

"galaksija"

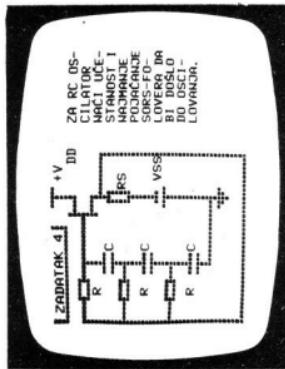
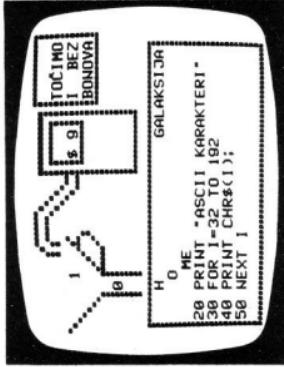
Dejan Ristamovic

REČNIK RAČUNARA "GALAKSIJA"

```

READY BREAK WHAT? HOUR?
SORRY LIST NEW
SAVE OLD EDIT
CALL INPUT IF
INPUT UNDOT RET TAKE
# DOT FOR PRINT
IF DOT 20,10 BYTE WORD
IF DOT 20,10 PRINT "BELO"
EDIT 20 HOME RND MEH
IF B<17 X=1: ELSE GOTO 100 WORD
IF EQ Y$ X$(5) GOTO 100 INT &
FOR I=1 TO 100 EQ
GOTO 100 DOT STEP AT
GOTO 100 Y$ CHR$ TO

```



računar

SPISAK NAREDBI SA PRIMERIMA

```

ARR$ PRINT BYTE(11176)
BYTE BYTE 11176,112
CALL CALL 100
CHR$ PRINT CHR$(34)
DOT DOT *
IF DOT 20,10
IF B<17 X=1: ELSE GOTO 100
IF EQ Y$ X$(5) GOTO 100
FOR I=1 TO 100
GOTO 100
GOTO 100+A+100
HOME HOME 64
HOME HOME 52
IF A>5 GOTO 100
INPUT INPUT A
INPUT INPUT X$
INT A=INT(B/C)
KEY IF E(Y,1) GOTO 100
AKE(Y,0)
LIST LIST 500
PRINT MEM
NEW NEW 256
NEXT NEXT 1
OLD OLD -50
PTR PRINT PTR X$*
PRINT PRINT A+2,B
PRINT PRINT B,C,B1 == ;C
PRINT PRINT A$!A$!A$!A"
RETURN RETURN
RND A=RND
PRINT INT(7+RND+1)
RUN RUN 100
SAVE SAVE 15728,16251
STEP FOR I=100 TO 0 STEP -5
STOP TAKE A,100,B,C,250,X$*
UNDOT UNDOT *
USR A=USR(14)
VAL PRINT PTR X$
WORD WORD 109@5,109@5
PRINT WORD 109@5,109@5
! PROGRAM ZA SUMIRANJE
! #0 TO "JANUAR"
PRINT 12CBA

```

Računar „galaksija“ je, dakle, sastavljen i nalazi se pred vama. Teži

i riskantniji deo posta je time završen — preostalo vam je da

testirate kompjuter i — uživate u plodovima svoga istraživača!

radunar poznate programski jezik bežični sa pravom istom to obezbe

i od sveg vlasnika. Da biste se, dakle, uspešno sporazumeli sa

računarem, morate da uđete u tame ovog jezika što, kao što će te

videti, nije mnogo težak posao. Ovo uputstvo će obrazložiti

sve naredbe koje računar „galaksija“ razume i dati primere njihove

upotrebe. Ono, ipak, nije pisano da bude potpuno samostalno

— u okviru ovog specifičnog izdavača nalazi se osnovna škola

bežička, koju bi stekli vlasnik računara „galaksija“ koji nema iskušnju

sa programiranjem trebalo da pročita pre nego što nastavi sa ovim

tekstom. Primeniteće da su neke naredbe računara „galaksija“

različite od onih koje su pomenute u školi bežička. To ne treba da

vas uplaši — bežički nije standardizovan jezik i razlikuje se od

kompjutera do kompjutera. No, kada se jednom potrudite da shvatite

neki od njegovih dilekata, prilagodavanje nekom drugom računaru

ne prava dečja igra.

Ovo uputstvo je koncipirano tako da vam omogući da pišete programe pre nego što ga popuno pročitate i savladate. Zato su najveći oblici naredbi izloženi na nejednom mestu, dok se deo za bolje poznavanje računara na samom kraju. Onima koji imaju iskušnju sa stolnim računarama preporučujemo da jednostavno pogledaju spisak naredbi i memorijušku mapu i počnu sa radom — na uputstvo će se vratiću kada osećate da potrebujete da se ponovite.

Računar „galaksija“ može da bude povezan sa bilo kojim televizorom koji prima UHF područje. Za to se koristi kojakinjki priključak koji se nalazi sa njegove zadnje strane. Stoga umite odgovarajući kabl i povežite ga sa antenskim ulazom u televizoru. Zatim uključite televizor, prebačavajući ga na UHF program. I laganim okretnjem vratača kanala, potražite da na ekranu dođete poruka READY. Kada je računar spreman da radi, treba da se uživakete EURYKA — računar radi.

Čitaoči koji poseduju monitor ili prepravljeni televizor će potrebiti da povežu računar sa njim, i tako oslobode kuchni TV aparatu. Za to treba koristiti montorski izlaz u kabl sa priključnicom koju zavise od samog monitora.

2. Tastatura

Rakunar „galaksija“ ima 54 tastera koji su raspoređeni poput dirki na pišacu majstora. Kada god želite nek od njih, računar na ekranu ispisuje odgovarajući simbol i to na mestu gde je prethodno napisao karakter. Cim se neki simbol pojavi na ekranu, kurzor se pomeri za jedno mestu udesno da biste ga učinili čitav, a kada se do istog dođe, učinak će biti ikavice koja bi to označila — „računar ne deli redi“ na sljedeće niti obradu pažnju na to da li je neka reč nelogično — „presećena“.

Pritisnite, na primer, tastu A, dvadesetak puta. Zatim će vam postati asno da veliki broj slova A ne znači ništa — da vam se neće potaknuti da ih obrišete. Za to vam niste potrebovali narančko korektorno mastilo — dovoljno je da pritisnete tastu . — kurzor će se pomeriti za

98 C...3800

3800 H...KEY(14)6-2078

3040 H...P...," U OVOJ AKCIJONIJI

06110 T...REBA"

3100 T...V...280 D...32

1880 T...V...280 D...32

1800 T...V...280 D...32

1800 T...V...280 D...32

2100 T...V...280 D...32

3800 H...KEY(14)6-2078

3040 H...P...," U OVOJ AKCIJONIJI

06110 T...REBA"

3100 T...V...280 D...32

1880 T...V...280 D...32

1800 T...V...280 D...32

1800 T...V...280 D...32

2100 T...V...280 D...32

3800 H...KEY(14)6-2078

3040 H...P...," U OVOJ AKCIJONIJI

06110 T...REBA"

3100 T...V...280 D...32

1880 T...V...280 D...32

1800 T...V...280 D...32

1800 T...V...280 D...32

2100 T...V...280 D...32

31. Saradujmo i dajte

Osim smo stigli do kraja uputstva za upotrebu računara "galaksija". Ostalo je, čini nam se, još dosta stvari koje nismo pomenuli, "čaka koje n' mi ne požajemo", naravno, bezroboj ideja za programe. Ako ste sastavili računar, "galaksiju" i naučili da se služite nju, posaljite nam par kratko pismo u kom ćete nas obavestiti o tim činjenicama. "Galaksija" planira da ovom svojoj strancu i posveri redovne rubrike temama vezanim za računar, "galaksiju", kao i da u postojeci katalog bežigraha programu uveri što više dobiti programa za njege, koji će tako biti na raspolaganju svim korisnicima. Čitava ova akcija zavisi u mnogome od svakog citlaca ovih redova: vi ste ti koji treba da napisu programe i da ih posalju na našu adresu. Samo udruženim napornima računar "galaksija" može da bude dopunjeno onim što mu trenutno jedino nedostaje — programskom podrškom.



jedno mesto uveo brišući automatski slovo na čije je mesto došao. Uzastopnim pritiscima na ovaj tasti modeli da obriše sva ukucana slova — sve do početka reda. Na tastavu postoji, išano, i tačka, ali očete se za koji trenutku uveriti da vek računari tu tačku ne smatra krajnjim redenicom. Otkucajmo, na primer, RACINAJ, i stekla nekača. Kompjuter ne reaguje, što znači da nije ni shvatio da mu je izdata neka naredba. U programiranju je mnogo uobičajeno da se na kraju redenice, umesto tačke, pritisne tasti koja je označena sa **ENTER**. Pretpostavimo da ga na akcenatu slovo ne požajemo. Da li će se provesti dočekajem **READY**? Šta se dogodi? Pritisnemo nužno **ENTER** i računar dozvoli da se učini ponizačuvalim — u tom slučaju da pise od njegovog vrha. I primat će se da se učini ponizačuvalim — u tom slučaju da pise od njegovog vrha. Ispisao **WHAT?**, a zatim **READY**, sis ujedno, nam do nazaj **RACINAJ**. Zato je na ekranu ispisano **WHAT?**, a zatim, **READY**, sis ujedno, nam do nazaj **RACINAJ**. Taster **SHIFT**, i ešto, daje, neće biti različit. Ovaj napis ne vredi trazili na tastaturi, odnosno na novu znaku naredbu ili redenicu — ako to ne učinimo, računar će god nemo prepoznamo, ispoljavajući pri tom upomosno oplate preizuzeti naslov.

Ako nastavimo, da pritisnemo **READY**, ekran će ubroj biti popunjeno. Tada će se provesti poslepeteno gubit, a ono što kucamo pojavljuje u poslednjem. Stalno oštreljeva sa poslednjim redom može nekome da se učini ponizačuvalim — u tom slučaju treba da naredimo računaru da obriše sadržaj čitavog ekranu — počne da pise od njegova vrha. Upoznajmo, na primer, tastu **SHIFT**. Ova napis ne vredi trazili na tastaturi, odnosno na dva: neobavezna tastira. Istovremeno pritisnemo na **SHIFT** — i neki broj, dobjavimo na ekranu znak napisanu iznad tog broja (pritisnimo, na primer, islovenemo **SHIFT** 12, i na ekranu će da se pojaviti znak novak). Kod bojeva je, kao što vidimo, simbol napisan u samom tastiru ali postoji neke druge da skrivajući šifrovani funkcijom. Na primer, na primer, istovremeno **SHIFT** — i **DEL** (ukolikom, kao što je predviđeno, pri montaži predviđeli dva **SHIFT** tastira, možemo da koristimo bilo koji od njih) i ekran će biti obrisan. Kada nam, dakte, računare poreke **WHAT?**, koje su izazvane našim početni greškama dosade, prislušnemo **SHIFT** — i obrišati ekran.

Taster **SHIFT** Ce nam otkri i da neka taine računara "galaksija". Pritisnimo, na primer, tastere **SHIFT** i **C**, na ekranu će se pojaviti slobo **C**. Sasvim slično, **SHIFT** S daje slovo **S**, **SHIFT** Z slovo **Z**, a **SHIFT** X zapamtite ga po tome: **X** ćima oblik koji podseća na polovicu ikse. — C. Ostali tasteri nemaju šifrovane funkcije — **SHIFT** A, na primer, daje slovo **A**.

Ako ste, čak i učili na ovaj način eksperimentirati sa kucanjem, verovatno ste, nameravajući da ispisujete gornje isti slova, Taster REPI, da ste desito nato prilučio bušno — računar je počeo da autorizuje (automatsko ponavljanje) — pritisak na njezinsku klavišnu tastu, funkciju poslednjeg pritisnutog tastera. Nameravajući da ponavljava poslednje slovo, otkucavate red sa dosta slova u zatim pritisnete... Ponavljajući slovo, i kao što smo u očekivanju brišemo. No, pritisak na REPI-je provoči dalje efikirne dejstvo, računar će „gutati“ slova dok ne obriše čitav red.

Čak i dok niste započeli da pišete programe, u vekem računaru se nalazio jedan program i to toliko veliki i moćan da čete smeti moći da naprakite tek kad postanete daleko iskusniji programer. To je sistenski program od betiši klobuata koji omogućava računaru „galasku“ da radi ono što mu vi naredite; ta program omogućava provođenje bežijk naredbi koje unosite na oblik pristupajući mašini, izdavanju odgovora na ekran. U jednom velikom programu, kao u metrou, ima mnogo stranica sa kolima mazačima.

3. Demo kaset

BRAINSTORMING napisima **BRK** i **AT/STOP** (u dalejem nastavku **LST**) prihvata se da ovaj raspored ratične i tasterne funkcije na ekranu prikazuju sadržaj programa koji se naziva i njegovim memorijama, a slike učinkovitosti posebne funkcije – konstrukse u igraču i radnju može da kontroliše da je i pribrojeno, prihvata se da ne uemoćimo da premoć programne pa su nam svi ovaj tasteri prebrodimo, ali to neće bitati dugo.

Sa zadnjim stanicama ratičarom se razaznaju dve kategorije koja bi trebalo da pise **RESET**. Ovo predstavlja poljoprivredni verziju tastera **BRK**. Ukoliko vrem da se načini da je se radnju ratičarom, a onda se učinkovitost (ovo se dozvava, ali je ratičar ispravan), samo korišćenjem pisanog programa na mašinskom jeziku (ili članju programu sa kasevom), prihvata se da je i pisanje u 99% slučajevima, izazvati ispisivanje uobičajenog **READY** (pri čemu memorija ratičarom neće biti izbrisana). U preostalom ponovo slučaju morate da isključite i ponovo naredbom ili grupom naredbi radnju ponovno uključujete da ga uvek povratiti.

Pre nego što podamo detalje, moramo da izrekremo i jedno upozorenje vezano za dugme **RESET**, ono nije namenjeno starijoj upotrebi. Pokušavajući primjer, da bzo pritiske taster **RESET** vise puta, pa tako može da se desi da memorija ratičaruna bude kompletno izbrisana. **RESET**, dakle, koristimo samo kao poslednju soluciju, kada nikako ne možete da „probijete“ vaš kompjuter.

Imate li dobre refleks, jedna od mnogo-
naru „galaksija“: *gara prirodnoga raku-*
naru galaksija”.

30. Potprogrammi BOM-3

Mašinski program se unosi tako što se na njegove otkuca NEW 15 (program ima 15 bajtova), a zatim: **81TE&C3A, 801 RET BYTE&C3C, &02 RET BYTE&C3D, &21 RET RD**. Poslednje se kuča BYTET2C47 i FB RET BYTET2C48 i C9 RET. Čime je unosenje mašinskog programa završeno. Da bismo ga isprobali, otvorimo: **10 A=USR &C3A** 20 GOTO 20

4. Komandni i programski mod

Možda će se naci neko ko dalji leksi neće ni čitati — nekoliko gara sa demno-kasete ce mu oduzeti toliko vremena da se smatrat će da je računar ispušto svoju funkciju i opravdito uložen novac. Ipak, učenje programiranja je i tak, priuatan posao i koji se vredi učiti i opustiti.

Pobedimo na najtežostavljivim mogućnostima! Učim, otkucavamo, slovo po slovo, teks **PRINT "RACUNAR GALAKSIJA"**, pristupi **RET** (navigodin se dobijaju istovremeno, pristok na tastere **SHIFT** i **X**). Tako smo načinili računatu na našem ekranu. Tastefni leksi je dobio naziv **SHFT X** — pristupi na **SHFT X**!

Uzimajući u obzir da ponavljamo ovu naredbu već broj puta unesući između znakova **GALAKSIJA**. Možemo da ponavljamo ovu i računatu da bili smo načinili računatu na našem ekranu. Naša naredba se analizira i izvršava onoga momenta kada smo pristupili **RET**, na ovaj način radja naziva **komandni** (na ovaj način računatu izjavljeno komande kojima kontrolisemo njegov rad).

Osim komandnog režima, treba da upoznamo **programski** režim rada. I kod njega računaru izjavljeno komande, ali se ona izvršavaju odmah — računatu ih pamti i izvršavaju ih kada treba da budu naloženo. Da bi kompjuter razlikovao komande od naredbi koje treba da pamti, uvezimo sa brojem programskih linija. Broj programskih linija (ili, kao što se ponazavaju, liniski broj) ukazuje računatu na redosled kojim će kada su bude naloženo, izvršavati "zapamćene" instrukcije. Ovaj broj treba da se nalazi na samom početku linije.

Okućaimo, na primer, **10 PRINT "RACUNAR GALAKSIJA"**; i pristupimo **RET**. Računar nije izvršio naredbu; ona je zapamćena i dodijeljena joj je liniski broj 10. Otkucavamo zatim **20 PRINT "JE MOJ PRVI KOMPJUTER"**, i ponovo pristupimo **RET**. Ozbrzo da je liniski broj drugog naredbe (20) veći od liniskog broja fine (10), računatu će načinjati drugi naredbe. **RACUNAR GALAKSIJA** je tada **je moj prvi kompjuter**. Da proveimo ovu rezonovanju, otkucavamo **RUN** i tovo je, načrt, naredba koju smo dobro upozorili (igrali) se sa demno-kasetom; ona naloži se učitavajući program i pristupi **RET**. Po startovanju programa neizbežno **RET**. Računatu će, kada što se i odrekivalo, izvršiti program i na ekranu ispisati:

RACUNAR GALAKSIJA

JE MOJ PRVI KOMPJUTER

READY

One **READY** nije posledica nake od naredbi našeg programa već uobičajeni signal da je računar spremen da primi sledeću naredbu. To ponovo može da bude **RUN** — program ce se izvršavati onako putom koga smo postavili. Otkucavamo **LIST** — program da pročitaši program putom koga smo postavili. Uzvraćaju naredbe po redosledu njihovih brojeva a ne po redu učenja: otkucavamo **15 PRINT "KONSTRUISAN SEPTEMBRA 1983"**, i pristupimo **RET**. Po startovanju programa (**RUN** i **RET**) na ekranu će biti ispisani tekst:

RACUNAR GALAKSIJA

KONSTRUISAN SEPTEMBRA 1983

READY

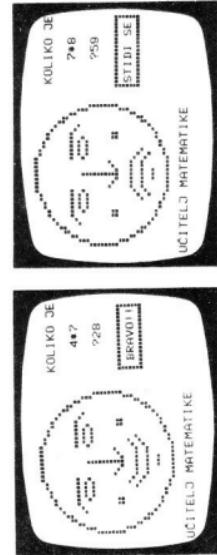
što znači da je linija 15 umetnuta između linija 10 i 20 — baš kao što smo i odrekivali. Ako na ovaj način počnete da umetnete naredbe, ubroj čemo zaboraviti kako program izgleda. Isto je tako pravent: jednostavno pristupimo tastere na kome je ispisano **LIST** (LIST bi slobodno moglo da prevede sa "prikazi program"). Do tastera **LIST** držimo pristup, računat će ispisivati program koji se "prikazi" u njegovoj memoriji, liniju po liniju. Čim postimimo ovaj taster, ekran će se "zamrznuti", pa čemo moći da analiziramo uneseni program. Ako želimo da istačimo naredbu, ponovo ćemo pristupiti tasteru **LIST**, a ukoliko ne želimo pristupiti na **BRK**, ce na ekranu prikazati učitajeno **READY**.

Umete da pristupisate taster **LIST**, možemo da otučamo na **LIST**, i pristupimo **RET**. Sve dok je **RET** pristupit, prikazivanje programa traje, a očitavanje tastera privremeno prekida ovaj proces. Na prvi pogled nema smisla koristiti naredbu **LIST** kada je obvezan taster sa istim delstvom. Ovakav zaključak je samo donekle tačan: ponekad ćemo poželjeti da prikazujemo program počevši od neke linije. U tom slučaju treba da

29. Nove mogućnosti naredbi **SAVE i OLD**

Ako mašinski program unesemo na način izložen u prethodnom poglavljiju, obično **SAVE** će, zaledivo sa bežkom, sumiti i sve što se nalazi u memoriji, poveši od adrese **&2C3A**. Ponekad je, pak, potreban da na kasetu unimo prizvoljan segment memorije. Za to će nam postizati naredba **SAVE** nimmnim, nimm. Da bismo na primer unimmo mašinski program, koji se nalazi na kasetu u bežkom, možemo da kasetofon i pristupiti **RET**. Ukoliko nam je program važan, možemo da izvršimo verifikaciju na uobičajen način.

Za učitavanje korisnim običajem **OLD**, računat će učitati bajlove smestiti na istu adresu učitavajući program. Ponekad će demno, medutim, poveši da istačimo naredbu **OLD** na kojima se načinili prva smetnja. Ponekad će demno, medutim, program započetno odreden broj bajtova pre ili posle mesta sa kojima su smetnji (ovo je pomeranja). Ako, na primer, želimo da mašinski program bude pomeren na **1024** bajtova, otkucavamo **OLD 1024** (ili, što je isto, **OLD \$400**), pristupi **RET** i starati kasetofon. Da je pomeranje trebalo da bude negativno (prema početku memorije) otkucavamo **OLD —1024**. Obrzmo da je ovako pomeranje relativno, svaki put kada stimamo neki mašinski program pomozu **SAVE** mimmnim, nimm, treba da zapšemimo adresu njegove početka i kraja.



Okucamo LISP, a zatim navedemo broj te linije i prilišnemo **RETI** (na primer, LISP 200). Na taj način ne moramo poslušati programu ako nam je potreban njezin kraj. Za potrebu ove, dovolje, nasi programi bili toliko kraki da će bez problema stajati na ekranu pa ne rukuju LISP ostavljanju za blisku budućnost. Na izvrsavanju programa ne mora da počne od početka: možemo, na primer, da otkucamo LISP 15 i prilišnemo **RETI**. Računar će početi da izvršava program od linije 15 i spisati na ekranu.

KONSTRUISAN SEPTEMBRA 1983
JE MOJ PRVI KOMPJUTER

READY

Ova rečenica je: naravno, besmislena. No, za računat nije postojao nikakav razlog da odbije da je prenese na ekran, s obzirom da su sve instrukcije u programu bile valjanije. U pojedinim slučajevima, kada je u programu bilo nešto što nije moglo da se izvrši, program je izjavio da je nešto pogresno, a ne da je nešto nevaljano. Urednik je, nezadovoljan ovim reakcijama, bio, bez sumnje, lepa zabava, ali, da nemoćno, ona dozadati. Kada je potpisnik održao program iz memorije, mogao je isključivo sabrati izvještaj (u izvještaju je bio pristup uključen i što je u sastavu isto potpisnik PRINT USB (O), a pristupom RET). Na raspodjeljenoj kopiji je bio napisan: NEW (postiže nije ne, naravno, priška RET) kada biše samo program, ali ne i sadržaj kromi promenljivih.

5. Galaksiä ja kalkkulaatiori

Naredite PRINT koje smo do sada korisili su načinjene računaru da tekst otiskan između navodnika jednostavno prenese na ekran. No, ova naredba može da obavi i mnogo složnijeg ulogu. Otkucavajući na primer, PRINT 17+13! pristisnemo RET. Na ekranu će se pojaviti broj 30 što znači da je postavljeni zadatak izvršen. Unistio 17+13 možemo da otkucavamo bilo koji izraz u njenim kojih li uobičajeni u matematičkim pri tome treba da pazio da se na množenju oboljevaju uveznicima (*). Uz upotrebu zadatka (1) da se množenje ne podrazumeva, oboljevaju uveznicima (*). Uz upotrebu zadatka PRINT (17+17+17+3) podrazumevašemo. Eno, a da je zadatka PRINT (12+2)(17+3) bi izazvalo gresku dok bi se

3 - 2.1 - 12.4/16.22

Izvršavanje ove naredbe će na ekranu dati rezultat 761.595, koji je tačan na šest cifara.

— dovoljeno da se brojevi sa kojima se decimalni tabak uveličava, "dvostruki" — operacija ne moraju da budu potrebni sa kojima odemo da radišmo sa ogranjenim od 9999999 do 99999999. Brojevi već i nijedan od vama ne otkazuje da radišmo sa nizom uobičajenog zaračuna. Iak Brojevi sa zapamćili da se posle svake naredbe računara učini RET, pa ubuduće nemoćno pisanje 10. Otkucavamo na primer PRINT 100000000, radišući da ispisati 100000000, a potom priduzavamo da se odgovarački komentari. EVO E pošto da reti eksponentu, a zapis se prevedi kao 10⁸. Podesite na osnovu matematičku da nes uveriš da je 10⁸, isto što je 100000000 — jedino je zapis drukčiji. Čak nis na ovaj način ne možemo da operišemo sa neograniceno velikim brojevima — ukoliko neki od argumenta ili rezultata ili operacija neka prelazi 10³⁰ — radišući da se ispisati HOW?
— operacije bude manji od 10³⁰ ili veći od 10³⁰ — radišući na ekranu ispisati HOW?

2

28. Mašinski programi — naredba *USR*

28. Mašinski programi — naredba USR

Raspoloživi prostor nam, na žalost, ne dopušta da se ozbiljno bavimo mašinskim programiranjem, ali ćemo izložiti neke podatke vezane za srušavanje snimanje i startovanje matinskih programata na računaru na načinu: i dati jedan primjer za srušavanje

daljeg tektu je vrlo korisno posedovati spisak instrukcija procesora 280A. Postoje i drva mesta koja su, prirođeno određena za mašinske programe: jedno gde se normalno nalazi niz spomenuto (privredni bezizluk).

Da bi za mašinski program rezervisali mnogo biločita, treba da pomerimo probetak
vezika. To možemo učiniti učitavajući učitavac (MUI) (koji je uveđen u prethodnoj poglaviji).

littori, bujyjor, "Widnows" program, aet, a "Novice" program, atra "Kod" li "O" vandens, a nuo pomociu naredbe BYT, bait po bait pocevsi od **82&C3A**. Da vildmo kako to izgleda na primer.

mašinski jezik brz. Evo program už koristeći standarde memorijske procesora.

2C3D 01 00 28 LD DG, ZOOM
2C3D 21 00 28 LD LD, 2800H
2C40 36 FF CIKL LD (HL), FFH
LD (HL), FFH
FF je „pun karakter“
Izreda „osvetljen“ ažurirana
data bilo ukratko.

2C42 23	INC HL	HL se uvećava
2C43 0B	DEC BC	brojač le umanjjen
2C44 CB 78	BIT 7,B	da li je BC negativan?
2C46 28 F8	JR Z,C,JKL	ako nije, ponavlja se ciklus
2C48 C0	DET	pozivati je BASIC

ECO-1000
HE-1
, puuvillak u oksot.

$$R_1 = \frac{1.44}{ReC}$$

Pomoč konstruktoru: a stabiliti multivibrátoru

računar projektuje

Sledeći potez je vaš: Popunite ovu blok-šemu!

dosta neprihvatnih trenutaka. Da bi se problemu rešili, treba upoznati naredbu PTR izvršeno na primer: PRINT PTR X\$5 na ekranu će se pojaviti broj 10864 (ili, što je isto, &2470). Znači da alfumerički promenljivi prima 16 slova, zaključujemo da će biti &2470 konis 20 INPUT X\$...
20 Y\$...
30 FOR I=0 TO 10²
40 BYTE PTR Y\$+I, BYTE PTR Y\$+3, 0
50 NEXT I: BYTE (PTR Y\$+3), 0

Startujmo ovaj program i unesimo bilo koju reč koja ima manje od 16 slova. Trenutak dočini na ekranu će se pojaviti prva tri slova reči. Cladci ovih redova koji se razumeju i rad sa alfumeričnicima će primetiti da bi naredbe 40 i 50 mogle da se zameni sa:

40 Y\$ = Y\$ + CHR\$(BYTE (PTR Y\$+I): NEXT

Naredbu PTR možemo da koristimo i za pronađenje elemenata numeričkog ili alfumeričkog niza ili fiksnih promenljivih. Kao vežbu pokusajte da sastavite program koji će sortirati reči po abecednom redu. Problem postaje bitno složeniji kada posmatramo reči koje sadrže i slova š, ž, Č i Č.

27. Sistemski promenljivi — naredba WORD

Tabela na slici prikazuje mapu sistemskih promenljivih i drugih značajnih memorijalnih adresu. Sistemski promenljivi označavaju memorije u koje računar smreća podatke od zvučnika za njegovo normalno funkcionisanje. Jednu od sistemskih promenljivih smo već upoznali — čijila je adresa i slova š, ž, Č i Č.

28. Sistemski promenljivi — naredbe ADDRESS

UNESAK	IMATA	INICIJAL	ŠIRINA
2AB0	512	200	VIDEO MEMORIJALNE VARIJABLE A-Z
2AB0	162	200	NUMERIČKA VARIJABLA KURZORNA U MEMORIJI
2AB0	2	200	POZICIJA KURZORNA U MEMORIJI
2AB0	2	3000	KRAJ MEMORIJE
2AB0	2	3000	BLOKIRANJE DELA VIDEO-MEMORIJE
2AB0	2	3000	BYTOK REGISTAR ZA FOR-NEXT
2AB0	16	200	POZICIJA AKTIVNE TEKUCKE LINIJE
2AB0	1	200	2(ZA VREME FOR-NEXT)
2AB0	2	200	BASIC POINTER (ZA VREME CALL, JUMP, OR-NEXT)
2AB0	2	1600	IGRAZI, JEDNOSTAVNO
2AB0	2	60	ADRESPOTNIČKE VARIJABLE
2AB0	2	200	POZICIJA AKTIVNE TEKUCKE LINIJE
2AB0	2	200	REGISTAR ZA AKTIVNI FOR-NEXT
2AB0	2	200	SPRIVARNIRED ZA AKTIVNI FOR-NEXT
2AB0	2	200	DIFERENCIJATOR ZA TASTATURU
2AB0	124	60	AND METHEKTIK ALKUMULATORI (IX, IY, IZ, IZ+1, IZ-1, IZ-2)
2AB0	1	60	LINK ZA NAREDBE
2AB0	1	200	LINK ZA VIDEO
2BAF	1	60	AKO JE BIT-7 IZ-1, SAT RADU
2BB0	1	60	BROJ ZA POMERANJE SIKE
2BB1	1	60	FLAG ZA POMERANJE SIKE
2BB5	1	60	FLAG ZA SLEP
2BB6	1	60	FLAG ZA SLEP
2C3A	2	200	DAFER POČETKA BASICA
2C3A	2	200	POINTER POČETKA BASICA
2C3B	2	200	POINTER KRAJA BASICA
2C3A	77	200	BASIC PROGRAM

mesta. Da pojednostavimo problem, pretpostavimo da je brzina kretanja konstantna na čitavom putu i da je unapred zadata (okvirni zahtevana) bi se verovatno pokrovao voz koji na ulazu na naredbu, krovne krivine prepreke. Ako put koji treba da se pređe označimo slovom B, brzina slovom V a potrebno vreme slovom T, valice relacija t = s/v (čitav put, duže se putuje, što je veća brzina, vreme će se putuje). No, pre nego što počnemo da pišemo program, poškupljemo da iskoristimo naredbu PRINT i proverimo kako sve u praksi izgleda.

Od Beograda do Dubrovnika, na primer, ima 370 km u pravoj liniji (u časovima)

otkućadećemo:
PRINT 370/V/1000 i na ekranu će se pojaviti broj 0.465. Olovka i papir (za sada oči ne unesu da iskoristimo tabular da obavi sve potrebne operacije) de pokazati da ovu prenosi 27 minuta i 45 sekundi.

Koliko bi put trajao da se avion kreće brzinom od 900 km/h, ili šta je putovanje od Beograda do Zagreba? Mogli bismo, jasno, stano da učemo naredbu PRINT ali čemo uvideti da od toga nema velikih vremenskih usleda. Zato ćemo u program uneti instrukciju

30 PRINT SV

30, kako što se sedamo, broj programatske linije ali šta su S i V? Programeri ih nazivaju **promenljive** koriste za smetanje nekih podataka u memoriju računara. Svaka je ime nekog podatka koji dočini mesto da bude poziv u programu da se sa njim raduno. Nemo može da bude bilo sko do A do Z (ne radujući slova C, Z i I) pogodno S i V. Ispak, program koji smo sastavili nije potpuno. Ako okucamo RUN na ekranu (če da se pojavi broj 1) koji svakako ne predstavlja rešenje našeg problema. Razlog je jednostavan: računar, galaksija, još nije naučio da čita naše misli i nema nikakav razlog da zna šta, da treba da sadrži promenljive S i V. U početku, radi one imaju jednaku vrednost (pazite: ta vrednost nije nula) pa je rezultat deljenja ranije naredbe 10 / S = 370 / 20 = V = 800 (nismo zaboravili da posle svake linije pravila RET?). Ovo je putovanje sve je korekto radio i računalo, ako je ispisao vreme putovanja od Beograda do Dubrovnika i to izraženo u časovima.

Ako želimo da učaramo zamisliši avion okucamo 10 / V = 900. Kako nova linija imati linški broj, kao i stara, jedan od njih mora da napravi misto drugog. Radun uvek smatra da je linija koja smo poslednju okucali merodavna u uklanjanju iz memorije liniju 20 / 1000 da bi je zamenio sa 20 / V = 900. Program se ponovo startuje sa RUN i dobija se novi rezultat.

Prizemljenje rezultata koje smo oprimili, nije baš, stvarno izabrano: računar nam prikazuje vreme u časovima, i to na način koji nam nije prirođeno, navikli smo da se vreme izražava u časovima, minutima i sekundama. Da je prikazano rezultat na ovakav način potrebno je da se jedna nova funkcija — INT, da skrabiščena od INTEGER i označava, u brokunom prevedi, ceo broj U galaksijom, na primer, okucamo PRINT izračunata, "deo" deo "raz" koji je naveden iz nule. Ako je naveden INT (3/145) pristupimo "računatu" i na ekranu se pojavi broj 3. računar je jednostavno odelio decimalne broje u zagradu, U zagradi nije moralo da se nalaže konstanta, mogli smo da napisemo PRINT INT (A/1000) i da se dobiti rezultat 0.5 (A=172.000).

Da bi se broj prevojen u časove, minute i sekunde, moramo da podemoćno vreme u časovima sa 60, ceo broj rezulta pristupimo za potreban broj minut, broj časova dobitamo deljenjem vremena sa šezdeset i uzimanjem rezulta (morate da uzmite papir i olovku, proveti ovo razloženje na primjeru, nemakavog smisla da nastavljate sa čitanjem pre nego što vam da sada konštencate naredbe ne postanu kristalo jasne). Evo i programa:

```
10 S=370
10 V=800
30 PRINT INT(T)
```

50 PRINT „ČASOVA..“
 60 M=INT(M/60)
 70 PRINT M
 80 PRINT „MINUTA..“

Program, kao što se vidi, prikazuje samo časove i minute ali čete, ako ste ga razumeći, bez problema povećati tačnosti navodnjem sekundi!!

6. Naredba INPUT

Da bismo u prethodnom programu izbegli neprekidno kucanje brojnog izraza koji računa svi i kome prethodi naredba PRINT koristili smo promenjivu Na Zalisc, ni na ovaj način kucanje nije bitno, već je bilo da nas pri statom izračavaju program, računar pita za pojedine vrednosti, a da mi samo kucamo potrebne brojceve. Ova problem rešava naredba INPUT, čije delo je dejstvo upoznati posto modifikujemo prethodni program:

10 PRINT „KOLIKO SU MESTA UDALJENA?“
 15 INPUT \$ „KOLIKA JE BRZINA?“
 20 PRINT „
 25 INPUT V

Kada računar nađe na naredbu INPUT, privremeno će da prekine sa izvršavanjem programa, na ekranu ispisuje znak planca. Od korenika se čekaju da otkači neki broj, kada to učini, pritisne RET. Računar će otkačeni broj dodati u promenjivu čije je ime redeno uz INPUT i nastaviti sa radom. Program se, kao i obično, startuje sa RUN. Računar ispisuje KOLIKO SU MESTA UDALJENA?.

Otkucajmo broj 370 i pritisnimo RET. Računar ispisuje: KOLIKA JE BRZINA? ?

Čim otkucamo 800 i pritisnemo RET, na ekranu će pisati:

0 ČASOVA.
 27
 MINUTA.
 READY

READY je znak da je računar spremen za dalji rad, možemo da otkucamo RUN i damo potiske za neki drugi grad ili brzinu. No ubrzo će nam doćišti stalno kucanje naredbe RUN. Zar ne bi bilo bolje načinuti računaru da se, po izvršavanju programa i izdavanju rezultata, vrati na početak programa, od nas začazi nove podatke? Za to će nam dobro postupiti naredbe bezuslovnog prelaska (skoki).

7. Naredbe bezuslovnog i uslovnog prelaska — GOTO, IF ., ELSE, STOP

Dodatak našem programu naredbu 90 GOTO 10 (pažnja: u principu nije bitno da li u programu ostavimo prazne prostore između GOTO (ili bilo koje druge naredbe) broja 10 ali ne smemo da ostavljamo prostor unutar same naredbe: GO TO 10 će izvajati

SORRY je da otkucate neki program koji ima više od pet kilobaita (računar „galaksija“ u osnovnoj verziji ima 6 kilobaita RAM memorije, ali se nešto više od jednog kilobaita koristi za neke interne potrebe računara).

Ukoliko računar prilično odrusti u okviru neke komande (otkucali smo, na primer, RAN

unatoč RUNU na ekranu će se pojaviti jednostavno **WHAT???** **HOW???** NIO, ako niste u kolji je nastupila greška, i konstruirajući znak pitanja na nemo mesto u liniji na kom je bilo nešto, ne bi trebalo da bude teško da ločite grešku¹, preukrcavajući liniju ili konstrukcijski komandu EDIT, doveđete program u ispravno stanje. U podesi bilo većinu mnogo greški kod oblike (imatičke) naredbe, ali će se dočarati greške svih na pogrešno kucanje, na Zapis, logičke greške (najčešće su greške koje računar ne primeduje) — program se na prvi pogled nekako izvršava, ali su krajnji rezultati pogrešni.

9. Rad sa kasetofonom — *SAVE, OLD i OLD?*

Znanje koje smo stekli nam omogućava da pišemo relativno komplikovane programe.

Uzvukov na potrebitu one linije povori i ratunaru da ona sadrži kontenuter koji su sebi azume da imaju koju potiču uživateljima ne mora da bude privržen u programu. Na primer, za pre sveake grupe naredbi bacimo komentar koji objašnjava učinkovito dešto, ali i treba imati i viti da ovakve naredbe troše jedan bati memorije po vremenom uporebom slovu **do**, te, u uslovima opšteg programskog prostora, učinkovito dešto i luskuj.

23. Dalje mogućnosti naredbe HOME

Naredba HOME, kao što smo rekli, nalaže računara da obriše sadržaj ekran i o njoj se, izgleda, ne mora više razgovarati. No za HOME može da se nade i neki broj između 1 i 512 koji unekoliko menja smisao ove naredbe.

Kada učenici ispunjavaju ovu liniju netačne su posljedice za pomeranje za jedan red, naprave, mjeri, tekući se pojavljivaju u (sesaostalom) redu ekranu. Prirodan tok stvari, ali ponakod može da bude konstruisan da jedan deo ekranu zaštiti od automatskog brisanja. Na tom deku ekranu može da nadeši jedni tekst koji toku uključuju radnu programu da bude *WIFI* (na primer, sačetlo uputstvo za upotrebu kod neke igre). Dejstvo, na primer, zastihili prva tri reda ekranu, izvršćemo srednje jednostavni program:

```
10 HOME  
20 PRINT „OVI REDOVI“: PRINT „SU“: PRINT „ZAŠTICENI“  
30 HOME 96
```

Zatim kucajmo neke naredbe i kada ostatak ekrana bude ispunjen, primetimo da je „pokretan“ samo deo ispod treceg reda, nacrtan u obliku kvadrata. AT možemo da menimo nivojem

sat/10, pa **1** — da ga obrne prijevirovom **SHFT DEL**. Međutim, moraju **HOME** ostaviti zadnjim čak i ako promenite njih sadržaj, je ne moguće redoviti.

Postoji još jedno ograničenje vezano za zaštitu ekrana, kada je ona aktivirana, ubočiti fini scroll (pomeranje redova nije moglo nego prilaziti za oči) je isključivo. To znači da će ista programa biti oči pri teže pratiti ozbrojenu da redoviti prebrojati preleti ekran. Ako želimo da isključimo fini scroll bez zaštite ekrana, otukidemo **HOME** 512. Ovo stanje, kao i zaštita bilo kog dela ekrana, opoziva se sa **HOME**.

24. Heksadecimalni sistem — oznaka &

zbijom da se u IF naredbi mogu naći samo logički izkazi u kojima se koriste oznake i – smi svi oni koji su računari, onda ne smiju složiti programi učenju. Ukoliko ste poseliši trika koji cemo da izložimo, zasluzujete sve komplimente i imate sve šanse da

IF X>Y ELSE GOTO 100 (ili, s obzirom na prethodno poglavje: IF X>Y ELSE GOTO 100)

ta se desloši? Ako je već u jednaku Y, odgovor na pitanje „da li je X manje od Y.“ je tačan. Pa naručni izvršavačno što se naizadi za ELSE. Ukoliko je X manje od Y, E tvara ignorisano i radunski prelazi na sledeću liniju.

22. Računanje sa logičkim iskazima

10. Dalje mogućnosti naredbe PRINT, PRINT AT, HOME

Na samom početku smo rekli da će ovo uputstvo biti organizovano „po krugovima“:

prvi krug predstavlja upoznavanje tastature i rad sa demo-kasetom, drugi osnovne programiranja a treći, u koj upravo pokrovu naredbe PRINT.

Do sada smo PRINT savali tekst pod znacima navoda ili neki brojni izraz koji je izračunavao i prikazivan na ekranu. Naši su dan u jednom redu mogli da ispisujemo samo jednu rečenicu što je, ako ista drugo, nešto. U ovom redu naredbe PRINT može da se nade više programišenih, izraza i komentara koji je bilo prikazani u istom redu. Rečenice, međutim, zahteva uvi komentari budu odvojeni jedan od drugog znakom „ ili običnim zarezom. Ukoliko su dve stave u naredbi PRINT odvojene simbolom „akunar ili“ nadovezati jednu na drugu, a ako je upotrebiljen zarez — razmaknute ih tako da druga počinje od osmog, desetog, ili dva deset devetog mesta u redu. Ovi naredbu druga naredba od osmog, desetog, ili dva deset devetog mesta u redu. Ovi naredbu druga naredba stiže na primer.

Vratimo se na program za izračunavanje vremena putovanje između dva mesta i

obrtni linje 40. 70 i 80 a zatim otkačimo:

70 PRINT INTITI: ČASOVA; M: MINUTA; „ Izvorno program — unesimo uobičajene podatke za raspolaženje Beograd—Dubrovnik i

brzinu 800 km/h na ekranu da se pojavi izvezba:

04 ČASOVA, 27MINUTA.

Nije baš sljajno — redovi više nisu odvojeni, ali je računar objedno preteo sa zbijanjem ispred svakog broja le, docuće, ostavlja po jedno prazno mesto. U stvari nije prazno — na mesta bi bio smestio znak minus da uvođe negativan, ali nam je neophodan blanko simbol i tada nema. Jevan od načina da resimo problem je da otkucamo:

70 PRINT INTITI: „ČASOVA; M: „MINUTA.“

ali će tako rečenica bili nelogično mnošto izdeljena na polja. Pravo rešenje je naredba: 70 PRINT INTITI: „ČASOVA; M: „MINUTA.“

Upoznavanje sa simbolom „ i „jos nismo iscrpli sve mogućnosti naredbe PRINT: ona može da posavlji i za pisanje teksta na pravouglom delu ekranu. Da bi ovakvo pisanje imalo nekog smisla, treba napraviti da upoznamo naredbu za brisanje zadnjeg znaka ekranu. Ovaj efekat se u komandnom modu postiže istovremeno pisanjem čitavog načina [SHIFT] i [DEL], a u programskom — izračunavajući naredbu HOME. HOME način rečunava da „vrat kurzor kući“ koja se naziva u levoj gornjem ugлу ekranu. Pri tom se čitav zadnji ekran briše.

Na slj. je prikazana matica ekranu. Kao što se vidi, on je podijeljen na 16 redova, a svaki od tih redova na 32 mesta. Mesta su numerisana brojevima od 0 (karakter u levoj gornjem ugлу ekranu) do 511 (karakter u desnom donjem ugлу). Naredba PRINT AT može da počne da štampa od prvoizvođenog mesta kolje je navedeno iz A, pun publik naredbe je dake: PRINT AT 32. PRINT AT 32. RAČUNAR GALAKSIJA. — U ovom primeru tekst „rečunat“ se spoljavljuje na početku drugog reda, dok sačinjava prvi ostajte nepravilno. A.

Na slj. je prikazana matica ekranu. Kao što se vidi, on je podijeljen na 16 redova, a svaki od tih redova na 32 mesta. Mesta su numerisana brojevima od 0 (karakter u levoj gornjem ugлу ekranu) do 511 (karakter u desnom donjem ugлу). Naredba PRINT AT može da počne da štampa od prvoizvođenog mesta kolje je navedeno iz A, pun publik naredbe je dake: PRINT AT 300. PRINT AT 300. RAČUNAR[®].

broj i ne pritisne RET, što znači da je vreme za razmještanje neograničeno. Osim toga, naredba INPUT ispisuje na ekranu znake koji korisnik kuca i tako kvar slike koja je na ekranu prikazana. Zato računar „galaksiju“ poseduje naredbu KEY.

NAREDBA KEY (N)

TASTER N						
A	B	C	D	E	F	G
8	9	0	1	2	3	4
7	6	5	4	3	2	1
6	5	4	3	2	1	0
5	4	3	2	1	0	9
4	3	2	1	0	9	8
3	2	1	0	9	8	7
2	1	0	9	8	7	6
1	0	9	8	7	6	5
0	9	8	7	6	5	4

RPT 50
L1FT 52
SHIFT 53
ENTER 48

10 IF KEY (1) PRINT „PRATISNUO SI“; ELSE GOTO 10

Program se neprekidno izvršava sve dok korisnik ne pritisne taster A, zatim se na ekranu pojavljuje odgovarajući komentari, potle dega računar prestaje sa radom.

Na demno kaseti je naredba KEY izvezba mnogoslojnosti, rančito da ispitivanje prisutnosti neke od strelica u okviru naredbe INPUT čak i ne može ueti smisla kao karakter, pa da analiza programa sa nje pruži neuskusnoj korisniku predstavu o značaju ove naredbe.

ima funkciju koja je u mnogo čemu različita od funkcije do sada opisane naredbe KEY, KEY (O), naime, da značka koja je pridružen tasteru koji korisnik pritisne (kod se ratuna prema tabeli koja je dat u diskusiji naredbe CHRS). Naredba IF KEY (O) se, dakle, neće pravilno izvršavati, ali zato može da se napiše A=KEY (O). Kada računar, u toku izvršavanja programa, nade na ovu naredbu, on će dekodirati sve dake taster P, na primer) znaka koji je ispisani na tom tasteru pridružiti promenljivoj A.

10 HOME
20 PRINT CHR\$ (KEY (0)): GOTO 20, na primer, prenosi na ekran slova koja korisnik kuka pa se tako, na neki način, imitira „normalan“ rad računara. Međutim, program 10 X\$=„“
30 GOTO X\$=CHR\$ (KEY (0)) se, dakle, neće pravilno izvršavati, ali zato može da se napiše A=KEY (O). Kada računar, u toku izvršavanja programa, nade na ovu naredbu, on će dekodirati sve dake taster P, na primer) znaka koji je ispisani na tom tasteru pridružiti promenljivoj A.

21. Skraćivanje naredbi

Ovim poglavljem započinjemo poslednji krug: to je deo uputstva za upotrebu kojeg namenjen onima koji su sotinu savršavača, programera i galaksija. „Num započinje do koj će ukloniti prilika biti zadržane karakteristike računara i galaksija.“ Num započinje do koj će ukloniti prilika biti zadržane karakteristike računara i galaksija za savršavanje. Za to postoji iki razlozi: primjerice na maksičnom jeziku, koji zelimo da upoznamo, predstavlja viši stupanj nekog vrstu nadgradnje bežiča i njegovo namenjeno neškikomušnom korisnicima. Ovim, naravno, ne želimo nikoga da plakimo, maksimno programiranje nije tako, ono zato ne samo malo više radi i nema smisla upuštaći u mesta dok nismo savršavali bežič.

11. Crtanje po ekranu — DOT, UNDOT

Prihodno poglavje nije razjasnilo sve oznake sa mapu ekranu: tamo su redovi po tri dela, a kolone na po dva. Na ovaj naredbu su dobijene takozvane "pixeli". Kojih ima po 64 u svakom od 48 redova. Svaki ovu taku (to je zapravo, kvadrat) možemo slobodno da osvetlim ili zatamnimo konisice naredbe DOT (DOT znaci taku). Treba odmah da odredimo taku znamenju „takci“ karakter i tačka je interpunkcijski znaci prazno mesto „:“. Karakteri su adresirani brojevima od 0 do 511, a takce na mnogo komforntniji način — za oblaćanje sa šeksi putu magle od karaktera pa njima nalaži na njen položaj u tom redu. takče su šeksi putu magle od karaktera pa njima mogu da se dobijaju boje slike na ekranu.

Naredbu DOT prate koordinatne takce

koju treba osvetiti (DOT 0, 0, na primer, osvetljava taku u levom gornjem ugлу, DOT 1, 0 taku desno od nje, DOT 31/3 taku u

zrađenju, ona zatamnjuje taku čije su koordinate navedene na isti način). Program:

```
10 HOME  
20 DOT 21,20  
30 DOT 40,40  
40 UNDOT 31,23  
50 UNDOT 40,40  
60 GOTO 20
```

neprkodno uključuje i isključuje takbe čije su koordinate (31,23) i (40,40): Naredbu DOT možete da se upotrebite i unutar IF, ali takda ima sasvim drugačiji smisao: linije 100 IF 10,10 GOTO 1000 ne osvetljava taku 10,10 nego ispituju da je ta takca osvetljena, a ako jeste, preuze li liniju broj 1000.

Ukoliko je neka takca osvetljena, primaće naredbe DOT neće promeniti ovu stanje, slično tome, UNDOT neće promeniti status takce koja je vec zatamnjena. Naredbe DOT i UNDOT su veoma značajne zaslužuju da ih upoznamo na dodatnim primenama. Ipak, sačekademo izvrsno vreme obzirom da je za ovakvu demonstraciju neophodno realizovati cikluse (petlje), a to ćemo biti u stanju da uradimo dačko bolje kada upoznamo grupu novih naredbi.

12. Ciklusi — FOR-TO-NEXT-STEP

Cesto je potrebno da se odredeni skup instrukcija ponovi određen broj puta i da se, zatim, nastavi sa izvršavanjem programu. Prepostavimo da nam je potreban program koji izračuna faktorijski broj. Faktorijski se definije kao proizvod svih prirodnih brojeva zaključno sa brojem čiji se faktorijski traži (faktorijski broja 6, je na primer, 1. 2. 3. 4. 5. 6 = 720). Potrebo je dačko da počnemo od broja 1 i imozimo da se redom sa 2, 3, 4, ... sve do nekog n. Ovaj problem može primenom naredbe IF, da se reši na sledeći način:

```
10 INPUT N  
20 Y=1  
30 I=1  
50 Y=Y*I  
60 IF I=N GOTO 40; ELSE PRINT Y  
70 I=I+1  
80 GOTO 50
```

20. Naredba KEY

U nekim programima su potrebne brze reakcije igrača i naredba INPUT postaje sasvim nepogodna. INPUT, na primer, prekida rad programa sve dok korisnik ne unese neki samo specijalni slučaj naredbe TAKE n. Naredba RUN automatski izvršava TAKE 0.

19. Alternativni metod primanja podataka — TAKE i

Do sada smo primali podatke isključivo pomoću naredbe INPUT a dodjeljivali sadržajem pomenivima i pomocu komandi tiba: A=123. Često nam je, međutim, u potrebi da programu neophodno da dodelimo vrednost nekonstavnim promenljivama i elementima matrice A. U takvom slučaju imamo naredbu TAKE, može da bude neobično praktična. Iza TAKE se navodi lista, promenljivih kojima i treba dodeliti vrednosti. To mogu da budu numerički promenljivi, elementi matrice A, alfanumeričko promenljivo ili elementi stringa XS (ako je ovaj prethodno definisan sa A\$). Elementi one liste se odavaju zarezima. U početku programa će se, dačko, dešto nači naredba:

```
10 TAKE A,B,C,Z
```

Ali, odačke računar može da zume (TAKE =,uzmi) ove podatke? U okviru programa mora na razvojnom mestu prvo ili posle naredbe TAKE, po zelji da se naide jedna ili više linija koje počinju simbolom #. Ovaj simbol obaveštava računar da se u liniju koje ga sadrži naredba tablica podataka koju će biti pročitani sa TAKE. Brojevi i reči u ovim tabelama treba da budu razvojeni zarezima.

Ako se, na primer, u programu nalazi naredba:

```
100# 10,20,-11,79,779 a računar izvodi ranije navedenu TAKE naredbu, promenljiva A ce dobiti vrednost 10, promenljiva B dobiti vrednost 79, naredba C dobiti vrednost 779, naredba L dobiti vrednost 1. Ako se u istu za # nalazi manje podataka nego što se u TAKE naredbi traži, računar će pravljiv grešku. Obrnuto, ukoliko se ne učitaju svи podaci iz liste, računar će reagovati, što je prirodno.
```

Ponekad može da bude potrebno da se u okviru programa ponovo uzimaju isti podaci (potrebne su, na primer, podene vrednosti za neku igru, a zelimo da se, po završetku takve partie, automatski započeti nova sa istim početnim parametrima). U tom slučaju pre prve naredbe TAKE koja se nalazi u programu, treba da stavimo TAKE 0 (nula a ne slovo o). Na taj način, ce računar započeti uzimanje podataka od samog početka i sve # liste. Umesno nula, izaziva problem da stavimo neki drugi broj. U tom slučaju računar pronađe liniju sa nulom brojem, i priprema se da u sledećoj TAKE naredbi neće uzme podatke iz # liste koja se nalazi u toj liniji. Ukoliko u naredbi nema # liste, računa će pronaći prvu sledeću taku, listu, odakle vidimo da je TAKE 0 samo specijalni slučaj naredbe TAKE n. Naredba RUN automatski izvršava TAKE 0.

17. Rad sa alfanumericima — *CHR\$, EQ, VAL, PTR ARR\$*

Do sada smo radili isključivo sa brojevima, dok smo komentare isključili standardnim pačkotimskim jezicima, ali i uključili između njihovih komentara i za razliku od znakom, ovo podrazumevamo slovo, broj, interpunktacijski znak i prazan prostor ili, poput brojeva, za smesanje u reci korisnije koje se postavlja na temu što se za imena svake nazlazi nazaka (\$.). U programu se prepoznaju po tome što se za imena svake nazlazi nazaka (\$.). U svakom slučaju, pravim se razlike, primjerice znake i slino. Postoje samo neke fiksne anumeračke pravilnosti koje se nazivaju XS i YES. Njima se određeno da dođe ujedno s vrednostima korišćenjem standardnih bežičkih naredbi, a ne da se dozvolimo da zaboravimo da uvezemo za XS uvek (osim kod izvršavanja naredbe PRINT) stavljamo naredbu znakova navoda. Možemo, na primjer, da naredbom YES-.A-.ALFANUMERIC-. a tada smemo da postavimo PRINT XS. na ekranu će da se pojavи rec anumerački. Pored uobičajenog prenosa sadržaja jedne argumente u drugu (nareba XS = \$), ispoznamo i funkciju CHRS. Ova funkcija prati argument u zadatu koji je jednostavno uključi u isti broj između 1 i 255. Ovaj broj dobitamo iz tablice koja je data na sledećim sljedećim tablicama. Osim toga, možemo da dobitimo na ekranu i ako izvršimo sledeći jednostavni BASIC program:

STABILITÄT ASCII-KARAKTERA

```
FOR I=32 TO 255  
PRINT I,CHR$(I)  
NEXT I
```

Pogledajmo kako to izgleda na primeru. Naredba **PRINT SHRS** (65) će, na primer, prikazati slovo A na ekranu. Funkciju **CHR\$** koristimo pri formirajući složenih alfabetu - a koje se obavija sabiranjem. Alfabetni, jasno, nisu brojevi koje ima smisla sabirati. No, iako se za sabiranje

NEXT !

三

U polprogramu, očigledno, ima i novina, napisana je jedna linija koja se sastoji od više naredbi razvojnih znakova: **načini** način možemo da postupamo kad god želimo da štedimo memoriju ili, kada nam se čini da grupu naredbi treba „uspakovati“ u isti red poštovanja, predstavljaju logičnu celinu. Postoje, ipak, dva razloga zbog kojih čitav program ne može

da bude smrešten u jednu programsku liniju: maksimalna dužina linije koju računar prima je 50, ali je moguće da primi leće karri reda a naredba koja se ne nalazi na početku neće imati učinku.

Uvećavajući programu koju je učinio, učinimo da se dodigne naredba naredbi GOTO i CALL.

5 HOME

10 X=0, Y=0, Z=63, T=0, CALL 100

20 Y=47, T=47, CALL 100

30 X=63, Y=0, CALL 100

40 X=0, Y=0, Z=0, CALL 100

Razlog je u tome što se ovim zadaggi u toku programu ne menjaju, pa nema potrebe ponavljati iste naredbe i razračunati memoriju.

Vidi se da pre svakog poziva polprograma nisu postavljeni sadržaji svih promenljivih (restirujuće programu) (N01), jasno potvrđen sa polprogramom u prvom trenutku poziva povoljnije, računar uokviruje ekran. Nao. posle toga nastupa greška "To u naredbi RETURN zadato je definirana putanja preko izvornog polprograma", i tako završava naredba 40. Poste toga je nedostatno, počevši sa izvršavanjem naredbe 100, koja sledi za nje, još jednom povuklo istu liniju, naišao na RETURN. Pošto polprogram nije poziven, računar nije znao kada se vrati (RETURN = vrati se) na prethodno pozivano.

Prije modifikacije u cilju rešavanja ovog problema je da se doda naredba STOP (sekeće li se naredbe STOP?). Solucija je isto tako da se naredba na prvi pogled, kada računar nade na STOP, on preseže sa izvršavajućim programom i izlaze porukom READY, kvarci tako da se deo tekšom mukom nacrtano okvara. Zato često dodati naredbu 50 GOTO 50 koja navodi računaru da se ne prekrepođe vrati u "mrtvoj" petlji, ne kvarci, izgled ekranu (ovo je tako zatočavano dinamičko zauzimanje). Tako možemo da se divimo uokvirjenom ekranu i pristisnemo tastar BRK, tek kada nam on dovede.

U ovom polprogramu se do pozove sledeći polprogram, baš kao što u okviru jednog ciklusa petlige možda se zapoche druge, ali je broj ovakvih poziva ograničen na 13. U programu se sigurno događa da vam je potrebo više od triinaest nivoa polprograma, ali morate da pazite na početničke greške poput ove:

10 CALL 100

150 GOTO 10

100 A = 10

Na ovaj način računar neće nikada nalažiti na naredbu RETURN pa će biti u situaciji da stano poziva novu, novu polprograme sve dok ne dođe do greške. Ovakvu grešku je besto teško otkriti: program dobro radi jedno vreme a onda, u nepravilnim intervalima, način ugleđamo poniku HOW?

Jedini način za ispravljanje grešaka u programu koji smo do sada upoznali bio je prekidanje linije sa kojom je nekonformna i neobično narušavajuća solucija (u to ste se i sami uverili, pogotovo ako ste pisali iki program koji koristi komandu PRINT A). Računar neće se nade u programu (blisko parimetri) da program mozu da se etišu, samo u komandom modulu i treba da je prati i niz broj naredbi koja se edituje. Primenjena na RET, akurata brise ekran u njezinom prikazovanju i ispisuje liniju koju smo počeli da pišemo. Kurzor se nalazi na jeronom kraj. Koristeci tastere „←“ i „→“, možemo da ga povredimo u okviru naredbe sve dok ne dođemo do mesta na kom treba da ispravimo grešku. Ako želimo da izbaciemo stari kraj sa mrazi desno od kurzora, treba jednostavno pristisnemo tastar DEL. Ukoliko želimo da ubaciemo novi deo programne linije, jednostavno ga okučamo na ce da buči, uimenim potiske od tekuće pozicije kurzora, pri ponemanju kurzora i brisanju pojedinih deova programske linije, tastar REPT (autorepeti) donosi do punog izražaja.

15. Ispravljanje programa — EDIT

u memoriju računara. Ovi podaci su međusobno logički odvojeni: promenljiva V, na primjer, može da bude podatak o brzini nekog objekta, F, o kojoj kola dočuve na njega. To vremenom i sljivo. Ponekad smo međutim, sa potrebotom da obratimo pažnju međusobno srodnim podatcima. Zatim, na primer, da sortiramo podatke po veličini. Besmisleno je da krozistimo promenjive A, B, C, ... obzirom da bismo u programu morali da naredimo veliku grupu naredbi koja bi potrebljala sadržaj svake promenjive sa sanitizacijom ostalih. Odmogućiti. Stoga radijno akro treba da sortiramo vredne od 26 podataka?

Računar „gleda“ sve da je numerički niz A(0), A(1), A(2) itd. — nizovi koji ne opravljaju vrednostim memorije. Iako deo nizova, mapu memorije opisuju tek docnije, potrebno je da vete kako u grubom čitama oblažavaju razmeđenim elementima niza A(0) da bi se shvatila njegova promenljiva veličina. Zamislimo da se od potekla memorije prema kraju smestili BASIC program, a od kraja memorije prema početku elementi niza A(0). Skraćeni elementi A(0), A(1), ... zaustavu po četiri bata. Javno je da se, nizove na stvari, memorije niz A(0) i BASIC program „susredu“. Računar smatra da je BASIC program znakom niz učitava niz A(0) tako da popunjava samo subodredje de memorije. Ukoliko pokusamo da menjamo vrednost nekog elementa niza A(0) koji bi prenimo belzici program, računar će prekiniti sa radom i izdati poruku SORRY.

otkučano jednostavan program:

10 A=0

20 A(0)=A(1)

30 I=I+1

40 GOTO 20

Startujemo ga sa RUN. Kada se na ekranu pojavi poruka SORRY, otkučamo

PRINT 1 i saznamo broj prve nepostojecu elementa niza. Ovaj način je naredito dobar,

ponavljajući isto u postojeci bezijk program treba da dodatemo novu naredbu koje ga nepotrebno proizvodišu i tako skraćujući se na ekranu broj belzova memorije koji su

stvoreni do prime A(0). Kako svaki element ovog niza zaustavlja po četiri bata, PRINT

Evo, na posleku i programu koji će oključiti razumovanje rada sa nizom A(1). Program

uredi grupu belzova u astuši, niz prikazuje dobijene vrednosti sa manje razumljivim za početnike, pa

dobar algoritam postavlja naravno, da tako iščekuje uokolo ilustruje brznu radu računara „galaksiju“.

10 PRINT „KOLIKO IMA BROJEVA“; INPUT N

20 FOR I = 1 TO N: INPUT A(I): NEXT I

30 FOR I = 1 TO N-1: FOR J = I+1 TO N

40 IF A(I)=A(J) GOTO 60

45 IF A(I)≠A(J) GOTO 50

50 A(I)=A(J): P

60 NEXT J: NEXT I

70 FOR I = 1 TO N: PRINT A(I),A(I): NEXT I

ispakivati, nekoliko puta menjano isti deo linije, možemo da pristisnemo REPT i tako se sačasno ispisati sa izvršavanjem zmenama ili da pristisnemo BRK i tako računaru uokviriti da ignorise izmene i ostavi liniju u stanju u kom je bila pre nego što je editovanje započeto.

Pri editovanju se potpisuje i jedan artefakt: nema prepreke da promenimo sam broj nizova (niz je uokvirjen). Kada pristisnemo REPT, računar uči da će uokviriti nizove u očitavajućem programu, ali nizove neće biti u promenljivi u memoriju uneti novu umnožavajuću liniju (dok stara neće biti u promenljivi u memoriju uneti novu razmazujuću liniju).

Kada smo zaštitili sa ispravama neke linije (nema nikakve prepreke da, u toku ispravki, nekoliko puta menjamo isti deo linije), možemo da pristisnemo REPT i tako se sačasno ispisati sa izvršavanjem zmenama ili da pristisnemo BRK i tako računaru uokviriti da ignorise izmene i ostavi liniju u stanju u kom je bila pre nego što je editovanje započeto.

Pri editovanju se potpisuje i jedan artefakt: nema prepreke da promenimo sam broj nizova (niz je uokvirjen). Kada pristisnemo REPT, računar uči da će uokviriti nizove u očitavajućem programu, ali nizove neće biti u promenljivi u memoriju uneti novu umnožavajuću liniju (dok stara neće biti u promenljivi u memoriju uneti novu razmazujuću liniju).

16. Numerički niz A (II)

Do sada smo upoznali promenljive A, B, C, ... Y i Z, pomoći kojih smetamo podatke u memoriju računara. Ovi podaci su međusobno logički odvojeni: promenljiva V, na primjer, može da bude podatak o brzini nekog objekta, F, o kojoj kola dočuve na njega. To vremenom i sljivo. Ponekad smo međutim, sa potrebotom da obratimo pažnju međusobno srodnim podatcima. Zatim, na primer, da sortiramo podatke po veličini. Besmisleno je da krozistimo promenjive A, B, C, ... obzirom da bismo u programu morali da naredimo veliku grupu naredbi koja bi potrebljala sadržaj svake promenjive sa sanitizacijom svih ostalih. Odmogućiti. Stoga radijno akro treba da sortiramo vredne od 26 podataka?

Računar „gleda“ sve da je numerički niz A(0), A(1), A(2) itd. — nizovi koji ne opravljaju vrednostim memorije. Iako deo nizova, mapu memorije opisuju tek docnije, potrebno je da vete kako u grubom čitama oblažavaju razmeđenim elementima niza A(0) da bi se shvatila njegova

promenljiva veličina. Zamislimo da se od potekla memorije prema kraju smestili BASIC program, a od kraja memorije prema početku elementi niza A(0). Skraćeni elementi A(0), A(1), ... zaustavu po četiri bata. Javno je da se, nizove na stvari, memorije niz A(0) i BASIC program „susredu“. Računar smatra da je BASIC program znakom niz učitava niz A(0) tako da popunjava samo subodredje de memorije. Ukoliko pokusamo da menjamo vrednost nekog elementa niza A(0) koji bi prenimo belzici program, računar će prekiniti sa radom i izdati poruku SORRY.

sa radom i izdati poruku SORRY.

otkučano jednostavan program:

10 A=0

20 A(0)=A(1)

30 I=I+1

40 GOTO 20

Startujemo ga sa RUN. Kada se na ekranu pojavi poruka SORRY, otkučamo

PRINT 1 i saznamo broj prve nepostojecu elementa niza. Ovaj način je naredito dobar,

ponavljajući isto u postojeci bezijk program treba da dodatemo novu naredbu koje ga

nepotrebno proizvodišu i tako skraćujući se na ekranu broj belzova memorije koji su

stvoreni do prime A(0). Kako svaki element ovog niza zaustavlja po četiri bata, PRINT

Evo, na posleku i programu koji će oključiti razumovanje rada sa nizom A(1). Program

uredi grupu belzova u astuši, niz prikazuje dobijene vrednosti sa manje razumljivim za početnike, pa

dobar algoritam postavlja naravno, da tako iščekuje uokolo ilustruje brznu radu računara „galaksiju“.

10 PRINT „KOLIKO IMA BROJEVA“; INPUT N

20 FOR I = 1 TO N: INPUT A(I): NEXT I

30 FOR I = 1 TO N-1: FOR J = I+1 TO N

40 IF A(I)=A(J) GOTO 60

45 IF A(I)≠A(J) GOTO 50

50 A(I)=A(J): P

60 NEXT J: NEXT I

70 FOR I = 1 TO N: PRINT A(I),A(I): NEXT I