

računar
„galaksija“

R2

ON

Dejan Ristanović

Časopis „Galaksija“
Beograd
Zavod za udžbenike
i nastavna sredstva, Zemun
Elektronika inženjeringu, Beograd

Sistemski softver računara „galaksija“ zauzima, u osnovnoj konfiguraciji, 4 kilobajta memorije. Opremljen njime, računar „galaksija“ predstavlja potpuno zaokruženu celinu. Svaki savremen računar je, međutim, koncipiran tako da ga vlasnik, u zavisnosti od potreba i materijalnih mogućnosti, neprekidno proširuje i tako oprema za rešavanje sve složenijih praktičnih problema. ROM 2 je prvo komercijalno softversko proširenje „galaksije“. Ono je — u saradnji Zavoda za udžbenike i nastavna sredstva Elektronike inženjeringu i časopisa „Galaksija“ kao i sva ostala proširenja besplatno stavljeni na raspolaganje svim graditeljima ovog već pomalo nacionalnog računara.

ROM 2 predstavlja logičnu nadgradnju osnovnog operativnog sistema. On će „naučiti“ vaš računar većini standardnih matematičkih funkcija, proširiti njegov jezik, podržati rad sa portom za proširenja i štampačem koji je eventualno priključen i omogućiti „galaksiji“ da, uz proširenje memorije i hardverske dodatke (emulator i EPROM programator), postane moćan sistem za razvoj softvera za druge mikroprocesorske sklopove koji koriste Z80. Za to je, jasno, posebno značajan kompletan asembler (program za rad sa mašinskim jezikom) koji je ugrađen u ROM 2. One koji planiraju da koriste asembler za razvoj mašinskih programa koji će se izvršavati na samoj „galaksiji“ posebno će obradovati činjenica da je omogućeno postavljanje tačaka prekida programa ('break points') i nekoliko drugih monitorских naredbi.

Opremljena ROM-om 2, „galaksija“ ostaje potpuno kompatibilna sa neproširenim modelima. Iako ROM 2, po inicijalizaciji, deluje na neke sistemske promenljive (posebno linkove), sve do sada objavljene informacije o dodavanju naredbi i sličnim „specijalnim efektima“ ostaju u važnosti.

Kao što je ROM 2 logična nadgradnja ROM-a 1, tako i ovaj tekst predstavlja nastavak 'Uputstva za upotrebu računara „galaksija“'. On će vam omogućiti da iako koristite mogućnosti dodatka koji vam je upravo stigao iz „Galaksije“ i, uz školu mašinca koju objavljujemo u „Računarima“ pomoći da sastavite interesantne programe.

INICIJALIZACIJA ROM-a 2 A=USR (&1000)

Da bi iskoristili potencijale ROM-a 2, posle svakog uključivanja računara treba izvršiti tzv. inicijalizaciju — obavestiti računar da novi čip očekuje njegovu pažnju. Posle izvesnog vremena, inicijalizacija će postati toliko prirodna da je nećete ni primećivati. Ukoliko, međutim, baš ne možete da se naviknete na nju, postoji rešenje koje možete da primenite uz malo truda i hardvera: mala izmena ROM-a 1 koja će automatski inicijalizovati ROM 2 posle svakog uključenja „galaksije“. Od opreme vam je potreban programator EPROM-a i uređaj za njihovo brisanje. Ako vaša „galaksija“ kontroliše EPROM programator, moraćete da pronađete prijatelja koji poseduje „galaksiju“ ili da kupite ili pozajmите jedan prazan EPROM 2732.

Najpre otkucajte SAVE 0,&FFF i snimite na kasetu čitav sadržaj ROM-a 1. Zatim sa OLD 12288 relocirano upišite ovaj sadržaj u RAM tako da počinje od &3000 (ili od neke druge adrese na kojoj softver vašeg programatora EPROM-a pretpostavlja da je smešten bafer koji treba prepisati u EPROM). Sledi izmene: BYTE &33F9, &C4:WORD &33FA, &1000:BYTE &30/37,29 (time ste, zapravo, smestili CALL &1000 na adrese &3F9-&3FB i promenili bajt &37 u koji je upisana verzija ROM-a 1; do sada je verzija bila 28, a sada je 29).

Posle ovih izmena, isprogramirajte prazan EPROM, zamenite njime ROM 1 i isprobajte računar. Ako je sve u redu, možete da izbrišete vaš bivši ROM 1, isprogramirate u njega sadržaj koji imate na traci i vratite ga u računar vašeg

prijatelja. Uključite vaš računar i, dok ste u društvu, prionite na čitanje ostatka ovog uputstva.

Posle svakog uključenja računara, odnosno njegove namerne (PRINT USR(0)) ili slučajne (posle neke vaše grube greške, najčešće pri radu sa asemblerom) reinicijalizacije, otkucajte A=USR(&1000) i pritisnite ENTER. Time je „galaksiji“ stavljen do znanja da je priključen ROM 2 i izvršene neophodne promene područja sistemskih promenljivih. Ukoliko zaboravite ovu inicijalizaciju, neće se dogoditi ništa strašno — normalno ćete raditi sa ROM-om 1 i računar će prijaviti grešku WHAT? kad prvi put pokušate da koristite neku specifičnost ROM-a 2. U tom trenutku možete da otkucate A=USR (&1000) i „galaksija“ će raditi sa drugim ROM-om kao da je on aktiviran u samom početku. Neće doći do gubitka programa ili podataka osim promenljive A. Ukoliko želite da sačuvate i nju, upotrebite A=A+0*USR (&1000).

Čitaoci koji su malo zakasnili sa programiranjem ROM-a 1 pa ga šalju redakciji zajedno sa ROM-om 2, dobijaju verziju 29. Oni ove retke mogu da preskoče.

Zagriženje „hakere“ će interesovati tačan opis onoga što se dešava u toku inicijalizacije ROM-a 2. „Galaksija“, pre svega, briše ekran, što je neophodno za slučaj da je inicijalizacija automatska (u tom slučaju je LD &A, C/RST &20 zamenjeno sa CALL &1000). Zatim se ispravlja bag koji se ispoljava kada je priključeno memorijsko proširenje od 48 Kb; nema, dakle, više potrebe da kucate WORD &2A6A, &FFF0 kad god uključite računar. Posle ovoga, u sistemsku promenljivu 'horizontalna pozicija slike' (&2BA8) upisuje se broj 12; pokazalo se, naime, da je na većini televizora ova promena neophodna; broj 11 odgovara jedino profesionalnim monitorima.

Nakon ispravljanja bagova, menjaju se sadržaji linkova za naredbe (JP &100F) i za video (JF &106F). Ova promena ne bi trebalo da utiče na kompatibilnost propisno napisanog softvera u najgorem slučaju, neki program će isključiti ROM 2 utičući na linkove. Na kraju inicijalizacije ROM-a 2 kontrola se vraća osnovnom operativnom sistemu.

Posle inicijalizacije možete da koristite sve potencijale ROM-a 2 kao da je „galaksija“ sa njim rođena. Naredbe se pišu i skraćuju na uobičajeni način. Možete, na primer, da otkucate DUMP &1000,10 i na ekeranu će se pojaviti prvi $10^8=80$ bajtova ROM-a 2. Umesto DUMP, odgovara i DU. &1000,10 uz iste efekte.

Pošto „galaksija“ opremljena ROM-om 2 ima više naredbi od slova abecede, nije više moguće skratiti svaku od njih na jedno slovo i tačku. Da ste, u gornjem slučaju, otkucali D. &1000, 10, „galaksija“ bi izvršila DOT 4096,10, sa neželjenim efektima. Pri prepoznavanju naredbi, dakle, prioritet ima tablica u ROM-u 1, sledi tablica u RAM-u (ako ste je sastavili) i, na kraju, naredbe ROM-a 2. Kompletan spisak naredbi sa primerima i neophodnim skraćenicama dat je na kraju ovoga teksta.

Da bi se olakšala upotreba češće korišćenih funkcija, neki specijalni znaci su dobili posebne uloge. Ukoliko, na primer, poželite da zabavite prste i otkucate /., MNY, a zatim pritisnete ENTER, „galaksija“ neće prijaviti WHAT? Pogodite zašto! Čestitaćemo vam ako uspete u tome, ali mislimo da nećete — odgovor je na sledećih nekoliko stranica.

ROM 2 se isključuje, svako će reći, tako što se odstrani iz računara! Ovakvo isključivanje ne bismo, naravno, mogli da vam savetujemo (ukoliko je ROM 1 promenjen tako da inicijalizuje ROM 2, ovo rešenje je čak i nemoguće) pa predlažemo drugo, čisto programsко: otkucajte BYTE &2BA9,&C9:BYTE &2BAC&C9 i pritisnite ENTER. Time je ROM 2 skoro sasvim isključen. Odakle ovo „skoro“? Ukoliko se u vašem programu nalazi naredba tipa A=USR(&1000), računar u kome se fizički ne nalazi ROM 2 će kahirati, dok kompjuter sa ROM-om 2 neće pretrpeti nikakve posledice. Potpunije isključenje nije moguće.

MATEMATIČKE FUNKCIJE

Računar „galaksija“ u osnovnoj verziji ima dobru aritmetiku pokretnog zareza koja je potpuno prilagođena njegovim karakteristikama — šest tačnih cifara je sasvim dovoljno za računar te klase. „Galaksiji“, međutim, potpuno nedostaju matematičke funkcije, što za mnoge primene, pogotovu kada se radi o srednjoškolskom obrazovanju, predstavlja vrlo ozbiljan hendikep. Ovaj problem se delimično rešava dodavanjem bejzik potprograma, ali je pravo rešenje tek mašinski rešen set rutina za izračunavanje matematičkih funkcija koji je uklopljen u standardni bejzik. ROM 2 nudi upravo to.

Matematičke funkcije se, naravno, uvek nalaze sa desne strane znaka jednakosti ili iza PRINT naredbe. Iza svake od njih sledi argument koji može da bude konstanta ili brojni izraz; u oba slučaja argument mora da se nalazi u zagradi. Pokušajte, na primer, da otkucate PRINT SQR(4) i pritisnete ENTER. Dobićete 1.99998 — dobrodošli u svet primenjene numeričke analize!

Pre nego što bacite „galaksiju“ u koš i vratite se „digitronu“ koji ste davno sahranili u fioku, pružite nam još jednu šansu i pročitajte sledeći pasus.

Stepene i eksponencijalne funkcije

PRINT SQR (2)

Trigonometrijske, korene, logaritamske i druge funkcije nikako ne mogu da se izraze kao konačan niz sabiranja, oduzimanja, množenja i deljenja, matematičari su se vekovima mučili da ih izračunaju na ovakav način, sve dok nije dokazano da je ovako nešto nemoguće. Danas koristimo kompjutere koji vrše približno izračunavanje vrednosti ovih funkcija preko beskonačnih redova. Ne biste, sigurno, poželeti da vaša „galaksija“ beskonačno dugo radi da bi sabrala beskonačan red, zar ne? „Galaksija“ se zato zaustavlja po sabiranju određenog broja članova reda u trenutku kada je tačnost 'zadovoljavajuća'.

Samo, šta je to 'zadovoljavajuće'. Isprobavajući algoritme matematičkih funkcija, morali smo da pravimo kompromise između utrošenog memorijskog prostora, brzine rada i tačnosti. Smatrali smo da je „galaksiji“ dovoljno 5 do 6 tačnih cifara; za većinu inženjerskih primena dosta bi bile i dve. Problem, međutim, može da nastupi kada poželimo da dobijemo koren koji bi trebao da bude ceo — i osnovci znaju da je $SQR(4)=2$ a ne 1.99998. Nema problema — upotrebimo PRINT INT(SQR(4)+0.5). Ukoliko želimo da zaokružimo rezultat na dve decimale, koristićemo:

PRINT INT(SQR(4)*100+0.5)/100).

Upoznali smo, dakle, funkciju SQR (kvadratni koren) i njena ograničenja. Da nismo malo požurili? Ako ne želimo da već sutra zaboravimo koje funkcije naš računar ima, moraćemo da ih obradimo sistematično i to po grupama.

Svako zna kako se definiše stepenovanje kada je eksponent ceo broj: 2^5 je isto što i $2^2 \cdot 2^2 \cdot 2^2 = 32$. Znamo li šta da radimo kada je eksponent ceo negativan broj? Ništa lakše: 2^{-3} je isto što i $1/(2^2 \cdot 2^2) = 1/8 = 0.125$. Za neke racionalne brojeve problemi su i dalje rešivi: $2^{0.5}$ je isto što i SQR(2) (kvadratni koren). Šta je, međutim, 2^{PI} ili, još gore, PI^{PI} ?

PRINT POW (8,1/3)

Matematički posmatrano, $a \cdot b$ je definisano za sve realne brojeve a i b , pri čemu a mora da bude veće od nule ili, eventualno, jednako nuli (tada b mora biti različito od nule). To znači da $\text{PI} \cdot \text{PI}$ može da se izračuna. Ovo računanje se, najčešće, obavlja primenom logaritama. O logaritmima ćemo reći par reči nešto docnije; dovoljno je da znate da vaša „galaksija“ sasvim lepo ume da računa $a \cdot b$ korišćenjem funkcije POW. POW ima dva argumenta koji se nalaze u zagradi i odvojeni su zarezom. Argumenti su, jasno, a i b , pa ćete $2 \cdot 3$ izračunati tako što ćete otkucati $\text{PRINT POW}(2,3)$ i dobiti 8 ili, da ne budemo prestrogi, bar nešto slično tome.

Posle određenog razmišljanja zaključite da a ne mora uvek da bude negativno da bi $a \cdot b$ bilo definisano. Jasno je, na primer, da je $(-2)^3 = (-2) \cdot (-2) \cdot (-2) = -8$. „Galaksija“, međutim, računa funkciju POW korišćenjem logaritama, pa će $\text{PRINT POW}(-2,3)$ dati grešku. Da ostvarite celobrojno stepenovanje koje radi u svim mogućim slučajevima, moraćete, dakle, da koristite običnu FOR-NEXT petlju za koju ROM 2 nije ni potreban.

Dok nekom prijatelju sa ponosom pokazujete vaš ROM 2 koji je „galaksiju“ naučio matematički, možete da se nađete u velikoj nevolji kada čujete pitanje „a kako se računa kubni koren iz 8?“ Ne brinite, „galaksija“ ume i to uz malo programerske veštine. Treba da se prisjetimo prvog paragrafa ovog poglavlja u kome smo rekli da je $\text{SQR}(4)$ isto što i $4^{0.5} = 4^{(1/2)}$. Slično tome, i kubni koren iz 8 jednak je $8^{(1/3)}$, a sedmi koren iz 1200: $1200^{(1/7)}$. Znajući da argumenti u POW mogu da budu i izrazi, lako ćemo napisati program poput sledećega:

5 ! RAČUNANJE N-TOG KORENA

```
10 PRINT „KOJI BROJ TREBA DA KORENUJEM“ : INPUT A
20 PRINT „KOJI KOREN TREBA DA IZVADIM“ : INPUT B
30 PRINT POW(A, 1/B)
```

Program poput ovoga možemo da koristimo da nađemo kvadratni koren. Zašto je, onda, potrošen prostor u ROM-u za funkciju SQR? Kvadratni koren je u upotrebi mnogo češće od bilo kog drugog, pa ga većina kompjutera odvojeno realizuje. Osim toga, odaćemo vam i jednu malu tajnu: „galaksija“ računa $\text{SQR}(X)$ kao $\text{POW}(X, 0.5)$!

PRINT EXP (1)

„Galaksija“ poseduje i funkciju EXP, koja je, za korisnika koji tek ulazi u svet matematike, specijalan slučaj stepenovanja. $\text{EXP}(X)$ je, naime, isto što i e^X , pri čemu je e matematička konstanta barem jednako važna koliko i PI. Broj e , poput broja PI, ne može da se izrazi kao konačan decimalni broj, niti kao razlomak; kažemo da se radi o 'transcedentnoj' konstanti. Prvih nekoliko cifara broja e su 2.718281828 (može li ovo nekako da se zapamti? Možda i može: u broju e se dva puta ponavlja 1828, što je godina Tolstojevog rođenja. Ako znate kada je Tolstoj rođen, znate i broj e . Ako, kao većina „hakera“, znate broj e , možda ćete zaraditi neki poen na času književnosti). $\text{EXP}(X)$ je, dakle, isto što i 2.71828^X .

SQR smo nekako objasnili, ali kome je potrebna funkcija EXP kada već postoji POW? Odgovor će vas možda začuditi: EXP je potreban „galaksiji“; bez ove funkcije se, jednostavno, ne može realizovati POW; e^X je funkcija koja se lako razvija u red i tako izračunava dok ćemo, za nekoliko trenutaka, videti da se $\text{POW}(a,b)$ izračunava kao $\text{EXP}(b \cdot \text{LN}(a))$. U međuvremenu vam ostaje da otkucate $\text{PRINT EXP}(1)$ i tako saznote sa kolikom tačnošću vaš računar „pamti“ konstantu e .

PRINT 20*LN (2.367)

Čitaoci ovih redova, baš kao i njihov autor, verovatno pripadaju generaciji koja ne zna šta su to logaritamske tablice i koja koristi kalkulator umesto šibera. Pa dobro, šta su to logaritmi?

Uopšte govoreći, logaritam je funkcija koja ima dva argumenta, **a** i **b**, i koja se piše u obliku $\log a$. Obzirom da naša štamparija ponekad pravi probleme sa indeksima, unekoliko ćemo promeniti ovu konvenciju i pomenuti logaritam pisati kao $\log(b)$ a. Ovde je sa **a** označen broj čiji se logaritam traži, a sa **b** takozvana „baza logaritma“. Kažemo da je $\log(b)$ a=c ako je $b^c=a$. Logaritmovanje je, dakle, operacija inverzna stepenovanju, baš kao što je deljenje inverzno množenju.

Sve će postati mnogo jasnije kada pogledamo jedan primer. Već smo videli da je $2^3=8$, što znači da je $\log(2) 8=3$. Funkcija \log može da nam posluži da nađemo eksponent (ovde 3) ako nam je data osnova (2) i rezultat (8). To samo po sebi i nije mnogo korisno, ali treba upoznati još neke fascinantne osobine ove funkcije. Jeste li, na primer, znali da je $\log(a*b)=\log a + \log b$? Da bismo, dakle, pomnožili dva broja treba da nađemo logaritam svakoga od njih, saberemo te vrednosti i, kako bi se to nekada reklo, „antilogaritmujemo rezultat“. Znajući da se logaritmi i „antilogaritmi“ lako pronalaze u tablicama i da je lakše sabirati nego množiti, vidimo da logaritmi mogu nečemu i da posluže.

Još bolju primenu su logaritmi nalazili kod stepenovanja. Treba, naime, znati da je $\log(a^b)=b*\log a$, pa ćemo biti u stanju da zamenimo stepenovanje realnih brojeva (čak i π^{π}) množenjem koje, ako želimo da idemo do kraja, možemo zameniti sabiranjem i još jednim logaritmovanjem.

Pažljiv čitalac će primetiti da smo u zadnja dva pasusa izneverili konvenciju koju smo uveli samo trenutak ranije: umesto $\log(b)$ a pisali smo $\log a$. Šta to može da znači i koliko, najzad, argumenata ima funkcija logaritam?

Oblik koji smo najpre napisali je, strogo posmatrajući, pravilniji: logaritam ima dva argumenta, od kojih je jedan baza logaritma. Pokazalo se, međutim, da se u praksi koriste logaritmi sa bazama 10 (LOG ili dekadni logaritam) i e (LN ili prirodni logaritam). Računar „galaksija“ poseduje samo LN kao češće korišćen logaritam.

Šta da radite ako vam nekada zatreba dekadni logaritam ili logaritam za neku drugu bazu? Nema većih problema ako zapamtite formulu $\log(B) A = \ln(A)/\ln(B)$ ili otkucate program poput sledećeg:

```
10 P. „KOJI SE BROJ LOGARITMUJE“;
20 INPUT A
30 P. „KOJA JE BAZA“;
40 INPUT B
50 C=LN(A)/LN(B)
60 PRINT „LOG (“;B;“)“;A; “=”; C
```

Ostalo je da kažemo još par reči o „antilogaritmovanju“ i da izložimo jedno ograničenje matematičke prirode. EXP je funkcija inverzna LN, pa bi PRINT EXP(LN(X)), za bilo koje X, trebalo da dà isto to X u granicama računske greške.

Ovo 'svako X' ne treba shvatiti baš bukvalno: argument funkcije LN mora da bude broj veći od nule, što je posledica definicije logaritma i činjenice da je a^b uvek pozitivan broj.

```
PRINT SIN (2*PI/3)
PRINT SIND (60)
A=TG (PI/2)
PRINT TGD (45)
A=COS(PI/3)
PRINT COSD(30)
```

Trigonometrija... Zvuči zastrašujuće! Bilo bi lako preskočiti ovo poglavlje i preći na mnogo lepši naslov 'proširenja bejzika'. Ipak, obećavamo da poglavlje neće biti teško za čitanje i da ne zahteva neko posebno predznanje. Ako izdržite do kraja, bićete u stanju da, na primer, sastavite program koji na ekranu crta vaš bioritam!

Trigonometrija je prilično stara geometrijska disciplina koja se bavi rešavanjem trouglova. Pod rešavanjem trouglova podrazumevamo računanje svih stranica i uglova jednog trougla kada su neki njegovi elementi poznati. Da bi trougao bio potpuno određen, treba poznavati tri njegova elementa. To ne mogu da budu tri ugla (jer je, znajući da je zbir uglova u trouglu 180 stepeni, treći ugao određen čim su poznata dva) niti dve stranice i ugao naspram manje od njih (jer tada mogu da se konstruišu dva trougla koji zadovoljavaju uglove); bilo koja druga kombinacija je dobra. Pokušajte, na primer, da izračunate uglove trougla čije su stranice 10, 20 i 25 cm. Ne ide, zar ne? Videćete da se problem rešava sasvim jednostavno, uz elementarno poznavanje trigonometrije.

Pokušaćemo, najpre, da „rešimo“ pravougli trougao sa prve slike. Jedan od uglova pravouglog trougla je poznat (90 stepeni) pa ga određuju dva elementa. Ukoliko znate dve stranice, lako ćete da odredite treću znajući Pitagorinu teoremu koja, „hakerski“ iskazana, glasi $C = \sqrt{a^2 + b^2}$. No, kako da odredite uglove trougla? Ili, kako da odredite dve stranice ako je poznata treća i jedan od uglova?

Potrebne su vam formule:

$$\sin A = a/c \quad \cos A = b/c \quad \tan A = a/b$$

Ovde je sa A označen ugao između stranica b i c ; slovo 'alfa' bi možda bolje poslužilo, ali je ovako lakše jer vaša „galaksija“, kao i većina računara, nema grčka slova. Bilo kako bilo, sinus ugla je količnik dužina naspramne katete i hipotenuze, kosinus je količnik dužina nalegle katete i hipotenuze, a tangens količnik dužina naspramne i nalegle katete. Setimo se malopređašnjeg problema. Neka je ugao A 20 stepeni a stranica b 10 cm. Tada je $c = b/\cos A$ i $a = b \cdot \tan A$. Bilo koji džepni kalkulator sa funkcijama će nam reći da je, iz ovih formula, $c = 10.6418$ cm i $a = 3.6397$ cm.

Pre nego što napišemo program koji će rešiti problem, treba da upoznamo novu jedinicu za izražavanje uglova — radijan. Izražavanje uglova u radijanima ima mnoge prednosti u odnosu na izražavanje u stepenima kada se radi o numeričkoj analizi i, posebno, rešavanju transcedentnih jednačina. Za početnika je jednostavnije da uglove izražava u stepenima, jer je na njih navikao. „Galaksija“ omogućava obe jedinice: ukoliko je ugao izražen u radijanima, njegov sinus ćemo računati sa $X = \text{SIN}(A)$. Ukoliko je A u stepenima, koristićemo

$X = \text{SIND}(A)$ (D dolazi od reči „degree”, stepen). Na raspolaganju su nam i funkcije CO8, COSD, TG i TGD. Ukoliko nam nekada zatreba da izvršimo konverziju manualno, 180 stepeni je isto što i PI radijana. „Galaksija” može da nam pomogne pri ovoj konverziji — ROM 2 dodaje pseudo promenljivu PI koja ima vrednost ove poznate konstante. Prvih desetak decimala broja PI je 3.141592654, a vi otkucajte PRINT PI da vidite kako ga „galaksija” pamti.

Sledeći program ilustruje neke osnovne trigonometrijske relacije na „galaksiji”:

```
10 PRINT „UNESI KATETU B“; : I. B
20 PRINT „UNESI UGAO A“; : I. A
30 C=B/COSD(A)
40 PRINT „HIPOTENUZA JE“; C
50 Z=B*TGD(A)
60 PRINT „KATETA A JE“; Z
70 Y=90-A
80 PRINT „UGAO B JE“; Y
```

Znanje koje smo stekli nam i dalje ne omogućava da rešimo svaki pravougli trougao. Ponekad ćemo, naime, iz formula koje imamo izračunati sin, cos ili tg nekog ugla i poželeti da odredimo sam taj ugao. Za to nam služe inverzne trigonometrijske funkcije arcsin, arccos i arctg (čita se 'arkus sinus', 'arkus kosinus' odnosno 'arkus tangens'). PRINT TG(ARCTG(X)), u granicama računske greške, daje X za bilo koju vrednost promenljive X.

PRINT ARCTG (PI/4)

Poput većine računara, „galaksija” od inverznih trigonometrijskih funkcija ima jedino arkus tangens. Razlog leži u tome što je izuzetno jednostavno naći arkus sinus i arkus kosinus nekog ugla ako znamo njegov arkus tangens. Važi, naime, $\text{arcsin } x = \text{arctg} (x/\sqrt{1-x^2})$, odnosno $\text{arccos } x = \text{arctg} (\sqrt{1-x^2}/x)$. Vama ostavljamo da napišete program koji bi ovo potvrdio; za kontrolu će vam poslužiti neki kalkulator ili urođeni osećaj za proporcije u trouglu.

Dva upozorenja vezana za inverzne trigonometrijske funkcije: ARCTG vraća ugao u radijanima. Ako želite rezultat u stepenima, iskoristite PRINT ARCTG(X)*180/Pi. Obzirom na opšte poznatu činjenicu da je bilo koja kateta u trouglu kraća od hipotenuze, sinus i kosinus ugla su uvek manji od jedan (za tangens ovo ograničenje ne važi). Ako pokušate da simulirate arcsin x, gde je x veće od 1, dobićete poruku o grešci jer će računar pokušati da nađe koren negativnog broja ($1-x^2$ je manje od nule ako je x veće od jedan).

Došli smo, najzad, i do rešavanja bilo kog trougla. Za to nam služe sinusna ($a/\text{SIN}(A) = b/\text{SIN}(B) = c/\text{SIN}(C)$) i kosinusna ($a^2 = b^2 + c^2 - 2 \cdot b \cdot c \cdot \text{COS}(A)$) teorema. Iako su relacije jednostavne, predložili bismo vam da zavirite u neki udžbenik srednjoškolske geometrije pre nego što ih примените; ne možete, na primer, da očekujete da nađete uglove trougla čije su stranice 10, 20 i 100 cm jer takav trougao, jednostavno, ne može da postoji! Postoje mnoga ograničenja poput ovoga; ukoliko ih ne poznajete, vaša „galaksija” će stalno ispisivati WHAT? i HOW? iako je program ispravan.

Funkcija ABS

PRINT ABS (100) PRINT SQR (ABS(A))

Uspeli smo da grupišemo matematičke funkcije u nekoliko poglavija, ali je najjednostavnija od njih ostala sasvim usamljena. ABS (x) daje, naime, apsolutnu

vrednost ili modul argumenta. PRINT ABS (10.25) i PRINT ABS(-10.25) daju isti rezultat: 10.25. Dalji komentar teško da je potreban.

PROŠIRENJE BEJZIKA

Bejzik interpretator koji je ugrađen u ROM 1 je pisan sa težnjom da zauzme što manje memoriskog prostora. Zbog toga su korisnici „galaksije“ lišeni određenog broja standradnih naredbi i funkcija. Uložen je, međutim, maksimalan trud da se ostave one naredbe koje će omogućiti veštrom programeru da sintetizuje sve bez čega je ostavljen. Dodatkom ROM-a 2 situacija se unekoliko popravlja ali su i njegova 4 kilobajta bila jedva dovoljna za asembler i brojne matematičke funkcije. Ponovo smo bili u situaciji da biramo bejzik dodatke koji će biti od najveće moguće pomoći, ali pre svega onima koji žele da pišu mašinske programe.

Brisanje segmenta i prenumeracija

DEL 100,200

Mašinski programi se, u odnosu na odgovarajuće bejzik rutine, sastoje od velikog broja kratkih naredbi. Obzirom da, kao što ćemo videti, svaka programska linija sme da sadrži samo jednu asemblersku instrukciju, očekujemo programe sa mnogo linija. Ukoliko poželite da obrišete veći segment ovakvog programa, naći ćete se u velikoj nevolji. Lepo je što znate da brojite od 1000 do 10000 u intervalima po 10, ali nije praktično da to znanje potvrđujete previše često! Nikakva programerska „caka“, pa čak ni normalna upotreba naredbi BYTE i WORD, ne može da obriše veliki segment linija. To je dovoljan razlog da u ROM-u 2 podržimo naredbu DEL (DELETE=obriši).

DEL ima dva argumenta koji se razdvajaju zarezima. DEL 1000,10000 će, na primer, obrisati sve linije između 1000 i 10000 uključujući i ove dve. Napominjemo da je DEL destruktivna funkcija — ne postoji apsolutno nikakav način da povratite linije obrisanе pomoću nje. Pre nego što pritisnete ENTER, treba, dakle, da dobro proverite granice i uverite se da u okviru neke od njih ne postoji višak ili manjak cifre (100 umesto 1000 će vas „zaviti u crno“!). Da bi se bar unekoliko umanjio rizik koji nosi ova naredba, „galaksija“ strogo proverava njenu sinteksu: ukoliko ma koja od dve linije pomenute u DEL ne postoji, biva prijavljeno HOW? bez ikakvih loših posledica po program u memoriji.

REN 100

Programiranju u asembleru namenjena je i naredba REN (RENUMBER=prenumeriši). U asemblerim programima se, naime, koriste samo labele, pa linijski brojevi nisu mnogo bitni. Ovi brojevi, međutim, ponekad mogu da budu i velika smetnja: pišući mašinski program, često ćemo poželeti da umetnemo pedeset instrukcija između dvadesete i dvadeset pete linije!

REN 100 će nam pomoći u takvim situacijama: ova naredba vrši prenumeraciju celog programa tako da linijski brojevi budu 100, 200, 300... Umetnemo da REN 100, možemo da koristimo REN 10 (linije će biti 10, 20, 30...), REN 50 (50, 100, 150...), REN 72 ili nešto slično.

Naredba REN ne sme da se koristi za prenumeraciju bejzik programa. Za ove programe su linijski brojevi veoma bitni: ukoliko je postojala naredba GOTO 125, a linija 125 posle prenumeracije dobije obeležje 500, GOTO 125 će dovesti do greške HOW? ili do potpuno pogrešnog izvršavanja programa. REN će, dakle, pretvoriti bejzik program koji dobro radi u galimatijas, osim ako se odlučite da ručno promenite sve GOTO i CALL i neke TAKE naredbe i tako ih uskladite sa novim brojevima. Obrzirom da „galaksija“ nema naredbu AUTO, koja bi olakšala kucanje programa sa „okruglim“ linijskim brojevima, ne čini nam se da je trud utrošen na prenumeraciju dužeg programa (čak i kad on treba da bude objavljen u nekom časopisu) uopšte isplativ!

Potrudili smo se da vas na najbolji mogući način zaštитimo od slučajnog kucanja naredbe REN. Pre svega, „galaksija“ interpretira R. kao RUN a RE. kao RETURN. čak i ako otkucate REN, računar će prijaviti grešku ukoliko ne sledi propisan broj (normalno bi bilo da se, ako broj nije naveden, podrazumeva 10). Kada, međutim, otkucate REN n i pritisnete ENTER, povratka na staro nema — najbolje je, zato, da program prethodno snimite na kasetu.

Treba pomenuti još jednu „sitnicu“: linijski brojevi naredbi mogu da budu najviše 32767; ukoliko upotrebite REN ne uvažavajući ovo ograničenje (REN 250 kod programa koji ima nekih 140 linija) dobićete program na čijem su kraju linije sa negativnim brojevima! U ovakovom slučaju možete da spasete program: jednostavno ponovo otkucajte naredbu REN sa nekim manjim brojem iza nje.

Konvencija o navodnicima

A = „X“

Navodnici se u bejziku koriste kao graničnici alfanumerika. Nema, zato, smisla napisati A=10+ „X“ jer broj i string ne smeju da se sabiraju. Ukoliko, međutim, otkucate ovakvu naredbu kada je u vašoj „galaksiji“ ROM 2, neće se pojaviti nikakvo WHAT?; računar će prikazati uobičajeno READY. Kolika je vrednost promenljive A? Otkucaćemo PRINT A i dobiti 98. Stvari počinju da bivaju jasnije: ASCII kod slova X je 88 a „galaksija“ mu je dodala 10.

Prilikom izračunavanja nekog aritmetičkog izraza možete, umesto broja ili funkcije, iskoristiti navodnike između kojih se nalazi neki string. Funkcija će vratiti ASCII kod prvog znaka u tom stringu. Umesto da, na primer, kucate CP 88 u nekom mašinskom programu, napisate CP „X“. Na taj način ćete najpre sebi olakšati posao, a zatim omogućiti nekom drugom da razume čemu ta naredba služi: vi, zapravo, ne poredite akumulator sa 88, već ASCII kodom slova X, — zašto da to i ne naglasite?

Sličnom logikom možete da se vodite i pri pisanju bejzik programa: BYTE &29FF „A“ će ispisati na samom kraju ekranu slovo A, ne izazivajući pomeranje slike (ovo iz programa ne može da se postigne ni na koji drugi način).

Problem može da nastupi ako želite jednostavno da znate ASCII kod nekog znaka. Ne smete da otkucate PRINT „X“, jer ćete na ekranu dobiti obično slovo X. Jedno od rešenja je da otkucate A= „X“: PRINT A ali postoje i dva duhovitija:

Otkucajte PRINT 0+ „X“ i pritisnite ENTER. „Galaksija“ će, analizirajući uneti izraz, najpre detektovati nulu kao brojni podatak. Odatle će izvesti zaključak da se ne očekuje alfanumerik i pozvati sistemski potprogram za izračunavanje vrednosti izraza. Ovaj potprogram će, jasno, znati da je 0+ „X“ isto što i 0+88 i prikazati na ekranu ASCII kod slova X.

Drugi način je još lakši — otkucaćemo PRINT % „X“ i dobiti &0058. ASCII kod slova X je, dakle, &58 što je, kada se prevede, isto što i 88 dekadno. Dobra ili loša (zavisi kako posmatrate stvar) strana ovoga metoda je, dakle, što ASCII kod biva izražen heksadecimalno. Potrebno je, ipak, da malo bolje objasnimo znak % koji smo upravo upotrebili: i on je jedna od novina koje donosi ROM 2.

Prikazivanje heksadecimalnih brojeva

PRINT% (&2C3A)

PRINT &3F će, kao što znamo, dati prevod heksadecimalnog broja 3F u decimalan zapis. Kada je god u nekom izrazu moguće napisati decimalan broj, možete da napišete &i neku heksadecimalnu konstantu (svako pravilo ima po neki izuzetak, pa tako i ovo: heksadecimalna konstanta ne sme da se navodi iza NEW i OLD). Obrnuta konverzija, međutim, nije moguća na standardnoj „galaksiji“. Dok radite sa mašinskim jezikom, često ćete poželeti da je primenite, jer većina tabela operiše sa heksadecimalnim brojevima. Zato je znak za procenat (%) rezervisan za prevodenje decimalnih brojeva u heksadecimalne. Jasno je da ovaj znak može da ima smisla jedino kod PRINT naredbe.

PRINT % izraz će, dakle, izračunati (izraz) i pretvoriti ga u četvorocifreni heksadecimalni broj koji će biti prikazan na ekranu uz vodeće & kako bi se izbegla zabuna. Neke od vodećih cifara mogu da budu nule, i to ponajviše zbog „galaksijine“ konvencije o celim brojevima.

Sa korisničke tačke gledišta, „galaksija“ ne operiše sa celim brojevima: sve promenljive čuvaju brojeve u takozvanom pokretnom zarezu. Za interne potrebe „galaksija“, međutim, koristi cele brojeve prikazane binarno u potpunom komplementu (ukoliko vam reči „potpuni komplement“ i „pokretni zarez“ ništa ne znače, zanemarite ostatak ovoga pasusa). Brojevi manji od &8000 su pozitivni (bit 7 više bajta veće težine resetovan), a konstante veće do &8000 su negativne. Zato će PRINT MEM, ukoliko imate memorijsko proširenje od 48 Kb, dati utisak ne samo da nemate memorije, već i da je nekome dugujete! Ovaj problem rešavate tako što uz ROM 2 kucate PRINT %MEM i interpretirate dobijenu heksadecimalnu vrednost kao broj slobodnih bajtova ili, ako ne želite da imate posla sa heksadecimalnim brojevima, PRINT MEM + 65536 (odakle li je došao broj 65536? To je 2¹⁶ ili 64 Kb). Na sličan način možete da koristite i PRINT %WORD(&xxxx) i tako izbegnete mnoge neprijatne situacije.

Prikazivanje sadržaja memorije

DUMP &1000,212

Često je potrebno prikazati sadržaj niza suksesivnih memorijskih celija. To može da se učini uz pomoć FOR-NEXT petlje, ali je takvo rešenje „sirotinjsko“: sporo je i zahteva mnogo kucanja. Zato je u ROM-u 2 našla mesto naredba DUMP koja ima dva argumenta odvojena zarezom.

DUMP x,n prikazuje sadržaje memorijskih celija počevši od one čija je adresa x. Prikazuje se sadržaj osam celija u svakom redu, pri čemu se na početku svakog reda nalazi adresa njegove prve celije. Biće prikazano n redova (n može da bude najviše 255), tj. dampovano 8*n bajtova počevši od onoga čija je adresa x. Sve konstante se prikazuju heksadecimalno, dok x i n mogu da budu dekadni ili (ako nekom od njih prethodi &) heksadecimalni brojevi, pa čak i izrazi.

Otkucajte, na primer, DUMP &1000,200 i na ekranu će biti prikazano prvi 1600 bajtova ROM-a 2. Cifra će, narevno, biti previše da biste ih pratili (sa razumevanjem tih cifara ćete još malo pričekati — treba da naučite mašinski jezik) pa ćete morati da pritisnete DEL (privremeni prekid prikazivanja) ili BRK (trajni prekid i poruka READY).

Dampovanje memorije se najčešće koristi za traženje nekog dela mašinskog programa. Traženje neke naredbe u bejziku programu je daleko lakše ako pročitate sledeće poglavlje.

Traženje stringa u bejziku

/ „GALAKSIJA“

„Galaksija“ čuva bejzik programe u memoriji na jednostavan i prirodan način — bajt po bajt. Zato nije bilo ni malo teško realizovati funkciju koja će vam pomagati pri traženju nekog teksta u bejzik programu. Ovakva funkcija postaje apsolutna nužnost kada se prede na asembler sa mnoštvom labela.

Želeli smo da se nova funkcija što lakše koristi, pa smo za nju morali da iskoristimo neki nešifrovani specijalni znak. Jedini sloboden je kosa crta (/). Otkucajte, dakle, /PRINT i „galaksija“ će izlistati sve programske linije u kojima se nalazi reč PRINT (ne i one u kojima je P., PR. ili nešto slično); ako ih ima previše, možete da koristite DEL da usporite prikazivanje.

Pomoću / možete da tražite bilo koju programsku naredbu, ime promenljive, konstantu, komentar ili, jednostavno, bilo koji deo bejzik programa. Nije moguće tražiti jedino programske linije, ali takvo traženje i nema smisla; ako znate broj neke linije, možete da iskoristite LIST da je vidite.

PODRŠKA PORTA I ŠTAMPAČA

ROM 2 je softversko proširenje „galaksije“, što znači da je bilo hardverskih i da će ih biti još više. Takva proširenja se priključuju na port za ekspanziju koji je smešten sa zadnje strane računara. Da bi takva proširenja bila na najbolji mogući način kontrolisana, ROM 2 je opremljen naredbama INP i OUT. Pošto je štampač najpotrebnija periferijska jedinica koja pretvara kompjuter u nešto korisno, podržali smo rad sa njim u ROM-u. Ukoliko u ovom trenutku nemate nikakav uređaj koji biste priključili na vašu „galaksiju“, ne smatrazte da je prostor u ROM-u 2 koji opisujemo bačen — vaš računar je donedavno imao i jedno prazno podnožje, pa ste ga popunili ovim ROM-om; na sasvim sličan način ćete uskoro „popuniti“ i port!

Naredbe INP i OUT

X=INP (10) OUT 10,&FF

Mikroprocesor Z80A može da opšti sa periferijskim uređajima na dva načina: preko memoriske i input/output (ulazno/izlazne) mape. Opštenje preko memoriske mape se zasniva na tome da memoriska lokacija &FFFF, na primer, deluje na neki priključeni relej. Tada će BYTE &FFFF,1 uključiti a BYTE &FFFF,0 isključiti uređaj koji je kontrolisan relejom.

Kontrola uređaja uz pomoć memoriske mape nije uvek sjajno rešenje: šta ako imamo pun adresni prostor popunjen ROM-om i RAM-om? Zato su konstruktori mikroprocesora predviđeli još jednu, I/O mapu, koja ima 256 ćelija koje nazivamo portovima. Neki periferijski uređaj može, na primer, da bude konstruisan tako da ga upis broja &FF u port nula uključuje, a broja &00 u isti port isključuje. Komunikacija može da bude i dvosmerna: priključeni uređaj može da saopštava neke informacije koje će kompjuter „čitati“ iz nekog drugog (ili tog istog) porta. EPROM programator, na primer, može da prima od računara sadržaj koji treba programirati ili da izveštava kompjuter o njegovom sadržaju.

Ni jednom od I/O portova ne možete da pridete naredbom BYTE ili WORD. Da biste iz bežika pročitali sadržaj nekog porta, koristite funkciju INP a da biste upisali nešto u port — naredbu OUT.

Funkcija INP ima jedan argument koji se obavezno nalazi u zagradi — broj porta koji je između 0 i 255. PRINT INP(10) će, na primer, prikazati sadržaj desetog porta.

Naredba OUT ima dva argumenta odvojena zarezom. Prvi od njih predstavlja broj porta (0—255), a drugi sadržaj koji treba upisati u port (obzirom da svaki port predstavlja jednu memorisku ćeliju, i ovaj sadržaj mora da bude između 0 i 255 odnosno, što je isto, 0 i &FF). OUT 100,&F0 upisuje konstantu &F0 u port čiji je broj 100. Umesto ovoga oblika, mogli smo da koristimo i OUT &64,&F0 ili OUT 100,240.

Port broj 255 namenjen je radu sa štampačem i treba izbegavati njegove druge primene. Kada je bit 7 ulaza sa porta &FF resetovan, štampač je spreman za prijem sledećeg znaka, a ako je setovan, „galaksija“ treba da sačeka.

Podrška štampača

LPRINT „GALAKSIJA“

Bilo koji standardni matični štampač može da se poveže sa računaram „galaksije“ može preko Centronics interfejsa, koji se dokupljuje ili gradi kao posebna opcija, da se poveže sa bilo kojim standardnim matičnim štampačem. Od tog trenutka možemo da stampamo programe, rezultate njihovog izvršavanja, segmente memorije i asemblerirane rutine. Sve ovo nam omogućavaju naredbe LPRINT, LLIST i LDUMP koje dodaje ROM 2.

LPRINT je, u suštini, isto što i PRINT, s tim što se sadržaj liste prikazuje na štampaču umesto na ekranu. Iza ove naredbe sledi lista koja liči na onu iza „običnog“ PRINT-a, osim u jednoj sitnici: iza LPRINT sme da se nalazi samo jedan element. To može da bude promenljiva, izraz, alfanumerik ili CHR\$, ali nije dozvoljeno navoditi nekoliko podataka koji bi bili razdvojeni zarezom ili tačkom i zarezom. Nije, takođe, dozvoljeno korišćenje naredbe LPRINT AT. Ukoliko vam je ona neophodna, ispišite niz blankova ispred teksta koji treba stampati.

Većina štampača prihvata različite kontrolne kodove koji određuju neke njihove funkcije: tip slova, broj redova na strani, širina papira . . . Ovi kodovi se nalaze u uputstvu za korišćenje vašeg štampača i „galaksija“ će biti srećna da ih pošalje preko LPRINT CHR\$(x). Treba обратити пажњу на мањи бас „galaksijinog“ оперативног система који онемогућава да се пошalje CHR\$(0). Ово је баг уobičajen за Microsoftов бејзик, па сvi štampači dozvoljavaju korišćenje неког другог кода umesto nule sa istim efektom.

Da bi se na štampač preneo sadržaj ekranu na kome je nešto crtano primenom naredbe DOT, potrebno je posedovati printer opremljen takozvanom „blok grafikom“. Kod ovakvog printer-a kodovi 128—192 daju blokove koji odgovaraju организацији „galaksijine“ video memorije. Preciznija obaveštenja o ovoj организацији су већ objavljena i biće ponovljena u okviru uputstva za upotrebu oficijelnog štampača.

LLIST LLIST 1000

Naredba LLIST omogućava prenošenje čitavog programa ili jednog njegovog dela na papir. Upotreba je jednostavna: otkucajte LLIST i pritisnite ENTER. Listanje će trajati sve dok je ENTER pritisnut; njegovo otpuštanje izaziva privremeni, a pritisak na BRK trajan prekid rada štampača. Iza LLIST može da sledi naredbe od koje listanje treba da počne.

LDUMP 0,10

Naredba LDUMP je potpuno identična već opisanoj DUMP, s tim što se sadržaj šalje na printer i na ekran. LDUMP 0,512 će, na primer, ispisati heksadecimalni sadržaj ROM-a 1.

U podršku štampača spada, strogo gledajući, i jedna od opcija asemblera (OPT 4). Ova mogućnost će biti objašnjena nešto docnije.

Ukoliko ste dovoljno nestrpljivi da isprobate sve pomenute naredbe bez štampača, „galaksija“ će se blokirati, ali će je BRK ili RESET vratiti u normalno stanje.

Tretiranje slova Č, Ć, Ž i Š

Set znakova računara „galaksija“ je, kao što znamo, dounesen našim latiničnim slovima. Većina raspoloživih štampača, međutim, ima malo razumevanja za naše probleme ovoga tipa — ASCII kodovi 91—94 koji, u „galaksijinom“ setu, odgovaraju našim slovima, izazivaju štampanje specijalnih simbola, najčešće strelica. Listing programa koji sadrži naša slova bi, dakle, izgledao u najmanju ruku smešno!

Da bi se izbegao ovaj problem, „galaksija“ umesto slova Č i Ć štampa C, umesto Ž štampa Z dok S zamenjuje Š. Ova konvencija može da bude neprijatno ukoliko nabavite štampač sa našim setom karaktera ili promenite EPROM u štampaču. Ukoliko želite da isključite pomenutu pogodnost, otkucajte BYTE &2AAB,1 a ponovo da je uključite — BYTE 2AAB,0.

ASEMBLER

U ROM 2 je ugrađen originalni asembler napisan uz poštovanje svih Zilogovih konvencija i prilagođen specifičnostima „galaksijinog“ operativnog sistema. On omogućava komforno pisanje i ispravljanje mašinskih programa uz podršku

nekih „specijalnih efekata”, kao što je postavljanje prekidnih tačaka u programima.

„Galaksijin“ asembler je organizovan tako da se mašinski programi pišu kao deo bejzika. Time je, najpre, ušteden prostor jer nije bilo neophodno sastavljati novi editor, a zatim olakšano pisanje bejzik programa sa mašinskim potprogramima, što će verovatno najviše interesovati početnike. Da bi se prevazišlo ograničenje nastalo zbog toga što, u ovakvoj organizaciji, nije moguće pisati program koji će se smeštati u svaki segment memorije (npr. programa koji će se upisivati od linka za video), omogućeno je relocirano asembliranje, dopunjeno programom za relocirano snimanje koji smo dali u „Računarima 2“.

Mašinske programe pišete, dakle, poput bejzik programa: svaka linija počinje brojem, sledi (neobavezna) labela iza koje mora da se nađe bar jedan blanko simbol. Zatim pišemo standardnu mnemoničku skraćenicu instrukcije, a iza nje, ukoliko je potrebno, adresni deo. Adresni deo je od instrukcije odvojen bar jednim blanko simbolom. Iza instrukcije sledi (neobavezni) komentar koji počinje uzvičnikom. Dozvoljeno je postojanje linija u kojima se nalazi samo labela, ili samo komentar, ali ne i smeštanje većeg broja instrukcija u jednu liniju.

Evo nekoliko primera korektno napisanih linija:

```
100 SCF
100 LD A, (HL)
100 ! POČETAK PROGRAMA
100 CIKLUS PUSH BC
100 PROVERA
100 ! PROVERA
100 PROVERA CP „A“ ! DA LI JE A?
```

A zatim i nekoliko nepravilnih linija:

SCF	nedostaje broj
100 LDA,\$10	nema blanka između LD i A
100 PRVA PROMENA	blanko u labeli ili komentar bez!
100 SCF:ADC (HL)	dve instrukcije u liniji
100 EX BC, DE	nepostojeća instrukcija
100 1CIKLUS	ime labele ne sme da počinje cifrom

Početak asemblera

Asemblererski programi se, kako rekosmo, uklapaju u bejzik. Da bi rekli računaru kada započinje mašinski potprogram, tj. kada treba da aktivira ROM 2, iskoristićemo programsku liniju na čijem se početku nalazi znak (manje). U tu liniju nećemo više ništa upisivati, tako da se u docnjem listingu lako uoči mašinski potprogram. Od momenta kada „otvorimo asembler“ više ne radimo na bejziku — naredbe PRINT, STOP, IF i slične će izazvati grešku ili biti interpretirane kao labele. Na raspolaganju su nam jedino komande koje će biti opisane u daljem tekstu.

Prva naredba mašinskog segmenta treba da bude ORG (origin, početak). Iza ove naredbe stavljamo adresu (dekadnu ili, ako joj prethodi &, heksadecimalnu) na koju treba da bude smešten mašinski program. ORG &3000 će, na primer, smeštati mašinski program tako da počinje od &3000, pa ćemo docnije moći da ga pozovemo sa A=USR(&3000). Treba, međutim, da pazimo gde smeštamo mašinski program!

Smeštanje na adrese 0—&1FFF (prostor koji zauzimaju ROM 1 i ROM 2) ne može da donese nikakve probleme ali ni koristi: „galaksija“ pokušava da upiše nešto u ROM i u tome, na svu sreću, ne uspeva. Smeštanje programa u prostor latcha i tastature je korisno jedino ako želimo da vidimo naš računar u haosu.

Područje video memorije i sistemskih promenljivih je dostupno ali opasno: video memorija se stalno menja ako zahtevamo listing programa, područje numeričkih varijabli i buffer za tastaturu su dostupni bez opasnosti, dok je ostatak manje-više zabranjen. Programi se obično upisuju od &2C3A ali ne treba zaboraviti da u tom slučaju jezik treba da bude pomeren sa NEW 100, NEW 1000 ili nečim sličnim; u protivnom će rastući mašinski program „pojesti“ svoj izvor sa nepredvidljivim ali uvek vrlo neprijatnim rezultatima.

Smeštanje mašinskog programa na sam kraj memorije je sasvim moguće, ali uz razumevanje jednog ograničenja. „Galaksiji“ je potreban prostor za smeštanje imena labela i njihovih adresa. Za ovo se koristi prostor ispod RAMTOP-a. Ukoliko ovde smeštate mašinac nastaje totalna konfuzija. Osim toga, asembliranje će svakako uništiti određen broj elemenata alfanumeričke matrice X\$(I) i/ili elemenata numeričkog niza A(I).

Problem se rešava na jedan od dva načina. Pre asembliranja možete da promenite RAMTOP (sistemska promenljiva na adresi &2A6A) i tako zaštite važne podatke ili zonu na kraju memorije u koju će mašinac biti smešten.

Drugo rešenje je relocirano upisivanje programa u memoriju. Program se asemblira tako da mu je oridžin, na primer, &3FA0 a upisuje &500 bajtova niže. Po završetku asembliranja ovaj program premeštate na pravo mesto primenom FOR-NEXT petlje ili snimate na kasetu, a zatim relocirano upisujete sa OLD 1280. Pri ovakvoj egzibiciji neophodno je dobro paziti na to da ORG pokazuje adresu na kojoj će se program izvršavati (u našem slučaju &3FA0), dok će se relociranje postići primenom komande OPT.

Komanda OPT

Kao i ORG, OPT je opisna komanda koja ne rezultira smeštanjem nekog bajta u memoriju; ona jedino govori računaru koju od opcija asembliranja da izabere.

OPT ima dva argumenta razdvojena zarezom: OPT n,r. Prvi argument je obavezan a drugi neobavezan: moguće je, dakle, upotrebiti OPT n. Suprotno uobičajenom redu stvari, objasnićemo najpre drugi argument jer je neposredno povezan sa sadržajem prethodnog poglavlja.

Slovo r simbolizuje broj bajtova za koji treba relocirati mašinski program pri upisu u memoriju. Sekvencu:

```
10<
20 ORG &2C3A
30 OPT 3,&1000
40! OSTATAK PROGRAMA
```

koristimo da asembliramo program koji će pri izvršavanju biti smešten od &2C3A, ali će u toku asembliranja biti upisivan od &3C3A (&2C3A + &1000). Argument r može da bude i negativan (npr. OPT 3,—&100 bi upisivalo program 256 bajta pre ORG-a) uz uvažavanje „galaksijine“ predstave brojeva. U granicama o kojima ćemo govoriti u sledećem poglavlju r može da bude i izraz.

Argument n je ceo broj između 0 i 7. Odredićete sa konsultujući sledeću tabelu:

n	Funkcija
1	asembleriski listing se prikazuje na ekranu
2	rezultujući mašinski program se upisuje u memoriju
4	asembleriski listing se ispisuje na štampaču

-
- | | |
|---|--|
| 1 | asembleriski listing se prikazuje na ekranu |
| 2 | rezultujući mašinski program se upisuje u memoriju |
| 4 | asembleriski listing se ispisuje na štampaču |
-

Ukoliko nam je, na primer, potrebno da se kod ne upisuje u memoriju i da se listing ne prikazuje na ekranu (prvo isprobavanje programa da bi se jednostavno ispravile sintaksne greške) upotrebite OPT 0. Ukoliko vam je potrebno da se kod upiše u memoriju uz izdavanje listinga na ekranu (ovu opciju koristite u toku razvoja i testiranja programa), upotrebite OPT 3 (3=1+2). Ukoliko ste finalizovali neki program vratićete se na OPT 2, kako korisnik ne bi bez potrebe gledao listing koji će ga možda zbunjivati. Za vašu upotrebu ćete, uz OPT 5, ispisati program na ekranu i štampaču.

Ukoliko ne navedete OPT, podrazumevaće se OPT 1. To znači da će program biti asembleriran i njegov listing prikazan na ekranu zajedno sa porukama o greškama ali rezultujući kod neće biti upisan u memoriju da ne bi, usled neke greške korisnika, uništio njegov izvorni program. Za stvarnu upotrebu asemblera morate, dakle, da navedete OPT 2 ili OPT 3.

Kako se listing prikazuje? Sa leve strane ekrana vidite adresu instrukcije posmatrano apsolutno (eventualno relociranje se ovde ne prikazuje) u memoriji. Sledi nekoliko bajtova koji predstavljaju kodiranu instrukciju (već smo rekli da neke instrukcije zauzimaju jedan a neke, zajedno sa adresnim delom, čak četiri bajta). Iza toga sledi mnemonik instrukcije i eventualni komentar koji je mogao da pređe i u sledeći red. Labele su posebno istaknute, tako što je ostatak mnemonika uvučen za dva mesta.

„Galaksija“ će, prirodno, prijaviti greške u slučajevima kada neka viša akcija nije bila propisna. WHAT? je standardna poruka koja će se pojaviti kada upotrebite nepostojeću instrukciju, HOW? će nastupiti ako, na primer, upotrebite JR tako da pokazuje na instrukciju koja je izvan domena relativnog skoka a SORRY ako se tabela vrednosti labela „sudari“ sa bejzik programom. U svakom slučaju biva prikazana linija u kojoj je nastupila greška sa upitnikom na njenom početku. Iskoristite standardnu „galaksijinu“ naredbu EDIT ili prekucajte pogrešnu liniju a zatim ponovo startujte program.

Labele i aritmetika

„Galaksijine“ labele su reči proizvoljne dužine (sva slova su značajna tj. labela RACUN se razlikuje od labele RACUNAJ) koje počinju slovom a dalje se sastoje od slova i brojeva. Svakoj od njih se dodeljuje ceo broj uz poštovanje „galaksijinih“ internih konvencija o kojima smo već dosta govorili. Rezervisane reči asemblera ne smeju da se koriste kao imena labela (ako iskoristite labelu HALT računar neće moći da je razlikuje od odgovarajuće mašinske instrukcije) ali o tome ne treba mnogo da brinete: upotrebite labelu koja vam je potrebna a računar će prijaviti grešku ako mu njeni ime ne odgovara. Primenom / lako ćete locirati sve greške i promeniti nazive.

Vrednost može da se dodeli labeli samo jednom u toku asembleriranja i to isključivo na jedan od dva načina: pomoću EQU ili prostim navođenjem na početku neke linije.

Koristeći EQU dodelujemo labeli vrednost neke konstante ili izraza. Opšti oblik je:

„labela, EQU „izraz, (npr. START EQU &2C3A).

Navodeći labelu na početku neke linije dodelujemo joj vrednost koju će registrar PC imati u trenutku kada nađe na to mesto pri docnjem izvršavanju tog programa. U programu:

```
10 <
20 ORG &3000
30 OPT 3
```

40 SCF
50 ULAZ LD A, &10
60 ! OSTATAK PROGRAMA
RUN

labela ULAZ dobija vrednost &3001 (lokacija &3000 je popunjena kodom instrukcije SCF). Labele koje su dobile vrednost na ovaj način se najčešće koriste kod CALL i JUMP naredbi.

Asembler je opremljen potprogramima za obavljanje računskih operacija koje su potrebne pri radu sa mašinskim jezikom. Sva aritmetika je, po prirodi, celobrojna i nije povezana sa aritmetikom koju koristi bejzik; bejzik promenljive nisu raspoložive u asembleru, kao što labele nisu raspoložive u bejziku.

Izračunavanje izraza se vrši sa leva u desno, pri čemu zagrade nisu dozvoljene. U izrazu mogu ravnopravno da se koriste celobrojne konstante i imena labela. Na raspolaganju su sledeće operacije:

- + celobrojno sabiranje
- celobrojno oduzimanje
- || logičko „i“ (AND)
- <n šiftovanje na levo za n mesta
- >n šiftovanje na desno za n mesta

Sabiranje i oduzimanje ne zaslužuju poseban komentar. Minus može da se koristi i kao unarni operator promene znaka, mada je jasno da se umesto -2 uvek može iskoristiti &FFFE.

Logičko množenje (AND) se obavlja bit po bit. Da bismo upoznali njegovo dejstvo, proučićemo primer:

KRAJ EQU &2C3A &1FA0

Prikažimo najpre brojeve &2C3A i &1FA0 binarno. Dobijamo:

&2C3A = 0010 1100 0011 1010
&1FA0 = 0001 1111 1010 0000

KRAJ = 0000 1100 0010 0000
= 0 C 2 0

Labela KRAJ je, dakle, dobila vrednost &0C20. Moramo, međutim, da kažemo da se logičko množenje retko koristi za ovako „opšte“ operacije — njegova slovna primena je pri takozvanom „maskiranju“.

Ponekad će nam, naime, biti potrebno da izdvojimo nekoliko bitova neke vrednosti, a da ostale zanemarimo. Pretpostavimo, na primer, da želimo da labela STRANA dobije vrednost adrese početka strane na kojoj se nalazi labela RADNI (svaka strana memorije ima 256 bajtova). Ukoliko, na primer, RADNI ima vrednost &2DAD, STRANA treba da dobije vrednost &2D00. Potrebno nam je, dakle, da izdvojimo osam signifikantnijih bitova labele RADNI. Za to će nam poslužiti: STRANA EQU RADNI||&FF00

&FF00 će (napišite brojeve binarno i to proverite) zakloniti niži bajt labele RADNI (bilo koji broj logički množen sa nulom daje opet nulu), dok će &FF ostaviti više bajt nepromenjenim.

Šiftovanje ćemo takođe upoznati na primeru:

STRANA EQU RADNI >8

Neka RADNI ima vrednost &2DAD kao u prethodnom primeru. Prikazan binarno ovaj broj glasi:

&2DAD=0010 1101 1010 1101

Šiftovanjem na desno za osam bita dobijamo:
0000 0000 0010 1101 =&002D.

Kada bismo ovaj broj šiftovali levo za osam bita, dobili bismo &2D00 što znači da promenljivoj STRANA vrednost možemo da dodelimo i sa:
STRANA EQU RADNI >8<8.

U izrazima može da se pojavi i znak za dolar (\$) koji ima sasvim specijalnu funkciju — njime je definisana pseudo promenljiva koja ima vrednost koju će imati PC kada program, u toku izvršavanja, dođe na to mesto. Naredba LD A,(\$+5) će dovesti u akumulator sadržaj pet bajta udaljene memorijske lokacije. \$, dakle, donekle odgovara naredbi PTR bez adresnog dela u bejziku.

Asemblerске naredbe *BYTE*, *WORD* i *TEXT*

U okviru asemblerorskog programa ne sme se nalaziti PRINT, IF ili neka druga bejzik konstrukcija. Asembler, međutim, omogućava umetanje nekih naredbi koje Z80 ne bi razumeo. Dve takve smo već upoznali: ORG i OPT. Preostale tri su *BYTE*, *WORD* i *TEXT* (sličnost sa bejzicom je namerna).

Ponekad nam je potrebno da u nekom segmentu mašinskog programa umetnemo neki bajt ili par bajtova koji neće predstavljati kodove neke instrukcije. Potrebno nam je, na primer, da definišemo internu promenljivu SPACE koju ćemo koristiti za privremeno smeštanje sadržaja nekog šesnaestobitnog registra. Na kraju mašinskog programa ćemo, posle poslednjeg RET-a, dodati:
1580 SPACE WORD &2C3A

Time smo definisali dva bajta kojima možemo da se obraćamo sa LD HL,(SPACE), LD (SPACE), DE ili nekom sličnom naredbom. Početna vrednost promenljive SPACE je &2C3A, ali treba обратити pažnju da će ona ovu vrednost imati samo pri prvom startovanju programa; za docnija startovanja ćemo sami morati da je postavimo.

BYTE, *WORD* i *TEXT* se, dakle, koriste za definisanja bajta, reči ili niza bajtova u toku asembleriranja. Sledeći jednostavan program će vam pomoći da shvatite upotrebu ovih naredbi.

```

10
20 ORG &3000
30 OPT 3
40 DFILE EQU &2800
50 LD A, &C
60 RST &20 ! SISTEMSKI POZIV ZA BRISANJE EKRANA
70 LD HL, PORUKA
80 LD DE, DFILE+231: ! KAO PRINT AT 231
90 LD BC, KRAJ—PORUKA
100 LDIR
110 RET ! POVRATAK U BEJZIK
120 PORUKA BYTE " "
130 TEXT „GALAKSIJA“
140 BYTE " "
150 TEXT " 8 K ROM"
160 KRAJ
170 >
180 A=USR(&3000)

```

Otkucajte i izvršite program pa će se na sredini ekrana pojaviti tekst „GALAKSIJA“ 8 K ROM. Primetite da je pojava navodnika u poruci omogućena dvema naredbama *BYTE* u kojima je korišćena konvencija o navodnicima i to na pomalo čudan, ali za računar sasvim prihvatljiv način.

Završetak asemblera

Iza poslednje asemblerske naredbe, kao što smo videli u prethodnom primeru, mogu da slede definicije i početne vrednosti nekih promenljivih, poruke i tabele. Iza poslednje od njih treba dodati programsku liniju u kojoj se nalazi jedino znak $>$ (veće). Znacima $>$ i $<$ je, dakle, oukviren mašinski deo programa. Kada „galaksija“ nađe na $>$ kontrola se vraća bejzik interpretatoru, dok se oridžin i vrednosti svih labela trajno „zaboravljuju“ (ove vrednosti se, istina, ne brišu, tako da bi vešt programer mogao da iskoristi neku od njih).

Iza kraja asemblera može, kao u prethodnom primeru, da se nađe poziv upravo asembliranog programa. Umesto toga, početnici će verovatno radije započinjati osnovni bejzik program koji će docnije i po potrebi pozivati asemblirane potprograme. Kada završite razvoj čitavog programa, sačuvajte izvorni oblik, a zatim, pomoću DEL, obrišite čitav asemblerски deo. Mašinske potprograme pripremite tako da se nalaze ispred bejzika, a zatim snimite samo ono što je neophodno za izvršavanje programa. Na taj način će i oni koji nemaju ROM 2 moći da koriste vaš program. Pored toga, samo će zauzimati manje memorije, pa će se brže učitavati sa kasete.

Postavljanje prekidnih tačaka

U toku razvoja mašinski potprogram neće, jasno, proraditi „iz prve“. Moraćete, dakle, da radite na njegovim ispravkama i modifikacijama. U takvim slučajevima, biće vam neophodno da „zavirite“ u sadržaj registara i flegova u nekoj fazi izvršavanja programa. Za to vam služi naredba REG.

Postavite REG u neki mašinski program i izvršite ga. Kada mikroprocesor nađe na REG, prikazaće izveštaj koji se sastoji od sledećih elemenata:

AF	BC	DE	HL	1x	SP
AF'	BC'	DE'	HL'	IY	(SP)

Vidimo, dakle, sadržaje osnovnih i alternativnih registara, oba indeks registra, vrednost stek pointera i poslednji broj koji je stavljen na stek. Računar čeka da pritisnemo bilo koji taster (osim BRK, DEL i LIST) da bi nastavio izvršavanje našeg mašinskog programa.

Iza REG možemo da napišemo dva argumenta razdvojena zarezom. U tom slučaju, dobijamo i dva segmenta memorije koji nas interesuje. Brojevi iza REG predstavljaju početnu adresu bloka i broj redova koje treba prikazati — baš kao kod naredbi DUMP i LDUMP.

Dejan Ristanović

Ulagne adrese rutina u ROM-u

U sledećoj tabeli su date ulazne adrese važnih potprograma operativnog sistema i ROM-a 2. Razlike u odnosu na tabelu objavljenu u „Računarima u vašoj kući broj 2“ se svode na dodavanje ulaznih adresa matematičkih funkcija. Kod svih trigonometrijskih funkcija i kod POW po pozivu potprograma čija je ulazna adresa data DE treba da pokazuje ASCII string u memoriji koji predstavlja argument u zagradi. Za ostale funkcije argument treba da se nađe na aritmetičkom steku.

SISTEMSKE ADRESE

adresa	broj bajtova	sadržaj
2800	512	video memorija
2A00	104	promenljive A-Z
2A68	2	pozicija kurzora u memoriji
2A6A	2	kraj memorije (RAMTOP)
2A6C	2	broj HOME zaštićenih bajta video memorije
2A6E	2	izlazni kriterijum FOR-NEXT petlje
2170	16	alfanumerička promenljiva X\$
2A80	16	alfanumerička promenljiva Y\$
2A91	2	STEP za tekuću petlju
2A93	2	pozicija tekuće linije u FOR-NEXT
2A95	2	bejzik pointer u toku CALL i FOR-NEXT
2A97	2	lokacija od koje treba dampaovati
2A99	2	$16^*ARR\$ + 16$
2A9B	2	adresa NEXT promenljive
2A9D	2	pointer za TAKE
2A9F	2	pozicija linije koja se izvršava
2AA1	2	koristi se u toku FOR-NEXT
2AA3	2	privremeni SP (kod CALL)
2AA5	2	diferencijator za tastaturu
2AA7	3	„seed“ za RND
2AAA	1	koji prolaz kroz asembler (1. ili 2.)
2AAB	1	Č, Č, Ž, S postaju C, C, Z, S za bit 7=0
2AAC	124	aritmetički akumulatori i stek Z80A
2BA8	1	horizontalna pozicija teksta (inic. &C)
2BA9	3	link za naredbe
2BAC	3	link za video
2BAF	1	sat radi ako je bit 7 setovan
2BB0	1	brojač za pomeranje slike
2BB1	1	fleg za pomeranje slike
2BB2	1	koliko redova treba dampaovati
2BB3	1	OPT pri asembliranju
2BB4	1	registar za REPT
2BB5	1	fleg za štampač
2BB6	125	bafer (koristi se pri padu editora, kod bejzik naredbe INPUT i za vreme asembliranja)
2C36	2	početak bejzik programa (početno &2C3A)
2C38	2	kraj bejzik programa

Gde i kako asembler upisuje labele

U trenutku kada asembler završi posao, sve labele bivaju zaboravljene. Ponekad, međutim, može da bude korisno da bejzik program odredi gde se nalazi neki mašinski potprogram označen labelom.

Asembler upisuje labele počevši od kraja memorije. Najpre je upisano ime prve labele kao niz ASCII karaktera, od kojih je poslednjem setovan sedmi bit. Iza (tačnije ispred, jer se sve upisuje od viših adresa prema početku memorije) imena sledi dva bajta koja predstavljaju broj dodeljen labeli. Iza toga, bez ikakvog posebnog terminatora, sledi sledeća labela, zatim sledeća... Kraj tabele označen je bajtom nula.

SISTEMSKE ADRESE ROM-a 2

Ovaj dodatak nije namenjen početnicima u njemu, uz minimum teksta i podrazumevajući značajno predznanje, navodimo informacije koje će vam pomoći da u potpunosti iskoristite vaš ROM 2.

U sledećoj tabeli je data mapa sistemskih promenljivih „galaksijinog“ operativnog sistema, bejzik interpretatora i ROM-a 2. Tabela je u mnogome slična onoj koja je objavljena, na strani 26. Uputstva za upotrebu vašeg računara. Pažljiviji čitalac će, međutim, primetiti da su ispravljene dve nebitne greške i dodate nove sistemske promenljive koje koristi ROM 2. Prostor „ispod“ bejzika je, dakle, sada potpuno popunjeno!

Opis	CALL &
Izraz sa DE — HL	008
Izraz u zagradi DE—HL	A6A
Preskoči blankove	104
Ako je (DE) zarez, kao RST &8 inače WHAT?	5
KEY(0)	CF5
Karakter A na ekran	20
HL na ekran kao ASCII	8FB
Alfanumerik adresiran sa DE na ekran	937
Lociraj varijablu čije ime po- kazuje DE. PTR u HL uz C i Z	125
Sledeća naredba	30
Slobodna memorija ARR\$- RAMTOP u HL	183
Element izraza sa DE—HL	18D7
Ceo izraz u HL	1866
Pokaži sve registre na ekranu	1978
DUMP od HL za A redova	19DE
Link za naredbe	100F
Link za video	106F
Stanje aritmetičkog steka	8F6
Izraz sa DE na ar. stek	AB2
HL na aritmetički stek	ABC
Četiri bajta adresirana sa HL na arit. stek	A45
Broj sa aritm. steka u HL	A6D
Broj sa aritm. steka u četiri bajta adres. sa HL	73B
Izraz u zagradi sa DE — (IX)	781
+	B32
-	B1E
*	AE6
/	AF7
SIN	1C30
COS	1C24
TAN	1C04

39/ rom 2	ARCTG	1D81
	SQR	1D09
	POW	1CDA
	EXP	1D1B
	LN	1B6B
	ABS	1BFC
	RND na stek	C8F
	Poređenje elemenata sa aritm. steka (Z,C)	B10

1

NAREDBE I FUNKCIJE ROM-a 2

naredba	skraćenica	primer
„		A=„Z“
%		PRINT %(&2C3A + 517)
/		/LABELA
<		početak mašinskog programa
>		kraj mašinskog programa
ABS		PRINT SQR (ABS(A))
ARCTG	AR.	PRINT ARCTG(PI/4)
COS		A=COS(PI/3)
		PRINT COSD(30)
DEL		DEL 100, 200
DUMP	DU.	DUMP &1000,212
EXP		PRINT EXP(1)
INP		X=INP(10)
LDUMP	LD.	LDUMP 0,10
LLIST	LL.	LLIST
		LLIST 1000
LN		PRINT 20*LN(2.367)
LPRINT	LP.	LPRINT „GALAKSIJA“
OUT		OUT 10,&FF
PI		PRINT PI
POW		PRINT POW (8,1/3)
REN		REN 100
SIN		PRINT SIN (2*PI/3)
SQR		PRINT SIND(60)
TG		PRINT SQR(2)
		A=TG(PI/2)
		PRINT TGD(45)

```

1 !
2 !
3 !      B I O R I T A M
4 !
5 !
6 !      UPUTSTVO ZA ROM 2
7 !
8 !
10 HOME
20 P.AT165,"*****"
30 P.AT197,"* B I O R I T A M *"
40 P.AT229,"*****"
71 A=KEY(0)
80 RRR$(7)
82 FOR I=0 TO 6:TAKE X$(I):N.I
84 # "SUBOTU", "NEDELJU", "PONEDELJAK", "UTORAK", "SREDU", "CETVRTAK", "PETAK"
100 HOME
110 PRINT "      B I O R I T A M"
120 PRINT "      ====="
130 PRINT " "
150 PRINT "OTKUCRJTE DATUM RODJENJA U OB--"
160 PRINT "LIKU < DAN, MESEC, GODINA >."
165 PRINT
170 INPUT X$
180 CALL 3000
210 CALL 1000
220 G=F/7:CALL 2000
230 PRINT:PRINT "RODJENI STE U ";X$(F-G*7);"."
240 PRINT
245 PRINT "OTKUCAJTE DATUM ZA KOJI SE CR--"
250 PRINT "TA BIORITAM U ISTOM OBliku."
255 PRINT
260 INPUT X$
270 CALL 3000
280 X=F:CALL 1000:F=F-X
285 PRINT
290 PRINT "STARI STE";F;" DANA":IF F>32740 P.,"":PRINT "STO JE PREVISE ZA OVAK
PROGRAM!":G.999:E.P.,""
300 PRINT
310 PRINT "      PRITISNITE RET"
320 IF KEY(0)=13 ELSE G.320
330 HOME
340 FOR I=0 TO 48:DOT 0,I:N.I
350 FOR I=0 TO 64:DOTI,24:N.I
360 TAKE 900
370 FOR J=1 TO 3
380 TAKE A
385 L=0
390 FOR I=F-10 TO F+9
395 FOR K=0 TO 2
400 B=SIND(360*(I+K/3)/A)
401 B=INT(22-B*22+.5)
410 DOT L,B:L=L+1:NEXT K:NEXT I
411 TAKE X:BYTE10271+INT(B/3)*32,X
420 NEXT J
900 #23,70,28,69,33,73
999 A=KEY(0):GOTO 100
1000 F=365*Y+D+31*(M-1)
1010 IF M<3 F=F+INT((Y-1)/4)-INT(.75*(INT(((Y-1)/100)+1))):RET
1020 F=F-INT(.4*M+2.3)+INT(Y/4)-INT(.75*(INT(Y/100)+1)):RET
2000 H=-1
2010 IF G<0 G=32768*H+INT(G+32768):RET:ELSE G=G-32768:H=H+1:GOTO 2010
3000 D=VAL(PTR X$)
3002 I=0
3003 CALL 4000
3010 M=VAL(PTR X$+I)
3011 CALL 4000
3020 Y=VAL(PTR X$+I)
3030 RET
4000 IF BYTE(PTR X$+I)=44 I=I+1:RET:ELSE I=I+1:GOTO 4000

```

