,3,120,230,
7
1220 DATA 194,212,250,13,65,114,
35,114
1230 DATA 35,175,179,40,1,114,12
5,198
1240 DATA 30,111,124,206,0,103,1
6,237
1250 DATA 58,60,251,61,242,34,25
0,201
1260 DATA 33,120,255,6,16,54,0,3
5
1270 DATA 16,251,58,60,251,230,7
,46
1280 DATA 0,93,103,31,203,27,87,
25
1290 DATA 17,0,238,25,221,33,88,
255
1300 DATA 6,16,221,126,0,119,35,
221
1310 DATA 126,16,119,35,221,126,
32,119
1320 DATA 35,221,35,16,237,84,93
,1
1330 DATA 48,0,237,66,1,80,1,8
1340 DATA 126,31,16,35,19,8,11,1
20
1350 DATA 177,32,244,201
```

„Spektrum"

SCREEN (Y,X)

„Spektrumova" funkcija SCREEN(Y,X) ima jedan prilično ružan bag: ne radi za UDG i blok grafičke karaktere. Program beta bejzik (VI.8) istina, otklanja taj bag, ali ni to rešenje nije savršeno: blok grafički karakteri i dalje bivaju ignorisani. Stvarno potpuno rešenje predstavlja ovaj kratak mašinski program smešten u baferu za štampač. Program se izuzetno lako koristi. U bejzik variable X i Y stave se koordinate polja na ekranu koje treba ispitati. Zatim, USR 23300 daje kod znaka koji se tu nalazi. Jedan fleg, na adresi 23540, ukazuje na to da li je prepoznati znak u normalnom (0) ili „inverse" (1) obliku. Prilikom upotrebe ovog programa treba obratiti pažnju na nekoliko sitnica. Znak koji nije prepoznat dobiće kôd 0. Zatim, grafički karakter sa kodom 128 (prazan kvadratić) neće biti prepoznat kao takav, već kao razmak (32), što je uostalom i sasvim logično. Takođe, grafički znak 143 (puni kvadratić) biće prepoznat kao inverzni razmak, dakle kôd 32 i fleg postavljen na jedinicu. Ovaj program ima još jednu caku: moguće je ispitivati i donji deo ekrana. Naime, Y koordinata može da ide od 0 do 23.

Ako je ovu rutinu potrebno pozvati iz nekog mašinskog programa, onda je ulazna tačka na 23325. U registru B treba da se nalazi horizontalna koordinata (0 do 31), a u registru C vertikalna kordinata (od 0 do 23). Na izlazu, kôd znaka se nalazi u registru C. Flag se, naravno, i dalje nalazi na 23540.

Kada otkucate program, snimite ga na kasetu sa SAVE „SCREEN" CODE 23300,245. Prilikom učitavanja, nije potrebno nikakvo spuštanje RAMTOP-a. Treba samo paziti da se tokom upotrebe programa ne pozovu slučajno naredbe COPY, LPRINT ili LLIST koje brišu bafer za štampač.

Vladimir Kostić

```
SCREEN$(Y,X)
CTRL CODE HEX LIST

23300: 21F55BCDD65BFE208D
23308: 3802CF01F521F75B72
23316: CDD65BFE1830F3C1F8
23324: 4F2A365C1100011936
23332: 3E60CD8B5B385C2A0F
23340: 7B5C3E15CD8B5B300D
23348: 04C625184E11001076
23356: 3E1832AA5BAFCB4B52
23364: 28023EF0CB43280290
23372: C60F676F22EA5B2234
23380: EC5BAFCB5B28023E84
23388: F0CB532802C60F6774
23396: 6F22EE5B22F05B2168
23404: EA5B3E01CD8B5B3067
23412: 057BC68018061C1515
23420: 20C33E00F53E2032A6
23428: AA5BF106004FC9C5D9
23436: D5F5AF32F45B790F82
23444: 0F0FE6E0A85F79E64A
23452: 18EE4057F147C5D56F
23460: E51AAE280B3C201E5A
23468: 3DF53E0132F45BF1E3
23476: 4F060714231AAEA904
23484: 200C10F7C1C1C13EB4
23492: 809037D1C1C9E11194
23500: 080019D1C110CFAF41
23508: 18F1ED5B5D5CD52201
23516: 5D5CCDFB24D1ED53B6
23524: 5D5CCD1423C9000086
23532: 0000000000000000000
23540: 00580D590D000000CB
```

3. STACK_A CALL &2D28

Potprogram STACK_A će preneti na računski stek broj sadržan u akumulatoru. Izlaznih parametara nema, a registri nisu očuvani.

4. STACK_BC CALL &2D2B

Ova rutina, za razliku od prethodne, prenosi na računski stek broj sadržan u BC. Inače, i STACK_A sama prvo izvrši **LD C,A** i **LD B,&00**, pa dolazi na STACK_BC. Izlaz je tako zajednički — registri nisu očuvani.

5. STACK_NUM CALL &33B4

Potprogram STACK_NUM donosi na računski stek proizvoljan broj iz memorije, prikazan u uobičajenom zapisu sa pet bajtova. Na ulazu HL sadrži adresu prvog bajta.

U toku potprograma akumulator je očuvan, BC na izlazu uvek sadrži nulu, a HL će se pomeriti iza broja koji je bio prenet. DE sadrži adresu STKEND.

Najčešće se STACK_NUM koristi kada se na stek šalju elementi nekog numeričkog niza. Čim se nađe adresa zadatog elementa(pomoću STK_VAR), izvrši se **INC HL** i odmah zatim STACK_NUM. Na izlazu će HL pokazivati sledeći element niza, spreman za novo STACK_NUM.

6. STK_VAR CALL &2996

O ovoj rutini smo već govorili prilikom traženja adrese elemenata numeričkog niza. Međutim, isti potprogram se koristi i kada na računski stek treba poslati parametre nekog alfanumeričkog niza.

Na ulazu, već smo to rekli, STK_VAR koristi izlazne parametre iz LOOK_VARS. Ako je pri tome promenljiva bila znakovnog tipa, njeni elementi će se ubaciti na stek, nikakvih izlaznih parametara neće biti i registri ne ostaju očuvani.

7. STK_STORE CALL &2AB6
STK_ST_0 CALL &2AB1

Potprogram STK_STORE šalje na računski stek pet bajtova smeštenih redom u registre A, E, D, C i B. Na izlazu su registri očuvani, a HL sadrži adresu STKEND.

Najčešće se na ovaj način prenose parametri nizova. DE treba da sadrži adresu, a BC dužinu niza. Možete se pozvati i SKT_ST_0 koji će automatski staviti nulu u akumulator. Nama taj prvi bajt ionako nije važan.

FP_TO_A CALL &2DD5
FIND_INTI CALL &1E94

Poslednji broj sa računskog steka može se preneti u registar BC pozivom rutine FP_TO_BC. Ulaznih parametara nema. Na izlazu BC sadrži traženi broj, a njegov predznak je određen indikatorom nule „Z": ako je Z=0, broj je negativan. Poslednja vrednost sa računskog steka biva automatski obrisana, HL pokazuje ispred nove poslednje vrednosti, a DE iza nje (STKEND). Akumulator sadrži isto što i registar C.

Ukoliko broj ne može da se smesti u 16 bita, na izlazu će biti setovan indikator prenosa „C". Brojevi koji nisu celi se zaokružuju.

Pozivom FIND_INT2 postići ćemo to da se u slučaju negativnih brojeva ili brojeva većih od 65535, automatski poziva prekid "B Integer out of range".

9. FP_TO_A CALL&2DD5
FIND_INT1 CALL&1E94

Potprogram FP_TO_A uzima sa računskog steka poslednji rezultat i smešta ga u akumulator. Ulaznih parametara nema, a izlaz je potpuno isti kao da smo pozvali FP_TO_BC, sa jedinom razlikom što je indikator prenosa „c" setovan u slučaju da broj premašuje 8 bita.

Ako se pozove FIND_INT1, onda će negativni brojevi i brojevi veći od 255 izazvati prekid „B Integer out of range".

10. STK_FETCH CALL &2BF1

Rutina STK_FETCH uzima sa računskog steka poslednjih pet bajtova i smešta ih redom u registre, A, E, D, C i B. Ulaznih parametara nema, a na izlazu je sa steka obrisana poslednja vrednost i HL sadrži novu adresu STKEND.

Obično se na ovaj način uzimaju parametri niza: adresa DE i dužina BC.

struktura memorije i potprogrami iz ROM-a

sve ‚spektrumove' rutine

Jovan Skuljan

Verovatno nema programera koji se, pre ili kasnije, ne zasiti bejzika. Dojade, konačno, mamutski listinzi i programi koji se, iznad svega, izvršavaju katastrofalno sporo. Naravno, postoje i situacije kada je bejzik sasvim prikladan i kada od njega ne vredi bežati bez razloga, ali to još uvek ne znači da ne treba potražiti bolja i elegantnija rešenja.

U ovom tekstu pokušaćemo da detaljnije upoznamo računar „spektrum", odgovarajući na pitanja kako on radi i šta se s njim može postići ako se napusti bejzik.

Pošli smo od pretpostavke da naši čitaoci već imaju kakvu-takvu predstavu o mašinskom jeziku, tako da im praćenje ovog teksta neće biti naročito teško. Ako i bude nekih problema, ne treba se zbog toga obeshrabriti: već u sledećem broju „Računari" donose svoju školu mašinskog programiranja.

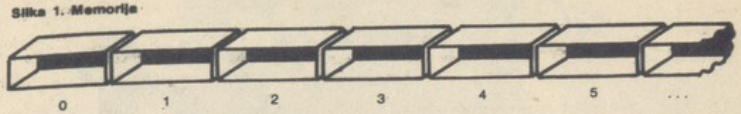
Struktura memorije

Memoriju računara najjednostavnije je zamisliti kao dugačak niz pregradaka, ili ćelija, koje su u stanju da upamte ono što se u njih stavi. Jedna takva ćelija zove se **bajt.** Veličina memorije kojom računar raspolaže potpuno je određena brojem bajtova.

Jedan kilobajt (1K) sadrži 1024 bajta, a jedan megabajt (1M) 1024 kilobajta. Malo je neobično što se umesto „okruglog" broja 1000 koristi 1024, ali računar uopšte i ne radi u dekadnom brojnom sistemu. Uostalom, ako se prikaže heksadekadno ili binarno, broj 1024 je sasvim okrugao.

Svaki bajt ima neku oznaku, pomoću koje se može lako pronaći među drugima. Ta oznaka zove se **adresa,** a u stvari je ceo broj u rasponu od 0 do 65535. Prvi bajt memorije ima adresu nula, drugi ima adresu jedan, itd.

Slika 1. Memorija



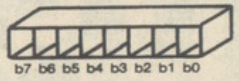
Češće ćemo adrese i druge konstante označavati heksadekadno, i u tom slučaju ispred njih će stajati oznaka „&" (CHR$ 38).

Prema tome, adresa može biti bilo koji broj između &0000 i &FFFF.

Svaki bajt za sebe sastoji se iz osam elementarnih ćelija, koje zovemo **bitovima.** Jedan bit može da upamti samo jednu od dve osnovne informacije: ili je bit **resetovan** (sadrži nulu), ili je **setovan** (sadrži jedinicu). Tada se čitav bajt može shvatiti kao binarni broj sa osam cifara.

Bitovi u okviru jednog bajta označavaju se indeksima između 0 i 7, onako kako raste odgovarajuća težina cifarskog mesta. Bit 0 je bit najmanje težine.

Slika 2. Bajt



Mikroprocesor može da pronađe bilo koji bajt memorije i da pročita bilo koji bit u njemu. To je, čak, izvodljivo i iz bejzika, korišćenjem funkcije **PEEK.** Baš kao što **SQR n** daje koren broja „n", tako **PEEK n** daje sadržaj bajta na adresi „n". Na primer, **PRINT PEEK 0** će dati rezultat 243, što znači da prvi bajt memorije sadrži vrednost 243, odnosno BIN 11110011, ili heksadekadno &F3. Nijedan bajt ne može sadržati više od BIN 11111111, odnosno 255 dekadno.

Memorija je podeljena u dva velika bloka. Prvih 16K, počev od adrese 0, pa zaključno sa 16383, čini tzv. ROM (Read Only Memory). To je memorija iz koje se može samo čitati, ali u koju se ništa ne može upisivati. U ROM-u su sadržane sve instrukcije potrebne za uspešan rad računara i ta informacija, naravno, ostaje očuvana čak i kada se računar isključi.

Ostatak memorije, počev od adrese 16384 (heksadekadno &4000), naziva se RAM (Random Access Memory). Pristup tim bajtovima je potpuno slobodan. U njih u bilo kom trenutku možemo upisati bilo koji sadržaj, ali kada se računar isključi, svi podaci iz RAM-a bivaju nepovratno izgubljeni.

Naredba **POKE n, b** iz bejzika upisuje vrednost „b" u bajt na adresi „n". Međutim, odmah treba reći da ovakvi eksperimenti nisu preporučljivi, ukoliko nemamo jasnu predstavu o tome šta zapravo radimo. Jer, nije nikakav problem ubaciti neki broj u neki bajt, ali treba znati da i „spektrum" koristi RAM za sopstvene potrebe. Ako je on, na primer, na adresu 23607 upisao broj 60, ne možemo sada i mi tamo upisati bilo šta, jer vrlo lako može doći do nesporazuma.

Potrebno je, dakle, poznavati strukturu čitave memorije, odnosno tzv. **memorijske mape.** Tada će nam biti jasno šta znači broj 60 na adresi 23607, ili broj 255 na adresi 13610, ili bilo koji drugi bajt u RAM-u i ROM-u.

BROJEVI U POKRETNOM ZAREZU

Jedan bajt dovoljan je za prikazivanje celih brojeva između 0 i 255. Dva bajta, odnosno šesnaest bita, mogu služiti za smeštanje celih brojeva sve do 65535. Na primer, broj 3000 bi

Proizvoljna funkcija f(x), gde „x" po modulu **nije veće od jedinice,** može se razviti u red Čebiševljevih polinoma:

$f(x)=c_0T_0+2c_1T_1+2c_2T_2+2c_3T_3+\ldots,\ |x|\leq 1$

gde su c_0, c_1, c_2, itd. koeficijenti razvoja, unapred poznati za svaku funkciju.

Tačnost izračunavanja zavisi od broja članova reda. „Spektrum", na primer, računa funkciju SIN sa šest, funkciju EXP sa osam, a LN sa čak dvanaest članova. Pri tome je uvek argument podesnom smenom doveden u interval između −1 i +1.

Ako se aproksimacija funkcije vrši redom od „n" članova, onda se operacija „series" poziva kodom **&80+n.** Recimo, za red od 8 članova, treba izvršiti naredbu &88.

Iza samog koda operacije mora se navesti svih „n" koeficijenata, počevši od c_{n-1}, pa zaključno sa c_0. Za svaki koeficijent se prvo navodi eksponent, a zatim mantisa, na potpuno isti način kao u slučaju naredbe „stk-data". Podrazumeva se, naravno, da je pre poziva generatora reda argument „x" smešten na stek. Na povratku će on biti zamenjen sa f(x).

h) „Spektrumov" kalkulator raspolaže sa šest **memorijskih lokacija,** označenih sa „mem-0", mem-1", itd. do „mem-5", u kojima se privremeno mogu čuvati međurezultati u toku računanja.

Naredba tipa „st-mem" će preneti u odabranu memorijsku lokaciju poslednji broj sa računskog steka, mada se sam stek pri tome ne briše. Slično tome, podatak iz bilo koje memorije može se dovesti na stek operacijom tipa „get-mem". Kodovi za „st-mem" su &C0+n, a za „get-mem" su &E0+n, gde je „n" oznaka memorijske lokacije, između 0 i 5.

B. U tabeli računskih operacija namerno smo ispustili neke kodove, kao što su recimo &09 — &0E, ili &11 — &16. Postoje naredbe koje odgovaraju i tim kodovima, ali se, na žalost, ne mogu koristiti na način koji smo opisali. Očigledno se radi o previdu koji je, eto, prošao neopaženo kroz sve testove i sada ukrašava ROM vašeg računara. Sve te dodatne operacije, naime, očekuju da u procesorskom B registru zateknu svoj kôd! Pošto se to normalno ne dešava, moramo sami da prvo prekinemo kalkulator (operacija „end-calc"), zatim da izvršimo **LD B, nn** i onda ponovo startujemo kalkulator, da bismo izvršili operaciju „nn"... Bez tog manevra, operacija se neće izvršiti korektno.

Na sreću, ove ekscentrične naredbe nam uglavnom nisu potrebne, jer obavljaju obično poređenje dva broja ili dva niza, što se uvek može svesti na „less-0" i „greater-0". Ali, tu spada i jedna dosta važna operacija: „val". Možda se pitate kako onda „spektrum" u bejziku uspešno računa funkciju VAL? Na to nije teško odgovoriti: „spektrum" pri izračunavanju bilo kakvih izraza koristi isključivo operaciju „fp-calc-2", tako da je registar B uvek pripremljen.

Dajemo i tabelu preostalih operacija.

DODATNE OPERACIJE KALKULATORA

x , y	&09	no-l-eql	u=(x<=y)		u
x , y	&0A	no-gr-eq	u=(x>=y)		u
x , y	&0B	nos-neql	u=(x<>y)	poredjenje	u
x , y	&0C	no-grtr	u=(x>y)	brojeva	u
x , y	&0D	no-less	u=(x<y)		u
x , y	&0E	nos-eql	u=(x=y)		u
x$, y$	&11	str-l-eql	u=(x$<=y$)		u
x$, y$	&12	str-gr-eq	u=(x$>=y$)		u
x$, y$	&13	strs-neql	u=(x$<>y$)	poredjenje	u
x$, y$	&14	str-grtr	u=(x$>y$)	nizova	u
x$, y$	&15	str-less	u=(x$<y$)		u
x$, y$	&16	strs-eql	u=(x$=y$)		u
x$	&18	val$	u$=VAL$ x$, vrednovanje niza		u$
x$	&1D	val	u=VAL x$, vrednovanje niza		u

Jedina operacij koju do sada nismo pomenuli je „e-to-fp", sa kodom &3C, ali se ona uopšte i ne može koristiti u okviru kalkulatora, već se mora pozvati kao potprogram. Ta operacija inače prevodi brojeve iz oblika „x E m" u pokretni zarez, što nama za sada nije interesantno.

```
2. HL=HL*DE                                   CALL &30A9
   GET_HL*DE                                  CALL &2AF4
```

Potprogram HL=HL*DE obavlja množenje dva šesnaestobitna broja koji su na ulazu smešteni u HL i DE. Rezultat na izlazu je smešten u HL, registar DE je očuvan, a BC se ne koristi. Indikator prenosa „c" će biti setovan u slučaju prekoračenja opsega.

Potprogram GET_HL*DE radi isti posao, ali samo pod uslovom da je u toku izvršenje bejzik programa. U fazi sintaksne provere, povratak se vrši odmah. Osim toga, ako u toku množenja dođe do prekoračenja, biće automatski pozvan prekid „4 Out of memory".

d) Pomoću naredbe „stk-data" može se na stek dovesti **bilo koja konstanta,** direktno navodeći svaki bajt njenog binarnog zapisa. Tačnije, obavezno je navesti eksponent, a od bajtova mantise može se dati jedan, dva, tri ili svih četiri, pri čemu će svi bajtovi koji nedostaju automatski biti zamenjeni nulama.

Informaciju o tome koliko će bajtova za mantisu biti dato, nosi prvi bajt iza naredbe „stk-data", i to bitovi 7 i 6, sa svojim kombinacijama: 00, 01, 10 i 11. Bitovi od 0 do 5 služe za smeštanje eksponenta broja koji se šalje na stek. Ako je taj prostor nedovoljan, bitovi 0 do 5 se svi postavljaju na nulu, a eksponent dodaje posebno kao sledeći bajt. U oba slučaja, bilo da je eksponent stao u prvi bajt, bilo da je dodat kao samostalni bajt, moraju se navesti još i bajtovi mantise u očekivanom broju.

Da bi sve to bilo jasnije, daćemo konkretan primer dovođenja na stek broja 43.217. Odgovarajući binarni zapis je: &86, &2C, &DE, &35, &40, što se jednostavno može dobiti i u bejziku, stavljajući prvo LET A=43.217, a zatim čitajući varijablu A bajt po bajt.

Procedura pripreme bajtova koji će slediti iza naredbe „stk-data" je sledeća:

1) Redukujemo eksponent (prvi bajt binarnog zapisa), oduzimajući od njega &50:
redukovani eksponent=stvarni eksponent−&50
U našem slučaju, eksponent iznosi &86, pa posle redukcije imamo &36.

2) Odabiramo koliko bajtova za mantisu ćemo navesti. Taj broj neka bude „n". Kod nas je n=4.

3) Ako je redukovani eksponent manji od &40, onda se on može smestiti u šest bita u okviru prvog bajta iza „stk-data". Tada prvi bajt iznosi:
b=(n−1)*64+ redukovani eksponent
Ali, ako je redukovani eksponent jednak &40 ili veći, onda se dodaje kao posebni bajt, dok prvi bajt iznosi samo:
b=(n−1)*64

4) Konačno se daje „n" bajtova za mantisu.

U našem primeru, redukovani eksponent je manji od &40, pa je:
b=3*64+&36=&F6
Konačno, konstanta 43.217 se dovodi na stek kao što je prikazano u primeru 35.

```
35.   DEFB &34              ; stk-data
      DEFB &F6, &2C, &DE
      DEFB &35, &40
```

e) Operacija „truncate" nalazi celobrojni deo broja, tj. ono što se nalazi levo od decimalne tačke. To se poklapa sa rezultatom funkcije INT samo ako je broj pozitivan.

f) Operacija „fp-calc-2" je zapravo **zamena za bilo koju drugu,** pod uslovom da se u B registru nalazi stvarni kod operacije koju treba izvršiti. Tako je, recimo, svejedno da li ćemo postupiti prema primeru 36. ili 37.

```
36.   RST &28              ; FP_CALC
      DEFB &04             ; multiply
      DEFB &38             ; end-calc

37.   LD B,&04             ; priprema za „multiply"
      RST &28              ; FP_CALC
      DEFB &3B             ; fp-calc-2, konkretno: multiply
      DEFB &38             ; end-calc
```

Upotreba operacije „fp-calc-2" je naročito značajna kada se u programu ne zna unapred koju operaciju treba izvršiti, na primer ako se izračunava neki izraz koji je korisnik upravo ukucao. Tada se potreban kod za operaciju ubacuje u B i izvršava se „fp-calc-2".

g) Operacije tipa „series" su veoma značajne, jer omogućuju izračunavanje funkcija pomoću redova. U tu svrhu se, međutim, ne koriste obični stepeni redovi, već redovi **Čebiševljevih polinoma,** što obezbeđuje bržu konvergenciju i veću tačnost. Ako vas matematika previše ne interesuje, možete preskočiti ovaj deo. Ni mi nećemo mnogo ulaziti u detalje, ali bismo želeli da vam barem damo ideju kako da sami koristite generator redova.

Polinomi Čebiševa se obično označavaju sa $T_n(x)$, gde je „x" promenljiva, a „n" stepen polinoma (n=0, 1, 2, 3 . . .). Prvih nekoliko polinoma su:

$T_0(x)=1$
$T_1(x)=x$
$T_2(x)=2x^2-1$
$T_3(x)=4x^3-3x$
•
•
•

Svaki polinom se može izračunati na osnovu prethodna dva, po rekurentnoj formuli:
$T_n(x)=2xT_{n-1}(x)-T_{n-2}(x), \qquad (n=2, 3, 4,. . .)$

se prikazao sa BIN 0000101110111000. Prvi bajt (bajt veće težine) sadrži jedanaest, odnosno BIN 00001011, a drugi 184 (BIN 10111000), tako da je:
11×256+184=3000

Mikroprocesor Z80, međutim, upisuje 16-bitne brojeve u memoriju obrnutim redom: **prvo bajt manje težine!** Dakle, u slučaju broja 3000, prvi bajt će sadržati 184, a drugi 11. To moramo stalno imati u vidu ako iz memorije želimo da pročitamo neki 16-bitni broj. Na primer, dva bajta na adresama 23635 i 23636 uvek sadrže adresu na kojoj počinje bejzik program u memoriji. Da bismo pročitali tu adresu, moramo izvršiti **PRINT PEEK 23635+256×PEEK 23636.**

Daleko veći problem nastaje kada treba zapisati neki **decimalan broj.** „Spektrum", kao i većina drugih računara, koristi tehniku tzv. **pokretnog zareza,** i sve brojeve prikazuje u eksponencijalnom obliku **±mx2↑e,** gde je „m" mantisa (uvek manja od jedinice, a veća ili jednaka 0.5), a „e" je eksponent u rasponu između −127 i +127. Nije teško proveriti da ovakav način zapisivanja pokriva sve realne brojeve po modulu manje od oko 10↑38. Pri tome, svaki broj čiji je modul manji od oko 10↑−39 tretira se kao nula.

Uzećemo kao primer broj 43.217, koji se može prikazati kao:
43.217=0.675265625×2↑6

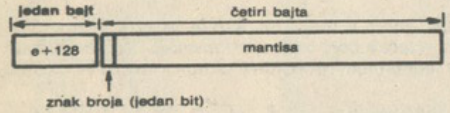
Mantisu 0.675265625 dalje ćemo prevesti u binarni oblik. U binarnom brojnom sistemu takođe postoji decimalna tačka (možda bolje binarna tačka!), s tim što cifarska mesta iza tačke vrede redom: 1/2, 1/4, 1/8, itd., umesto dekadnih 1/10, 1/100, itd. Tako će, recimo, dekadni broj 0.5 biti zapisan sa 0.1 binarno. Mi se sada nećemo upuštati u detalje oko prevođenja brojeva iz jednog sistema u drugi. Prosto ćemo dati konačni zapis mantise, zaokružen na 32 cifre iza tačke:
m=0.10101100110111100001101010100000

Ne izgleda baš pregledno, ali računar jedino ume da pamti jedinice i nule, pa mu ni ovo neće predstavljati nikakav problem.

„Spektrum" uvek odvaja **pet bajtova** za zapis brojeva u pokretnom zarezu. U prvi bajt se upisuje eksponent „e" uvećan za 128, a u preostala četiri bajta (32 bita) smešta se mantisa, tačnije samo cifre iza decimalne tačke.

Pošto je usvojeno da mantisa bude uvek veća ili jednaka 1/2, onda je prvi bit iza decimalne tačke jedinica! Umesto te jedinice, koja se podrazumeva, u taj bit se upisuje znak broja: nula za „+", a jedan za „−".

Slika 3. Zapis broja u pokretnom zarezu



Naš broj 43.217 je pozitivan, pa će prvi bit mantise biti nula. Eksponent sabran sa 128 daće 134 (&86), pa tako konačno imamo zapis od pet bajtova: &86, &2C, &DE, &35, &40

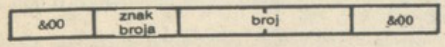
Sve ovo nije teško proveriti, ako u bejziku izvršimo, na primer **LET a=43.217,** a onda pomoću funkcije **PEEK** pročitamo svaki bajt promenljive „a". Treba samo znati gde se u memoriji nalaze bejzik promenljive, a o tome ćemo govoriti kasnije.

Što se tiče celih brojeva između −65535 i +65535, za njih „spektrum" takođe rezerviše pet bajtova, ali je zapis drugačiji:

- prvi bajt uvek je nula, što znači da sledi ceo broj;
- drugi bajt sadrži nulu za pozitivne, a 255 za negativne brojeve;
- treći i četvrti bajt sadrže sam broj (prvo bajt manje težine); ali, ako je broj negativan, dodaje mu se 65536, i
- peti bajt je uvek nula.

Zapis celih brojeva prikazan je na slici 3a, a kao primer možemo uzeti broj −1: &00, &FF, &FF, &FF, &00.

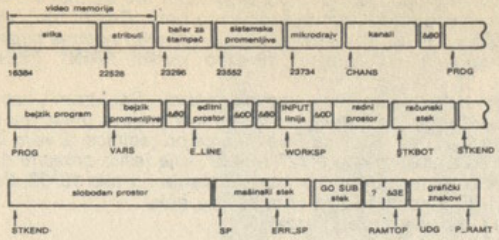
Slika 3a. Zapis celog broja



Međutim, prilikom obavljanja aritmetičkih operacija, i celi brojevi se, po potrebi, prevode u potpuni oblik pokretnog zareza. Zapis nule je pri tome jedinstven: &00, &00, &00, &00, &00.

Memorijska mapa

Slika 4. Mapa memorije



RAM (sl. 4) je podeljen na više zasebnih blokova, čiju funkciju ćemo detaljno objasniti. Neki blokovi imaju fiksnu početnu adresu, ali većina ipak ne, pošto u toku rada dolazi do stalnog pomeranja i preraspodele prostora. Recimo, unošenjem nove programske linije, svi blokovi desno od adrese PROG biće pomereni ka višim adresama, smanjujući tako slobodan prostor iza STKEND. Sama adresa PROG, s druge strane, zavisi od veličine bloka koji koristi mikrodrajv, a takođe i od sadržaja kanalskih informacija. Ukoliko, međutim, nemamo mikrodrajv, i ne diramo kanale, CHANS će uvek biti na 23734, a PROG na 23755.

„Spektrum" u svakom trenutku zna gde počinje koji blok, jer se sve adrese čuvaju u posebnom prostoru tzv. **sistemskih varijabli**, o čemu ćemo još govoriti.

VIDEO MEMORIJA

1. „Spektrumov" ekran sastoji se od 49152 tačke: 256 po horizontali i 192 po vertikali, a sva slova, cifre i ostali znakovi ispisuju se u poljima veličine 8×8. Takvih polja ima ukupno 32×24. Svaka tačka u okviru jednog polja može imati samo jednu od dve boje: ili će to biti boja podloge (PAPER), ili boja teksta (INK), dok polje za sebe može imati i pojačan sjaj (BRIGHT), a može i treptati, tj. naizmenično menjati boju podloge i teksta (FLASH). Sve te informacije o slici zapisane su u prvih 6912 bajtova RAM-a, odakle ih „spektrum" čita i šalje na TV prijemnik.

Tačka na ekranu predstavljena je **jednim bitom** u memoriji. Ako je tačka boje podloge, odgovarajući bit je resetovan (nula), a ako je tačka boje teksta, odgovarajući bit je setovan (jedinica). Osam tačaka, prema tome, čine jedan bajt. Tih bajtova ukupno ima 6144 i rađaju se počev od adrese 16384.

Jednom bajtu odgovara na ekranu horizontalna crtica dužine osam tačaka. Ako odgovarajući bajt sadrži BIN 11111111, crtica je neprekidna, jer sve tačke imaju boju teksta.

Prvi bajt video memorije predstavlja crticu u gornjem levom uglu ekrana, što jednostavno možemo proveriti sa **POKE 16384,255.** Međutim, raspored ostalih bajtova nije baš uvek onakav kakav bismo očekivali. Najbolje je umesto bilo kakvog objašnjenja izvršiti kratak bejzik program, koji redom popunjava video bajtove u memoriji, i onda pratiti šta se dešava na ekranu. Stvar će se ubrzati izbacivanjem pauze sa linije 30. Druga pauza, na liniji 50, samo služi da spreči ispisivanje raporta, jer bit to inače odmah izbrisalo dva donja reda.

```
1. 10 FOR n=16384 TO 22527        10 FOR n=22528 TO 23295
   20 POKE n, 255                 20 POKE n, 0
   30 PAUSE 10                    30 PAUSE 10
   40 NEXT n                      40 NEXT n
   50 PAUSE 0                     50 PAUSE 0
```

Imajući sve u vidu, izgleda da je vrlo komplikovano pronaći u memoriji pravi bajt, a zatim i odgovarajući bit u okviru tog bajta, na osnovu koordinata „x" i „y" neke tačke. Na sreću, oko toga ne moramo mnogo brinuti, jer u ROM-u postoji jedan veoma elegantan mašinski potprogram koji obavlja taj zapetljan proračun. Nije, međutim, loša vežba napraviti bejzik potprogram koji simulira naredbu **PLOT** isključivo korišćenjem **POKE.**

2. Preostalih 768 bajtova video memorije, počev od 22528, nose informaciju o boji za svako znakovno polje. To su tzv. **atributi.**

Od osam bita, sa koliko svaki atribut raspolaže, najniža tri rezervisana su za boju teksta, između INK 0 i INK 7. Sledeća tri nose boju podloge (PAPER 0 — PAPER 7), bit 6 sadrži informaciju o sjaju (BRIGHT 0 ili BRIGHT 1), a bit 7 informaciju o treptanju (FLASH 0 ili FLASH 1). Šematski prikaz dat je na slici 5. Redosled atributa u memoriji odgovara redosledu polja na ekranu, onako kako se inače ispisuju.

a) Operacije „jump-true", „jump" i „dec-jr-nz" koriste se za grananje u procesu računanja. Iza svake od tih naredbi, tj. iza odgovarajućeg koda &00, &33, ili &35, mora se navesti još jedan bajt, koji daje **relativno rastojanje odredišta** na koje će se skočiti, slično mašinskoj naredbi JR

Operacija „jump-true" je uslovni skok, koji se izvršava samo ako je poslednji rezultat na steku bio logički istinit, tj. različit od nule. U protivnom se skok ne vrši, već se izvršava sledeća operacija. U oba slučaja obavlja se **brisanje poslednjeg rezultata** sa steka, kao da je izvršena operacija „delete". Kao primer navodimo situaciju u kojoj se testira broj „x": ako je X>0, skok se vrši na kraj računanja, a ako nije, obavlja se pre izlaska još nekoliko operacija, da bi krajnji rezultat bio x*PI:

```
32.   RST &28      ; FP_CALC          x
      DEFB &31     ; duplicate        x, x
      DEFB &37     ; greater-0        x, (0 ili 1)
      DEFB &00     ; jump-true        x
      DEFB &05     ; to END           x
      DEFB &A3     ; stk-pi/2         x, PI/2
      DEFB &31     ; duplicate        x, PI/2, PI/2
      DEFB &0F     ; addition         x, PI
      DEFB &04     ; multimply        x*PI
END   DEFB &38     ; end-calc         x ili x*PI
```

U ovom primeru, relativna vrednost skoka iznosi &05, a to je, u stvari, razlika između adrese odredišta i adrese samog bajta za vrednost skoka. Nema nikakve potrebe da tu razliku računamo, već sve treba prepustiti asembleru. Tako bismo, umesto DEFB &05, ukucali:

```
DEFB  END — $
```

Pri tome je sasvim svejedno da li se skok obavlja unapred, ili unazad.

Naredba „jump" radi slično kao i „jump-true", s tom razlikom što se skok na zadato odredište obavlja bezuslovno, a računski stek ostaje neizmenjen.

Konačno, operacija „dec-jr-nz" služi za formiranje programskog ciklusa u okviru procesa računanja, uz kontrolu broja ponavljanja. Da bi se ova naredba uopšte mogla koristiti, pre poziva kalukatora se u registar B mora ubaciti broj ciklusa koji će biti izvršeni u toku računanja. Taj broj će, čim se izvrši instrukcija RST &28, automatski biti prenet u sistemsku varijablu BREG i odatle će se kasnije koristiti.

Kada računar naiđe na naredbu „dec-jr-nz", on umanjuje BREG za jedinicu i obavlja skok na naznačeno odredište samo pod uslovom da brojač još nije stigao do nule — potpuno slično mašinskoj naredbi DJNZ.

Primer koji dajemo predstavlja ciklus sa ukupno sedam prolaza, pri čemu se u svakom prolazu broj „n" na steku sabira sa jedinicom. Na ulazu je n=1, a na izlazu n=8.

```
33.   LD B,7          ; sedam ciklusa
      RST &28         ; FP_CALC
      DEFB &A1        ; stk-one          1
LOOP  DEFB &A1        ; stk-one          n, 1
      DEFB &0F        ; addition         n=n+1
      DEFB &35        ; dec-jr-nz        n
      DEFB LOOP — $   ; to LOOP          n
      DEFB &38        ; end-calc         8
```

Unutar ovakvih ciklusa **ne smeju se naći** operacije koje i same koriste BREG za svoje potrebe (generator redova i sve funkcije koje se računaju preko redova).

b) Operacija „x-mod-y" ne postoji u bejziku, ali može biti veoma korisna. Ona obavlja deljenje dva broja, a u rezultatu daje istovremeno i ceo deo količnika v=INT(x/y) i ostatak u=x−y*v.

c) Operacije: „stk-zero", „stk-one", „stk-half", „stk-pi/2" i „stk-ten" služe da bi se na stek dovele neke često korišćene konstante: 0, 1, 0.5, PI/2 i 10. Ponekad se na osnovu tih konstanti mogu izvesti i druge, kao recimo PI:

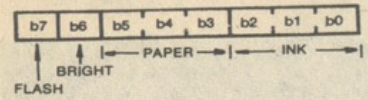
```
34.   DEFB &A3      ; stk-pi/2         PI/2
      DEFB &31      ; duplicate        PI/2, PI/2
      DEFB &0F      ; addition         PI
```

tabela 2:

OPERACIJE KALKULATORA

x	&00	jump-true	skok ako je "x" istinito	-
x , y	&01	exchange	zamena mesta	y , x
x	&02	delete	brisanje sa steka	-
x , y	&03	subtract	u=x-y, oduzimanje	u
x , y	&04	multiply	u=x*y, množenje	u
x , y	&05	division	u=x/y, deljenje	u
x , y	&06	to-power	u=x↑y, stepenovanje	u
x , y	&07	or	u=x OR y, broj OR broj	u
x , y	&08	no-and-no	u=x AND y, broj AND broj	u
x , y	&0F	addition	u=x+y, sabiranje	u
x$, y	&10	str-and-no	u$=x$ AND y, niz AND broj	u$
x$, y$	&17	strs-add	u$=x$+y$, sabiranje nizova	u$
x$	&19	usr-$	u=USR x$, adresa grafičkih znakova	u
x	&1A	read-in	čitanje podatka "u$" sa struje "x"	u$
x	&1B	negate	u=-x, promena znaka	u
x$	&1C	code	u=CODE x$, ASCII kodiranje	u
x$	&1E	len	u=LEN x$, dužina niza	u
x	&1F	sin	u=SIN x, sinus	u
x	&20	cos	u=COS x, kosinus	u
x	&21	tan	u=TAN x, tangens	u
x	&22	asn	u=ASN x, arkus sinus	u
x	&23	acs	u=ACS x, arkus kosinus	u
x	&24	atn	u=ATN x, arkus tangens	u
x	&25	ln	u=LN x, prirodni logaritam	u
x	&26	exp	u=EXP x, eksponencijalna funkcija	u
x	&27	int	u=INT x, najmanji ceo broj	u
x	&28	sqr	u=SQR x, kvadratni koren	u
x	&29	sgn	u=SGN x, znak broja	u
x	&2A	abs	u=ABS x, apsolutna vrednost	u
x	&2B	peek	u=PEEK x, sadržaj adrese "x"	u
x	&2C	in	u=IN x, čitanje porta "x"	u
x	&2D	usr-no	u=USR x, poziv mašinskog programa	u
x	&2E	str$	u$=STR$ x, prevod broja u niz	u$
x	&2F	chr$	u$=CHR$ x, ASCII dekodovanje	u$
x	&30	not	u=NOT x, logička negacija	u
x	&31	duplicate	udvajanje poslednjeg broja	x , x
x , y	&32	x-mod-y	u=x MOD y, v=x DIV y	u , v
x	&33	jump	bezuslovni skok	x
x	&34	stk-data	dovodjenje podatka "u" na stek	x , u
x	&35	dec-jr-nz	kontrola ciklusa (kao DJNZ)	x
x	&36	less-0	u=x<0, poredjenje sa nulom	u
x	&37	greater-0	u=x>0, poredjenje sa nulom	u
x	&38	end-calc	završetak računanja	x
x	&39	get-argt	priprema argumenta za SIN i COS	u
x	&3A	truncate	celobrojni deo	u
x , y	&3B	fp-calc-2	izvršenje operacije sa kodom u B	?
x	&3D	re-stack	prevod broja u pokretni zarez	x
x	&81	series-01		u
.	.	.	generator redova	.
.	.	.	Čebiševljevih polinoma	.
x	&9F	series-1F		u
x	&A0	stk-zero		x , 0
x	&A1	stk-one		x , 1
x	&A2	stk-half	dovodjenje konstante na stek	x , 1/2
x	&A3	stk-pi/2		x , PI/2
x	&A4	stk-ten		x , 10
x	&C0	st-mem-0		x
.	.	.	smeštanje broja	.
.	.	.	u memoriju	.
x	&C5	st-mem-5		x
x	&E0	get-mem-0		x , u
.	.	.	uzimanje broja	.
.	.	.	iz memorije	.
x	&E5	get-mem-5		x , u

Slika 5. Atribut



Bilo kakva promena atributa uopšte neće uticati na adržaj slike, već jedino na boju. Recimo, možemo napisati na ekranu neki tekst, a onda ubaciti u sve atribute vrednost BIN 00111111, pomoću naredbe **POKE**, naravno. Time će ekran postati potpuno prazan (FLASH 0, BRIGHT 0, PAPER 7, INK 7). Međutim, ako vratimo atribute na BIN 00111000, slika će se ponovo pojaviti bez ikakvih izmena (INK 0).

BAFER ZA ŠTAMPAČ

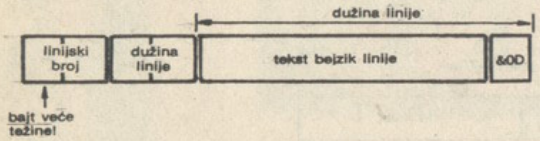
Blok od 256 bajtova, sa početkom na 23296, naziva se bafer za štampač. Tu se formira tekst koji se štampa, i tek onda se šalje na printer. Inače, u drugim prilikama možemo ove bajtove koristiti i za svoje potrebe.

BEJZIK PROGRAM

Sve linije bejzik programa čuvaju se u memoriji jedna iza druge, onim redom kojim se pojavljuju i kada ih izlistamo na ekranu. Prva dva bajta u zapisu svake linije sadrže linijski broj, pri čemu prvo dolazi bajt **veće težine!** To je jedini slučaj takvog prikazivanja brojeva na „spektrumu" i treba ga dobro upamtiti.

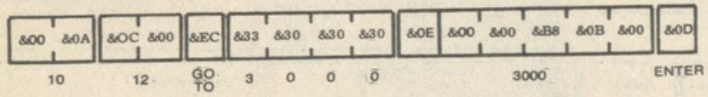
Iza linijskog broja slede dva bajta koja sadrže dužinu linije (tekst+ENTER), u normalnom poretku: prvo bajt manje težine. Konačno, dolazi tekst same linije, označen na kraju sa ENTER (&0D), kao što je prikazano na slici 6.

Slika 6. Bejzik linija



Tekst se sastoji od niza naredbi i drugih znakova, pri čemu jednoj naredbi, ili znaku, odgovara jedan bajt, sa odgovarajućim ASCII kodom. Iza svake numeričke konstante, koja se u liniji pojavljuje, umetnut je bajt &0E, praćen sa pet bajtova binarnog zapisa konstante u pokretnom zarezu. Na slici 7. je prikazan kompletan zapis bejzik linije 10 GO TO 3000, što obuhvata ukupno 16 bajtova.

Slika 7. Primer bejzik linije



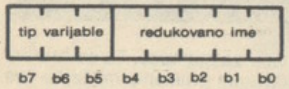
Šest bajtova se ubacuje i u liniju sa naredbom DEF FN iza svakog argumenta navedenog unutar zagrade. Prvi bajt je opet &0E, a ostalih pet je samo rezervisano za stvarnu vrednost argumenata, koji će tu biti upisan u fazi izračunavanja same funkcije. Na slici 8. dajemo primer linije 20 DEF FN p(x, g$)=x+LEN g$.

Slika 8. Primer bejzik linije

BEJZIK PROMENLJIVE

Promenljive se smeštaju u blok memorije počev od adrese VARS, onim redom kojim su i definisane. „Spektrum" raspoznaje šest različitih tipova varijabli. Odgovarajući tip sadržan je u prvom bajtu zapisa promenljive (bitovi 7, 6 i 5), dok preostali bitovi 0–4 sadrže ime promenljive umanjeno za &60.

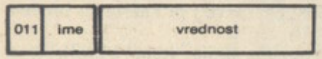
Slika 9. Prvi bajt varijable



Redukovano ime uvek je u opsegu između &01 i &1A, što odgovara stvarnom imenu između „a" i „z" (&61–&7A). Ako ime sadrži više od jednog slova, redukuje se samo prvo.

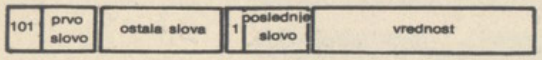
1. Numerička varijabla čije je ime jedno slovo, prikazuje se sa šest bajtova. Prvi nosi oznaku tipa 011 i redukovano ime, a ostalih pet sačinjaćvaju vrednost same promenljive.

Slika 10. Numerička promenljiva



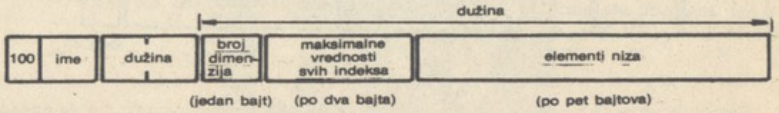
2. Numerička varijabla sa dužim imenom ima tip 101. Prvo slovo imena se redukuje i ubacuje u prvi bajt zapisa, ostala slova se navode redom bez ikakvih izmena, a poslednje slovo je obavezno dato sa setovanim sedmim bitom (stvarni kôd je sabran sa 128). Iza imena dolaze pet bajtova za sam broj.

Slika 11. Numerička promenljiva sa dužim imenom



3. Numerički niz ima tip 100, a ime je uvek jedno slovo. Dva bajta rezervisana su za dužinu cele varijable, zatim sledi jedan bajt koji sadrži ukupan broj dimenzija, a onda redom dolaze maksimalne vrednosti svih indeksa, po dva bajta svaki. Konačno imamo elemente niza od po pet bajtova, takvim redosledom da se najbrže menja poslednji, a najsporije prvi indeks. Dakle, prvo dolaze elementi koji imaju zajednički prvi indeks, a od njih prvo oni koji imaju zajednički drugi indeks, itd.

Slika 12. Numerički niz



4. Dimenzionisan znakovni niz (tip 110) sasvim je sličan numeričkom nizu, samo što elementi zauzimaju po jedan bajt.

KALKULATOR

Već smo ranije rekli da, pre bilo kakvog računanja, argumenti moraju biti poslati na računski stek. Taj posao, svakako, obavljaju posebni potprogrami iz ROM-a.

Pretpostavimo da je u početku stek prazan (to znači da je STKBOT=STKEND), a mi želimo da tamo ubacimo broj 85. Sav posao se sastoji u tome što broj 85 treba smestiti u akumulator, a onda pozvati rutinu koja šalje akumulator na stek. U rezultatu toga, imaćemo na računskom steku pet bajtova, počev od STKBOT, u koje je upisan broj 85. STKEND pokazuje odmah iza tih pet bajtova.

Dalje želimo na stek da pošaljemo broj 19. Opet ćemo pripremiti akumulator i pozvati pomenuti potprogram. Broj 19 će doći odmah iza (ili iznad) prethodnog broja 85, a ceo stek sada ima deset bajtova. Svaki sledeći broj obavezno dolazi iznad ostalih.

„Spektrum" uvek obavlja operacije sa brojevima u samom vrhu računskog steka. Ako mu naredimo da obavi operaciju „kosinus", on će, u stvari, izračunati kosinus poslednjeg broja na računskom steku. Sam taj broj će odmah biti uklonjen sa steka, a umesto njega će doći rezultat izvršene operacije. Potpuno isto rade i funkcije: SIN, TAN, LN, EXP, ABS, INT, itd. Sve su to primeri tzv. „unarnih" operacija. One deluju na poslednji broj na steku i zamenjuju ga rezultatom.

Postoje i tzv. „binarne" operacije. Deluju na dva broja, da kažemo „x" i „y", pri čemu je „y" u vrhu steka, a „x" odmah pod njim. Po završenoj operaciji, oba broja nestaju sa steka, a ostaje samo rezultat. Primer binarnih operacija su: sabiranje, oduzimanje, množenje, deljenje, stepenovanje, itd.

To je za sada sve što se tiče brojeva i operacija sa njima. Međutim, „spektrum" ume da radi i sa nizovima slova i znakova. Razume se, i nizovi sa kojima se obavljaju neke operacije, moraju doći na računski stek. Pri tome, da se stvari ne bi previše komplikovale, svaki niz na steku takođe zauzima pet bajtova, kao da je u pitanju broj. U tih pet bajtova, međutim, nije zapisan sam niz, već parametri koji ga potpuno određuju. Niz se, inače, nalazi negde u memoriji (obično u radnom prostoru), a na steku se čuva njegova adresa (drugi i treći bajt) i dužina (četvrti i peti bajt). Prvi bajt koristi samo naredba LET, a nama uopšte nije važan. Obično ga stavljamo na nulu. Što se tiče prenošenja parametara niza na stek, o tome opet ne treba brinuti: postoje i tu posebni potprogrami. Treba samo upamtiti da „spektrum" uopšte ne pravi razliku između brojeva i nizova na steku. Recimo, bez obzira na to šta zaista sadrže poslednjih pet bajtova na steku, podjednako uspešno će se izvršiti i operacija **COS** i operacija **LEN**, jer će svaka od njih tretirani onih pet bajtova onako kako joj odgovara.

1. FP_CALC — RST &28

A. Potprogram FP_CALC poziva se kada treba obaviti bilo kakvo računanje. Ulaznih parametara nema, a podrazumeva se da su prethodno svi brojevi sa kojima se računa preneti na računski stek u očekivanom redosledu.

Iza naredbe **RST &28** mora doći niz bajtova koji definišu računske operacije, onim redom kojim će se izvršavati. Svaka operacija ima svoj kôd i predstavljena je jednim bajtom. Na kraju tog niza obavezno dolazi bajt &38, kao znak da je računanje završeno.

Izlaznih parametara nema, registri nisu očuvani, a rezultat računanja ostaće na računskom steku kao poslednja vrednost.

Primer 31. prikazuje niz aritmetičkih operacija.

```
31.   RST  &28        ; FP_CALC
      DEFB &1F        ; sinus
      DEFB &0F        ; sabiranje
      DEFB &27        ; INT
      DEFB &04        ; množenje
      DEFB &0F        ; sabiranje
      DEFB &38        ; kraj
```

Daćemo tabelu svih operacija ugrađenih u „spektrumov" kalkulator. Većina njih je neposredno prisutna i u bejziku, dok su druge dostupne samo u mašinskom jeziku.

Svaka operacija (tabela 2.) ima svoj heksadekadni kod i skraćeni naziv. Zatim sledi kratko objašnjenje same operacije i, konačno, stanje računskog steka po njenom izvršenju. Prva kolona tabele prikazuje računski stek na ulazu u operaciju.

Objašnjenje rada nekih specijalnih operacija dato je i posebno, van tabele.

poslednjeg bajta te linije (ENTER) nalaziće se u registru BC. Ali, ako je pretraživanje počelo iz sredine neke linije i pri tome je naredba E otkrivena u istoj toj liniji, onda varijabla NEWPPC i registar BC na izlazu iz LOOK_PROG imaju nepromenjenu vrednost sa ulaza.

U svim slučajevima, slično kao kod EACH_STMT, akumulator na izlazu sadrži karakter sa adrese HL.

Kao primer ćemo napraviti program koji pronalazi sve **GO TO** naredbe, da bi ih na neki način obradio (recimo da se radi o prenumeraciji). Samu tu „obradu" označili smo uopšteno sa MODIFY, a podrazumeva se da registar HL pri tome ostaje očuvan.

Pretraživanje počinje od prve bejzik linije (PROG), a registar E sadrži potreban kôd &EC. Poziva se LOOK_PROG i odmah obavlja povratak ako je traganje završeno (RET C). EACH_STMT se koristi da bi se prešlo pred sledeću naredbu:

```
29.   LD        HL, (PROG)
      DEC       HL
LOOP  LD        DE, &00EC
      CALL      LOOK_PROG
      RET       C
      CALL      MODIFY
      DEC       HL
      LD        DE, &0200
      CALL      EACH_STMT
      JR        LOOP
```

14. LOOK_VARS — CALL &28B2

Potprogram LOOK_VARS pronalazi zadatu bejzik promenljivu u memoriji, ili nalazi adresu odgovarajućeg argumenta u DEF FN naredbi, ukoliko je upravo u toku izračunavanje neke DEF FN funkcije.

Ulaznih parametara nema, ali sistemska varijabla CH_ADD mora sadržati adresu na kojoj počinje tekst sa imenom promenljive. Obično se to ime nalazi u okviru nekog aritmetičkog izraza u bejzik programu, ali mi možemo formirati isti tekst i bilo gde na drugom mestu. Jedino moramo voditi računa o tome da u slučaju dimenzionisanih nizova, iza svakog indeksa **mora stajati** odgovarajući binarni zapis u pokretnom zarezu (oznaka &0E, plus pet bajtova za sam broj). Ukoliko tekst nema smisla, tj. ne predstavlja ime bilo kakve promenljive, usledиće prekid sa greškom „C Nonsense in BASIC".

Na izlazu iz rutine LOOK_VARS, indikator prenosa „C" je setovan ukoliko varijabla nije pronađena. Tada HL sadrži adresu imena varijable (kao CH_ADD na ulazu), a indikator nule „Z" setovan je samo ako se tražila promenljiva tipa dimenzionisanog niza.

Ukoliko je na izlazu C=0, to znači da je promenljiva pronađena i tada HL sadrži odgovarajuću adresu. Ako se ime promenljive sastoji iz više slova, HL će pokazivati poslednji bajt imena. Dakle, u bilo kom slučaju, naredbom INC HL se prelazi iza imena varijable. Indikator nule „Z" je resetovan samo ako je promenljiva običnog numeričkog tipa, uključujući i FOR_NEXT. Za bilo koju vrstu niza je Z=1.

Ostali registri nisu očuvani.

Primećujemo da u slučaju dimenzionisanih nizova, na izlazu ne dobijamo adresu traženog elementa, već adresu niza kao celine. Izdvajanje naznačenog elementa moguće je primenom rutine STK_VAR.

15. STK_VAR — CALL &2996

Potprogram STK_VAR može da obavlja dva potpuno različita posla. Prvi je, kao što smo već napomenuli, pronalaženje pojedinačnih elemenata numeričkog niza, a drugi je: smeštanje na računski stek parametara nekog alfanumeričkog niza. O ovoj drugoj funkciji ćemo posebno govoriti u okviru kalkulatora, a sada se samo zadržavamo na prvoj.

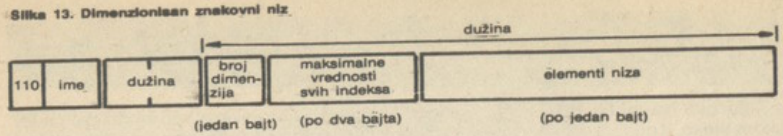
Ulazni parametri u STK_VAR su isti oni koje na izlazu daje LOOK_VARS. Zapravo, STK_VAR se uvek i zove neposredno iza LOOK_VARS, ali samo pod uslovom C=0 (varijabla je pronađena) i Z=1 (varijabla je tipa niza:

```
30.   CALL      LOOK_VARS
      JP        C, REPORT_2
      CALL      Z, STK_VAR
```

Sa REPORT_2 smo označili adresu na kojoj se poziva raport „2 Variable not found", a skok se obavlja samo ako je C=1. Za C=0 i Z=1, poziva se STK_VAR.

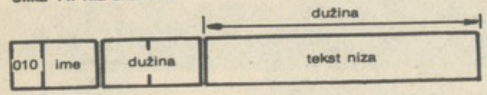
Na izlazu će HL sadržati adresu neposredno ispred zadatog elementa numeričkog niza, a BC će pokazivati bajt ispred prvog elementa niza. Dakle, posle INC HL, pomeričemo se na prvi od pet bajtova elementa niza koji smo tražili. DE na izlazu uvek sadrži &0005, a akumulator je nula.

Slika 13. Dimenzionisan znakovni niz



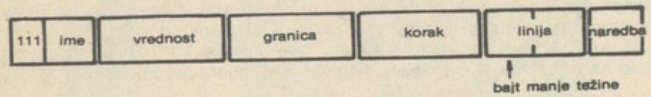
5. Običan znakovni niz (string) ima tip 010. Dva bajta sadrže dužinu niza, odnosno broj znakova. Ukoliko je niz prazan, onda je dužina nula, i teksta nema. Inače, sledi tekst sa svim znakovima.

Slika 14. Niz znakova



6. Kontrolna varijabla za FOR–NEXT petlje (tip 111) zauzima ukupno 19 bajtova. Po pet bajtova je rezervisano za trenutnu vrednost promenljive, graničnu vrednost u petlji i korak sa kojim se promenljiva menja. Od tri poslednja bajta, prva dva sadrže linijski broj petlje, odnosno linije sa naredbom FOR, pri čemu prvo ide bajt manje težine! Poslednji bajt je redni broj naredbe u liniji na kojoj počinje petlja, a to je uvek za jedan veće od rednog broja naredbe FOR.

Slika 15. FOR-NEXT varijabla



EDITNI PROSTOR

Iza bloka bejzik varijabli nalazi se jedan graničnik u vidu bajta &80, koji računaru samo govori da promenljivih više nema. Iza tog bajta počinje tzv. editni prostor, koji služi za prihvatanje programske linije koja se trenutno ukucava, ili koja se poziva pomoću komande **EDIT.**

Dok kucamo liniju, ili je prepravljamo, ona je u editnom prostoru zapisana prosto kao niz znakova, onako kako je vidimo u dnu ekrana (nema binarnog zapisa linijskog broja, niti bilo kakvih umetnutih bajtova). Tek kada pritisnemo **ENTER,** računar će proveriti sintaksu, dovesti liniju na format koji smo ranije opisali, i preneti je u blok sa bejzik programom ili je neposredno iz editnog prostora izvršiti, ako se radi o direktnoj naredbi.

Poslednji bajt u editnom prostoru je uvek **ENTER,** a za njim sledi graničnik &80.

RADNI PROSTOR

Na adresi WORKSP počinje radni prostor računara. Prvi deo je rezervisan za liniju koja se unosi u toku izvršenja naredbe **INPUT,** a drugi, iza oznake &0D, koristi se u različitim situacijama, kada privremeno treba smestiti neke podatke radi njihove obrade. Tu, recimo, dolazi zaglavlje programa koji se uzima sa kasete, ili nizovi znakova sa kojima se vrše računske operacije. Za vreme naredbe **MERGE,** čitav program sa kasete će biti unet u radni prostor, da bi odatle bio prenet na svoje pravo mesto, kombinujući se sa već postojećim programom u bejzik bloku.

Radni prostor nema fiksnu veličinu, već se stvara po potrebi. Kada se ne koristi, sadrži

samo bajt &0D. U ROM-u, kao što ćemo kasnije videti, postoje gotovi potprogrami za stvaranje i uklanjanje radnog prostora, što i sami možemo koristiti.

RAČUNSKI STEK

Sve računske operacije „spektrum" isključivo obavlja sa brojevima koji se nalaze na tzv. računskom steku. To je blok memorija čiji početak (ili dno) i kraj (odnosno vrh) su određeni adresama STKBOT i STKEND. Brojevi na steku su poređani jedan iza drugog (ili jedan iznad drugog) i svaki zauzima pet bajtova.

Postoje posebne rutine u ROM-u koje smeštaju brojeve na računski stek, ili ih odande uzimaju. Svaki nov broj dolazi na sam vrh steka, zatrpavajući ostale. I mi, ako želimo da nešto računamo u mašinskom jeziku, moramo svoje brojeve poslagati na računski stek, o čemu ćemo detaljno govoriti kasnije.

SLOBODAN PROSTOR

Iznad računskog steka, počevši od adrese STKEND, prostire se slobodan prostor, odnosno neiskorišćeni bajtovi, koji čine preostalu memoriju dostupnu bejziku. Kada se bilo koji od već pomenutih memorijskih blokova poveća, to ide na račun smanjenja slobodnog bloka. Međutim, čim slobodnog prostora nema dovoljno, računar prekida rad i daje izveštaj „4 Out of memory".

MAŠINSKI STEK

Prvo na šta nailazimo iza slobodnog prostora jeste mašinski stek procesora Z80. Tu su smeštene adrese za povratak iz mašinskih potprograma, a privremeno se čuvaju i razni drugi podaci. Za razliku od računskog steka, mašinski raste **naniže,** ka manjim adresama, pa zato prvo nailazimo na vrh steka, koji je inače određen sadržajem procesorskog registra SP.

Dva bajta u samom dnu mašinskog steka sadrže adresu &1303, što je u stvari rutina za štampanje raporta iz ROM-a. Bajt manje težine je označen adresom ERR_SP, kojom će inače biti napunjen registar SP u slučaju bilo kakve greške (efekat brisanja mašinskog steka). O tome ćemo kasnije još govoriti.

Ispod mašinskog steka (po adresama iznad) je prostor koji koristi bejzik naredba **GO SUB,** pa se i zove GO SUB stek. Pri svakom pozivu potprograma iz bejzika, na stek se postavljaju tri bajta: linijski broj (dva bajta) i redni broj naredbe na koju će se izvršiti povratak (to je odmah iza **GO SUB**). Naredba **RETURN** će prosto skinuti sa steka poslednja tri bajta i na osnovu njih obaviti povratak iz potprograma. Međutim, ako pre toga nije izvršena ni jedna **GO SUB** naredba, i stek je prazan, imaćemo grešku „7 RETURN without GO SUB".

Kako uopšte računar može znati da li je GO SUB stek prazan? Jednostavno, u samom dnu steka se nalaze dva granična bajta. Prvi nije određen, a drugi obavezno sadrži &3E. Ako naredba **RETURN** pokupi sa steka ova dva bajta umesto linijskog broja, to će odmah biti jasno, jer ni jedan linijski broj ne može imati bajt veće težine jednak &3E!

Adresa bajta &3E označava se sa RAMTOP i to je poslednja adresa RAM-a dostupna bejziku. Sve što je iza toga, bezbedno je čak toliko da će biti pošteđeno i od naredbe NEW.

Obično se RAMTOP nalazi blizu stvarnog kraja RAM-a (P_RAMT), tako da bezbedni deo memorije zauzima samo 168 bajtova, predviđenih za grafičke znakove (21 znak po osam bajtova). Otuda se odmah iza RAMTOP-a nalazi adresa UDG (User Defined Graphics).

RAMTOP možemo postaviti i sami na neku nižu adresu. Na primer, posle **CLEAR 30000,** RAMTOP će biti na 30000, a čitav prostor počev od 30001 osloboden je za naše potrebe.

KANALI

Komunikacija sa nekim periferijskim uređajem, bilo da je u pitanju tastatura, štampač, ili TV ekran, obavlja se preko tzv. **kanala** i **struja.** Na primer, da li će pri štampanju nekog teksta informacija otići na gornji deo ekrana, odnosno tamo gde normalno radi naredba

10. NUMERIC — CALL &2D1B

Pomoću ove rutine ispituje se da li neki karakter predstavlja cifru dekadnog sistema. Na ulazu akumulator sadrži ispitivani karakter, na izlazu su svi registri, uključujući i A očuvani, a indikator prenosa „C" je resetovan samo ako se radi o cifri.

11. ALPHANUM — CALL &2C88

Potprogram ALPHANUM na ulazu ima neki karakter u akumulatoru, a na izlazu će indikator prenosa „C" biti setovan samo ako se radi o slovu ili cifri. Svi registri, uključujući i A, su sačuvani.

12. EACH_STMT — CALL &198B

Potprogram EACH_STMT služi za nalaženje adrese proizvoljne naredbe u okviru jedne bejzik linije. Na ulazu HL sadrži polaznu adresu za pretraživanje. To će, ako se pretražuje cela linija, biti adresa neposredno ispred prve naredbe (nikako ne adresa linijskog broja!) ili, ako se polazi iz sredine linije, biće to adresa znaka „:". Dakle, u bilo kom slučaju, HL na ulazu treba postaviti neposredno ispred neke naredbe.

Naredba koja se traži može biti zadata na dva načina: ili ćemo u E registar ubaciti odgovarajući kôd, ili u D registar odgovarajući redni broj, meren počev od naredbe ispred koje je i postavljen HL. Da ne bi bilo nesporazuma, najbolje je ubaciti nulu u onaj od ova dva registra (D ili E) koji ne koristimo.

Kao primer, tražićemo adresu sedme naredbe po redu u liniji 7000. Prvo nalazimo adresu te linije i pomeramo se pred sam tekst. Registar E ne koristimo, pa ćemo ga postaviti na nulu:

```
27.   LD        HL, 7000
      CALL      LINE_ADDR
      INC       HL
      INC       HL
      INC       HL
      LD        DE, &0700
      CALL      EACH_STMT
```

Povratak iz EACH_STMT usledиće bilo da je pronađena naredba sa kodom E, bilo da je pronađena naredba sa rednim brojem D, bilo da je dostignut kraj linije (ENTER).

Na izlazu će indikator prenosa „C" biti resetovan samo ako je naredba pronađena. U protivnom je C=1, dostignut je kraj linije i HL sadrži adresu karaktera ENTER.

Ako je pronađena naredba E, indikator nule „z" biće resetovan, a HL sadrži adresu same naredbe. Međutim, ako je pronađena naredba D, biće Z=1, a HL pokazuje neposredno ispred naredbe, odnosno neposredno iza prethodne (znak „:").

U svim slučajevima registar E na izlazu ostaje očuvan, registar B se ne koristi, a A sadrži bajt sa adrese HL.

Često se EACH_STMT poziva da bi se preskočila neka naredba, odnosno da bi se došlo pred sledeću. U tom slučaju, ako je adresa naredbe koja se preskače smeštena u HL, treba postupiti kao u primeru:

```
28.   DEC       HL
      LD        DE, &0200
      CALL      &198B
```

13. LOOK_PROG — CALL &1D86

Rutina LOOK_PROG koristi se za pretraživanje bejzik programa u cilju nalaženja neke zadate naredbe. Automatski se ispituje linija po linija, uzastopnim pozivanjem EACH_STMT.

Adresa od koje počinje traganje nalazi se u registru HL na ulazu. Slično kao u slučaju EACH_STMT, mora se uzeti adresa **ispred neke naredbe,** tačnije adresa znaka „:". Međutim, ako se počinje od početka neke linije, uzima se adresa **neposredno ispred linijskog broja,** a ne ispred prve naredbe kao kod EACH_STMT. Obično se ispred linijskog broja nalazi karakter ENTER, kao kraj prethodne linije, ili oznaka &80 (početak bejzik bloka).

Naredba koja se traži smešta se u registar E. Najsigurnije je uvek pri tome staviti i nulu u D, mada je to samo neophodno ako pretraživanje počinje iz sredine linije.

Na izlazu će indikator prenosa „C" biti setovan ukoliko u čitavom programu, počev od zadate adrese, nije pronađena ni jedna naredba sa kodom E. U tom slučaju HL sadrži adresu iza programa (VARS), a indikator nule „Z" je takođe setovan.

Međutim, ako je C=0, to znači da je naredba E pronađena. Registar HL sadrži njenu adresu, sistemska varijabla NEWPPC sadrži linijski broj odgovarajuće bejzik linije, a adresa

kodom većim od &20, uz izuzetak znaka &0D (ENTER). Svi nevažeći karakteri se ignorišu pri interpretaciji bejzika (prazna polja, komandni simboli za boju, itd.). Ukoliko na ulazu u GET_CHAR varijabla CH_ADD sadrži adresu nekog nevažećeg znaka, pozicija se pomera unapred za jedno mesto i traži se **prvi važeći** karakter. Tek kada se on pronađe, obavlja se povratak uz odgovarajući sadržaj registara A i HL, kao i varijable CH_ADD.

Primer 25. će dati prvi važeći bejzik karakter linije u editnom prostoru.

```
25.  LD      HL, (E_LINE)
     LD      (CH_ADD), HL
     RST     &18
```

Ukoliko želimo da se iz našeg mašinskog programa uspešno vratimo u bejzik, moramo obezbediti da CH_ADD pri povratku ima **istu vrednost kao i na početku programa.** A taj uslov neće biti ispunjen ako koristimo rutine kao što je GET_CHAR. Najbolje je, zato, na samom početku našeg programa staviti:

```
     LD      HL, (CH_ADD)
     PUSH    HL
```

a na kraju, neposredno ispred RET:

```
     POP     HL
     LD      (CH_ADD), HL
```

6. NEXT_CHAR RST &20

Potprogram NEXT_CHAR je praktično isto što i GET_CHAR, samo što pre početka rada uvećava CH_ADD za jedinicu. Prema tome, na izlazu se neće dobiti „prvi važeći karakter", već „sledeći važeći karakter". Uzastopnim pozivom NEXT_CHAR može se pročitati čitava bejzik linija.

7. LINE_ADDR CALL &196E

Potprogram LINE_ADDR služi za nalaženje adrese bejzik linije čiji linijski broj se na ulazu nalazi u registru HL.

Na izlazu, pre svega, treba proveriti indikator nule „Z". Ako je Z=1, onda je pronađena linija koju smo tražili i njena adresa je vraćena u HL. Ukoliko je, međutim, Z=0, onda to znači da linija nije pronađena, a u HL će se nalaziti adresa prve sledeće linije u bejzik listi ili, ako ni takve nema, adresa odmah iza bejzik liste (VARS).

U oba slučaja DE će na izlazu sadržati adresu linije koja prethodi onoj iz HL. Ako takva ne postoji, onda je DE=HL=PROG.

Registar BC na izlazu sadrži ulazni linijski broj iz HL.

8. NEXT_ONE CALL &19B8

Ulazni parametar u ovaj potprogram je adresa proizvoljne bejzik linije ili promenljive, a treba je smestiti u registar HL. Na izlazu ćemo u registru DE dobiti adresu sledeće linije, odnosno promenljive, preciznije adresu odmah iza linije (promenljive) koju smo naznačili na ulazu. Registar HL i dalje će čuvati ulaznu adresu, a u BC ćemo dobiti dužinu zadate linije (promenljive) tako da je HL+BC=DE.

Ako, recimo, neka bejzik linija počinje na &5D00, onda ćemo adresu sledeće dobiti sa:

```
26.  LD      HL, &5D00
     CALL    &19B8
```

Na ovaj način, znajući još da prva linija počinje na adresi PROG, možemo jednostavno pronaći adrese svih linija u programu. Isto važi i za promenljive, koje inače počinju na adresi VARS.

Potprogram NEXT_ONE **ne sme se pozivati** ako HL stoji na kraju bejzik bloka, tj. iza poslednje promenljive (bajt &80).

9. ALPHA CALL &2C8D

Potprogram ALPHA proverava da li je neki karakter alfabetski, tj. da li predstavlja veliko ili malo slovo abecede. Ulazi se sa kodom ispitivanog karaktera u akumulatoru.

Svi registri, uključujući i A, su sačuvani, a rezultat testa nalazi se u indikatoru prenosa „C". Samo ako je on setovan, radi se o slovu.

PRINT, ili na donji deo ekrana, gde se inače ispisuju poruke i raporti, na štampač, ili na neko drugo odredište, zavisi od toga koji je kanal trenutno otvoren. Kao da informacije zaista „teku" nekakvim „kanalima", koji se po želji mogu otvarati i zatvarati. A radi se, u stvari, o tome da su rutine koje obavljaju ulazno-izlazne poslove **univerzalne**, i mogu da opsluže bilo koji periferijski uređaj. Jednostavno, u tim rutinama se vrši grananje, u zavisnosti od toga koji periferijski uređaj je aktivan. A otvaranje kanala je prosto setovanje nekih bitova u memoriji, koji će biti testirani u fazi grananja ulazno — izlazne rutine.

Postoje četiri osnovna kanala, označena sa „K", „S", „R" i „P", a povezuju računar redom sa tastaturom (i donjim delom ekrana), gornjim delom ekrana, radnim prostorom i štampačem.

Jedan kanal obuhvata jednu ili više struja, kojima „teku" podaci. Struje se označavaju brojevima, a da bi se otvorio neki kanal, dovoljno je navesti struju koja tom kanalu pripada.

Kanal „K" obuhvata struje &00, &01, i &FD, kanal „S" struje &02 i &FE, kanal „R" struju &FF, a kanal „P" struju &03.

O otvaranju kanala još ćemo detaljno govoriti, a sada nas interesuje samo sadržaj bloka kanalskih informacija, počev od adrese CHANS. Tu ukupno ima 20 bajtova, po pet za svaki od četiri kanala. Prva dva bajta sadrže adresu rutine koja tim kanalom šalje informacije na izlaz, druga dva sadrže adresu rutine koja prima informacije sa ulaza, a peti bajt je oznaka samog kanala „K", „S", „R", ili „P".

Čitajući sadržaj kanalskih informacija, videćemo da se izlazna rutina za kanale „K", „S" i „P" nalazi na adresi &09F4, dok je za kanal „R" to &0F81. Dalje, ulazna rutina za kanal „K" je &10A8, dok ostali kanali nemaju definisan ulaz, pa tamo figuriše adresa &15C4 za izdavalje raporta „J Invalid I/0 device".

Pošto su kanalske informacije u RAM-u, možemo ih i sami menjati, ubacujući umesto standardnih ulazno-izlaznih rutina neke svoje adrese. Na žalost, to neće biti baš naročito jednostavno, jer „spektrum" neke adrese stalno obnavlja. Jedino rešenje je da i mi svoje adrese stalno obnavljamo, pa — ko duže izdrži . . .

Iza bloka sa kanalima nalazi se graničnik &80.

SISTEMSKE PROMENLJIVE

Prostor između adresa 23552 i 23733 zove se blok sistemskih promenljivih, jer svaki od tih bajtova nosi neku važnu informaciju, neophodnu za uspešan rad čitavog sistema.

Neke sistemske promenljive sastoje se od samo jednog bajta, neke od dva bajta, a neke i od više bajtova, u zavisnosti od toga kakvu informaciju čuvaju. Na primer, sistemska promenljiva zvana SUBPPC, na adresi 23623, čuva redni broj bejzik naredbe u liniji koja se upravo interpretira. Za to je dovoljan jedan bajt. S druge strane, varijabla PPC sadrži sam linijski broj linije koja se interpretira, a za to su potrebna dva bajta (23621 i 23622). Prvi bajt je uvek bajt manje težine.

Čitanje sadržaja sistemskih varijabli vršićemo u bejziku pomoću funkcije **PEEK**, a upisivanje novog sadržaja obavićemo naredbom **POKE**. Naravno, dobro treba proveriti da li upisujemo pravi broj na pravo mesto, jer posledice se, inače, ne mogu uvek predvideti.

„Spektrum" neprestano u procesorskom registru IY drži adresu 23610, i u odnosu na nju određuje relativni položaj ostalih sistemskih promenljivih. Na primer, varijabla SUBPPC se nalazi na adresi IY+13.

Dajemo pregled svih sistemskih varijabli, sa kratkim opisom njihovog sadržaja. Osim dekadne i heksadekadne adrese, navedena je i relativna adresa u odnosu na registar IY, a iza imena promenljive stoji i broj bajtova koje zauzima.

23552	&5C00	IY-58	KSTATE	8	Koristi se prilikom očitavanja tastature.
23560	&5C08	IY-50	LAST_K	1	Poslednji ukucani znak.
23561	&5C09	IY-49	REPDEL	1	Vreme mereno u pedesetim delovima sekunde posle koga počinje automatsko ponavljanje pritisnutog tastera.
23562	&5C0A	IY-48	REPPER	1	Vreme mereno u pedesetim delovima sekunde, između dva uzastopna ponavljanja pritisnutog tastera.
23563	&5C0B	IY-47	DEFADD	2	Adresa prvog argumenta iza otvorene zagrade u naredbi **DEF FN**, ali samo ako je u toku izračunavanje funkcije. Inače je sadržaj nula.
23565	&5C0D	IY-45	K_DATA	1	Kôd za boju prilikom unošenja komandnih znakova direktno sa tastature.

23556	&5C0E	IY-44	TVDATA	2	Kodovi koji slede iza komandnih znakova, prilikom njihovog ispisivanja.
23568	&5C10	IY-42	STRMS	38	Tabela koja određuje kojoj struji odgovara koji kanal. Prva dva bajta u tabeli odnose se na struju &FD, druga dva na struju &FE, zatim na &FF, &00, &01, itd. U tim parovima bajtova upisan je **relativni položaj** zapisa odgovarajućeg kanala u bloku kanalskih informacija. Na primer, treći par bajtova sadrži broj &000B, a to znači da struji &FF odgovara kanal „R", jer njegov zapis počinje na **jedanaestom bajtu** u bloku kanalskih informacija.
23606	&5C36	IY-4	CHARS	2	Adresa umanjena za 256, na kojoj počinje tabela sa ASCII znakovima. U ROM-u ova tabela počinje na &3D00, tako da CHARS sadrži &3C00. Možemo napraviti i svoju tabelu znakova. Svaki znak je predstavljen sa osam bajtova, odnosno osam linija iz kojih je sačinjeno znakovno polje na ekranu.
23608	&5C38	IY-2	RASP	1	Trajanje zvuka upozorenja.
23609	&5C39	IY-1	PIP	1	Trajanje tona koji se generiše pri pritisku bilo kog tastera.
23610	&5C3A	IY+0	ERR_NR	1	Kôd za izdavanje raporta, umanjen za jedinicu.
23611	&5C3B	IY+1	FLAGS	1	Pojedini bitovi koriste se za indikaciju različitih stanja sistema prilikom interpretacije bejzika.
23612	&5C3C	IY+2	FVFLAG	1	Indikatori stanja pri radu sa TV.
23613	&5C3D	IY+3	ERR_SP	2	Pozicija na mašinskom steku gde se nalazi adresa rutine za prijavljivanje grešaka.
23615	&5C3F	IY+5	LISTSP	2	Pozicija na mašinskom steku gde se nalazi adresa za povratak iz automatskog listanja.
23617	&5C41	IY+7	MODE	1	Indikator trenutnog stanja kursora.
23618	&5C42	IY+8	NEWPPC	2	Linija u bejzik programu na koju će biti obavljen skok (npr. **GO TO).**
23620	&5C44	IY+10	NSPPC	1	Redni broj naredbe u liniji na koju će biti obavljen skok. Ako je sadržaj &FF, onda nema skoka.
23261	&5C45	IY+11	PPC	2	Linijski broj naredbe koja se upravo izvršava.
23623	&5C47	IY+13	SUBPPC	1	Redni broj naredbe u liniji koja se upravo izvršava.
23624	&5C48	IY+14	BORDCR	1	Atribut za donji deo ekrana. Boja podloge je istovremeno i boja ivičnog dela ekrana.
23625	&5C49	IY+15	E_PPC	2	Broj bejzik linije u kojoj se nalazi programski kursor.
23627	&5C4B	IY+17	VARS	2	Adresa bloka sa bejzik varijablama.
23629	&5C4D	IY+19	DEST	2	Adresa bejzik varijable kojoj se upravo dodeljuje vrednost.
23631	&5C4F	IY+21	CHANS	2	Adresa kanalskih informacija.
23633	&5C51	IY+23	CURCHL	2	Adresa trenutno važećeg kanala u bloku kanalskih informacija.
23635	&5C53	IY+25	PROG	2	Adresa bejzik programa.
23637	&5C55	IY+27	NXTLIN	2	Adresa bejzik linije neposredno iza one koja se trenutno izvršava.
23639	&5C57	IY+29	DATADD	2	Adresa iza poslednjeg pročitanog podatka u **DATA** listi.
23641	&5C59	IY+31	E_LINE	2	Adresa editnog prostora.
23643	&5C5B	IY+33	K_CUR	2	Adresa kursora.
23645	&5C5D	IY+35	CH_ADD	2	Adresa karaktera (u bejzik liniji) koji je na redu da se interpretira.
23647	&5C5F	IY+37	X_PTR	2	Adresa na kojoj je otkrivena greška u bejzik liniji.
23649	&5C61	IY+39	WORKSP	2	Adresa radnog prostora.

Treba imati u vidu da će se u povratak iz LINE_SCAN obaviti **samo ako je sinteksa bila korektna.** U protivnom ćemo imati prekid sa raportom za grešku „C Nonsense in BASIC". Međutim, to se može izbeći ako se vešto iskoristi sistemska varijabla ERR_SP, o čemu ćemo govoriti u okviru sledeće rutine.

3. ERROR_1 — RST &08

Potprogram ERROR_1 ima zadatak da brzo i efikasno razreši sve probleme nastale pri pojavi greške u interpretaciji bejzika. Možda se pitate šta tu uopšte ima problematično? Zar ne može računar, čim utvrdi grešku (npr. deljenje nulom), jednostavno da prekine rad i vrati se tamo odakle je dotična rutina bila pozvana? Svakako da bi mogao, kada bi znao gde treba da se vrati. Problem je zapravo u tome što je mašinski stek, koji inače sadrži i adresu za povratak, u međuvremenu verovatno zatrpan drugim adresama i podacima i pitanje je gde u toj gomili tražiti ono što nam treba. Zato se prosto pozove rutina ERROR_1, koja bukvalno poništi čitav mašinski stek i postavi ga na adresu sadržanu u sistemskoj varijabli ERR_SP. Ako nekim slučajem budete analizirali rutinu ERROR_1, videćete da se ona praktično samo sastoji iz:

```
LD      SP, (ERR_SP)
RET
```

Ostatak čine manje važni poslovi, kao što su: priprema pozicije za znak „?" u liniji sa greškom, prikupljanje koda za izdavanje raporta itd.

Pitanje koje se ipak nameće je: šta se dešava posle naredbe RET? Gde se to obavlja povratak iz ERROR_1? Svakako ne tamo odakle je usledio poziv sa RST &08, već na neku novu adresu, koja se zatekla u vrhu mašinskog steka posle instrukcije **LD SP, (ERR_SP).** Obično je to adresa za izdavanje raporta, &1303, tako da svaka greška izaziva prekid programa.

Naredbu **RST &08** obavezno mora da prati još jedan bajt, kojim se definiše tip greške. Vrednost tog bajta treba da je za jedan manja od stvarnog koda greške: &FF za „0 OK", &00 za „1 NEXT without FOR", itd. Recimo, ako u našem mašinskom programu, na bilo kom mestu želimo prekid sa raportom „F Invalid file name", postupaćemo kao u primeru 23.

```
23.   RST     &08
      DEFB    &0E
```

Međutim, cilj nam je da stavimo pod kontrolu i greške koje se mogu javiti dok se izvršava neki potprogram iz ROM-a. Želimo da grešku analiziramo i sami, pa da donesemo odluku da li treba ići na raport ili ne. Nećemo, dakle, da se posle ERROR_1 uvek odlazi na &1303, već na neku adresu u RAM-u, koju sami odaberemo.

Sve što treba uraditi je: ubaciti željenu adresu na mašinski stek, a u ERR_SP ubaciti pri tome dobijenu vrednost procesorskog registra SP. Razume se, prethodno ćemo upamtiti stari sadržaj ERR_SP, da bismo se kasnije uspešno vratili u bejzik. (primer 24.)

```
24.   LD      HL, (ERR_SP)
      PUSH    HL
      LD      HL, „naša adresa za greške"
      PUSH    HL
      LD      (ERR_SP),SP
```

Posle ovoga, svaka greška će izazvati skok na našu adresu. Sistemska varijabla X_PTR sadržaće poziciju na kojoj je greška otkrivena, tj. poslednje stanje varijable CH_ADD. Sama greška, tj. njen kôd umanjen za jedan, nalaziće se u ERR_NR.

4. REMOVE_FP — CALL &11A7

Potprogram REMOVE_FP uklanja iz bejzik linije sve binarne zapise numeričkih konstanti, kao i odgovarajuće bajtove umetnute iza argumenata DEF FN funkcija. Na ulazu se zadaje početna adresa za pretraživanje, a treba je smestiti u HL. Povratak iz potprograma će uslediti čim se naiđe na kraj linije (&0D, ENTER). Registri nisu očuvani. Izlaznih parametara nema a BC je na povratku uvek nula.

5. GET_CHAR — RST &18

Ovaj potprogram koristi se za čitanje znakova iz kojih se sastoji bejzik linija. Ulaznih parametara nema, a na izlazu će akumulator sadržati karakter sa adrese CH_ADD. Registri su očuvani, osim HL, koji će i sam sadržati adresu CH_ADD.

Međutim, ne radi se prosto o čitanju sa zadate adrese, jer bi inače bila dovoljna i naredba **LD A,(HL).** Ovde na izlazu dobijamo samo **važeće bejzik znakove,** dakle one sa

očuvani, a posle obavljenog štampanja broj biva izbrisan sa računskog steka. PRINT_FP se ne sme pozivati ako je stek prazan.

Treba voditi računa i o jednom ne baš beznačajnom „bagu", koji je skriven u ovoj rutini. Naime, ako je broj koji se štampa bio po modulu manji od jedinice, tako da mu je celobrojni deo nula (ali sam broj je različit od nule), onda će, na povratku iz PRINT_FP, na računskom steku ostati zaboravljena jedna nula, zamenjujući broj koji je otštampan! Upravo je to razlog zašto „spektrum" ne ume da izračuna, recimo, izraz oblika „bilo koji niz"+STR$ 0.5. Problem je u tome što funkcija STR$ i sama koristi PRINT_FP da bi „odštampala" niz direktno u radni prostor, preko kanala „R". Nula koja pri tome ostaje na steku remeti dalje računanje.

11. OUT_LINE — CALL &1855

Potprogram OUT_LINE obavlja štampanje bejzik linije, izdvojene iz programa. Tako, na primer, naredba LIST prosto poziva uzastopno OUT_LINE, dok se ne izlistaju sve linije.

Na ulazu se jedino zahteva početna adresa linije i treba je staviti u registar HL, a podrazumeva se, kao i do sada, da je odgovarajući kanal otvoren. Regitri na izlazu nisu sačuvani. Primer 20. pokazuje kako se može odštampati prva bejzik linija iz programa.

```
20.     LD      HL, (&5C53)
        CALL    &1855
```

Početak bejzik programa je ovde uzet iz sistemske varijable PROG. Ako u programu ne postoji ni jedna linija, ili je HL postavljen na kraj bejzik programa, usledиće povratak bez bilo kakvih problema.

BEJZIK INTERPRETER

1. EDITOR — CALL &0F2C

EDITOR je jedan od najvažnijih potprograma u „spektrumovom" ROM-u. Bilo šta da ukucavamo preko tastature: naredbu, programsku liniju, ili INPUT liniju — sve će se to odvijati pod kontrolom ove rutine.

EDITOR možemo pozivati i sami, u okviru svog mašinskog programa, kad god od korisnika očekujemo da unese neki test. Prethodno ćemo obrisati editni prostor (pozivom SET_MIN), pripremajući ga tako za prihvatanje nove linije. Osim toga, obavezno se mora otvoriti kanal „K", a ako želimo K-kursor, moramo resetovati bit 3 varijable FLAGS. Inače, ulaznih parametara nema.

```
21.   CALL &16BO      ; SET_MIN
      RES 3,(IY+1)    ; K-kursor
      SUB A           ; Struja &00
      CALL &1601      ; Kanal „K" — CHAN_OPEN
      CALL &0F2C      ; EDITOR
```

U dnu ekrana će se pojaviti kursor, a korisnik može da počne sa kucanjem bejzik linije. Povratak iz editora će uslediti čim bude unet karakter **ENTER**, a ukucani tekst ostaće u memoriji počev od adrese E_LINE, što je, u stvari, početak editnog prostora. Tamo možemo analizirati liniju znak po znak i proveriti njenu sintaksu.

Izlaznih parametara nema. Registri nisu očuvani. Posle svakog ukucanog znaka automatski se poziva rutina BEEPER, tako da na izlazu iz editora nije sačuvan ni registar IX.

2. LINE_SCAN — CALL &1B17

Potprogram LINE_SCAN koristi se za sintaksnu proveru bejzik linije koja je prethodno bila uneta u editni prostor, recimo pomoću editora:

```
22.   CALL &0F2C      ; EDITOR
      CALL &1B17      ; LINE_SCAN
```

Ulaznih i izlaznih parametara nema, a registri nisu očuvani. U toku sintaksne provere, iza svake numeričke konstante, kao i iza svakog argumenta u DEF FN naredbama, biće umetnut bajt &0E, praćen sa još pet bajtova binarnog zapisa.

23651	&5C63	IY+41	STKBOT	2	Adresa računskog steka.
23653	&5C65	IY+43	STKEND	2	Adresa iza računskog steka.
23655	&5C67	IY+45	BREG	2	Sadržaj procesorskog B registra u trenutku poziva kalkulatora.
23656	&5C68	IY+46	MEM	2	Adresa memorijskog prostora koji se koristi iz kalkulatora.
23658	&5C6A	IY+48	FLAGS2	1	Indikatori različitih stanja.
23659	&5C6B	IY+49	DF_SZ	1	Ukupan broj redova iz kojih se sastoji donji deo ekrana.
23660	&5C6C	IY+50	S_TOP	2	Linijski broj programske linije od koje počinje automatsko listanje.
23662	&5C6E	IY+52	OLDPPC	2	Linijski broj na koji će skočiti naredba **CONTINUE.**
23664	&5C70	IY+54	OSPPC	1	Broj naredbe na koju će skočiti **CONTINUE.**
23665	&5C71	IY+55	FLAGX	1	Razni indikatori.
23666	&5C72	IY+56	STRLEN	2	Dužina promenljive kojoj se upravo dodeljuje vrednost (ako je promenljiva znakovnog tipa).
23668	&5C74	IY+58	T_ADDR	2	Adresa sledećeg podatka u parametarskoj tabeli, za vreme interpretiranja neke bejzik naredbe. Parametarska tabela počinje u ROM-u na adresi &1A7A i sadrži podatke o sintaksi svake bejzik naredbe.
23670	&5C76	IY+60	SEED	2	Početna vrednost niza slučajnih brojeva, kako je to definisano narebom **RANDOMIZE.**
23672	&5C78	IY+62	FRAMES	3	Časovnik — brojač mašinskih prekida. Svakih 20 ms brojač se uveća za jedinicu. Prvi bajt ima najmanju težinu.
23675	&5C7B	IY+65	UDG	2	Adresa grafičkih znakova.
23677	&5C7D	IY+67	COORDS	2	„x" i „y" koordinata poslednje tačke nacrtane na ekranu.
23679	&5C7F	IY+69	P_POSN	1	„33" minus kolona na kojoj će štampač ispisati sledeći znak.
23680	&5C80	IY+70	PR_CC	2	Adresa sledeće pozicije u baferu za štampač koju će koristiti naredba LPRINT.
23682	&5C82	IY+72	ECHO_E	2	„33" minus kolona i „24" minus linija za poziciju u donjem delu ekrana, koja odgovara kraju **INPUT** linije.
23684	&5C84	IY+74	DF_CC	2	Adresa sledeće PRINT pozicije u video memoriji.
23686	&5C86	IY+76	DFCCL	2	Kao DF_CC, samo za donji deo ekrana.
23688	&5C88	IY+78	S_POSN	2	„33" minus kolona i „24" minus linija za sledeću PRINT poziciju.
23690	&5C8A	IY+80	SPOSNL	2	Kao S_POSN, samo za donji deo ekrana.
23692	&5C8C	IY+82	SCR_CT	1	Brojač vertikalnih pomeranja ekrana (skrolovanja). Kada stigne do nule, ispisuje se poruka „scroll?".
23693	&5C8D	IY+83	ATTR_P	1	Stalni atribut za glavni deo ekrana, onako kako je definisan naredbama za boju.
23694	&5C8E	IY+84	MASK_P	1	Stalna „maska", postavljena naredbama **OVER, INVERSE,** itd.
23695	&5C8F	IY+85	ATTR_T	1	Privremeni atribut, postavljen prilozima za boju u okviru jedne **PRINT** narebe.
23696	&5C90	IY+86	MASK_T	1	Privremena „maska" za tekuću PRINT naredbu.
23697	&5C91	IY+87	P_FLAG	1	Indikatori.
23698	&5C92	IY+88	MEMBOT	30	Šest memorijskih lokacija od po pet bajtova koje koristi kalkulator.
23728	&5CB0	IY+118		2	Ne koristi se.
23730	&5CB2	IY+120	RAMTOP	2	Poslednja adresa dostupna bejziku.
23732	&5CB4	IY+122	P_RAMT	2	Poslednja adresa RAM-a.

KOMUNIKACIJA SA KASETOFONOM

Kada govorimo o strukturi memorije i načinu na koji „spektrum" zapisuje razne informacije, moramo nešto reći i o tome kako se organizuje snimak programa na magnetofonskoj traci. Naročito će nam ovo biti važno kada budemo direktno koristili mašinske rutine za rad sa kasetofonom, o čemu će biti reči kasnije.

Bejzik naredba **SAVE** obavlja svakako jednu od najsloženijih operacija na „spektrumu" uopšte. Snimiti sadržaj memorije na običnu kasetu, i to tako da se sve kasnije bez ikakvih problema može uzeti natrag, zaista je vrhunac programerskog umeća.

Mi, na žalost, ne možemo ovde da se detaljno upoznamo sa procedurom snimanja, već ćemo samo razjasniti neke najvažnije stvari.

Snimak se uvek sastoji iz dva zasebna dela. Bez obzira šta se šalje na kasetu, prvo dolazi tzv. **zaglavlje,** a tek iza njega glavni blok podataka. Zaglavlje nosi informaciju o tome šta je snimljeno, a sastoji se od 17 bajtova.

Prvi bajt zaglavlja (označimo ga sa B0) može imati vrednost **&00, &01, &02,** ili **&0,3,** zavisno od toga da li se snima **bejzik program, numerički niz, znakovni niz, ili prosto blok bajtova.**

Narednih deset bajtova (B1—B10) sadrže ime programa, odnosno bloka koji se snima. Ime mora imati makar jedan karakter, a ako je kraće od deset, ostatak se dopunjava praznim poljima.

Bajtovi B11 i B12 uvek sadrže dužinu bloka koji će biti snimljen, pri čemu B11 ima manju težinu.

Preostala četiri bajta zavise od tipa podataka koji se snimaju:

- **Bejzik program (tip &00).**

B13 i B14 sadrže linijski broj od koga će se program automatski startovati po učitavanju sa kasete (B13 ima manju težinu). Ako nema automatskog starta, onda B14 sadrži &80.

Bajtovi B15 i B16 sadrže dužinu samog programa, bez promenljivih.

- **Numerički i znakovni niz (tipovi &01 i &02).**

B14 sadrži ime niza, tačnije prvi bajt zapisa niza u prostoru bejzik promenljivih (redukovano ime i tip promenljive). Ostali bajtovi nisu značajni.

- **Blok bajtova (tip &03).**

B13 i B14 sadrže početnu adresu bloka u memoriji odakle će se bajtovi slati na kasetu. Preostala dva bajta ne nose nikakvu informaciju

POTPROGRAMI IZ ROM-a

Sve što „spektrum" ume da radi, zapisano je u njegovom ROM-u, u obliku mašinskih programa ukupne dužine 16 kilobajta. Tu se mogu naći uputstva za rad sa periferijskim uređajima, za prihvatanje, razumevanje i izvršenje bejzik naredbi, i za izdavanje raporta o tome kako je izvršavanje proteklo.

Većina rutina je organizovana na principu potprograma, tako da se jednostavno mogu pozivati i koristiti kada god to zatreba. Na primer, „spektrum" poziva potprogram za štampanje znakova više od 20 puta, u raznim prilikama. Nikakav problem nije da i mi to uradimo u okviru svog mašinskog programa.

Pozivanje potprograma iz ROM-a se vrši, naravno, pomoću naredbe CALL (ili RST ako adresa to dopušta). Na primer, program za štampanje znakova na ekranu, nalazi se na adresi & 0010 i pozivaćemo ga sa **RST & 10.** Međutim, koji će se znak štampati? Odgovor na to pitanje je ovde prost: štampaće se onaj znak čiji se kod nalazi u akumulatoru mikroprocesora. Prema tome, ako hoćemo da štampamo, recimo, slovo „A", moramo uraditi **LD A, & 41** pre poziva rutine iz ROM-a. Na taj način pripremamo **ulazne parametre** neophodne za uspešan rad potprograma. Time, međutim, uopšte nisu rešeni svi problemi oko štampanja. Pre svega je ostalo nejasno kuda će otići slovo „A": na glavni deo ekrana (tj. tamo gde inače radi bejzik naredba **PRINT),** na donji deo ekrana (tamo gde se ispisuje raport), ili možda čak na neki drugi periferijski uređaj (rutina za štampanje je inače univerzalna).

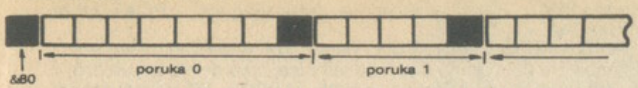
Ovakvu nedoumicu treba rešiti još na samom početku našeg programa, **otvaranjem odgovarajućeg kanala.** Za sada ćemo samo reći kako se to radi za glavni deo ekrana: u akumulator treba ubaciti &02, i onda pozvati rutinu &1601. (primer 3.)

```
3.    LD      A, &02
      CALL    &1601
```

Čim se ove dve instrukcije izvrše, nadalje će svako štampanje ići na glavni deo ekrana, sve dok ne otvorimo neki drugi kanal.

Prema tome, veliki deo posla oko poziva nekog potprograma iz ROM-a predstavljaju pripreme za taj poziv. Neke od priprema su sasvim jasne (recimo, zadavanje ulaznih parametara), ali neke su prilično skrivene. Dovoljno je napraviti mali previd pa da nikad ne

Slika 23. Tabela sa porukama



Ulazni parametri u PO_MSG su adresa tabele, smeštena u DE, i redni broj poruke u tabeli, smešten u A. Redni brojevi počinju od nule.

Kao primer ćemo otštampati tekst „Nonsense in BASIC", koristeći tabelu sa raportima iz ROM-a, na adresi &1391. Redni broj izabranog raporta je &0C:

```
16.   LD      A, &0C
      LD      DE, &1391
      CALL    &0C0A
```

Na izlazu iz potprograma očuvan je registar HL.

8. OUT_CODE — CALL &15EF

Ovo je, u stvari, ista rutina kao i PRINT_A_1, samo što na početku akumulatoru dodaje &30, što je ASCII kôd za karakter „0". Posledica toga je da ćemo na ekranu dobiti cifru dekadnog sistema koja odgovara numeričkom sadržaju akumulatora na ulazu (naravno samo ako je taj sadržaj između 0 i 9). Lako se postiže da za vrednost akumulatora 0—15 dobijemo cifre heksadekadnog sistema (primer 17.)

```
17.         CP      10
            JR      C,COD_OK
            ADD     A,7
COD_OK      CALL    &15EF
```

Na izlazu iz OUT_CODE nije očuvan registar E.

9. OUT_NUM_1 — CALL &1A1B

Potprogram OUT_NUM_1 služi za štampanje četvorocifrenog celog broja koji se na ulazu nalazi u registru BC. Ispisivanje se obavlja bez vodećih blankova, kao, recimo, u primeru:

```
18. LD BC, 7000
    CALL &1A1B
```

Na izlazu su sačuvani registri DE i HL.

„Spektrum" koristi ovaj potprogram da bi štampao linijski broj iza raporta za grešku. Preporučujemo vam da ipak, ako već štampate cele brojeve, omogućite ispis sve do 65535 i uvedete izbor vodećih blankova. U tu svrhu treba koristiti potprogram &192A, o kome nećemo inače posebno govoriti, a koji štampa pojedinačne cifre dekadnog broja. Kompletan potprogram bi izgledao:

```
19.
DIGIT5  LD      BC, -10000
        CALL    &192A
        LD      BC, -1000
        CALL    &192A
        LD      BC, -100
        CALL    &192A
        LD      BC _10
        CALL    &192A
        LD      A,L
        CALL    &15EF
        RET
```

Pre poziva DIGIT5, u HL treba ubaciti broj koji se štampa, a u registar E: &FF za slučaj bez vodećih blankova, a &20 ako su vodeći blankovi potrebni. Registri na izlazu neće biti očuvani.

10. PRINT_FP — CALL &2DE3

Ova rutina se koristi za štampanje proizvoljnih brojeva. Nema ulaznih parametara, a broj koji se štamapa mora biti na računskom steku. Izlaznih parametara nema, registri nisu

5. PLOT — CALL &22DC
PLOT_BC — CALL &22DF

PLOT je izvršna rutina odgovarajuće bejzik naredbe i obavlja crtanje tačke na ekranu. Nema ulaznih parametara, ali se podrazumeva da se koordinate „x" i „y" nalaze na računskom steku: „y" na vrhu steka, a „x" odmah ispod. O upotrebi računskog steka govorićemo kada budemo objašnjavali rad „spektrumovog" kalkulatora.

Ukoliko ulazak u rutinu obavimo na adresi PLOT_BC, onda sami prethodno moramo da smestimo koordinate u BC (C=x, B=y). Primer 13. daće tačku u centru ekrana.

```
13.  LD      BC, &5880
     CALL    &22DF
```

Izlaznih parametara iz potprograma nema, a registri nisu očuvani.

6. PRINT_A_1 — RST &10

Potprogram PRINT_A_1 obavlja štampanje ASCII karaktera koji se na ulazu nalazi u akumulatoru. Prihvataju se svi kodovi navedeni u Dodatku A „spektrumovog" uputstva, a u zavisnosti od kanala koji je trenutno otvoren, znak će se poslati na odgovarajući izlaz. Izlaznih parametara nema, registri su sačuvani, a ako se štampanje vrši na ekranu, koristiće se ista pozicija koju bi inače uzela i naredba **PRINT.**

Kao primer ćemo odštampati slovo „A" u donjem delu ekrana (pr. 14). Prethodno otvaramo kanal „K", a pre povratka pozivamo i WAIT_KEY, jer bi u protivnom preko slova „A" odmah došao raport.

```
14.  SUB     A
     CALL    &1601
     LD      A, „A"
     RST     &10
     CALL    &15D4
     RET
```

U bejziku se ovo isto izvodi sa **PRINT #0; „A";: PAUSE 0.**

Posle poziva PRINT_A_1 pozicija za štampanje se pomera odmah iza odštampanog znaka.

Komandni znakovi se tretiraju na uobičajen način. Recimo, bejzik naredba **PRINT AT 7,13; PAPER 1; INK 5; „M"** ostvaruje se u mašinskom jeziku:

```
15.  LD      A, &16     AT 7,13;
     RST     &10
     LD      A,7
     RST     &10

     LD      A,13
     RST     &10
-----------------------------------
     LD      A, &11     PAPER 1;
     RST     &10
     LD      A,1
     RST     &10
-----------------------------------
     LD      A, &10     INK 5;
     RST     &10
     LD      A,5
     RST     &10
-----------------------------------
     LD      A, &4D     „M";
     RST     &10
```

7. PO_MSG — CALL &0C0A

U slučajevima kada treba štampati neki tekst, ili poruku, koristi se potprogram PO_MSG. Sve poruke moraju se smestiti u memoriju u vidu tabele. Početak tabele se označava bajtom &80, ili bilo kojim drugim čiji je bit 7 setovan. Onda redom slede tekstovi poruka. Poseldnji znak svake poruke takođe mora imati setovan bit 7. Ako se, recimo, poruka završava slovom „a", odgovarajući bajt neće biti &61 već &E1:

dočekamo povratak iz rutine koju smo brzopleto pozvali. Jedna od takvih opasnih sitnica je sadržaj registra IY. Ako on slučajno nije &5C3A (a to je adresa sistemske varijable ERR_NR), „spektrum" će se vrlo verovatno zbuniti. On dosta često koristi registar IY, jer u odnosu na njega određuje adrese svih sistemskih varijabli. Najsigurnije je, zato, uopšte ne koristiti IY u svom programu.

Drugi deo posla oko rada sa ROM-om, jeste prikupljanje i tumačenje **izlaznih parametara**, tj. onoga što nam potprogram vraća. Neke rutine, kao ona za štampanje, nemaju nikakvih izlaznih parametara, ali, recimo, potprogram za traženje neke bejzik promenljive će nam, razume se, vratiti adresu te promenljive, ili će javiti da nije našao ono što je tražio.

I konačno, treba znati još i koji od procesorskih registara će u toku potprograma biti sačuvani. Ne retko ćemo sami morati da obavimo par PUSH instrukcija pre skoka, kako bismo sačuvali ono što nam je važno. Na primer:

```
PUSH    BC
PUSH    HL
CALL    ... NEKA ADRESA ...
POP     HL
POP     BC
```

Potprogram za štampanje &0010 ima tu lepu osobinu da čuva sve registre, pa nikakve posebne manevre ne moramo preduzimati. Ali, takvih programa je ipak malo. Šta više, ima i takvih kod kojih ne vredi ni pamtiti neke stvari pre poziva! Kakva je, recimo, korist od toga što upamtimo adresu bejzik linije 1000, ako pozovemo rutinu koja izbacuje ili ubacuje neke linije, pomerajući tako i liniju 1000? Treba, dakle, misliti na sve.

Izdvojili smo iz „spektrumovog" ROM-a samo najvažnije rutine, koje se jednostavno mogu koristiti. Za potpuno upoznavanje sa svim detaljima preporučujemo knjigu „The complete Spectrum ROM Disassembly", iz koje smo, inače, i preuzeli sve nazive potprograma. Nije loša ideja da bar proverite adrese, jer svaka štamparska greška može da zablokira vaš program i bez razloga vam pokvari raspoloženje.

Potprogrami su podeljeni u grupe prema poslu koji obavljaju. Iza imena svakog potprograma sledi naredba za poziv sa heksadekadnom adresom, a zatim tekst sa objašnjenjima i primerima.

Među ulazno-izlaznim parametrima se često javljaju i indikatori nule i prenosa (zero flag i carry flag), koje ćemo označavati sa „Z" i „C".

Izuzev ako posebno ne naglasimo suprotno, uvek se podrazumeva da stanje akumulatora na izlazu **nije očuvano.** Slično tome, registar IX će gotovo uvek biti očuvan. „Spektrum" koristi ovaj registar samo izuzetno i to ćemo naglasiti kada bude trebalo.

Neki potprogrami rade sasvim drugačije u fazi sintaksne provere bejzik naredbi nego u fazi **izvršenja** bejzik programa. „Spektrum" pravi razliku između ova dva slučaja tako što koristi sistemsku varijablu FLAGS: ako je njen sedmi bit setovan, onda je u toku izvršenje programa, a ako je resetovan, onda se samo proverava sintaksa. Za korisnika je značajniji prvi slučaj, jer se svi mašinski programi obično izvršavaju u okviru bejzika — konkretno za vreme izračunavanja funkcije **USR.** Mi, zato, uglavnom nećemo ni pominjati detalje vezane za sintaksnu proveru.

1. GLAVNA IZVRŠNA PETLJA

Čim uključimo „spektrum", on počinje da izvršava mašinski program iz ROM-a počev od adrese &0000. Tu ima niz poslova oko postavljanja sistemskiz varijabli i memorijske mape, a onda se ekran briše ispisuje poruka proizvođača i ulazi se u tzv. „glavnu izvršnu petlju". Računar u njoj ostaje praktično sve dok ga ne isključimo.

Nije teško zamisliti šta se dešava u okviru glavne petlje: „spektrum" prvo čeka da korisnik nešto ukuca i pritisne ENTER. Zatim proverava koliko tekst ima smisla po sintaksnim pravilima bejzika. Konačno, izvršava zadatu naredbu ili, ako je bila uneta nova programska linija, dodaje je postojećem programu. Bilo kako bilo, po ovavljenom poslu se ponovo odlazi na početak glavne izvršne petlje i tako sve u krug.

Glavna izvršna petlja nije nikakav potprogram. Korisnik može samo da ostvari nepovratan skok u nju, obično da bi izvršio resetovanje računara.

1. START — JP &0000

Skokom na adresu nula postiže se isti efekat kao kada se računar tek uključi. Uništava se sadržaj čitave memorije, a resetovanje je potpuno i efikasno.

2. NEW JP &11B7

Skok na &11B7 ekvivalentan je izvršenju naredbe NEW iz bejzika. Procesor prelazi u „interapt mod 1", a poništava se ceo bejzik prostor do RAMTOP-a.

3. MAIN_EXEC JP &12A2

To je početak glavne izvršne petlje. Obaviće se automatsko listanje bejzik programa, kao da je pritisnut **ENTER**, a u dnu ekrana će se pojaviti kursor.

4. MAIN_1 JP &12A9

Ulazom na &12A9 se jedino izbegava automatsko listanje, a inače je sve isto kao kod MAIN_EXEC.

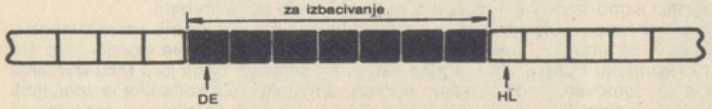
2. PRISTUP MEMORIJI

U ROM-u postoji niz potprograma pomoću kojih se vrlo prosto mogu ubacivati ili izbacivati proizvoljni blokovi memorije. To se, pre svega, odnosi na prostor sa bejzik linijama i varijablama, gde se stalno nešto dodaje ili oduzima. Nije, naravno, problem izbrisati neku liniju iz programa i ubaciti novu. Problem je obračunati novu dužinu programa i nove adrese svih blokova iza programa, da bi se odgovarajuće sistemske varijable u skladu sa tim izmenile. Bilo bi zaista preterano očekivati od korisnika da sam o tome vodi računa. Ovako, dovoljan je samo jedan CALL da bi se svi poslovi obavili.

1. RECLAIM_1 CALL &19E5

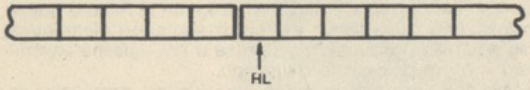
Potprogram RECLAIM_1 obavlja izbacivanje nekog memorijskog bloka. Na ulazu se u DE mora staviti adresa na kojoj počinje blok koji se izbacuje, a u HL mora biti adresa iza tog bloka, tj. prva adresa koju ne treba izbaciti. (sl. 16.)

Slika 16. RECLAIM_1 (ulaz)



Na izlazu (sl. 17.) HL će sadržati novu adresu bloka koji se nalzio iza izbačenog (to je zapravo ista adresa koja je bila u DE na ulazu).

Slika 17. RECLAIM_1 (izlaz)



U toku potprograma procesorski registri nisu očuvani, a na izlazu je BC uvek nula.

Uzećemo kao primer da iz bejzik programa treba izbaciti neku liniju koja počinje na adresi &5DOO. Pri tom je, recimo, adresa sledeće linije &5DFF. Sve što treba uraditi je pripremiti HL i DE, i onda pozvati potprogram.

```
4.   LD      DE,&5D00
     LD      HL,&5DFF
     CALL    &19E5
```

Na izlazu će linija iza izbačene biti pomerena na &5D00 i HL će sadržati tu adresu.

Recimo, učitaćemo blok bajtova koji smo snimili u prethodnom primeru:

```
10.  LD      A, &FF
     SCF
     LD      DE, 30000
     LD      IX, 25000
     CALL    &0556
```

Na izlazu registri nisu očuvani, a indikator prenosa „C" je setovan ako je operacija obavljena uspešno do kraja. Međutim, ukoliko dobijemo C=0, to znači da je nastupila neka greška: ili tip podataka na kaseti ne odgovara sadržaju akumulatora na ulazu, ili bajtova više nema, a očekivani broj je bio veći, ili, konačno, provera sa kontrolnim bajtom nije uspela.

Pritisak na taster BREAK_SPACE u toku čitanja kasete izaziva prekid „D BREAK-CONT repeats". Što se tiče mašinskih interapta, važi isto kao i za SA_BYTES: na izlazu se uvek obavlja Ei.

4. KONTROLA EKRANA

1. CLS CALL &0D6B

Ovaj potprogram je, u stvari, izvršna rutina bejzik naredbe CLS i obavlja brisanje celog ekrana. Nikakvih ulaznih parametara nema, a registri na izlazu nisu očuvani. Pozicija za **PRINT** se postavlja u gornji levi ugao ekrena, a za **PLOT** u donji levi ugao.

2. CLS_LOWER CALL&0D6E

Rutina CLS_LOWER služi samo za brisanje donjeg dela ekrana. Koliko će linija time biti obuhvaćeno — određeno je sadržajem sistemske varijable DF_SZ, koji možemo i sami menjati. Na primer, ako treba izbrisati linije 17—21, dakle donjih pet na glavnom delu ekrana, onda, imajući u vidu još i dve linije u donjem delu ekrana, pre poziva CLS_LOWER treba u DF_SZ ubaciti broj sedam.

```
11.  LD      (IY+49), &07
     CALL    &0D6E
```

Ulaznih i izlaznih parametara nema, a registri neće biti očuvani.

3. CL_SCROLL CALL &0E00
CL_SC_ALL CALL &0DFE

Pomoću potprograma CL_SCROLL vrši se vertikalno pomeranje ekrana za jedan red naviše — „skrolovanje". Jedini ulazni parametar je broj redova koji će učestvovati u pomeranju i treba ga smestiti u registar B. Na primer, ako u B ubacimo broj petnaest, to znači da će 15 donjih redova (13 iz gornjeg i 2 iz donjeg dela ekrana) krenuti naviše. Linija 8 pri tome biva prebrisana (na njeno mesto dolazi linija 9), a linije 0—7 ostaće nedirnute:

```
12.  LD      B, &0F
     CALL    &0E00
```

Izlaznih parametara nema. Registri nisu očuvani.

Napominjemo da po obavljenom skrolovanju pozicija za štampanje **neće biti vraćena na red 21,** već to moramo sami uraditi, recimo sa **PRINT AT 21,0;** u bejziku, ili na odgovarajući način u mašinskom jeziku, kao što će kasnije biti objašnjeno.

Ulaz na CL_SC_ALL koristi se za pomeranje **čitavog ekrana.** Nikakav ulazni parametar nije potreban, jer se automatski obavlja naredba **LD B, &17.**

4. PIXEL_ADD CALL &22AA

Svaka tačka na ekranu predstavljena je jednim bitom u video memoriji. Potprogram PIXEL_ADD pronalazi u memoriji odgovarajući bit za proizvoljnu tačku sa koordinatama „x" i „y". Na ulazu mora biti C=x, a B=y. Na izlazu će HL sadržati adresu bajta, a akumulator će određivati poziciju bita u okviru tog bajta: bit=7—A. Očuvan je registar DE.

Ukoliko je na ulazu „y" veće od 175, imaćemo grešku „B Integer out of range".

4. BEEPER CALL &03B5
BEEP CALL &03F8

Potprogram BEEPER koristi se za generisanje tona na „spektrumovom" ugrađenom zvučniku. Na ulazu se zadaje trajanje tona i njegova visina, ali ne baš tako jednostavno kao u bejziku.

Ako treba odsvirati ton frekvencije „f" (u hercima) i trajanja „t" (u sekundama), onda sadržaj registara na ulazu mora biti:

$$DE = f*t$$
$$HL = 437500/f - 30.125$$

Naravno, rezultate treba zaokružiti.

Frekvencije tonova u temperovanoj lestvici dobijaju se po formuli:

$$f = 440*EXP((n-9)*l\ n\ 2/12),$$

gde je „n" ceo broj koji odgovara datom tonu, kao u slučaju bejzik naredbe BEEP. Recimo, da bismo iz mašinskog programa proizveli notu „C" (n=0, f=261.63 Hz), moramo postupiti kao u primeru 8.

```
8.  LD      DE, 262
    LD      HL, 1642
    CALL    &03B5
```

Nema izlaznih parametara, registri nisu očuvani, a čak se koristi i registar IX. Na izlazu su A i DE uvek nule. Pre povratka se uvek obavlja mašinska instrukcija EI, a u toku same rutine prekidi su onemogućeni. Ako vaš program zahteva isključene prekide, moraćete posle svakog poziva potprograma BEEPER ponovo da izvršite DI.

Drugi način generisanja zvuka je korišćenjem potprograma BEEP. To je izvršna rutina odgovarajuće bejzik naredbe, i radi malo sporije. Izlaz je isti kao i iz BEEPER, ali ulaznih parametara nema. Umesto toga, trajanje note i njena visina moraju se nalaziti na računskom steku (visina note poslednja). O upotrebi steka tek ćemo govoriti.

5. SA_BYTES CALL &04C2

Ova rutina obavlja snimanje blokova bajtova na kasetu. Nju naredba **SAVE** normalno poziva dva puta: prvo da bi snimila zaglavlje, a zatim i sam programski blok.

Na ulazu akumulator sadrži &00 ako se snima zaglavlje, a &FF ako se snima blok. U oba slučaja registar DE sadrži broj bajtova koji će se slati na kasetu, a IX njihovu početnu adresu. Izlaznih parametara nema. Registri nisu očuvani, a operacija se u svakom trenutku može prekinuti pritiskom na taster BREAK-SPACE, što dovodi do raporta „D BREAK-CONT repeats".

Na kasetu se prvo snima jedan vodeći signal konstantnog tona, koji traje oko 5 s za zaglavlje, a oko 2 s za programski blok. Prvi bajt koji se snima je oznaka tipa (zaglavlje ili blok), a zatim slede i sami bajtovi sa adrese IX. Iza njih se ubacuje još jedan kontrolni bajt dobijen logičkom operacijom **XOR** između svih snimljenih bajtova.

Kao primer ćemo snimiti na kasetu bajtove koji počinju na adresi 25000, sa dužinom 30000. Snimamo samo blok — bez zaglavlja:

```
9.  LD      A,&FF
    LD      DE, 30000
    LD      IX, 25000
    CALL &04C2
```

Ovako snimljen program ne može se učitati iz bejzika, jer nema zaglvalja. Još bolja zaštita je ako ceo blok snimimo pod oznakom zaglavlja (samo umesto **LD A, &FF** treba staviti **SUB A).** To će čak zbuniti i mnoge programe za presnimavanje!

U toku snimanja su onemogućeni mašinski prekidi, a na izlazu se obavlja EI.

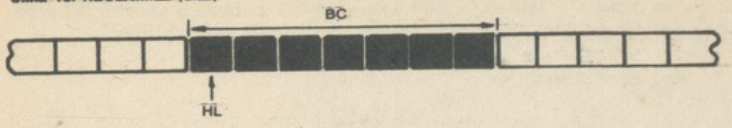
6. LD_BYTES CALL &0556

Potprogram LD_BYTES koristi se za učitavanje ili verifikaciju bajtova snimljenih na kaseti. Na ulazu akumulator sadrži &00 za zaglavlje, a &FF za blok. Indikator prenosa „C" je setovan za učitavanje a resetovan za verifikaciju. Registar DE sadrži broj bajtova koji se očekuju sa trake, a u IX je početna adresa u memoriji gde treba uneti učitane bajtove, ili odakle ih treba verifikovati.

2. RECLAIM_2 CALL &19E8

RECLAIM_2 je, u stvari, **isti** potprogram kao i RECLAIM_1, samo što se ulazni parametri razlikuju: sada HL treba da pokazuje na početak bloka koji se izbacuje, a u BC se mora smestiti dužina bloka. (sl. 18.)

Slika 18. RECLAIM_2 (ulaz)



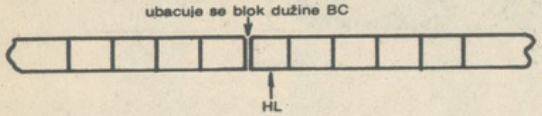
Izbacivanje bejzik linije sa adrese &5D00 sada bi se izvelo kao u sledećem primeru:

```
5.  LD      HL, &5D00
    LD      BC,&00FF
    CALL    &19E8
```

3. MAKE_ROOM CALL &1655

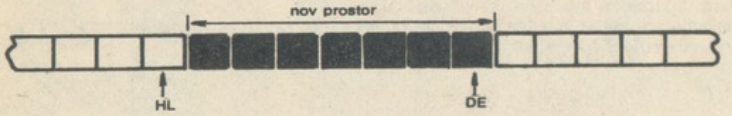
Potprogram MAKE_ROOM obavlja umetanje memorijskog bloka, pomerajući pri tome ostatak prostora udesno za potreban broj bajtova. Na ulazu HL sadrži adresu na kojoj će počinjati nov prostor, a u BC se nalazi veličina tog prostora.

Slika 19. MAKE_ROOM (ulaz)



Na izlazu HL pokazuje na bajt ispred novog prostora, a DE pokazuje na poslednji bajt novog prostora. Akumulator je očuvan a BC sadrži uvek nulu.

Slika 20. MAKE_ROOM (izlaz)



Kao primer, umetnućemo prostor dužine &CO na mesto izbačene linije sa adrese &5D00 (primer 6.).

```
6. LD       HL,&5D00
   LD       BC, &00C0
   CALL     &1655
```

Naravno, tek treba u taj prostor upisati tekst linije.

4. ONE_SPACE CALL &1652

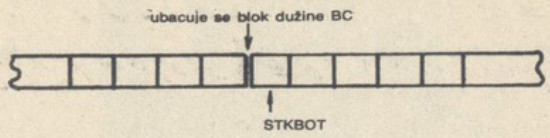
Kada treba napraviti mesta samo za jedan bajt, dovoljno je samo ubaciti adresu u HL i odmah pozvati ONE_SPACE. To je u stvari, ista rutina kao i MAKE_ROOM, samo što sadrži i dodatnu naredbu **LD BC, &0001.**

5. BC_SPACES RST &30

Kao što smo ranije već rekli, između adresa WORKSP i STKBOT u „spektrumovoj" memoriji nalazi se **radni prostor** za privremeno smeštanje podataka. Pre upotrebe, radni prostor se mora „otvoriti", odnosno u njega se mora umetnuti potreban broj bajtova, potpuno slično kao u slučaju MAKE–ROOM.

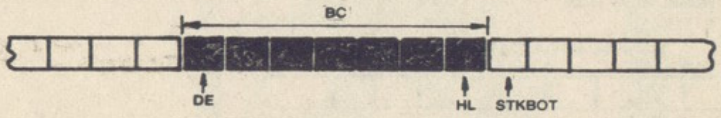
Otvaranje radnog prostora vrši se pozivom potprograma BC_SPACES. Jedini ulazni parametar je dužina bloka koji želimo da umetnemo, a treba da bude u registru BC. Novi bajtovi će biti umetnuti u samom vrhu radnog prostora, odmah ispod STKBOT.

Slika 21. BC_SPACES (ulaz)



Na izlazu DE pokazuje na prvi, a HL na poslednji bajt novog prostora. Akumulator je očuvan, a BC i dalje sadrži dužinu bloka.

Slika 22. BC_SPACES (izlaz)



6. SET_MIN CALL &16B0
SET_WORK CALL &16BF
SET_STK CALL &16C5

Potprogram SET_MIN koristi se za istovremeno brisanje editnog prostora, radnog prostora i računskog steka. Čitav prostor iza E_LINE će, tako, zauzeti minimalnu konfiguraciju.

Ako treba obrisati samo radni prostor i računski stek, poziva se SET_WORK, a za brisanje samog računskog steka koristi se SET_STK.

Ulaznih parametara nema, a u sva tri slučaja HL na izlazu sadrži novu vrednost STKEND. Ostali registri, uključujući i akumulator, ostaju očuvani.

7. TEST_ROOM CALL &1F05

Potprogram TEST_ROOM koristi se za proveru raspoloživog memorijskog prostora, dostupnog bejziku. Na ulazu BC sadrži broj bajtova sa kojim se slobodna memorija testira. Ako ima toliko praznih mesta, uslediće povratak, sa očuvanim registrima A i BC. U protivnom, obaviće se skok na izdavanje raporta „4 Out of memory".

Rutine MAKE_ROOM i BC_SPACES same pozivaju TEST_ROOM da bi proverile teren, tako da mi to ne moramo činiti.

3. PERIFERIJE

1. CHAN_OPEN CALL &1601

„Spektrum obavlja ulazno-izlazne operacije sistemom kanala i struja, o čemu smo već i govorili. Da bi se preciziralo kuda će otići izlazni podaci, ili odakle će se uzeti ulazni podaci, mora se otvoriti odgovarajući kanal koji vodi do periferijskog uređaja. To je sve, naravno, samo slikovit opis. U suštini, otvaranje kanala se svodi na postavljanje sistemske varijable CURCHL i setovanje nekih drugih indikatora (FLAGS i TVFLAG).

Otvaranje kanala obavlja se pozivom rutine CHAN_OPEN. Ulazni parametar je numerička oznaka struje koja odgovara datom kanalu, a treba je smestiti u akumulator. Kao primer, ranije samo već imali otvaranje kanala „S" korišćenjem struje &02. Potpuno isti efekat bi se postigao i sa strujom &FE. Sve ovo važi, naravno, samo ako ne vršimo izmene u oblasti struja i kanala u RAM-u (adrese STRMS i CHANS).

Izlaznih parametara nema, a jedino je registar B očuvan.

2. WAIT_KEY CALL &15D4

Potprogram WAIT_KEY, jednostavno, čeka da korisnik pritisne neki taster i tek tada se vrši povratak. To najčešće ima primenu u situacijama tipa „Press any key . . .", kada se prvo u dnu ekrana ispiše neka poruka i onda čeka akcija korisnika.

O ispisivanju poruka ćemo govoriti kasnije. Sada je važno upamtiti da pre poziva WAIT_KEY obavezno treba setovati bit 5 u sistemskoj varijabli TVFLAG, jer će se inače u dnu ekrana pojaviti kopija bejzik linije iz editnog prostora, što verovatno ne želimo.

```
7.    SET  5, (IY+2)
      CALL &15D4
```

Ulaznih parametara nema, svi registri ostaju očuvani, a akumulator na izlazu sadrži ASCII kôd ukucanog znaka (istovremeno se taj kôd upisuje i u sistemsku varijablu LAST_K). Osim toga, zahvaljujući naredbi **SET 5, (IY+2)**, donji deo ekrana će biti automatski izbrisan čim taster bude pritisnut.

Podrazumeva se da je pre poziva WAIT_KEY negde bio otvoren kanal „K" (a sigurno će i biti, ako već štampamo poruku u dnu ekrana). U protivnom ćemo imati grešku „J Invalid I/O device".

3. KEY_SCAN CALL &028E

Ovaj potprogram obavlja očitavanje tastature, ali za razliku od WAIT_KEY ne čeka ništa, već se vraća čim proveri svih 40 tipki. Na izlazu ćemo tako dobiti podatak o tome da li je trenutno neki taster bio pritisnut ili ne.

Kodovi koje vraća KEY_SCAN ne odgovaraju ASCII znakovima, već **neposredno samim tasterima.** Na primer, ako je pritisnuta tipka „A", dobićemo na izlazu jedinstven broj bez obzira na to što se isti taster može tumačiti i kao „a", ili kao „NEW", „READ", itd. Tek naknadno, ako nam je potrebno, možemo dobijene kodove pretvarati u ASCII znakove, znajući stanje kursora. Međutim, sasvim je dovoljno u većini slučajeva znati samo koji od 40 tastera je pritisnut, da bi se preduzela odgovarajuća akcija.

Tipke su podeljene u osam grupa od po pet, pri čemu spoljašnji taster u svakoj grupi ima najveći kôd, a idući ka unutrašnjosti tastature kôd opada sa korakom osam. Svi kodovi su dati u tabeli 1.

Tabela 1.

BREAK-SPACE	.B	&20	.&00
ENTER	.H	&21	.&01
P	.Y	&22	.&02
0	.6	&23	.&03
1	.5	&24	.&04
Q	.T	&25	.&05
A	.G	&26	.&06
CLAPS SHIFT	.V	&27	.&07

Pre poziva KEY_SCAN ne moraju se vršiti nikakve pripreme, niti treba voditi računa o kanalima. Ulaznih parametara nema, a registri na izlazu nisu očuvani. Kodovi pritisnutih tastera biće smešteni u D i E, a indikator nule „Z" biće resetovan ako je pritisnuto više od dva tastera, ili su pritisnuta dva a ni jedan od njih nije CAPS ili SYMBOL SHIFT. Samo ako je pritisnut jedan taster, eventualno sa SHIFT-om, biće Z=1. Sadržaj registra DE je sledeći:

a) Ako nije pritisnut ni jedan taster, oba registra sadrže &FF. Dovoljno je testirati samo E.

b) Ako je pritisnut samo jedan taster, makar i neki od SHIFT-ova, onda E sadrži njegov kôd, a D i dalje &FF.

c) U slučaju dva pritisnuta tastera, jedan ide u E, a drugi u D. Bilo koji SHIFT će obavezno ići u D, a ako su oba tastera SHIFT-ovi, onda CAPS ide u D.

Spektrum

PRINTER

Svi koji su bili u prilici da prisustvuju dodeli nagrada na prošlom „Galaksijinom" Konkursu za najboljeg YU programera, mogli su da upoznaju jednog dečaka, čiji rad, istina, nije svrstan među četiri najbolja (pre svega zbog nepotpune dokumentacije), ali koji je zato izazvao veoma povoljne reakcije za vreme demonstracije. Reč je o Aronu Bošnjaku i njegovom programu „Mastermajnd".

Prisutni su, slobodno možemo reći, bili zadivljeni pojedinim efektnim rešenjima kojima je autor obogatio ovu popularnu igru, učinivši je tako po mnogo čemu različitom (i boljom!) od nekih drugih, pa čak i komercijalnih rešenja.

Aron Bošnjak je osvojio sedmo mesto, a mi se iskreno nadamo da njegov „Mastermajnd" ipak neće pasti u zaborav. Da bismo vam dočarali bar nešto od onoga što ovaj program pruža, odlučili smo da izdvojimo jednu kratku mašinsku rutinu, koju je autor koristio za ispisivanje uputstava pre početka same igre. Slova munjevito doleću sa desne strane ekrana, i uz karakterističan zvuk, zauzimaju svoja mesta, jedno za drugim, formirajući reči. Ideja je veoma duhovita i originalna, a iz nje se dosta toga može i naučiti. Zato smo odlučili da program izlistamo u asembleru (DEVPAC GENS). Morali smo da izvršimo i neke manje izmene u originalnoj rutini, pre svega da bismo dobili nezavisan potprogram, koji bi se jednostavno pozivao iz bejzika. Budući da kao ulazni parametar mora figurisati neki niz znakova koji će se štampati, iskoristili smo metod DEF FN funkcija, o čemu smo već pisali u „Računarima 3".

Naravno, program možete koristiti i bez njegovog analiziranja. Jedino ga morate ukucati, liniju po liniju, i zatim asemblirati (sami izaberite adresu na kojoj će se formirati mašinski kôd i unesite odgovarajuću ORG naredbu).

Ako je mašinski program unet, recimo, na adresu 50000, onda u bejziku treba ukucati liniju:

10 DEF FN p(a$) = USR 50000

Ime funkcije i argumenta, naravno, može biti i drugačije, a adresa iza USR obavezno mora odgovarati prvom bajtu mašinskog programa.

Time su sve pripreme obavljene. Razume se, mašinski kôd treba snimiti na kasetu, za kasniju upotrebu (dužina bloka je 117 bajtova).

Tekst koji se štampa pomoću rutine „PRINTER" pojaviće se na ekranu na istom mestu gde bi ga inače smestila naredba PRINT. To znači da se početna pozicija teksta može unapred definisati uobičajenim postupkom, pomoću PRINT AT ili PRINT TAB. Isto tako, po završenom štampanju, PRINT pozicija će se nalaziti odmah iza poslednjeg odštampanog znaka.

Recimo da treba ispisati tekst „Galaksija". Tada prosto treba izvršiti RANDOMIZE FN p(„Galaksija"). Ili, ako je tekst koji se štampa smešten u varijabli h$, otkucaćemo RANDOMIZE FN p(h$).

Tekst duži od 32 znaka će se prenositi u nov red. Međutim, treba znati da će pokušaj štampanja iza 21. reda dovesti do prekida sa izveštajem „5 Out of screen", a ne do poruke „scroll?", što bi se inače moglo očekivati. Razlog je u tome što program „PRINTER" koristi zapravo naredbu PRINT AT u toku svog rada, a ova naredba, kao što korisnici „spektruma" dobro znaju, odbija da radi sa 22. redom.

Primetićete da se od svih znakova jedino prazna polja (blankovi) ne ubacuju sa strane. Oni se tretiraju malo drugačije i ne bi ih trebalo stavljati na kraj niza koji se štampa, jer će se inače poremetiti PRINT pozicija za sledeće štampanje.

Jovan Skuljan

Program „PRINTER"
Autor: **Aron Bošnjak**

```
START   LD      A,&02       Biće otvoren kanal „S" (glavni deo ekrana)
        CALL    &1601       Otvaranje kanala.
        LD      IX,(&5C0B)  DEFADD: adresa parametra funkcije FN p.
        LD      L,(IX+4)    Adresa niza koji se štampa
        LD      H,(IX+5)    Ide u registar HL.
        LD      C,(IX+6)    Dužina niza koji se štampa
        LD      B,(IX+7)    ide u registar BC.
        LD      A,B         Da li je dužina niza
        OR      C           jednaka nuli?
        RET     Z           Ako jeste — povratak u bejzik.
        LD      A,24        Iz varijable S—POSN (viši bajt) uzima se
        SUB     (IY+79)     linija na kojoj će se štampati sledeći
        LD      E,A         znak. Rezultat se smešta u E.
        LD      A,33        Iz varijable S—POSN (niži bajt) uzima se
        SUB     (IY+78)     kolona na kojoj će se štampati sledeći
        LD      D,A         znak. Rezultat se smešta u D.
        SUB     32          Da li je kolona 32?
        JR      NZ,CONT     Ako nije idi napred.
LOOP3   LD      D,A         Broj kolon se postavlja na nulu,
        INC     E           a prelazi se u sledeći red.
CONT    DEC     D           Kolona koja prethodi tekućoj poziciji
        LD      (IY+71),D   smešta se u brojač.
LOOP2   LD      D,31        Ubacivanje slova se vrši sa desne strane.
LOOP1   LD      A,&16       To je kontrolni znak „AT".
        RST     &10         PRINT AT.
        LD      A,E         Broj reda se
        RST     &10         „štampa" iza „AT" simbola.
        LD      A,D         Broj kolone se
        RST     &10         „štampa" iza broja reda.
        LD      A,(HL)      Uzima se karakter iz niza koji se štampa.
        CP      &20         Da li je to blank?
        JR      Z,INSN      Ako jeste — idi napred. Nema ubacivanja.
        RST     &10         Štampanje karaktera.
        CALL    TONE        Zvučni efekat.
        LD      A,D         Testiranje kolone:
        CP      31          Da li je to poslednja kolona?
        JR      Z,NEXTLP    Ako jeste — idi napred.
        LD      A,&20       Brisanje traga koji ostavlja za sobom slovo
        RST     &10         vrši se štampanjem blanka.
        LD      A,&08       PRINT pozicija se odmah vraća
        RST     &10         za jedno mesto ulevo, iza poslednjeg slova.
NEXTLP  DEC     D           Slova klize ulevo.
        LD      A,D         Testiranje kolone: da li je slovo stiglo
        CP      (IY+71)     na svoje mesto u tekstu?
        JR      NZ,LOOP1    Ako nije — idi nazad u petlju.
INSN    INC     (IY+71)     Odštampano je još jedno slovo.
        INC     HL          Pomeranje na sledeći znak u nizu.
        DEC     BC          Brojač dužine niza se umanjuje za jedan.
        LD      A,B         Da li su odštampani svi
        OR      C           znakovi u nizu?
        RET     Z           Povratak ako jesu.
        LD      A,(IY+71)   Da li je popunjen
        SUB     31          čitav red na ekranu?
        JR      NZ,LOOP2    Ako nije — idi nazad u petlju.
        JR      LOOP3       Skok nazad, sa prelaskom u nov red.
TONE    PUSH    BC          Sačuvaj brojač dužine.
        PUSH    DE          Sačuvaj kolonu i liniju.
        PUSH    HL          Sačuvaj adresu znaka u nizu.
        LD      A,32        Visina tona će biti određena
        SUB     D           pomoću kolone na kojoj je
        RRA                 obavljeno štampanje.
        LD      H,A         Dobijeni rezultat ide u HL.
        LD      DE,&0001    Trajanje tona ide u DE.
        CALL    &03B5       BEEP.
        POP     HL          Obnavljanje registara.
        POP     DE
        POP     BC
        RET                 Povratak.
```

Galaksija

ŠIFRA

Zaštitom informacija bavili su se još stari Grci i Rimljani... Počelo je sa zaštitom (šifriranjem) pisanih poruka, ali danas to se proširilo na sve vrste informacija — govor, sliku, prenos podataka i sl.

Ovom problematikom su se pre svega, bavili matematičari, ali su im u tome, u dvadesetom veku pomogli, mašinci sa mehaničkim šifranskim mašinama, a zadnjih decenija i elektroničari.

Cilj ovog programa je da demonstrira mogućnost upotrebe računara „galaksija" i za ovakvu namenu.

Pošto se program unese i startuje, potrebno je tekst koji se šifrira ili dešifrira uneti prema rasporedu slova na tasteru „galaksije". Šifrirana odnosno dešifrirana informacija se pojavljuje na ekranu televizora — monitora. Svako slovo treba držati pritisnuto cca 1 do 2 sekunde (zbog petlje u liniji 199), što je učinjeno namerno jer prilikom skeniranja tastature vreme registrovanja pritiska na pojedine tastere nije isto, pa je dolazilo do pojave dupliranja slova na ekranu.

Program je napravljen tako da se ista rutina koristi i za šifriranje i za dešifriranje. Sam „ključ" je moguće izmeniti tako što, na primer, sva slova pomerimo za jedno mesto nadole, a slovo „M" u red 101 tako da bude:

101 IFK.(1)P.„M";
102 IFK.(2)P.„N";
103 IFK.(3)P.„O";
104 IFK.(4)P.„P"; itd.

Reči se rastavljaju pomoću „blanka", pri čemu se na ekranu pojavljuju tačke. Ispisani tekst na ekranu briše se pomoću horizontalne strelice koja pokazuje ulevo.

I, na kraju, jedna „zaštićena" poruka:CBMQENI... PVGNBPVZN... ENPHANEN...

Ivan Nador

```
50 C.500
101 IFK.(1)P."N";
102 IFK.(2)P."O";
103 IFK.(3)P."P";
104 IFK.(4)P."R";
105 IFK.(5)P."S";
106 IFK.(6)P."T";
107 IFK.(8)P."U";
109 IFK.(9)P."V";
110 IFK.(10)P."W";
111 IFK.(11)P."X";
112 IFK.(12)P."Y";
113 IFK.(13)P."Z";
114 IFK.(14)P."A";
115 IFK.(15)P."B";
116 IFK.(16)P."C";
117 IFK.(17)P."D";
118 IFK.(18)P."E";
119 IFK.(19)P."F";
120 IFK.(20)P."G";
121 IFK.(21)P."H";
122 IFK.(22)P."I";
123 IFK.(23)P."J";
124 IFK.(24)P."K";
125 IFK.(25)P."L";
126 IFK.(26)P."M";
131 IFK.(31)P.".";
132 IFK.(32)P."4";
133 IFK.(33)P."5";
134 IFK.(34)P."8";
135 IFK.(35)P."7";
136 IFK.(36)P."0";
137 IFK.(37)P."1";
138 IFK.(38)P."9";
139 IFK.(39)P."3";
140 IFK.(40)P."2";
146 IFK.(41)P."6";
188 IFK.(29)C.530
199 F.X=0 TO 100:N.X
200 G.100
500 H.
510 P.AT 134,"*** SIFRA ***"
520 P.AT 232,"NADOR IVAN"
521 P.AT 329,"JUN 1985"
523 P.AT 394,"V. 3.1"
525 F.X=0 TO 400:N.X
530 H.
540 P."SIFRIRANJE ILI DESIFRIRANJE SE VRSI TAKO STO SE TEKST U KUCAVA GLEDAJUCI U TASTATURU,POLAKO,"
550 P."SLOVO PO SLOVO,SIFRIRANI/DESIF- RIRANI TEKST SE POJAVLJUJE NA EKRANU."
640 P.AT 224,"TEKST SE BRISE STRELICOM"
655 P.AT 264,"NA LEVO <--"
660 P.
700 R.
```

Komodor 64

ARHIVA

Ako posedujete disk jedinicu, sigurno imate bar jednu disketu popunjenu velikim brojem programa, radnim verzijama i tekstovima koje retko koristite, ili vam više neće trebati, ali biste jednostavno voleli da ih sačuvate. Ovim programom se takve diskete mogu kopirati na kasetu i kasnije, ako vam zatrebaju, ponovo vratiti nazad. U program je ugrađena i opcija kopiranja diskete na disketu.

Kopiranje se vrši blok po blok i to cele diskete bez obzira da li je blok stvarno zauzet. Pošli smo od pretpostavke da će se na ovaj način kopirati samo popunjene diskete, pa bi izbegavanje slobodnih blokova nepotrebno komplikovalo program.

Program je (za razliku od mnogih) „miran" u radu, daje obaveštenja o toku (broj bloka koji se trenutno kopira), razlikuje programe na traci od kopija diskete i može se prekinuti STOP tasterom. Proces kopiranja traka-disk ili obratno odvija se uz samo jednu intervenciju i traje oko 15 minuta. Disk-disk zahteva 4 izmene diskete i traje oko 20 minuta.

Kada unesete čitav program, otkucajte RUN i računar će ispisivati kontrolne sume koje treba da uporedite sa datom listom. Ako se neka ne slaže, odgovorite sa „N" i na ekranu će biti izlistana grupa linija u kojoj je greška. Kada su sve kontrolne sume u redu, računar će sam izvesti proveru ukupnog zbira.

Sada program OBAVEZNO snimite na disketu i u direktnom modu otkucajte:

NEW
POKE25600,0
POKE44,100
NEW

i ponovo upišete program. Nakon:

RUN 20

dobijate gotov oblik koji se snima i koristi uobičajenim procedurama.

```
1 REM"———— A R H I V A ————
2 REM
3 REM      C64 & 1541 & DATASSETTE
4 REM
5 REM AUTOR:  ZORAN ZIVOTIC  (85)
6 REM"————————————————————
7 REM PROVERA DATA LINIJA:
8 REM"————————————
9 RESTORE
10 FORI=100TO356STEP4:S=0:FORJ=1TO32
11 READB:S=S+B:NEXTJ:PRINT"S"I/4-25"="S;
12 A$="D":INPUT" OK";A$
13 IFA$="D"GOTO16
14 PRINT"LIST"I"-"I+3:POKE631,19
15 POKE632,13:POKE198,2:SYS42115
16 NEXTI:PRINT"1. OK"
17 RESTORE:S=0:FORI=1TO2064:READB:S=S+B
18 NEXT:IFS<>21838ITHENPRINT"GRESKA!":END
19 PRINT"2. OK, CESTITAM":END
20 REM KREIRANJE PROGRAMA:
21 REM"————————————
22 RESTORE:FORI=2048TO4127:READB:POKEI,B
23 NEXTI
24 POKE44,8:POKE45,9:POKE46,16:CLR:LIST
100 DATA  0, 40,  8,193,  7,158, 50, 48
101 DATA 57, 48, 20, 20, 20, 20, 20, 20
102 DATA 20, 20, 46,173,173, 65, 82, 72
103 DATA 73, 86, 65,173,173, 32, 90, 46
104 DATA 90, 73, 86, 79, 84, 73, 67,  0
105 DATA  0,  0,186,142, 19,  3,169, 14
106 DATA141, 32,208,141, 33,208, 32,140
107 DATA 10,162,  0, 32,156, 14,169,  0
108 DATA133,198, 32,228,255,240,251,201
109 DATA 49,144, 36,201, 53,176,243, 72
110 DATA 32,210,255, 32,143, 10, 32,143
111 DATA 10,104,201, 49,208,  3, 76, 68
112 DATA  9,201, 50,208,  3, 76,165,  9
113 DATA201, 51,208, 30, 76, 18,  9,201
114 DATA  3,208,207, 76,116,164, 32,228
115 DATA255,240,251,208,188,  8, 32,117
116 DATA 10, 40,174, 19,  3,154,176,238
117 DATA144,175, 32, 13, 10,162,  1, 32
118 DATA156, 14, 32, 96,165,232,134,187
119 DATA200,132,188,162,255,232,189,  0
120 DATA  2,208,250,134,183, 32,143, 10
121 DATA173,  0,  2,201, 36,240, 24,162
122 DATA127, 32,201,255,162, 10, 32,156
123 DATA 14, 32,204,255, 56, 32, 74, 10
124 DATA 32,117, 10, 56, 76,125,  8,169
125 DATA126,162,  8,160,  0, 32,186,255
126 DATA 32,192,255,169,  0,133,144, 32
127 DATA146, 10,162,126, 32,198,255, 32
128 DATA 52, 10, 32, 52, 10,173,141,  2
129 DATA208,251, 32,155, 10, 32, 52, 10
130 DATA 32, 52, 10, 32, 52, 10, 72, 32
131 DATA 52, 10,168,104,170,152, 32,205
132 DATA189, 32,155, 10, 32, 52, 10, 32
133 DATA210,255,208,248, 32,143, 10, 76
134 DATA229,  8,162,  4,134,247,169,  1
135 DATA133,250, 32,131, 10, 32, 65, 11
136 DATA 32, 10, 11, 32,131, 10, 32, 65
137 DATA 11, 32, 38, 12,165,251,133,250
138 DATA198,247,208,230,169, 73,141,  0
139 DATA  2,169,  0,141,  1,  2, 32, 13
140 DATA 10, 76,175,  8,162,  8, 32,156
141 DATA 14, 32, 96,165,162,255,232,189
142 DATA  0,  2,157,  7, 16,208,247,224
143 DATA  2,176,  3, 76, 57,  8,134,248
144 DATA 32,149, 10, 32, 56,248, 32,137
145 DATA 12,162,  4,134,247,169,  1,133
146 DATA250, 32,131, 10, 32, 65, 11, 32
147 DATA 15, 11,165,251,133,250,169,  0
148 DATA133,172,169, 20,133,173,165,253
149 DATA133,174,165,254,133,175,169,  7
150 DATA133,187,169, 16,133,188,165,248
151 DATA133,183, 32,146, 13,198,247,208
152 DATA208, 24, 76,125,  8,162,  4,134
153 DATA247,162,  3, 32, 99, 11,169,  1
154 DATA133,250, 32, 16, 14,169, 12,170
155 DATA168,141, 53,  3,202,208,253,136
156 DATA208,250,206, 53,  3,208,245,169
157 DATA  0,133,172,169, 20,133,173, 32
158 DATA131, 10,173, 62,  3,133,174,173
159 DATA 63,  3,133,175, 32,100, 14,165
160 DATA144,240, 25,162,  9, 32,156, 14
161 DATA162,  4, 32,156, 14, 32,228,255
162 DATA201, 68,240,  8,201, 78,208,245
163 DATA 24, 76,125,  8, 32, 65, 11, 32
164 DATA 43, 12,165,251,133,250,198,247
165 DATA208,168, 76, 52,  9,169,  0,133
166 DATA183,169,127,162,  8,160, 15, 32
167 DATA186,255, 32,192,255, 96,169,126
168 DATA162,  8,160,  2, 32,186,255,169
169 DATA  1,162,195,160, 14, 32,189,255
170 DATA 32,192,255, 96, 32,207,255,166
```

```
171 DATA144,208,  1, 96, 32,204,255, 56
172 DATA 32, 74, 10, 32,117, 10, 56, 76
173 DATA125,  8,  8,162,127, 32,198,255
174 DATA160,  0, 32,207,255,153,  0,  2
175 DATA200,201, 13,208,245,169,  0,153
176 DATA  0,  2, 32,204,255, 40,176,  1
177 DATA 96, 32,149, 10, 32,143, 10,162
178 DATA 10, 32,156, 14, 96, 32,204,255
179 DATA169,126, 32,195,255,169,127, 32
180 DATA195,255, 96,169,  0,133,253,169
181 DATA 20,133,254, 96,169,147, 44,169
182 DATA 13, 44,169,  5, 44,169,144, 44
183 DATA169, 46, 44,169, 32, 44,169,145
184 DATA 32,210,255, 96,165,251,162,  7
185 DATA 32,242, 10,165,252,162, 10, 32
186 DATA242, 10,162,127, 32,201,255,162
187 DATA 11, 32,156, 14, 32,204,255, 24
188 DATA 32, 74, 10,173,  0,  2,201, 50
189 DATA144,  5, 32,105, 10, 56, 96,162
190 DATA126, 32,198,255,160,  0,165,  1
191 DATA 41,254,133,  1, 32,207,255,145
192 DATA253,200,208,248, 32,204,255,165
193 DATA  1,  9,  1,133,  1, 32,120, 11
194 DATA 24, 96,168,169, 48,157,242, 15
195 DATA152,201, 10,144,  7,254,242, 15
196 DATA233, 10,176,245,  9, 48,157,243
197 DATA 15, 96,162,  2, 32, 99, 11,169
198 DATA 49,141,243, 15, 32, 13, 10, 32
199 DATA 30, 10,165,250,133,251, 32, 80
200 DATA 11,232,134,  2,162,  0,134,252
201 DATA 32,164, 10,230,254,230,252,165
202 DATA  2,197,252,208,243,230,251,165
203 DATA251,198,249,208,225, 32,117, 10
204 DATA 96,162, 10,164,247,136,240,  5
205 DATA202,136,240,  1,202,134,249, 96
206 DATA162, 20,201, 18,144, 12,202,202
207 DATA201, 25,144,  6,202,201, 31,144
208 DATA  1,202, 96, 32,156, 14,169,  0
209 DATA133,198, 32,228,255,240,251,201
210 DATA  3,240,  1, 96, 24, 76,125,  8
211 DATA 32,146, 10,162,  7,189,242, 15
212 DATA 32,210,255,232,224, 12,208,245
213 DATA 32,143, 10, 32,158, 10, 32,225
214 DATA255,208,  4, 24, 76,125,  8, 96
215 DATA165,251,162,  7, 32,242, 10,165
216 DATA252,162, 10, 32,242, 10,169,  0
217 DATA141, 52,  3,165,251,201, 18,208
218 DATA 30,165,252,208, 26,169, 49,141
219 DATA 52,  3,141,243, 15,162,127, 32
220 DATA201,255,162, 11, 32,156, 14, 32
221 DATA204,255,169, 50,141,243, 15,162
222 DATA127, 32,201,255,162, 12, 32,156
223 DATA 14, 32,204,255,162,126, 32,201
224 DATA255,160,  0,165,  1, 41,254,133
225 DATA  1,177,253, 32,210,255,173, 52
226 DATA  3,240,  6,192,143,208,  2,160
227 DATA255,200,208,237, 32,204,255,165
228 DATA  1,  9,  1,133,  1, 32,120, 11
229 DATA162,127, 32,201,255,162, 11, 32
230 DATA156, 14, 32,204,255, 24, 32, 74
231 DATA 10,173,  0,  2,201, 48, 24,240
232 DATA  4, 32,105, 10, 56, 96,162,  3
233 DATA 32, 99, 11,169, 50,141,243, 15
234 DATA 32, 13, 10, 32, 30, 10,165,250
235 DATA133,251, 32, 80, 11,232,134,  2
236 DATA162,  0,134,252, 32,152, 11,176
237 DATA 22,230,254,230,252,165,  2,197
238 DATA252,208,241,230,251,165,251,198
239 DATA249,208,223, 32,117, 10, 96,162
240 DATA  4, 32,156, 14, 32,228,255,201
241 DATA 68,240,222,201, 78,208,245, 32
242 DATA117, 10, 24, 76,125,  8,165,  1
243 DATA  9,  1,133,  1,165, 10,141, 32
244 DATA208,173, 17,208,  9, 16,141, 17
245 DATA208,165,  1,160,  1,  9, 32,133
246 DATA  1,132,192, 88, 96,120,173, 17
247 DATA208, 41,239,141, 17,208,173, 32
248 DATA208,133, 10,169,  6,141, 32,208
249 DATA169,127,141,  0,220,165,  1, 41
250 DATA 31,160,  0,133,  1,132,192,162
251 DATA  0, 96,160,  0,169,  2, 32,218
252 DATA 12,162,  7,136,192,  9,208,244
253 DATA162,  5,198,171,208,238,152, 32
254 DATA218, 12,162,  7,136,208,247,202
255 DATA202, 96,133,189, 69,215,133,215
256 DATA169,  8,133,163,  6,189,165,  1
257 DATA 41,247, 32, 16, 13,162, 12,162
258 DATA 12,234, 72, 44,  1,220, 48,  3
259 DATA 76,139, 13,173, 32,208, 73,  1
260 DATA141, 32,208,104,  9,  8, 32, 16
261 DATA 13,162, 14,198,163,208,213, 96
262 DATA202,208,253,144,  5,162, 11,202
263 DATA208,253,133,  1, 96, 32,146, 10
264 DATA 32, 23,248, 32,149, 12, 96,160
265 DATA  0,132,215,169,  7,141,  6,221
266 DATA162,  1, 32,100, 13, 38,189,165
267 DATA189,201,  2,208,245,160,  9, 32
268 DATA 81, 13,201,  2,240,249,196,189
269 DATA208,232, 32, 81, 13,136,208,246
270 DATA 96,169,  8,133,163, 32,100, 13
271 DATA 38,189,234,234,234,198,163,208
272 DATA244,165,189, 96,173, 32,208, 73
273 DATA  1,141, 32,208,169, 16, 44,  1
274 DATA220, 48,  3, 76,139, 13, 44, 13
275 DATA220,240,243,173, 13,221,142,  7
276 DATA221, 72,169, 25,141, 15,221,104
277 DATA 74, 74, 96, 32,118, 12, 24, 76
278 DATA125,  8,162,  7,134,171, 32, 56
279 DATA248, 32,149, 12,165,  1, 41,254
280 DATA133,  1, 32,186, 12,169,  3, 24
281 DATA101,247,133,185,202,202, 32,218
282 DATA 12,162,  8,185,172,  0, 32,218
283 DATA 12,162,  6,200,192,  5,234,208
284 DATA242,160,  0,162,  4,177,187,196
285 DATA183,144,  3,169, 32,202, 32,218
286 DATA 12,162,  5,200,192,187,208,237
287 DATA169,  3,133,171, 32,186, 12,152
288 DATA 32,218, 12,132,215,162,  7,234
289 DATA177,172, 32,218, 12,162,  3,230
290 DATA172,208,  4,230,173,202,202,165
291 DATA172,197,174,165,173,229,175,144
292 DATA231,234,165,215, 32,218, 12,162
293 DATA  7,136,208,246, 32,118, 12, 96
294 DATA 32, 29, 13, 32, 39, 13,201,  0
295 DATA240,249,133,171, 32, 81, 13,145
296 DATA178,200,192,192,208,246,169,  0
297 DATA141,139,  3, 32,118, 12,162,  5
298 DATA 32,156, 14, 56,169,  8,229,171
299 DATA170,169,  0, 32,205,189, 32,152
300 DATA 10,162, 13, 32,156, 14, 32,143
301 DATA 10,165,171,201,  3,144, 12, 56
302 DATA233,  3,197,247,208,  1, 96,162
303 DATA  7,208,  2,162,  6, 32,156, 14
304 DATA 56, 76,125,  8, 32, 29, 13,165
305 DATA  1, 41,254,133,  1, 32, 39, 13
306 DATA 32, 81, 13,145,172,132,144,132
307 DATA144,132,144, 69,215,133,215,230
308 DATA172,208,  2,230,173,165,172,197
309 DATA174,165,173,229,175,144,225, 32
310 DATA 81, 13, 32,118, 12,165,189, 69
311 DATA215,133,144, 96,189,167, 14,168
312 DATA189,181, 14, 32, 30,171, 96, 14
313 DATA 15, 15, 15, 15, 15, 15, 15, 15
314 DATA 15,  2, 15, 15,  3,196, 54, 63
315 DATA 97,129,150,159,183,213,223,  0
316 DATA242,255, 65, 35, 13, 13,  5, 32
317 DATA 32, 65, 32, 82, 32, 72, 32, 73
318 DATA 32, 86, 32, 65, 32, 13,144, 32
319 DATA192,192,192,192,192,192,192,192
320 DATA192,192,192,192,192,192, 13,  5
321 DATA 32, 49, 46, 32,144, 68, 73, 83
322 DATA 75, 45, 84, 82, 65, 75, 65, 13
323 DATA  5, 32, 50, 46, 32,144, 84, 82
324 DATA 65, 75, 65, 45, 68, 73, 83, 75
325 DATA 13,  5, 32, 51, 46, 32,144, 68
326 DATA 73, 83, 75, 45, 68, 73, 83, 75
327 DATA 13,  5, 32, 52, 46, 32,144, 68
328 DATA 79, 83, 13, 32, 63, 46, 32,192
329 DATA192,192,192,192,192,192,192,192
330 DATA192,192, 13,145, 32,  0, 17,144
331 DATA 68, 73, 83, 75, 58,  5,  0,  5
332 DATA 62, 62, 32, 62, 62, 32, 80, 79
333 DATA 83, 84, 65, 86, 73, 32, 79, 82
334 DATA 73, 71, 73, 78, 65, 76, 32, 68
335 DATA 73, 83, 75, 69, 84, 85, 46, 13
336 DATA  0,144, 62, 62, 32, 62, 62, 32
337 DATA 80, 79, 83, 84, 65, 86, 73, 32
338 DATA 80, 82, 65, 90, 78, 85, 32, 68
339 DATA 73, 83, 75, 69, 84, 85, 46, 13
340 DATA  0,144, 78, 65, 83, 84, 65, 86
341 DATA 65, 75, 63, 40,  5, 68,144, 47
342 DATA  5, 78,144, 41, 13,  0,144, 70
343 DATA 79, 85, 78, 68, 58,  5,  0, 13
344 DATA144, 78, 73, 74, 69, 32, 75, 79
345 DATA 80, 73, 74, 65, 32, 68, 73, 83
346 DATA 75, 69, 84, 69, 46, 13,  0, 13
347 DATA144, 80, 79, 71, 82, 69, 83, 65
348 DATA 78, 32, 82, 69, 68, 78, 73, 32
349 DATA 66, 82, 79, 74, 32, 66, 76, 79
350 DATA 75, 65, 46, 13,  0,-13,144, 78
351 DATA 65, 90, 73, 86, 58,  5,  0, 13
352 DATA144, 84, 65, 80, 69, 32, 82, 69
353 DATA 65, 68, 32, 69, 82, 82, 79, 82
354 DATA 13,  0, 85, 50, 58, 50, 32, 48
355 DATA 32, 51, 53, 32, 49, 54,  0, 66
356 DATA 45, 80, 58, 50, 32, 48,  0,  0
357 DATA  0,  0,  0,  0,  0,  0,  0,  0
358 DATA  0,  0,  0,  0,  0,  0,  0,  0
359 DATA  0,  0,  0,  0,  0,  0,  0,  0
READY.
```

KONTROLNE SUME:

S 0 = 2118	S 32 = 2779
S 1 = 2563	S 33 = 2645
S 2 = 4088	S 34 = 5164
S 3 = 3848	S 35 = 3234
S 4 = 3757	S 36 = 3376
S 5 = 3452	S 37 = 3259
S 6 = 3440	S 38 = 4580
S 7 = 2407	S 39 = 2868
S 8 = 3312	S 40 = 3683
S 9 = 2841	S 41 = 3389
S 10 = 3337	S 42 = 4539
S 11 = 3115	S 43 = 3607
S 12 = 4827	S 44 = 3071
S 13 = 3425	S 45 = 3688
S 14 = 3718	S 46 = 4172
S 15 = 3287	S 47 = 4803
S 16 = 4281	S 48 = 3547
S 17 = 4213	S 49 = 3781
S 18 = 3320	S 50 = 3058
S 19 = 3248	S 51 = 3151
S 20 = 3752	S 52 = 4441
S 21 = 4002	S 53 = 2364
S 22 = 3157	S 54 = 3052
S 23 = 3817	S 55 = 2676
S 24 = 3193	S 56 = 1907
S 25 = 5102	S 57 = 3178
S 26 = 4235	S 58 = 2113
S 27 = 3584	S 59 = 2146
S 28 = 3890	S 60 = 1993
S 29 = 3731	S 61 = 2214
S 30 = 4105	S 62 = 1890
S 31 = 4672	S 63 = 1857
S 32 = 2779	S 64 = 313

→

Komodor 64

FUNKCIJSKI TASTERI

Tastatura „komodora" je ukrašena sa 8 funkcijskih tastera, ali oni uglavnom zvrje neiskorišćeni. Ovim programom dobijaju funkciju za koju su i namenjeni — programiranje sadržaja koji će biti ispisan jednim pritiskom na odgovarajući taster.

Uvedene u dve nove naredbe:

1.]DEF n,STR$ kojom se određuje „sadržaj" tastera (n = broj F tastera, STR$ = sadržaj)
2.] kojom se dobija pregled sadržaja

U naredbi]DEF oba parametra mogu biti i promenjive. Dužina stringa je ograničena na 127 znakova, o čemu sâm program vodi računa — ako je duži, sledi poruka STRING TOO LONG.

Definisanje i korišćenje tastera moguće je u direktnom i programskom modu. Taster f1 je, radi ilustracije, programiran za promenu boje. Često mogu biti od koristi i sledeći oblici:

]DEF 3, „OPEN4,4:CMD4:LIST"+CHR$(13)+„PRINT#4:CLOSE4"+CHR$(13) — listanje programa na printeru

]DEF 5, „LOAD"+CHR$(34)+„$"+SHR$(34)+„,8"+CHR$(13) — za upis direktorija diska

Pri ispisu isprogramiranog sadržaja pojedinih tastera, karakteri 13 i 141 će biti ispisani kao znak?.

Program koristi izmenu interapt vektora. Ako se u toku rada upotrebi prekid sa STOP+RESTORE, F tasteri će posle toga biti neaktivni. Ipak, njihov sadržaj nije izgubljen i dovoljno je tražiti samo pregled sa] da bi se opet aktivirali.

Pre startovanja, program obavezno snimite, jer se u njemu izvodi naredba NEW. Ako dobijete poruku SYNTAX ERROR, negde ste pogrešili pri unošenju DATA linija.

Zoran Životić

```
1 REM"————————— F - TASTERI —————————
2 REM"                 —— © 64 ——
3 REM
4 REM     Autor: Zoran Životić (85)
5 REM
6 REM  Naredbe: ]
7 REM                ]DEF n,str$  0<n<9
8 REM"————————————————————————————————
10 RESTORE:M=0:FORI=49152TO49610
11 READA:POKEI,A:M=M+A:NEXT
12 IFM<>58788THENPOKE58,255:SYS44808
13 SYS49152
14 POKE631,78:POKE632,69:POKE633,87
15 POKE634,13:POKE198,4:LIST-8
16 "————————————————————————————————
100 DATA169,23,162,192,141,8,3,142,9,3,1
20,169,206,162,192,141,20,3,142,21
101 DATA3,88,96,32,115,0,201,93,240,6,32
,121,0,76,231,167,32,10,192,32,115
102 DATA0,201,150,240,94,169,170,162,193
,160,0,133,251,134,252,132,253,169
103 DATA154,160,193,32,30,171,165,253,17
0,24,105,49,32,210,255,169,44,32
104 DATA210,255,169,34,32,210,255,189,16
0,193,240,24,170,160,0,177,251,72
105 DATA41,127,201,13,208,4,104,169,63,7
2,104,32,210,255,200,202,208,235
106 DATA169,34,32,210,255,169,128,24,101
,251,133,251,144,2,230,252,230,253
107 DATA165,253,201,8,208,177,76,174,167
,32,115,0,32,158,183,138,240,4,224
108 DATA9,144,5,162,14,76,55,164,134,253
,32,253,174,32,158,173,32,163,182
109 DATA201,128,144,5,162,23,76,55,164,3
2,189,255,166,253,202,165,183,157
110 DATA160,193,32,128,193,160,0,177,187
,145,251,200,198,183,208,247,76,174
111 DATA167,32,234,255,165,204,208,41,19
8,205,208,37,169,20,133,205,164,211
112 DATA70,207,174,135,2,177,209,176,17,
230,207,133,206,32,36,234,177,243
113 DATA141,135,2,174,134,2,165,206,73,1
28,32,28,234,165,1,41,16,240,10,160
114 DATA0,132,192,165,1,9,32,208,8,165,1
92,208,6,165,1,41,31,133,1,173,168
115 DATA193,208,63,32,135,234,166,198,24
0,47,189,118,2,201,133,144,40,201
116 DATA141,176,36,56,233,133,201,4,144,
5,233,3,10,144,3,10,105,1,170,189
117 DATA159,193,240,15,142,168,193,134,2
53,169,0,141,169,193,198,198,32,128
118 DATA193,173,13,220,104,168,104,170,1
04,64,166,198,224,10,176,241,172
119 DATA169,193,177,251,157,119,2,230,19
8,238,169,193,200,174,168,193,152
120 DATA221,159,193,144,218,169,0,141,16
8,193,240,211,169,170,162,193,133
121 DATA251,134,252,198,253,240,13,169,1
28,24,101,251,133,251,144,243,230
122 DATA252,176,239,96,13,93,68,69,70,0,
32,0,0,0,0,0,0,0,0,0,0,67,61,48,32
123 DATA58,80,79,75,69,53,51,50,56,48,44
,67,58,80,79,75,69,53,51,50,56,49
124 DATA44,67,141,145,29,29,0
```

KAKO PRILAGODITI VIEW (II)

Vlasnici računara BBC koji su pažljivo pročitali „Računare 3" i pokušali da prilagode svoj tekst procesor VIEW jugoslovenskim slovima su se svakako našli u velikom čudu. Tehničkom greškom uz tekst je bio objavljen program koji u samom tekstu nije bio ni pomenut ni objašnjen a osnovnog (dobro objašnjenog) programa nije ni bilo!

Greške poput ove se događaju kada postoje stari i novi tekst a zatim se od tehničkog urednika minut pred zaključivanje lista traži da uzme novi tekst, jedan program iz priloga starog teksta i drugi program iz priloga novog teksta. Tada, naravno, sve ispadne naopako — objavi se stari tekst i novi program!

Na slici je dat program za definisanje karaktera na ekranu vašeg BBC B koji, osim toga, odabira potrebne opcije za rad tekst procesora. Po njegovom startovanju (sa SHIFT BREAK ako imate disk ili *RUN !Boot ako se služite kasetofonom) tasterima na kojima je nacrtana otvorena srednja zagrada, funta, zatvorena srednja zagrada i „backslash" bivaju dodeljena naša slova č, ć, ž i š respektivno.

Da bi se naša slova prenela na papir potreban je, naravno, i printer drajver. U „Računarima 3" je objavljen drajver za Epson RX80 (radi i na FX80 premda za njega može da se napiše daleko bolji program) koji će odgovarati i vlasnicima mnogih štampača koji su kompatibilni sa Epsonom. Ovaj drajver, osim pisanja naših slova, omogućava podvlačenje teksta koje se koristi kao 'Highlight 2' prema uputstvu za korišćenje VIEW-a.

Drajver za štampač Seiksoha GP A koji smo najavili ne objavljujemo jer nam se čini da nije interesantan za širi krug vlasnika BBC-ja. Ukoliko se neki od njih interesuje za program, od Redakcije može da traži besplatan listing.

Dejan Ristanović

```
 10 REM
 20 REM
 30 REM              V I E W
 40 REM
 50 REM     definicije karaktera  2.2
 60 REM
 70 REM            11.06.1984.
 80 REM
 90 REM
100 OSWRCH=&FFE3
110 OSBYTE=&FFF4
120 CLI=&FFF7
130 FOR I=0 TO 3 STEP 3
140 P%=&3000
150 [ OPT I
160 LDA #&B    \ rezervise strane &B i
170 STA &369   \ &C za definicije.
180
190 LDX #0
200 LDY #80    \ definise se 8 slova
210            \ tj. salje 80 bajta.
220 .CYCLE
230 LDA &3100,X
240 JSR OSWRCH
250 INX
260 DEY
270 BNE CYCLE
280
290 LDA #144   \ izvrsava *TV 255,1
300 LDX #255
310 LDY #1
320
330 JSR OSBYTE
340 LDA #22    \ izvrsava VDU 22,6
350 JSR OSWRCH\ tj. MODE 6
360 LDA #6
370 JSR OSWRCH
380
390 LDA #15    \ *FX 15 0 brise bafere
400 LDX #0
410 JSR OSBYTE
420
430 LDX #0
440 .EXNEW     \ stavlja NEW CTRL M
450            \ i PRINTER ... u
460            \ bafer tastature.
470 LDA &30A0,X
480 CMP #ASC "."
490 BEQ CONT
500 TAY
510 TXA
520 PHA
530 LDA #138
540 LDX #0
550 JSR OSBYTE
560 PLA
570 TAX
580 INX
590 BNE EXNEW
600
610 .CONT
620 LDA #202   \ bira mala slova.
630 LDX #48
640 LDY #0
650 JSR OSBYTE
650 JSR OSBYTE
660
670 LDX #(TEXT MOD 256)
680 LDY #(TEXT DIV 256)
690 JSR CLI    \ izvrsava *WORD
700
710 RTS        \ startuje VIEW
720 .TEXT EQUS "WORD"
730 EQUB 13
740 EQUS "(C) Dejan Ristanovic 1984"
750 EQUS "."
760 ]
770 NEXT I
780
790 I=&30A0
800 REPEAT READ X$:X=ASC X$
810 IF X$="¦" THEN X=13
820 ?I=X
830 I=I+1:UNTIL X$="."
840
850 REM  Prva DATA lista daje string
860 REM  koji treba da se izvrsi po
870 REM  inicijalizaciji VIEW-a.
880 REM  String se zavrsava tackom
890 REM  a ¦ oznacava RETURN.
900
910 DATA n,e,w,¦,p,r,i,n,t,e,r," ",E,p
,s,o,n,¦,.
920
930 FOR I=1 TO 80:READ A:?(&30FF+I)=A
940 NEXT I
950
960 *SAVE "!Boot" 3000 3160
970 END
980
990 REM  Serija DATA tablica koje
1000 REM  sadrze definicije karaktera
1010 REM  tj. bajtove koji se direktno
1020 REM  prosledjuju rutini OSWRCH.
1030
1040DATA 23,96,12,60,102,96,96,102,60,0
1050DATA 23,123,108,60,102,96,96,102,60
,0
1060DATA 23,95,12,24,60,102,96,102,60,0
1070DATA 23,91,36,24,60,102,96,102,60,0
1080DATA 23,92,36,24,60,96,60,6,124,0
1090DATA 23,93,36,24,124,12,24,48,126,0
1100DATA 23,124,40,60,96,60,6,102,60,0
1110DATA 23,125,40,124,12,24,48,96,124,
0
```

BBC B LATINICA NA „KANONU"

Ovaj program služi za definisanje YU slova na štampačima „Kanon" i „kaga" i vezan je za tekst „kanon" protiv „epsona" koji ob javljujemo na stranama 8 i 9.

```
 10 REM
 20 REM
 30 REM  Definicije YU stand. slova
 40 REM
 50 REM
 60 REM          Verzija 0.1
 70 REM
 80 REM
 90 REM     Dejan Ristanovic 1985.
100 REM
110 REM
120 ADP=&A30
130 AD=ADP
140 ?AD=ASC "S":REM definicije standardnih slova
150 AD=AD+1
160 OSWRCH=&FFEE
170 OSBYTE=&FFF4
180 FOR I=0 TO 3 STEP 3
190 P%=AD
200 [OPT I
210 LDA #3
220 LDX #10
230 JSR OSBYTE
240 LDX #TABL MOD 256
250 .SALJI
260 LDA ADP AND &FF00,X
270 CMP #&FF
280 BEQ RET
290 JSR OSWRCH
300 INX
310 BNE SALJI   \  uvek JMP
320 .RET RTS
330 .TABL
340 ]:NEXT I
350 AD=P%
360 REPEAT
370 READ A:?AD=A
380 AD=AD+1
390 UNTIL A=255
400 *OPT 1 2
410 OSCLI "SAVE Stand "+STR$~(ADP)+" "+STR$~(AD+1)+" "+STR$~(ADP+1)
420 END
430 DATA 27,64:REM Inicijalizacija
440 DATA 27,79:REM Iskljucivanje preskakanja perforacije
450 DATA 27,54:REM Prosirivanje opsega znakova koji se ispisuju
460 DATA 27,58,0,0,0:REM Interni set slova u RAM
470 DATA 27,37,1,0:REM Aktiviranje PCG karaktera
480 DATA 27,38,0,123,126:REM Redefinisu se ASCII kodovi 123-126
490 DATA 139,24,36,66,128,66,128,66,0,36,0,0:REM CACAK
500 DATA 139,36,82,0,210,0,210,0,82,12,0,0:REM SABAC
510 DATA 139,0,66,4,194,8,210,32,66,0,0,0:REM ZABAC
520 DATA 139,0,0,0,28,34,0,98,128,34,0,0,0:REM celija
530 DATA 27,38,0,91,93:REM Redefinisu se ASCII kodovi 91-93
540 DATA 139,0,0,28,162,64,34,64,162,0,0,0:REM cacak
550 DATA 139,0,16,170,64,42,64,170,4,0,0,0:REM sabac
560 DATA 139,0,34,132,98,8,98,144,34,0,0,0:REM zabac
570 DATA 27,38,0,96,96:REM Redefinise se ASCII kod 96
580 DATA 139,24,36,66,0,66,128,66,0,36,0,0:REM CELIJA
590 DATA 255
```

Test sa zadrškom / prema svecu i drajv / Mikrodrajv nakon godinu dana

Fizički, mikrodrajv se predstavlja kao mala crna naprava dimenzije 9×8×5 cm. Nije ni naročito lepa, ni zgodno oblikovana. Pogled u unutrašnjost ne otkriva neku veliku mudrost: jedan motor, jedna magnetna glava, malo elektronike i dva konektora. Kertridž, koji se ubacuje sa prednje strane, nije ništa drugo do minijaturna kaseta (4,3×3×0,5 cm) koja sadrži 12 metara dugačku traku širine svega 2 milimetra. Cela stvar radi na principu beskonačne petlje, a Sinkler je to rešio tako što se na sredini kasetice (kertridža) nalazi jedan mali točkić na koji je namotana traka. Sa jedne strane se traka odmotava i prolazi ispred magnetne glave, a sa druge strane se namotava.

Ipak ne puca

Sve to izgleda krajnje delikatno i krajnje — nepouzdano! Ako se još zna da se traka kreće brzinom od skoro dva metra u sekundi, onda čoveku padne mrak (čitaj: PAPER 0) na oči. Ipak, koliko je poznato autoru ovog teksta, još niko u Beogradu nije imao problema sa mikrodrajvom! Razni slučajevi da traka puca verovatno su se dešavali samo sa prvim primercima koji su se pojavili na tržištu. Verovali u Sinklera ili ne, sa mikrodrajvom treba postupati jako pažljivo — isto, uostalom, važi i za najskuplje disk jedinice. Što se kapaciteta tiče, on vrlo čudno varira između 89 i 91 KB. Ako se to nekome učini malo, moguće je povezati do osam mikrodrajva u rednu vezu i time dobiti maksimalan kapacitet od 720 K, što nije nimalo zanemarljivo.

Hopa cupa preko rupa

Brzina prenosa podataka između „spektruma" i mikrodrajva je više nego izvanredna: 14 KB u sekundi. S obzirom da je ta brzina retko kad veća i kod najboljih disk jedinica, moglo bi se zaključiti da je mikrodrajv vrlo brza jedinica spoljne memorije. Na žalost, nije tako. Da bi se neki podatak učitao, treba najpre stići do njega. Prosečno vreme pristupa iznosi 3,5 sekunde, a maksimalno 7 — to je vreme za koje se obrne cela traka. Iz toga proizilazi da se program dugačak, recimo, 20 K učitava u proseku za 5 sekundi, a u najgorem slučaju za 7. Ako je kasetica prazna, ili vrlo malo popunjena, to je tačno. U protivnom, stvari se komplikuju. Naime, kod snimanja programa računar traži najveću praznu „rupu" na kasetici i tu snima program. Ako ta rupa nije bila dovoljno velika, traži sledeću i tu nastavlja snimanje. Jasno, što je program iseckaniji, to će njegovo učitavanje trajati duže — često mnogo duže nego onih maksimalnih 7 sekundi. U najgorem, slučaju učitavanje 20 K dugačkog programa ume da potraje i dobrih 15 sekundi. Trenutak u poređenju sa kasetofonom. Ali pravi disk . . .

Naredba SAVE se još duže izvršava nego LOAD. Računar mora da obrne celu traku da bi se uverio da za novi program ima dovoljno mesta. Zatim dolazi već opisno hopa cupa preko rupa i, na kraju, poruka OK. Zavisno od slučaja, to traje 10 do 20 sekundi. Ali to nije sve. Ako program sa istim imenom već postoji na kertridžu (prosto rečeno: stara verzija programa), potrebno ga je prvo obrisati pomoću ERASE naredbe. To traje koliko i učitavanje. Najzad, kod vrlo važnih programa, nije na odmet verifikacija, jer se to ne vrši automatski prilikom snimanja. Da se vratimo na naš 20 K dugačak program — jedna operacija tipa ERASE/SAVE/VERIFY u praksi retko kad traje manje od 30 sekundi.

Sintaksni teror

Ipak, nije sve tako crno. Ma koliko rad sa mikrodrajvom izgledao spor, on je ipak neuporedivo brži nego sa kasetofonom. Što je najvažnije, sve se odvija potpuno automatski: nema onog premotavanja kasete, traženja programa, podešavanja jačine, TAPE LOADING ERROR poruka i slično, što rad sa kasetofonom čini toliko neprijatnim. Istina, dešava se da program sa mikrodrajva neće da se učita kako treba, ali to je vrlo retko. U celini uzev, radi se o vrlo sigurnoj jedinici spoljne memorije.

Postoji još jedna prednost mikrodrajva u odnosu na kasetofon: to je rad sa datotekama. Operativni sistem podržava do 12 datoteka istovremeno. Pri tome, svaki podatak može biti proizvoljne dužine, ali pristup, na žalost, može biti samo sekvencialan. Postoje još neka ograničenja. Recimo, ne postoji naredba koja bi prepoznala kraj datoteke, tako da se softver koji kontroliše mikrodrajv teško može meriti sa nekim boljim DOS-om.

Ono što je stvarno katastrofalno kod mikrodrajva nije ni spor pristup, ni mali kapacitet, ni sekvencialne datoteke. Katastrofalna je sintaksa naredbi. Da bi se učitao neki program sa mikrodrajva, potrebno je otkucati ni više ni manje nego:

LOAD *"M"; 1; „ime"

Ona jedinica, na primer, označava broj mikrodrajva. Pošto na „spektrum" može biti priključeno do osam mikrodrajva, navođenje tog broja je sasvim opravdano. Ali, zaboga, zašto čovek stalno mora da kuca tu prokletu jedinicu ako ima samo jedan mikrodrajv? Ili ona zvezdica koja kod nekih naredbi mora da se navede, a kod drugih ne. Još gori primer je naredba CAT (katalog) koja, umesto da prikaže kompletne podatke o svim programima koji se nalaze na kertridžu, izbacuje samo njihova imena. Izgleda kao da su naredbe rađene tako da okane zagriženog korisnika od mikrodrajva.

Dos, tos ili samo — SOS

Da bi se mikrodrajv povezao sa „spektrumom", potreban je interfejs 1 koji se priključuje na port za proširenja sa zadnje strane računara. Pri tome, „spektrum" dobija vrlo fini nagib koji olakšava kucanje. Ono što, međutim, nije nimalo fino je dužina kabla koji povezuje interfejs i mikrodrajv: svega osam centimetara. Zahvaljujući tome, mikrodrajv se može nalaziti samo sa leve strane „spektruma" — tamo gde je konektor, a ne i sa desne. Ser Klajv u svojoj ekscentričnosti verovatno lakše ubacuje kasetice levom rukom.

Pogled u interfejs 1 otkriva jedan neizbežni ULA čip, dva standardna TTL kola, jedan ROM — nazovimo ga ROM 2. par tranzistora i to bi otprilike bilo sve.

U 8K, koliko sadrži taj ROM 2, upisan je kompletan program za rad sa mikrodrajvom, koji bi, uslovno, mogao da se nazove TOS — tape operating system. Na žalost, baš u koncepciji tog TOS-a leže glavni, već spomenuti, nedostaci mikrodrajva. Većina problema potiče još iz glavnog ROM-a, gde su, kao u šopskoj salati, katastrofalno izmešani operativni sistem i bejzik interpreter. Što je najgore, iako se za vreme pisanja glavnog ROM-a i te kako dobro znalo za planirani mikrodrajv, ništa nije urađeno da mu se pripremi teren. Kao rezultat, ROM 2 uvodi naredbe za rad sa mikrodrajvom na neke, gotovo neverovatne načine, jer, naravno, nigde u glavnom ROM-u nije predviđen ni jedan jedini link, a praksa pomešanog operativnog sistema i bejzik interpretera je nastavljena. Rezultat je konfuzija u kojoj se čak i najiskusniji programeri teško snalaze.

Srećom, prosečnog korisnika ne zanima mnogo šta i kako radi ROM. Za njega je samo važno to da se svi programi izvršavaju bez problema. Na žalost, ni to nije uvek slučaj. Praksa pokazuje da dobar deo uslužnih programa i većina igara neće da radi kako treba kada se na „spektrum" priključi interfejs 1. Razlog leži u tome što je početak bejzika pomeren sa 23755 na 23813, a mnoge stvari u području sistemskih promenljivih i kanalskih informacija se menjaju. Uz to, mikrodrajv može da zahteva i do nekoliko kilobajta radne memorije, zavisno od toga šta se sa njim radi. Bilo kako bilo, vrlo verovatno ćete dobar deo svog postojećeg softvera morati da prilagodite novim uslovima.

Disk na grani, mikrodrajv u ruci

Ostaje još da se razmotri odnos mogućnosti/cena. Trenutno se u Engleskoj komplet ZX INTERACE 1+MICRODRIVE+9 praznih kasetica prodaje za oko 65 funti, što je vrlo prihvatljivo. Tim pre ako se zna da razni kasetofoni tipa „data recorder", koji ni izdaleka nisu ravni mikrodrajvu, dostižu cenu i do 40 funti. Ako dalje

Prošlo je dosta vremena od kada su neozbiljne spravice, po imenu „mikrodrajv" putem pošte za 28 dana, počele da stižu najhrabrijim „spektrumovcima". Neki još uvek sumnjivo vrte glavama, drugi se groze, a treći ih kuju u zvezde. Gde je istina, i da li je to čudo genijalnog (?) Ser Klajva stvarno četiri puta jeftinije od prave disk jedinice? Napadan i osporavan sa svih strana, mikrodrajv je ipak ušao u život mnogih ljubitelja računara. Da li je i kako položio svoj najvažniji ispit — ispit pred onima koji svakodnevno rade sa njim?

nastavimo spekulacije, interfejs 1 sa četiri mikrodrajva daje kapacitet od 360 K za relativno skromnih 200 funti. S druge strane, prava disk jedinica sa kapacitetom od 200 K namenjena „spectrumu" košta baš isto toliko: 200 funti. Tu se, međutim, odmah postavlja jedno vrlo neugodno pitanje: ako je sopstveni Sinklerov proizvod toliko nekompatibilan sa postojećim softverom, šta tek treba očekivati od disk jedinice drugog proizvođača? Što se tiče kertridža, iako im je cena pala sa pet funti na dve, i dalje su dosta skupi. Za isti novac se može kupiti disketa na koju staje 400 K podataka.

U stvari, na mikrodrajv se može gledati sa nekoliko vrlo različitih pozicija.

Ako koristite „spektrum" samo da biste se igrali, onda zaboravite odmah na mikrodrajv. Prebacivati igre sa kaseta na mikrodrajv nije nimalo jednostavan posao i zahteva dobro poznavanje programiranja. Uz to, kasetice su i suviše skupe da bi se na njima držale igre.

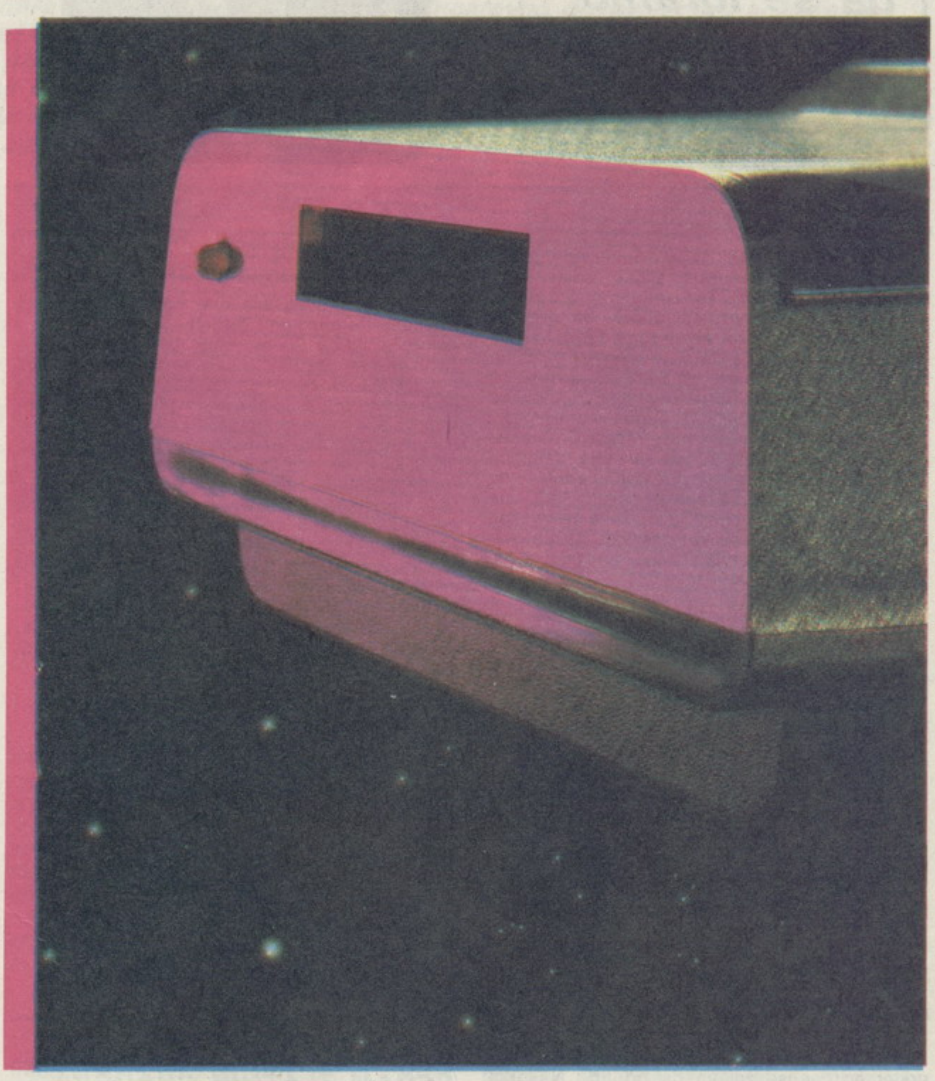
Ako „spektrum" koristite za neku manju obradu podataka, programiranje iz hobija ili nešto slično, mikrodrajv je dobrodošao i sasvim je dovoljan. Mašina sa gumenom tastaturom ne zaslužuje ništa bolje.

Najzad, ako ste svoju mašinu opremili profi tastaturom i ako je intenzivno koristite za programiranje ili obradu podataka, onda ste stavljeni pred ozbiljnu dilemu: mikrodrajv ili disk jedinica? Mikrodrajv boluje od sporog pristupa podacima i Sinklerovog imena na sebi, a disk jedinica od softversko-kompatibilnih problema. Šta je bolje za vas, odlučite sami.

Postoji još jedna tipično YU situacija: volite kompjutere, ali nemate dovoljno dolara za IBM PC, pa ste morali da se zadovoljite „spektrumom". Želite disk jedinicu, ali nemate 200 funti. U tom slučaju, smatrajte da je mikrodrajv fantastična zamena.

Ima mana, ima sporosti, ima naredbi čija je sintaksa van svake pameti, ali košta svega 65 funti i neuporedivo je bolji i brži od svakog kasetofona. Ako se silom finansijskih prilika odlučite za mikrodrajv, nećete se kajati.

Vladimir Kostić



Strah od kidanja: Mikrodrajv predstavlja prelazni oblik između kasetofona i disk jedinice i, mnogo bolji pouzdaniji od prvog i znatno lošiji od drugog, sasvim je primeren računarima za koje je namenjen

ZX MICRODRIVE I INTERFACE 1

— **KAPACITET:** oko 90K po kasetici.
— **PROSEČNO VREME UČITAVANJA 40K DUGAČKOG PROGRAMA:** 7 do 15 sekundi.
— **PROSEČNO VREME SNIMANJA 40K DUGAČKOG PROGRAMA:** 15 do 20 sekundi.
— **MOGUCNOST DA SE POVECA KAPACITET:** do osam mirkodrajva, ukupno 720K
— **NAREDBE ZA RAD SA MIKRODRAJVOM:** CAT, CLOSE, ERASE, FORMAT, INKEY$, INPUT, LOAD, MERGE, MOVE, OPEN, PRINT, SAVE, VERIFY.
— **MOGUĆNOST RADA SA DATOTEKAMA:** do 12 datoteka istovremeno, samo sekvencialne.
— **RS 232:** koristi nestandardan 9 polni priključak. Brzina prenosa do 19200 bauda.
— **LOCAL AREA NETWORK:** mogućnost da se do 64 „spektruma" povežu u mrežu.
— **CENA KERTRIDŽA:** 2 funte.
— **CENA KOMPLETA MIKRODRAJV+INTERFEJS 1+9 kasetica;** 65 funti.
— **ODNOS MOGUĆNOSTI/CENA:** za mikrodrajv zadovoljava, za kertridže dosta slab.
— **NAŠA OCENA:** u nedostatku prave disk jedinice, sasvim zadovoljava.

My chess II 3D

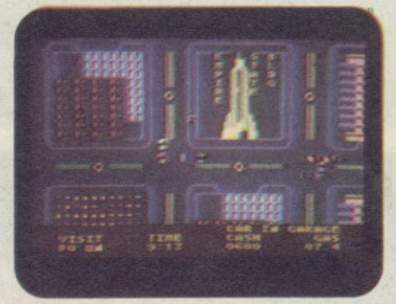
MOJ ŠAH II

Verovatno još od QL-ovog trodimenzionalnog šaha čekate da se nešto slično pojavi i za „komodor 64". Igranje šaha na ovom programu pruža zaista sjajan vizuelni utisak. Sa dve komande možete okretati tablu i zagledati je sa svih strana. Za one koji tvrde da ovakvi efekti samo smetaju kod izbiljne igre šaha, treba objasniti još neke stvari. Jaka strana ove igre nije samo trodimenzionalna projekcija, nego i veoma širok i bogat meni. Program može da rešava probleme, da igra na više nivoa, da vas savetuje, da snimi poziciju na disk ili učita neku staru partiju ili da na printeru odštampa čitavu partiju koju ste ranije odigrali. Mogućnosti ovog šaha će sigurno ohrabriti čak i one koji se užasavaju Kasparova, Karpova i sličnih genijalaca da oprobaju svoju sreću bez kompleksa.

NEW YORK CITY

NJUJORK

Ova igra je napravljena da bi sve one koji se preterano zanose željom da odu do Njujorka odvratila od takve namere sredstvima koja kao da su preuzeta iz nekog Karpenterovog filma. Igra počinje tako što ste ispred garaže u Njujorku. Naravno, u kolima ste. Pred vama je ceo grad. Čim krenete malo u grad, na vas naleti neki idiot u kolima koja su bar duplo veća od vaših. Otpremaju vas pravo u bolnicu u kojoj morate da platite da biste izašli. Ali, kola su vam sad na drugom kraju grada. Zato ulazite u banku da podignete novac koji vam je potreban, ali u banci vam ne daju novac ako ne odigrate jednu igru. Tu se već naslućuje morbidni pristup autora igre. Ako budete hteli da uđete u bilo koju od zgrada koje su pred vama, opet ćete morati da odigrate neku video igru. Lova koju ste uzeli u banci se polako troši i kad potpuno ostanete bez nje dobićete obaveštenje da je igra završena. Ovo je verovatno košmarno viđenje vikenda u Njujorku za nekog jadnika iz siromašne stare Evrope: prvo vam razlupaju kola, pa vas teraju da non-stop igrate video igre, pa vas autiraju kad ostanete bez love. Ako je tamo zaista tako, onda je najbolje da dugo igrate ovu igru i uvežbate preživljavanje u Njujorku.



SUPER BUNNIE

SUPER ZEKA

Ovo je igra za sve one koji su odrasli sa Duškom Dugouškom. Sadržaj je kao prenet iz nekog crtanog filma. Mali beli zeka pojede šargarepu (koja uopšte nije obična) i odjednom postaje super zec. Igra se sastoji iz nivoa na kojima čarobni zeka obavlja razne zadatke. Jedini problem mu je to što dejstvo čarobne šargarepe posle izvesnog vremena prestaje, pa mora po novu kod veverice koja mu je daje. Vi ćete se, naravno, upitati zašto veverica ne pojede šargarepu i postane super veverica? Zato što veverice ne vole šargarepe, ali su, izgleda, jako prijateljski nastrojene prema zečevima. Posebnu slast igre predstavljaju scene kad mali zeka sređuje čudovišta koja ga, inače, ometaju u dobijanju šargarepe. Možda bi Duško Dugouško to uradio drugačije, ali ni ovako nije loše. Posle silnih noći igranja, čak vam ni čarobna šargarepa neće povratiti vid. Ali, možda se isplati.

SEVEN CITIES OF GOLD

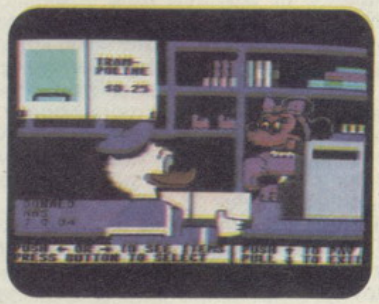
SEDAM ZLATNIH GRADOVA

Bilo je samo pitanje vremena kad će tvorci video igara početi da se vraćaju u prošlost i da motive za igre vade iz stvarnih istorijskih događaja. Motiv ove igre je otkrivanje i osvajanje Amerike. Igra sadrži maltene bukvalno prenesenu tehniku rada osvajača. Sa četiri broda i sto ljudi krećete u osvajanje. Kada stignete do kopna i sretnete domoroce, tehnika susreta je veoma prosta — najbolje je zatražiti da vas odvedu do poglavice i tada započeti borbu. Ako ubijete poglavicu, ostale domoroce ćete smaći mnogo lakše i brže. Autori nisu patili od manije ulepšavanja istorije, kao što vidite. Kada se domoroci predaju, pokupite im zlato, napravite malo utvrđenje u kome ostavite par ljudi da se domoroci ne bi pobunili i krenete u dalja istraživanja. Tražite reke i planine i sela sa zlatom. Posle peripetija i avantura (postoji čak i mogućnost dezertiranja posade) sa gomilom zlata najbolje je vratiti se tamo odakle ste i pošli, to jest u Španiju, gde će vas kraljica prihvatiti i nagraditi i dati vam sredstva za novu ekspediciju. Takođe ćete videti mapu oblasti koje ste osvojili i otkrili. Prava mala imperijalistička igra.

DONALD DUCK

PAJA PATAK

U trendu pravljenja igara po filmovima, stripovima, pa čak i knjigama, bilo bi zaista čudno da je neko zaboravio na najpoznatije gunđalo na svetu, Paju Patka. U ovoj igri Pajin zadatak je da nabavi igračke za svoje sestriće. Pošto nema para (igra je, dakle, prilagođena našim uslovima) primoran je da radi (tezgari) na četiri moguća mesta. To su železnica, prodavnica, farma voća i aerodrom. Količina para koju zaradi zavisi od kvaliteta obavljenog posla. Kad nakupi dovoljno para, Paja prelazi preko ulice i kupuje igračke u radnjama u kojima rade Miki, Mini i Šilja. Ostatak igre je prilično jednostavan — igračke odnose i raspoređuje po sobama, a zatim dovodi jednog od sestrića da igračke isproba. Igra dosta podseća na crtani film, osim što nedostaje Pajino gunđanje i napadi besa. Igrajući ovu igru, sve vreme očekujete da Paja počne da se svađa sa Mikijem oko cene neke igračke. Ko zna, možda u sledećoj verziji.



CRYSTAL CASTLES

KRISTALNI DVORCI

Još jedna iz dugog niza „sakupljačkih" igara. Vaš je zadatak da lutate kroz te, šta su da su, Kristalne dvorce i da skupljate dragulje koji su razbacani unaokolo. Naravno, u tom vas ometaju sve moguće spodobe koje je bolesna programerska mašta mogla da stvori. Deluje dosta jednostavno, ali kad se doda da je sve to u tri dimenzije, kao i da postoje hodnici, liftovi i skokovi na kojima bi vam i Karl Luis pozavideo, sasvim je očito da je igra zaista zanimljiva. Svi nivoi (a ima ih zaista puno) su izvanredno urađeni, pa vredi uložiti malo truda i par neprospavanih noći da se pređe što više nivao i vidi koliko dragulja može da se skupi, makar i u igri.

Pripremio: Vladimir Krstonošić

putovanje u provincije

Komunikacija sa periferijskim uređajima

Jedna od posebnih vrednosti računara „Komodor" je, svakako, jednostavno povezivanje i korišćenje periferijskih uređaja. Izabrano je rešenje da se operativni sistemi pojedinih uređaja ugrađuju u njih same, pa je računaru ostalo malo posla. Zbog toga su i bejzik naredbe za komunikaciju na gotovo elementarnom nivou — samo korak dalje od onoga što je potrebno pri pisanju mašinskim jezikom. To, međutim, ne znači da upravljanje iz mašinskog jezika nema nikakvog smisla. Mašinac i ovde, kao i u mnogim drugim primenama, može značajno da unapredi performanse računara.

Za komunikaciju sa periferijskim uređajima u bejziku su nam na raspolaganju sledeće naredbe: OPEN, PRINT#, INPUT#, GET#, CMD i CLOSE. Primećujete da nema nikakvih specifičnih naredbi za pojedine uređaje, kao na primer LPRINT ili LLIST za ispis na štampaču. To je posledica činjenice da se operativni sistemi nalaze u perifernim uređajima, pa računar sa svima opšti na isti način. Tačnije, svi uređaji priključeni preko serijskog interfejsa koriste iste rutine računarevog operativnog sistema, pa on „ne zna" kome se od njih obraća. Rad ostalih (kasetofon, tastatura, RS232 i video memorija) računar programski potpuno podržava. Ipak, za sve je zadržan isti koncept ostvarivanja veze, što ima dosta prednosti.

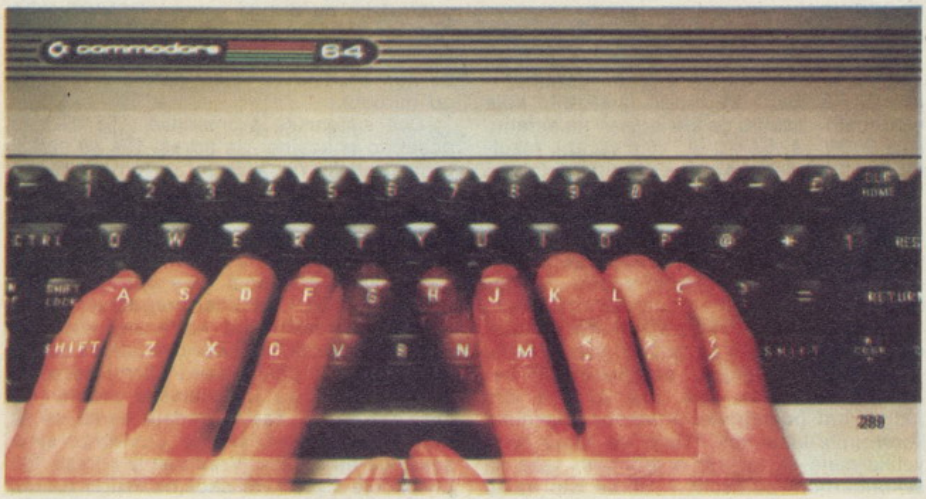
Prva . . .

Da bi se razlikovali jedni od drugih, uređajima su dodeljeni takozvani fizički brojevi (device naumbers) ili, kako se još nazivaju, *Prve adrese:*

TASTATURA	0	P
KASETOFON	1	P/S
RS 232 INTERFEJS	2	P/S
VIDEO MEMORIJA NISKE REZOLUCIJE	3	P/S
ŠTAMPAČ	4	P
DISK JEDNICA	8	P/S

U desnoj koloni je dat i moguć smer komunikacije (P-prijem, S-slanje) sa stanovišta računara.

Pri otvaranju veze treba navesti broj uređaja koji se koristi i računar na osnovu njega odlučuje kome da se obrati. Komunikacija sa pojedinim uređajima je zaista jednostavna — sa tastature se samo može dobiti informacija koji je taster pritisnut, pa je za ovakvu komunikaciju dovoljno navesti broj uređaja (u ovom slučaju 0) da bi sve bilo jasno. Disk jedinica može da obavlja daleko više poslova: da formatira disketu, zapisuje i čita podatke, briše pojedine programe itd. Pošto se ceo operativni sistem koji kontroliše rad diska nalazi u samoj disk jedinici, javlja se potreba da mu računar nekako stavi do znanja šta se od njega očekuje. Kako da raspozna da li ono što mu stiže treba da zapiše kao podatak, ili se radi o naredbi, kada i jedno i drugo prima na isti način, kao niz ASCII kodova? Zato je uvedena *Sekundarna adresa.*



. . . sekundarna . . .

Sekundarna adresa predstavlja dodatnu informaciju, moglo bi se reći vid naredbe, i njene su vrednosti, kada je disk u pitanju, izabrane na sledeći način: 0 i 1 označavaju disku da se radi o naredbama SAVE i LOAD koje se posebno tretiraju, 2 do 14 znači da slede podaci za datoteku, dok sekundarna adresa 15 ima posebno važnu namenu. Kada je primi, sve no što za njom sledi disk smatra naredbom i kao takvu pokušava da je prepozna i izvede.

. . . logička adresa

Da bi se, dakle, bilo šta saopštilo disku, treba prvo poslati njegov broj, a onda i sekundarnu adresu, i to svaki put kad koristimo na primer PRINT#. Moglo se desiti da „Komodor" ovde stane, pa bi bejzik naredba imala sledeći oblik:

```
PRINT#8,15, „NEW: naziv diskete, ID"
```

za formatiranje diskete. Da se ipak ne bi svaki put pisao i broj uređaja i potrebna sekundarna adresa, uvedena je nova: *Logička adresa.* U naredbi i OPEN, kombinaciji broj uređaja-sekundarna adresa dodeljuje se proizvoljan broj i kasnije samo on navodi. Tako se prethodni slučaj formatiranja svodi na:

```
OPEN 121,8,15
PRINT#121, „NEW: naziv diskete, ID"
```

Namerno smo izabrali „slučajan" broj 121 za logičku adresu, da bismo naglasili proizvoljnost njenog izbora.

U komunikaciji sa tastaturom ili video memorijom nema potrebe ni za kakvim dodatnim informacijama. Pa ipak, sintaksta bejzik naredbe je za sve uređaje ista, pa je moguće pisati i:

```
OPEN 56,0,34
GET= 56,A$
```

ako želimo da očitamo koji je taster pritisnut. Računar će sekundarnu adresu, tamo gde je spoljni uređaj ne zahteva, jednostavno ignorisati, ali i dalje ostaje potreba da se poštuje sintaksa OPEN naredbe.

Moguća su i neka skraćenja u njenom pisanju. Sekundarna adresa se može izostaviti, ali to nikako ne znači da se ona u tom slučaju ne razmatra. Jednostavno, ako nije napisana, računar joj daje vrednost 0 ili 255 (zašto baš ove vrednosti biće objašnjeno kasnije). Moguće je izostaviti Čak i broj uređaja u tom slučaju se interno postavlja vrednost 1, što znači da se radi sa kasetofonom.

Postonji još jedan skraćeni oblik otvaranja veze koji je u jednom slučaju obavezan. Naime, tamo gde odmah po otvaranju treba poslati i neki ASCII niz, na primer naziv datoteke ili naredba disku, odgovarajući niz se može dodati u OPEN naredbu. Na primer:

```
OPEN 12,8,15
PRINT# 12, „NEW: TEST"
```

je ekvivalentno sa:

```
OPEN 12,8,15, „NEW: TEST"
```

ali je pri otvaranju datoteke na disku ili kasetofonu obavezno:

```
OPEN 43,8,3, „NAZIV,S,R"
```

Parametri i OPEN naredbe mogu biti promenjive, numerički izrazi i slično. Tada ovako izabrano tretiranje perifernih uređaja dolazi do punog izražaja. Uz malo pažnje pri oblikovanju ispisa (izbegavajući TAB ili zareze), u programu se pri otvaranju veze broj uređaja može dodeliti varijabli i onda jednostavnom promenom njene vrednosti usmeravati ispis na ekran ili štampač, a u jednostavnijim slučajevima i direktno u neku datoteku na disku ili traci. Na primer, podprogram za ispis može izgledati ovako:

```
OPEN 10,DV,SA,A$
FOR F=0 TO 20: PRINT# 10,R (F):NEXTF
CLOSE 10
RETURN
```

Ako ga pozovete sa ulaznim parametrima DV=3 (SA i A$ mogu imati bilo koje vrednosti), rezultat će biti ispisan na ekranu. Ako je DV=4, SA=0 (A$ nije važan), rezultat će biti smešten u sekvencijalnu datoteku na disku, odakle kasnije može biti korišćen u nekom drugom programu, na primer tekst procesoru i slično.

Koncept sa logičkom, prvom i sekundarnom adresom se očigledno dopao „Komodoru", pa je dosledno poštovan i na nekim mestima gde komplikuje stvari. Na štampaču 1526 (MPS 802) se gotovo sve funkcije ostvaruju korišćenjem odgovarajućih sekundarnih adresa (čak 11), iako je isto moglo biti postignuto primenom nekoliko kontrolnih karaktera. Tako se izbor seta velika/mala slova vrši otvaranjem veze sa sekundarnom adresom 7, ali se isto može postići i bez njenog navođenja ako se pre stringa pošalje karakter 17 (kursor dole)!

Sistemske promenljive

Za rad sa periferijama rezervisan je veliki broj sistemskih promenjivih u nultoj i prvoj strani memorije. Mnoge od njih su interni — brojači, pokazivači, zastavice ili baferi i nisu od značaja — za korisnika ostaje svega nekoliko važnih koje ćemo nabrojati:

152 — broj otvorenih veza
153 — ulazna jedinica
154 — izlazna jedinica

183 — dužina naziva datoteke
184 — logička adresa
185 — sekundarna adresa
186 — prva adresa (broj uređaja)
187 i
188 — pročetna adresa naziva datoteke

601 do 610 — tablica logičkih adresa
611 do 620 — tablica prvih adresa
621 do 630 — tablica sekundarnih adresa

Kada zadamo OPEN, računar uzima prvi parametar iza naredbe i njegovu vrednost dodeljuje logičkoj adresi. Zatim se broj uređaja postavlja na 1 (kasetofon), sekundarna adresa i dužina naziva datoteka na 0. Sada se tek ispituje (odatle mogućnost skraćivanja) da li je zadat sledeći parametar (broj uređaja) i ako jeste, ranije dodeljena vrednost 1 se menja u zadatu. Isto važi i za sledeći parametar, kao i za string na kraju, s tim što, ukoliko nema sekundarne adrese a broj uređaja je veći od 3, ranije postavljena vrednost 0 se menja u 255. Sve ove vrednosti se smeštaju u sistemske promenjive 183—188

Sada se poziva rutina koja će pretražiti tablicu logičkih adresa da bi se utvrdilo da li je već ranije otvorena veza sa istim brojem i, ako jeste, izdala poruka FILE OPEN ERROR. Ako nije, logička, prva i sekundarna adresa se ubacuju na krajeve odgovarajućih tablica i broj otvorenih veza (152) povećava za jedan. Time je veza otvorena i dalji tok događaja zavisi od broja uređaja.

Najčešće se neće desiti ništa više. Tako se nakon OPEN 4,4, i ako štampač nije priključen, ne dobija poruka DEVICE NOT PRESENT. U slučaju disk jedinice odmah se, ako postoji, šalje string iz OPEN naredbe koji predstavlja naziv datoteke ili neku od naredbi.

Dve sistemske promenjive (153,154) su posebno značajne. Na adresi 154 se nalazi broj uređaja na koji je u tom trenutku usmeren prenos podataka iz računara. Ako pogledate njen sadržaj, videćete da se tamo nalazi broj 3, što znači da je ispis u tom trenutku usmeren na ekran. Šta se dešava kada nakon otvaranja veze upotrebite naredbu PRINT#? Računar pretražuje tablicu logičkih adresa i kad nađe onu koju ste naveli u naredbi, iz ostale dve tablice uzima odgovarajuću prvu i sekundarnu adresu i sve to prebacuje u sistemske promenjive 184, 185 i 186. Zatim uređaju naređuje da počne da sluša i, ako je potrebno, šalje sekundarnu adresu. Vrednost iz adrese 186 prebacuje u 154 i vrši skok na PRINT rutinu, istu onu koju koristi i „običan" PRINT. Kada je i poslednji bajt poslat, spoljnom uređaju se naređuje da prestane sa slušanjem i vrednost sistemske promenjive 154 se vraća na 3. Sem u jednom sučaju: ako je upotrebljena naredba CMD. Njenim zadavanjem nakon OPEN vrednost sistemske promenjive 154 se postavlja kao u gore opisanoj proceduri, ali na toj vrednosti i ostaje. Sada i „običan" PRINT ili LIST, umesto na ekran šalje ispis na spoljni uređaj. Ovakvo stanje će trajati sve dok se preko veze ne pošalje jedan „Prazan" PRINT#.

Dakle rutina koja vrši slanje karaktera ispituje jedino vrednost u lokaciji 154 da bi ustanovila o kom se uređaju radi. Slična procedura se odvija i pri toku informacije od periferije ka računaru — tada je važna sistemska promenjiva 153, s tim što u bejziku ne postoji ekvivalent za CMD u slučaju INPUT naredbe.

Komunikacija iz mašinca

Retko se pruža prilika da pisanje u mašinskom jeziku bude jednostavno kao u bejziku. U radu s spoljnim uređajima upravo je to slučaj. Sve naredbe bejzika se jednostavno zamenjuju pozivom jednog ili najviše tri potprograma. Usvojena je i sledeća konvencija koja je dosledno poštovana:

Svi potprogrami se pozivaju indirektno preko takozvane KERNAL JUMP tablice, čime se obezbeđuje nezavisnost mašinskog programa od verzije operativnog sistema računara (broj verzije možete saznati upotrebom PRINTPEEK (65408)

• Pri povratku iz potprograma CARRY zastavica, ako je 0, označava da je sve proteklo u redu. U suprotnom, u akumulatoru će se naći kod greške od 0 do 9, sa sledećim značenjem:

0 — rutina prekinuta STOP tasterom
1 — previše otvorenih veza (više od 10)
2 — veza već otvorena
3 — veza nije otvorena
4 — datoteka nije nađen
5 — uređaj nije priključen
6 — uređaj nije predviđen za slanje podataka
7 — uređaj nije predviđen za primanje podataka
8 — nedostaje naziv datoteke
9 — nepostojeći broj uređaja (device number)

Interesantno je da je „Komodor" predvideo mogućnost i da se broj greške u trenutku nastanka odštampa na ekranu. Da bi ovo omogućili, treba šesti bit u sistemskoj promenjivoj na adresi 157 postaviti na 1. Bejzik ga pri svakom prelasku u direktan mod vraća na nulu, ali probajte u jednoj liniji:

```
POKE157,192:PRINT#43
```

Iz navedenog spiska se vidi da se greške uglavnom odnose na samo otvaranje veze i pripremne rutine. Informaciju o stanju u toku samog slanja ili primanja podataka dobijamo iz sistemske promenjive STATUS na adresi 144. Postoji i potprogram koji očitava njenu vrednost i vraća je u akumulatoru. Iako je on krajnje jednostavan, zapravo izvodi jedino instrukciju LDA 144, ipak se preporučuje da ga koristite umesto direktnog očitavanja memorije zbog kompatiblinosti različitih verzija „Komodor" računara.

Svaki bit STATUSA ima posebno značenje od kojih su praktično samo 3 od važnosti. Ako je neki od njih 1, znači da je:

bit 5 — greška u kontrolnoj sumi pri čitanju sa trake
bit 6 — primljeni karakter poslednji u datoteci
bit 7 — uređaj nije priključen ili je naišao kraj trake

Bez obzira na ovako razgranat sistem obaveštvanja o greškama, on nije dovoljan da pokrije i disk jedinicu. Iz STATUSA se samo dobija informacija da nešto nije u redu, nekad ni to, i da bi se ustanovilo koja je stvarno greška nastala, potrebno je posebno očitavanje komandnog kanala.

U prilogu 1 dat je program koji uvodi novu naredbu LPRINT koja je standardna u većini bejzika. Ima istu namenu kao i PRINT, ali ispis ide na štampač. Iako na „komodoru" za njom ne postoji potreba, koristi se PRINT#, namerno smo izabrali ovaj primer da bi ilustrovali šta je to „Komodor" trebalo da dopiše u bejzik interpreteru da bi oslobodio korisnike razmišljanja o otvaranju veze i sekundarnoj adresi. Ujedno ćemo na tom primeru objasniti i pojedine potprograme operativnog sistema zadužene za komunikaciju. Dodavanje nove naredbe izvedeno je na način objašnjen u „Računarima 6", pa ga ovde nećemo obrazlagati.

Da bi se otvorila veza, potrebno je prvo postaviti odgovarajuće parametre. Ovo će obaviti dva potprograma, SETLFS (set logical, first and secondary addres) i SETNAM (set name), koji se pozivaju na sledeći način:

A = logička adresa
X = broj uređaja (prva adresa)
Y = sekundarna adresa
JSR SETLFS
A = dužina naziva
XY = početna adresa naziva (X niži bajt)
JSR SETNAM
U našem primeru nema potrebe za nazivom datoteke, pa je rutina koja ga postavlja pozvana sa nulom u A. Svejedno koju ćete od navedene dve rutine prvo pozvati. Važno je samo da su obe izvedene pre sledeće, a to je OPEN. Nakon nje, veza je otvorena na identičan način kao da je u bejziku izvedeno OPEN 100,4,0. Primećujete da nismo testirali CARRY zastavicu. Pretpostavili smo da se neće desiti da je negde u programu, pre izvođenja naredbe LPRINT, već otvorena veza sa logičkom adresom 100, što je praktično jedina greška koja bi u ovom trenutku mogla nastati (ako ste to ipak učinili, dešavaće se neobične stvari).

Veza je neaktivna sve dok se ne zada smer komunikacije. U bejziku se pri izvođenju PRINT# naredbe veza automatski proglašava izlaznom, ali u mašinskom programu se to mora posebno izvesti. Za to služi rutina sa ulaznim parametrom u X registru:

X = logički broj veze
JSR CHKOUT

Dejstvo ovog potprograma je slično kao naredbe CMD u bejziku (CMD se u suštini lokalno izvodi pre svakog PRINT#, ali se odmah po ispisu njeno dejstvo prekida). Sada je ispis stalno usmeren na štampač i to će potrajati dok se ne izvede rutina:

CLRCHN (clear I/0 channels)

Njen zadatak je da komanduje svim uređajima da prestanu sa „slušanjem" i da vrednost sistemske promenjive 154 vrati na 3 (za ispis na ekran). Dakle treba je izvesti odmah po završetku slanja, ili prijema. Ako to ne učinimo već otvorimo novu vezu za slanje, podaci će ići na obe. Ovo nije greška već predviđena mogućnost koja otvara interesantne primene, ali važi samo za uređaje priključene preko serijskog interfejsa.

CLRCHN ne zatvara vezu. Nakon ove rutine stanje je isto kao i odmah nakon OPEN. Da bi novi podaci bili poslati preko iste veze, dovoljno je prethodno izvesti samo CHKOUT.

Još jednom o greškama. U programu smo, nakon CHKOUT, ipak ispitali CARRY zastavicu zbog toga što je to prva rutina po otvaranju koja stvarno i dovodi računar u vezu sa periferijom (ako nema naziva datoteke, OPEN završava u samom računaru). Tada je tek moguće ustanoviti da li je uređaj priključen, pa je u tu svrhu testiranje i izvršeno. Ako nije, u akumulatoru će se naći broj 5 što je, igrom slučaja, i broj bejzik poruke DEVICE NOT PRESENT ERROR. „Igra slučaja" važi i za ostalih 8 mogućih grešaka! Jasno, ovo nije slučajnost ali ipak treba imati u vidu da se radi o odvojenim porukama. Ovako izabrani brojevi grešaka, tačnije raspred bejzik grešaka, omogućuje da se iskoristi i bejzik deo kako je to u ovom primeru i urađeno (X = broj bejzik greške, pa JSR 42039).

Zatvaranje veze se izvodi jednostavno sa:

PRILOG 2

```
2
49152                           .OPT P4
49152                           *=   49152
49152                  SETNAM   =    65469
49152                  SETLFS   =    65466
49152                  OPEN     =    65472
49152                  CHKIN    =    65478
49152                  CHKOUT   =    65481
49152                  CHRIN    =    65487
49152                  GETIN    =    65508
49152                  CHROUT   =    65490
49152                  CLRCHN   =    65484
49152                  CLOSE    =    65475
49152                  READST   =    65463
49152                  INPUT    =    42336
49152                  PRINT    =    43806
      ;//////////////////////////
      ;/  IZMENA BWS VEKTORA  /
      ;//////////////////////////
49152  169 011                  LDA  #<NEWBWS
49154  162 192                  LDX  #>NEWBWS
49156  141 002 003              STA  770
49159  142 003 003              STX  771
49162  096                      RTS
      ;//////////////////////////
      ;/       NOVI BWS       /
      ;//////////////////////////
49163  032 096 165   NEWBWS     JSR  INPUT
49166  134 122                  STX  122
49168  132 123                  STY  123
49170  032 115 000              JSR  115
49173  170                      TAX
49174  240 243                  BEQ  NEWBWS
49176  201 093                  CMP  #"]"
49178  240 006                  BEQ  DISKCOM
49180  032 121 000              JSR  121
49183  076 144 164              JMP  42128
      ;//////////////////////////
      ;/    DISK NAREDBE      /
      ;//////////////////////////
49186  169 255        DISKCOM   LDA  #255
49188  133 058                  STA  58
49190  032 115 000              JSR  115
49193  201 036                  CMP  #"$"
49195  208 003                  BNE  CP1
49197  076 077 192              JMP  DIRCT
49200  201 083        CP1       CMP  #"S"
49202  208 003                  BNE  OSTL
49204  076 110 192              JMP  SCRATCH
      ;//////////////////////////
      ;/ OSTALE DISK NAREDBE  /
      ;//////////////////////////
49207  032 253 192    OSTL      JSR  OPEERR
49210  162 099                  LDX  #99
49212  056                      SEC
49213  032 040 193              JSR  DIREC
49216  169 001                  LDA  #1
49218  160 002                  LDY  #2
49220  032 030 171              JSR  PRINT
49223  032 111 193              JSR  READER
49226  076 134 227              JMP  58246
      ;//////////////////////////
      ;/ DIREKTORIJ DISKETE   /
      ;//////////////////////////
49229  032 253 192    DIRCT     JSR  OPEERR
49232  032 014 193              JSR  OPDIR
49235  162 098                  LDX  #98
49237  024                      CLC
49238  032 040 193              JSR  DIREC
49241  032 058 193              JSR  GETBYT
49244  032 058 193              JSR  GETBYT
49247  032 080 193    DRD       JSR  GETLIN
49250  169 013                  LDA  #13
49252  032 210 255              JSR  CHROUT
49255  173 141 002    WAIT      LDA  653
49258  208 251                  BNE  WAIT
49260  240 241                  BEQ  DRD
      ;//////////////////////////
      ;/  BRISANJE FAJLA      /
      ;//////////////////////////
49262  032 115 000    SCRATCH   JSR  115
49265  170                      TAX
49266  240 195                  BEQ  OSTL
49268  201 042                  CMP  #"*"
49270  208 246                  BNE  SCRATCH
49272  032 253 192              JSR  OPEERR
49275  169 147        AGAIN     LDA  #"Q"
49277  032 210 255              JSR  CHROUT
49280  032 014 193              JSR  OPDIR
49283  162 098                  LDX  #98
49285  024                      CLC
49286  032 040 193              JSR  DIREC
```

```
PRILOG 1

2
49152                          .OPT P4
49152                          **  49152
49152                SETNAM    =   65469
49152                SETLFS    =   65466
49152                OPEN      =   65472
49152                CHKOUT    =   65481
49152                CLRCHN    =   65484
49152                CLOSE     =   65475
     ;//////////////////////////
     ;/  IZMENA BCD VEKTORA   /
     ;//////////////////////////
49152  169 011               LDA  #<NEWBCD
49154  162 192               LDX  #>NEWBCD
49156  141 000 003           STA  776
49159  142 009 003           STX  777
49162  096                   RTS
     ;//////////////////////////
     ;/   NOVA BCD RUTINA     /
     ;//////////////////////////
49163  032 115 000   NEWBCD   JSR  115
49166  201 076               CMP  #"L"
49168  240 006               BEQ  NEXTCHK
49170  032 121 000   OLDBCD   JSR  121
49173  076 231 167           JMP  42983
49176  160 001       NEXTCHK LDY  #1
49178  177 122               LDA  (122),Y
49180  201 153               CMP  #153
49182  208 242               BNE  OLDBCD
49184  032 115 000           JSR  115
49187  032 115 000           JSR  115
49190  008                   PHP
49191  072                   PHA
     ;//////////////////////////
     ;/   NAREDBA LPRINT      /
     ;//////////////////////////
     ;
     ;LOG, PRV, I SEK. ADRESA
     ;
49192  169 100               LDA  #100
49194  162 004               LDX  #4
49196  160 000               LDY  #0
49198  032 186 255           JSR  SETLFS
     ;
     ;IME
     ;
49201  169 000               LDA  #0
49203  032 189 255           JSR  SETNAM
     ;
     ;OTVARANJE VEZE
     ;
49206  032 192 255           JSR  OPEN
49209  162 100               LDX  #100
49211  032 201 255           JSR  CHKOUT
49214  144 004               BCC  START
     ;
     ;AKO JE NASTALA GRESKA
     ;
49216  170                   TAX
49217  076 055 164           JMP  42039
     ;
     ;VRACANJE PRVOG ZNAKA ZA STAMPANJE
     ;PA BASIC PRINT RUTINA
     ;
49220  104           START   PLA
49221  040                   PLP
49222  032 160 170           JSR  43680
     ;
     ;KRAJ STAMPANJA, ZATVARANJE VEZE
     ;
49225  032 204 255   EXIT    JSR  CLRCHN
49228  169 100               LDA  #100
49230  032 195 255           JSR  CLOSE
49233  076 174 167           JMP  42926
49202  208 003               BNE  START
49204  076 110 192           JMP
     ;//////////////////////////
     ;/ OSTALE DISK NAREDBE   /
     ;//////////////////////////
49207  032 253 192   OSTL    JSR  OPEERR
49210  162 098               LDX  #98
49212  056                   SEC
49213  032 040 193           JSR  DIREC
49216  169 001               LDA  #1
49218  160 002               LDY  #2
49220  032 030 171           JSR  PRINT
49223  032 111 193           JSR  READER
49226  076 134 227           JMP  58246
     ;//////////////////////////
     ;/ DIREKTORIJ DISKETE    /
     ;//////////////////////////
49229  032 253 192   DIRCT   JSR  OPEERR
49232  032 014 193           JSR  OPDIR
49235  162 098               LDX  #98
49237  024                   CLC
49238  032 040 193           JSR  DIREC
49241  032 058 193           JSR  GETBYT
49244  032 058 193           JSR  GETBYT
49247  032 080 193   DRD     JSR  GETLIN
49250  169 013               LDA  #13
49252  032 210 255           JSR  CHROUT
49255  173 141 002   WAIT    LDA  653
49258  208 251               BNE  WAIT
49260  240 241               BEQ  DRD
     ;//////////////////////////
     ;/  BRISANJE FAJLA       /
     ;//////////////////////////
49262  032 115 000   SCRATCH  JSR  115
49265  170                   TAX
49266  240 195               BEQ  OSTL
49268  201 042               CMP  #"*"
49270  208 246               BNE  SCRATCH
49272  032 253 192           JSR  OPEERR
49275  169 147       AGAIN   LDA  #"J"
49277  032 210 255           JSR  CHROUT
49280  032 014 193           JSR  OPDIR
49283  162 098               LDX  #98
49285  024                   CLC
49286  032 040 193           JSR  DIREC
49289  032 058 193           JSR  GETBYT
49292  032 058 193           JSR  GETBYT
49295  032 080 193           JSR  GETLIN
49298  169 013               LDA  #13
49300  032 210 255           JSR  CHROUT
49303  032 080 193   ASK     JSR  GETLIN
49306  230 211       POS     INC  211
49308  166 211               LDX  211
49310  224 032               CPX  #32
49312  144 248               BCC  POS
49314  169 058               LDA  #":"
49316  032 210 255           JSR  CHROUT
49319  032 204 255           JSR  CLRCHN
49322  032 228 255   GETKEY  JSR  GETIN
49325  240 251               BEQ  GETKEY
49327  170                   TAX
49328  201 068               CMP  #"D"
49330  240 002               BEQ  MSG
49332  169 078               LDA  #"N"
49334  032 210 255   MSG     JSR  CHROUT
49337  169 013               LDA  #13
49339  032 210 255           JSR  CHROUT
49342  224 068               CPX  #"D"
49344  208 050               BNE  NEXT
49346  169 145               LDA  #145
49348  032 210 255           JSR  CHROUT
49351  169 098               LDA  #98
49353  032 195 255           JSR  CLOSE
49356  169 003               LDA  #3
49358  133 153               STA  153
49360  032 096 165           JSR  INPUT
49363  169 000               LDA  #0
49365  133 153               STA  153
49367  169 083               LDA  #"S"
49369  141 003 002           STA  515
49372  169 058               LDA  #":"
49374  141 004 002           STA  516
49377  162 099               LDX  #99
49379  056                   SEC
49380  032 040 193           JSR  DIREC
49383  169 003               LDA  #3
49385  160 002               LDY  #2
49387  032 030 171           JSR  PRINT
49390  032 204 255           JSR  CLRCHN
49393  076 123 192           JMP  AGAIN
49396  162 098       NEXT    LDX  #98
49398  024                   CLC
49399  032 040 193           JSR  DIREC
49402  076 151 192           JMP  ASK
     ;
     ;OTVARANJE ERROR KANALA
     ;
49405  169 099       OPEERR  LDA  #99
49407  162 008               LDX  #8
49409  160 015               LDY  #15
49411  032 186 255           JSR  SETLFS
49414  169 000               LDA  #0
49416  133 183               STA  183
49418  032 192 255           JSR  OPEN
49421  096                   RTS
     ;
     ; OTVARANJE DIR. FAJLA
     ;
49422  169 098       OPDIR   LDA  #98
49424  162 008               LDX  #8
49426  160 000               LDY  #0
49428  032 186 255           JSR  SETLFS
49431  169 036               LDA  #"$"
49433  133 002               STA  2
49435  169 001               LDA  #1
49437  162 002               LDX  #2
49439  160 000               LDY  #0
49441  032 189 255           JSR  SETNAM
49444  032 192 255           JSR  OPEN
49447  096                   RTS
     ;
     ;IZBOR SMERA KOMUNIKACIJE
     ;
49448  144 006       DIREC   BCC  IN
49450  032 201 255           JSR  CHKOUT
49453  176 007               BCS  ERR
49455  096                   RTS
49456  032 198 255   IN      JSR  CHKIN
49459  176 001               BCS  ERR
49461  096                   RTS
49462  170           ERR     TAX
49463  076 055 164           JMP  42039
     ;
     ; UZIMANJE 1 BYTE-A IZ FAJLA
     ;
49466  032 207 255   GETBYT  JSR  CHRIN
49469  170                   TAX
49470  032 183 255           JSR  READST
49473  208 002               BNE  END
49475  138                   TXA
49476  096                   RTS
49477  032 111 193   END     JSR  READER
49480  169 098               LDA  #98
49482  032 195 255           JSR  CLOSE
49485  076 134 227           JMP  58246
     ;
     ; CITANJE I STAMPANJE I NAZIVA
     ;
49488  032 058 193   GETLIN  JSR  GETBYT
49491  032 058 193           JSR  GETBYT
49494  032 058 193           JSR  GETBYT
49497  072                   PHA
49498  032 058 193           JSR  GETBYT
49501  168                   TAY
49502  104                   PLA
49503  170                   TAX
49504  152                   TYA
49505  032 205 189           JSR  48589
49508  032 058 193   PRT     JSR  GETBYT
49511  240 005               BEQ  FINISH
49513  032 210 255           JSR  CHROUT
49516  144 246               BCC  PRT
49518  096           FINISH  RTS
     ;
     ; OCITAVANJE I ZATV. ERROR KANALA
     ;
49519  032 204 255   READER  JSR  CLRCHN
49522  162 099               LDX  #99
49524  024                   CLC
49525  032 040 193           JSR  DIREC
49528  032 096 165           JSR  INPUT
49531  032 204 255           JSR  CLRCHN
49534  169 099               LDA  #99
49536  032 195 255           JSR  CLOSE
49539  169 000               LDA  #0
49541  160 002               LDY  #2
49543  032 030 171           JSR  PRINT
49546  096                   RTS
```

A = logički broj veze
JSR CLOSE

Nešto bolji DOS

Primer 2 je nešto složeniji i ilustruje rad sa disk jedinicom kao i rad sa tastaturom i video memorijom niske rezolucije kao perifernim uređajima. Program bi se uslovno mogao nazvati DOS, jer omogućava korišćenje stvarnih DOS naredbi (koje se prepoznaju i izvode u samoj disk jedinici) na isti način na koji se koriste i bejzik naredbe. Izmena BWS vektora omogućava rad programa isključivo u direktnom modu.

Prvi deo (NEWBWS) je prepisan iz bejzik ROM-a. Linija koju otkucate do pritiska na taster RETURN naći će se u INPUT baferu, od adrese 512. Ako se na prvom mestu, ne računajući razmake o čemu se brine JSR 115, nađe znak uglaste zagrade, program nastavlja sa radom. U suprotnom se izvršavanje prepušta bejziku (tokenizovanje naredbi i izvršavanje ili smeštanje linije u program).

Dodali smo i dve opcije koje inače nisu na raspolaganju. Ako otkucate 1$ na ekranu će biti izlistan direktorij, bez upisivanja u memoriju računara. Ako želite da obrišete neku datoteku sa diskete pa zadate 1S:* disk bi shvatio da treba obrisati sve. U tom slučaju program neće odmah proslediti naredbu već će listati direktorij, jedan po jedan ulaz, i tražiti da pritisnete „D" za brisanje ili „N" ako brisanje nije potrebno. Tako se više datoteka može brže i jednostavnije izbrisati nego stalnim ponavljanjem naredbe.

U svakom drugom slučaju, bez obzira šta napišete iza zagrade, string se prosleđuje disku na obradu. Ovaj posao obavlja deo od labele OSTL. Otvara se komandni kanal (sekundarna adresa 15) i proglašava izlaznim. Naredba se šalje rutinom iz bejzik ROM-a (43806), čiji je ulazni parametar početna adresa stringa u A i X registru (A niži bajt). Kao terminator služi 0 ili 34 (znak navoda). Ovde je upotrebljena 0 koju rutina INPUT (42336) automatski postavlja za ulazni (uz prethodno CLRCHN) i očitava poruka sa diska (potprogram READER).

Moguće je otkucati samo zagradu. Tada se disku ne šalje ništa već samo očitava komandni kanal, što može biti korisno kad u toku rada crvena LED dioda, iz sasvim „nerazumljivih" razloga, počne da trepće.

Direktorij diska je posebna datoteka sa preciznim formatom, ali je u DOS ugrađena i mogućnost njenog tretiranja kao bilo kog programa. Treba ga otvoriti sa sekundarnom adresom 0 (u početku je rečeno da je ona rezervisana za LOAD naredbu) i disk će ga poslati u obliku koji odgovara programskoj datoteci. Prva dva bajta predstavljaju adresu od koje se program smešta u memoriju ako se upisuje sa LOAD „ime",8,1. Zatim za svaki ulaz slede dva bajta, takozvana „veza sa sledećom linijom" i još dva koja predstavljaju linijski broj, u ovom slučaju broj blokova koje datoteka zauzima. Zatim sledi naziv i bajt nula kao terminator linije.

Otvaranje direktorija je napisano kao potprogram OPDIR. Kao naziv potreban je samo znak $, pa smo ga smestili na adresu 2 (potpuno proizvoljno). Uzimanje jednog bajta sa veze poverili smo potprogramu GETBYT. On koristi rutinu operativnog sistema CHRIN (input character) i odmah proverava stanje STATUS varijable. U slu-

čaju da je vrednost različita od nule, znači da je uzet poslednji bajt iz datoteke, ili je nastala neka greška u čitanju sa diska. Tada se dalje čitanje prekida, očitava komandni kanal da bi se dobila poruka o stvarnoj grešci i program prekida sa radom. JMP 58246 je skok na takozvanu „farmu" za inicijalizaciju steka, ispis READY itd.

Čitanje bilo koje sekvencijalne ili programske datoteke se izvodi na identičan način. Tada je nepotrebna rutina GETLIN u kojoj se dva bajta (broj zauzetih blokova) štampaju kao string.

Opcija za brisanje datoteka otvara direktorij i uzima njegov sadržaj na prethodno opisan način. Brisanje ekrana se postiže slanjem karaktera 147. Rutina:

A = bajt
JSR CHROUT

šalje jedan bajt na uređaj koji je u tom trenutku proglašen otvorenim za ispis. Kako to u ovom slučaju nije urađeno prethodno ni za jednu vezu, ispis je usmeren na video memoriju.

Kada se ispiše naziv jedne datoteke, treba očitati tastaturu da bi se ustanovilo koji je taster pritisnut. Ovo se postiže pozivom rutine GETIN, koja u akumulatoru vraća kod pritisnutog tastera. Ako nijedan nije pritisnut, vrednost će biti 0. Primećujete da smo prethodno resetovali sve ulazne i izlazne veze sa CLRCHN. To nije urađeno slučajno, jer je rutina GETIN dizajnirana za čitanje ne samo tastature već i bilo koje veze koja je otvorena za upis. Istu namenu ima i ranije pomenuta CHRIN. U čemu je onda razlika?

GETIN je prevashodno namenjena tastaturi i RS232 interfejsu i uzima samo jedan bajt. Kada tastaturu očitavate CHRIN rutinom, na ekranu se pojavljuje kursor i izvodi isto što i zadavanjem bejzik INPUT naredbe. Tek kad detektuje taster RETURN, računar napušta ovu rutinu i vraća kod znaka koji je prvi otkucan. Sledeći poziv uzima redom dalje otkucane znake i kad su svi obrađeni vraća bajt 13. Novi poziv ponavlja proceduru. Upotreba GETIN umesto CHRIN za ostale uređaje (osim RS232) ne pravi razliku.

Ako ste pritisnuli taster „D" potrebno je da se disku pošalje naredba:

S:naziv datoteke

Kako dobiti naziv datoteke, kada je jedino ispisan na ekranu? Uzimanje direktno iz video memorije stvara dosta problema, jer treba ispitati na kojoj liniji se naziv nalazi i, ujedno, ekranske kodove prevesti u ASCII. Rešenje je jednostavno jer „Komodor" i video memoriju tretira kao periferni uređaj. INPUT sa ekrana će smestiti u input bafer sadržaj jedne linije od trenutnog položaja kursora do kraja reda ili znakova „:" i „," Pri tome se početna prazna mesta ignorišu (INPUT prazne linije vraća prazan string).

Da bismo otvorili vezu sa video memorijom i proglasili je otvorenom za upis, upotrebili smo „nedozvoljenu" proceduru: nismo je ni otvarali, nismo pozivali CHKIN, već smo sistemskoj promenjivoj 153 (broj ulazne jedinice) dodelili vrednost 3. Ovo je jedini sučaj u kome je ovakva procedura moguća — CHKIN u tom slučaju uradi potpuno isto, ali smo ovaj način dali radi ilustracije, a ne kao preporuku. Ušteda od 14 bajtova koja se njom postiže nije adekvatna gubitku razumljivosti vašeg programa.

Postavljanje kursora na početak linije smo obezbedili ispisom karaktera 13 (novi red) i 145 (kursor gore). U input bafer se dodaje slovo S i prvi znak navoda zamenjuje sa :.

Kada je naredba izvršena, direktorij mora ponovo da se čita od početka, jer smo mu brisanjem jedne datoteke izmenili sadržaj. Ovo se može izbeći prethodnim upisivanjem celog direktorija u memoriju, pa tek onda njegovom obradom, što bi zahtevalo odvajanje memorijskog prostora (u najgorem slučaju oko 3 K) ili upisom direktorija u memoriju blok po blok, kao RANDOM datoteke uz vođenje računa o formatu u kome su ulazi zapisani.

Šta je sa naredbama SAVE i LOAD? Njihovo izvršavanje je povereno posebnim rutinama operativnog sistema sa istim nazivima. Da bi segment memorije snimili na disketu, treba već pomenutim rutinama SETNAM i SETLFS postaviti naziv i logičku, prvu i sekundarnu adresu. Zatim se poziva SAVE (adresa 65496) na sledeći način:

A = adresa u nultoj strani i na kojoj se, zajedno sa sledećom, nalazi adresa prvog bajta koji treba snimiti
XY = adresa poslednjeg bajta (X niži bajt)
JSR SAVE

Potreba za SETLFS u, ovom slučaju, može da zbuni. SAVE rutina uopšte ne uzima u obzir, logički broj, a u slučaju diska ni sekundarnu adresu, već je uvek postavlja na 1. Zato u bejziku nema baš nikakve razlike u oblicima SAVE „ime" ,8 ili SAVE „ime",8,1. Sa kasetofonom je situacija nešto drugačija. Ako se snimanje izvrši sa sekundarnom adresom 1, program će kasnije, pri LOAD, uvek biti smešten na isto mesto sa koga je i snimljen, bez obzira na sekundarnu adresu u LOAD naredbi. Ako upotrebimo vrednost 2, posle programa će biti zapisano još jedno zaglavlje kao oznaka „kraj trake". Sekundarna adresa 3 kombinuje ove dve osobine.

Rutina LOAD (adresa 65493) je kombinovana sa VERIFY i zahteva postavljanje parametara sa SETNAM i SETLFS. Poziva se sa:

A = 0 za LOAD ili
A = 1 za VERIFY
XY = početna adresa
JSR LOAD

U ovom slučaju, sekundarna adresa ima svoj smisao. Ako je 1, nije potrebno da u XY registrima navodite početnu adresu. Biće iskorišćena ona sa koje je program snimljen. Ako je 0, onda se obavezno zadaje početna adresa, što omogućuje relociran upis.

Načuli uvo i slušaj

U prethodnim primerima dosta smo koristili dve rutine iz bejzik ROM-a: INPUT i PRINT. Obe u sebi sadrže petlje unutar kojih su CHRIN i CHROUT. U većini programa ćete sresti slučaj da su autori sami dopisivali slične potprograme. Ovo je svakako preporučljivije iz prostog razloga što ovakvi programi ne treba da se oslanjaju na pomoć bilo kog jezičkog interpretera da bi radili sa svakim od njih. Ujedno, čest je slučaj da se bejzik ROM izbacuje iz adresnog prostora mikroprocesora i za program koristi RAM „ispod" njega.

Pažljiviji čitalac je sigurno primetio da smo na nekoliko mesta u ovom tekstu upotrebili izraze „komanduje spoljnom uređaju da sluša" i slične, bez objašnjenja. Ovo je specifičnost uređaja priključenih preko serijskog interfejsa, ali se primećuje samo lokalno: kada se kanal proglašava otvorenim za ispis, korisniku je svejedno o kom se uređaju radi, uvek se koristi ista rutina CHROUT. Za štampač ili disk, računar preduzima daleko više radnji nego što je to slučaj za tastaturu ili video memoriju. Tako dolazimo do jednog, još elementarnijeg nivoa komunikacije sa ovim uređajima koji je u komercijalnim programima dosta čest.

Sve rutine za komunikaciju sa ovim uređajima koriste 8 osnovnih:

LISTEN (65457) — naređuje spoljnom uređaju da pređe na prijem (da „sluša")
SECOND (65427) — šalje sekundarnu adresu nakon LISTEN
CIOUT (65448) — „šalje jedan bajt
UNLSEN (65454) — komanduje svim uređajima da prestanu sa prijemom
TALK (65460) — komanduje spoljnjem uređaju da počne sa slanjem podataka
TKSA (65430) — šalje sekundarnu adresu nakon TALK
ACPTR (65445) — prima jedan bajt
UNTLK (65451) — komanduje svim uređajima da prestanu sa slanjem

Da bi se, na primer, odštampao neki znak na štapaču, bio bi potreban sledeći program:

```
LDA # 4
JSR LISTEN
LDA # 96
JSR SECOND
LDA # bajt
JSR CIOUT
JSR UNLSN
```

Prvo što se primećuje je neobična sekundarna adresa. Njena vrednost je, u stvari, 0, ali su joj bit 6 i 5 setovani. Ranije opisana rutina OPEN ovo obavlja automatski, ali ovde mi moramo o tome da vodimo računa. Sekundarnoj adresi se mora posvetiti dodatna pažnja i ako se otvara datoteka na disku, ali se ovde nećemo upuštati u detalje.

Nakon upoznavanja sa ovim elementarnim rutinama, postaje jasno kako su izvedene ostale. Tako, na primer, CHKOUT izvodi LISTEN i SECOND uz jedan dodatak: ako ste upotrebili sekundarnu adresu u kojoj je bit 7 jedinica, dakle veću od 127, onda kao takva neće biti poslata već će se spoljnjem uređaju staviti do znanja da sekundarna adresa ne sledi, što svaki od njih može protumačiti na svoj način. Štampač će se, na primer, postaviti kao da je primio sekundarnu adresu 0.

Moglo bi se reći da pisanje programa na ovaj način ima opravdanja samo u slučaju sistemskih programa. Ovako se izbegava svaki moguć nesporazum oko toga da li je korisnik u programu upotrebio logič ! broj veze koji je i nama potreban, a i izvršavanje programa je nešto brže.

Zoran Životić

Programiranje u bejziku
funkcije, potprogrami, procedura

Kad god je potrebno da se neki postupak više puta ponovi u programu, onda je zgodno da se naredbe koje ga opisuju izdvoje u posebnu celinu — potprogram (sabroutine) koja bi se pozivala iz svake tačke programa u kojoj je neophodno primeniti taj postupak i, po izvršenju, vraćala upravljanje glavnom programu na naredbu koja sledi neposredno iza poziva potprograma. Ovakva organizacija programa omogucava njegov kraći zapis. Ako je zapis postupka zahtevao **n** programskih redova i ponavljao se za razne argumente **m** puta u programu, onda umesto n*m programskih redova bez korišćenja potprograma pišemo samo **n** redova (+m poziva i bar **m** povratka u program). Ovo ima za posledicu smanjenje verovatnoće grešaka pri pisanju programa, olakšano testiranje i uštedu memorijalnog prostora. Ali, sve ima svoju cenu — gubi se na brzini izvršavanja programa. Dužina izvršavanja programa uvećava se za vreme potrebno za realizaciju naredbi poziva i povratka i prenos argumenata. Potpuno sprovedena koncepcija potprograma podrazumeva da su oni potpuno izdvojene celine, nezavisne od glavnog programa, te se, prema tome, mogu koristiti iz raznih programa i imaju sopstvene — lokalne promenljive koje nemaju nikakve veze sa istoimenim promenljivim glavnih programa. To omogućava timski rad na pisanju aplikacija jer je dovoljno da jedan programer d a globalni opis rešenja, a razradu detalja može prepustiti svojim kolegama koji će to učiniti u zasebnim potprogramima. Glavni programer projektuje samo veze između ovih celina i naznačava koje su promenljive globalne, tj. zajedničke za program i pojedine potprograme.

Uvedene funkcije

Odmah da kažemo da bejzici računara o kojima pišemo nemaju ovu poslednju mogućnost, dakle imaju vrlo veliku manu (napominjem da ovde ne govorimo o sajmonsu i drugim proširenim bejzicima koji imaju tako dobre i bogate mogućnosti da zaslužuju da se pomenu u svakoj teoriji programskih jezika). Ono čime raspolažemo u bejzicima kućnih računara su mogućnost uvođenja funkcija, tzv. funkcijskih naredbi (sa izuzetkom „galaksije", ali i ZX81, TRS 80 i drugih koji ih nemaju) i korišćenja internih potprograma — potprogramskih segmenata koji mogu da stoje jedino u okviru glavnog programa. Nešto bolje mogunosti imaju jedino Acornovi računari.

Računari obično imaju ugrađene potprograme za računanje elementarnih funkcija koji se pozivaju navođenjem imena funkcije sa konkretnim argumentima za koje treba izračunati vrednost. Naredba

PRINT SIN (0)

poziva ugrađeni potprogram za računanje vrednosti funkcije sinus po manje ili više dobroizabranoj približnoj metodi. Argument 0 je ulazna veličina u potprogram, stvarni argument za koji se računa vrednost po postupku opisanom za fiktivne argumente. Izračunata vrednost se, zatim, vraća u program (ovde u naredbu u direktnom režimu) i štampa. Nemaju svi računari isti broj ugrađenih funkcija, niti je računanje njihovih vrednosti istog kvaliteta. Od računara o kojima govoriomo najlošija je situacija sa „galaksijom" — ona ne samo što ima najmanje funkcija, nego su one i najlošije urađene (kao uostalom, i na TRS 80). Ništa bolja situacija nije ni kod „spektruma" — to sigurno nije njegov adut. „Komodor 64" ima podnošljivo urađene elementarne funkcije sa ponekim „biserom" (o tome ćemo drugom prilikom), a Acornovi računari, mada nisu oslobođeni bagova, nude pristojne mogućnosti.

Mogućnost uvođenja korisničkih funkcija je veoma važna, ne samo što tako možemo definisati sinus hiperbolički, neki obrazac za proračun kamate ili bilo koju drugu funkciju koja nam je potrebna, već i zato što možemo umesto loše isprogramirane ugrađene funkcije da uvedemo svoju koja će davati tačnije vrednosti. Na žalost, kao što rekosmo ranije, „galaksijin" bejzik nam ne pruža ovu mogućnost.

Uvedene funkcije — funkcijske naredbe definišu se posebnom naredbom oblika:

DEF <ime funkcije> (<argumenti>)=<izraz>

Ime uvedene funkcije obavezno počinje slovima FN za kojima slede jedno ili dva slova kod „komodora" (drugi simbol može biti i cifra, a teoretski je moguće da se stavi i duže ime, ali kao takvo neće biti upamćeno, pa to, praktično, nema smisla — dakle nastavak može biti ime realne promenljive, ime azbučne ili celobrojne promenljive uzrokuje prijavu greške), a kod „spektruma" i Acorna imamo veću komociju — ime može imati i više simbola.

Po broju argumenata koji se mogu navesti opet „komodor" ima lošije mogućnosti od druga dva računara — može da prihvati samo jedan argument za razliku od više njih koliko je dozvoljeno za „spektrum" i Acorn i, što je naročito velik hendikep za „komodora" — on omogućava jedino uvođenje numeričkih funkcija, dok druga dva računara mogu da rade i sa uvedenim azbučnim funkcijama. Ilustrujmo korišćenje funkcijske naredbe jednim jednostavnim programom.

spektrum
```
10 CLS
20 PRINT „BROJ", „KUB"
30 FOR a=1 TO 20
40 PRINT a, FNk(a)
50 NEXT a
60 DEF FNk(x)=x*x*x
```

komodor
```
10 DEF FNK(X)=X*X*C
20 PRINT „BROJ", „KUB"
30 FOR A=1 TO 13
40 PRINT A, FNK(A)
50 NEXT A
```

Acorn
```
10 CLS
20 PRINT „BROJ", „KUB"
30 FOR A=1 TO 13
40 PRINT A, FNK(A)
50 NEXT A
60 END
70 DEF FNK(X)
80=X*X*X
```

Možda vam se čini da je ovo program bez smisla; jer treći stepen možete dobiti i pišući jednostavno X ↑ 3, ali grešite, jer se ovako računa tačnije i skoro deset puta brže vrednost trećeg stepena. U to se možete uveriti ako izbacite definiciju funkcijske naredbe i umesto FNK štampate vrednost A ↑ 3. (Radi boljeg uvida u brzinu, stavite da ciklus ide do 1000)

Snaga argumenata

Ono što ste, verovatno, primetili iz ova tri listinga je da se kod Acorna ne mora definisati vrednost u jednom programskom redu. Po tome kod njega uvedene funkcije, u stvari, prevazilaze konvencionalni pojam funkcijske naredbe, jer omogućavaju da se postupak za računanje funkcije definiše u više programskih redova — dakle pre bi se moglo reći da je ovo analogon za funkcijski potprogram u fortranu (koji, za razliku od opšteg potprograma, ima samo jednu izlaznu veličinu). Kada se još doda mogućnost rekurzivne definicije funkcije, o kojoj ćemo posebno govoriti, dobijaju se moćna svojstva o kojima možda niste ni slutili. Formalna razlika u odnosu na definisanje funkcija kod drugih računara je da se opisna naredba (poput DATA, na primer) DEF FN mora nalaziti iza fizičkog kraja programa. „Spektrum", pak, uvek traži DEF FN od početka programa kad se pomene ime neke uvedene funkcije, pa je, naročito kod dužih programa, dobro sve funckije definisati na samom početku programa.

Argumenti koji se navode između zagrada u definiciji funkcijske naredbe zovu se fiktivni argumenti. Fiktivni, jer se razlikuju od promenljivih u programu — za njih program ne rezerviše posebne memorijske registre i možete sasvim nezavisnoi od fiktivnih argumenata u programu koristiti istoimene promenljive. Možemo fiktivne argumente smatrati nekom vrstom lokalnih promenljivih za funkcijsku naredbu.

Rekli smo da „spektrum" i Acorn računari mogu da koriste uvedene funkcije sa više agrumenata. Otkucajte NEW i unesite sledeći program.

Danas je nezamislivo da se rešenje bilo koje veće aplikacije daje u obliku jednog programa. Koncept potprograma, predložen u samu zoru razvoja programskih jezika, zahvaljujući nesumnjivim olakšicama koje pruža, još nije sišao s top liste omiljenih programerskih tehnika. Ova koncepcija ugrađena je, sa više ili manje mogućnosti, u sve više programske jezike. Da bismo analizirali koliko je uspešno izvršeno njeno ugrađivanje u bejzike kućnih računara, zadržimo se prethodno na samoj ideji potprograma i pokušajmo da sagledamo šta sve ona nudi.

„Spektrum", Acorn

```
10 DEF FNa(a, b, c, d, f)=a*b*b
20 PRINT „unesite dva broja"
30 INPUT a, b
40 PRINT FNa(a, b)
```

Posle startovanja programa videćete da on — ne radi. Ali ako promenite liniju 40 na sledeći način:

```
40 PRINT FNa(a, b, O, O, O)
```

program će raditi. Upamtite da se prilikom svakog poziva funkcije moraju navesti stvarni umesto svakog fiktivnog argumenta.

Potprogramski segmenti

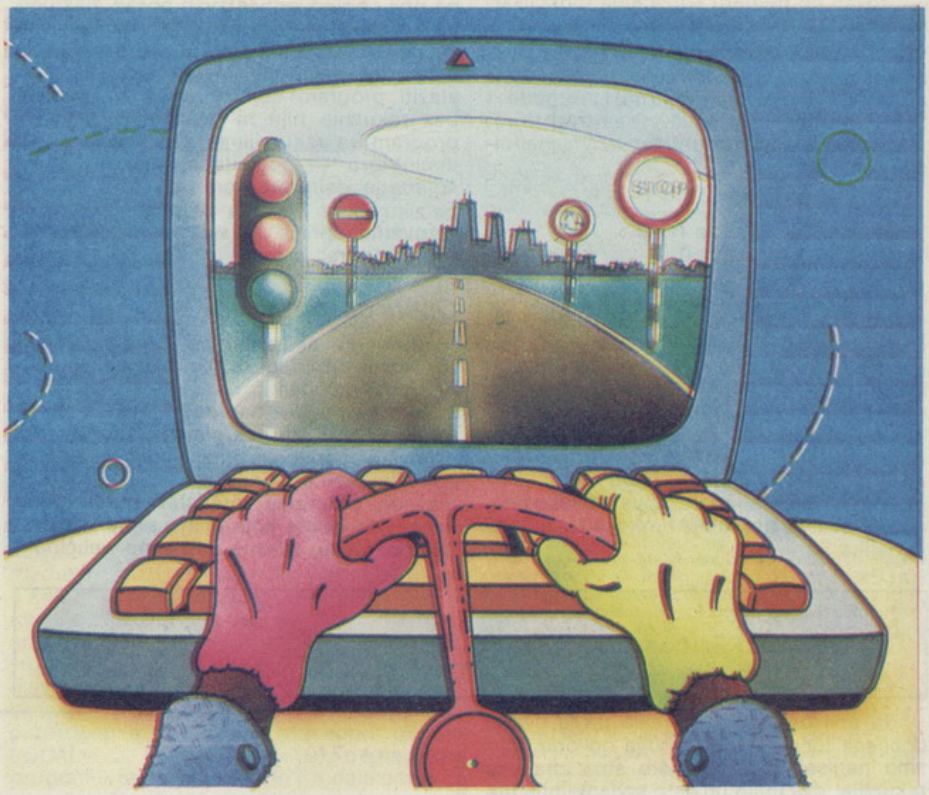
Mnogo više mogućnosti na svim računarima pružaju potprogramski segmenti, posebni delovi programa na koje se upravljanje prenosi naredbom GOSUB <broj prog. reda> (CALL kod „galaksije"). Postoji značajna razlika između naredbi GOTO i GOSUB. Realizacija GOTO naredbe podrazumeva da se u mesto gde se čuva adresa sledeće naredbe stavi adresa naredbe na koju ukazuje GOTO. GOSUB, pak, uz sve ono što proizvodi GOTO, još i čuva adresu naredbe koja je bila zapisana neposredno iza nje (upisuje je u stek). Ovo omogućava da se po izvršavanju potprogramskog segmenta „osveži sećanje" na mesto odakle je pozvan potprogram i vrati mu se upravljanje. Taj povratak u program ostvaruje se naredbom RETURN (to nije tipka RETURN kod „komodora") koja vraća upravljanje na poslednju adresu upisanu u stek.

Zahvaljujući steku, moguće je vršiti i pozivanje potprograma po dubini. To znači da glavni program zove potprogram, 1, potprogram 1 zove potprogram 2 i tako redom, a povratak se vrši obrnutim redom od reda pozivanja. Specijalan slučaj pozivanja potprograma po dubini je rekurzivno pozivanje potprograma — slučaj kada potprogram poziva samog sebe. Na „školskom" primeru računanja vrednosti funcije n! pokazaćemo kako izgleda i kako se realizuje rekurzivno pozivanje potprograma u bejziku koji (sem kod funkcijske naredbe za Acron) nije projektovan za rekurziju i na osnovu najveće vrednosti n za koje se može izvršiti izvesti zaključke o tome do koje dubine se mogu pozivati potprogrami na svakom od računara.

„Spektrum", Acorn, „komodor"

```
1 REM GLAVNI PROGRAM
10 I=O
20 N=I
30 F=1
40 PRINT N;"!=";
50 GOSUB 100
60 PRINT F
70 I=I+1
80 GOTO20
90 STOP
99 REM POTPROGRAM
100 IF N=O THEN RETURN
110 F=F*N
120 N=N-1
130 GOTO 100
```

(napomena: za „spektrum" treba kod naredbi dodele staviti službenu reč LET)



Startujte ovaj program i utvrdićete da se za svaki računar on završava porukom o prekoračenju posle izračunatog 33!.

Izmenite sada potprogram na sledeći način.

```
99 REM REKURZIVNI POTPROGRAM
100 IF N=O THEN RETURN
110 N=N-1
120 GOSUB 100
130 N=N+1
140 F=F*N
150 RETURN
```

„Galaksija"

```
1 ! GLAVNI PROGRAM
10 I=O
20 N=I
30 F=1
40 PRINT N;"!=";
50 CALL 100
60 PRINT F
70 I=I+1
80 GOTO 20
90 STOP
99 ! POTPROGRAM
100 IF N=0 RETURN
110 F=F*N
120 N=N-1
130 GOTO 100
```

Analizirajmo kako će se izvršavati ovaj potprogram kada se pozove za N=1. GOSUB u liniji 30 prenosi upravljanje na liniju 100, a adresa 40 ide na stek. Kako N nije O, ide se na liniju 110. Sada N postaje O i prelazi se na naredbu 120 u kojoj potprogram zove samog sebe. To znači da u stek iznad 40 upisuje adresu 130 i prelazi na liniju 100. Pošto je zadovoljen uslov, vrši se povratak na adresu koja se uzima iz steka, tj. 130. N se uvećava za jedan, F se množi sa N i stiže se do linije 150, koja opet vrši povratak u program, ali sada na adresu 40, jer je po uzimanju 130 ona ostala na vrhu steka.

Na ovaj način, dobićemo kao i u prethodnom slučaju tablicu vrednosti faktorijala, ali ne zaključno sa 33 (sem kod „spektruma" koji koristi mašinski stek). Program se na „glaksiji" završava već kod N=13, a na komodorú i Acornu imamo nešto veći stek predviđen za GOSUB pa dobijamo

➔

vrednosti do 23. Sada je razlog za prekid programa ? OUT OF MEMORY ERROR, što ne znači da nema uopšte slobodnog memorijalskog prostora, već da ga nema tamo gde je potreban — na steku.

Kule Hanoja

Ovaj primer nije izabran da demonstrira prednosti rekurzije (bila bi to antipropaganda), već da pokuša da objasni kako se u stvari realizuju instrukcije GOSUB i RETURN i odmeri koliko se duboko mogu pozivati potprogrami. Pravu rekurziju ćemo ilustrovati na primeru korišćenja funkcijske naredbe Acornovog bejzika.

Pošto smo smatrali da je funkcija n! svima poznata (proizvod broja n sa svim prirodnim brojevima koji mu prethode i 0!=1) nismo je navodili, ali za ovaj drugi primer navodimo prirodnu rekurzivnu definiciju funkcije n!.

N!=1 za N=0
N!=N*<N-1>! za N>0

Pokušajmo da dokažemo da je 3! isto što i 1*2*3. Po gornjoj definiciji je 3!=3*2!. Dalje je 2!=2*1!, a 1!=1*0!. 0! je 1 po definiciji, pa je onda 1!=1*1=1; onda je 2!=2*1!=2*1=2 a 3!=3*2!=3*2=6. Slično tome bismo mogli da pokažemo da je 5!=120 <pokušajte!> i da na taj način bar poverujemo u gornju definiciju faktorijala ako ne i da matematički dokažemo njenu ekvivalentnost sa ranije izloženom. Sastavimo, sada, novu verziju ranije funkcije za faktorijal (slika 5):

slika 5:

```
3000 DEFFNfaktorijel(N)
3010 IF N=0 THEN =1 ELSE =N*FNfakotirjel(N-1)
```

Na prvi pogled bi se reklo da je ovako dobijena funcija mnogo bolja od one koju smo napisali pre nego što smo znali za rekurzije: dve linije umesto sedam i nijedna pomoćna promenljiva (ranije smo koristili dve, S i I). Moramo, međutim, da kažemo da je ova funkcija veći potrošač memorije: za N=25 će rekurzivno biti pozvano 25 nivoa potprograma; ako se za pozivanje svakog nivoa troši po osam bajtova za beleženje adrese povratka i drugih informacija, plus još bar 6 bajtova za smeštanje novog fiktivnog argumenta (uvek se zove N), dobijamo da bi se primenom ove funkcije, uz tridesetak kilobajta RAM-a, moglo najviše izračunati 2000! (2000! se, jasno, ne može izračunati, jer je opseg brojeva sa kojima barata računar ograničen,dok je 2000! broj sa preko 5730 cifara), dok u prvom primeru, kada smo koristili petlju, takvog ograničenja nije bilo. Uopšte važi da su programi koji koriste rekurzije naoko kratki, elegantni i jednostavni, ali su veliki potrošači memorije i računarskog vremena, pa ih treba izbegavati kad god je to moguće (a moguće je uvek, uz više ili manje problema).

Završavajući ovu kratku priču o rekurzijama, ne možemo da ne navedemo jedan divan primer njihove upotrebe. Verovatno vam je poznat problem pod nazivom „Kule Hanoja": priča se da u gradu Benaresu postoji hram u kome je induski bog Brama pri stvaranju sveta postavio tri dijamantska štapića i na jedan od njih stavio 64 zlatna kotura različitih veličina: na zemlju je stavio najveći kotur, na njega manji i tako sve do najmanjeg kotura koji se nalazi na vrhu ovako dobijene kupe. Stotinu sveštenika toga hrama bez prestanka, danju i noću, prenosi koturove sa prvog na drugi stubac služeći se trećim kao pomoćnim i poštujući ograničenja koja je zadao Brama: sme se premeštati samo jedan po jedan prsten i nikada se veći prsten ne sme staviti preko manjeg. Kada sveštenici budu obavili ovaj na prvi pogled jednostavan posao, nastupiće kraj sveta (ne plašite se: jednostavan račun pokazuje da nam je do kraja sveta ostalo još nekih 500 milijardi godina). Sastaviti program koji rešava ovaj problem bez rekurzija nije ni malo lako; sastaviti program sa rekurzijama predstavlja pravu dečju igru — pogledajte sliku 6 i 7 na kojima je, osim programa pisanog na BBC bejziku, data i verzija na paskalu koju će verovatno isprobavati vlasnici „spektruma" i „komodora".

Opšti potprogrami

Konačno, zadržimo se na mogućnosti korišćenja procedura koju imaju jedino Acornovi računari.

Sve funkcijske naredbe i opšti potprogrami se stavljaju iza glavnog programa i, osim što moraju da imaju linijske brojeve koji su veći od linijskih brojeva naredbi glavnog programa, predstavljaju potpuno nezavisne celine koje imaju svoje ulazne i izlazne tačke, ulazne i izlazne veličine i

slika 6:

```
 10 REM
 20 REM
 30 REM          Kule Hanoja
 40 REM
 50 REM     Dejan Ristanovic 19
 60 REM
 70 REM
 80 REM          Racunari 7
 90 REM
100 REM
110 INPUT "Koliko diskova";N
120 CLS
130 PRINT "DISK","SA:","NA:"
140 PROCtowers(N,1,2,3)
150 END
200 DEFPROCtowers(N,X,Y,Z)
210 IF N=0 THEN ENDPROC
220 PROCtowers(N-1,X,Z,Y)
230 PRINT ;N,;X,;Y
240 PROCtowers(N-1,Z,Y,X)
250 ENDPROC
```

lokalne promenljive. Posmatrajmo naredbe sa slike 1:

slika 1:

```
1000 DEFPROCispis(A%,B%,X,Y)
1010 LOCAL I,J
1020 FOR I=1 TO A%
1030     FOR J=1 TO B%
1040         REM  radni deo petlje
1200     NEXT J
1210 NEXT I
1220 ENDPROC
```

slika 7:

```
ISO-Pascal compiler V. R1.00

  1 0 - program hanoj(input,output);
  2 0 - (*  rekurzivno resenje problema 'Kule Hanoja'
  3 0 C              Dejan Ristanovic 1985              *)
  4 0 -
  5 0 - const maxn=64;
  6 0 -
  7 0 - var n:integer;
  8 0 -
  9 0 - procedure prenos(N,X,Y,Z:integer);
 10 1 - begin
 11 1 -    if N<>0 then begin
 12 1 -                    prenos(N-1,X,Z,Y);
 13 1 -                    writeln('Sa',X:3,' na',Y:3);
 14 1 -                    prenos(N-1,Z,Y,X)
 15 1 -                  end
 16 1 - end;
 17 1 -
 18 0 - begin   (* glavni program *)
 19 0 -    write('?');readln(N);
 20 0 -    if n>maxn then writeln('Cekaj do kraja sveta!')
 21 0 -              else prenos(N,1,2,3)
 22 0 - end.
 23 0 -
 24 0 -
0 Compilation error(s)
Code size = 220 bytes
```

Ovim kratkim segmentom je opisana jedna procedura. Njeno ime je **ispis** i ima četiri takozvana fiktivna argumenta: A%, B%, X i Y, od kojih su prva dva celi, a druga dva racionalni brojevi. Zašto ove promenljive nazivamo fiktivnim argumentima? Da to razumemo, pogledaćemo sliku 2: primer segmenta koji poziva proceduru koju smo upravo definisali.

Primetili smo da je u poslednje vreme među našim čitaocima sve više vlasnika „amstrada". Stoga ćemo od sledećeg broja „Računara" u školu bejzika uključiti i bejzik „amstrada", dajući na početku pregled mogućnosti o kojima smo govorili u dosadašnjim brojevima.

slika 2:

```
100 A%=256
110 INPUT "Unesite granice";DONJA%,GORNJA%
120 PROCispis(DONJA%,GORNJA%,25,73)
130 PRINT A%
999 END
```

U prvoj liniji ovog programa promenljiva A% dobija vrednost 256, a zatim se sa tastature unose dva cela broja koja „pamte" promenljive DONJA% i GORNJA%. Sledi kritična linija: poziv procedure. Bejzik interpretator će najpre da pronađe proceduru i uporedi argumente:

DONJA%
GORNJA%
25
73
stvarni argumenti

A% B% X Y fiktivni argumenti

Vidimo da se fiktivnim promenljivima X i Y dodeljuju vrednosti 25 i 73 respektivno, dok se fiktivnim promenljivima A% i B% dodeljuju vrednosti stvarnih promenljivih DONJA% i GORNJA%. Obratimo, međutim, pažnju na stvarnu promenljivu A% kojoj je u prvoj liniji programa dodeljena vrednost 256. U potprogramu se, takođe, nalazi promenljiva A% kojoj je, vrlo verovatno, dodeljena neka vrednost različita od 256. Dok se potprogram izvršava, A% će imati tu vrednost. Kada se, međutim, naiđe na naredbu ENDPROC, kontrola će se vratiti glavnom programu, pri čemu će 256 ponovo postati aktualna vrednost promenljive A%, tako da će naredba 130 ispisati baš ovaj broj.

Dodajmo sada našem glavnom programu jednu liniju tako da glasi kao na slici 3:

slika 3:

```
100 A%=256
105 I=12:J=14
110 INPUT "Unesite granice";DONJA%,GORNJA%
120 PROCispis(DONJA%,GORNJA%,25,73)
130 PRINT A%,I,J
999 END
```

Startujmo ovako prepravljeni program i dobićemo, ako se izuzme ono što radi **ispis**, rezultate 256, 12 i 14. Očekivano? Na prvi pogled jeste, ali ne smemo da zaboravimo da se u proceduri **ispis** koriste promenljive I i J kao brojači u petlji, što znači da im je vrednost morala biti promenjena. Da to proverimo, ubacićemo u potprogram liniju:

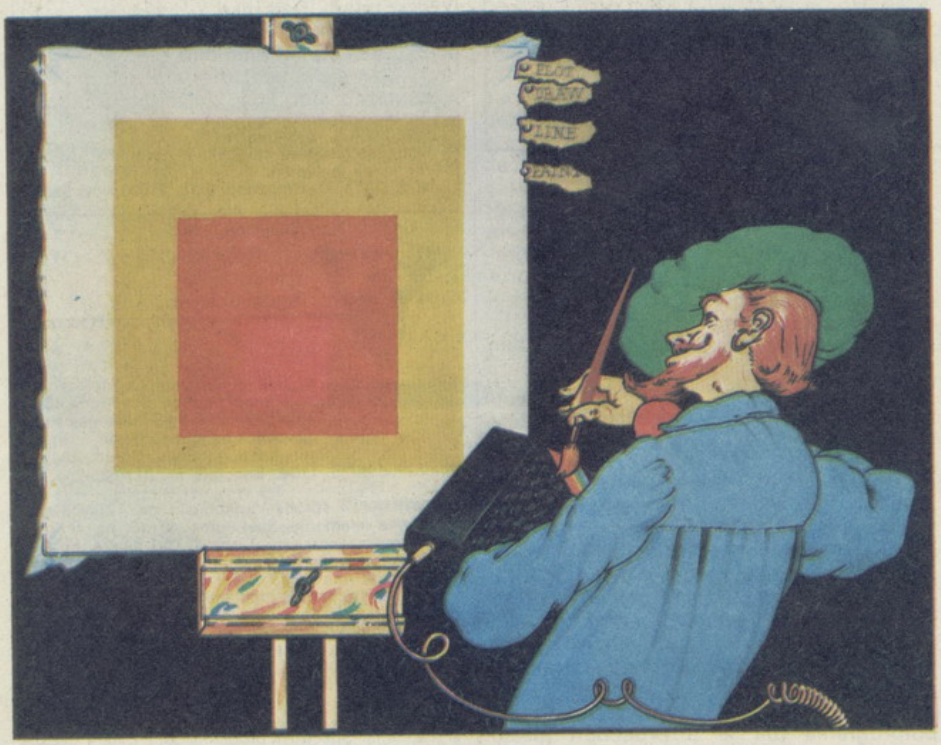
1215 PRINT I, J

i dobićemo vrednosti I i J u proceduri koje su, jasno, različite od 12 i 14. Obratimo, međutim, pažnju na naredbu **1010 LOCAL I, J** kojom su promenljive I i J deklarisane kao lokalne za tu proceduru. Slično formalnim argumentima, lokalne promenljive imaju vrednost samo u proceduri i po povratku u glavni program postaju nedefinisane ili dobijaju vrednosti koje su imale pre poziva potprograma (da je u proceduri pisalo LOCAL I, J, K a zatim K=100, u glavnom programu bi linija 140 PRINT K dala poruku o greški 'No such variable").

Suprotnost lokalnim promenljivima su takozvane globalne promenljive. Ako u glavnom programu dodamo liniju 5 M=123, a u listi fiktivnih argumenata i u listi lokalnih promenljivih ne pominjemo M, PRINT M će kako u programu tako i u potprogramu dati broj 123. Ukoliko, slično tome, u proceduri promenimo vrednost promenljive M, ta će promena biti u važnosti i posle njenog završetka. Standardni bejzici imaju isključivo globalne promenljive i jedini jezikom predviđen način za povezivanje potprograma je naredba GOSUB (CALL).

Dejan Ristanović
Nevenka Spalević

Grafika na računaru — kako to radi „amstrad"



Po svojim grafičkim mogućnostima (kao i zbog ostalih izvanrednih karakteristika) CPC 464 se među kućnim kompjuterima nalazi pri samom vrhu. Pošto tih kompjutera kod nas ima sve više logično ga je uporedivati s ostalim najpopularnijim računalima — „spektrumom", „commodorom 64 i BBC-em B.

Pri radu s „spectrumom" stoji na raspolaganju upotreba tekst formata od 32 znaka u 24 reda, te grafika visoke rezolucije od 256x192 točke. Na respolaganju ima 8 boja. Definiranje sprajtova nije sadržano u „spectrumovim" osnovnim mogućnostima, **ali se može naknadno programski ostvariti.**

„Komodor" 64 ima prikaz teksta od po 40 znakova u 25 redova, te grafiku visoke rezolucije 320x200 točaka. Moguće je koristiti 16 boja, te definirati 8 sprajtova. Mogućnosti računala C64 su bolje nego „spectrumove." Upotrebu grafike otežava korištenje instrukcija PEEK i POKE, odnosno potreba za direktnim pristupom određenim memorijskim lokacijama.

BBC B ima znatno bolje grafičke mogućnosti i od „spektruma" i od C64, a nešto malo bolje i od CPC-a 464. Maksimalni format teksta je 80 znakova u 32 reda. Ovisno o modovima, dolazi se do maksimalne grafičke rezolucije od 640x256 točaka, što je neznatno bolje od CPC-a, ali je izbor modova veći. BBC B ima svega 8 boja, a sprajtovi se moraju naknadno programski definirati. U biti, moguće je korištenje 16 „nijansi".

Amstrad/šnajder ima maksimalni format teksta od 80 znakova u 25 linija, tri moda grafike, te najvišu moguću rezoluciju od 640x200 točaka. Moguće je koristiti 16 od 27 boja a sprajtovi se moraju posebno definirati. Iz svega ovoga se vidi da uz odnos cijena/kvaliteta CPC 464 za one koji žele da crtaju predstavlja najbolji izbor.

Grafički modovi i upotreba boje

Kod većine broja računala izbor grafičkih modova ovisi o željenom broju boja koji se želi upotrijebiti za prikaz. Točnije, to znači da rezolucija slika ovisi o broju upotrebljenih boja — veća finoća povlači za sobom upotrebu manje boja i obratno. Za ove činjenice je CPC 464 dobar primjer.

Tri moda rada u kojima CPC 464 može raditi su označeni kao modovi 0,1 i 2.

Mod 0 daje najgrublju grafiku, ali se zato u tom modu može upotrijebiti 16 boja. Tekst je podijeljen na 20 znakova u 25 linija. Grafička rezolucija je 160x200 točaka (160 u liniji i 200 vertikalno). Taj mod se najčešće upotrebljava u igrama kod kojih se želi upotrijebiti što više boja, a kvaliteta prikaza nije od osobite važnosti.

Mod 1 je osnovni mod, a to znači da se računalo nalazi u njemu odmah po uključenju. Tekst je podijeljen na 40 znakova po retku, a grafička rezolucija je 320x200. Moguće je koristiti samo 4 boje odjednom. Taj mod je vrlo dobar za normalno programiranje, za poslovnu i zananstvenu grafiku, gdje je potrebno prikazati više detalja na uštrb korištenih boja.

U modu 2 tekst se prikazuje sa 80 znakova po liniji. Grafička rezolucija je najveća u ovom modu (640 – 200 točaka). Mogu se koristiti samo 2 boje. Taj mod se upotrebljava tamo gdje je potrebna vrlo fina grafika, te za obradu teksta i tabela (word-processing i spreadsheets). Ponekad se javljaju problemi pri prikazu 80 znakova u liniji zbog nestabilne slike na monitoru.

CPC 464 može „proizvesti" 27 različitih boja. Dakle, kao da imamo 27 patrona s raznobojnom tintom, odnosno 2 pera (PEN) za crtanje u izuzetno viskojoj grafici, 4 za normalne crteže i 16 za grube crteže. Može se, dakle, koristiti boja tinte (INK) prema želji, samo mora biti različita od boje pozadine.

PAPER, INK i PEN su instrukcije sadržane u CPC-ovom Locomotive bejziku. INK predstavlja jednu od 27 upotrebljivih boja, PEN je odabriranje željenog pera za crtanje, a PEPER-om se odabire boja pozadine. U praksi, sva „pera" imaju različite vrijednosti, pa često nije potrebno koristiti instrukciju INK, odnosno moguće je koristiti boju naznačenu od kompjutera.

Ako se, na primer, utipka PEN 2 i pritisne ENTER — poruka READY će biti obojena u svijetlo plavo.

„Pera" moda 1 jesu: plava (PEN 0), žuta (PEN 1), svijetlo plava (PEN 2) i crvena (PEN 3). PEN 0 se koristi za pozadinu (PAPER), a PEN 1 za bojenje prednjeg plana. Kada se utipka PAPER 1, tekst će biti u crvenoj boji na žutoj pozadini. Jasno je da u slučaju iste boje teksta i pozadine tekst postaje nevidljiv.

Uz instrukciju INK dolaze dvije vrijednosti npr.: INK 1,2. Prvi broj uz instrukciju je broj „pera" koje želimo koristiti, a drugi broj je oznaka upotrebljene „tinte", odnosno boje (0—26).

Moguće je označiti svih 16 „pera", ali treba znati da broj pera ovisi o modu u kojem radimo. Tako za mod 0 imamo pera 0—16, za mod 1 pera 0—3 i u modu 2 pera 0 i 1. Odabrano pero nije ovisno o promjeni moda, tj. ako odaberemo pero 3 u modu 1 (crveno), te potom mod promijenimo u mod 0, tekst će ostati u crvenoj boji.

U slučaju da se želi odabrati pero većeg broja nego što je moguće u modu u kojem se trenutačno radi promijeniti će se vrijednost, npr. pero 4 u modu 1 će dati pero 0 u višem modu, odnosno pero (pen) 5 će biti pero 1. Ako se odabere pero 6 u modu 1, dobiti će se pero 2 u višem modu.

Kućni kompjuteri sve više privlače korisnike i svojim grafičkim mogućnostima. Uzevši kao primjer više nego praktičan kompjuter amstrad/šnajder CPC 464, može se tvrditi da kompjuterska grafika ne mora biti samo u funkciji igara i ilustracija proračuna, već i solidan poligon za kreativno stvaralaštvo.

Računar izvanrednih grafičkih potencijala: Amstrad/šnajder CPC 464

Demonstracija boja

Program „Demonstracija boja" prikazuje kolor mogućnosti moda 0. Prva petlja, linije 35 do 65 boje linije teksta u svaku od 16 mogućih boja. Linija pen 0 se ne može uočiti jer je iste boje kao i pozadina. Drugi dio programa, linije 75 do 125, čine petlju FOR . . . NEXT koja omogućuje da svih 16 pera kruži kroz sve moguće boje. Ostatak programa resetira četiri pera koristeći mod 1.

```
  5 REM demonstracija boja
 15 MODE 0
 25 CLS
 35 FOR boja=0 to 15
 45 pen boja
 55 PRINT „To je pen"; boja
 65 NEXT boja
 75 FOR a=0 TO 15
 85 FOR boja=0 TO 26
 95 INK a, boja
105 FOR k=1 TO 200 : NEXT k
115 NEXT boja
125 NEXT a
135 MODE 1
145 INK 0,1
155 INK 1,24
165 INK 2,2
175 INK 3,3
185 PEN 1
195 END
```

Program se može modificirati tako da resetira svih 16 pera (penova) koristeći FOR . . . NEXT petlju te instrukciju DATA. Isto tako, mogu se linije 125 do 195 zamijeniti sa CALL &BBFF, tj. subrutinom operativnog sistema koji resetira boje ekrana i modove, ali ne i postojeći odabir pera (penova).

Instrukcija INK može, također, postaviti pero tako da se izvodi „flešing" efekt, odnosno bljeskanje. To se postiže određivanjem dvije boje (dva INK-a) koji će se alternativno koristiti. Fleširanje je najjednostavnije postići postavljenjem boje INK-a (tinte) na neku različitu pozadinu, tako da se u jednom momentu fleširajući objekt razlikuje od pozadine, a u drugom momentu stapa s pozadinom.

Brzina fleširanja — bljeskanja se određuje instrukcijom SPEED INK. Uz instrukciju dolaze dva broja koji moraju imati cjelobrojne vrijednosti. Prvi broj određuje vrijeme trajanja prve boje, a drugi vrijeme trajanja druge boje. Dakle, pomoću ove instrukcije moguće je postići točno željeno vrijeme izmjene fleširajućih boja.

Crtanje linija

CPC 464 raspolaže setom instrukcija za crtanje linija i plotiranje točaka na ekranu. Pri tome se koristi sistem kordinata za određivanje vertikalnih i horizontalnih pozicija.

Grafička rezolucija je u sva tri moda različita, ali koordinatne vrijednosti ostaju iste. To znači da se kroz dani set DRAW instrukcija može nacrtati lik na „istom mjestu" u sva tri moda, što predstavlja vrlo korisnu mogućnost. U stvari, sva tri moda imaju istu vertikalnu rezoluciju — 200 linija. Koordinatni sistem koristi vertikalnu skalu do vrijednosti 400.

To znači da koordinate 0 i 1 odgovaraju jednoj stvarnoj točki, 2 i 3 drugoj itd. Horizontalne koordinate idu do vrijednosti 640. To odgovara najvećoj rezoluciji, tj. modu 2. Mod 1 ima 320 točaka u liniji, dakle koordinate 0 i 1 su u stvari predstavljene kao jedna stvarna točka, 2 i 3 kao druga itd., kao u slučaju kod vertikalnih koordinata. Kod moda 0, koordinate 0,1,2 i 3 predstavljaju jednu točku; 4,5,6 i 7 drugu itd.

Grafičke koordinate imaju vrijednosti od 0 do 639 horizontalno, te 0 do 399 vertikalno. Početna lokacija (koordinata 0,0) koja se nalazi u donjem lijevom uglu naziva se „origin". To je različito od pojma „text-origin" koji se nalazi u gornjem lijevom uglu. Posebna pogodnost je ta što se „origin" može premještati bilo kuda po ekranu koristeći instrukciju ORIGIN.

Kompjuterom CPC 464 se može crtati i „izvan" ekrana. Takozvane „off-screen" pozicije će biti ispravno zabilježene, što u praksi znači da će ako crtamo „izvan" ekrana, pa se opet vraćamo u njegove okvire, sve biti na svome mjestu, odnosno linije iscrtane pod ispravnim kutevima.

Kao što postoji kursor pri tekstualnom prikazu, tako postoji i „nevidljivi" grafički kursor. Tako kada se npr. iscrtava linija, ona se crta od početne pozicije kursora prema novo određenoj poziciji.

Postoje dva načina određivanja koordinata — relativni i apsolutni. Apsolutno određivanje podrazumijeva, kako to i naziv govori, apsolutno određivanje na zamišljenoj koordinatnoj mreži ekrana, a može se shvatiti i kao relativno prema „oridžinu".

Relativno određivanje predstavlja određivanje kao relativno pomicanje pozicije kursora. Koordinate s negativnim vrijednostima se mogu koristiti pri oba načina određivanja. Karakteristika apsolutnog određivanja je jednostavnost, pa se i više koristi, a relativnog — da u praksi pojednostavljuje mnoge crteže.

Instrukcije DRAW,PLOT i MOVE pred-

→

stavljaju glavne naredbe amstrad/šnajderove grafike. U ovim oblicima se koriste pri apsolutnom određivanju. Za relativno određivanje imaju oblike DRAWR, PLOTR i MOVER.

DRAW crta linije od pozicije grafičkog kursora prema definiranoj novoj poziciji. MOVE pomiče grafički kursor bez posebnog označavanja na ekranu. PLOT je slična instrukcija kao MOVE, ali crta točke ne novoj poziciji kursora.

Sljedeći primjer ilustrira primjenu instrukcija MOVE i DRAW, te relativne i apsolutne koordinate. Program crta kvadrat na ekranu, čija se veličina i pozicija unosi linijama 55 i 65. Dio programa za crtanje kvadrata je pisan kao sabrutina koju čitaoci mogu ugraditi u vlastite programe. Linija 115 pokreće grafički kursor na „apsolutnu" poziciju u liniji 45. Slijedeće četiri linije crtaju kvadrat koristeći „relativne" koordinate. Redosljed iscrtavanja je: lijeva strana, vrh kvadrata, desna strana, dno kvadrata. Pažnju treba obratiti na činjenicu da crtanje gore i udesno ima pozitivne vrijednosti, a dolje i udesno negativne vrijednosti.

```
 15 REM crtanje kvadrata
 25 MODE 1
 35 WHILE 1
 45 INPUT „Pozicija donjeg ugla
    (x,y)";x,y
 55 INPUT „Visina ";v
 65 INPUT „Sirina ";s
 75 CLS
 85 GOSUB 105
 95 WEND
105 REM subrutina za crtanje kvadrata
115 MOVE x,y
125 DRAWR 0,v
135 DRAWR s,0
145 DRAWR 0,-v
155 DRAWR -s,0
165 RETURN
```

Pri crtanju često treba izvesti kružnice. Za razliku od npr. „spectruma", amstrad-/šnajderov Locomotive bejzik nema mogućnost korištenja instrukcije CIRCLE. Međutim, moguće je lako crtati kružnice koristeći funkcije sinus i kosinus (SIN i COS).

Primjer za takvo crtanje se može vidjeti u programu:

```
 15 MODE 1
 25 BORDER 2
 35 REM crtanje kvadrata
 45 DRAW 0,399,1
 55 DRAW 639,399,1
 65 DRAW 639,0,1
 75 DRAW 0,0,1
 85 REM crtanje kružnice
 95 a=320 : b=200 : c=196
105 MOVE a+c,b
115 FOR d=0 TO 2*PI STEP 0.03
125 DRAW a+c*COS(d),b+c*SIN(d),2
135 NEXT d
145 REM crtanje elipse
155 c=92
165 FOR m=0 TO PI STEP PI/5
175 FOR n=0 TO 2*PI STEP 0.03
185 p=100*COS(n)
195 ax=a+p*SIN(m)+c*SIN(n+m)
205 by=b+b*COS(m)+c*COS(n+m)
215 IF n=0 THEN MOVE ax,by
225 DRAW ax,by,3
235 NEXT n
245 NEXT m
255 END
```

Program crta kružnicu koristeći elepsu kao šablonu. Linija 25 postavlja boju okvira ekrana, a moguće je, prema želji, izvesti i fleširanje. Broj uz instrukciju BORDER — (2) dolazi iz palete od 27 boja. Linije 45 do 75 crtaju kvadrat oko ekrana. Pažnju treba obratiti na poseban parametar uz DRAW instrukciju. Naime, boja linije nije kontrolirana definiranjem „pen-a", već parametrom uz DRAW koji mora biti broj „pen-a". Ako boja linije nije označena uz DRAW, program koristi posljednje definiranu boju. Linije 95 do 135 crtaju kružnicu. Varijable a i b su koordinate središta kružnice, a varijable c je radijus. Linije 155 do 245 crtaju elipsu koja se koristi kao šablona.

Upotreba instrukcije PLOT je uobičajena kao i kod drugih računala. Jednostavno rutinom za plotiranje točaka na slučajno određenim mjestima i s slučajno određenim bojama može se prikazati primjer za upotrebu instrukcije PLOT:

```
 5 MODE 0
15 WHILE 1
25 PLOT RND(1)*640,RND(1)*400,RND(1)*
   15
35 WEND
```

Grafički efekti

Jednostavne, a efektne crteže je moguće dobiti upotrebom linija i spirala prema sljedećem primjeru:

```
  5 MODE 0
 15 BORDER 2
 25 PAPER 5
 35 CLS
 45 a=1 : b=350
 55 WHILE B>34
 65 FOR m=0 TO 2*PI STEP 0.1
 75 IF a=5 THEN a=6
 85 IF a=14 THEN a=1
 95 p=b-50
105 IF p<0 THEN p=0
115 MOVE 320+b* COS(m), 200+b*
    SIN(m)
125 DRAW 320+p* COS(m-0.5),
    200+p*SIN(m-0.5),a
135 b=b-1
145 IF p<0 THEN p=0
155 a=a+1
165 NEXT m
175 WEND
185 GOTO 185
```

Linije 135 i 115, točnije element b, u njima proizvodi spiralu. Druga spirala se tvori linijama 125 i 95. Linije se iscrtavaju točkom prve spirale prema točkama druge spirale. Varijabla a određuje boju. Izmjenom varijabli mogu se dobiti razni drugačiji grafički efekti.

Prozori (Windows)

Kod računala CPC 464 je moguće ekranski prikaz podijeliti na osam „prozora", u kojima se instrukcije LIST i PRINT mogu normalno koristiti. Definiranje prozora je vezano uz instrukciju WINDOW kojoj se pridružuje 5 parametara. Prvi parametar je kanal ili broj toka podataka, a ima značenje indentifikacije odgovarajućeg prozora koji može biti odabran PRINT instrukcijom. Ostala 4 parametra određuju granice prozora koristeći uobičajeni koordinatni sistem računala CPC 464. Prva dva od četiri parametra određuju širinu prozora (lijevu i desnu) granicu, a druga dva parametra određuju visinu odnosno donju i gornju granicu.

Primjer:

```
WINDOW  1,15,25,20,30
```

Dakle, 1 je broj kanala, širinske koordinate su 15 i 25, a visinske su 20 i 30.

Točan prostor ekrana koji se zauzima ovisi o modu koji se upotrebljava, odnosno numeriranje koordinata je različito za svaki mod. Uz instrukciju WINDOW postoji instrukcija WINDOW SWAP koja se koristi za mijenjanje broja kanala između dva prozora, npr.:

```
WINDOW SWAP 2,3
```

znači da će se podaci kanala 2 prenijeti na kanal 3 i obrnuto. Ta instrukcija se može koristiti za usmjeravanje podataka s „glavnog" ekrana (kanal 07) prema prozoru, npr.:

```
WINDOW SWAP 0,3
```

znači da se podaci s ekrana šalju na kanal 3, odnosno na prozor što se u programima može iskoristiti na razne načine.

JEDNOSTAVNA ANIMACIJA

Amstrad/šnajderom se može izvoditi vrlo kompleksna animacija, ali svakako treba prvo shvatiti bit te animacije na jednostavnim primjerima.

Princip animacije je jednostavan — štampa se neki znak (karakter) na ekranu, a potom se isti taj znak štampa na drugom mjestu, s time da se prethodni briše. Također je vrlo lako animirati objekt sastavljen od više znakova u vodoravnoj liniji.

Primjer:

```
 15 REM pokretanje
 25 CLS
 35 pokret$=CHR$(32)+CHR$(205)
    +"__"+CHR$(204)+CHR$(32)
 45 a=17 : b=24
 55 WHILE 1
 65 IF NOT INKEY(71) THEN a=a-1
 75 IF NOT INKEY(22) THEN a=a+1
 85 IF a<1 THEN a=1
 95 IF a>34 THEN a=34
105 CALL &BD19
115 LOCATE a,b : PRINT pokret$
125 WEND
```

Pokretanje objekta po ekranu je moguće tipkom „z" ulijevo, te tipkom „/" udesno. Te tipke su određene programskim linijama 65 i 75. Linije 85 i 95 spriječavaju bježanje objekta koji se pomiče izvan ekrana. Linija 105 predstavlja subrutinu operativnog sistema koja određuje čekanje stroja na ispis nove video-slike. Konkretno, time je izbjegnuta „dupla" slika objekta pri kretanju.

Vertikalno pomicanje se može prikazati sljedećim primjerom:

```
 5 REM vertikalno pokrtanje
15 CLS
25 RANDOMIZE TIME
35 a=1 : b=1
45 LET C=INT(RND*30+5)
55 FOR d=1 TO 23
65 LOCATE c, d : PRINT CHR$(228)
75 LOCATE a,b : PRINT CHR$(32)
85 a=c : b=d
95 NEXT d
99 GOTO 45
```

Programska linija 25 postavlja generator slučajnih brojeva. Linija 35 postavlja inicijalne vrijednosti dvije varijable koje se koriste za pamćenje posljednje pozicije. Linija 45 postavlja horizontalnu poziciju objekta. Petljom FOR . . . NEXT kontrolira se kretanje. Kontrolna varijabla se koristi kao vertikalna koordinata. Linija 65 prikazuje objekt, a 75 ga briše s prethodne pozicije.

Ovime su prikazane neke najjednostavnije osnove grafike na računalu CPC 464. Potpuno savladavanje grafičkih mogućnosti zahtijeva mnogo vježbe, ali trud će se isplatiti, jer ovo računalo predstavlja vrlo dobar „kist" za crtanje.

Zvonimir Vistrička, dipl. ing.

Programiranje na mašincu | kodovi iz rukava

Spektrum

Postoje mnoge knjige koje se bave Z80 procesorom. Postoji, na žalost, i mnogo onih kojima je strana literatura iz ovog ili onog razloga teško dostupna. Najzad, postoje mnoge male tajne procesora Z80 koje se otkrivaju dugogodišnjim radom, a koje ni u jednoj knjizi nisu opisane. Ovaj članak je, pre svega, namenjen onima koji su, uz teške muke, već savladali osnove mašinskog programiranja, i koji su sada gladni onih malih sitnih trikova ima ih bar milion — koji omogućavaju pisanje kompaktnog i efektnog koda.

Kako obrisati akumulator . . .

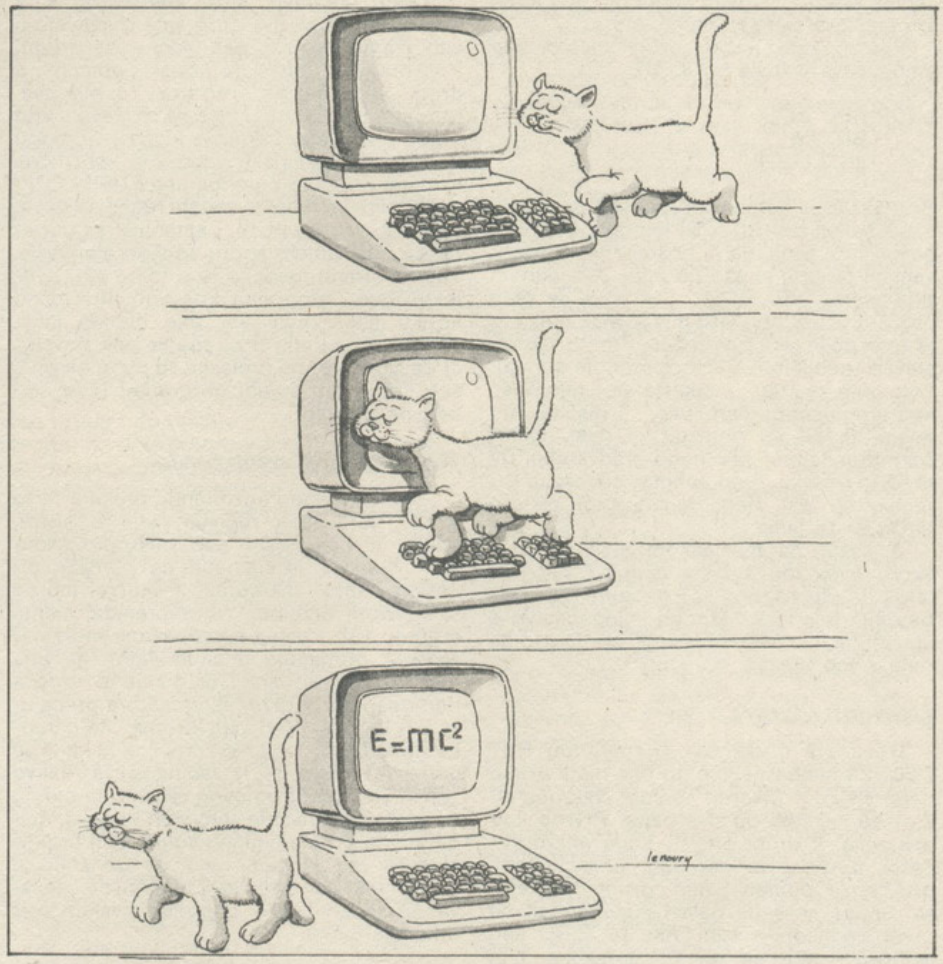
Krenimo od naredbe XOR. Radi se o ekskluzivnom OR-u, što će reći da 0 XOR 0 daje nulu, baš kao i 1 HOR 1, dok dve preostale kombinacije (0 XOR 1 i 1 XOR 0) daju jedinicu. Dakle, prosto rečeno, dva ista daju nulu, a dva različita jedan. Sama naredba XOR s znači izvrši ekskluzivno OR nad akumulatorom i operandom s (neki registar, brojni podatak, (HL), (IX+d) ili (IY+d)), a rezultat smesti u akumulator. Pri tome se prvo uzme bit 0 akumulatora i bit 0 operanda, izvrši XOR i rezultat stavi u bit 0 akumulatora. Zatim se prelazi na bit 1, 2, 3, i tako redom sve do sedmog. Na kraju se CARY fleg (C) postavi na nulu, a ZERO fleg (Z) na jedinicu ako je novi sadržaj akumulatora jednak nuli, odnosno na nulu ako nije.

Lepo, a sada XOR A. Prvo, moramo da primetimo da se u ovom slučaju XOR operacija vrši sa dva potpuno ista operanda — sa akumulatorom i akumulatorom. Zamislimo, za trenutak, da je bit 0 akumulatora jednak nuli. 0 XOR 0 daje nulu. Neka je sada bit 0 jednak jedinici. 1 XOR 1 takođe daje nulu. Dakle, ma kakav bio sadržaj akumulatora, XOR A garantovano daje nulu.

Zar nismo mogli da kažemo LD A, 0? Pa, mogli smo. Ali, LD A, 0 zahteva dva bajta i izvršava se za sedam otkucaja kloka. XOR A zahteva samo jedan bajt i izvršava se za četiri otkucaja. Zašto razbacivati vreme i memoriju? Uzgred rečeno, neki autori više vole da koriste SUB A, sa potpuno istim efektom — SUB A znači od akumulatora oduzmi akumulator i rezultat smesti u akumulator.

. . . a kako C fleg

Naredba AND A ima potpuno istu namenu: štedi vreme i memoriju. Potpuno je identična naredbi XOR, s tim što se vrši logička operacija „i" — 1 AND 1 daje jedan. Sve ostale kombinacije daju nulu. Ako sada uzmemo da izvršimo AND operaciju sa dva potpuno ista operanda, rezultat, u našem slučaju akumulator, ostaće potpuno nepromenjen. AND A nema nikakvog dejstva. U stvari, ipak ga ima. ZERO fleg će biti setovan ako se u akumulatoru nalazi nula. Umesto da sadržaj akumulatora ispitujemo sa CP 0, možemo da kažemo AND A, i uštedimo jedan bajt i tri otkucaja kloka.



To nije sve. AND A postavlja CARY fleg na nulu. Postoji, naime, naredba koja setuje CARY fleg, ali ne i ona koja ga resetuje. To se, istina, može postići sa SCF CCF, ali XOR A deluje mnogo elegantnije i brže.

Da li je u registru nula

Što se ostalih osmobitnih registara tiče, da bi proverili da li je neki od njih na nuli, možemo da upotrebimo sledeću caku; INC s DEC s. Prvo se registar poveća za jedan, a zatim smanji za jedan. Pri tome će, naravno, zero fleg biti setovan, ako se u registru nalazi nula. Prosto, zar ne?

Digitalni ringišpil

A sada, ciklusi. Suprotno vrlo rasprostranjenom mišljenju, sa DJNZ naredbom se mogu realizovati ciklusi i do 256, a ne samo do 255. Tajna je u tome što se nula može interpretirati i kao nula, i kao 256. Ako je potreban ciklus koji ide preko 256, onda DJNZ više ne pali, pa se treba setiti nečeg boljeg. Prvo, pada na pamet da se dva ciklusa ostvarena pomoću DJNZ upakuju jedan unutar drugog. Lepo, ali može i ovako:

```
DEC BC
LD A,B
OR C
JR NZ, LOOP
```


➜

BC registarski par služi kao brojač. Prvo se BC smanji za jedan. Na izuzetno veliku žalost, naredba DEC BC neće delovati na ZERO fleg, pa treba nekako ispitati da li je BC stigao do nule. Prvo se sadržaj B registra prebacuje u akumulator (LD A,B). Sada treba proveriti da li su i A i C na nuli. Zašto da ih ne saberemo sa ADD A,C, pa ako rezultat nije nula, onda treba ponoviti ciklus! To je u redu, ali se, na žalost ne može primeniti prosto sabiranje. Šta ako je, na primer, A=130, a C=126? Dobije se 256, a i to je nula!

Primenićemo logičku operaciju OR nad akumulatorom i registrom C (OR C). Pri tome, dve nule daju nulu, a ostale tri moguće kombinacije jedinicu. Dakle, ako je bilo koji bit akumulatora ili registra C jednak jedinici, rezultat sigurno neće biti nula. Provera je gotova, ciklus radi. Primetimo da će se stvar vrteti 65536 puta ako je na početku ciklusa u BC stavimo nulu. Jedina nevolja sa ovim metodom je u tome što uništava sadržaj akumulatora.

Postoji zato jedan drugi metod koji omogućava cikluse do 32767:

```
LOOP DEC BC
     BIT 7,B
     JR Z,LOOP
```

Broj 32736 je petnaestobitni, što će reći da je šesnaesti bit (BIT 7,B) jednak nuli. Pretpostavimo, sada, da je postepenim smanjivanjem taj broj stigao do nule. Još jednom primenimo DEC BC i... iz nule se rađa 65535. Sesnesti bit više nije jednak nuli, i to je znak da je ciklus okončan. Prosto, jednostavno, genijalno! Samo, primetite sitnicu: kriterijum za izlaz iz ciklusa više nije nula, kao u ranijem primeru, već −1 (65535), što znači da će se ringišpil... ovaj, ciklus obrnuti jedanput više nego predviđeno. Da bi se to izbeglo, pred početak ciklusa, u BC registarski par treba staviti broj ciklusa umanjen za jedan.

A zašto na mašinskom jeziku ciklusi skoro uvek idu s brda nadole? Zar nije lakše da idu nagore, sa korakom 1, kao na bejziku? Nije lakše. Mnogo je jednostavnije ispitati da li je neki registarski par stigao do nule, nego recimo, do 3160.

Uslovni restart

Da li ste ikada čuli za naredbu RST Z,38? Za neobaveštene, to nije ništa drugo nego JR Z,-1. Stvarno suludo! Naredba JR Z,−1 se sastoji od dva bajta. Prvi je kod operacije, a drugi ono−1. Kada mikroprocesor izvršava tu naredbu, prvo „učita" prvi bajt, a odmah zatim i drugi. Programski brojač se sada nalazi pozicioniran na prvoj sledećoj naredbi. Ako je ZERO fleg setovan, onda se od PC oduzima 1, i on pokazuje bas na ono−1 — što će reći 255. Program se odatle nastavlja, a 255 je kôd za... RST 38!

Alternativni set

Alternativni set registra je jedna vrlo specifična tvorevina. U dobrim rukama, stvara čuda i strahovito ubrzava program. Krenimo redom. Pored AF registarskog para postoji još jedan, alternativni, AF registar obeležen sa AF'. Mikroprocesor može u JEDNOM trenutku da koristi SAMO JEDAN AF registar. Koji će to biti, određuje jedan FLIP-FLOP. Pomoću naredbe EX AF, AF', može se promeniti stanje tog FLIP-FLOP-a. Jedna EX AF, AF' naredba, i mikroprocesor radi sa alternativnim akumulatorom. Tada se sve naredbe tipa LD A, s, AND s, OR s, ADD A, s, itd, odnose na alternativni registar AF'. Još jedna EX AF, AF' naredba, i sve se vraća na normalno. Baš kao što postoji alternativni registar AF', tako postoje i alternativni registri BC', DE', i HL'. Naredba EXX naređuje procesoru da „zaboravi" na BC, DE i HL registre i „uključi" BC', DE' i HL'. Naravno, još jedna EXX naredba, i sve se vraća na staro stanje. Prilikom izvođenja naredbi EX AF, AF' i EXX ne gubi se nijedan podatak ni u „običnim" registrima, ni u „alternativnim". Nije na odmet spomenuti broj 2758 (heksadekadno) koji mora da se nalazi u HL' registru pri povratku u bejzik. U protivnom, sledi krah sistema (ovo važi samo za spectrum).

Čemu, zapravo, služi alternativni set? Zamislimo da se dva programa izvršavaju u isto vreme (to jest, čas jedan, čas drugi). Prvi program može da koristi „obične", a drugi „alternativne" registre. To nije sve. Naredbe EX AF, AF' i EXX se izvršavaju vrlo brzo — za četiri otkucaja kloka — i mogu da posluže kao neka vrsta vrlo specifične zamene za spore naredbe tipa PUSH i POP. Jednostavno, umesto sedam registara opšte namene, imamo ih 14, i smanjuje se potreba za pozivanjem spore spoljne memorije. Tamna strana medalje je u tome što „projektovanje" programa koji vrlo intenzivno koristi alternativni set nije nimalo lako. Programer se vrlo brzo spetlja pokušavajući da prati stotine prelaska sa ovog na onaj set. Treba biti suludi programer (čitaj haker) pa istrajati.

Registar za osveženje

Evo još jednog egzotičnog registra — to je R (REFRESH) registar, koji u stvari, spada u hardverski deo mikroprocesora Z80. Njegova je namena da pomaže pri „osvežavanju" memorije, a sadržaj mu se vrtoglavom brzinom, i bez prekida, menja od 0 do 127. Postoji naredba koja sadržaj R registra prebacuje u akumulator, ali ona nije naročito korisna. Još manje je korisna naredba koja sadržaj akumulatora prebacuje u R registar. Kod „spektruma", ta naredba ne može da izazove krah, čak ni kada se ponavlja u ciklusu (?!). Jedina kakva—takva softverska korist od ovog registra bi moglo da bude generisanje slučajnih brojeva, mada je vrlo problematično koliko bi ti brojevi stvarno bili slučajni ako se generišu u nekom ciklusu. Sadržaj R registra se uvećava za jedan posle odprilike svakih šest otkucaja kloka.

Kako testirati H fleg

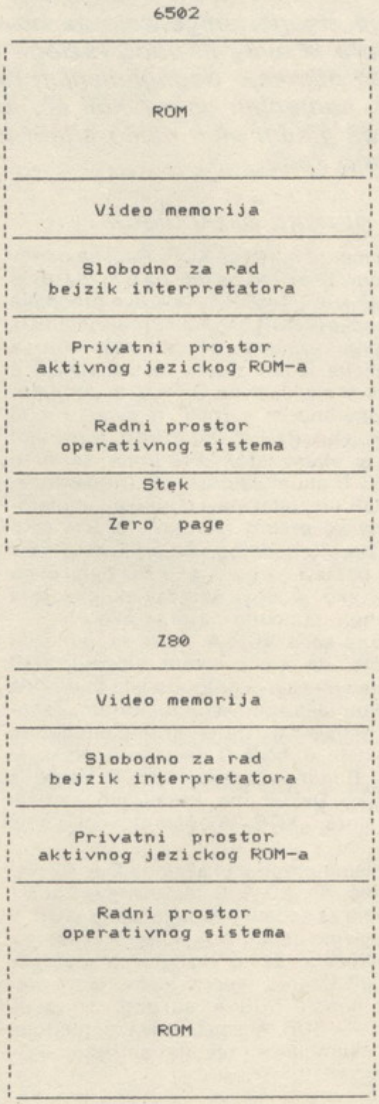
I, nešto ne previše mračno za kraj. Pored toga što postoji CARY fleg, koji reaguje kada rezultat „pretekne" iz sedmog bita, tako postoji i HALF CARY fleg (H) koji reaguje na prenos između četvrtog i petog bita u akumulatoru. Na žalost, ne postoji naredba koja će ispitati H fleg. Kako to izvesti?

```
PUSH AF
POP BC
BIT 4,C
```

Ko razume, shvatiće.

Vladimir Kostić

lične



Moguce memorijske mape računara sa Z80 i 6502 su prikazane na slici 1; vidimo da je jedina bitna razlika u tome što Z80 očekuje da ROM zauzima memorijske lokacije počevši od adrese 0, a 6502 za ROM rezerviše visoke adrese, u našem slučaju počevši od 8000. Na početku RAM-a je radni prostor operativnog sistema i trenutno aktivnog jezičkog ROM-a (deo memorije koji ćemo zvati „privatni radni prostor bejzika" smo, dakle, već odvojili, jer je bejzik, kao što smo videli, često samo jedan od raspoloživih ROM-ova). Uveli smo promenljivu LOMEM koja sadrži prvu slobodnu memorijsku adresu iznad radnog prostora operativnog sistema. Od kraja RAM-a smo smestili video memoriju koja u principu može da zauzima manje ili više RAM-a, zavisno od trenutno izabranog grafičkog

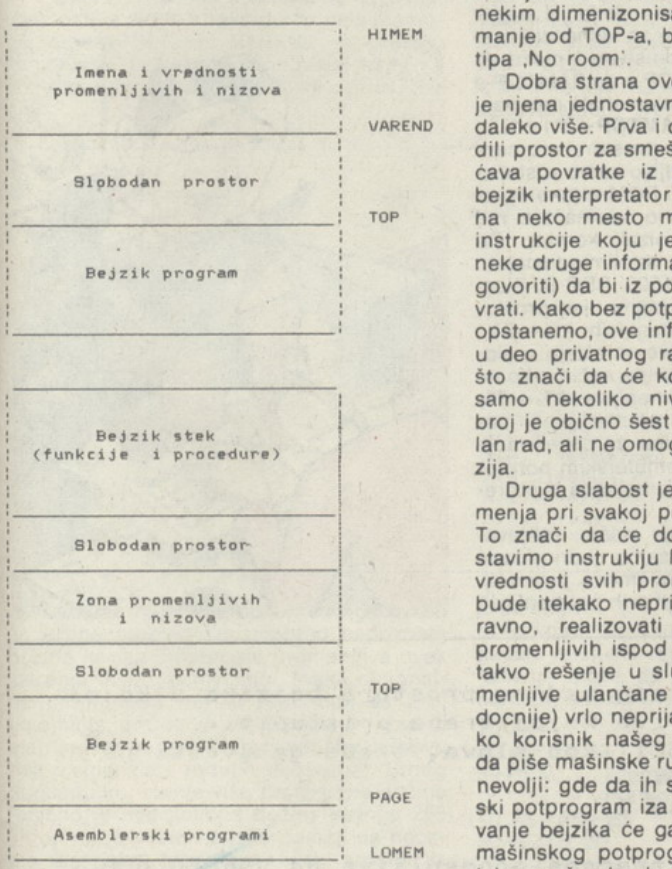
put u središte „ROM"-a/5/

stvari bejzika

Za svoj normalan rad, interpretator bejzika, ili bilo kog drugog programskog jezika, mora da odvoji određeni deo radne memorije u koji će biti smeštana imena i vrednosti promenljivih i nizova, stek za pozivanje potprograma i, naravno, sam program koji se izvršava. Kako organizovati taj prostor da bi se obezbedio minimalan utrošak RAM-a, najjednostavnija konstrukcija samog interpretatora ili, kao poseban šlager, najveća brzina izvršavanja programa.

moda; potrebna nam je, dakle, promenljiva HIMEM koja čuva adresu prvog bajta video memorije. Sav prostor između LOMEM i HIMEM je slobodan za rad bejzika.

slika 2:



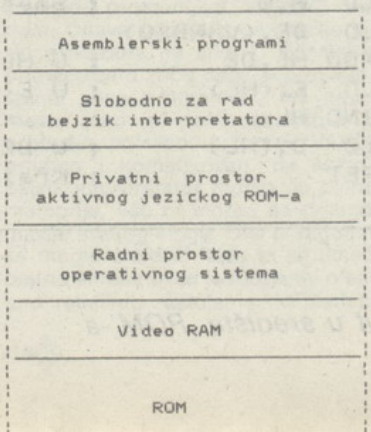
Prostor između LOMEM-a i HIMEM-a možemo da rasporedimo na nekoliko načina, od kojih su dva prikazana na slici 2. Prvi od njih je sasvim jednostavan: počevši od LOMEM-a, u memoriju se smešta bejzik program a počevši od HIMEM-a, prema nižim adresama, vrednosti promenljivih. Uveli smo promenljivu TOP koja sadrži adresu prvog bajta posle kraja bejzika i promenljivu VAREND koja pokazuje kraj zone numeričkih i alfanumeričkih promenljivih i nizova. Samo se po sebi razume da se vrednost TOP-a menja svaki put kada ubacimo, izbrišemo ili editujemo neku programsku liniju, dok se vrednost VAREND-a menja kad god definišemo neku novu promenljivu ili dimenizonišemo niz. Ukoliko bi nekim dimenizonisanjem VAREND postalo manje od TOP-a, bila bi prijavljena greška tipa ‚No room'.

Dobra strana ove organizacije memorije je njena jednostavnost, dok je loših strana daleko više. Prva i osnovna: nismo obezbedili prostor za smeštanje steka koji omogućava povratke iz potprograma: kad god bejzik interpretator treba da izvrši GOSUB, na neko mesto mora da upiše lokaciju instrukcije koju je trenutno izvršavao (i neke druge informacije o kojima ćemo još govoriti) da bi iz potprograma mogao da se vrati. Kako bez potprograma ne možemo da opstanemo, ove informacije ćemo smeštati u deo privatnog radnog prostora bejzika, što znači da će korisnik moći da pozove samo nekoliko nivoa potprograma. Ovaj broj je obično šest i dovoljan je za normalan rad, ali ne omogućava realizaciju rekurzija.

Druga slabost je činjenica da se HIMEM menja pri svakoj promeni grafičkog moda. To znači da će docnije, kada u program stavimo instrukiju MODE n, biti izgubljene vrednosti svih promenljivih, što može da bude itekako neprijatno. Moglo bi se, naravno, realizovati prepisivanje vrednosti promenljivih ispod novoga HIMEM-a, ali je takvo rešenje u slučajevima kada su promenljive ulančane u listu (o tome nešto docnije) vrlo neprijatno. Na posletku, ukoliko korisnik našeg računara bude poželeo da piše mašinske rutine, naći će se u velikoj nevolji: gde da ih stavi? Ako stavlja mašinski potprogram iza TOP-a, bilo kakvo editovanje bejzika će ga upropastiti. Smeštanje mašinskog potprograma ispod VAREND-a lako može da dovede do istih posledica. Ostaje samo da se rutina smesti negde u sredinu memorije između TOP-a i VAREND-a u nadi da ga bejzik program i sadržaji promenljivih neće „pojesti", pri čemu računar neće imati praktično nikakvu mogućnost da detektuje situaciju u kojoj se ovo dešava i prijavi grešku.

Drugo rešenje je daleko bolje ali i komplikovanije za realizaciju. Uveli smo, najpre, promenljivu PAGE koja označava stvarni početak bejzik programa. U početku rada računara je PAGE=LOMEM, ali korisnik ima mogućnost da otkuca nešto poput PAGE=PAGE+&1000 i tako rezerviše četiri kilobajta za mašinski program od LOMEM do LOMEM+&1000. Kraj bejzik programa, kao i do sada, označava promenljiva TOP, ali se sada odmah iza bejzika smeštaju promenljive i nizovi. Njihov početak je promenljiva VARBEG, a kraj, kao i do sada, VAREND. U početku je VARBEG=TOP, što znači da će se posle svakog editovanja programa izgubiti vrednosti svih promenljivih i nizova, što može ali i ne mora da bude problem: retki su programi koje posle ispravke vredi izvršavati od mesta na kome je nastupila greška. Ukoliko je to baš potrebno, korisnik će otkucati nešto poput VARBEG=VARBEG+&1000 i tako predvideti četiri kilobajta za produženje bejzik programa.

Od HIMEM-a naniže smo rezervisali prostor za stek pozvanih procedura; njegov kraj obeležava promenljiva FRETOP. Prostor između VAREND-a i FRETOP-a je potpuno slobodan i nije ga preporučljivo koristiti za mašinske potprograme obzirom da se prostor za njih rezerviše na drugi način. Šta smo dobili ovakvom organizacijom memorije? Bez većih žrtvi smo, pre svega, uklonili sve probleme koje smo ranije spomenuli. Činjenica je da smo se odrekli mogućnosti da u nekom od potprograma (kada su neke vrednosti smeštene na stek procedura) menjamo grafički mod, tj. utičemo na vrednost HIMEM-a, ali je to daleko manji problem nego gubitak vrednosti promenljivih pri pomeni moda. Otežali smo, na žalost, proširenje radnog prostora operativnog sistema: ukoliko nam je, na primer, potreban dodatni memorijski prostor za „pamćenje" definicija karaktera, moraćemo da pomerimo LOWMEM naviše što će izazvati ne samo gubitak vrednosti promenljivih već i gubitak čitavog bejzik programa. Ovakvi gubici su, na žalost, neposredna posledica našeg opredeljenja za grafičke modove koji će zauzimati različite kapacitete RAM-a.



Ukoliko je našoj video memoriji potreban fiksan prostor, možemo da se opredelimo za mapu poput one na slici 3. Vidimo da je video memorija smeštena zajedno sa privatnim radnim prostorom operativnog sistema kome, strogo uzevši, i pripada. Umesto HIMEM imamo promenljivu RAMTOP, koja označava zadnju postojeću memorijsku adresu. Kada nam bude zatrebao prostor za mašinske potprograme ili za definicije karaktera, otkucaćemo RAMTOP=RAMTOP−&1000 i, kao i ranije, imati četiri „skrivena" kilobajta. Neki računari, kao što je spectrum imaju naredbu CLEAR kojom se utiče na vrednost RAMTOP-a, dok drugi, kao što je TRS 80, zahtevaju da se veličina „skrivene" memorije specificira po uključenju računara.

Kako ulančati promenljive

Modul za rad sa promenljivim i nizovima predstavlja relativno složen deo bejzik interpretatora kome ćemo posvetiti dužnu pažnju. Rad sa promenljivim treba posmatrati sa dva aspekta: možemo da proučavamo način na koji su one raspoređene u memoriji i algoritme koji mogu da se primene da bi neka od njih bila pronađena, ili da razmišljamo o tome kako računar „pamti" cele i racionalne brojeve. U ovom nastavku našeg putovanja u središte ROM-a ćemo se zabavljati samo prvim aspektom problema, dok ćemo preostale razmotriti u sledećim „Računarima".

Svi znamo šta su bejzik promenljive: njihova imena su kod nekada popularnih računara sadržavala samo jedno slovo ili, eventualno, slovo i broj, dok se danas za imena promenljivih mogu koristiti i čitave reči. Posmatrajmo, najpre, najjednoyavniji slučaj u kome postoje samo celobrojne promenljive, od kojih svaka zauzima samo dva bajta (vrednosti, dakle, mogu da se nalaze između −32768 i +32767) i čije ime predstavlja jedno slovo. Postoji, dakle, 26 promenljivih (A, B, C, ... Y, Z) koje, sve ukupno, zauzimaju 26*2=52 bajta. Obzirom da 52 bajta nisu neka silna stavka, rezervisaćemo za njih prostor odmah iznad LOWMEM-a, dok će PAGE u startu imati vrednost LOWMEM+52. Modul bejzik interpretatora koji pronalazi promenljivu čije je ime smešteno u ćeliju koja se zove IME izgleda poput slike 4 (izuzeli smo delove modula koji obrađuju eventualne greške):

Obzirom da je ovo prvi modul bejzik interpretatora ili operativnog sistema koji smo u celini napisali, možete da pretpostavite da je prilično jednostavan, što je i jedina prednost ovakvog statičkog dodeljivanja prostora promenljivima. Prva mana koju ćemo zapaziti je da smo za numeričke promenljive odvojili 52 bajta, čak i u slučaju da programer koristi jedino I i J koje bi, same po sebi, zauzele samo četiri bajta. Sa druge strane, već smo rekli da 52 bajta ne predstavljaju neku veliku memoriju, a sada ćemo priznati da su ta 52 bajta vrlo racionalno iskorišćena: nimalo prostora nije utrošno na pamćenje imena promenljivih, pošto njih možemo da nađemo koristeći fiksan algoritam. Druga mana je što imamo samo 26 promenljivih, dok nam može zatrebati daleko više od toga. Treća i ubedljivo najozbiljnija mana ovakve organizacije memorije je postojanje samo jednog tipa promenljivih: nema stringova, nema nizova i nema matrica! Obzirom da se čak i najjednostavniji programi ne mogu napisati bez korišćenja nizova, možemo da pribegnemo rešenju zastupljenom kod „galaksije": počevši od RAMTOP-a u memoriju se upisuju elementi niza A(1) i to tako da A(0) zauzima poslednja dva bajta memorije, A(1) dva bajta ispred njega i tako dalje. U bejzik programu nema potrebe za dimenzionisanjem niza jer je on jedinstven i može da zauzme sav prostor od TOP-a do RAMTOP-a dok se adrese njegovih elemenata računaju po formuli **adresa::=RAMTOP−2*1+1.**

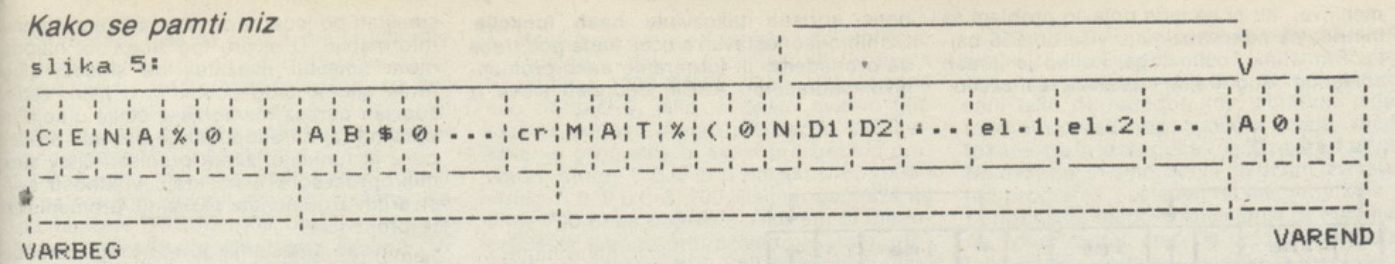
Statičko dodeljivanje prostora promenljivima je, dakle, prihvatljivo samo u slučajevima kada je prostor u ROM-u jako ograničen i kada je potrebno veoma brz rad računara a prihvatljiv nizak komfor. Od njega ne treba, međutim, sasvim odustajati: mnogi računari (Sharp 1500, BBC B, Electron ...) odvajaju fiksan memorijski prostor za 26 celobrojnih promenljivih A%—Z%, dok se ostale promenljive smeštaju u slobodan memorijski prostor na način o kome ćemo još govoriti. Promenljive A%—Z% (nazivaju se „rezidentne promenljive") mogu vrlo lepo da se iskoriste za povezivanje bejzik programa sa asemblerskim potprogramima, kao i u situacijama kada je potreban brz rad računara.

Sledi malo proširenje naših zahteva: umesto da dozvolimo samo jednoslovne promenljive, dozvolićemo imena proizvoljne dužine, ali će vrednosti svih promenljivih i dalje biti celi brojevi koji staju u po dva bajta. Realizacija nije naročito problematična: počevši od VARBEG upisujemo ime prve promenljive koja je definisana u programu i to slovo po slovo, koristeći standardni ASCII kod. Iza poslednjeg slova imena stavljamo bajt 0 a sledeća dva bajta rezervišemo za vrednost promenljive. Šta smo na ovaj način izgubili? Pre svega, utrošak memorije za promenljivu A je sada četiri bajta (jedan za slovo „A", a drugi za nula bajt iza njega a dva za vrednost), dok smo kod statičkog dodeljivanja memorije promenljivima trošili dvostruko manje. Osim toga, ako se u bejzik programu pojavi mnogo promenljivih, svako pozivanje neke od njih će odnosti zametno procesorsko vreme. Dok za prvi problem leka praktično nema (mali dobitak postižemo ako ne stavljamo nula bajt iza poslednjeg slova imena promenljive, nego setujemo najznačajniji bit tog slova) — jer da bi računar prepoznao promenljivu **ADRESA** ta reč mora da bude negde upisana na rešavanju problema brzine bi se dalo poraditi. Pre nego što se time zabavimo, razmisličemo o mogućnostima za smeštanje matrica i nizova kao i skalarnih promenljivih različite dužine (racionalne, celobrojne, stringovi ...)

```
slika 4:
LD   A,(IME)        ; IME je celija privatnog radnog prostora bejzika u kojoj
                    ; se cuva ime promenljive kojoj treba pristupiti.
SUB  A,ASC "A"      ; Obzirom da je u A ASCII kod slova, treba ga svesti na
                    ; broj izmedju 0 i 255.
ADD  A              ; Mnozimo akumulator sa 2.
LD   L,A
LD   H,0            ; Smestamo udaljenost trazene promenljive od VARBEG u HL.
LD   DE,(VARBEG)
ADD  HL,DE          ; U HL dolazi adresa prvog bajta promenljive.
LD   E,(HL)         ; U E ide LSB promenljive.
INC  HL
LD   D,(HL)         ; U DE je sada vrednost promenljive.
RET                 ; Kraj modula.
```

Kako se pamti niz

slika 5:



Na slici 5. vidimo način na koji se pamte razni tipovi skalarnih promenljivih. Vidimo, na primer, da se celobrojne promenljive prepoznaju po znaku za procenat na kraju imena, dok stringovi, umesto njega sadrže dolar. Na istoj slici je prikazano da je format u kome se čuva matrica znatno drugačiji: poslednje slovo imena matrice je obavezno leva zagrada, a zatim sledi bajt koji sadrži ofset do početka elemenata matrice, pri čemu je **ofset::=2*D+1**, gde je D broj dimenzija. Iza ofseta slede dimenzije niza navedene u DIM naredbi i, iza poslednje od njih, prostor rezervisan za elemente niza. Pogledajmo kako to izgleda na primeru: na slici 6. je prikazan bejzik program koji je dimenzionisao celobrojnu matricu A%(1, 2) i dodelio njenim elementima slučajne vrednosti, a zatim i heks-dump dela memorije od VARBEG do VAREND uz prateće komentare. Izgleda jednostavnije kada se pogleda nego kada se o njemu priča, zar ne?

slika 6:

```
10 DIM A%(1,2)                          VARBEG=&3000
20 FOR I=0 TO 1                         VAREND=&3013
30     FOR J=0 TO 2
40         A(I,J)=RND(255)
50     NEXT J
60 NEXT I
70 END

3000: 41 25 28                  A%(     ime niza.
3003: 05                        .       ofset do prvog elementa, 3008-3003.
3004: 01 00                     ..      prva dimenzija 1.
3006: 02 00                     ..      druga dimenzija 2.
3008: 05 00                     ..      A(0,0)=5
300A: C8 00                     ..      A(0,1)=&C8=200
300C: DC 00                     ..      A(0,2)=&DC=270
300E: 21 00                     !.      A(1,0)=&21=33
3010: 1B C0                     ..      A(1,1)=&1B=27
3012: 3B 00                     ;.      A(1,2)=&9B=59
```

Preostao nam je problem stringova koji su jedine promenljive unapred nepoznate dužine (svaka celobrojna promenljiva uvek zauzima dva ili četiri bajta, svaka racionalna promenljiva pet ili šest, dok se broj (kilo)bajta potrebnih za pamćenje niza ili matrice može uvek izračunati na osnovu tipa elemenata i zadatih dimenzija). String je, međutim, manje-više proizvoljne dužine (obično je maksimalna dužina stringa 256 slova, ali je sasvim blisko pameti da nećemo rezervisati toliki memorijski prostor za svaku alfanumeričku promenljivu), pa se iza imena alfanumeričke promenljive obično upisuje njena vrednost, slovo po slovo, pri čemu se iza poslednjeg slova stavlja &0D (CR tj. Carridge Return). To sasvim lepo funkcioniše sve dok korisnik ne upotrebi naredbu poput TEKST$=TEKST$+SLOVO$, kada bejzik interpretator mora da produži jednom definisani string. Ukoliko posle definisanja tog stringa nisu definisane nikakve druge promenljive, nema problema: &0D će, jednostavno, biti prebrisano daljim tekstom, a zatim će na njegov kraj biti ubeležen novi CR karakter. No, šta da se radi ako je izvršavan program poput:

```
10 I=0
20 A$="Zdravo"
30 INPUT B$
40 A$=A$" "+B$
50 ...
```

Jasno je da je posle promenljive A$ bilo definisano B$ i da nema mogućnosti da se A$ produži bez pomeranja B$ (već smo rekli da se pomeranja ovoga tipa obično izbegavaju). Tada će bejzik interpretator stistnuti zube, prebrisati sadržaj promenljive A$, a zatim formirati novu promenljivu sa istim imenom na vrhu steka i dodeliti joj odgovarajuću vrednost. Prostor između promenljivih I i B$ će ostati prazan i nekoristan. Ukoliko se ovakve situacije događaju relativno retko, nestabilizaciono ponašanje može da se prežali. No, šta ako korisnik napiše program poput sledećeg:

```
10 A$=" "
20 INPUT B$
30 REM ... segment koji poziva potprograme koji definišu promenljive
40 A$=A$+B$
50 GOTO 20
```

Vidimo da se u svakom prolazu kroz petlju na A$ dodaje po jedno slovo, pri čemu svaki put mora da se formira novo A$ i upropasti po malo memorije. Šta da se radi kada je memorija puna? Neki bejzik interpretatori tada startuju proces koji se popularno naziva „trash colecting" ili sakupljanje đubreta. Interpretator vrši potpuno prepakivanje zone promenljivih, tako da nekoristan prostor bude popunjen. Moguće bi bilo, naravno, vršiti „trash colecting" posle produžavanja bilo kog stringa, ali se pokazalo da je takva radnja gubljenje vremena: u većini primena neće ni doći do prepunjenja memorije. Obzirom da je „trash colecting" prilično neprijatna radnja koja može da dovede do toga da se isti segment u različitim prilikama izvršava za bitno različito vreme, mnogi bejzik interpretatori jednostavno ne podržavaju ovu operaciju: memorija se puni i kada bude napunjena prijavljuje se greška. Imao vaš bejzik interpretator „trash colecting" ili ga nemao, nećete pogrešiti ako na početku programa definišete stringove na njihove maksimalne dužine. Naredba A$=STRING$(80,"") ima smisla čak i ako će odmah iza nje slediti naredba A$="""!

U znaku brzine

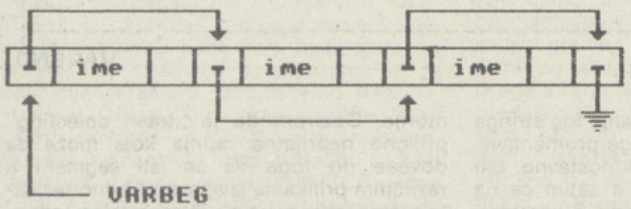
Program sa mnogo promenljivih, ako je njihovo pronalaženje organizovano na način koji smo opisali, nema mnogo šansi da bude šampion u brzini. Vešt programer će se snaći definišući na samom početku bejzik programa promenljive koje će se često koristiti, ali je takvo rešenje, kao i sva rešenja koja traže od programera da poznaje konstrukciju interpretatora, za dobrog konstruktora neprihvatljivo. Zbog toga ćemo pokušati da ubrzamo pretraživanje zone promenljivih, ali je pre toga neophodno razumeti da se nikakav značajan dobitak u brzini ne može dobiti besplatno: biće neophodno ne samo više prostora u ROM-u za programe koji će operisati sa promenljivima već i više bajtova RAM-a za njihovo smeštanje. Obzirom da cene memorijskih čipova danas nisu prevelike, trampa memorija za brzinu se obično pokazuje unosnom.

Umesto da prosto nabrojimo imena promenljivih i njihove vrednosti, možemo ove elemente da ulančamo u listu. Slog koji se odnosi na promenljivu koja je u programu najpre definisana počinje od adrese VARBEG. Prva dva bajta tog sloga sadrže adresu sloga koji se odnosi na drugu promenljivu, sledi ime prve promenljive i prostor predviđen za njenu vrednost. Kada bejzik interpretator docnije bude tražio TEKST$, neće morati da pretražuje sve prethodne stringove bajt po bajt da bi prepoznao kraj jednog i početak drugoga: čim primeti da ime promenljive na koju je naišao nije TEKST$, znaće gde se nalazi sledeći kandidat! Poslednji definisani element liste sadrži dva nula bajta umesto ukazatelja na sledeći element, što čini sistemsku promenljivu VAREND nepotrebnom. Čitava ovakva organizacija je šematski prikazana na slici 7. Posmatrajući je, primećujemo zbog čega je problem premeštanja zone promenljivih na neko drugo mesto (ako je, pošto se bejzik program produžio, potrebno promeniti VARBEG) neprijatan i komplikovan: ne samo što bi bejzik interpretator morao da pomera blok memorije, već bi morao da prilagđava zaglavlje svakog sloga. Ovo prilagođavanje bi se moglo olakšati kada bi se umesto dvobajtne adrese uveo jednobajtni ofset, odnosno relativno rastojanje do sledeće pro-

→

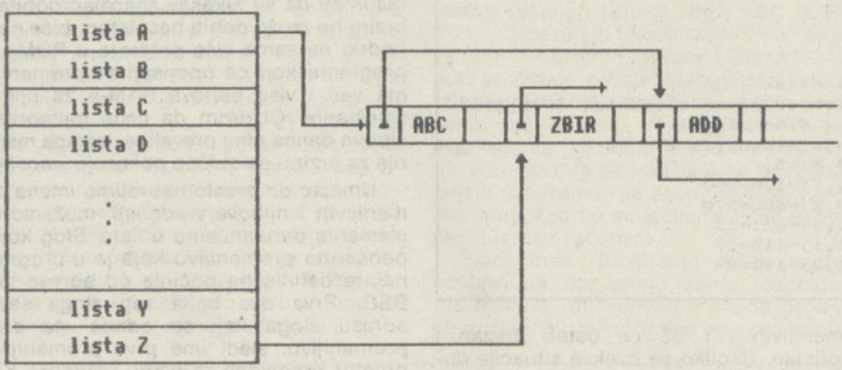
menljive, ali bi se tada pojavio problem sa matricama koje zauzimaju više od 256 bajta. Primetimo, osim toga, koliko je „trash colecting" dugotrajan i težak za realizaciju.

slika 7:



Ulančavanjem elemenata u listu se ostvaruju određeni dobici, ali je njeno pretraživanje i dalje zametan posao: dobitak je jedino što ne moramo da rešavamo problem stringova promenljive dužine. Međutim, mala varijacija ovog principa može da da izvanredne rezultate: umesto jedne, formiraćemo 26 raznih lista, pri čemu su u svaku od njih ulančane promenljive koje počinju istim slovom. Negde u privatnom radnom prostoru bejzika sada moramo da odvojimo 52 bajta za ukazatelje na početak listi promenljivih koje počinju slovima A, B, C, . . . Z. U slogu koji pripada prvoj promenljivoj čije ime počinje sa A nalazi se ukazatelj na sledeću promenljivu koja počinje istim slovom i, tako sve do poslednje, koja sadrži ukazatelj 0. Ukoliko nije definisana nijedna promenljiva koja počinje slovom K, odgovarajuća dva bajta u tabeli iz privatnog radnog prostora bejzika treba da sadrže nule. Čitava ova koncepcija je shemtaski prikazana na slici 8.

slika 8:



Ukoliko sastavite program koji realizuje ideje iz prethodnog pasusa i isprobate ga, bićete zadivljeni rezultatima toliko da nikada nećete poželeti da dalje komplikujete organizaciju promenljivih. Posvetite nam, ipak, još malo vremena: iako objektivno izvanredan, izloženi metod ima mane subjektivne prirode. Pregledajte sve programe koje ste u životu napisali i izbrojte sva imena promenljivih koja počinju slovom Q. Verovatno nećete naći ni jednu. Pokušajte sada sa promenljivima koje počinju slovima A, I i J — brojanje će vam brzo dosaditi! Ljudi, jednostavno, imaju običaj da daju slična imena promenljivima i tako, u nekom graničnom slučaju, iskoriste samo jednu ili dve od raspoloživih 26 lista. Zato je nešto bolje koristiti takozvane **hash funkcije.** Uzmimo jednostavan slučaj: kada god treba da pronađemo ili formiramo neku promenljivu, sabraćemo ASCII vrednosti slova iz njenog imena (za ABD$ bi ovaj zbir bio 65+66+68+36=235), a zatim naći ostatak pri deljenju tog zbira sa 26 (u našem slučaju taj je ostatak 1) i na taj način dobiti broj liste u koju je ulančana naša promenljiva. Negde u privatnom radnom prostoru bejzika ćemo naći ukazatelj na ovu listu, a zatim, pretražujući je, i našu promenljivu. Zgodna strana ovoga metoda je u tome što ne moramo da formiramo 26 lista, već onoliko koliko želimo: da smo želeli 10 lista jednostavno bismo pronašli ostatak pri deljenju 235 sa 10 i tako dobili broj liste. Nezgoda je jedino u tome što ASCII kodovi ne zauzimaju sve vrednosti od 0 do 128, što znači da će neke liste biti opterećenije od drugih. To možemo da izbegnemo uvođenjem neke složenije hash funkcije, ali pri tom ne treba preterivati: izračunavanje vrednosti ove funkcije može da odnese dobar deo vremena koje je dobijeno njenom primenom!

Rad sa potprogramima

Raspoređujući memorijski prostor između LOMEM-a i HIMEM-a, rekli smo da se na kraju memorije nalazi prostor rezervisan za rad sa potprogramima. „Bejzik iz Neolita" je podržavao samo jedan način za pozivanje potprograma — naredbu GOSUB. Obzirom da ova naredba ne podržava rekurzije i ne omogućava prenošenje argumenata, nije potrebno realizovati pozivanje više od pet-šest nivoa potprograma, što može da se ostvari statičkim dodeljivanjem memorijskog prostora u privatnoj zoni bejzik interpretatora. Moderni bejzici, sa druge strane, omogućavaju prenošenje argumenata u potprograme, pa se interpretator mora više potruditi. Zato će se, pri svakom pozivu neke procedure, na bejzik stek (bejzik stek nije isto što i stek mikroprocesora — stek mikroprocesora zauzima deo radnog prostora operativnog sistema, dok se bejzik stek smešta između HIMEM-a i FRETOP-a) smeštati po jedan slog koji sadrže potrebne informacije. U okviru tog sloga se, najpre, mora smestiti ukazatelj na sledeći slog (nula ako je slog poslednji u listi), zatim zapisati adresa memorijske ćelije u koju je upisan kraj naredbe koja je pozvala proceduru ili funkciju, zatim prepisati čitav stek mikroprocesora i, na kraju vrednosti svih stvarnih argumenata i lokalnih promenljivih te procedure.

Smisao smeštanja adrese kraja naredbe koja je pozvala proceduru na bejzik stek je jasan: od tog mesta će se, po završetku izvršavanja potprograma, nastaviti interpretacija glavnog programa. Potreba za pamćenjem lokalnih promenljivih i stvarnih argumenata će biti jasna svima koji pročitaju tekst „Funkcije i procedure" u ovom broju „Računara", pa ćemo se ovde zadržati na prepisivanju steka mikroprocesora. Ovaj stek dele operativni sistem i bejzik interpretator, pri čemu su njegove dimenzije sasvim dovoljne za rad sistema. Bilo bi, međutim, veoma komplikovano kontrolisati popunjenost steka pre svake PUSH instrukcije (osim, naravno, ako se za takvu proveru brine sam mikroprocesor) što znači da će prekoračenje kapaciteta steka dovesti do kraha sistema. Bejzik interpretator, kao i svaki složeni mašinski program, u velikoj meri koristi stek ostavljajući na njemu brojeve koje će možda i znatno docnije koristiti (u ovu tvrdnju ćete poverovati tek kada, u sledećim „Računarima", budemo govorili o izračunavanju brojnih izraza). Ukoliko je između stavljanja nekog podatka na stek i njegovog korišćenja pozvana neka procedura koja je rekurzivno pozvala samu sebe i šezdesetak puta ponovila istu operaciju, stek mikroprocesora će se prepuniti sa katastrofalnim posledicama. Da bismo to izbegli, prepisaćemo čitav stek na bejzik stek i zapamtiti vrednost registra SP, a zatim inicijalizovati ovaj registar tako da se stek mikroprocesora puni od početka. Kada rad potprograma bude završen, registar SP će dobiti staru vrednost, stek će biti restauriran i rad interpretatora će se normalno nastaviti. Jedino ograničenje ove koncepcije je činjenica da upotreba steka mikroprocesora u proceduri mora da bude „čista": ne može se u toku interpretacije procedure staviti nešto na stek da bi to bilo pročitano kada se bude dalje izvršavao glavni program.

Prepisivanje vrednosti stvarnih argumenata na bejzik stek možemo da izostavimo a da time nešto i dobijemo. Ukoliko vrednosti svih stvarnih argumenata prenesemo na stek i sa njega ih skinemo po završetku rada procedure, bilo kakva izmena koju je na nekom od argumenata izvršio potprogram će biti poništena, što će onemogućiti vraćanje rezultata iz potprograma. Zašto bi se nešto ovakvo izbegavalo? Zamislite da autor bejzik programa napiše nešto poput:

```
100 PROCproblem(1, 2, 3)
110 . . .
200 DEFPROCproblem (A, B, C)
210 A=5:B=7:C=9
220 ENDPROC
```

Bejzik interpretator će pokušati da konstanti 1 dodeli vrednost 5, što će izazvati vrlo čudno ponašanje računara u budućnosti. Neki programski jezici (npr. paskal) rešavaju ovaj problem tako što zahtevaju od korisnika da u listi formalnih argumenata specificira koji će od njih biti menjani a koji neće (ukoliko se docnije toga ne bude držao, sam je kriv za ono što će se dogoditi), dok drugi (fortran 77) podrazumevaju da

potprogram sme da menja sve argumente, a funkcija ne smi ni jedan. Ukoliko sastavljate bejzik interpretator, možete po želji da se opredelite za bilo koju od ovih koncepcija a da vaše rešenje ima jednako mnogo dobrih i loših strana.

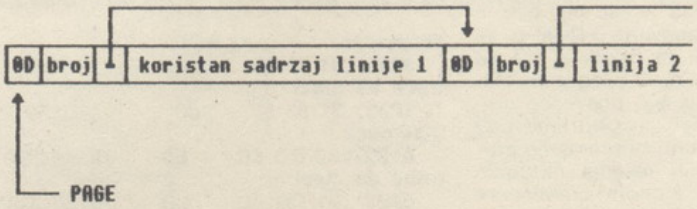
Problem u vezi sa procedurama i funkcijama predstavlja i njihovo pronalaženje: kada napišete GOSUB 1000, bejzik interpretator će pretražiti prostor između PAGE i TOP i pronaći linijski broj 1000 (ako on postoji), posle čega će znati gde počinje potprogram. Pri pozivu procedure se navodi njeno ime što je daleko više u skladu sa normama struktuiranog programiranja (kada docnije budete gledali program, znaćete da PROCx0y crtą koordinatni sistem dok vam GOSUB 2788 neće ništa značiti), ali će zato pronalaženje proceure biti zametan posao. Linije programa su, kao što ćemo videti, ulančane u listu, pa je pronalaženje neke od njih manje-više brz posao, dok se traženje nekog imena može realizovati jedino mukotrpnim pretraživanjem zone programa, bajt po bajt. Rešenje je mali trik: pri prvom pozivu potprograma moraćemo da pretražujemo memoriju dok ga ne nađemo. Tada ćemo u zonu numeričkih promenljvih (iznad VARTOP-a koji ćemo zatim modifikovati) upisati slog koji sadrži ime pronađene procedure i njenu adresu u memoriji. Sve ovakve slogove povezujemo u jednu listu i izdvajamo dva bajta u privatnom radnom prostoru bejzik interpretatora za pokazivač na njen početak. Kada docnije bude pozvana neka procedura, proverićemo da li se njeno ime već nalazi u listi i, ukoliko ga nađemo, eliminisati potrebu za pretraživanjem programa. Dalje dobitke u brzini bismo dobili ako bismo formirali 26 lista, po jednu za imena procedura koje počinju istim slovom, ali je ovakvo nešto obično sasvim nepotrebno.

Pakovanje programa

Bejzik program je, u stvari, običan tekst koga bismo mogli da formiramo uz pomoć nekog standardnog tekst editora (tako se to i radi na velikim sistemima). Bejzik interpretator koji sastavljate mora, međutim, da sadrži i neki editor koji će omogućiti nesmetano i komforno (ovu poslednju karakteristiku zaboravite ako planirate da tražite posao u firmi Sinclair Research) kucanje i ispravljanje programa. Pošto je ovaj editor integralni deo bejzik inzrpretatora i neće biti korišćen ni za šta drugo, bejzik interpretator može da na njega prebaci deo svojih poslova, pa će se u memoriji nalaziti tokenizovan bejzik program.

Najjednostavniji editor bi ubacivao linije u memoriju baš onako kako ih korisnik kuca: linija 1000 GOTO 700 bi bila upisana kao na slici 9. Iako bi nešto ovakvo bilo sasvim jednostavno za realizaciju, pri izvršavanju programa bi svaki put trebalo pretvarati string 1 0 0 0 u linijski broj 1000 i string 7 0 0 u broj 700. Osim toga, traženje linije 700 bi se obavljalo tako što bi mikroprocesor lagano pretraživao program, pronalazio karaktere &0D koji označavaju krajeve linija i prepoznavao linijske brojeve iza njih. Ovakvo je rešenje prihvatljivo samo za računare sa veoma malim ROM-om; malim usložnjavanjima interpretatora mogu da se postignu ogromni dobici u brzini.

slika 10:

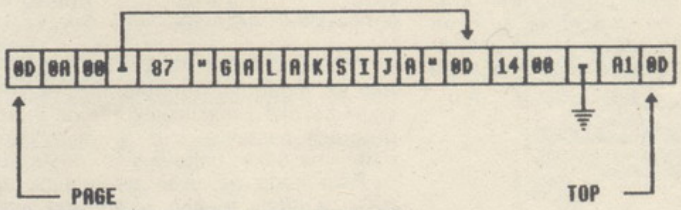


Na slici 10 vidimo da se na početku svake linije nalazi njen broj, ali predstavljen u obliku dvobajtnog celog broja, a ne u obliku stringa. Sledi bajt koji sadrži dužinu linije, što znači da bejzik program možemo da zamislimo kao ulančanu listu linija. Sledi koristan sadržaj linije i, na kraju , &0D što označava njen kraj. Na prvi pogled bi se reklo da je ovaj bajt nepotreban, pošto je dužina linije navedena na njenom početku. Postojanjem ovoga bajta je, međutim, sigurnosni ventil koji će popustiti čim program bude učinjen neupotrebljivim neopreznom upotrebom naredbe POKE: bejzik interpretator će u određenim prilikama proveravati da li se na kraju svake linije nalazi CR i prijavi grešku ako to nije slučaj.

Što se tiče dela linije koji smo na prethodnoj slici nazvali „koristan sadržaj", on se neće sastojati od alfanumeričkog teksta naredbi: primeniće se tokenzacija. Svaka prepoznata bejzik naredba će biti zamenjena jednim jedinim bajtom, tokenom te naredbe. Tako će, na primer, naredbi PRINT biti dodeljen token &87, a naredbi END token &A1 pa će bejzik program

10 PRINT „Galaksija"
20 END biti smešten u memoriju kao na slici 11.

slika 11:



Osim naredbi često se tokenizuju i specijalni znaci. Time se, dalje, ubrzava izvršavanje programa: tokeni su obavezno grupisani tako da naredbe koje obrađuje jedan modul operativnog sistema imaju slične tokene, pa ih je moguće vrlo brzo prepoznati. Na ovaj način smo dobili na brzini izvršavanja programa i uštedeli na memoriji koja je potrebna za njegovo smeštanje uz minimalne žrtve: morali smo da usložinmo operativni sistem, dodajući module za tokenizaciju i detokenizaciju naredbi.

Osim tokenizacije, transformišu se i sve numeričke konstante: za bejzik interpretator je konvertovanje stringa 1.234 E15 u broj u pokretnom zarezu zametan posao koji nije racionalno obavljati pri svakom izvršavanju naredbe. Zato će većina računara iza alfanumeričke predstave broja rezervisati četiri ili pet bajtova u koje će biti upisana njegova binarna predstava; pri izvršavanju programa će string jednostavno biti ignorisan. Ako ubuduće negde pročitate da je na „spektrumu" bolje napisati LET A=VAL „1234" nego LET A=1234, jer prva varijanta zauzima manje memorije, imaćete priliku da se nasmejete. Jasno je da autor takve pojave ne zna da se primenom izložene „veštine" odrekao onoga što su konstruktori „spectruma" uradili za njega: trampe nekoliko bajtova memorije koje „spectrum" i tako ima dosta za dobitak u brzini rada, koja ovom računaru nije jača strana.

Ukoliko ste pročitali i razumeli ova četiri nastavka našeg „Putovanja u središte ROM-a" i možete da kreativno primenite stečeno znanje, nećete imati mnogo problema sa razumevanjem strukture i rasporeda modula operativnog sistema i bejzik interpretatora bilo kog kućnog računara. Ukoliko poželite da sami pišete neke od tih modula, potrebno vam je više detalja kojima posvećujemo naše sledeće nastavke.

slika 9:

```
                                PAGE=&2000
                                TOP= &200B

2000: 31 30 30        1000      broj linije.
2003: 20                        blanko pre naredbe
2004: 47 CF D4 CF     GOTO      tekst naredbe.
2007: 20                        blanko izmedju GOTO i 700.
2008: 37 30 30        700       adresni deo naredbe.
200B: 0D                        kraj naredbe.
```

Daleko vam je, naravno, potrebnije iskustvo koje jedino možete steći ako zagrejete stolicu, računar i monitor.

Dejan Ristanović

rom od sedam milja (2)

Računari u domaćoj radinosti

Projekat megaroma za „spektrum" uneo je dosta uzbuđenja u već pomalo učmao život YU spektrumovaca, koji su počeli ozbiljno da strahuju da ih vreme polako gazi zajedno sa njihovom mašinom. Bilo je, međutim, i protesta tipa „zašto nije bilo ni reči o NM! i ostalim bagovima u ROM-u" ili „zašto je uveden link za bejzik a nije se nimalo mislilo na mašinske programere".

Članak, uistinu, nije bio posvećen mogućim prekrajanjima „spektrumovog" ROM-a — za sledeći broj pripremamo pregled ideja i praktičnih rešenja mogućih zakrpa — već samo jednom praktičnom projektu, od koga upravo programeri u mašincu imaju najviše koristi. Primedbe ovoga tipa, očigledno, ukazuju na to da su ovakve izmene najčešće rezultat snažnog poriva da se računar prilagodi sopstvenim potrebama, a one retko kada mogu biti od opšteg interesa. Zato, uostalom, pomoćni ROM-ovi i postoje — u njih može staviti svako što god hoće.

Ambicioznije vlasnike „spektruma" najviše je zbunila nekompatibilnost rešenja sa mikrodrajvom. Da li je bilo baš toliko komplikovano da se ona očuva? Nimalo! Za one koje ne zanima mikrodrajv to predstavlja najelegantnije rešenje, a oni koji ga imaju mogu sa dve-tri intervencije — pod uslovom da smisle zaista bezbedno mesto za ROM OS (operativni sistem za rad sa alternativnim romovima) — da povrate kompatibilnost veoma jednostavno:

- umesto adresne linije A4, na nožicu 1B integrisanog kola 74LS139 treba dovesti liniju A7; adresa za uključivanje eproma će sada biti 127;
- instrukcije tipa OUT (239), A treba prepraviti u OUT (127), A;
- program treba disasemblirati i promeniti mu pseudonaredbom ORG tzv. origin (određuje mesto na kome se nalazi i izvršava program u memoriji).

Sledeća grupa pitanja odnosila se na drugi link za naredbe. Odakle se on poziva? Iz programa koji trajno zauzima prvi link. Nimalo, međutim, ne smeta da se i on poziva iz ROM-a, i to odmah nakon pozivanja prvog linka. Tada bi izmena 4 mogla da glasi:

```
0008 CD B6 5C CALL &5CB6
000B CD B9 5C CALL &5CB9
000E 18 43 JR &53
```

Zbog nedostatka prostora, na samom prelomu lista, morali smo da izostavimo tabelu sa strukturom zaglavlja — „lične karte" koja prethodi svakom programu u epromu. Format zaglavlja nije isti za sve tipove programa. Ako se u epromu nalazi samo bejzik ili samo program na mašinskom jeziku, onda zaglavlje ima 22 bajta.

U listinzima izmena u osnovnom ROM-u načinjene su dve štamparske greške. Obe su u delu za objektni kod, dok je deo sa izvornim kodom (mnemonicima) sasvim u redu.

štampano
1235 21 46 5E LD HL,&5E53
treba da glasi
1235 21 53 5E LD HL,&5E53
štampano
04B2 32 CO 5C LD HL,&5CCO
treba da glasi
04B2 21 CO 5C LD HL,&5CC0

apsolutna adresa	relativna adresa	funkcija
0000	IX−01	Broj eproma (0-FF).
0001	IX+00	Tip programa: 00 bejzik 01 bejzik+mašinac 03 mašinac.
0002	IX+01	Ime programa (najviše 7 slova); neiskorišćene bajtove popuniti sa FF.
0009	IX+08	Oznaka kraja imena (FF).
000A	IX+09	Dvobajtna adresa u EPROM-u sa koje treba preslikati program.
000C	IX+0B	Dvobajtna dužina programa.
000E	IX+0D	Adresa na koju treba preslikati program; za bejzik programe broj linije od koje se program startuje.
0010	IX+0F	Dužina programa sa promenljivim (bejzik); nova adresa mašinskog steka (mašinac); FFFF ako za mašinac ne treba nova adresa steka.
0012	IX+11	Adresa sledećeg zaglavlja u epromu ili FFFF.
0014	IX+13	00: u EPROM-u nema više programa; FF: u EPROM-u ima još programa.
0015	IX+14	00: program se ne nastavlja u drugom epromu; NN: program se nastavlja u epromu NN.
0016	IX+15	Prvi bajt mašinskog programa ili bejzika; Za kombinacije bejzik/mašinac i za programe koji se nastavljaju u drugom epromu: izvorna adresa mašinskog bloka ili nastavka.
0018	IX+17	Dužina mašinskog bloka ili nastavka.
001A	IX+19	Odredišna adresa mašinskog bloka ili nastavka
001C	IX+1B	Adresa novog mašinskog steka ili FFFF.
001E	IX+1D	Prvi bajt programa.

Ako se, međutim, u njemu nalazi kombinacija i bejzika i mašinskog jezika, ili ako se program nastavlja i u drugom epromu, onda ima osam bajtova više, dakle 30.

Prvo zaglavlje uvek počinje na adresi 0000, a sledeće, ako u epromu ima još programa, na adresi = dužina prvog programa + dužina njegovog zaglavlja. Adresa zaglavlja stalno se čuva u registru IX, a do svih potrebnih podataka se dolazi preko indeksa. Prvi bajt u zaglavlju (IX−1) u ovoj verziji programa ROM OS nije iskorišćen — njime je samo pripremljen teren za savršeniju verziju koja će, između ostalog, imati i naredbu *CAT(alog). Na pojedinim adresama u zaglavlju nalazi se na istom mestu nekoliko raznorodnih podataka. Program ROM OS se, međutim, nikada neće zbuniti — potrebna grananja nastaju na osnovu podatka o tipu programa u epromu na relativnoj adresi IX.

Konstruktorima koji su se prošlog leta u ovo vreme upustili u samogradnju programatora eproma život nije bio nimalo lak — cene integrisanih kola su divljale na svetskom tržištu, skačući i do 700 odsto! Ovoga leta duvaju znatno povoljniji vetrovi — cene epromima od 8 i 16 kilobajta su toliko pale da su se izjednačile sa cenom najobičnije igre na kaseti. Tako eprom tipa 27128 sada košta ispod 30 maraka. Ova okolnost, verujemo, čini naš projekat još privlačnijim.

J. Regasek

Računari u domaćoj radinosti

brisač eproma

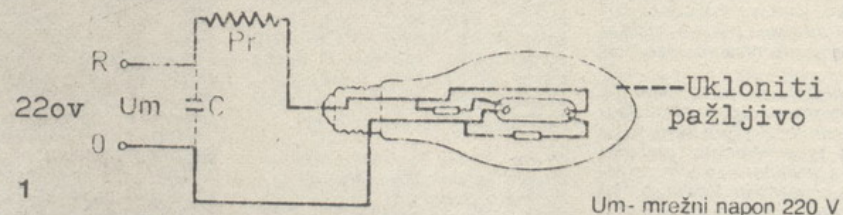
Brisanje eproma može da bude izuzetno naporno i složeno, naročito ako nije na raspolaganju odgovarajući pribor. Kao sredstvo za brisanje služi UV (ultravioletna) lampa, koja mora da ispuni nekoliko elementarnih uslova — treba da emituje ultraljubičastu svetlost određene talasne dužine (254 m) i da ima energiju zračenja od 12mW/cm². Od domaćeg materijala i u domaćoj radinosti može se, za samo 2000 dinara, napraviti brisač eproma koji u potpunosti zadovoljava ove standarde.

Najuporniji konstruktori, koji su u potrazi za lampom za brisanje eproma stigli čak do Nemačke i Engleske i vratili se odande praznih ruku, biće, bez sumnje, iznenađeni našim otkrićem da se „toliko željena" lampa može naći na policama svake veće prodavnice elektrotehničke robe. Nije, međutim, čudno da je toliko dugo ostajala neprimećena — dobro je, kao biser u školjci, skrivena u neprozirnom sijaličnom balonu! Sijalica se može naći pod komercijalnim imenom „Živina sijalica visokog pritiska VTFE tipa".

Na slici 1. se vidi izgled VTFE sijalice ispod balona; za nas je taj deo izuzetno značajan, jer se u njemu nalazi „mala sijalica" — žižak. Slika 3. prikazuje da samo žižak emituje ultravioletnu svetlost koja je neophodna za brisanje. Da bi se do njega došlo, pažljivo da otklonite samo spoljni balon — razbijte ga ili odrežite brusilicom — a da pri tom svi unutrašnji delovi ostanu u onom stanju u kakvom su i bili.

Sijalica se ne sme povezati direktno na 220 V, jer ćete tada morati u radnju po — novu. Kao što se vidi na shemi 2. vezivanje predspojne sprave (prigušnice) i VTFE je serijsko (redno). Kompenzacioni kondenzator, nacrtan isprekidano, koji je vezan paralelno u odnosu na mrežni napon, nije obavezan. U obzir dolaze tri tipa sijalica za: 80 W, 125 W i 250 W. Mi vam preporučujemo da odaberete sijalicu od 80 W zbog niže cene i, naročito, manjeg toplotnog efekta.

Uz svaki tip sijalice dolazi i odgovarajuća prigušnica dimenzionisana prema odgovarajućoj snazi upotrebljene VTFE-sijalice. Prigušnica se obeležava sa PPO (nazivna snaga) sijalice — za sijalicu od 80 W odgovarajuća prigušnica je PPO80. Ako imate sijalicu VTFE od 80 W, a ne možete da pronađete odgovarajuću prigušnicu, sasvim dobro će poslužiti i klasična prigušnica za „neonku" od 65 W. Za 2000 dinara i pola časa rada, konstruktori dobijaju uređaj koji se, po efikasnosti, može meriti sa bilo kojim inostranim, a cena mu je, pri tome, nekoliko puta manja.



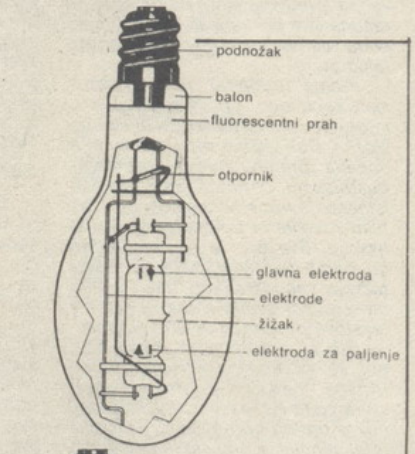
1

Um- mrežni napon 220 V
C kompenzacioni kondenzator
Pr prigušnica

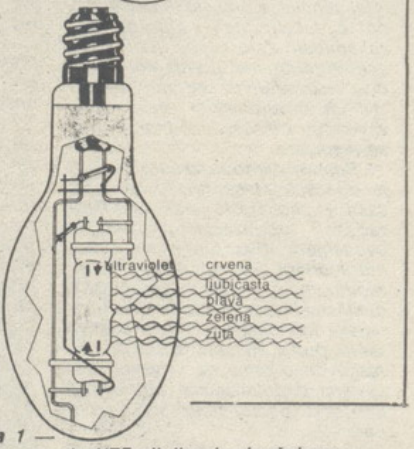
Opasne krivine

- Uređaj krije dve potencijalne opasnosti od kojih se moramo dobro čuvati. Sve delove koji su pod mrežnim naponom potrebno je dobro izolovati, a lampu postaviti u potpuno zatvorenu kutiju, jer je zračenje opasno za vid (ovaj tip svetlosnog zračenja je identičan sa zračenjem koje se stvara prilikom elektrolučnog zavarivanja)! U okolini svetiljke se oseća i (inače potpuno bezopasna) pojačana jonizacija vazduha.
- Razmak između dva paljenja treba da bude oko 5 minuta, a lampa postiže maksimalnu jačinu svetlosti za 4—6 minuta.
- Eksperimentiše sa rastojanjem između lampe i eproma tako da vreme brisanja bude između 15 i 20 minuta. Autor je izbrisao nekoliko tipova eproma za računar „galaksija" na rastojanju od 10 cm, sa lampom od 80 W (vidi tablicu), za manje od 10 minuta. (Najbrže su izbrisani „Hitači" epromi — 7 min.). Optimalan položaj za postavljanje eproma je paralelan u odnosu na osu lampe.

Ukoliko potrebne elemente ne pronađete u maloprodaji, možda ih imaju njihovi proizvođači:
TEP, Tvornica elektrotehničkih proizvoda-Zagreb
41090 Zagreb, II Oranički odvojak 18
TESLA, fabrika sijalica
26000 Pančevo, P. Marganovića 14 A



2



3

Tabela

rastojanje od podloge	25	50	75	mm
provodni sunđer	62,5	32,4	zan.	°C
aluminijum	48,6	40	zan.	°C

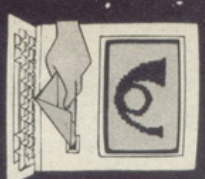
Zavisnost grejanja kućišta eproma od rastojanja i vrste podloge kada je izložen zračenju 15 minuta; merenje je izvršeno pri temperaturi ambijenta od 23,2 °C instrumentom „Tektronix" DM 501A

Slika 1 —
Shema spoja VTF sijalice i prigušnice: direktno priključivanje na 220 V nije mo[illegible]će
Slika 2 —
antaomija VTF sijalice
Slika 3 —
Spektar zračenja VTF sijalice: pošto sijalični balon apsorbuje ultraljubičastu svetlost, treba ga pažljivo ukloniti

Dragoljub Jokić

(79/računari u domaćoj radinosti.

Priprema
Branko Đaković



Load „pismo"

Za mase a ne za klase

Javljam Vam se povodom komentara i diskusija brojnih čitalaca kao i intervjua profesora Joza Dujmovica.

Činjenica je da mi stojimo na pragu informacijskog društva i da u ovom postindustrijskom periodu, u kome se sada nalazimo, moramo podržati sve napore za širenje računarske pismenosti. Međutim, čini mi se da je u sadašnjem trenutku računarstvo u našoj zemlji pomalo poprimilo neke odlike pomodarstva. No, to je i razumljivo kada nauka nije mogla ili nije imala uslove da preuzme kormilo u širenju računarske pismenosti. Logično je, onda, da je programerska mladost rekla svoje.

Prema mišljenju nekih francuskih sociologa, put čovečanstva u informacijsko društvo biće veoma složen. Dok visoko razvijene zemlje Zapada spremno koračaju u postindustrijskom periodu, dotle zemlje Istočne Evrope i Jugoslavija još nisu završile ni potpunu industrijalizaciju. Sve ovo je sigurno jedan od bitnih uzroka sadašnjeg stanja razvoja računara kod nas. Mislim da u ovom svetlu treba tumačiti i tekstove drugarice Jelene Rupnik.

Što se programskih jezika tiče, oni će se stalno razvijati i trpeti izmene. Sada je u modi bejzik, sutra će to biti paskal, lisp, prolog i mikro prolog i drugi jezici veštačke inteligencije, a ono što je univerzalno to su algoritmi i algoritamske zakonitosti. Zato mlade ljude treba učiti algoritmima, prema kojima se pišu odgovarajući programi na različitim programskim jezicima, u zavisnosti od vrste računara kojim se raspolaže.

Pojava mikroprocesora dovela je do računarske revolucije, koju sada imamo. Došlo je do automatizacije i robotizacije proizvodnje uvođenjem mikroprocesora i mikroračunara i samo takvom proizvodnjom mi možemo podići produktivnost rada. I dok mikroračunarska tehnologija u Evropi dostiže svoju punu zrelost, izlišni su svi razgovori o tome da li je kod nas suvišna popularizacija računara, a posebno personalnih mikroračunara.

Prema tome, mišljenja sam da je svaka aktivnost, koja doprinosi širenju računarske pismenosti, u sadašnjoj situaciji za pohvalu.

S druge strane, smatram da širenje računarske pismenosti treba voditi naučno i organizovano i da bi Univerzitet kao i Republika morali u većoj meri da utiču na dalje širenje tokova kibernetske naučne misli, koja objedinjuje više naučnih disciplina: teoriju sistema, teoriju upravljanja, teoriju informacija, teoriju algoritama i teoriju igara kao i njihovu primenu u daljem razvoju društva. Tako će sve aktivnosti računara, kao osnovnih kibernetskih sredstava, naći svoje mesto i neće se nekim stvarima davati veći značaj nego što to zaslužuju.

Sa ovako koordiniranim aktivnostima, zašto nemati poverenja u domaću softver?

Naprotiv, treba smelo i organizovano zakoračiti u razvoj informacionih dobara i informacionih tehnologija, koje su u Evropi već dobile status novih tehnologija. Takođe smatram da treba podsticati razvoj komercijalno raspoloživih aplikacionih programskih paketa kao i ekspertnih sistema ih oblasti obrazovanja, nauke i proizvodnje.

Dr ing Jelenka Savkovic-Sretenović
docent Tehnološko-metalurškog fakulteta u Beogradu

„Amstrad" protiv „amstrada"

Da li ste ikada uspeli da napišete program na „amstradu"? Ja nisam.

Dobre osobine mnogih računara kod nas došla je do izražaja tek kada, zbog prepunih magacina, dobro pojeftine. Tako se, na primer, za „atari 800XL" nije ni znalo dok mu cena nije drastično snižena. Kralj je jeftin, živeo Kralj! Nedavno se u prodaji pojavio računar koji za svoju cenu nudi dosta. Već posle nekoliko meseci, u našoj štampi su počeli da mu pišu hvalospeve, dižući ga u nebesa. A zna se ono staro: ko visoko leti . . . Pogađate, reč je o „amstradu."

Posle svih tih aplauza koje je pobro na YU sceni, kralj „amstrad" se našao i na mom stolu. A onda . . .

Na našem softverskom tržištu još uvek ima veoma malo programa za „amstrad" (a zar ih negde ima više), pa bi to trebalo da ohrabri (prisili?) programere da i sami napišu po neki program. Tako sam i ja krenuo da napišem program za svoj „amstrad" i to program za crtanje. I računar je počeo da prijavljuje sintakstne greške. Za divno čudo, greška je bila već u prvom redu 10 PRINT „PROGRAM ZA CRTANJE". Lupao sam glavu oko ove linije pola dana i, pošto nisam uspeo da je ispravim, rešio sam da ovaj red izbacim (i tako nije mnogo važan). Međutim, računar je prijavio grešku i u sledećoj liniji 20 GOTO1000. Katastrofa se ponovila! Rešio sam da se okanem pisanja sopstvenih programa i krenuo da prepisujem programe iz priručnika. I, gle, čuda — oni su radili (!!!). Najzad sam shvatio da sve argumente treba odvajati od naredbi (!?), inače računar odbija da radi. Pomalo zbunjen, ugasio sam računar i ostavio sve za sutra. Kakav dan!

Sutradan sam, pun eleana, ponovo uključio svoj „amstrad" i učitao program sa demo kasete. Uskoro je slika počela da beži sa ekrana. „Ne može to tako", uzviknuo sam. Prvo sam osuo drvlje i kamenje na elektrodistribuciju: „amstrad" je fina mašina, smeta mu nestabilan napon! Ugasio sam računar (drugi put) i ostavio sve za bolje dane — do kojih nikada nije ni došlo (posle pola sata rada, slika mi je uvek bežala sa ekrana). Bio sam na ivici da dam oglas „Menjam amstrad za komodor 64 ili spektrum 48K bez doplate."

Kasnije se pokazalo da su to još i najmanje mane „amstrada". Velika greška konstruktora je što kod „amstrada" stalno postoji ekran visoke rezolucije, za koji se troši 16K RAM memorije, u svakom od tri moda. Zahvaljujući tome, od 42K slobodne memorije za bejzik ostaje svega 26K. To je manje nego čak i kod onog malog crnog računara sa šarenim šraftama sa strane!

Kada sam isprobao muzičke sposobnosti toliko hvaljenog čipa AY-3-8910, prilično sam se razočarao (čip je dobar, ali podrška . . .). Od tri glasa koje „amstrad" ima, vi možete da podesite samo jedan, a sva tri zvuče isto tako, pa ne možete napraviti da vam sviraju tri različita instrumenta u isto vreme. Vrlo loše. Ipak, najgora strana „amstradovog" zvuka je mali zvučnik smešten negde kod kasetofona. Ton je vrlo loš i slab, a to, opet, neodoljivo podseća na jedan mali računar.

Kad već pomenuh „Ghostbusters", za „amstrada" su se pojavili i prvi programi na domaćem (piratskom) tržištu. Možete ih nabaviti preko Astrad User Club-a (adresu nađite u malim oglasima) i to po ceni istoj kao i u Engleskoj!

Pošto samo pomenuli programe, njih treba i učitati u računar. Kod „amstrada" postoje dve brzine učitavanja: 1000 i 2000 boda. Vrlo brzo, rekli biste vi. Ne, ne, kažem ja. Kada učitavate programe brzinom od 2000 boda, oni se učitavaju istom brzinom kao kod „spektruma" koji učitava 1200 boda! Rešenje je jednostavno. „Amstrad" snima i učitava u blokovima od po 256 bajtova. A pre svakog bloka imate heder u kome je zapisano sve i svašta, tako da za program od jednog kilobajte imate četiri (!!!) hedera. Za svaki sledeći kilobajt još četiri, pa tako u beskraj. Znači, kasetofon otpada kao spoljna periferija. Ostao nam je disk. I opet greška. Disketa od 3 inča izgubila je bitku sa onom od 3,5. Čak se i u Engleskoj sve teže nabavljaju diskete od 3 inča. A što je retko, to je i skupo. Od ostalih periferija, „amstrad" podržava još i printer. U priručniku piše (a i u reklamama) da to radi preko Centronix interfejsa. Meni to mnogo više liči na običan EDGE konektor, što znači da je priključivanje standardnih printera potreban poseban kabl.

Za džojstik postoji samo jedan port — ako hoćete da odigrate nešto protiv komšinice, moraćete da kupite „amstrad" police koje su duplo skuplje od veoma sličnih QUICKSHOOT I.

Mnogi kupuju „amstrad" zbog toga što je najjeftinija CP/M mašina (C-128 ?). Prvo daju pare za računar i disk, a onda otkriju da ne mogu da nabave ni jednu jedinu CP/M disketu od 3 inča. A ako neku, ko zna kako, i presnime, može se desiti da program ne radi, jer od 64K, koliko računar nominalno ima za CP/M, ostaje jedva pola, a to za većinu programa nije dosta. Takođe, 80 slova u redu sve ne vide baš najbolje ni na „amstradovom crno-belom monitoru, dok je u boji još gore. A šta tek da kažu oni optimisti koji su kupili „amstrad" sa UHF modulatorom.

Za one koji žele CP/M mašinu, na raspolaganju im je „komodor 128" ili, za rodoljube, Iskrin „partner". Ili mogu da kupe neki računar koji ima CP/M kao proširenje. Što se tiče grafike i zvuka, mogu se nabaviti i računari sa sličnim ili, čak, i boljim karakteristikama koji su, uz to, mnogo potpuniji na drugoj strani. U svakom slučaju, „amstrad" je dobar, ali suviše izvikan računar koji tek treba da se dokaže — ima i puno boljih od njega i to po sličnoj ceni. Što se tiče bejzika, „amstradov" je vrlo brz, ali i vrlo neprecizan (radi sa samo tri decimale), ali to nije toliko važno. Ima i važnijih stvari za jedan računar.

Nadam se da se „amstradovci" neće ljutiti zbog ovog članka. Vreme je da se ukaže i na neke „amstradove" mane, kojih ima dosta, a niko ih ni ne spominje iz ko zna kojih razloga! Vreme je da Amstradovci svhate da nemaju najbolji kućni računar na svetu — ima i puno boljih.

Dragan Grbic

Svako ima svoj ugao iz koga posmatra kvalitete i mane nekog računara. Dužni smo, međutim, da skrenemo pažnju na nekoliko netačnih činjenica:

1. Amstrad ima 64K RAMa, a za bejzik ostaje slobodno oko 43K, što saznajemo kucajući naredbu PRINT FRE("") , (a ne 26 K.)

2. Tačno je da zvuk iz zvučnika koji je ugrađen u računar nije naročit, ali zato postoji mogućnost priključivanja na stereo pojačalo. Inače, zvuk se iz „komodora" ne može uopšte čuti (iz televizora može).

3. Amstrad učitava programe u blokovima od 2048 bajta jer poseduje bafer veličine 2K, pa tako 1K programa zauzima otprilike 1/2 bloka (a ne 4).

4. Što se tiče tačnosti prikazivanja realnih brojeva, ona iznosi nešto preko devet cifara. U binarnom obliku broj zauzima 5 bajtova. Pošto je u pitanju eksponencijalna notacija, mantisa zauzima 4 bajta, a eksponent 1 bajt, što je uobičajena tačnost za 8-bitne računare i programski jezik bejzik.

(B. Tomić)

Ko je uvredio IBM-a?

Javljam vam se povodom napisa vaše novinarke, drugarice Jelene Rupnik, pod naslovom „Vita jela, zelen bor, . . . ", „računari" br. 6.

Lepo je i za pohvalu to što ima onih koji žele da budu drugačiji i da neke teme posmatraju na nov način. Problem je u tome što takav jedan pristup traži mnogo više od puke želje — traži da autor i sam

bude neka nova i svakako dobro verzirana ličnost. Čini mi se da primer koji nam pruža drugarica Rupnik ilustruje ko i kako to ne treba da radi. Koristeći izraze kao što su „šiznuti" i „zeznuti", ona verovatno pokušava da bude neposredna i neopterećena konvencijama. Ni to ne bi samo po sebi bilo loše, kada ne bi služilo pokrivanju nedostatka i duha, i nekog stvarnog poznavanja onog o čemu tako smelo i autoritativno piše. „Zdravonarodni humor" ipak nije zamena za otsustvo duha.

Postavka da narod kupuje SINKLERA, malo bolje stojeći narod KOMODORA, profesionalci BBC B, a snobovi HEWLETT-PACKARD, MACINTOSH i IBM, u najmanju ruku ilustruje retku površnost autora. Ovako smelu tvrdnju bih potkrepio sledecim predlogom: neka drugarica Rupnik sedne za „narodni" SINKLER i neka na njemu obrađuje tekstove po osam ili deset sati na dan, mesec ili više dana uzastopce. Zatim neka isto ponovi i sa računarom srednjih klasa, pa onda sa „profesionalnim" BBC računarom. Na kraju, kao verovatno ugroženom pripadniku amaterskih masa, i ukoliko je savladala svoje gađenje prema toj ogavnoj mašini muškog roda, ja je kao „snob" pozivam da sedne za gadni i klasno diferencirani IBM i ponovi prethodni test, i to, njoj za ljubav, sa programom koji košta kao i 2,2 SINKLERA. Kada i ako nakon toga obradi rezultate, bice joj, nadam se, jasnije da su „snobovi" dosta često ljudi za koje taj računar predstavlja oruđe rada („Bez alata . . . "), i da su iz tog razloga zainteresovani da svojih osam ili više sati rada provedu za takvom mašinom koja će ih u najvecoj mogucoj meri osloboditi razmišljanja o alatu i koja će im podici produktivnost. Koliko je meni poznato, upravo to je i bila osnovna svrha računara.

Iz tog ugla posmatrano, nisu li daleko veci snobovi upravo oni „jadnici" koji moraju da obijaju programe (koji su često na nivou idiota) u kojima je cilj da izbegnete zlog mrava, uhvatite duha ili ne padnete u rupu i potrošite jedan od svoja tri života, a prave žrtve oni ljudi koji bi želeli da od računara imaju i neke praktične koristi, ali ne mogu sebi da dozvole tako skupe mašine? Kratak pregled naših časopisa ce vrlo brzo pokazati da se, po mojoj slobodnoj proceni, barem 60% prostora i barem 90% oglasnog prostora u njima posvećuje igrama, a ne ozbiljnim korisničkim programima. Piratstvo je, bez sumnje, funkcija našeg relativnog siromaštva, ali su zato predmeti piratstva sigurno stvar našeg duhovnog siromaštva. Obijaju se beskrajne i jednolične igre, jer ogromna vecina vlasnika računara ih i ima da bi se sa njima igrala, a ne da bi sa njima ponešto, pored igre, i uradila — pa zar zato da ih žalimo? Pametnije bi uradili da su desetinu tog novca utrošili na učlanjivanje u biblioteke, koje zvrje prazne. Lično bih bio zadovoljan kada bih znao da se barem 10% računara kod nas u privatnim rukama koristi i za malo pametnije stvari od jurenja baba i žaba.

Svetsko tržište je odavno prihvatilo ponudu koju sam uputio drugarici Rupnik, koja pored ekonomske, ipak sadrži i namensku komponentu. Pogledajte bilo koji katalog programa za SINKLER/KOMODOR računare, pa ga zatim uporedite sa katalogom namenjenom, recimo, IBM računarima; prvi će se sastojati 95% od igara i 5% od korisničkih programa, dok će stvari u drugom stajati potpuno obrnuto. Pošto svetsko tržište ne zavisi od lokalnih hirova, time se jasno stavlja do znanja ko šta treba da kupi u funkciji ličnih POTREBA, stvari koja je kod nas potpuno nepoznat pojam. Za ovih par godina unazad, samo sam nekoliko puta imao prilike da čujem konstataciju u smislu: „Kupiću računar GXL, jer on odgovara mojim potrebama." Ko hoće da se igra, i ne treba da ima bilo šta više od SINKLERA ili KOMODORA, mada se ovaj drugi može i te kako dobro upotrebiti i za ozbiljniji rad. Na žalost, još nisam video oglas nekog pirata koji nudi, recimo, programe za obradu teksta na KOMODORU.

Voleo bih da mi drugarica Rupnik, dostojna naslednica Mirjane Bobić po pitanju računara, odgovori za koji još operativni sistem na svetu postoji makar i deo korisničkih programa koji postoje za IBM-ov MS DOS. Da li je moguće da su IBM PC i njegovi brojni klonovi proizvedeni u preko četiri miliona primeraka samo za snobove po svetu, ili je pak subjektivna i carinska slabost to što je on za nas jednostavno preskup i nedostižan?

A da li je zaista i preskup? Kada se malo preračunamo, uz pomoc cena u Engleskoj, videćemo da jedan KOMODOR sistem (računar, dve disk jedinice, monitor i štampač) koštaju približno 700; poredenja radi, jedan ADVANCE 86 B sistem (kompatabilan sa IBM-om, sa 128 umesto 64 kB RAM-a i dalje proširiv na 768 kB, dva diska od po 360 kB, monitorom i štampačem EPSON 80 FX, te paketom programa) košta oko £ 1 500, dok slično konfigurisan IBM PC može da se dobije za oko £ 1 800. Naizgled, razlike su poveće, ali samo naizgled. Razlika u ceni izmedu KOMODOR 64 sistema i ADVANCE 86 B nije 3:1, vec gotovo da je ni nema, obzirom na paket programa koji biste za KOMODORA morali da dokupite da biste postigli jednakost — ukoliko takvih programa za KOMODOR uopšte i ima. Šteta je, ali na žalost ipak istina, da tzv. sistemske potrebe za ozbiljne korisničke programe daleko nadmašuju kapacitete SINKLERA i KOMODORA. Vecina osvedočeno dobrih programa (LOTUS 1—2—3, FORMULA II, MULTIMATE, razni kompajleri) zahteva najmanje 256 kB RAM-a i neretko dve disk-jedinice od po 360 kB. Ako nekome LOTUS 1—2—3 može da obavi posao, da li je onda takva osoba snob ako kupi računar na kome može upotrebiti taj program? Ili je možda nezgoda u tome što oni koji donose zakone koji onemogucavaju kupovinu ozbiljnih računara (a SINKLER i KOMODOR to nisu, jer kao takvi nisu niti bili zamišljeni, niti proizvedeni, niti se kao takvi prodaju) nisu najbolje upoznati sa onim u vezi čega donose zakone? Kako može naša omladina da nauči COBOL, FORTRAN, LISP i druge jezike bolje od BASIC-a kada ne postoji tehnički osnov za to?

I kako tako površne i paušalne, da ne kažem neozbiljne, postavke uopšte mogu da nađu mesta u časopisima kao što je „Računar", koji je na sebe preuzeo ozbiljne, nužne i veoma odgovorne zadatke? Zašto se nekima dozvoljava da vežbaju kucanje na skupoj novinskoj hartiji? Umesto da se isti taj prostor posveti tekstovima koji upućuju na ozbiljniji rad, kao recimo onoj koji su napisali drugovi Spasić, Perzić i Ristanović.

S drugarskim pozdravima,
D.V., računarski snob
Br. lične karte 34182

P.S. U krugu snobova, u kome se takođe razmenjuju piratirani programima, što i nije čudno, obzirom na prosečnu cenu jednog takvog programa, koja iznosi koliko i BBC B računar, nezamislivo je naplacivanje bilo kakve usluge kopiranja, a normalna je stvar da se krug odmah obavesti o prispeću nekog novog programa. Neosporno, to je i dalje piratstvo, ali na nekom drugom osnovu — demokratija snobova?

Poštovani V.D. računarskog snoba, interesantno je da je Vaše pismo oko 4,5 puta duže nego tekst na koji se žalite, i u kojem se, uzgred budi rečeno, nigde ne kaže da SVI vlasnici IBM PC moraju biti snobovi. Stoga se umoljavate da se u najskorije vreme javite drugarici Rupnik radi izdavanja potvrde da ne pripadate računarskoj snobovštini. P.S. Niste dobro obavešteni: cilj igre nije da izbegnete zlog mrava; mrav je dobar, a škorpije su loše.

I car i kralj i cela monarhija

Pišem povodom članka „Amstrad" protiv „Komodora". Odmah u uvodu se vidi greška! . . . . Stavio je nedavno pod istu lupu „amstrada" i nekrunisanog kralja među kućnim računarima „komodora 64" . . . .

Mislim da je nekrunisanog kralja (komodora 64) trebalo krunisati kao cara!! On je nastao 1981. a „amstrad" čak četiri godine kasnije! Stoga mislim da nije trebalo upoređivati ova dva mikroračunara, jer se razlikuju po godini proizvodnje i mikroprocesoru („amstrad" koristi Z—80).

Kakav bi test ispao da ste poredili „BBC B" i „atari 800XL"? Zašto niste upoređivali „amstrad" i „orika"??

Mislim da ste bacili ljagu na „komodor 64" bez prava. Kako ste ga samo hvalili u „Računarima 2" sa „Nikog ne ostavlja ravnodušnim". U vreme kad se pojavio, pošto je pretekao sve dotadašnje računare, on je POSTAO i OSTAO naj-naj mikrac!

Mislim da ovakvi testovi više ne bi smeli da vam „izlete", jer možete pokolebati i najčvršceg kupca.

Ako već upoređujete računare različitih procesora, tada uporedite „amstrada" i „komodor 128" (ako ga nabavite).

Vidoje Veselinovic

Aca nije fer

Javljam se prvi put i to u vezi programa „Velika akcija". U prošlom broju „Računara" Čala Zoltan je imao primedbu da je program „Velika akcija" već bio distribuiran. Vi ste mu na to odgovorili da su za to krivi pirati. Međutim, niste u pravu! Aleksandar Radovanović ima svoju programersku (i piratsku) firmu „Computerland". Naručivši katalog od A. Radovanovića, video sam da je i on sam prodavao „Veliku akciju".

Mislim da je „Velika akcija" nezasluženo osvojila drugo mesto. Uzgred, veoma bih se zahvalio drugu Radovanoviću na programu „Kompresija" („Računari 6") koji sam ukucao u kompjuter i savršeno mi funkcioniše. Pomoću „Kompresije" sam napravio avanturu „Dvorac" koja ima mnogo slika i većim delom je pisana u mašincu. Smatram da je bolja od „Velike akcije", ali je ne želim nikom pokazivati, jer je čuvam za drugi konkurs. Molio bih da mi odgovorite na pitanje koje ne zanima samo mene nego i većinu hakera širom zemlje. KADA ĆE SE ODRŽATI DRUGI KONKURS ZA NAJBOLJEG YU PROGRAMERA I KOLIKO PROGRAMA MOŽEMO POSLATI NA KONKURS?

Nešić Aleksandar

„Velika akcija" i gresi mladosti

Tačno je da sam pripadao piratskoj „firmi" Computerland. Međutim, maja 1984. godine „firmu" preuzima drugi čovek. Program „Velika akcija" pisan je u toku leta iste godine, dakle posle moje piratske avanture.

Činjenica je da je program raden na Computerland-ovom računaru jer ja nisam posedovao svoj. Radna verzija programa je ostala njima, međutim, na moj zahtev nisu prodali ni jedan jedini snimak. Program je „procurio" zahvaljujući jednom prijatelju kome je dat radi ocene i sugestija. Pravu, konačnu verziju programa koji je učestvovao na konkursu i koji je prikazan na dodeli nagrada ne poseduje niko od preprodavaca. Verzija koja kruži među korisnicima je nedovršena. Moguće je preći 60% igre, postoje neki bagovi, a i naslovni ekran predstavlja radnu verziju. Sve to znaju oni koji imaju taj program.

Da nisam imao nikakve zle namere potvrđuje i činjenica da sam Veliku akciju pisao sa namerom da se emituje u radio emisiji „Ventilator 202" u septembru 1984. Program nije emitovan jer me je Zoran Modli, smatrajući da je šteta da toliki trud ode samo u etar nagovorio da ga pošaljem na konkurs. Posle otprilike mesec dana došlo je do jedne smešne zgode. Pojavila se

➔

grupa pirata sa svojim potpisima na Velikoj akciji, sa željom da je emituju u istoj emisiji. Dakle,

a) Od maja 1984 ne pripadam Computerlandu, niti su oni prodavali moj program, mada ga imaju u katalogu.

b) Na programu nisam ostvario nikakvu zaradu.

c) Spectrum od sada koristim isključivo za igru.

Aleksandar Radovanovic

Jelena suzama ne veruje

Možda je postalo hronično da tekstovi vaše saradnice J. Rupnik iritiraju samo muški deo građanstva, ali ovo je prevršilo svaku meru. Mislim konkretno na njen tekst „Obračun kod Tjuringove mašine". U tom tekstu ona, verovatno, pokušava da kaže nešto, ali šta, to mi još uvek nije jasno. Ona javno vređa kompjuterske stručnjake koji se, po njoj, „pojave na TV da diskutuju o kućnim računarima. Čak i relativno neupućeni ljubitelji mogu da primete da ovi pojma nemaju o čemu govore. Propustili su da se prethodno kod lokalnih hakera obaveste o čemu se radi." (????!!)

Dobro, molim vas, na šta ovo liči? Ako smo dotle došli da doktori i magistri nauka treba da uče od hakera, onda kuda ide ovo društvo? Ima još! Na primer: „Oni koji se profesionalno bave računarima ne vole da im se po dvorištu muvaju tamo neki hakeri . . . Oni strepe da se na kraju ne ispostavi da je njihovo čedo-dinosaurus u stvari maloumno".

Iz ovog citata se može videti totalna neupućenost drugarice J. R. u tu problematiku. Ovom, naročito poslednjom rečenicom, verovatno je htela da kaže da veliki sistemi ne mogu biti bolji od mikroračunara. Drugarice Rupnik, hteo bih da vas pitam mogu li uopšte da se porede IBM—370 koji ima petnaestak terminala sa jednim kućnim mikroračunarom, pa makar on bio i vrhunski te vrste, npr. BBC ili QL . . .

Drugarice Jelena, mislim da bi vam stvarno bilo bolje da pođete na mali kurs iz osnova računarske tehnike. Ako ne mislite da učinite tako nešto, zašto bolje ne počnete da pišete o onome šta vrlo dobro znate — a to su igre, kuvanje ručka i slično — ili da se konsultujete sa nekim od kolega saradnika iz redakcije pre nego što počnete da pišete stručan tekst . . .

Uzgred, mogli biste angažovati i Ljubu Moljca da vam učini tekstove malo smešnijim!

Račić Samir
Dragoševa 3/VI
18300 Pirot

Žensko koje piše

Tačno sam znala da će, čim počnete da obavljujete pisma jedna od žrtvi napada biti i Jelena Rupnik. Od kad su počeli da se pojavljuju njeni tekstovi u „Računarima", znala sam da piše drugačije, izazovnije i svežije nego što bi smela. Meni se to jako sviđa, ali sam bila sigurna da će biti čitava gomila uštogljenika i kojekakvih tipova koji neće moći da podnesu njene tekstove, da ne pominjem zadrte muškarčine koje sigurno iritira njen neofeministički ton.

Kako ste vi kao redakcija uopšte mogli da objavite ono pismo Šaše Kovačevica? Ja sam to shvatila kao da vi žrtvujete Jelenu Rupnik i odričete se njenog učešća u vašem radu. To je apsolutno izdajnički. Gotovo sam sigurna da u redakciji sede sve sami muški šovinisti i računarski čistunci koji su jedva dočekali pismo kakvo je Saša Kovačević napisao da bi se odmah ogradili od onog što ih ugrožava, a to je „žensko što piše". Niste mogli na bolji način da potvrdite Jelenine teze iz članka „Računari su muškog roda".

Nadam se da moje pismo nećete shvatiti kao histeričan napad na sve muškarce i da ćete ga objaviti da bi se videlo da ima i nas kojima se Jelenini članci veoma sviđaju.

Ana Trojanovic
Otona Župančica 1
Novi Beograd

Siledžija iz javnih glasila

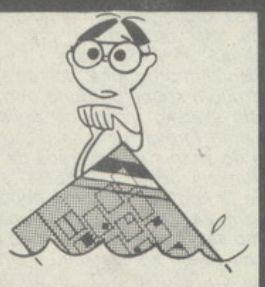
Prilično sam se obradovao kad sam video da ste uveli pisma čitalaca, pardon load pismo, u „Računare". Baš taj prvi broj u kome je ta rubrika objavljena me je i ponukao da napišem ovo pismo.

Pročitao sam pismo koje je napisao Saša D. Kovačević, a koje se tiče pisanja Jelene Rupnik. Na stranu to kako Jelena piše. Ali, ako se dobro sećam, to je onaj isti Saša koji već duže vremena koristi svaku priliku da opali po Dubravki Markovic preko talasa „Indexa 202", čiji je najveći greh što se zove Duca, a bio je izuzetno aktivan i u imitiranju i spominjanju Sonje Savić. Ako se dobro sećam, to je takođe onaj isti Saša koji prilično često gnjavi jadne reportere „Indexa". E, pa mislim da uopšte nije slučajno to što su i Jelena i Dubravka i Sonja i reporterke ženskog roda.

Pretpostavljam da je dosta visoko obrazovanje uskratilo Saši priliku da devojke prepada i plaši po liftovima, pa njegova podsvest to kanališe prema verbalnim i pisanim napadima po raznim sredstvima informisanja, a sad i u vašem časopisu. Zar Saša zaista misli da se to tako radi? I zar ne bi bilo jednostavnije da Saša sazna telefone pomenutih djevojaka i da svoja obrušavanja obavlja u miru privatne veze?

Jelačić Boško
Konjarnik, Beograd

hakeri u nevolji



Ponovo monitori

Tekst o monitorima, objavljen u prošlom broju računara, izgleda da je raspalio strasti u mnogo većoj meri nego što je to redakcija predvidela. Javilo nam se nekoliko desetina vlasnika „spektruma" i „komodora" koji traže precizan odgovor na pitanje koji monitor odgovara njihovoj mašini i kako ga spojiti.

Da krenemo od „spektruma" i zelenog monitora. Problem je, naravno, u tome što „spectrum" nema nikakvog monitorskog izlaza. Ipak, uz malo poznavanja elektronike, moguće je skoro bez ikakvih problema povezati „spektrum" i zeleni monitor. Najčešće je dovoljno uzeti signal ispred FR modulatora. Što se tiče kolor monitora, procedura je potpuno ista, s tim što treba strogo paziti da monitor ima kompozitni ulaz — nikako ne RGB.

Sa „komodorom" ima zntano manje problema. To jest, jedino mogu da smetaju različiti konektori. Zeleni monitor može bilo koji, a kolor mora obavezno da bude kompozitni. RGB, baš kao i kod „spektruma", neće valjati. Ipak, nije sve tako roze ni kod „komodora". Dobićete divnu sliku, ali bez tona. Kod zelenog monitora tu nema leka, dok kod kolor monitora možete da birate: ili ćete kupiti originalni „komodor" po ceni od oko 210 funti, ili ćete naći neki kompozitni koji ima i zvuk. Jedan od takvih je FIDELITY CM 14 12 MHz RGB-/COMP/SOUND, sa cenom od 190 funti.

Važno je primetiti da su „spektrum" i „komodor" računari sa relativno slabom rezolucijom, pa će svaki, i najjeftiniji monitor, odlično odgovarati. Cene zelenih monitora se obično kreću između 70 i 150 funti. Da pomenemo neke: PHILIPS 12" GREEN (90 funti), SANYO SM12N GREEN 15MHz (90 funti), SANYO DM8112 GREEN 18 MHz (130 funti), KAGA 126 (120 funti). Ako vam se više sviđa amber (žuti) monitor, njihove cene su obično nešto malo veće. Recimo: NOVEX AMBER (125 funti), ili KAGA 12A (140 funti). Kolor monitori su, naravno, mnogo skuplji. Za model niže klase treba izdvojiti između 200 i 400 funti. Pored već spomenutog originalnog komodorovog monitora i FIDELITY CM 14, pomenimo još MICRIVITEC 1431 MZ specijalno namenjen „spektrumu" (230 funti).

Nemaskirani interapt

Zoran Jovanović iz Novog Mesta se interesuje za nemaskirani interapt procesora Z80 i za grešku koja se nalazi u „spektrumovoj" rutini za tretiranje nemaskiranog interapta.

Kada procesor Z80 preko nožice NMI primi zahtev za nemaskirani interapt, on prekida svoj redovan posao, stavlja na stek sadržaj PC (PROGRAM COUNTER) registra i skače na adresu 66 heksadekadno. U „spektrumovom „ROM-u, na toj adresi se nalazi sledeći program:

```
RESET     PUSH AF
          PUSH HL
          LD HL, (23728)
          LD A, H
          OR L
          JR NZ, NO-RESET
          JP (HL)
NO-RESET  POP HL
          POP AF
          RETN
```

Program ima jedan vrlo glup bag: umesto JR NZ, trebalo bi da stoji JR Z. Konstruktori su bili predvideli sledeće: uzme se sadržaj sistemske promenjive NMIADD (23728, 23729). Ako je taj sadržaj jednak nuli, ne radi se ništa. Preciznije, ide se na NO-RESET, a odatle se nastavlja izvršavanje redovnog programa. U suprotnom, ako nije nula, skače se na adresu na koju ukazuje NMIADD.

Korist od nemaskiranog interapta može da bude u sledećem: „spektrumu" dodamo jedan specijalan taster za RESET spojen jednim krajem na NMI nožicu procesora, a drugim na masu, u RAM stavimo jednu kratku rutinu za normalizaciju sistema, a u sistem variablu NMADD, adresu te rutine, posle toga, možemo, recimo, da testiramo neki mašinski program. Ako stvari krenu naopako, pritisnemo reset taster, izvrši se rutina za normalizaciju sistema i na kraju operativni sistem preduzima kontrolu. Naravno, sve što je bilo u memoriji ostaje netaknuto. U tome je i sva prednost ovakvog reseta u odnosu na onaj kada pomoću jednog tastera spojimo RESET nožicu sa masom.

Sve bi to lepo radilo da nema onog glupog baga. Da bi se to ispravilo, nema druge nego da ceo sadržaj ROM-a (sa ispravljenom greškom) prekopiramo u jedan 27128 eprom i stavi na mesto ROM-a. Ta ideja je izuzetno dobra, pored ostalog i zbog toga što se u ROM-u nalazi ceo kilobajt praznog prostora. Tu možemo sasvim lepo da stavimo celu rutinu za normalizaciju sistema, pa čak i još nekoliko korisnih programa. Rutinu za normalizaciju sistema možete naći u RAČUNARIMA 1 pod naslovom „Ekskluzivni reset". U sledećem broju „Računara" Petar Putnik, jedan od pobednika „Galaksijinog" konkursa za najboljeg YU programera, pisaće o mogućim prepravkama „spektrumovog" ROM-a.

ZABAVA★UČENJE★POTREBA

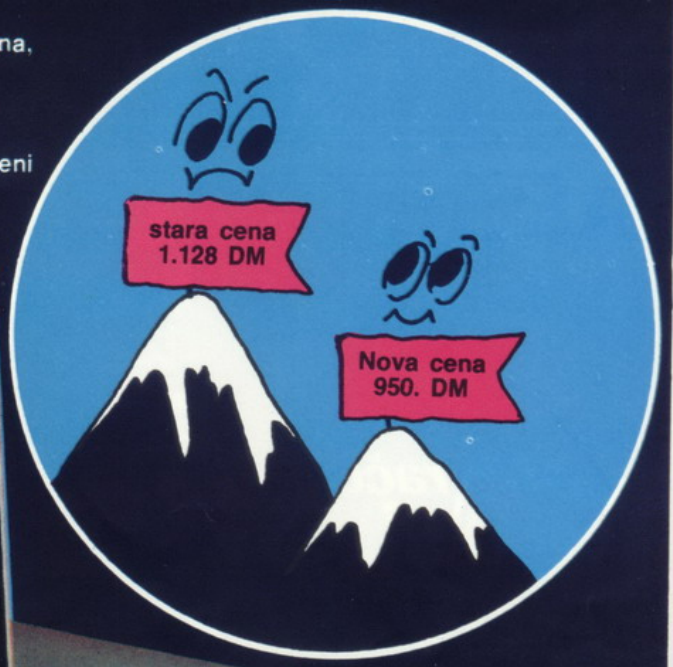
Iz serije 700, predstavlja vam s novom, nižom cenom, kompaktni personal kompjuter SHARP MZ — 731.

Kasetofon — Cetvorobojni štampač (crna, plava, crvena, zelena) — Kapacitet memorija 64 KB RAM.

Uz MZ 731 možete posebno naručiti:

„Zeleni" monitor po ceni od 420 DM ili TV kolor monitor po ceni od 840 DM.

Dinarske dažbine cca 65%.

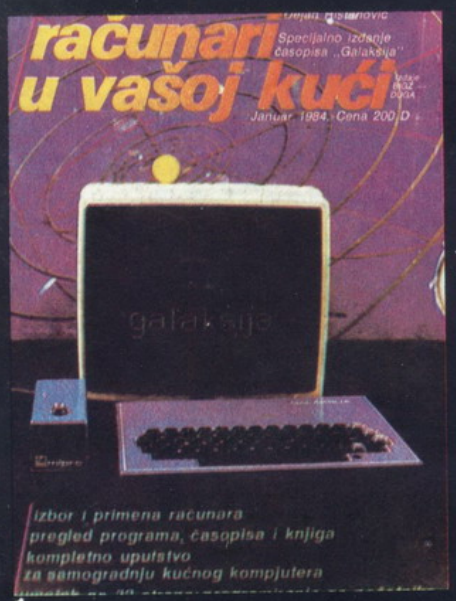


SHARP

Mercator — Mednarodna trgovina

Za sva detaljna obaveštenja obratite se:

TOZD CONTAL
61 000 Ljubljana
Titova 66
Telefon: 061/328 441 lokal 939



1

2

3

4

5

6

NARUDŽBENICA

GALAKSIJA, Bulevar vojvode Mišića 17, 11000 Beograd

Ovim neopozivo naručujem sledeća specijalna izdanja časopisa „Galaksija": 1 — 2 — 3 — 4 — 5 — 6 — 7 (zaokružiti odgovarajući broj), po ceni od 200 dinara po primerku. Iznos od ukupno _____ dinara uplatiću poštaru prilikom preuzimanja pošiljke — POUZEĆEM.

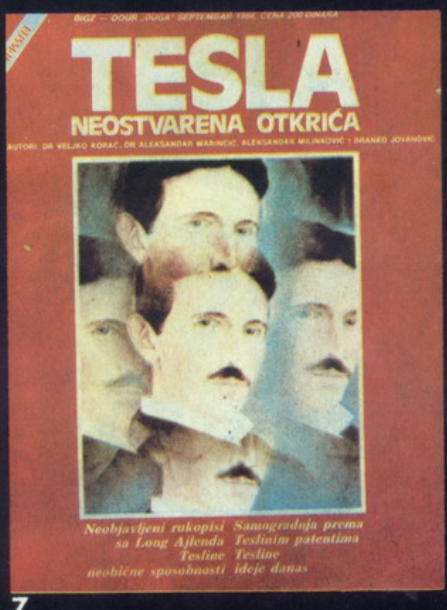
Ime i prezime _____

Ulica i broj _____

Broj pošte i mesto _____

Datum Potpis

1 2 3 4 5 6 7



7