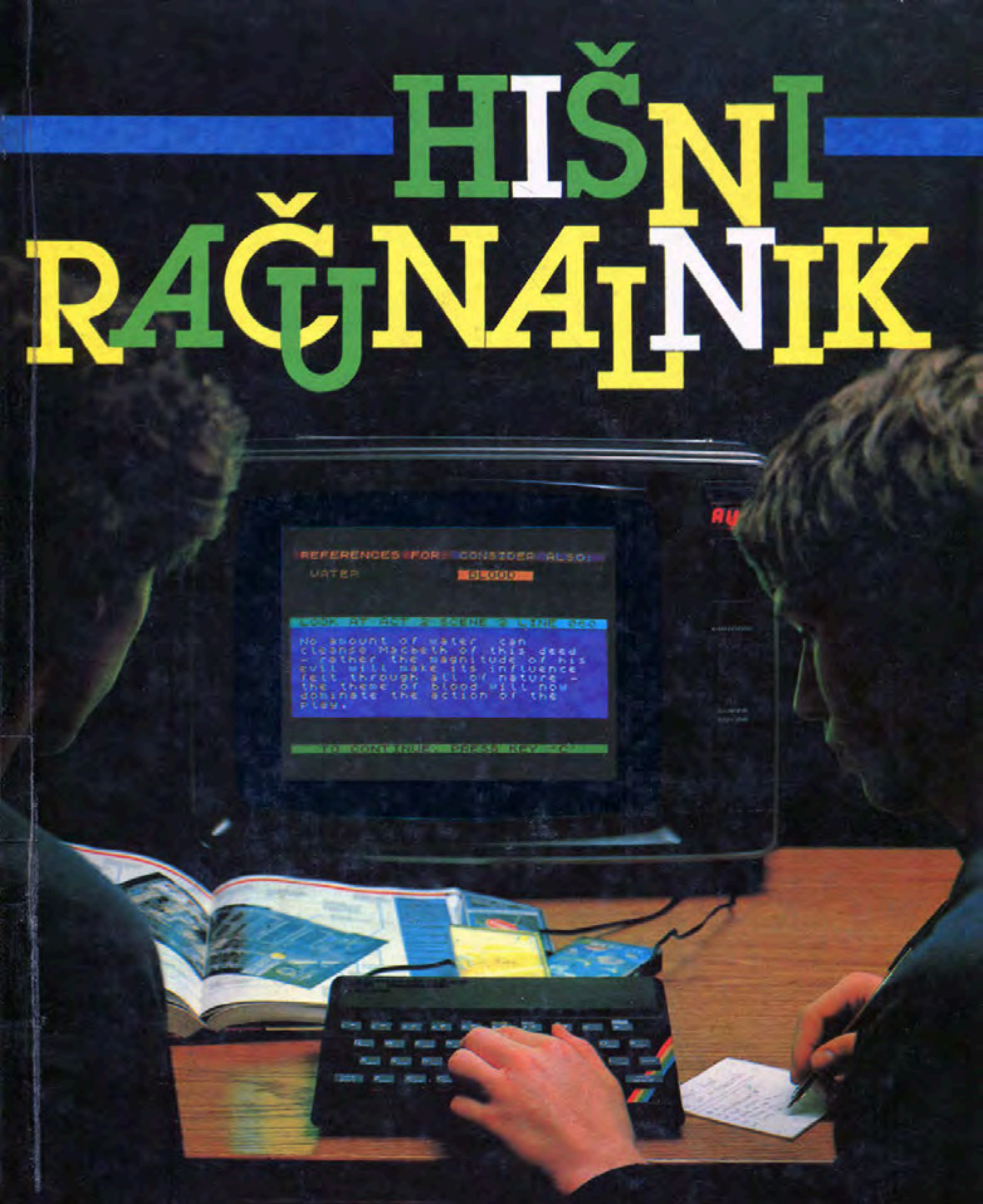
HIŠNI RAČUNALNIK



založba mladinska knjiga

HIŠNI RAČUNALNIK

Angleški izvirnik:

Choosing and Using Your Home Computer

Edited by Jonathan Hilton

© Orbis Publishing Limited 1984
© za Jugoslavijo Mladinska knjiga 1984

Slovenska izdaja:

Strokovna ureditev in koordinacija prevajalcev:
dr. Jan Lokovšek in Nikola Simić

Uredila: Ciril Trček in Jože Vilfan

Prevode strokovno lektoriral: mag. Boris Sobočan

Prevod oziroma predelave:
mag. Andrej Jesenko
d.i. Borut Jurčič
dr. Jan I. Lokovšek
mag. Zlatko Matič
dr. Radko Osredkar
mag. Tone Pleteršek
d.i. Nikola Simić
d.i. Damir Smerdelj
mag. Boris Sobočan
dr. Marjan Špegel
d.i. Slavko Starašinič
d.i. Maks Štiglic
dr. Peter Tancig

Originalni prispevki:

Prispevek o hišnih računalnikih domače proizvodnje: Boris Sovič
Poglavje o računalniškem razumevanju naravnega jezika: dr. Peter Tancig
Prispevek o ekspertnih sistemih: dr. Ivan Bratko in mag. Nada Lavrač
Prispevka o strojni kodi in zbirniku: Tone Pleteršek

Lektor: Janez Gradišnik

Korektor: Franc Lenaršič

Tehnična ureditev: Peter Miklič

Oprema: Virgilij Dariš

Izdala Založba Mladinska knjiga

Glavni urednik Založbe Borut Ingolič
Za Založbo Vid Štempihar

Natisnila Tiskarna Mladinska knjiga,
Ljubljana 1984

Hišni računalnik

Založba Mladinska knjiga
1984

Uvodna beseda

Doba zabavne elektronike mineva s tem, da zlasti mladina nadomešča svoje glasbene bokse z malimi računalniki, ki v prvi fazi dajejo tudi mnogo zabave, vendar ti malčki lahko popeljejo (ne samo mladega) človeka v čudoviti svet informacij, v informacijske metode dela, v nove procese dela itd. Zadnje nas vodi k novim izdelkom, ki so že danes odvisni od znanja in nas sploh vodijo k drugačnemu odnosu do dela. Posebej mladina vidi v računalništvu novo možnost uveljavitve, še posebej v tem času, ko je nezaposlenost mladih po vsem svetu visoka.

Ko opazujemo prehod »industrijske« družbe v »informacijsko«, je vredno poudariti, da računalništvo in nasploh vsa informacijska tehnologija daje človeku predvsem možnosti za optimalnejše delo in življenje. Z njima se večajo informiranost in komunikacije med ljudmi in v prihodnje vedno bolj tudi med stroji. Informacijski izdelki delajo družbo bolj komunikativno, zadnje pa je tudi pogoj za boljše medsebojne odnose na vseh ravneh in v vseh smereh.

Vzrok za prehod v »informacijsko« družbo je po eni strani vsestransko pomanjkanje (energije, cenene delovne sile, živeža, papirja itd.), po drugi strani pa čipna usmerjenost informacijske tehnike, ki bo zaradi izredno nizke cene sistemov z visoko stopnjo funkcionalnosti omogočala: urade, ki bodo delovali brez papirja, delovne linije v tovarni praktično brez žive delovne sile, vsebino knjige v bralnem pomnilniku (ROM) in ne na papirju itd. Omenjenih sprememb ne bomo dosegli čez noč, temveč počasi s trdim delom in z nabiranjem informacijskega znanja. Osebni in hišni računalniki so samo prvi korak v tem pogledu. Povsem na mestu je misel iz pričujočega dela: težko je napovedati razvoj za več ko nekaj let, nekaj pa je zanesljivo: tisti, ki se bodo ukvarjali z osebnimi in hišnimi računalniki, bodo laže razumeli in dojeli prihodnji razvoj.

Knjiga poskuša na prijazen način, v kratkih sestavkih predvsem preprostemu bralcu povedati, kaj je hišni računalnik in kaj z njim lahko dela. Hitro se znajdemo v množici podatkov s področja tržnega, aparaturnega in programskega dela hišnih računalnikov, vendar je omogočeno tudi hitro dojemanje pojmov, saj so razlage jedrnate in preproste. Če preberemo delo, imamo obilo informacij zadnjega časa in dobro izhodišče za delo z lastnim mikroračunalnikom.

Iz našega dnevnega časopisja in javnih občil izhaja precej zelo približnih ali celo napačnih razlag in pojmov s področja informacijske tehnike. Zanesljivo je za to krivo veliko pomanjkanje slovenske strokovne literature in prenaglo naraščanje zanimanja za računalništvo. Prevod pričujočega dela lahko pripomore tudi k odpravi omenjene nevšečnosti. Če bodi tisti, ki pišejo in govorijo o računalništvu, prebrali vsaj obravnavano delo in delali z osebnim ali vsaj hišnim računalnikom, bo dobesedna pismenost tudi v javnosti zadostna.

Jernej Virant

Vsebina

1. poglavje

Spoznajmo računalnik

9 Kaj je računalnik
13 Prva pomoč kupcu računalnika
14 Postavitev računalniškega sistema
15 Krmiljenje računalnikov

2. poglavje

Izbiramo računalnik

20 Drobni tisk
21 Sinclair Spectrum
24 Vprašanja in odgovori
25 Commodore 64
28 Dragon 32
30 Vprašanja in odgovori
31 BBC model B
34 Sinclair QL
36 Vprašanja in odgovori
37 Atari 400 in 800
40 Acorn Electron
42 Hišni računalniki Made in Yugoslavia
47 Epson HX-20

3. poglavje

Osnovni pojmi

51 Biti in zlogi
53 Ko je 1+1=10
55 Seštevanje in odštevanje
57 Množenje v dvojiškem sestavu
58 Pravilno ali napačno
60 Logična vsota in seštevalnik
62 Popolno pomnjenje
64 Shranjevanje podatkov
66 Razumem sporočilo
68 Živčni center
70 Pravilno naslovljeno
72 Digitalni dialog
74 Čakalnica
76 Dva načina prevajanja
78 PEEK in POKE
79 Parno pravilo
80 Detektivski posli
82 Vplivne zveze

4. poglavje

Računalnik in njegova oprema

86 Tesen dotik
88 V arhivu
90 Hitri reaktorji
92 Ploščati pomnilnik
94 V žarišču
96 Opremljen za tiskanje
99 Na dveh kolesih
102 Hitro risanje
104 Halo, računalnik...
106 Branje grafičnih podatkov
108 Načrtovanje po naročilu

5. poglavje

Računalnik pri delu in igri

110 Ljudje in igre

112 Elektronski vzgojitelj
115 Mikrosvetovi
117 Lepa beseda
120 Preglednica
123 Doba prenosnih računalnikov
126 Komunikacija človek—računalnik
132 Miške
134 Enoroki rokovnjači
136 Gibajoče se slike
139 Vojna in mir
142 Šola na zaslonu
145 Ekspertni sistemi

6. poglavje

Umetnost programiranja

152 Diagram poteka
154 Urejanje podatkov
156 Razvrščanje informacij
158 Verižna sporočila
160 Urejanje kart — pravila igre
162 Koda za urejanje
163 Optimizacija hitrosti
165 Inteligentne igre
168 Sam svoj avtor
170 Programski pripomočki
172 Strojna koda
173 Zbirni jezik
174 Branje zemljevida

7. poglavje

Tečaj programiranja v basicu

177 Izvajamo ukaze
180 Pomembne zanke
183 Preidimo k stvari
185 Pomen rutinskih programov
187 Novo leto v basicu
190 Izziv elementom
193 Uredimo si programe
197 Funkcije v basicu
201 Prepustimo jih naključju
205 Neka druga dimenzija
209 Kontrola

8. poglavje

Grafika in zvok

214 Še zvok...
214 ...in svetloba
216 Ovojnica
216 Igra s svetlobo
218 Zvok in VIC 20
219 Računalnik Dragon 32
220 Preprosti zvoki
220 Prve slike
222 Zvočni sistemi
222 Svetlobni program
224 Kvalitete zvoka
224 Svetlobna zgradba
226 Preizkus zvoka
226 Igra s svetlobo
228 Spektakularni zvok
228 Light show
230 Sliši se neverjetno
230 Svetlobni relief

1. poglavje

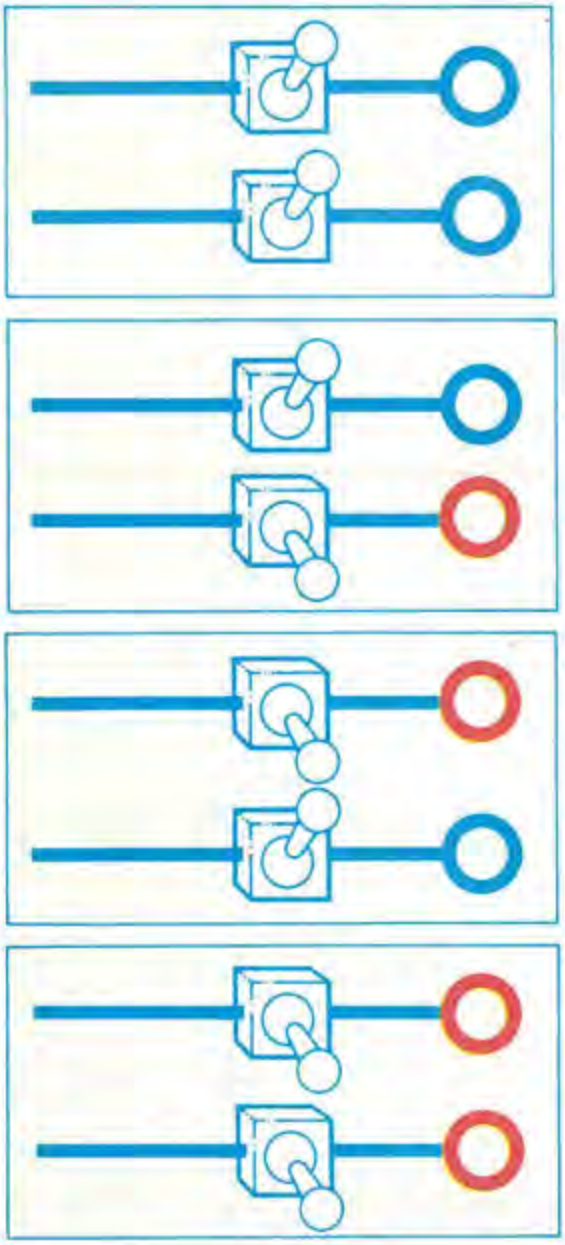
Spoznajmo računalnik

Veliko beremo o računalnikih, toda kljub temu preprosto ne najdemo odgovora na najosnovnejše vprašanje: kaj je sploh računalnik in kaj zmore?

Kaj je računalnik

Kako računalnik »misli« in koliko »ve«? Odgovor je bistven za razumevanje delovanja računalnika

Na vprašanje, kaj je računalnik, ni lahko odgovoriti. Ta odgovor ni tako preprost, kot če bi odgovarjali na vprašanje, kaj je televizija ali kaj je pralni stroj, kajti računalnik je v primerjavi s tema dvema napravama večnamenski. Digitalni računalnik, kakršnega lahko na Zahodu kupimo po sorazmerno nizki ceni, je naprava nove vrste, ki lahko opravlja različne naloge, odvisno pač od vrste programa, ki mu ga da uporabnik. Pojem programiranja pa v sodobnem domu ni neznan, saj imajo mnoge naprave, kot npr. pralni stroj, štedilnik, pečica itd., že vdelano različno število programov, ki določajo, kako naj naprava opravi posamezno nalogo. Računalniku lahko funkcijo spremenimo tako, da mu vstavimo nov program. Z računalnikom



Ponazoritev števil
Računalniki uporabljajo električne signale za ponazoritev števil. Vezja, ki manipulirajo s temi električnimi signali, so sestavljena iz množice drobnih elektronskih stikal, ki so lahko v dveh položajih: vključena ali izključena. Z vključitvijo oziroma z izključitvijo dveh stikal lahko dobimo 4 različne kombinacije: izključeno/izključeno je nič, izključeno/vključeno je 1, vključeno/izključeno je 2 in vključeno/vključeno je 3. Z uporabo večje množice stikal lahko prikažemo tudi večja števila. Računalnik lahko hitro opravlja zapletene matematične operacije z velikimi števili tako, da uporablja na tisoče mikroskopsko majhnih stikal.

se lahko igramo, urejujemo besedilo, spremljamo stanje našega bančnega računa, računamo itd.

Kako lahko računalnik opravlja tako različne naloge? To bomo zvedeli v tej knjigi. Vendar si najprej oglejmo nekaj temeljnih lastnosti računalnika.

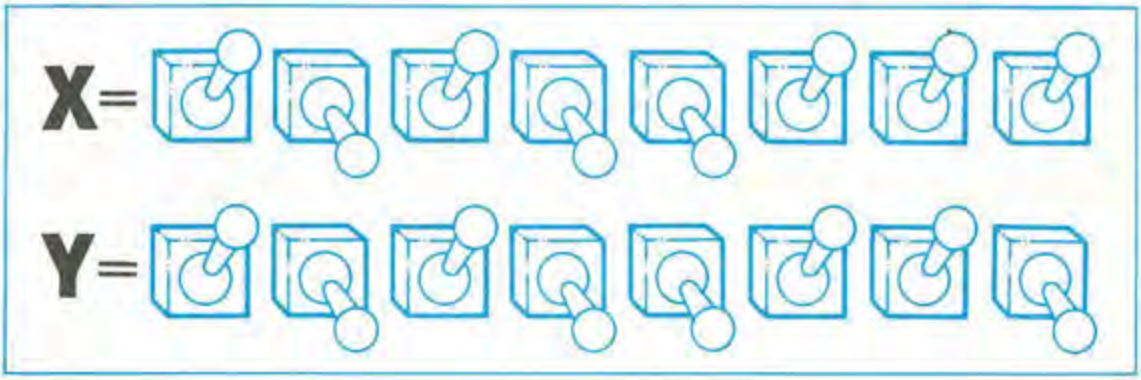
Na en način računalnik ni nič drugega kot škatla, polna drobnih elektronskih stikal, ki se lahko med seboj povezujejo na različne načine. Toda to ni najboljši pristop k razumevanju delovanja računalnika. Takšno poznavanje računalnika je potrebno strokovnjakom, ki načrtujejo in izdelujejo računalnike. Za nas pa takšno znanje ni potrebno. Gotovo pa je, da je sodobni računalnik izredno zapletena in obsežna naprava. Za hitri razvoj sodobnih računalnikov se imamo zahvaliti osupljivemu razvoju mikroelektronike, to je integriranim vezjem na silicijevi ploščici. Tako lahko majhen hišni računalnik vsebuje več ko 250000 majhnih stikal, ki so lahko vključena ali izključena. Pri enem stikalu lahko imamo dve stanji: stikalo je lahko vključeno ali izključeno. To »zmešano« možnost pa zakriva program, ki ga ima računalnik trajno vdelanega in nam omogoča, da lahko komuniciramo z računalnikom že z nekaj okrajšanimi angleškimi besedami.

Veliko ljudi je presenečenih, ko prvič vključijo računalnik, saj takrat ugotovijo, da računalnik nič ne ve. Presenečenje je še toliko večje, ker nekateri imenujejo računalnik kar »elektronski možgani«. Zato bi računalnik moral vedeti, katero je glavno mesto Slovenije ali kako visok je Triglav. V resnici pa drobno integrirano vezje vsega tega ne ve. Ne pozna ne črk ne številk. Vse, kar »ve« ali »pozna«, je nekaj sto številskih kombinacij, ki jih spreminja po natanko določenem vrstnem redu, kakor mu to določa program. Majhna stikala si lahko zapomnijo le dve stanji. Kadar je stikalo vključeno, imamo dvojiško število »1«, kadar je izključeno, pa nam predstavlja dvojiško število »0«. Dejstvo, da lahko računalnik pomni oziroma hrani informacije, je bistvenega pomena za njegovo delovanje. Elektronski pomnilnik računalnika Sinclair Spectrum lahko vsebuje toliko informacij kakor 6 strani besedila te knjige. Seveda lahko shrani veliko več na kaseto ali disketo.

Tako dobro, kakor zna računalnik hraniti števila v svojem pomnilniku, zna z njimi opravljati tudi druge operacije: lahko jih sešteva in odšteva, medsebojno primerja in premika znotraj svoje pomnilniške enote. Vse, kar naj računalnik opravi, je podano s preprostimi ukazi. Vzemimo, da želimo vstaviti v računalnik besedilo. Pri tem mora biti vsaka črka podana kot število. Računalnik tedaj shrani besede kot števila. Pri tem so nekatera drobna stikala vključena, nekatera izključena. Podobno je, če se želimo igrati igro »Žabar«. Vzamemo lik, ki predstavlja žabo, in ga pomikamo po kvadratih mreže. Vsak izmed njih je podan kot število. Ta kodirana števila je treba potem seveda prevesti. Toda brez strahu, tega ne bomo počeli mi, to so storili že strokovnjaki, ki so izdelali program.

Kaj je program? To je niz ukazov računalniku, ki mu določajo opravljanje preprostih opravil, kakor so seštevanje, primerjanje itd., podobno kakor pletilji vzorec določa zaporedje preprostih vbodov, da izdela želeno obleko. Toda kaj so ti ukazi in kako pridejo v računalnik? To so le števila, ki so shranjena v računalnikovem pomnilniku. To nas spominja na paradoks kure in jajca. Računalnik ne zna narediti ničesar brez programa, ki bi mu to ukazoval. Če pritisnemo črko A na tipkovnici računalnika, bo računalniški program »preiskal« tipkovnico, ugotovil, katera tipka je pritisnjena, in potem povedal računalniku številsko kodo za črko A. Na začetku razvoja računalnikov ta program za »čitanje« tipkovnice še ni obstajal. Skrbni načrtovalci so s posebnimi aparaturami shranili številske kode naravnost v pomnilnik, ki označuje posamezne tipke tipkovnice. Tako računalnik razume črke, ki jih odtipkamo, in nam jih tudi prikaže na televizijskem zaslonu.

Takrat, ko je bil izdelan prvi temeljni program, je postalo vse lažje. Sedaj lahko vtipkamo v računalnikov pomnilnik nova števila. Ta stopnja se imenuje programiranje v strojni kodi. O tem se bomo seznanili pozneje. Programiranje v strojni kodi je precej zapleteno in dolgočasno in samo bistroumni programerji pišejo takšne programe, ki prevedejo angleške besede, kakor so PRINT, LOAD in LIST, v ukaze v strojni kodi, ki jo lahko računalnik razume. Večina hišnih računalnikov



Kode za črke in števila
Osem stikal, ki jih lahko vključimo ali izključimo, nam omogoča 256 kombinacij. To je več kot dovolj, da označimo z različno kodo vse črke, številke, ločila ter še nekaj posebnih znakov, kot npr. znak za računalnikov tiskalnik itd. Na sliki je prikazano, kako sta v računalniku ponazorjeni črki X in Y v kodi ASCII.

ima vdelan tak program. Tako lahko programiramo v preprostem računalniškem jeziku, ki ga imenujemo basic. To je vsekakor ugodnejše kakor pa programiranje z zaporedjem številk. Vsakokrat, ko bomo uporabljali basic, se spomnimo, da je vanj vloženih na stotine ur dela mnogih programerjev.

Z računalniškimi jeziki, kot je npr. basic, je precej preprosto pisati programe, ki opravljajo uporabne in zabavne reči. Z lahkoto je npr. mogoče napisati program, ki nam bo hranil glavna mesta republik in pokrajin. Če odtipkamo računalniku vprašanje, katero je glavno mesto Slovenije, nam bo takoj izpisal, da je to Ljubljana. To pa hkrati pomeni, da »elektronski možgani« znajo samo to, kar jim povemo, sami ne znajo misliti. Če je to res, zakaj so potem računalniki tako uporabni? Shranijo lahko velikansko količino informacij, s katerimi lahko opravljajo številne naloge dosti bolje in hitreje, kakor to znamo ljudje. Seveda pa mora informacijo računalniku dati človek. Če izdelamo program, ki ima shranjena glavna mesta republik in pokrajin, nam rabi računalnik kot elektronska knjiga. Če pa imamo program za urejanje besedila, ki ga vtipkamo v računalnik, lahko v njem preprosto naredimo razne popravke v besedilu. Kot naslednji zgled navedimo še program za urejanje baze podatkov. Če v računalnik vstavimo podatke o knjigah v domači knjižnici, njihovih avtorjih, letnicah izdaje itd., imamo izredno hiter pregled nad stanjem v njej. Dejstvo, da ubogi neumni računalniki razumejo samo številke, je prej prednost kakor slabost. Vzemimo za zgled računalnik, ki nam na televizijskem zaslonu lahko riše le v nekaj glavnih barvah. Če želimo več barvnih odtenkov, to še ne pomeni, da bi morali kupiti drugačen računalnik. Vse, kar je potrebno, je to, da mora programer znati opisati problem tako, da ga bo računalnik razumel. V končni fazi bo to zaporedje številk.

Če npr. želimo izdelati program, da bi nam računalnik zaigral želeno melodijo, opišemo vsako noto iz notnega črtovja kot število, ki ustreza višini tona oziroma njegovi frekvenci. Računalniku ukažemo, da pošlje električne signale, ki jih uporablja za prikazovanje števil, na zvočnik namesto na televizijski zaslon. Tako lahko slišimo rezultat svojega dela.

Kako na zaslonu izstrelimo raketo proti prihajajočemu vesoljskemu napadalcu? To storimo s premikanjem številk, ki predstavljajo obliko izstrelka, iz enega prostora v drugega v računalnikovem pomnilniku, ki ima vlogo »zemljevida« televizijskega zaslona. Slike, gibanje, barve, zvok itd., vse to je podano z ustreznimi

Zakaj programska oprema

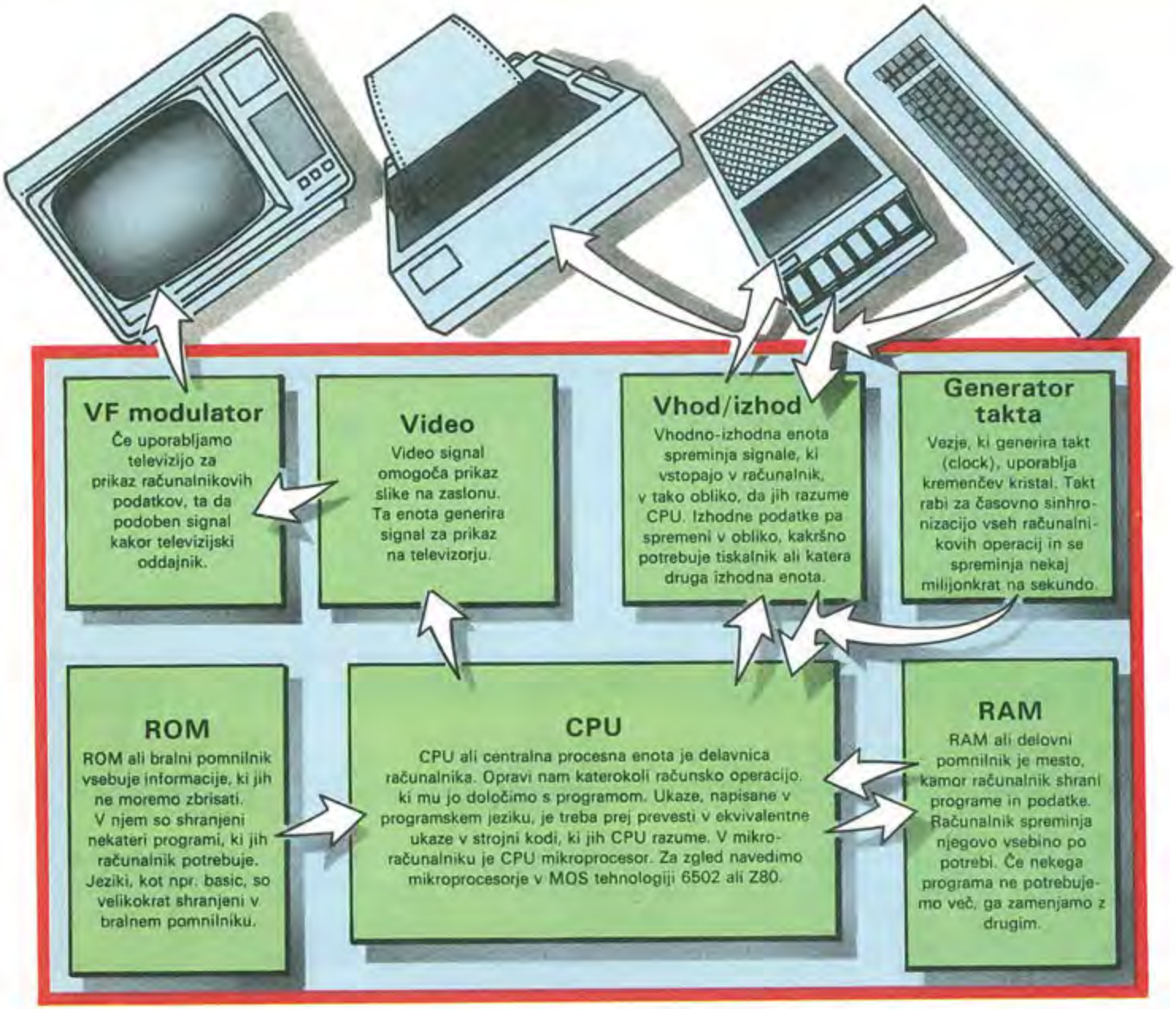
Mikroračunalnik je učitelj

Uporaben za pisanje in računanje

Računalnik, uporabljen v znanosti

Igre in razvedrilo

Računalnik je večnamenski stroj za izvajanje različnih opravil, ki mu jih določa programska oprema. Poslovneži uporabljajo računalnike s poslovno programsko opremo, tehnologi uporabljajo programsko opremo tehnološke narave, doma pa oskrbujemo računalnik s programsko opremo, ki nam rabi pretežno za zabavo. Programi so torej tisti, ki določajo računalniku, kaj naj dela.



Kaj se dogaja znotraj računalnika

Da je celotni računalniški sistem uporaben, moramo medsebojno povezati nekaj posameznih enot. Integrirana vezja, ki so omogočila izdelavo hišnih računalnikov, so vdelana v ohišju, običajno pod tipkovnico. Če odpremo ohišje, vidimo nekaj integriranih vezij in še nekaj drugih elementov.

številskimi kodami, s katerimi ravna računalnik. Primeren medij, kakor je televizijski zaslon ali zvočnik, jih spremeni v signale, ki nam nekaj povedo. Sedaj lahko končno odgovorimo na vprašanje, ki smo si ga zastavili na začetku: »Kaj je računalnik?« To je naprava, ki hrani električne signale, ki predstavljajo števila. Nekatera od teh števil so ukazi, ki povedo računalniku, kaj naj stori z drugimi števili. Računalnik sledi tem ukazom natanko in brez napak s hitrostjo nekaj tisoč operacij na sekundo. (Seveda pa računalnik natanko ponovi napake, ki smo jih vstavili z napačnim programiranjem.) Nasledek teh neutrudnih manipulacij s števili je še večja množica števil. Ta se nato »prevede« v podatek, ki ga želimo, in v tako obliko, ki jo razumemo.

Torej programerji so tisti, ki izrabljajoč računalnikovo spretno manipuliranje s števili naredijo računalnik koristen. Če ne bi bilo njih, bi bilo preoblikovanje in vnašanje različnih podatkov dolgočasno, zamudno ali pa prezahtevno.

Prva pomoč kupcu računalnika

Dileme ob nakupu hišnega računalnika so velike. Naj pomagamo pri izbiri!

Večina hišnih naprav ima svoje natanko določene naloge. Video naprave, televizijski in drugi sprejemniki se lahko med seboj razlikujejo po izvedbah, toda stereo sistem lahko reproducira le zvok, pralni stroj zna le prati in TV sprejemnik lahko prikazuje le oddajano sliko in zvok.

Računalnik je nekaj povsem drugega. Cela vrsta ljudi lahko kupi enak tip računalnika, pa ga vendar vsak od te množice uporablja na drugačen način. Zato je tudi izbira hišnega računalnika nekaj posebnega, kar se ne da primerjati npr. z nakupom pomivalnega stroja. Če se že odločite za tak nakup, se morate najprej vprašati, kaj pravzaprav nameravate početi z njim. Najbolje je napraviti nekakšen seznam nalog, ki naj bi jih računalnik opravljal. Poglejmo zgled. Če se želite naučiti programiranja v basicu, je stroj za vas Sinclairjev Spectrum. Če si želite računalnik, ki bi poleg iger lahko rabil tudi kot urejevalnik besedil in bil na voljo tudi za vodenje domačega gospodinjstva, potem je najbrž najboljša izbira Commodore 64 ali BBC model B. Prav gotovo pri vsem tem tudi cena ni nepomembna.

Računalnik, ki ga imate oziroma nameravate kupiti, bo gotovo šele srce sistema. Brez nekaterih perifernih enot namreč ni mogoče izkoristiti vseh potencialov takega stroja. Za hranjenje programov potrebujete kasetni magnetofon ali morda diskovno enoto. TV sprejemnik je danes res že del vsakega gospodinjstva, vendar hišni računalnik takoj prinese prepir v hišo. Prav gotovo vseh članov družine ne zanima računalništvo, temveč želijo spremljati TV spored. Logična posledica tega je nakup drugega, morda manjšega in cenejšega televizorja.

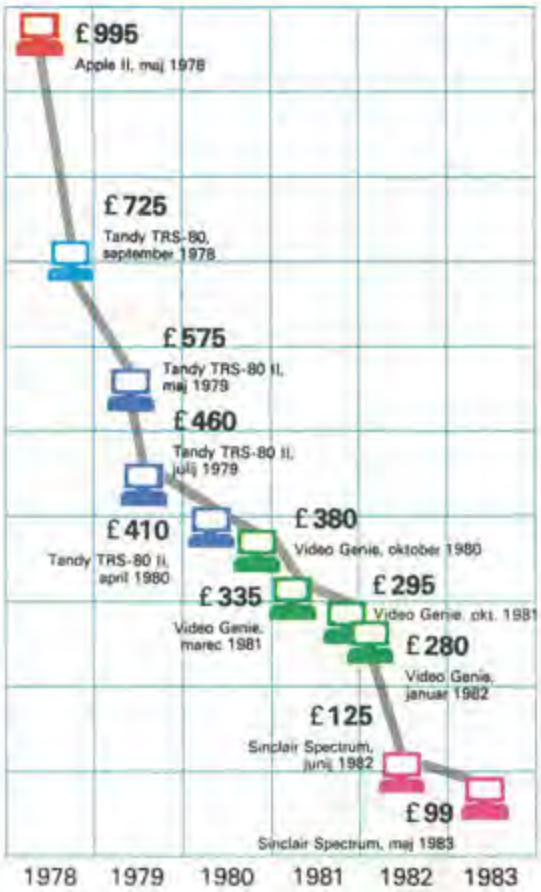
Za malo resnejše delo potrebujete izpis na papirju in takrat ne gre brez ustreznega tiskalnika.

Ta je seveda nepotreben, če nameravate računalnik uporabljati samo za igre. V tem primeru je eno pomembnejših dejstev, koliko je primerne razpoložljive programske opreme. Tukaj se držijo na vrhu Sinclair Spectrum, VIC-20 in BBC, saj je zaloga programov zanje na kasetah videti skoraj neizčrpna.

Koliko pomnilnika potrebujete? To je odvisno od kompleksnosti programov, ki jih želite uporabljati. Velja pravilo, da za daljši in bolj sestavljen program potrebujemo več prostora v pomnilniku. Urejevalnik besedil potrebuje večji pomnilnik; v splošnem zadošča 32 kb, medtem ko je za najnavadnejše igre dovolj tudi 16 kb. Izkušnje kažejo, da je najboljše kupiti stroj kar z največjim pomnilnikom, ki je na voljo. Večina kupcev namreč kasneje dokupi del pomnilnika.

Nekateri malo dražji hišni računalniki, kakor so Commodore 64, BBC in Atari 800, so primerni tudi za resno delo v pravem pomenu besede. Vsi trije lahko uporabljajo tiskalnik, modem, disketne in druge enote, in jih tako lahko po telefonskem omrežju povežemo v računalniški sistem.

Torej, prepričajte se, ali ste pri sestavljanju zahtev pomislili na vse, kar od računalnika pričakujete, in kaj od tega vam je zares tudi na voljo. Vse to naj bo kristalno jasno, preden se odločite za nakup.



Upadanje cen mikro-računalnikov

To so cene, po katerih so prodajali najbolj znane tipe računalnikov z najmanj 16 kb RAM v standardni izvedbi, tj. brez dodatkov.

Postavitev računalniškega sistema

Za dobro komuniciranje z računalnikom, za shranjevanje programov in podatkov ter za različne igre potrebujemo še dodatno strojno opremo

Tiskalnik

Tiskalnik uporabljamo za izpis programov ali rezultatov na papir. Na tržišču je veliko različnih vrst tiskalnikov, ki se razlikujejo po ceni, hitrosti in kvaliteti izpisovanja.

Diskovna enota

Podobno kakor s kasetofonom tudi z diskovno enoto shranjujemo programe in druge podatke. Namesto kasete uporabljamo diskete (gibke diske). Na disketi lahko shranimo več informacij kakor na kaseti. Je tudi veliko hitrejša in seveda dražja od kasete. Diskovno enoto na splošno uporabljajo računalniki za izvajanje poslovnih programov.

Televizor

Običajen televizor nam omogoča prikaz računalnikovih sporočil. Med pisanjem programa se vsaka tipka, ki jo pritisnemo na tipkovnici, izpiše na zaslonu. Monitor, ki ga vidimo za televizorjem, pa omogoča boljšo kvaliteto slike.

Kasetofon

Hišni kasetofon omogoča najcenejše shranjevanje podatkov. Med delovanjem ima računalnik program shranjen v delovnem pomnilniku. Z izključitvijo računalnika se vsebina delovnega pomnilnika izbriše. Zato moramo pred tem s kasetofonom posneti program na kaseto. Če pa želimo ta program zopet izvajati, ga predvajamo računalniku, da si vsebino vpiše v delovni pomnilnik.

Računalnik

Je srce računalniškega sistema. Za boljšo povezavo z uporabnikom potrebuje še dodatno strojno opremo. Ima podobno tipkovnico kakor pisalni stroj, vendar z nekaj dodatnimi posebnimi tipkami. Konektorji, ki so običajno na zadnji strani, skrbijo za povezavo z drugimi napravami, kakor so kasetofon, diskovna enota in televizor.

Miška

Miško uporabljamo pri različnih igrah. S pomikanjem kroglice v držalu lahko pomikamo element v neki igri po zaslonu. V primerjavi s krmilno ročico nam omogoča hitrejše in natančnejše pomikanje po zaslonu. Tipke so namenjene za streljanje z »laserjem« in podobno.

Krmilna ročica

Krmilno ročico uporabljamo največ pri raznih igrah. Njena vloga je v raznih igrah različna. Z njo lahko vodimo vesoljsko vozilo ali pa točko po labirintu. Nekatere krmilne ročice imajo tudi po nekaj tipk. Uporaba le-teh je v raznih igrah različna.

Krmiljenje računalnikov

»Strojna oprema« računalnika bo neuporabna brez ustrezne »programske opreme«. Pojasnili bomo ta pomembni pojem in nakazali, kakšno programsko opremo potrebujete

Programska oprema je nevidni del računalniškega sistema, toda brez nje je računalnik neuporabna kopica elektronike. Brez programske opreme ne more računalnik opraviti dobesedno ničesar.

Če bi pogledali v notranjost integriranega vezja računalnika, bi odkrili, da je v njem na tisoče, morda celo na milijone mikroskopsko majhnih elektronskih stikal. Podobno kakor se stikalo za luč ne vklaplja in izklaplja samo od sebe, potrebujejo stikala v računalniku nekaj, kar jih vklaplja in izklaplja. Seveda se ne smejo vsa stikala vklopiti in izklopiti hkrati, ampak se mora vsako posamezno stikalo posebej vklopiti (ali izklopiti), in sicer v natanko določenem trenutku glede na vse druge tisoče stikal. To nalogo opravlja programska oprema.

Programska oprema je ime za ukaze, ki omogočajo delovanje računalnika. Ti ukazi imajo obliko številk, ki v centralni procesni enoti (glej stran 12) vklapljajo in izklapljajo stikala tako, da se dogajajo natanko določene stvari. Te številke »razume« računalnik samo, če so v t.i. dvojiški obliki (spremenjene v enice in ničle, kot je razloženo na strani 51). Enice in ničle, o katerih je govor in ki jih računalnik razume (v takem smislu, da ga pripravijo do tega, da opravi določeno nalogo), so končni izdelek dolge verige dogodkov, ki so se pričeli z zamislijo v programerjevi glavi.

Računalniški program (»program« je beseda za vsak posamezni del programske opreme) lahko obstaja v mnogih različnih oblikah. Edino, kar lahko o nekem programu z gotovostjo trdimo, je to, da mora biti končna oblika računalniku razumljiva. Poglejmo si zgled. Recimo, da želi prometni inženir krmiliti skupino semaforjev z računalnikom. Za to bo krmilni računalnik potreboval program, ki bo v njem vzbujal pravilna zaporedja dogodkov (zelene luči na vseh semaforih hkrati niso najboljša rešitev!). Še preden se

Bralni pomnilnik (ROM)

Bralni pomnilnik je ena glavnih spominskih naprav računalnika. Je otrok revolucije, ki jo je povzročilo silicijevo integrirano vezje, in omogoča stalno shranitev računalniškega programa. Večina hišnih računalnikov ima bralni pomnilnik, ki vsebuje programski jezik basic. Za nekatere računalnike lahko dokupite dodatne bralne pomnilnike, ki omogočajo uporabo drugih programskih jezikov in tako izboljšujejo njihove lastnosti. Na voljo so tudi bralni pomnilniki z urejevalnikom besedil.

Kaseta

Programska oprema se dostikrat dobi na kasetnem traku, enakem tistemu za snemanje glasbe. V taki obliki se običajno dobijo igrice. Program spravimo s traku v računalnik s tem, da povežemo stroj z običajnim kasetofonom in »predvajamo« program s kasete. Ko je program naložen, se trak ustavi. Običajno računalniku ni treba ponovno »pogledati« na trak.

Disketa

Programska oprema se lahko shrani na magnetni disk. Zapisi so shranjeni v »sledeh« na površini diska. Za pisanje in branje z diska rabi bralno-zapisovalna glava. Diskete imajo veliko kapaciteto in veliko hitrost delovanja, to pa moramo dobro plačati. Potrebujemo namreč zapletene diskovne enote (glej stran 14), ki so drage.

Vtični modul (Cartridge)

Vtični modul je bralni pomnilnik, vdelan v primerno ohišje. Nekateri hišni računalniki imajo zlahka dostopne vtičnice, v katere se lahko vtični moduli priključijo. Programska oprema, ki jo dobimo na vtičnih modulih, sega od programskih jezikov (na primer basic) do zelo zapletenih iger.

Da bo vaš računalnik sploh deloval, ga morate »nahraniti« s programsko opremo (množico elektronskih ukazov). Naprave na teh slikah so namenjene shranjevanju programov (programske opreme). To so štirje najbolj običajni načini shranjevanja programske opreme. Vsak ima svoje prednosti. Programska oprema je narejena za vsako vrsto računalnika posebej, tako da program, napisan za eno vrsto, najbrž ne bo delal na drugi.

inženir lahko loti pisanja programske opreme, mora dobro premisliti, kaj sploh želi od računalnika. Navadno se to zapiše v običajnih stavkih (npr.: V tem hipu želim, da se na semaforju številka 1 prikaže oranžna luč, rdeča pa naj še kar gori. Nato naj ugasneta obe, rdeča in oranžna, in naj se prižge zelena). Jasno je, da teh stavkov v tej obliki ne more razumeti noben računalnik, zato jih je treba prevesti v program. Ta je napisan v programskem jeziku, kakor je na primer basic. Jezik take vrste omogoča logično razporejanje misli (v angleščini) tako, da ga interpreter za basic lahko razume. Interpreter za basic je tudi program, katerega naloga je pretvorba originalnega programa (pisanega v basicu) v obliko, razumljivo centralni procesni enoti računalnika. Programska oprema v taki obliki se imenuje »strojni jezik« ali »strojna koda«. Dejansko je vsa programska oprema, ki si jo kupite, v strojnem jeziku in shranjena tako, da je računalniku zlahka dostopna. Včasih je programska oprema shranjena kar v računalnikovem bralnem pomnilniku, vendar se najpogosteje dobi na kaseti ali disketi. Ti dve stvari sami po sebi nista programska oprema, ampak samo medij, na katerem je ta podana. Če jo želimo uporabiti v računalniku, moramo programsko opremo prenesti iz kasete (ali iz diskete ali bralnega pomnilnika) v računalnik. Ko so ukazi naloženi (tako se imenuje postopek prenašanja programske opreme), lahko program steče.

Nakup programske opreme

Z nekaj sto tisočaki v banki bi si lahko rekli: »Kupil si bom avto.« Zelo čudno pa bi bilo, če bi nekdo dejal: »Kupil bom neki stroj.« Takoj bi ga vprašali: »Kakšno vrsto stroja? Kaj bo ta stroj delal?«

Podobno je s programsko opremo. Računalnik sam po sebi je hrom, nekoristen, programska oprema, ki jo kupite, pa vam ga spremeni v zabavišče, avtomatski pisalni stroj ali hišnega računovodjo. Zato se morate najprej odločiti, kaj naj računalnik dela. Začnite pri problemu in nato poiščite programsko opremo, ki ponuja rešitev. Pri iskanju primernega programa boste vedno natančneje spoznavali svoje potrebe.

Če na začetku iščete nedeljsko zabavo za svoje otroke, potem je vaša naslednja naloga, da ugotovite, kateri programi vam takšno zabavo dajejo. Računalniške igre segajo od arkadnih iger s pokoli Marsovcev do zahtevnih fantazijskih simulacij (glej stran 110). Ko boste natanko vedeli, kakšne igre si želite, morate ugotoviti, ali so sploh izvedljive na vašem računalniku.

Ker so razlike med računalniki resnično zelo velike (vsak računalnik ima svojo elektroniko in potrebuje ustrezno programsko opremo), so modeli med seboj inkompatibilni. Program, namenjen Atariju 800, ne bo deloval na Spectrumu (razen, če obstaja verzija za Spectrum), tako da morate kupiti programsko opremo, ki je narejena za vaš računalnik.

Vendar to še vedno ni dovolj, da bi se lahko odločili za nakup. Naslednje, kar morate premisliti, so omejitve vašega računalnika. Poglejte, koliko spomina ima vaš računalnik. Če ima 16 kb delovnega pomnilnika, potem poglejte, ali je program, ki ga kupujete, daljši in zanj potrebujete dodaten pomnilnik. Velja na primer pravilo, da je program tem daljši, čimbolj zanimive in zahtevne igre vsebuje.

Strojno realizirana programska oprema (Firmware)

Izvir izraza »strojna oprema« je očiten. To je mehanski in elektronski del računalnika: napajalni priključki, tipkovnica, integrirana vezja... Prav tako je očiten izvir izraza »programska oprema«, s katerim mislimo množico ukazov, združenih v programu. Računalniški strokovnjaki govorijo tudi o »strojno realiziranie programski opremi«. V otroških letih računalništva, pred 20 do 30 leti, je bila programska oprema zakodirana in shranjena na preluknjanem papirnatem traku, podobnem tistemu na teleprinterju. Potem so ga zamenjale kasete in magnetni diski. V sedemdesetih letih je bila odkrita nova tehnika shranjevanja programske opreme v bralnih pomnilnikih (glej stran 12). Tabletke bralnega pomnilnika imajo programsko opremo že vdelano, zato tudi izraz »strojno realizirana programska oprema«.

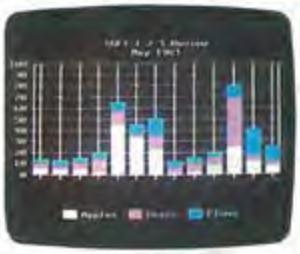
Kar je računalnik in s tem tudi naš hišni računalnik v bistvu v naprej anedoločene stroj, je prav zaradi tega primeren za najrazličnejša opravila. Ko ga »nahranimo« s programom, nam enako ustrežljivo pomaga pisati, kot da bi bil »inteligentna« pisalni stroj, vodi naše računovodstvo, zbira podatke v podatkovnih bazah, napoveduje, kaj se bo zgodilo z našimi financami ali pa nam omogoča, da se z njim igramo.

Popravljanje besed

Urejevalnik besedil

S programsko opremo za urejevalnik besedil ste dobili nekaj več kakor pisalni stroj. Tudi najboljše strojepiske se zmotijo, z urejevalnikom besedil pa lahko vedno izpišete pisma brez napak in tudi vaša učinkovitost bo večja.
Računalniška tipkovnica prevzame vlogo pisalnega stroja, televizijski zaslon pa vlogo papirja. Besede, ki jih tipkate, se kažejo na zaslonu ravno tako kakor na papirju v pisalnem stroju. Toda tu je podobnosti konec in računalnik pokaže svoj moč. Napake lahko popravljate sproti — na zaslonu. Besede lahk pretipkavate ali brišete, cele odstavke lahko odstranite. Če mislite, da bo drugačen vrstni red stavkov bolje izražal vaše misli, lahko to mirno naredite — na zaslonu seveda. Besede ali stavke, ki jih želite na kaki drugi strani, lahko začasno odstranite in nato vstavite, kjer želite. Urejevalnik besedil jih bo vzel z zaslona in shranil v pomnilniku računalnika.
Ko je tekst napisan po vaših željah, se lahko izpiše s tiskalnik ali se za kasnejšo uporabo shran na kaseti ali disketi.

Vodenje knjig



Obračunski paket

Ker računalniki zlahka opravljajo matematične operacije, ni čudno, da je na voljo zelo veliko programov za poslovne ljudi. Količina poslovne programske opreme je neverjetna — od avtomatskega knjigovodstva do popolnih obračunov. Te vrste programov običajno obdelujejo velike količine informacij in potrebujejo veliko zapisov. Zato običajno zahtevajo vsaj eno disketo, s katero potešijo velike potrebe po pomnilniku.
Obračunski programi po navadi delajo po sistemu vprašanj (izpisana so na zaslonu) in odgovorov (da jih uporabnik). Informacijo, ki jo uporabnik odtipka, računalniški program obdela. Opravi potrebne račune, rezultate pa shrani na disketo ali izpiše na tiskalniku, odvisno od tega, kaj je najprimernejše.
Takšni programi obsegajo avtomatsko izdajanje računov, obnavljanje zalog v skladišču, vodenje poslovnih knjig in pregled nad opravljenimi deli.
Cene te vrste programske opreme se gibljejo od deset tisoč do nekaj sto tisoč dinarjev za posamezni program. Tako draga programska oprema je lahko dobra naložba za podjetja, saj poceni dela in hitro daje rezultate.

Zbiranje podatkov

Podatkovne baze

Računalniki lahko iščejo podatke po datotekah veliko hitreje kakor ljudje. Čim več je informacij, tem bolj vam lahko računalnik pomaga. Podatkovna baza je v svoji najpreprostejši in tudi najcenejši obliki malo boljši računalniški naslovnik, ki lahko poišče imena, naslove in telefonske številke. Bolj obdelane in seveda dražje podatkovne baze pa lahko opravljajo še dosti bolj zapletene operacije.
Da prikažemo vso moč, ki jo dajejo podatkovne baze, si zamislite botanika, ki zbira podatke za knjigo o redkih in strupenih gobah. Zbrati bo moral ogromne datoteke o različnih vrstah gob in okolju, v katerem rastejo. Poleg tega bo imel množico zapiskov o drugih knjigah s tega področja in dolg seznam specialistov za gobe. Preden so postali računalniki široko dostopni, bi vse te informacije izpisali na kartice, ki bi bile urejene po abecednem redu.
S programom za podatkovne baze in primernim računalnikom pa lahko vse te podatke shranimo v pomnilniku računalnika. Z uporabo podatkovne baze lahko botanik kadarkoli dobi odgovore na svoje probleme. Če na primer želi seznam gob, ki so jih kadarkoli opazili na Dolenjskem, mu ga podatkovna baza lahko da. Če želi seznam knjig, v katerih se pojavljajo besede »strupi«, »strupen« ali »goba«, in to po abecednem redu, mu podatkovna baza lahko da tudi to.
Podatkovne baze operirajo z velikimi količinami podatkov in jih navadno dobimo na disketah. Običajno so precej drage, cene se gibljejo od deset do sto tisoč dinarjev.

Računanje

Preglednice

Preglednica je računalniški odgovor na vprašanje: »Kaj bi se zgodilo, če bi?« Po navadi so se jih lotevali s kalkulatorjem in z gorami papirja. Na ceno vsakega izdelka vpliva veliko spremenljivk. Če spremenimo katerokoli izmed njih, se običajno spremenijo tudi druge. Zamislite si težave, ki nastanejo v podjetju, ki mu dovolijo, da zviša svoje cene. Za koliko naj jih zviša? Če bo podražitev prevelika, bo prodaja upadla, tovarna ne bo mogla izkoristiti svojih kapacitet, vmešal se bo tržni inšpektor in rezultat bo lahko celo manjši čisti dohodek kakor pred podražitvijo. Kako najti ceno, ki bo za podjetje najugodnejša? Preglednica je program, ki hitro daje odgovore na takšna vprašanja.
Vse pomembne številke so razvrščene v vrstice in stolpce, med katerimi se določi medsebojna odvisnost (primer take odvisnosti je, da števila v stolpcu C dobim z razliko stolpcev A in B). Brž ko so zbrani vsi potrebni podatki, lahko opazujemo, kako sprememba nekaterih podatkov vpliva na vse druge. Cena preglednic je od tisoč do nekaj sto tisoč dinarjev. Običajno potrebujejo diskovne enote in tiskalnik.

Zabava

Računalniške igre

Računalniki niso primerni samo za obdelavo besed in številk, ampak vam lahko dajejo tudi obilo zabave, če uporabite katerega od mnogih programov za igre. Njihov široki izbor sega od šaha in kockanja do arkadnih iger in simulacij (kot na primer »pristanek na Luni«).
Obstajajo tudi izredno obširne pustolovske igre, ki lahko trajajo dneve in tedne. Mnoge igre niso samo zabava, temveč so namenjene tudi izobraževanju. Računalniške igre so izrazito interaktivne, drugače povedano, terjajo stalno pazljivost in dejavnost igralca. To se dogaja prek tipkovnice: posamezne tipke lahko sprožijo »laser« ali »izstrelek« ali kontrolirajo gibanje nekega lika na zaslonu. Število v ta namen uporabljenih tipk se spreminja odvisno od igre in koliko kontrole program potrebuje.
Privlačen nadomestek tipkovnici je krmilna ročica. Priključi se na računalnik in deluje približno tako kakor njen soimenjak v letalih. Omogoča boljšo kontrolo in večji užitek ob igri.

Nakup programske opreme

Programsko opremo največkrat prodajajo na disketah ali kasetah in s priročnikom, ki opisuje način uporabe in kako najbolje izkoristimo lastnosti programa. Applov program »Writer« je tipičen urejevalnik besedil in stane okrog tri tisoč dinarjev. Za ta denar dobimo disketo in obširen priročnik z razlago, kako se program uporablja. V priročniku imamo nasvete v obliki hitrega tečaja, tako da lahko tudi začetniki takoj pričnó delati.
Ker obstajajo enotni standardi glede dokumentacije, se včasih zgodi, da dobimo programsko opremo s tako nepopolnimi in slabo spisanimi priročniki, da je uporaba programske opreme skrajnje otežena ali celo nemogoča. Zato je pomembno, da pri nakupu programske opreme dobro pregledamo njen priročnik.

Seveda ne pozabite, da programi prihajajo v najrazličnejših oblikah (glej prejšnjo stran). Če se program dobi na disketi, vi pa imate samo kasetofon, ga ne boste mogli uporabiti, ne da bi kupili drago diskovno enoto. Nekateri deli programske opreme (posebno igre) zahtevajo dodatke, kakor so na primer krmilne ročice. Za igrice res ne potrebujete tiskalnika, toda poslovna programska oprema ga zelo pogosto potrebuje za izpis rezultatov.

Na koncu gre seveda še za vprašanje cene. Igrice na kasetah se dobijo že za manj ko tisočak, vendar so cene zelo gibljive. Poslovni paketi na diskih pa stanejo več deset tisoč dinarjev.

Vrste programske opreme

Na neki način so igre razred zase, saj so namenjene zabavi. Večina druge programske opreme je narejena za pomoč pri delu. Načinov, kako znajo pisci programske opreme povečati koristnost in učinkovitost, je nešteto. Zamislite si strojepisko, ki ne zmore dobro opraviti zahtevanega dela. Z mikroračunalnikom in z delom programske opreme, imenovanim urejevalnik besedil, prevzame računalnik nalogo strojepiske. Popravke je mogoče narediti na zaslonu. Ko so vse besede popravljene, se cela stran s pritiskom na gumb avtomatsko izpiše na papir. Prihranimo si neizmerno veliko časa in truda.

Naslednja vrsta utrujajočega dela, ki sama po sebi kliče po računalniku, je denarno poslovanje. Mnoge dejavnosti, ki so zaposlovale cele vojske pisarniškega osebja — računanje plač in letnih poročil — lahko sedaj opravimo s primerno programsko opremo. Programi pa so precej specializirani, zato je malo verjetno, da bi ena vrsta programske opreme lahko zadostila vsem delovnim potrebam. Tako so na voljo programi za računanje plač in pisanje plačilnih kuvert, programi s seznami prodanega ali porabljenega materiala (nekateri programi celo sami naročajo material), obstajajo celo programi za računanje najgospodarnejše velikosti in kvalitete papirja, na katerem tiskamo knjige ali revije.

Naslednje delo, ki ga lahko računalniki opravijo neverjetno dobro, je zbiranje informacij. Ta vrsta programa se imenuje »podatkovna baza.« Podatkovne baze lahko nadomestijo cele skladovnice spisov in povrh še preurejajo ali iščejo določene podatke.

Zadnja velika vrsta programske opreme je znana kot »preglednica«. Ta vrsta programov omogoča izdelavo in brezkončno dopolnjevanje sicer zamotanih proračunov ali finančnih načrtov, hkrati pa z njimi prihranimo goro papirja.

Vse omenjene vrste programske opreme se prodajajo kot »konfekcija«. Njihov pisec je za določeno vrsto problemov, ki se jih je lotil, našel določeno vrsto rešitev. Vsekakor pa se lahko zgodi, da vam ne bo ustrezal noben kupljen program. In kaj sedaj? Ena od mogočih rešitev, nekoliko draga sicer, je, da najamete računalniškega programerja, ki bo izdelal program natanko po vaših željah. Druga rešitev je, da se sami naučite pisanja programov. Z znanjem jezika, kakor je na primer basic, boste lahko naredili programe, ki bodo počeli na vašem računalniku prav neverjetne stvari. Edini strošek bo čas, ki ga boste porabili za pisanje programa.

2. poglavje

Izbiramo računalnik

Odločili smo se, da si bomo kupili računalnik.
Sedaj smo pred odločitvijo, katerega naj izberemo.

Drobni tisk

O mikroračunalniku povedo tehnične specifikacije vse

Tipkovnica

Tipkovnica je posebej izdelana zato, da je njena uporaba preprosta in prijetna. Tipke so razporejene tako kakor pri pisalnih strojih. Dokler je tipka pritisnjena, se znak ponavlja. Lahko pišemo z velikimi in z malimi črkami. Na voljo imamo še posebno skupino tipk za številke.

Zaslon

Večina računalnikov uporablja za predstavitev črk, številk in drugih simbolov kodo ASCII (American Standard Code for Information). Nekateri računalniki lahko na zaslonu prikažejo znake v 25 vrsticah in 80 kolonah, in sicer na tv sprejemniku ali posebnem monitorju (zaslonu).

Pomnilnik

Številke pomenijo velikost pomnilnika v kb. ROM (bralni pomnilnik) vsebuje zapise, potrebne za temeljne operacije računalnika, skupaj s programskim jezikom, kot je npr. basic. RAM (pomnilnik) služi za shranjevanje uporabniškega programa in podatkov.

Vmesniki

Naprave, kakor so tiskalnik, kasetofon, komunikacijske naprave ali priključeni modul, so lahko z računalnikom povezane s posebnimi priključki. Priključeni modul je ROM, ki lahko vsebuje program, programski jezik ali celo nov operacijski sistem.

CPU

CPU je kratica za centralno procesno enoto, izdelano na drobni silicijevi tabletki. Je srce vsakega računalnika. Opisana Z 80, proizvajalec Zilog, je ena najbolj razširjenih. Takt, ki narekuje izvajanje vseh operacij v računalniku, je natanko 2,2 milijona utripov v sekundi.

Lastnosti »tipičnega« računalnika

Pomnilnik	16 kb ROM, možnost naslavljanja 48 kb RAM
Zaslon	Lahko prikaže ASCII znake v 25 vrsticah s po 80 v vsaki, izhod za tv in monitor
CPU	Z 80, takt 2,2 MHz
Tipkovnica	Ergonomska oblika, QWERTY tipkovnica, možnost ponavljanja, posebna tipkovnica za številke
Vmesniki	Vmesniki za tiskalnik, komunikacije, kasetofon in vtični modul
Basic	Ukazi za grafiko in zvok, preverjanje zapisov, izpis, možnosti strukturiranja programov
Grafika	Videoteks, največja ločljivost 640 × 256 točk, tridimenzionalni prikaz
Zvok	Sintetizator zvoka, obseg pet oktav, hi-fi izhod
Razpoložljiva periferija	Kasetofon, diskovna enota, enota za trdi disk, tiskalnik, risalnik, digitalizator, krmilna ročica, modem, sintetizator zvoka
Razpoložljivi jeziki	Forth, pascal, logo, lisp, prolog, zbirnik

Basic

Temeljni jezik računalnika vsebuje tudi ukaze za delo z zvokom in grafiko. Prevaja vse instrukcije in opozarja na napake. Vsebino, prikazano na zaslonu, je mogoče s tiskalnikom natiskati na papir. Dodatni ukazi v basicu omogočajo strukturno programiranje, kar pomeni, da se dajo programi enostavno brati in popravljati.

Zvok

Razpon tonov ali akordov znaša pet oktav. Možna je priključitev na hi fi ojačevalec.

Razpoložljiva periferija

Periferija je skupno ime za posebne enote, ki jih lahko priključimo na računalnik, kakor npr. kasetofon, diskovna enota, tiskalnik in podobno. Medtem ko na kasetofonu in diskovni enoti lahko shranimo programe ali podatke, pa tiskalnik, risalnik in digitalizator rabijo za izpis ali pa prenašanje podatkov v računalnik. Krmilna ročica je dobrodošel pripomoček pri igrah.

Grafika

Videoteks lahko prikazujemo na zaslonu z ločljivostjo 256 vrstic s po 640 točkami. Mogoče je ustvarjati tudi tridimenzionalne učinke.

Razpoložljivi jeziki

Namesto basica lahko uporabljamo tudi enega od v preglednici naštetih jezikov. Vsak je dober za svoj specifični namen. Zbirni jezik je vrsta programskega jezika, ki se ga težje naučimo kot npr. basica, vendar pa programi, napisani v njem, tečejo veliko hitreje.

Sinclair Spectrum

Izvrstna barvna grafika in edinstveni sistem za shranjevanje podatkov Microdrive, ob neverjetno nizki ceni, sta glavni značilnosti Spectruma — vendar pazite na tipkovnico

Sinclair Spectrum je majhen hišni računalnik z barvno grafiko, zvokom in velikim pomnilnikom. Glavni razlog za njegovo uspešnost je njegova nizka cena — Spectrum s pomnilnikom 16 kb je bil prvi računalnik z barvno grafiko, ki je stal manj kot 100 funtov. Zaradi nizke cene se je zelo dobro prodajal, in ustvarili so veliko podjetij, ki izdelujejo programe za ta računalnik.

Spectrum ponuja uporabnikom velik delovni pomnilnik, do 48 kb, s katerim lahko računalnik izvaja obsežne programe v strojni kodi ali lastnem basicu. Ti programi so lahko že pripravljeni (spravljeni na kaseti ali disketi), ali pa jih uporabnik napiše sam. Spectrumova verzija basica ima, kot pri večini drugih računalnikov, svoje posebnosti, vendar je dovolj podobna splošno sprejeti standardni obliki basica, da bo blizu vsakemu programerju, ki ta jezik obvlada.

Vezja za proizvajanje zvoka in risanje grafike lahko krmilimo z basicom. Krmiljenje zvoka na Spectrumu je precej primitivno, z uporabo ukaza BEEP. Z nekaj domiselnosti lahko z njim ustvarimo majhen repertoar zvočnih učinkov. Nasprotno pa so grafične zmožnosti prav osupljive. Zanje je značilna vrsta ukazov, od duhovito izbranih PAPER in INK, ki določata barve podlage in znakov ter risb, pa do ukazov za risanje krogov in ukazov za utripanje.

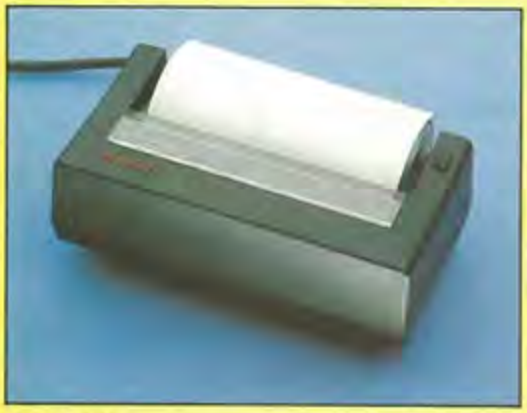
Med pripravljenimi programi, ki jih imamo za Spectrum, je na voljo vrsta različnih iger, od popularnih Space Invaders in Pac-Man do pustolovskih iger in simulatorjev letenja. Zasnova nekaterih od teh iger je presenetljivo izvirna.

Spectrumovo zmožnost za razširitev je v neki meri pokril sam Sinclair s tiskalnikom ZX Printer in s sistemom za shranjevanje podatkov ZX Microdrive. Poleg tega vrsta drugih proizvajalcev ponuja razne dodatke, med njimi krmilne ročice, vmesnike za krmiljenje in komuniciranje z drugimi napravami, profesionalne tipkovnice ter sintetizatorje zvoka in govora.

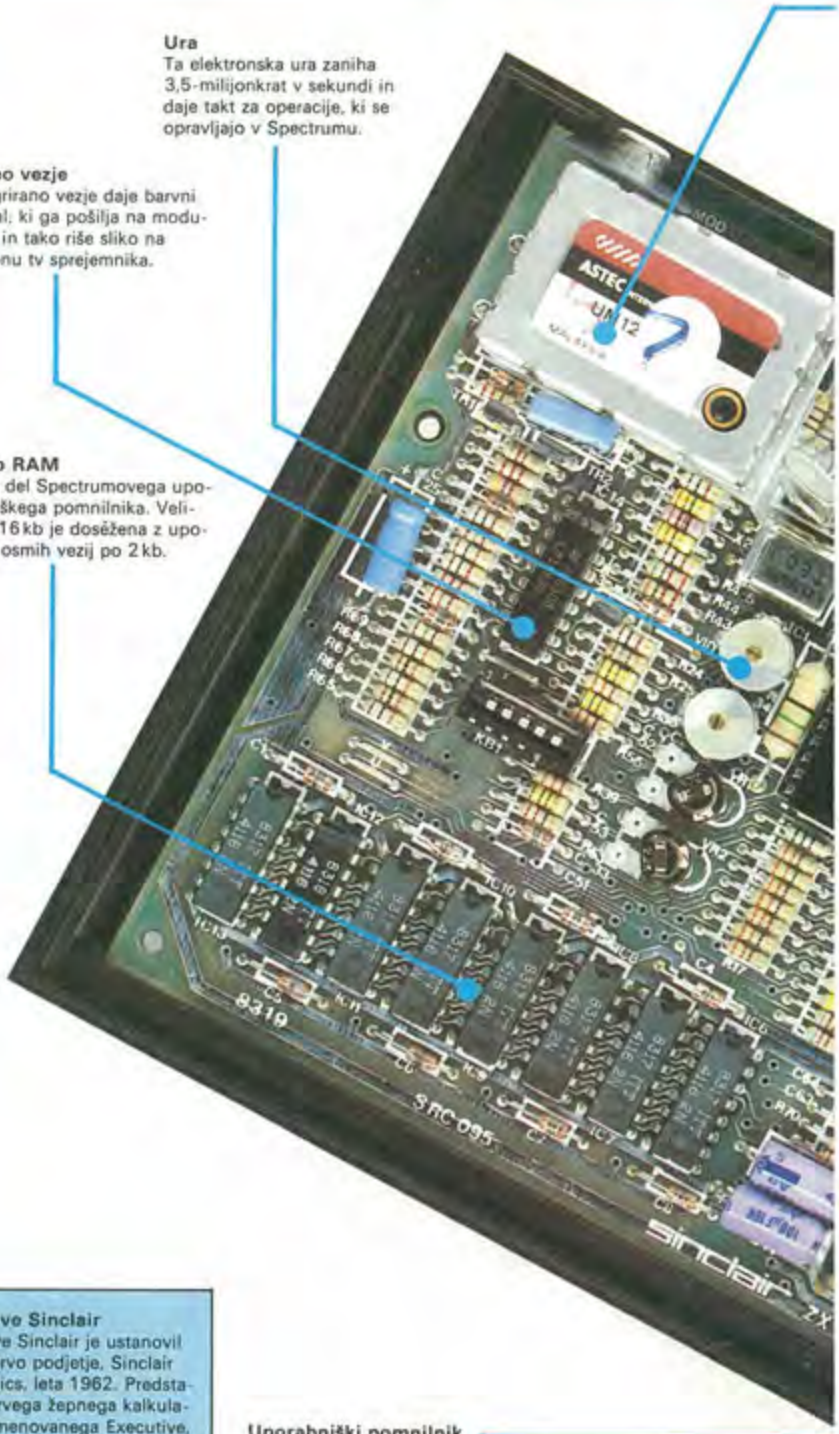
Sinclairova tipkovnica

Spectrum ima ceneno tipkovnico z omejenim gibanjem tipk. Sinclair se je v primerjavi z ravno tipkovnico ZX81 odločil za izboljšano verzijo, vendar brez dodatnih stroškov za gibajoče se tipke. Spectrumova tipkovnica je narejena v enem kosu iz gume, s tipkami vred, ki gledajo iz ohišja. Ko pritisnemo na tipko, se sklene stikalo pod njo. Računalnik tako spozna pritisnjeno tipko in napiše ustrezen znak ali izpolni ukaz. Ko tipko spustimo, se ta vrne v mirujočo lego zaradi prožnih gumijastih robov, ki se napnejo ko tipko pritisnemo.
Tak način izdelave je znižal ceno tipkovnice in prispeval svoje k nizki ceni Spectruma.

ZX Microdrive
Enoto ZX Microdrive priključimo na Spectrum z vmesnikom, ki ga nataknemo na zadnji del ohišja računalnika. Omogoča shranjevanje 100 kb podatkov, povprečni čas dostopa do podatkov pa je 3,5 sekunde.



ZX tiskalnik
ZX tiskalnik priključimo naravnost na Spectrum. Na 10 cm — 3 vrstice na centimeter — natisne okoli 35 vrstic teksta. Tiska tudi grafične znake. Z ukazom COPY lahko natisnemo na papir celotno vsebino zaslona.



Ura
Ta elektronska ura zaniha 3,5-milijonkrat v sekundi in daje takt za operacije, ki se opravljajo v Spectrumu.

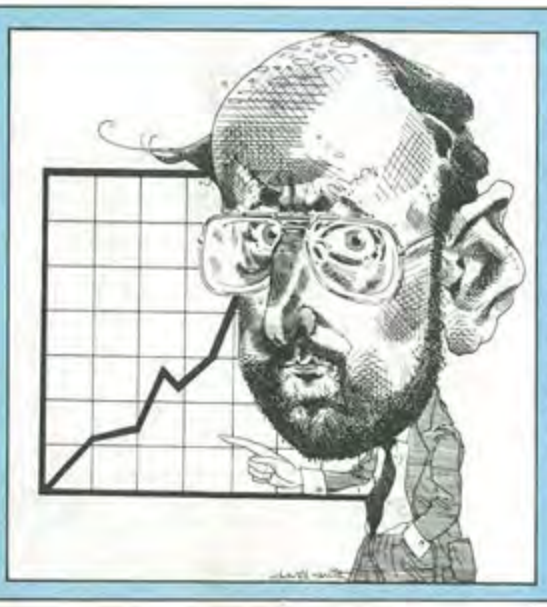
Video vezje
Integrirano vezje daje barvni signal, ki ga pošilja na modulator in tako riše sliko na zaslonu tv sprejemnika.

16 kb RAM
To je del Spectrumovega uporabniškega pomnilnika. Velikost 16 kb je dosežena z uporabo osmih vezij po 2 kb.

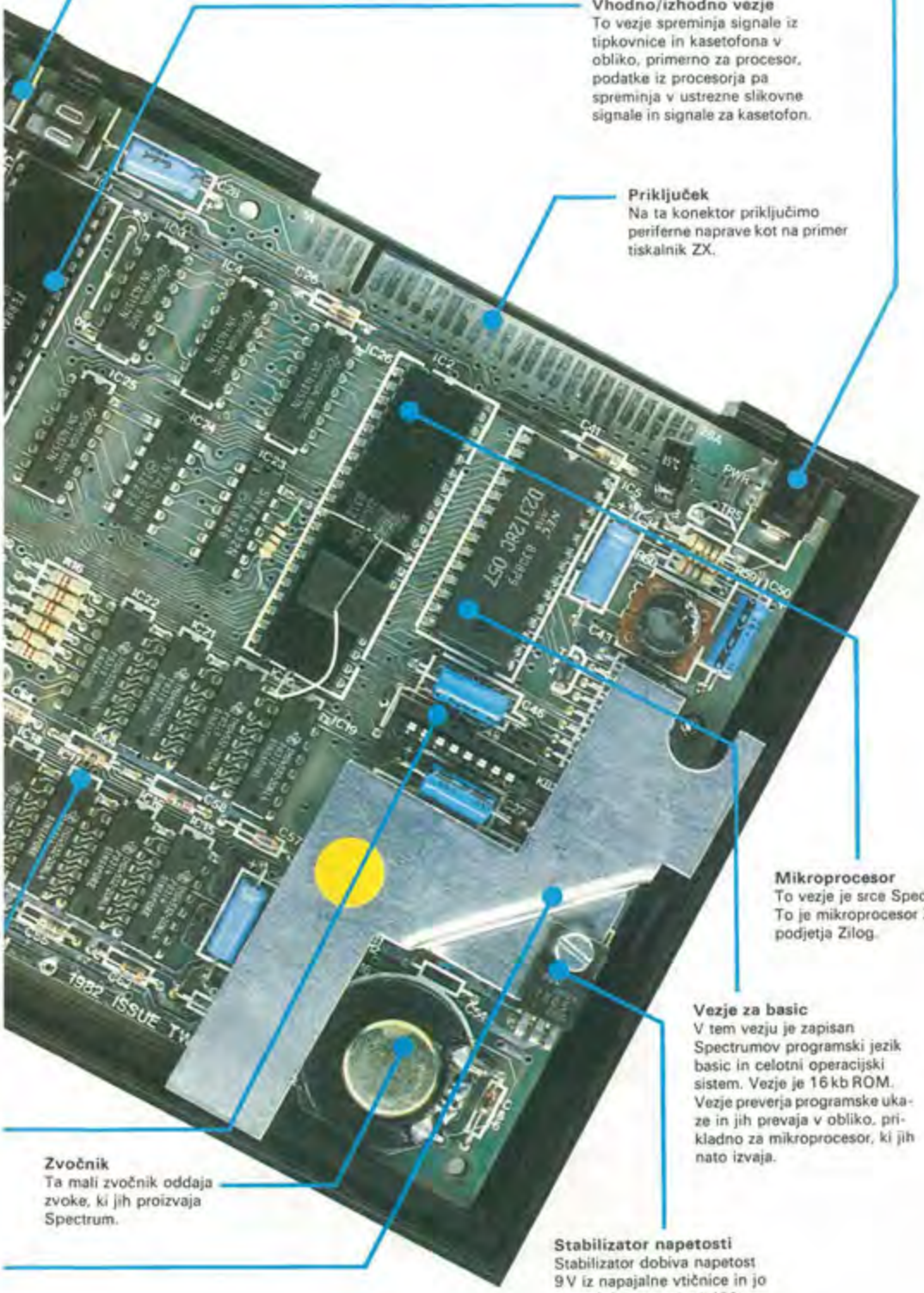
Uporabniški pomnilnik
Ta pomnilnik je v računalniku namenjen za shranjevanje programov in podatkov. Manjša verzija Spectruma ima 16 kb, večja pa 32 kb uporabniškega pomnilnika.

Priključek za tipkovnico
Tukaj se tipkovnica priključi na računalnik.

Hladilna površina
Velika aluminijasta plošča oddaja odvečno moč kot toploto. Spectrum se po daljšem času uporabe segreje, kar pomeni, da hladilna površina opravlja svoje delo.



Sir Clive Sinclair
Sir Clive Sinclair je ustanovil svoje prvo podjetje, Sinclair Radionics, leta 1962. Predstavitev prvega žepnega kalkulatorja, imenovanega Executive, je potrdila njegov občutek za miniaturizacijo in oblikovanje popularnih izdelkov, kakor tudi njihovo množično prodajo. Leta 1979 je sir Clive zapustil Sinclair Radionics in ustanovil Sinclair Research. Leta 1980 je razvil ZX80, ki mu je naslednje leto sledila izpopolnjena verzija, ZX81. Oba sta bila monokromatska računalnika, že leto 1982 pa je prineslo ZX Spectrum. Leta 1983 je Sinclair ustanovil lastni raziskovalni center v Cambridgeu.



Modulator
Modulator dobiva barvni signal z video vezja in ga spremeni v obliko, ki jo tv sprejemnik lahko pokaže na zaslonu.

Vtičnici EAR in MIC
Prek teh dveh vtičnic priključimo kasetofon na računalnik za prenos podatkov med računalnikom ter kasetofonom in narobe.

Vhodno/izhodno vezje
To vezje spreminja signale iz tipkovnice in kasetofona v obliko, primerno za procesor, podatke iz procesorja pa spreminja v ustrezne slikovne signale in signale za kasetofon.

Priključek
Na ta konektor priključimo periferne naprave kot na primer tiskalnik ZX.

Vtičnica za napajanje
Prek te vtičnice dobiva Spectrum energijo za delovanje iz 9-voltnega napajalnika.

Mikroprocesor
To vezje je srce Spectruma. To je mikroprocesor Z80A podjetja Zilog.

Vezje za basic
V tem vezju je zapisan Spectrumov programski jezik basic in celotni operacijski sistem. Vezje je 16 kb ROM. Vezje preverja programske ukaze in jih prevaja v obliko, primerno za mikroprocesor, ki jih nato izvaja.

Zvočnik
Ta mali zvočnik oddaja zvoke, ki jih proizvaja Spectrum.

Stabilizator napetosti
Stabilizator dobiva napetost 9 V iz napajalne vtičnice in jo spreminja v napetost, ki je primerna za delovanje elektronskih vezij v računalniku.

Sinclair Spectrum

CENA
99,95 funta za model 16 kb

VELIKOST
232 X 144 X 30 mm

TEŽA
552 g

CPU
Z80A

TAKT
3,5 MHz

POMNILNIK
16 kb RAM, razširjen na 48 kb
16 kb ROM vsebuje basic

ZASLON
24 vrstic po 32 znakov ali grafika z visoko ločljivostjo s 192 × 256 točkami.
V obeh primerih 8 barv.

PRIKLJUČKI
Tv priključek, priključek za kasetofon (brez daljinskega upravljanja), konektor z 28 priključki za periferne naprave.

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Forth, pascal, lisp, logo, prolog

OPREMA
Napajalnik, antenski kabel, kabel za kasetofon, kaseta za demonstracijo, dva priročnika.

TIPKOVNICA
40 premičnih tipk na skupni gumijasti podlagi.

DOKUMENTACIJA
Spectrumu je priložen tanek začetni priročnik za priključitev računalnika in znatno obsežnejši priročnik za programiranje v basicu. Slednji se začenja z zelo natančnim opisom tipkovnice in funkcij posameznih tipk.
Poglavja o programiranju v basicu kažejo Spectrumove zmožnosti s številnimi zgledi posameznih ukazov in z nekaj kratkimi programi.
Vrsta dodatkov podaja precej popoln referenčni priročnik za Spectrum in njegov basic. Oba priročnika sta dobro napisana in podajata celoten pregled podatkov, kako priključiti Spectrum, in vse o njegovih zmožnostih.

Vprašanja in odgovori

Pogosto si zastavljamo vprašanja s področja računalništva, vendar ustrezne odgovore le redkokdaj najdemo v revijah in priročnikih

Ali lahko hišni računalnik rabimo še za kaj drugega kot za igranje?
Hišne računalnike lahko uporabljamo za vodenje obrti, za hišno računovodstvo, za urejanje besedil, za računanje pri konstruiranju, za učenje... Uporaba je torej prepuščena naši iznajdljivosti, vendar pa lahko vsa ta področja uporabe na kratko označimo kot: obdelava števil in obdelava besed ter hranjenje in prikazovanje podatkov.

Računalniki so vedno manjši in vedno cenejši. Se bo ta razvoj kdaj ustavil?
Z razvojem tehnologije se računalniki resda nenehno manjšajo. Toda manjša se lahko sama notranjost računalnika, kajti tipkovnica mora ostati tako velika, da lahko s prsti tipkamo po njej. Dokler bo torej tipkovnica ostala poglavitni način sporazumevanja z

računalnikom, se računalniki ne bodo mogli še naprej manjšati. Kar se pa tiče cen, lahko rečemo, da so te danes že tako nizke, da si ne moremo predstavljati, da bi se še znižale.

Je učenje basica težavno?
Basic ni težek jezik. V primerjavi z naravnimi jeziki, ki smo se jih pač vsi naučili, je basic skromen jezik in ima toga pravila. Veliko enostavneje se naučimo basic kot tuji jezik; toda to še ne pomeni, da je pisanje dolgih in zapletenih programov enostavno.

Kdaj naj se lotim učenja basica?
Ko ugotovite, da vam je znanje basica potrebno. Morda se boste v celoti zadovoljili s programi, ki jih lahko dobite pri drugih; v tem primeru se vam seveda ni treba učiti basica. Na žalost pa gotovi programi ne zmorejo točno tega, kar potrebujete. Že z nekoliko poznavanja basica se dajo tuji programi prilagoditi vašim potrebam, toda če hočete z vašim računalnikom početi točno to, kar potrebujete, potem se morate čimprej naučiti programiranja v basicu.

Monitor je dražji kot televizor. Se nam dodatni stroški obrestujejo?
Monitor daje veliko jasnejšo sliko kot televizor. Če nameravate z računalnikom presedeti dalj časa pred zaslonom, vas ob monitorju ne bo bolela glava, ker boste besedilo brali z lahkoto. Tudi grafika je veliko lepša na monitorju.

Koliko elektrike porabi hišni računalnik?
Manj kot 60-vatna žarnica.

Bral sem, da se lahko otroci ob računalniku tudi učijo. Je to res?
Da. Računalniki bodo igrali vedno pomembnejšo vlogo v izobraževanju, ne le otrok ampak tudi odraslih. Za to pa so seveda potrebni primerni programi. Teh je v zahodnih državah že veliko. Posebno zanimiva je uporaba računalnikov v izobraževalne namene s programiranjem v jeziku logo. Z logom otroci raziskujejo tako imenovani »mikrosvet«, izvajajo poskuse in odkrivajo, kaj se dogaja. Otroci se torej učijo s programiranjem računalnika.

Nekateri računalniki, kot npr. Spectrum, imajo gumijaste tipke, drugi, kot npr. Commodore, pa imajo »pravo« tipkovnico. Je to bistveno?
Bistvene razlike ni, razen če znate tipkati slepo, ko lahko s pravo tipkovnico delate veliko hitreje kot z gumijasto.

Commodore 64

Imenitna grafika in hi-fi zvok — to je osebna izkaznica hišnega računalnika Commodore 64

Commodore 64 je dejansko prvi od nove generacije hišnih računalnikov, ki ni namenjen zgolj za domačo rabo in igre, temveč je enako uspešen tudi na poslovnem področju.

Zares imenitna grafika in 64 kb pomnilnika je dovolj celo za tako resno delo, kakor so npr. razne preglednice, mrežni diagrami, urejanje besedila, podatkovne baze in podobno. Mogoče je uporabiti tudi nekaj programske opreme sorodnih računalnikov, kakor sta Commodore VIC-20 in poslovni sistem PET.

Na voljo je cela vrsta vmesnikov, ki omogočajo uporabo večine standardne strojne opreme z VIC tiskalnikom in diskovno enoto vred. Danes se tak sistem, ki sestoji iz računalnika, tiskalnika in diskovne enote, prodaja po sprejemljivi ceni 700 funtov, kar je kot nalašč za delo v manjših podjetjih ali pri zasebnikih.

Dva od najmočnejših adutov računalnika Commodore 64 sta tako imenovana »sprite« (izg. sprajt) grafika velike ločljivosti in popolna sinteza zvoka. Sprite je ime za grafični lik ali objekt, ki ga na zaslonu ustvarimo z grafiko, to je z delom programske opreme, ki omogoča risanje, barvanje ipd. Tak lik (glej stran 136) je celota zase in ga lahko po zaslonu premikamo s preprostimi ukazi, npr. s pritiskom na določeno tipko. So kot nalašč za razne igre, kjer nastopajo v vlogah vozil, letal, vesoljskih ladij in podobno.

Take like je mogoče ustvarjati tudi brez te grafike, vendar to zahteva veliko več programiranja.

Na tem računalniku je mogoče te like tudi povečevati, zmanjševati, jim spreminjati barve in jih voditi pred drugimi, sorodnimi liki ali statičnimi grafičnimi predmeti, kot je ozadje prizorišča, ali za njimi ter tako pričarati pravo prostorsko sliko. Mogoče je določiti celo, kdaj dva taka lika trčita. To bi bilo lahko tudi znak za eksplozijo.

Sinteza zvoka je prav tako dognana, resnično nasprotje »krakanja« ali preprostih piskajočih signalov

Tipkovnica računalnika Commodore 64

Commodore 64 ima odlično tipkovnico s tipkami, ki so posebej oblikovane tako, da je tipkanje lažje. Poleg običajnih je na voljo še cela vrsta grafičnih znakov. Večnamenska tipka s Commodorejevim znakom poleg drugega tudi izbira med posameznimi vrstami znakov. Barve spreminjamo s tipko CTRL (CONTROL) in eno od tipk v gornji vrsti tipkovnice.
S posebnim programom lahko priredimo določene naloge štirim funkcijskim tipkam na desni strani tipkovnice in si tako ustvarjamo bližnjice za programiranje.

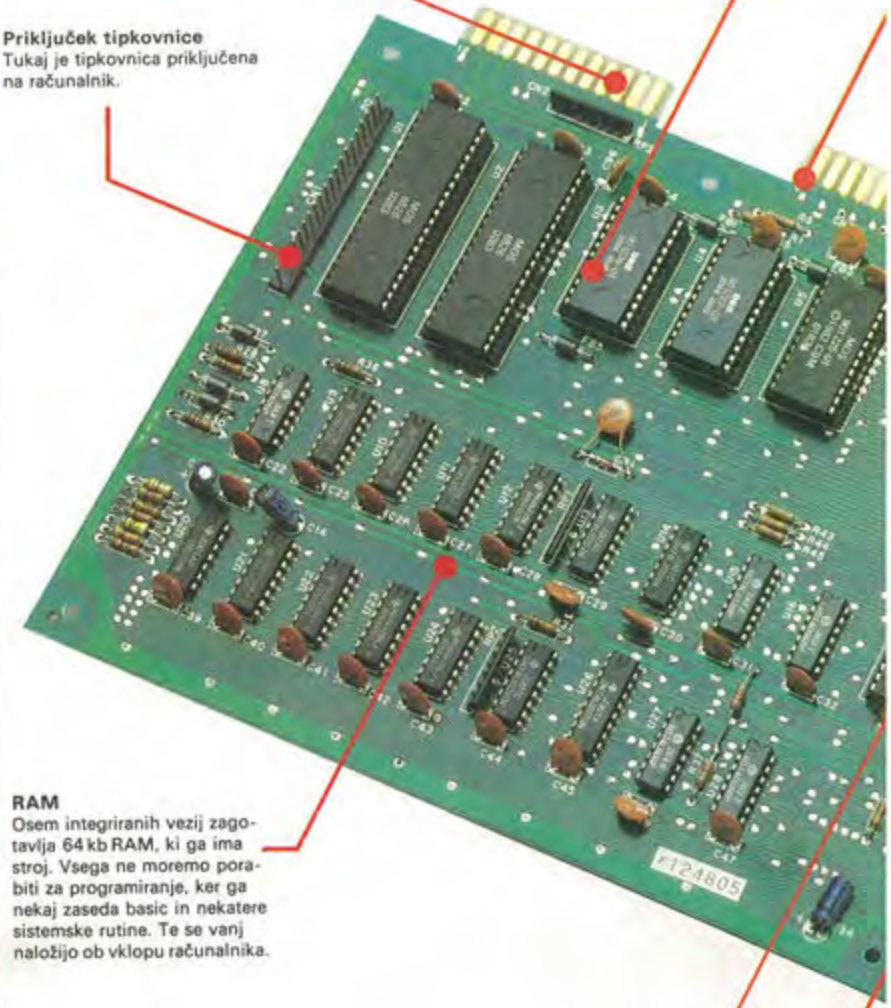
Kasetni magnetofon 1530 C2N
To je Commodorejeva lastna standardna in zanesljiva kasetna enota, izdelana tako, da jo priključimo neposredno na računalnikov priključek. Kasetofon se napaja kar iz računalnika. Računalnik obenem tudi krmili delovanje elektromotorja kasetne enote.

Diskovna enota 1540
To je enota za eno disketo, ki jo na računalnik priključimo s serijskim priključkom. Velikost informacije, ki jo lahko shranimo na 5 1/4 palca veliko disketo, znaša 170 kb. Čas, potreben za dostop do katerekoli shranjene informacije, je manjši od dveh sekund.

Tiskalnik 1525
Ta matrični tiskalnik lahko natiska prav vse znake, ki jih »64« prikazuje na zaslonu, tudi grafične. Priključimo ga s serijskim priključkom. Hitrost tiskanja je 30 znakov v sekundi.

Uporabnikov priključek
S tem 24-polnim priključkom lahko priključimo najrazličnejše naprave. Razne vrste sponk tega priključka lahko uporabljamo kot vhodne ali izhodne linije.

Priključek tipkovnice
Tukaj je tipkovnica priključena na računalnik.



RAM
Osem integriranih vezij zagotavlja 64 kb RAM, ki ga ima stroj. Vsega ne moremo porabiti za programiranje, ker ga nekaj zaseda basic in nekatere sistemske rutine. Te se vanj naložijo ob vklopu računalnika.

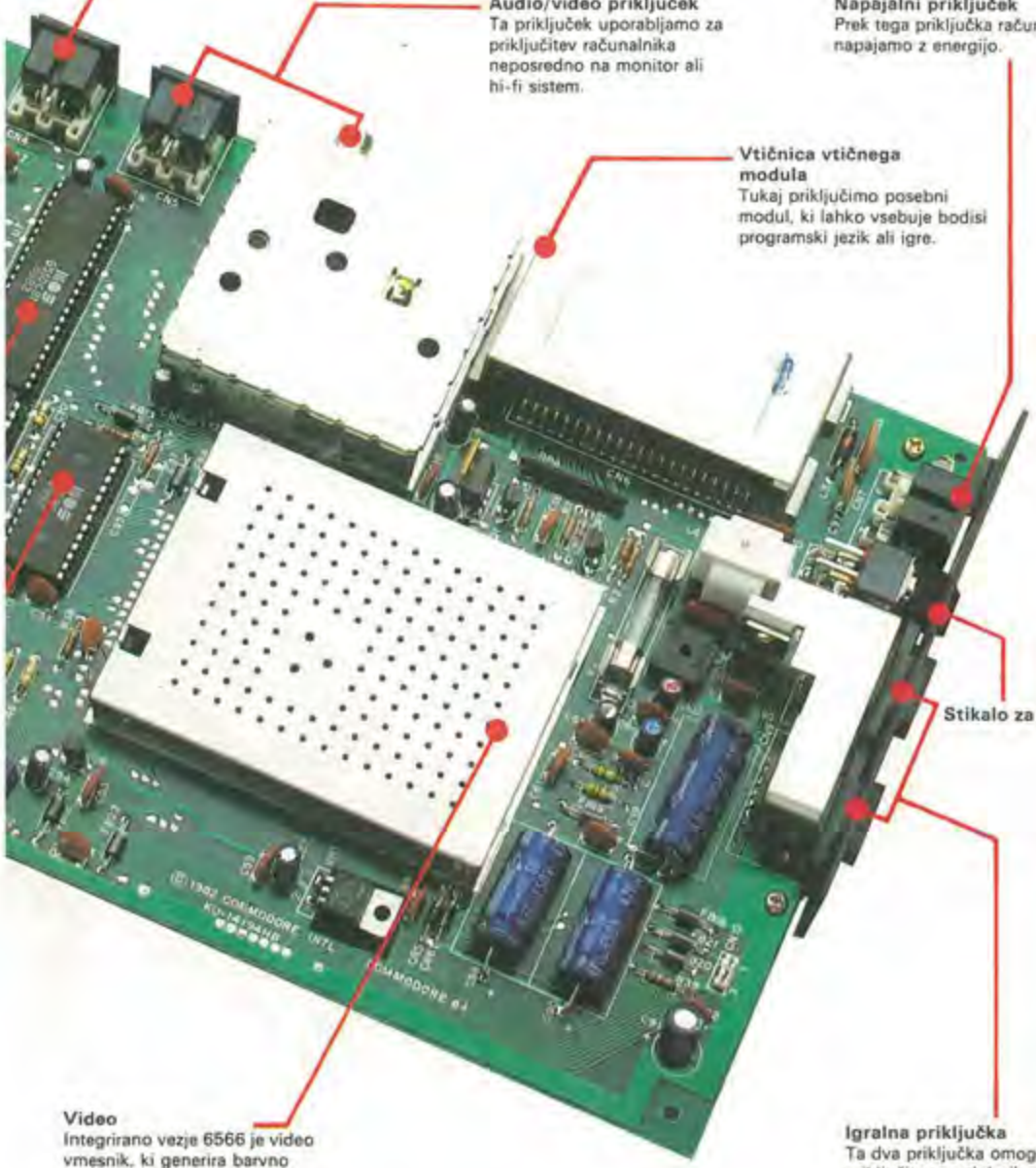
Mikroprocesor
Glavni procesor je integrirano vezje 6510. Kontrolira vse operacije v računalniku in tudi večino vhodno/izhodnih operacij.

Zvok
Integrirano vezje 6581 je vmesnik, ki generira ves zvok računalnika.

podobnih cenejših strojev. Poleg posamičnih tonov so mogoči tudi akordi in harmonije (do treh hkratnih tonov). Na voljo so še ukazi za kontrolo drugih parametrov igrane note. Z drugimi besedami, Commodore 64 lahko posnema veliko število glasbil, pa tudi šume, poke in podobne zvoke.

Šibka točka računalnika je njegov programski jezik, najbrž še vedno tista izvedenka basica, ki ga je Commodore uporabljal pri svojih prvih računalnikih že pred mnogimi leti. Namesto primerne množice priročnih ukazov, ki bi zares izkoristili vse prednosti sicer izredno zmogljivega računalnika, moramo veliko zahtevnih operacij izvajati z nepriljubljenima ukazoma PEEK in POKE.

Rešitev je v uporabi drugih programskih jezikov ali izvedenk, pri tem je odlični Simonov basic na voljo tudi v obliki posebnega priključnega modula.



Commodore 64

CENA
229 funtov

VELIKOST
404 × 216 × 75 mm

TEŽA
1820 g

TAKT
1 MHz

POMNILNIK
Vsega 64 kb. 20 kb ROM vsebuje operacijski sistem in basic. Če ne uporabljamo basicovega interpreterja, lahko izkoriščamo do 54 kb RAM.

ZASLON
25 vrstic s po 40 znaki. Visoka ločljivost s 320 × 200 točkami. 16 barv.

PRIKLJUČKI
Priključki za kasetofon, tv, monitor, uporabniški, za priključeni modul in RS 232.

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Forth, comal, pilot, logo, UCSD, pascal in druge izvedenke basica.

OPREMA
Napajalna enota, antenski kabel, navodila.

TIPKOVNICA
Qwerty, kot pri pisalnem stroju z 62 tipkami in štirimi funkcijskimi tipkami.

DOKUMENTACIJA
»User's Guide« je eno revnejših Commodorejevih navodil. Opisuje sicer napravo, delovanje in jezik, vendar ne tako, da bi bilo to začetniku povsem razumljivo. Ni priročnega, zgoščenega pregleda o naslavljanju posebnih registrov, ki jih potrebujemo za programiranje grafike in zvoka. To je vsekakor velika pomanjkljivost. Zato je za tiste, ki niso popolni začetniki, morda najboljše prezreti »User's Guide« in uporabiti raje »Commodore 64 Programmer's Reference Guide«, priročnik, ki ga je izdala Commodore Business Machines Inc. and Howard W. Sams.

Dragon 32

Računalnik z izboljšano grafiko je zasnovan na novih integriranih vezjih

Dragon 32 se je pojavil na angleškem tržišču konec leta 1982 in že na začetku dosegel lep uspeh. To sta mu omogočila sorazmerno zmogljiv delovni pomnilnik z 32 kb in pa lasten prevajalnik za Microsoft basic.

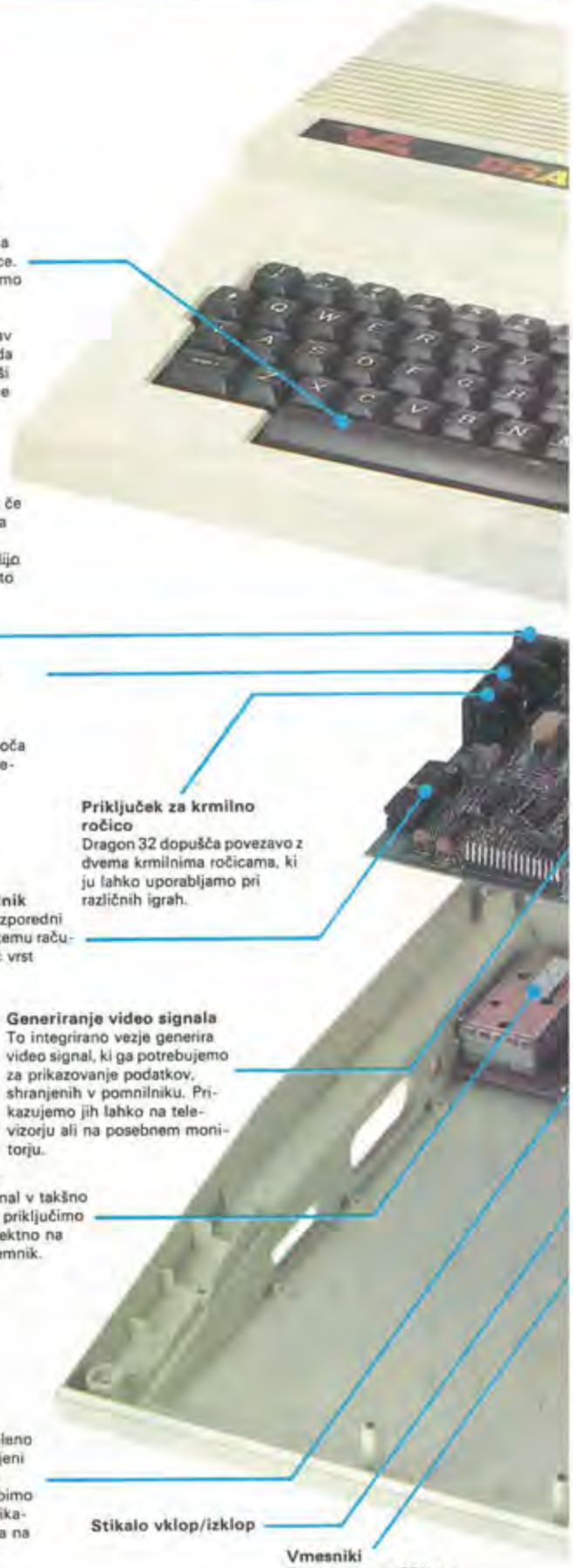
Dragon 32 se močno ujema z računalnikom Tandy Color, tako da lahko uporablja vse njegove dodatke. Prav tako lahko uporablja nekatere igre na vtičnih modulih, ne more pa žal uporabljati mnogih iger na kasetah, namenjenih za nekatere druge računalnike. Oba omenjena računalnika (Dragon 32 in Tandy) namreč uporabljata enak mikroprocesor 6809E, ki ga izdeluje Motorola. Glede tega se ločita od večine hišnih računalnikov, ki uporabljajo največkrat mikroprocesorje 6502 ali pa Z80.

Glede grafike ima ta računalnik mnoge prednosti pred tekmeci. Oblikujeta ga tako boljša ločljivost, kakor tudi posebni ukazi, kot npr.: DRAW, CIRCLE, PAINT, COLOUR in MOVE. S temi ukazi lahko preprosto rišemo kroge in razne druge like, jih premikamo in tudi barvamo z različnimi želenimi barvami.

Dragon lahko zaigra hkrati samo en zvok; to je v primeri z drugimi računalniki, ki lahko zaigrajo več tonov hkrati (akorde), nekoliko skromno. Posebni ukazi v basicu pa omogočajo, drugače kakor pri drugih računalnikih, izredno lahko programiranje razpoznavnih melodij. Čeprav ima Dragon 32 precej vmesnikov, je razen krmilne ročice na voljo malo dodatne opreme.

Predvideno povečanje delovnega pomnilnika na 64 kb bo omogočilo uporabo računalnika za poslovne namene in ga postavilo ob bok takim računalnikom, kakor je recimo Commodore 64.

Dragonova diskovna enota
Ta enota je sestavljena iz dveh delov: iz kontrolnega vezja, ki ga priključimo na računalnik, in glavne škatle, v kateri je diskovna enota.



Tipkovnica
Dragon 32 ima tipkovnico podobno pisalnemu stroju. Tipki za pomik gor/dol in levo/desno sta postavljeni dokaj nenavadno, in sicer na levi in desni strani tipkovnice. S tipko BREAK lahko ustavimo izvajanje programa, tipka CLEAR pa rabi za brisanje podatkov na zaslonu. Čeprav je oblika računalnika taka, da lahko nanj postavimo manjši televizor, je bolje, če tega ne počnemo.

Tipka za resetiranje
S pritiskom na to tipko dosežemo enak učinek, kot če bi računalnik izključili ter ga ponovno vključili. Pogosti vklopi in izklopi lahko škodijo napajalniku računalnika, zato je boljše, če za resetiranje uporabljamo tipko.

Priključek za kasetofon
Na tem priključku lahko priključimo običajen magnetofon. Basic, ki ga uporablja Dragon 32, omogoča kontrolo nad motorjem kasetofona.

Priključek za krmilno ročico
Dragon 32 dopušča povezavo z dvema krmilnima ročicama, ki ju lahko uporabljamo pri različnih igrah.

Priključek za pisalnik
Standardni 8-bitni vzporedni priključek omogoča temu računalniku uporabo več vrst pisalnikov.

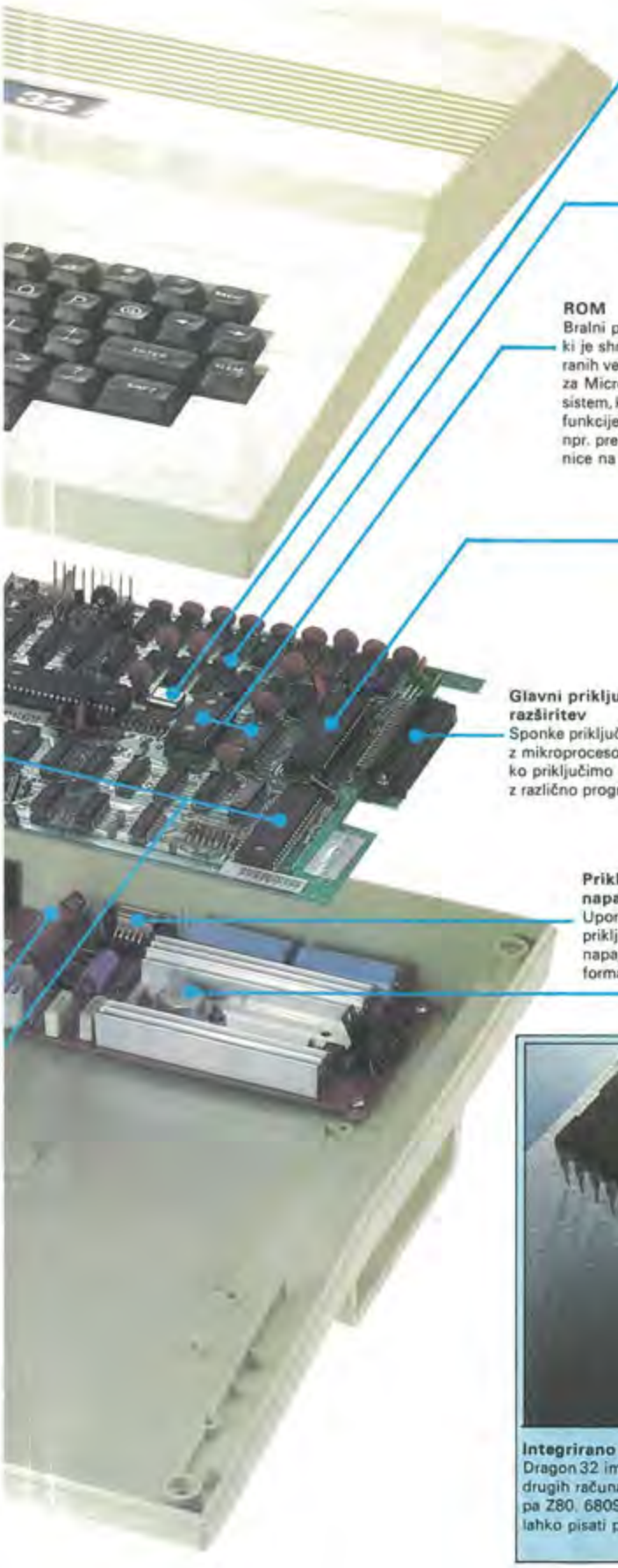
Generiranje video signala
To integrirano vezje generira video signal, ki ga potrebujemo za prikazovanje podatkov, shranjenih v pomnilniku. Prikazujemo jih lahko na televizorju ali na posebnem monitorju.

VF modulator
Priredi video signal v takšno obliko, da lahko priključimo njegov izhod direktno na televizijski sprejemnik.

Video priključek
Ločeni signali za rdečo, zeleno in modro barvo so namenjeni za priključitev na ustrezen monitor. Na monitorju dobimo namreč boljšo kvaliteto prikazovanja podatkov kakor pa na televizorju.

Stikalo vklop/izklop

Vmesniki
Povezavo med CPU ter tipkovnico, kasetofonom, pisalnimi ali drugimi zunanjimi napravami nam omogočata dve integrirani vezji 6821 (Peripheral Interface Adaptor — PIA).



Kremenčeva kristala
Kremenčeva kristal dajeta signale za sinhronizacijo posameznih enot računalnika. En kristal določa hitrost delovanja mikroprocesorja, drugi pa krmili v glavnem vezja za video signale.

RAM
Dragon 32 ima 32 kb delovni pomnilnik, ki ga sestavlja 16 integriranih vezij, od katerih ima vsako po 2 kb pomnilnika.

ROM
Bralni pomnilnik računalnika, ki je shranjen v dveh integriranih vezjih, vsebuje interpreter za Microsoft basic operacijski sistem, ki uravnava vse notranje funkcije računalnika, kot je to npr. prenos podatkov s tipkovnice na zaslon itd.

Mikroprocesor
Motorolin mikroprocesor 6809E je hitrejši od mnogo bolj popularnega 6502, vendar je zanj napisane nekoliko manj programske opreme. Enak mikroprocesor je uporabljen tudi v računalniku Tandy Color — tudi zato se dobro ujemata.

Glavni priključek za razširitev
Sponke priključka so povezane z mikroprocesorjem. Nanj lahko priključimo vtične module z različno programsko opremo.

Regulator napetosti
Z zunanjim transformatorjem znižamo vhodno napetost na vrednost, primerno za vezja v notranjosti računalnika. Napetostni regulatorji s cilindričnimi kondenzatorji skrbijo za odstranjevanje motenj pri napajanju. Postavljeni so na velikih železnih hladilnikih, ki preprečujejo pregrevanje.

Priključek za napajanje
Uporablja se za priključitev na napajalni transformator.

Integrirano vezje 6809E
Dragon 32 ima vgrajen 8-bitni mikroporcesor 6809E, večina drugih računalnikov pa ima vdelan mikroprocesor 6502 ali pa Z80. 6809E je zlasti cenjen pri programerjih, ker je zanj lahko pisati programe v strojni kodi.

Dragon 32

CENA
175 funtov v Veliki Britaniji

VELIKOST
380 × 325 × 97 mm

TEŽA
2,1 kg

TAKT
1 MHz

POMNILNIK
V temeljni izvedbi je 32 kb RAM, ki pa ga lahko dopolnimo na 64 kb (vendar je od tega samo 48 kb dosegljivih pri programiranju v basicu). Ima še 16 kb ROM za Microsoft basic in za operacijski sistem.

ZASLON
Prikaz na zaslonu vsebuje 16 vrstic po 32 karakterjev. Ima 8 barv. Maksimalna grafična ločljivost je 256 × 192 točk.

PRIKLJUČKI
Ima priključek za kasetofon, televizijski monitor, vtični modul, vzporedni priključek za pisalnik ter dva priključka za krmilni ročici.

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Drugih jezikov ni moč uporabljati.

OPREMA
Napajalnik in navodilo za uporabo.

TIPKOVNICA
Podobna kakor pri pisalnem stroju, ima 53 tipk.

DOKUMENTACIJA
Priročnik za uporabo računalnika, ki ga je napisal načrtovalec Richard Wadman, je bogato opremljen.
Tiskan je v barvah in vsebuje obilo koristnih nasvetov.
Basic je opisan od najpreprostejših ukazov do dela z grafiko in zvokom. Sledi niz dodatkov, med njimi tudi koristna navodila o pripravi grafičnih podatkov pred pisanjem programa.
Na voljo je tudi več knjig o Dragonu 32; vsebujejo tako način programiranja kakor tudi izpise nekaterih programov.

Vprašanja in odgovori

Ali bodo imeli računalniki čustva?
Računalniki nimajo in tudi ne bodo imeli čustev. Izredno zanimivo vprašanje pa je, zakaj ne. Današnji računalniki niso inteligentni — sami torej ne morejo misliti. Ni znano, koliko časa bo potrebno, da bodo strokovnjaki izdelali računalnik, ki bo znal misliti, verjetno pa tisti, ki sedaj živimo, tega ne bomo dočakali. Mnenje nekaterih raziskovalcev je, da je ustvarjalni miselni proces neločljiv od čustev. Računalniki, ki bodo lahko mislili, bodo glede na to mnenje računalniki s čustvi.

Kakšna je razlika med računalnikom in robotom?
Robot je mehanski podaljšek računalnika; roke in oči sledijo navodilom, ki jih pošiljajo možgani računalnika. Roboti pomagajo pri izdelavi današnjih avtomobilov, stereo in različnih drugih naprav, ki imajo vdelane tudi mikroračunalnike. Toda roboti so še vedno precej okorni ob srečanju z nepredvidenim položajem, saj v takšnem primeru kratko malo ne vedo kaj storiti. Roboti jutrišnjega dne bodo imeli vdelane bolj zmogljive računalnike. Takšne, ki bodo imeli omejeno inteligenco, bomo imeli prav kmalu.

Zakaj je nekatera programska oprema tako draga?
Programi za igre so poceni, medtem ko so poslovni programi pogosto silno dragi. Pisanje poslovnih programov, še zlasti takšnih, ki so temeljito preskušeni, opravlja skupina dobro plačanih programerjev mesece in mesece, včasih tudi leta. Da se povrne denarna investicija in da je še nekaj dobička, morajo podjetja prodajati svoje izdelke po ceni, ki jim zagotavlja pokritje stroškov. Računalniške igre pa prodajajo v več sto ali celo v več tisoč kopijah, tako da jim že sorazmerno nizka cena zagotavlja dobiček. Če je potencialno tržišče omejeno, je prodajna cena precej višja. Mnogi programi so zelo specializirani. Za zgled vzemimo program za tiskalnik, ki nam ureja podatke, ki jih želimo hitro in točno izpisati. Za takšen program je potencialno tržišče omejeno na skupno število tiskalnikov v državi. Če vložimo v takšen program veliko denarja, se nam bo povrnil šele, ko bo prodanih več tisoč takšnih programov.

Pravijo, da bodo zaradi integriranih vezij na siliciju milijoni brezposelni. Ali bomo postali zaradi mikroračunalnikov odveč?
Dolgoročne družbene posledice zaradi razvoja mikroračunalnikov je težko predvideti, toda postalo je jasno, da smo na začetku nove industrijske revolucije. Računalniki, še zlasti miniaturni in ceneni mikroračunalniki, ki so vdelani v mehanske robote, lahko zamenjajo marsikatero drago in zamudno ročno delo. Celo dela, za katera je potrebna precejšnja spretnost, niso več varna. Knjigovodstvo in računovodstvo lahko že upravljamo z računalnikom. Tudi stavni stroji za časopise in knjige so že direktno povezani z računalnikom. Z računalnikovo tipkovnico lahko oblikujemo besedilo, natisnjeno v časopisu ali knjigi. Računalnik izredno natančno in hitro opravlja zapletene aritmetične operacije. Tudi roboti opravljajo zapletena mehanska dela tako dobro, da je za taka dela potrebnih manj delavcev.

Ali lahko z računalnikom oropamo banko? Ali lahko z njim začnemo tretjo svetovno vojno?
Odkar lahko računalnik komunicira prek vsake telefonske linije, je teoretično mogoče, da se priključi v banki na centralni računalnik in ukaže prenos denarja na vaš bančni račun. V praksi pa to ni tako preprosto. Banke uporabljajo sodobne metode varovanja podatkov, ki nepoklicanim preprečujejo dostop do informacij. Ta koda je dostopna samo pooblaščenim osebam in zanjo ne vedo niti bančni uslužbenci. Nekatere kode, uporabne za najbolj zaupne podatke, so tako dobro zavarovane, da bi za njihovo identifikacijo tudi najzmogljivejši računalnik porabil bilijone let. Vdreti v vojaški računalniški sistem pa je še težje.

Vojaški računalniki za medsebojne povezave večinoma ne uporabljajo javnih telefonskih linij. Mikrovalovne in satelitske linije so težko dostopne celo izvrstnim računalniškim strokovnjakom. Če uspe komu vzpostaviti mikrovalovno zvezo, kljub temu zelo težko pride do zaupnih informacij, saj je zaščitna koda v praksi še vedno nepremostljiva ovira.

BBC model B

To je računalnik angleških šol, tehnično zelo dognan in z veliko programske opreme

BBC model B izdeluje Acorn Computers v Cambridgeu s podporo BBC. Najprej so izdelovali dva modela računalnika, vendar so kasneje opustili cenejši in manj dognani model A.

Ta računalnik ima izjemen uspeh na tržišču; pri tem ne gre zanemariti dejstva, da ga je angleška vlada odobrila za pouk računalništva na šolah.

Kot šolski računalnik premore BBC zares izjemno količino programske opreme, posebej namenjene vzgoji in izobraževanju.

Tehnične lastnosti modela B veljajo še vedno za izjemne, boljše celo od mnogih mlajših tipov podobnih strojev. Ne moremo, da ne bi omenili basica, ki premore mnogo ukazov tudi za izvajanje posebnih funkcij, kar poenostavi tako razvijanje kakor tudi popravljanje programov.

Na voljo je osem različnih grafičnih načinov. To pomeni, da ima uporabnik možnost izbire grafike z nizko, srednjo ali pa visoko ločljivostjo, vendar je pri zadnji omejeno število razpoložljivih barv. Najboljša dosegljiva ločljivost znaša 640 × 256 točk.

Večina uporabnikov priključuje hišne računalnike na navaden tv sprejemnik, kjer taka dobra grafika ne pride do izraza. Boljša in seveda tudi dražja rešitev je uporaba ustreznega monitorja.

Grafični ukazi programskega jezika omogočajo risanje linij, krogov in konstruiranje najrazličnejših likov na zaslonu.

Videz naprave je preprost in ličen, saj je celo napajalna enota pod pokrovom ohišja. Število vmesnikov na zadnji, spodnji strani je večje kot pri večini podobnih strojev. Zato so možnosti za priključitev vrste zunanjih naprav velike.

Poleg standardnih vmesnikov za diskovno enoto, tiskalnik in celo analognih vhodov imamo še enega, ki omogoča povezovanje v mrežo. To je kot nalašč za rabo v učilnici, kjer si več uporabnikov lahko deli isti tiskalnik ali diskovno enoto.

Končno je v napravi še vmesnik, imenovan »The Tube«. Nanj lahko priključimo drug mikroprocesor in si s tem omogočimo bodisi hitrejše računanje ali pa uporabo programske opreme, pisane za druge vrste računalnikov. Krog uporabnikov, ki izkoriščajo to možnost, pa je ožji.

Tipkovnica BBC model B
Glede na izoblikovanost, obseg in kvaliteto konstrukcije je tipkovnica gotovo ena izmed odlik računalnika BBC model B. Tipke imajo pravšnjo obliko in razporeditev, tako da se takoj počuti domačega tudi tisti, ki piše na slepo. S tipkami, ki nosijo oznake v obliki puščic, kurzor premikamo po zaslonu in besedilu ali programu. V zgornji vrsti je deset rdečih programskih tipk, posebno koristnih pri šolskih programih, ker mora uporabnik samo izbrati pravi odgovor izmed desetih mogočih. Dobrodošla lastnost so tri signalne svetleče diode, ki kažejo, ali teče motor kasetofona in ali so aktivirane tipke »Capitals Locke«.

Diskovna enota
Enojna naprava, narejena posebej za BBC, je sicer privlačna, toda pretirano draga (260 funtov), poleg tega pa lahko shrani le 100kb podatkov. Ni dvoma, da je boljše ozreti se po podobnih enotah drugih proizvajalcev, cenejših in boljših, ki jih je mogoče priključiti tudi na računalnik BBC.

Analogni vhod
Analogni signal lahko na računalnik priključimo na tem mestu. Tak signal navadno daje kak senzor ali pa laboratorijski merilni instrument. Vhod se večinoma uporablja pri laboratorijskem in eksperimentalnem delu.

Video kontrolnik
To integrirano vezje spreminja digitalno informacijo iz uporabniškega pomnilnika v video signal.

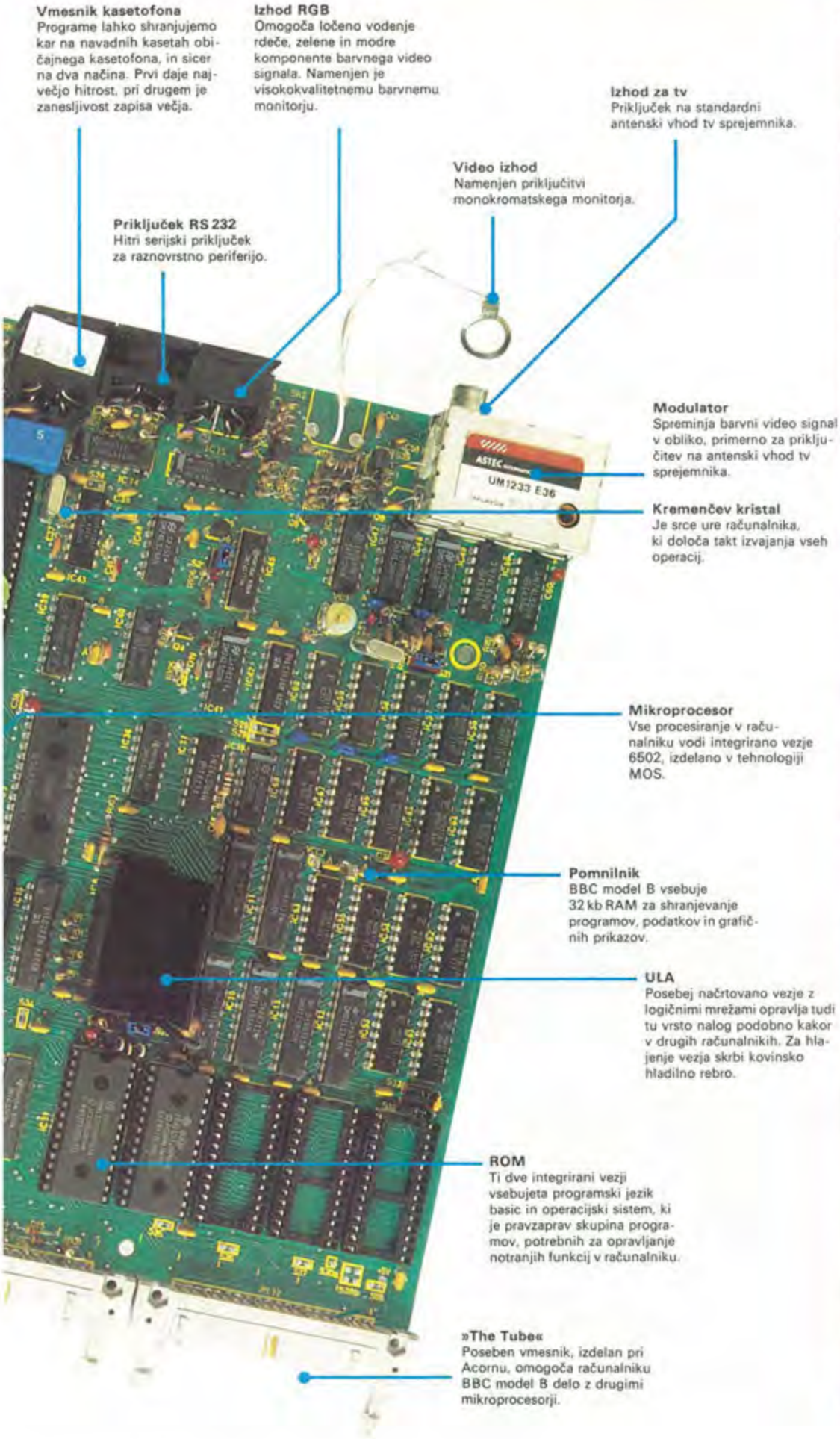
Integrirana vezja z vseh koncev
Pazljiv pogled v notranjost računalnika pove, iz katerih dežel so integrirana vezja. Naj jih naštejemo: Malezija, Japonska, Portugalska, Škotska in ZDA.

Vmesnik
Razni vmesniki, kakor je npr. ta 6522, integrirano vezje, izdelano v tehnologiji MOS, pošiljajo signale zunanjim napravam. Čeprav ta vezja po funkciji niso mikroprocesorji, so po sestavi prav tako zapletena.

Priključki periferije
Periferne enote priključimo na priključke na zadnji strani računalnika.

Priključek tiskalnika
Tukaj lahko priključimo katerikoli tiskalnik, ki uporablja vzporedni signal.

Uporabniški priključek
Namenjen je za eksperimentiranje z digitalnimi napravami in z doma izdelanimi logičnim vezji.



Vmesnik kasetofona
Programe lahko shranjujemo kar na navadnih kasetah običajnega kasetofona, in sicer na dva načina. Prvi daje največjo hitrost, pri drugem je zanesljivost zapisa večja.

Izhod RGB
Omogoča ločeno vodenje rdeče, zelene in modre komponente barvnega video signala. Namenjen je visokokvalitetnemu barvnemu monitorju.

Izhod za tv
Priključek na standardni antenski vhod tv sprejemnika.

Video izhod
Namenjen priključitvi monokromatskega monitorja.

Priključek RS 232
Hitri serijski priključek za raznovrstno periferijo.

Modulator
Spreminja barvni video signal v obliko, primerno za priključitev na antenski vhod tv sprejemnika.

Kremenčev kristal
Je srce ure računalnika, ki določa takt izvajanja vseh operacij.

Mikroprocesor
Vse procesiranje v računalniku vodi integrirano vezje 6502, izdelano v tehnologiji MOS.

Pomnilnik
BBC model B vsebuje 32 kb RAM za shranjevanje programov, podatkov in grafičnih prikazov.

ULA
Posebej načrtovano vezje z logičnimi mrežami opravlja tudi tu vrsto nalog podobno kakor v drugih računalnikih. Za hlajenje vezja skrbi kovinsko hladilno rebro.

ROM
Ti dve integrirani vezji vsebujeta programski jezik basic in operacijski sistem, ki je pravzaprav skupina programov, potrebnih za opravljanje notranjih funkcij v računalniku.

»The Tube«
Poseben vmesnik, izdelan pri Acornu, omogoča računalniku BBC model B delo z drugimi mikroprocesorji.

BBC Model B

CENA
399 FUNTOV

VELIKOST
409 × 358 × 78 mm

TEŽA
3,7 kg

CPU
6502A

TAKT
2 MHz

POMNILNIK
32 kb RAM
32 kb ROM, vsebuje basic in dognan operacijski sistem

ZASLON
8 različnih grafičnih načinov. Največje polje teksta: 32 vrstic po 80 znakov. Najboljša ločljivost: 640 × 356 točk. Pri nižji ločljivosti ima na voljo do 16 barv.

PRIKLJUČKI
Tv, monokromatski in barvni monitor, disk, tiskalnik, analogni vhod, uporabniški priključek in »The Tube« (za drug procesor).

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Lisp, forth, logo

OPREMA
Priključni kabli za kasetofon in tv, navodila za uporabo, demonstracijska kaseta z brošuro.

TIPKOVNICA
V stilu pisalnega stroja s 74 tipkami, med njimi 10 programskih tipk.

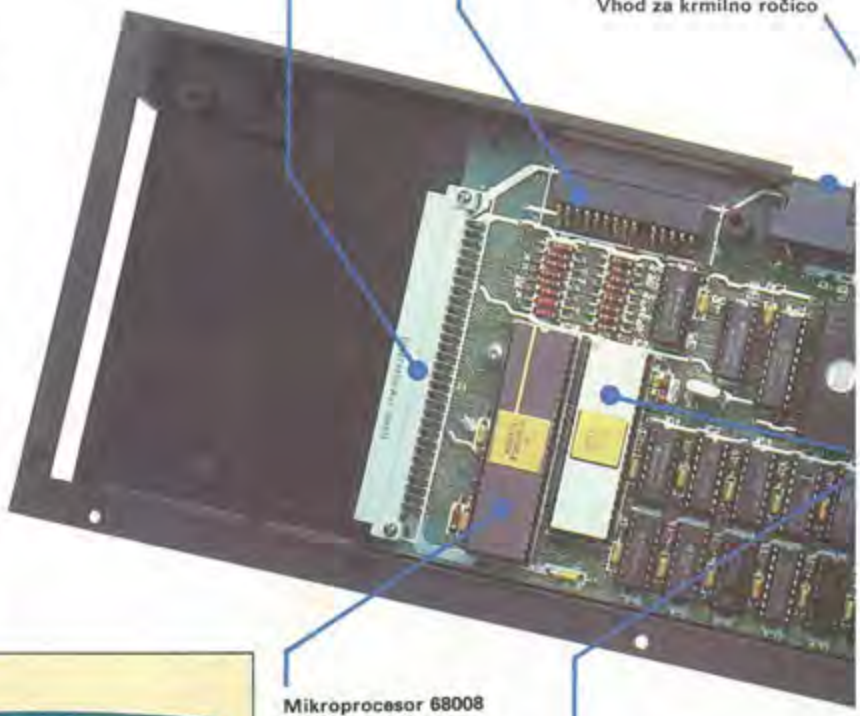
DOKUMENTACIJA
Navodila je očitno pisal dober strokovnjak, vendar z domnevo, da uporabnik že nekaj ve o računalnikih. Različna daljša poglavja so posvečena specializirani uporabi sistemskih programov grafike, zvoka in vhodno-izhodnim funkcijam računalnika. Podrobna in dokaj popolna je tudi razlaga delovanja ter programiranje mikroprocesorja 6502. Drugače kakor navodila pri drugih računalnikih to ni le kopija originalnih Rockwellovih dokumentov, temveč res koristno poglavje iz dokumentacije računalnika BBC model B.

Sinclair QL

Quantum Leap ponuja najnaprednejši mikroprocesor vdelan v katerikoli hišni računalnik in možnost razširitve pomnilnika na pol megabyta

Vse inovacije sira Cliva Sinclaira na področju hišnih računalnikov so pomenile velik skok tako na področju tehnologije, kot vrednosti, ki jo nudijo za svojo ceno. Njegov zadnji mikroračunalnik pa je prvi, ki je ta opis privzel tudi kot ime: Sinclair Quantum Leap (QL). S svojo ceno (399 funtov) osvaja vedno večji krog uporabnikov, ki so ali resni računalniški navdušenci ali pa ga kupijo zaradi posla ali resnega dela doma. Kot tak je QL zelo resen tekmec modelom, kot sta Commodore 64 ali BBC model B, kajti s svojimi tehničnimi lastnostmi ju daleč prekaša.

Jasno je, da so pri razvoju QL združili vse sestavine in lastnosti, ki predstavljajo ta trenutek vrh računalniškega napredka. Drugače kakor standardni mikroprocesorji Z80 ali 6502 ima QL vdelano centralno procesno enoto (CPU) iz Motoroline družine 68000, ki je za zdaj najmodernejši mikroprocesor, ki se uporablja v mikroračunalnikih in ga najdemo v napravah, kot je



QL programska oprema

QL Quill je programski paket za urejanje besedil, ki na zaslonu pokaže besedilo v enaki obliki, kakor bo kasneje natisnjeno.

QL Abacus je računovodski program z nenavadno lastnostjo, da lahko celice poimenujemo z imeni in ne samo s koordinatami.

QL Archive je paket za podatkovne baze. Uporabnik lahko z zaslonskim editorjem sam razmešča zapise.

QL Easel je grafični podporni program, s katerim lahko rišemo grafe in tabele; ima razne dodatke, kakor na primer avtomatsko nastavljanje merila.

računalnik Apple Lisa. CPU je 68008, kar pomeni, da ima podatkovno vodilo samo osem bitov, čeprav so njegovi notranji registri 16-bitni (in lahko izvaja precej operacij na 32 bitih). Taka organizacija rahlo upočasni delovanje CPU, ker opravlja vpis in branje registrov v dveh delih po osem bitov. To pa hkrati pomeni, da ostane cena pomnilnika zelo nizka (QL namreč uporablja standardne 8-bitne pomnilnike) — gospodarnost pa je bila pri Sinclairju vedno eden glavnih dejavnikov pri izbiri komponent.

QL je v temeljni izvedbi opremljen s 128 kb delovnim pomnilnikom, ki pa ga bo mogoče razširiti na 512 kb (ali po domače »pol mega«). Velik pomnilnik je še posebej prikladen za poslovne namene, ker zmanjšuje število naslavljanj zunanjega pomnilnika. Pomnilnik se sestoji iz dveh enot Microdrive, ki sta vdelani v ohišje računalnika in dajeta vsaka okoli 100 kb prostora za shranjevanje podatkov. Čeprav je zato QL samostojen poslovni sistem, sta enoti Microdrive vse-

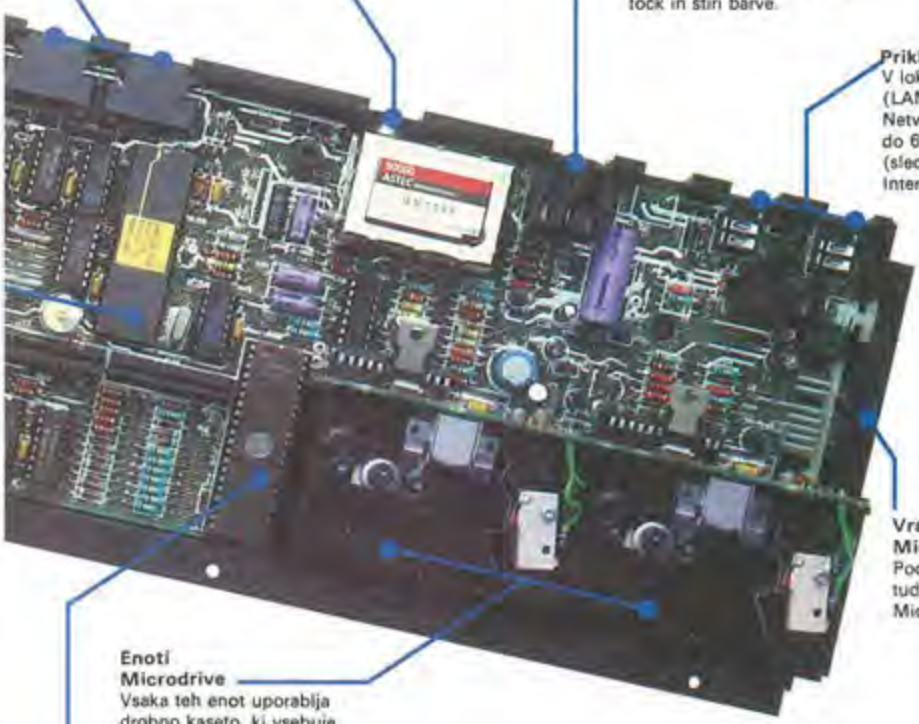
Tipkovnica
Čeprav je tipkovnica izdelana po načelu membrane (in je tako odporna tudi proti polivanju s kavo), ima 65 polno delujočih tipk in občutek pri tipkanju je popolnoma enak kakor pri najdražjih poslovnih računalnikih. Na tipkovnici so štiri kontrolne tipke za kurzor in pet funkcionalnih tipk, ki jih lahko programiramo.

Serijska vrata
Računalnik ima dvoje serijskih vrat RS 232, primernih za priključitev tiskalnikov in modemov. Bolj običajni (Centronics) vmesnik za tiskalnik je treba dokupiti.

Tv priključek
QL bo deloval na navadnem televizorju, vendar bo na zaslonu samo 40 ali 60 znakov v vrstici, na monitorju pa je mogoče zapisati 85 znakov v vrstico.

Priključek na monitor
Drugače kakor Spectrum lahko QL direktno krmili RGB monitor, ki je potreben, če želimo izkoristiti popolno ločljivost 512 × 256 točk in štiri barve.

Priključek za mrežo
V lokalno računalniško mrežo (LAN — Local Area Network) lahko povežemo do 64 QL in Spectrumov (slednjih z dodanim Interface 1).



Vrata za dodatne enote Microdrive
Podobno kot Spectrum lahko tudi QL krmili do osem enot Microdrive.

Enoti Microdrive
Vsaka teh enot uporablja drobno kaseto, ki vsebuje trak, na katerega lahko spravimo do 100 kb podatkov.

Pomožni mikroprocesor
Ta Intel 8049 krmili tipkovnico, zvok in serijska vrata ter s tem sprošča 68008 za izvajanje programov.

Sinclair QL

CENA
399 funtov

VELIKOST
472 × 138 × 46 mm

CPU
Motorola 68008

TAKT
7,5 MHz

POMNILNIK
128 kb RAM, z možnostjo razširitve na 512 kb, 32 kb ROM, z možnostjo razširitve na 64 kb

ZASLON
25 vrstic po 85 znakov (z monitorjem), grafika z visoko ločljivostjo: 512 × 256 točk (4 barve) ali 256 × 256 točk (8 barv)

VMESNIKI
Serijski RS232 (2), krmilne ročice (2), enote Microdrive, LAN, tv, RGB monitor

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
V pripravi je več jezikov, najbolj opazen je jezik »C«

OPREMA
Navodila za uporabo, štirje uporabniški programi

DOKUMENTACIJA
Priložen priročnik je zelo kvaliteten in uporaben.

eno šibka točka, če ju primerjamo z izredno zmogljivim procesorjem. Microdrive potrebuje okoli 3,5 sekunde, da najde podatek, nova generacija mini disketnih enot pa le polovico sekunde.

Sinclair zatrjuje, da nameravajo povezati računalnik z diskovno (Winchester) enoto, nimajo pa nobenih načrtov za povezavo na gibki disk — čeprav jo bo nekaj neodvisnih proizvajalcev nedvomno ponudilo. Brez diskovnih enot pa QL na žalost ne bo mogel izvajati operacijskega sistema Unix, za katerega menijo, da je bil glavni razlog pri izbiri Motoroline CPU 68000, in ki bo verjetno zamenjal CP/M kot standardni operacijski sistem za poslovne namene.

QL je običajno opremljen s štirimi poslovnimi programskimi paketi, ki so bili razviti pri programski hiši Psion. Quill je urejevalnik besedil, Abacus je računovodski paket, Archive je podatkovna baza in Easel je grafični paket. Vsi tečejo na procesorjevem operacijskem sistemu, ki ga je Sinclair imenoval QDOS. Popularnost, ki jo bo QL verjetno dosegel, pomeni, da bodo zanj razvili precej programske opreme, čeprav prenos že napisanih programov na QL ne bo lahek. Po drugi strani pa je res, da Sinclair ne bi vodil na tržišču, če bi za svoje izdelke privzemal že uveljavljene industrijske standarde.

Vdelani basic je izboljšana verzija Spectrumovega basica, in kakor da že ime Quantum Leap ne bi bilo dovolj, ga je Sinclair imenoval Super basic. Vsebuje možnosti za izvajanje postopkov (in s tem podpira strukturno programiranje) in za dostop v operacijski sistem iz programov, pisanih v basicu. Tako basic kot QDOS sta standardno vpisana v 32 kb delovni pomnilnik (ROM).

Sinclair QL je nedvomno zelo imenitna naprava in, to je mogoče še bolj pomembno, ima zadostne možnosti za širjenje, in ne bo kmalu zastarel. Lepo se vklaplja v dolgo vrsto Sinclairovih mejnikov: ZX80, ZX81 in Spectrum.

Prototipna plošča

Fotografija prikazuje prototipno tiskano vezje QL, zato bodo lahko posamezni elementi še spremenjeni.

Vprašanja in odgovori

Kaj je »peta generacija« računalnikov in kakšni so bili računalniki prejšnjih generacij?
Računalnike, ki jih programerji in računalniški strokovnjaki skupno načrtujejo za bližnjo prihodnost, imenujemo peta generacija računalnikov. Od teh računalnikov pričakujemo, da bodo veliko zmogljivejši od današnjih.
Izraz »peta generacija« je bil vzet iz japonskega raziskovalnega projekta, ki je opisoval razvoj računalništva v prihodnje. Računalniki »pete generacije« ne bodo imeli tipkovnice in jih ne bo treba programirati v računalniških jezikih, kot so basic ali pascal. Namesto tega pričakujemo, da bomo lahko računalniku povedali svoj problem z besedo, prav tako pa nam bo tudi odgovoril v kateremkoli jeziku. Računalniki »pete generacije« bodo torej sami zase pisali programe iz opisa, ki jim ga bomo prej povedali z besedami.
Prva generacija računalnikov se je pojavila po drugi svetovni vojni in je uporabljala predvsem elektronke. Pri drugi generaciji računalnikov so namesto elektronk in relejev že uporabili transistorje in tako naredili manjši in cenejši računalnik. Tretja generacija je imela že vdelana preprosta integrirana vezja na siliciju. Le-ta so začetna oblika obsežnih mikrovezij na siliciju in so že označevala začetek padanja cen računalnikov. Kljub temu pa so bili ti računalniki še vedno predragi za vsakdanjo domačo rabo.
Četrta generacija so računalniki na ravni sedanje tehnologije. Ti računalniki uporabljajo visoko stopnjo integracije

(LSI-Large Scale Integration). Razvoj teh mikrovezij pa je dal računalnike, ki so dostopni večini ljudi.

Kje je Silicijeva dolina?
Področje okrog San Joseja, južno od San Francisca, imenujemo Silicijeva dolina.
Tu imajo svoje razvojne oddelke vrhunska ameriška računalniška in mikroelektronska podjetja. Vzrok, da so se proizvajalci mikroelektronskih vezij zbrali na tako majhnem področju, je v ugodnostih, ki so jim jih dale lokalne oblasti. Želele so industrializirati območje, na katerem so še pred 20 leti pridelovali le sadje.

So ohlapne povezave na zadnji strani računalnika potrebne?
Večina današnjih računalnikov je načrtovana glede na zunanji videz, kakor tudi glede na elektroniko. Ohlapnih povezav običajno ni. Toda z razvojem sodobnih računalnikov so ohlapne žice postale pomembne. Vemo, da se elektrika giblje s hitrostjo svetlobe, pa kljub temu porabi nekaj časa, da pride po žici do želenega cilja.
Ti prototipni računalniki delajo tako hitro, da želena informacija mora prispeti na natanko določeno mesto ob zahtevanem času. Zato je dolžina žic izračunana natančno tako, da zagotavlja pravilne časovne odzive.

Večkrat slišimo, da imajo računalniki različne mikroprocesorje, npr. Z80 ali 6502. Kaj je pomen teh številk?
Številka sama po sebi nima nobenega pomena. »6502« ali »Z80« označujeta ime mikroprocesorskega vezja, pod katerim ga lahko poiščemo v katalogu proizvajalca. Vsak tip računalnika, ki je zasnovan na posameznem mikroprocesorju, razume samo zbirko temeljnih ukazov le-tega. Iz teh temeljnih ukazov je sestavljen program. Uporabnik pa navadno piše programe v jeziku na višji ravni. Tak jezik je lahko basic, ki ga računalnik prevede v strojno kodo ali v temeljne ukaze mikroprocesorja. Dokler ne želimo pisati programov neposredno v strojni kodi, je povsem vseeno, kateri tip mikroprocesorja ima naš računalnik.
Različni mikroprocesorji delajo z različno hitrostjo, toda hitrost reševanja nekega problema je veliko bolj odvisna od tega, na kakšen način je napisan program.

Kako lahko računalniki pomagajo v boju proti kriminalu?
Kmalu po uvedbi računalnikov so imeli tudi varnostni organi svoj računalnik. Najprej so izdelali podatkovno bazo, kjer so imeli shranjene podatke o ukradenih in sumljivih vozilih. Od tedaj so v računalnik shranjevani podatki o ljudeh, ki so kakorkoli prišli navzkriž z zakonom. Zadnje čase varnostni organi proučujejo možnost uporabe mikroračunalnikov pri delu intervencijskih oddelkov. Nekateri avtomobili varnostnih organov že imajo računalnik, ki je povezan s centralnim računalnikom, tako da imajo dostopne vse informacije kar v avtomobilu. Tehnologija shranjevanja podatkov je tako napredovala, da bi lahko imeli shranjene obsežne podatke o vsakem državljanu.

Atari 400 in 800

Specialiteta Atarijevih računalnikov so igre

Atarijeve tipkovnice
Že na prvi pogled se ta dva modela ločita po tipkovnicah. Medtem ko je večji model 800 opremljen s takšno tipkovnico, kot so na pisalnih strojih, ima manjši model membransko tipkovnico, ki je sicer boljša kot pri nekaterih drugih računalnikih, vendar ima sama po sebi še vedno nekatere pomanjkljivosti. Taka tipkovnica ne daje pravega občutka tipkanja, odzivi pa so včasih nepredvidljivi. Po drugi strani pa je res, da ravno take tipkovnice uporabljajo zelo dragi in odporni stroji, narejeni za uporabo v industrijske in vojaške namene. Na ta način dosežejo zaščito pred prahom in prevrnjenimi skodelicami kave!

Razmah Atarija — sedež podjetja je zdaj v Sunnyvalu v kalifornijski Silicijevi dolini — sloni na neverjetnem uspehu njegovih arkadnih iger. Prva od teh je »Pong«, ki jo igramo na črno-belem televizorju.

Atari je rastel od tega skromnega začetka in postal del velikanske firme Warner Communications Group. Danes, šest let pozneje, je med največjimi proizvajalci hišnih računalnikov. Velik delež ima tudi na trgu računalniških iger.

Z zanj značilno konstrukcijo, ki daje občutek trdnosti, so Atarijevi modeli 800 in 400 postali mnogim proizvajalcem nedosegljivi vzor. Izvrstna grafika, zelo izpopolnjena programska oprema in do nedavnega zelo visoka cena še povečujejo občutek, da gre za kvaliteten izdelek. Model 400 prodajajo za polovico ceneje kakor model 800, vendar ima le 16 kb pomnilnika, medtem ko se da večji model razširiti na 48 kb. Poleg tega ima cenejša verzija namesto dveh vtičev za vtične module samo enega. Model 400 se lahko uporablja samo s televizorjem, medtem ko je pri modelu 800 mogoče uporabljati tudi monitor. Modela se najbolj razlikujeta po tipkovnici. Da se je ognil različnim izhodnim nivojem navadnih kasetofonov, uporablja Atari rajši svoj lastni model. Velik del uporabnikov njegovih računalnikov ima vsaj po eno diskovno enoto, da lahko uporabljajo velik izbor programov, dostopnih na disketah.

Diskovna enota
Čeprav je bila Atari 810 ena najbolj koristnih perifernih naprav, je že nekoliko zastarela. Zmore 88 kb na disketo, kar je precej malo, in ker je povezana z računalnikom s serijskim vmesnikom, je tudi počasna. Ima pa dognan operacijski sistem z mnogimi lastnostmi drugih podobnih naprav.

Kasetofon
Kasetofon je posebej izdelan za Atari. Model Atari 410 je zanesljivejši in preprostejši od običajnih kasetofonov. Ker nima zvočnika, je manjši in lažji. Poleg tega uporablja posebne kasete. Glavna pomanjkljivost sistema je to, da ni mogoče delati poimenskih datotek na traku.

Atarijeva krmilna ročica
Krmilna ročica je eden slabših dodatkov. Je stikalnega tipa, brez možnosti zveznega spreminjanja signala. Daje samo smer in še precej okorna je. Pri drugih proizvajalcih se dobijo boljše.

Plošča centralne procesne enote

ANTIC
To je eno od specialnih integriranih vezij, ki dajejo Atariju njegovo izjemno učinkovitost. ANTIC nadzira izpis na zaslon, svetlobno pero in eno od prekinitev.

CTIA ali GTIA
To integrirano vezje, lastno Atariju, skrbi za barve nekaj različnih vhodno-izhodnih vrat in grafiko Player-Missile.

Ura za barve
Z obračanjem te kontrole se menjajo barve. Če spreminjamo hitrost ure, dobimo grafiko različne ločljivosti.

6502 centralna procesna enota

Nastavljanje barve

Tiskana vezja delovnega pomnilnika

V Atariju je dovolj prostora za tri tiskana vezja z delovnim pomnilnikom, tako da lahko razširimo pomnilnik na 48 kb.

Glavna ura

Pokey
POKEY je tretje od integrir vezij po naročilu. Skrbi za tipkovnico, serijska vhodno-izhodna vrata, sistemsk ure in nadzira zvok.

6520 PIA
To integrirano vezje skrbi krmilne ročice.

Zvočnik

Atari se z zamenjavo teh bralnih pomnilnikov spremeni v čisto drug stroj, na katerem lahko, na primer, uporabljamo drug programski jezik.

Preglednica

Temeljna plošča

Usmerniške diode
Atarijev napajalnik daje izmenično napetost, ki jo diode usmerijo v enosmerno.

Zunanja vtičnica

Stikalo za vklop/izklop

Razširitvena mesta 1, 2 in 3
V vsakega od teh konektorjev lahko priklopimo modul delovnega pomnilnika z dodatnimi 16 kb spomina.

Izbirnik kanala

Priključek centralne procesne enote
Centralna procesna enota je pri Atariju na posebni tiskani plošči, ki se pritrdi na to mesto.

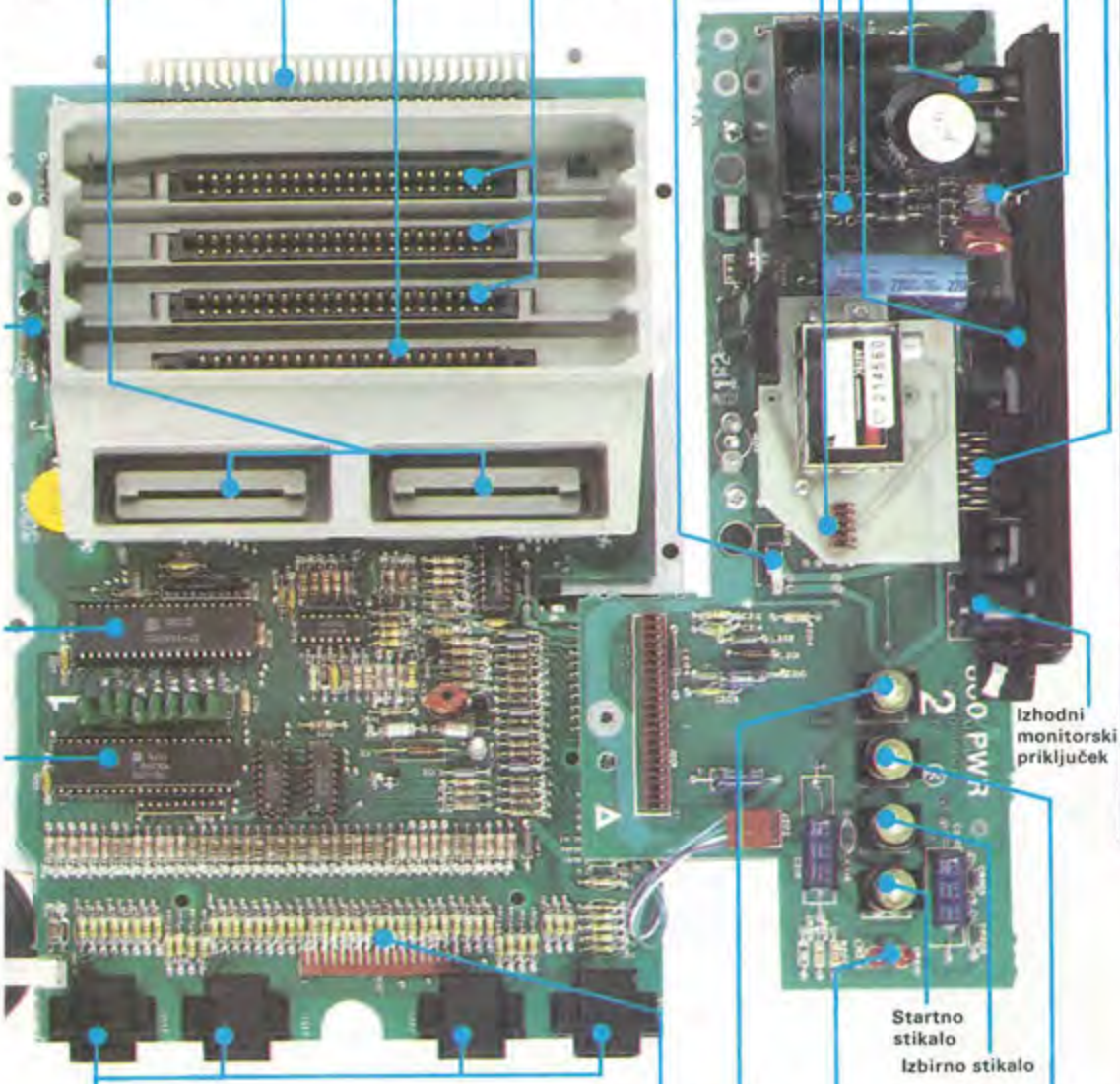
Konektor za video signal

Napajalna vtičnica

Levi in desni priključek za vtični modul
Z vsakim lahko spojimo vtični modul z igrami ali koristnimi programi.

Priključek za bralni pomnilnik

Varnostno stikalo
To stikalo prekine napajanje, kadar se vzdigne pokrov.



Izhodni monitorski priključek

Startno stikalo

Izbirno stikalo

Stikalo Option

Stikalo za reset sistema

Priključek za tipkovnico

Svetleči diodi za kontrolo napajanja

Priključki za krmilne ročice

Atari 800

VELIKOST
405 × 330 × 110 mm

TEŽA
4,2 kg

CPU
6502

TAKT
1,79 MHz

POMNILNIK
Od 16 do 48 kb

ZASLON
40 × 24 znakov

VMESNIKI
Priključki za tv, monitor, kasetofon (posebna izvedba), 4 krmilne ročice, serijska vrata

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Basic A+, pilot, C

OPREMA
Napajalnik (brez vtiča), priročnik

TIPKOVNICA
57 posameznih gibljivih tipk, 3 funkcijske tipke

DOKUMENTACIJA
Priročniki z navodili so jasno napisani, natančni in dobro natisnjeni. Uporabniki z več izkušnjami lahko dobijo še tehnične opise, ki so natančno isti, kakor jih uporabljajo Atarijevi inženirji. Poleg popolnih shem vezja dobite v njih izpiske mnogih internih programov, ki nadzirajo notranje delovanje računalnika (sistemska programska oprema).

Acorn Electron

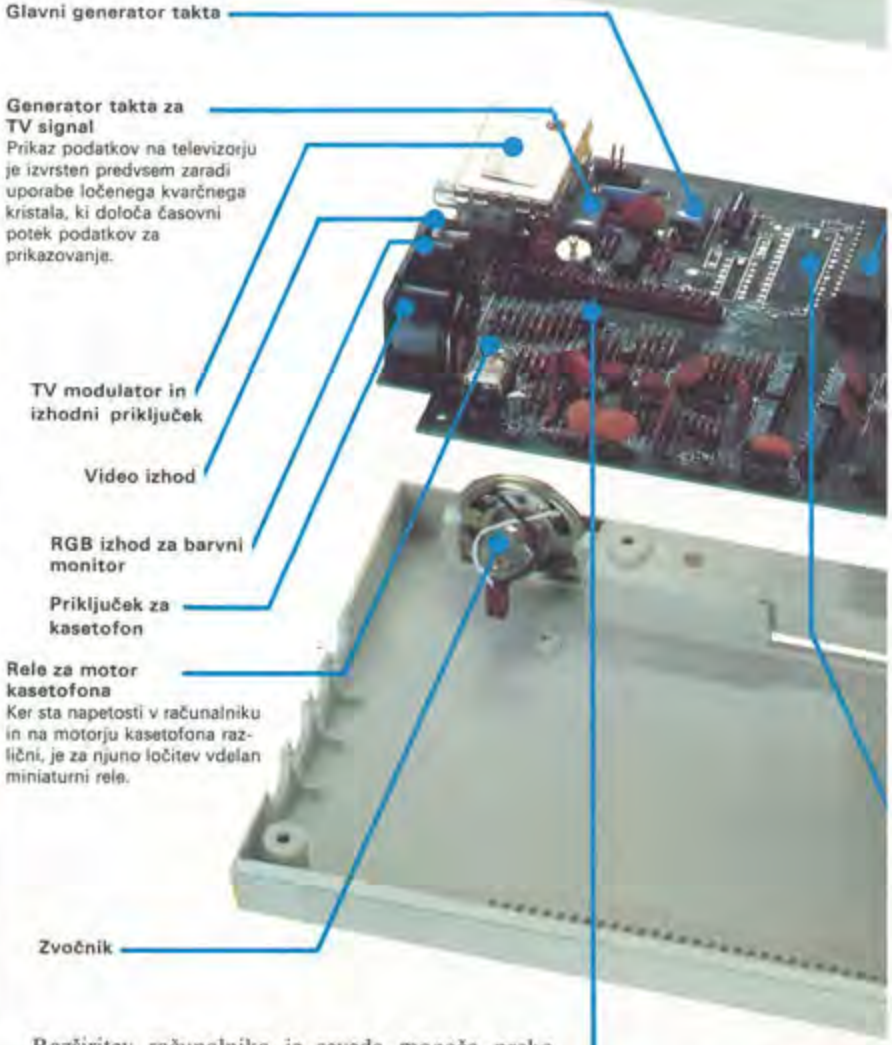
V dveh letih, ki sta minili med Acornovima računalnikoma BBC model B in Electron, se je računalniška tehnologija dramatično razvila

Acorn Electron je eleganten računalnik, ki potrjuje to, kar nam o njem govori že prvi vtis: da je čvrst in dobro zasnovan. To je pomanjšana verzija računalnika BBC Micro. Pri načrtovanju računalnika so dali več poudarka preprosti uporabi kakor pa vrhunskim lastnostim. Večina lastnosti BBC Micro je ohranjena tudi v Electronu. Za zgled sta nam lahko ukaza SOUND in ENVELOPE, oba namenjena generiranju zvokov, ki posnemajo različna glasbila.

Vsi grafični načini, ki so dosegljivi na BBC Micro, so razen enega dosegljivi tudi na Electronu. Ta izjema je videoteks (MODE 7), ki je izveden pri BBC Micro s posebnim integriranim vezjem. Na tiskani plošči Electrona to integrirano vezje ni predvideno, zato lahko podoben prikaz, kakor je videoteks, izvedemo z uporabo MODE 6, če prej preimenujemo večino znakov. Vendar je ta »ponaredek« videoteksa nekoliko slabši, ker sta z uporabo MODE 6 dobljeni samo dve barvi. To je pravzaprav pomanjkljivost Electrona, saj s teletekstom lahko prikažemo na zaslonu veliko podatkov ob sorazmerno majhni zasedenosti delovnega pomnilnika.

Tudi vhodno-izhodne sposobnosti so skromnejše kakor pri BBC Micro. Za prikaz podatkov ima Acorn Electron možnost priključitve na 36. kanal televizorja ali pa direkten video izhod na barvni monitor. Ima tudi priključek za magnetofon.

Ustvarjalna dvojica
Načrtovalca Electrona sta Cris Curry (levi) in Herman Hauser (desni). Oba sta sodelovala pri načrtovanju BBC Micro. Curry je bil razvojni inženir pri Clivu Sinclairju, ko je le ta zaposlil Hauserja. Potem sta skupaj ustvarila Acorn.



Razširitev računalnika je seveda mogoča preko konektorja, ki je na zadnji strani računalnika. Kadar tega priključka, ki štrli na spodnjem zadnjem robu ohišja, ne uporabljamo, je zaščiten s plastičnim pokrovom. V priročniku, ki ga dobimo ob nakupu računalnika, ni podrobnejših navodil, kakšni signali pridejo ven, ali kaj lahko na ta konektor priključimo. Vendar je očitno mišljeno, da bi naj bila to povezava računalnika z neko zunanjo enoto, saj so zraven montirani tudi medeninasti vtikači, ki bi rabili za mehansko povezavo konektorja.

Electron vsebuje BBC verzijo basica, ki pa jo lahko

Konektor za tipkovnico
Priključimo jo s konektorjem, ki ima 22 priključkov; iz tega lahko sklepamo, da izhod tipkovnice ni spremenjen v kodo ASCII, saj takrat potrebujemo le 10 priključkov (8 za podatke in 2 za napajalno napetost). 22 priključkov je mogočih zaradi vezja ULA (ULA — Uncommitted Logic Array ali logične mreže).

Tipkovnica
Electron ima tipkovnico podobno kakor pisalni stroj. Je zelo kvalitetna in spada med najboljše tipkovnice pri hišnih računalnikih. Razporeditev tipk je podobna kakor pri BBC Micro. Nima posebnih tipk za dodatne funkcije, pač pa to dosega s tipko, posebno za velike črke. Kadar je ta tipka pritisnjena skupaj s kako tipko za številke, se spremeni v funkcijsko. To velja tudi za tipke s črkami in tri tipke z ločili. Če jih pritisnemo, medtem ko držimo tipko za velike črke, dobimo ukaze v basicu, ki so napisani na sprednji strani tipk.

Konektor za razširitev
S tem priključkom je mogoče priključiti na računalnik nekatere dodatne enote. Žal v navodilih ni podan podrobnejši opis signalov, ki naj se pripeljejo na konektor.

CPU
Uporabljen je standardni mikroprocesor 6502A, ki dela na frekvenci 1,79 MHz. CPU odloča in se po tem loči od logičnih mrež.

ULA
ULA ali logična mreža. Poleg logične mreže, CPU, ROM in RAM je na plošči le še devet standardnih TTL integriranih vezij, od katerih je v vsakem le nekaj logičnih vrat.

RAM
Mikroprocesor pošilja zloge v delovni pomnilnik v dveh delih. Prvi štirje biti (vsak prihaja iz enega od štirih integriranih vezij) sledijo drugi četverici bitov. To je nekoliko neobičajna rešitev, kajti večina računalnikov hrani vseh 8 bitov vsakega zloga v istem integriranem vezju.

Napajalnik
Tudi glede napajalne napetosti je Acorn Electron posebnost, saj zahteva izmenično napetost 19 V. Tako napajanje je zelo stabilno, zahteva pa kompleksnejše vezje za pretvorbo izmenične napetosti v napetost, primerno za računalnik.

Prostor za vezje z 28 priključki
Označeni prostor je namenjen za vdelavo dodatnega ROM ali kakšnega drugega integriranega vezja.

razširimo, tako da nam nove lastnosti omogočajo prijetno in uporabno programiranje.

Zelo uporaben je program OSCLI, ki omogoča poseg v operativni sistem direktno z ukazi v basicu. Prav tako lahko razširimo tudi obseg ukazov v strojni kodi, ki pa so specifični za BBC basic. Na nekaterih tipkah ima celotne ukaze, to je še zlasti ugodno pri shranjevanju in izpisovanju podatkov.

Glede na opisane lastnosti lahko sklepamo, da je Acorn Electron boljši od povprečnih hišnih računalnikov. Prikaz podatkov na zaslonu je izredno oster, stabilen in v čistih barvah. Če mu dodamo še nekatere enote, kot npr. disk, pisalnik, dodaten RAM itd., dobimo prav zmogljiv in uporaben računalnik.

Acorn Electron

CENA
200 funtov v Veliki Britaniji

VELIKOST
340 × 160 × 65 mm

CPU
6502

TAKT
1,79 MHz

POMNILNIK
64 kb ROM
32 kb RAM — brez dodatne možnosti razširitve na osnovni tiskani plošči

ZASLON
Do 32 vrstic s po 80 znaki, 8 različnih barv z možnostjo izbire ozadja;
127 znakov; možnost definiranja 255 znakov

VMESNIKI
Izhod za televizor za kanal 36, video izhod, izhod za barvni monitor, priključek za kasetofon in sistemsko vodilo (nedokumentirano).

PROGRAMSKI JEZIK
BBC basic in pripadajoči strojni jezik

DRUGI PROGRAMSKI JEZIKI
Electron lahko uporablja še druge Acornove jezike, kot sta forth in lisp. Te jezike vpišemo v delovni pomnilnik. Jeziki pascal, bcpl in drugi, ki so shranjeni v ROM, so pri nerazširjeni verziji računalnika neuporabni (premalo prostora v pomnilniku).

OPREMA
Poleg računalnika je v ceno vračunan še priročnik za basic, tv priključni kabel, napajalnik in kaseta z navodili.

TIPKOVNICA
Sestavljena je iz 56 tipk. Ukaze v basicu tipkamo z eno tipko, 10 tipk pa je namenjenih za to, da jih uporabnik določi sam.

DOKUMENTACIJA
Je izvrstna. Vsebuje obilo podrobnosti, ki jih potrebuje začetnik ali tudi izkušen programer. Vsaka tipka za basic je posebej razložena. Prav tako je dobro opredeljen tudi strojni jezik. Dobro so opisane tudi funkcije operacijskega sistema. Manjkajo pa navodila za razširitev računalnika.

Hišni računalniki Made in Yugoslavia

Hišni računalniki jugoslovanskih proizvajalcev so se začeli pojavljati na domačih sejmih konec sedemdesetih let. Večina prototipov pa je nastala v letih 1983 in 1984

Jugoslovanske hišne računalnike karakterizira nekaj dejstev:
- nekajkrat višja cena kot za modele podobnih zmogljivosti v tujini
- dokaj dolga doba od zasnove do redne proizvodnje — pri enem od modelov več kot pet let
- vsak proizvajalec razvija svoj operacijski sistem in svojo verzijo programskega jezika basic — samo v Sloveniji so razviti trije operacijski sistemi in trije basici — ljubljanski, mariborski in piranski
- modeli praviloma nimajo posebno širokega izbora strojnih dodatkov
- vsi modeli imajo 8-bitne mikroprocesorje
- za posamezne modele je na razpolago skromen nabor programske opreme
- računalniki imajo po pravilu dokaj skromna navodila

Trenutno še ni moč govoriti o proizvodnji hišnih računalnikov v Jugoslaviji, zlasti ne, če uporabljamo za to proizvodnjo merila, ki veljajo v tujini. Skoraj hermetična carinska zapora pri uvozu mikroračunalnikov in devizne težave domačih proizvajalcev pri nakupu mikroelektronskih sestavnih delov, pa tudi počasno prilagajanje domače industrije tehnološkemu razvoju razvitih, vse to je ustvarilo specifične razmere na domačem tržišču. Poleg tega je treba upoštevati še bistveno večje povpraševanje od ponudbe. Samo začetne potrebe slovenskih osnovnih in srednjih šol je moč oceniti vsaj na 3200 računalniških delovnih mest, celotna proizvodnja hišnih računalnikov v SFRJ v letu 1984 pa ne bo presegala nekaj sto hišnih računalnikov in morebiti nekaj večje število osebnih računalnikov. Že v letu 1985 pa napovedujejo skoraj vsi domači proizvajalci večje prodore na tržišče.

Iz pregleda hišnih računalnikov jugoslovanskih proizvajalcev je razvidno, da je struktura domače ponudbe silno raznolika:
- cene so v obsegu od 35200 din do 200000 din, pri čemer cen za dva modela še ni;
- med mikroprocesorji prevladujejo:
  Zilogov mikroprocesor Z 80 A, ki je vgrajen tudi v računalnike: Genie I, II in Colour, Sharp MZ 700, Sinclair ZX-81 in Spectrum, Tandy TRS-80 — Model 4, Epson QX-10 in drugi
  Motorolin mikroprocesor 6809, ki ga je moč najti v računalnikih Tandy TRS-80 CC, Dragon 32 in Fujitsu FM-8. Gre za mikroprocesor, ki ni preveč razširjen
  mikroprocesor 6502, ki ga je najti v številnih računalnikih: Commodore Vic 20, Atari 600XL in 800, Apple IIe, Oric Atmos, BBC Micro, Model B in številnih drugih.
- kapacitete RAM segajo od 2kb do 64kb, pri čemer ima pet modelov v osnovni verziji manj kot 10kb RAM
- kapacitete ROM so skoraj enake pri vseh modelih
- grafična resolucija je praviloma skromna — dva modela imata več kot 32000 točk, ostali imajo manj kot 6000 točk, ki jih je moč prižigati, ugašati in testirati na zaslonu
- dva modela omogočata tipkanje 80 znakov na vrstico zaslona
- jugoslovanske tastature so v manjšini
- osem modelov hišnih računalnikov ima šest različnih vmesnikov

Oglejmo si pobliže hišne računalnike jugoslovanskih proizvajalcev in ponudnikov tovrstne opreme:

Hobi SR 84 — računalnik, ki ga je skonstruiral 17-letni dijak

Hobi SR 84 je najnovejši jugoslovanski hišni računalnik, ki bi po predvidevanjih moral priti na tržišče jeseni 1984. Njegov avtor, 17-letni dijak matematične gimnazije v Beogradu, Ivan Zindević, je računalnik zgradil okrog Zilogovega mikroprocesorja Z80A. Hobi SR 84 ima v osnovni verziji 4 kb RAM in 14 kb ROM, v razširjeni verziji z dodatnimi zunanjimi moduli pa 48 kb RAM in 17 kb ROM, ki ima poleg ostalih dodatkov še razširjen nabor znakov z malimi.

Z razširitvijo 16 kb RAM se pridobi pascal in močan assembler editor. Problem pa je z našimi črkami (Č, Š, Ž), ki jih text editor še ne podpira. Programsko je Hobi SR 84 kompatibilen z enim od znanih mikroračunalnikov, kar lahko pomeni, da bo na voljo močna programska podpora.

Ta računalnik ni nekonvencionalen samo po načinu nastanka, ampak tudi po tem, da bo izdelava tekla v obrtni delavnici Djemrovski Tomaša v Sremski Kamenici. Planirajo tudi izdelavo cenenega 40-kolonskega tiskalnika, ki bi s ceno pribl. 50000 din predstavljal enak cenovni prodor kot leta 1983 Galaksija v kit izvedbi za približno 15000 din.

Od osmih jugoslovanskih hišnih računalnikov sta le dva nastala v razvojnih laboratorijih delovnih organizacij. Ostali so nastali kot plod dela zanesenjakov, ki so sistem ustvarili v prostem času z običajno skromnimi sredstvi, veliko zagnanostjo in precejšnjim znanjem. Podobno so nastajali tudi nekateri znani sistemi v

tujini. Razlika je le v tem, da je tam pot od razvoja do proizvodnje mnogo krajša.

GALAKSIJA — najbolj razširjen jugoslovanski hišni računalnik

Pri nastajanju Galaksije so sodelovali tudi bralci časopisa »Galaksija«, in je odigral veliko vlogo pri širjenju računalniške pismenosti v SFRJ. Avtor Voja Antonić je zasnoval simpatičen hišni računalnik, ki je po nekaterih karakteristikah boljši od Sinclairovega modela ZX 81. Ima mehansko tipkovnico (qwerty), ki je domač izdelek. Domače so tudi ostale komponente, razen integriranih vezij, ki jih je potrebno uvažati. V začetku je bilo mnogo posameznikov, ki so Galaksijo kupili v kit izvedbi in jo sestavili sami. Galaksija je torej tudi prvi domači računalnik v kitu. Vse elemente je moč kupiti pri »Galaksiji«, integrirana vezja pa se naročajo pri nekaterih firmah v tujini, ki jih pošljejo v več pošiljkah, da vrednost posamezne ne prekorači predpisane vrednosti. Do sedaj je na naslov »Galaksije« prišlo že 10000 naročil za kit komponente, realizirali pa so jih že več kot 4000. V zadnjem času je prišlo do nekaj zastojev, ki jih občasno povečujejo še težave s carino.

Delovna organizacija Elektronika inženjering iz Zemuna izdeluje Galaksijo v sestavljeni verziji. Na voljo sta dve izvedbi: 4—4 in 8—6, pri čemer prva številka pomeni obseg ROM, druga pa RAM v kb. Razširjena verzija, 8—6, vsebuje za ceno 52500 din v okviru spominskih razširitev matematične funkcije, dodatne ukaze za basic za printer in za assembler.

Galaksija je postala najbolj razširjen jugoslovanski mikroračunalnik, ki ga odlikuje za domače razmere nizka cena in še kar dobra ponudba programske opreme, pa tudi že dodatne opreme. Na beograjskem valu 202 imajo vsako soboto poseben programski blok, namenjen Galaksiji. Pomen Galaksije je v tem, da je dokazala, da so lahko domači mikroračunalniki vendarle cenejši, pa tudi podprti s solidno programsko podporo domačega izvora in ne nazadnje, opremljeni s kvalitetnimi priročniki. Na voljo je Navodilo za uporabo računalnika Galaksija izpod peresa znanega avtorja Dejana Ristanovića na 35 straneh in učbenik prof. Nedeljka Parezanovića: Basic za Galaksijo na 200 straneh.

Galaksija — edini resnično dosegljivi hišni računalnik pri nas. Njegovi konstruktorji ponujajo vedno nove izboljšave. Ker ga prodajajo v kit izvedbi, njegova zunanja oblika ni določena.

PREGLED HIŠNIH RAČUNALNIKOV JUGOSLOVANSKIH PROIZVAJALCEV

model	GALAKSIJA	HOBI SR 84	ORAO	LOLA-8	GALEB	HR-84	DIALOG 20	COLOR GRAF
cena (din)	53200	50—60000.-	80000.-	95000.-	90—100000.-	100—130000.-	200000.-	
CPU	Z80A	Z80A	6502	8085 A	6502	6809	Z80A	6809
RAM kb od—do	4—54	4—48	8—32	6—38	9—64	16—48	64	64
ROM kb od—do	4	14—17	16	16	16	16	16	16—24
jeziki	basic assembler	basic pascal	basic pascal miniassembler	basic miniassembler	basic miniassembler	basic	basic cobol pascal	razš. basic pascal, macro-assembler
zaslon (vrstic + znakov)	16vr. × 32zn.	16vr. × 32zn.	16vr. × 32zn.	25vr. × 40zn.	16vr. × 48zn.	24vr. × 40zn.	24vr. × 80zn.	16vr. × 32zn.
grafična resolucija (točk)	48 × 64	48 × 128	128 × 256	75 × 80	48 × 96	semigrafika	semigrafika	192 × 256
barve da/ne	ne	ne	ne	ne	ne	ne	ne	da
tastatura (štev. tipk)	54	53	61	48	59	81	82	68
vmesnik	44-polni EDGE	Centronics	RS232C	64-polni EURO	IEEE488	/	RS232C	RS232C
zvok	/	4 oktave	5 oktav	8 oktav	5 oktav			6 oktav
proizvajalec	Elektronika inženjering Zemun	Djemrovski Tomaš, Sremska Kamenica	Velebit Zagreb	Ivo Lola Ribar	Velebit Zagreb	Iskra, Široka potrošnja, Škofja Loka	Gorenje, Titovo Velenje	Slovenijales inženiring

Galeb — domača varianta Applea. Dnevi se mu iztekajo...

Lola-8 — hišni računalnik z dobrim zvokom in neprofesionalno tipkovnico

Lola-8 je eden prvih jugoslovanskih hišnih računalnikov, pa tudi edini, ki nima profesionalne tipkovnice. Vgrajen Intelov, relativno star mikroprocesor 8085 A, ki deluje na frekvenci 5 MHz, vpliva na hitrejše odvijanje programov. V ROM ima kar zmogljiv basic, ki omogoča računanje trigonometričnih in logaritemskih funkcij z devetimi točnimi številkami in delo z večdimenzionalnimi polji.

Ta računalnik ima dobre zvočne karakteristike. Generator zvoka, ki ima 14 notranjih registrov, omogoča ločeno programiranje treh zvočnih kanalov za generiranje tonov v celotnem slišnem obsegu, doziranje šumov različnih amplitud in oblik.

Slaba stran Lole-8 je gotovo neprofesionalna tipkovnica. 48 tipk (najmanj od vseh jugoslovanskih hišnih računalnikov) ima poleg običajnih velikih črk še nekaj drugih grafičnih simbolov in naše črke Č, Š in Ž.

Poleg osnovne izvedbe je na voljo tudi razširjena izvedba za 233000 din. Ta izvedba ima 16 kb ROM, 38 kb RAM in paralelni vmesnik za standardne periferne enote, kot je upi tiskalnik. Razširjena izvedba nastane s priključitvijo posebnega dodatnega modula preko standardnega 64-polnega EURO konektorja na zadnji strani računalnika. Proizvajalec ponuja tudi matador kartico, ki je v bistvu prazna EURO kartica s konektorjem, na kateri lahko uporabnik sam realizira razširitve sistema.

Proizvajalec si prizadeva te računalnike uveljaviti zlasti v šolah in v proizvodnji. Podobno kot Elektronika inženjering Galaksijo je tudi SOZD Ivo Lola Ribar iz Železnikov, ki je proizvajalec Lole-8, prijavila ta računalnik kot učilo pri samoupravni interesni skupnosti SR Srbije za vzgojo in izobraževanje, z jasno ambicijo, da bi si tudi na ta način zagotovila plasma svojega računalnika. Poleg računalnika je na voljo tudi priročnik v srbohrvaškem jeziku na približno 80 straneh formata A4.

Galeb — v kratkem konec njegovega leta

Avtor tega hišnega računalnika Miroslav Kocjan je zgradil računalnik okrog mikroprocesorja 6502, ki je sorazmerno razširjen. Računalnik, tip 101, imenovan Galeb, proizvaja delovna organizacija PEL iz Varaždina v sodelovanju z Velebitom iz Zagreba, ki je odgovoren za celoten marketing.

Kot predlogo za oblikovanje programskega jezika so uporabili Microsoftov basic, ki ga je moč najti v mnogih sistemih, med drugim tudi v prenosnem računalniku formata A4 — tako imenovani hand held računalniki — Epsonu HX-20. Za uporabo v Galebu je avtor Microsoftov basic ustrezno modificiral.

Celotna proizvodnja tega računalnika je zaključena pri približno 250 proizvodih. Konec leta 1984 preneha proizvodnja, kar pomeni, da gre za prvi jugoslovanski hišni računalnik, ki se umika iz proizvodnje, ker prihaja za njim nov, izpopolnjen in cenejši (!) model. Ta pojav, znan iz razvoja tudi najbolj znanih hišnih računalnikov, vse prepočasi prerašča v tendenco tudi na domačem tržišču.

Galeb se uporablja pretežno v šolstvu, pa tudi v industriji.

Orel — novinec z dobrimi grafičnimi zmogljivostmi

Namesto tipa 101, se pojavlja tip 102, imenovan Orel. Konstruktor Miroslav Kocjan je z uporabo manjšega števila integriranih vezij dosegel nekatere poenostavitve. Uporabljena vezja imajo večjo kapaciteto. Najprej je sicer želel uporabiti mikroprocesor 6809, a se je zaradi cene raje odločil podobno kot pri Galebu za 6502. Vsi sestavni deli, razen integriranih vezij, so narejeni pri nas, integrirana vezja pa kupujejo z devizami, ki jih dobijo s prodajo pletenih košar. Delež uvoženih komponent v celotni ceni je manjši od 50%. Orel ima dobre grafične zmogljivosti. Programsko je dosegljivih 32768 točk (128 × 256), kar je približno enako kot pri Apple IIe. Razlika je le v tem, da Orel ne generira barv na zaslonu.

V primerjavi s predhodnikom ima Orel skoraj vse karakteristike boljše ob nižji ceni. Prvotno so celo predvidevali ceno približno 55000 din, vendar je prišlo do skoka cen nekaterih elektronskih komponent v tujini in pri nas, tako da so morali cene korigirati. Za Orla je zbranih že množica naročil (več kot 3000). Prodaja teh računalnikov bi po predvidevanjih morala steči do konca leta 1984.

Proizvajalec se ukvarja z mislijo, da bi dali Orla na trg tudi v kitu, kar bi bilo zlasti privlačno za posameznike in vzgojnoizobraževalne organizacije. Sestavljanje računalnikov v samogradnji je najuspešnejša pot za demistificiranje tega področja. Sestavljalec pri tem pridobi nova znanja, predvsem pa spoznanje, da je računalnik v svojem bistvu skrbno uglašen nabor elementov, ki lahko v drugačni konfiguraciji in uglasitvi opravljajo drugačno funkcijo.

Ena od pomanjkljivosti tega in večine ostalih domačih sistemov je pomanjkanje ustrezne programske opreme. V pomanjkanju softwarskih hiš, ki v svetu skrbijo za programsko podpiranje in kompletiranje posameznih računalnikov, bi se morali domači proizvajalci računalniške opreme močneje povezovati z

ustvarjalnimi potenciali, ki že obstajajo ali pa so v razvoju: računalniški klubi, specializirane računalniško orientirane revije, rubrike v časnikih in oddaje na radiu in TV.

HR-84 — zmogljivosti pomnilnika kot pri Spectrumu, grafika kot pri ZX 81

HR-84 ima svoj izvor v računalniku, imenovanem Abacus, ki ga je razvil Mikroračunalniški klub v Ljubljani že pred nekaj leti. Iz Abakusa je Iskra — Široka potrošnja razvila hišni računalnik HR-83. Model je dobil naslednje leto oznako HR-84. Medtem so na Visoki šoli — v TOZD Elektrotehnika — v Mariboru zanj razvili programski del. Kot kažejo prvi testi prototipov, je HR-84 zelo hiter pri izvajanju programov, hitrejši od Spectruma in Commodorja 64. Vzrok za takšno hitrost leži tudi v dejstvu, da računa HR-84 samo na šest mest natančno.

Računalnik je grajen modularno in za razliko od Galaksije, Lole-8, Galeba, Orla in Hobi SR 84 s tipkovnico, ločeno od centralnega dela, v katerem so tri plošče na osnovnem vodilu. Za razliko od robustnih prototipov, ki jih je bilo videti v začetku, so v Iskri — Široki potrošnji za končno verzijo namenili sodobno in profesionalno oblikovano ohišje. Do takrat, ko bodo začeli prve računalnike vgrajevati v lepo ohišje, bodo po napovedih izboljšali in kompletirali sam sistem in njegovo programsko opremo, omogočili njegove stike z zunanjo opremo (poleg kasetofona vsaj še s tiskalnikom) in poskrbeli za ustrezne priročnike.

HR-84 ima enako kapaciteto pomnilnika kot Sinclairjev Spectrum, grafična resolucija pa je podobna tisti pri ZX 81. V kratkem se bo pokazalo, kakšne bodo programske, predvsem pa strojne posledice na HR-84 zaradi predvidenega sodelovanja s Sinclairjem.

Hišni računalnik HR-84 je namenjen hišni, osebni in šolski uporabi — proizvajalec pa obljublja, da bo računalnik mogoče dopolnjevati. Upoštevaje finančne zmogljivosti možnih kupcev bo moral vsak proizvajalec, ki bo prodajal cenovno zahtevnejše računalnike, omogočiti širitve sistema.

Po zagotovilih proizvajalca bo končna cena v rangu cene barvnih TV aparatov.

Dialog 20 — zmogljivosti pomnilnika boljše kot pri Commodorju 64, grafika kot pri ZX-81

Gorenje najavlja računalnik Dialog 20, ki ima v osrčju mikroprocesor Z80A in frekvenco urnih impulzov 4 MHz.

Podobno kot HR-84 in Color Graf ima tudi Dialog 20 tipkovnico ločeno od centralnega dela. Tipkovnica je najobsežnejša od vseh jugoslovanskih hišnih računalnikov in ima ločen numerični del, 4 funkcijske tipke in velike črke itd.

ROM obsega 16 kb, RAM pa 64 kb, uporabniku pa je od tega na voljo 48 kb za delo z basicom, ob uporabi samo monitorskega programa pa 56 kb. Pri matematičnih operacijah računa Dialog 20 na sedem mest natančno pri enojni in na 17 mest natančno pri dvojni točnosti. Poleg basica, ki je že na voljo, obljublja proizvajalec tudi programska jezika cobol in pascal.

Dialog 20 je razvit v sodelovanju s Fakulteto za Elektrotehniko v Ljubljani, kjer so zanj razvili tudi basic. Dialog 20 je namenjen osebni, hišni in šolski uporabi. Skupaj s Color Grafom spada med naše zmogljivejše hišne računalnike. Ima pa podobno kot HR-84 le semigrafiko, kar bo oviralo njegovo učinkovito uporabo na nekaterih področjih. Razvojno gleda-

Iskrin računalnik HR 84. Sam računalnik je pod televizorjem

Gorenje pripravlja Dialog 20, zmogljiv hišni računalnik s CP/M operacijskim sistemom. Sam računalnik je v istem ohišju kot disketni enoti (levo od televizorja).

no, Dialog 20, po navedbah proizvajalca, v obliki kot je predstavljen, še ni dokončno izoblikovan. Zato tečejo prizadevanja za zagotovitev višje stopnje računalniške grafike (barve) in višje stopnje integriranosti elektronskih podsklopov.

Gorenje je podobno kot Iskra — Široka potrošnja tudi proizvajalec televizorjev, kar bi se moralo odražati tudi v večji fleksibilnosti pri ponudbi cenenih monitorjev. Že v tej fazi pa je možna priključitev preko vmesnika RS232C na terminal PAKA2000. Ker je Gorenje dobavitelj opreme za teletekst na RTV Ljubljana, bi bilo smiselno, če bi zagotovili medsebojno kompatibilnost njihove tačunalniške in teletekst opreme. Commondore je npr. že razvil Btx modul, ki ga je moč enostavno vtakniti v priključek za programske module na C64. S tem modulom, ki stane 250 DM, postane C64 sorazmerno cenena tastatura za Btx (okrajšava za zahodnonemški sistem Bildschirmtext) in kar je najvažnejše, s tem je moč funkcionalno povezati komunikacije preko Btx z obdelavo podatkov na C64. Poljuben tekst se npr. najprej lokalno editira na C64, eventualno shrani s pomočjo Datasette ali Floppy enote in kasneje direktno pošlje naslovniku preko Btx modula.

Color Graf — 64 kb delovnega pomnilnika in 9 barv

Burian Saša, ki je avtor Color Grafa, je zasnoval sistem, ki ima zelo dobre zmogljivosti. Odlikuje ga zelo dobra grafika z visoko natančnostjo — pod programsko kontrolo je skoraj 50000 točk na zaslonu, kar je enako kot pri Apple IIe. Vsako točko je moč prižgati, ugasniti, jo kontrolirati in spreminjati barvo. Računalnik lahko dela v 12 semigrafičnih načinih od 64 × 64 do 64 × 192 točk in 9 barvah, v 8 grafičnih načinih od 64 × 64 do 256 × 192 točk in v alfanumeričnem načinu 32 × 16 znakov. Grafika je barvna (8 + 1 barva) in omogoča zaradi hitrosti kar dobre animacije. Color Graf ima enak mikroprocesor kot HR-84, kompatibilen pa je tako po strojni kot programski opremi s Tandyjevim modelom TRS80 Colour Computer in delno z računalnikom Dragon 32, kar pomeni, da je uporabnikom Color Grafa na voljo zelo velik izbor programske in strojne opreme. V osnovni verziji ima računalnik 16 kb ROM, vendar obstaja možnost priključitve dodatnih 8 kb ROM pomnilnika v okviru posebnih vtičnih modulov, ki jih je za TRS80 na voljo precej.

Color Grafu je moč vgraditi tudi pascal in macroassembler — editor in operacijski sistem OS — 9, ki podpira do štiri gibke diske. Ima štiri analogno digitalne vhode, na katere je moč poleg krmilne ročice priključiti naprave za razna fizikalna merjenja. Obstaja pa tudi en digitalno analogni vhod.

Vsekakor gre pri Color Grafu za računalnik, ki zaradi karakteristik, ki jih že ima, in programske ter strojne podpore praktično predstavlja vrh jugoslovanskih hišnih računalnikov. Namenjen je za uporabo v šolah, kot poslovni računalnik, kot terminal in kot osebni računalnik.

Začetno partnerstvo med ustvarjalcem in Slovenijales inženiringom doživlja nihanja in če ne bo moč drugače koncipirati proizvodnje, bodo Color Graf proizvajali v manjših serijah in v okvirih, kot jih dovoljuje obrtna proizvodnja.

Perspektive domače računalniške industrije

V zadnjem obdobju je prišlo na domačem tržišču do nekaterih sprememb. Za dinarje sta na voljo Sinclair ZX Spectrum (16 kb) in Apple II, in to nekajkrat dražja kot v zahodni Evropi. Začenja tudi konsignacijska prodaja. Na voljo so proizvodi Commodore (C64, C116, C16, C plus/4 in drugi), Sharp (MZ 700, PC 1500A, PC 1401, PC 1251 in drugi) ter Philips (za zdaj le G 7000).

Razvojno gledano je smiselno omogočiti uvoz računalnikov vseh cenovnih in zmogljivostnih področjih, kjer domače ponudbe ni, ali pa je tako draga ali tehnološko podrazvita, da je nestimulativna za potencialne kupce.

Po začetnem obdobju, za katerega je značilna stihija in poskusi širšega kroga proizvajalcev za prodajo na tržišče, bo potrebno usmeriti prizadevanja domače mikroračunalniške industrije v uresničevanje enotnih ciljev. Za mikroračunalniško področje kaže, da bodo imeli v prihodnje uspeh na tržišču sistemi, kjer bodo prevladovale med drugimi naslednje karakteristike:

- močna programska podpora, ki bo spodbujala uporabo računalništva v vsakodnevnem delu in življenju
- uporabniško prijazni sistemi, kot je npr. Applov model Macintosh
- visoka grafična natančnost
- medsebojna kompatibilnost domačih mikroračunalnikov — pretehtati bo potrebno izhodišča za kompatibilnost domačih mikroračunalnikov z vidika procesorjev, operacijskih sistemov, nosilcev podatkov, sestavnih delov, tipkovnic in naborov znakov, zaslonov in sistemov
- takšna opremljenost računalnikov, ki bo omogočala povezovanje posameznih enot v računalniške mreže

Mikroračunalnik »made in YU« bo moral biti tako koncipiran, da ne bo zaviral, temveč spodbujal uveljavljanje računalništva v naši družbi.

Epson HX-20

Ne samo, da je ta računalnik lahko prenosljiv, ampak ga lahko uporabljamo celo na potovanjih

Epson HX-20 je prvi resnično prenosen računalnik. Ker se v celoti programira v basicu, omogoča veliko širšo uporabo kakor najzahtevnejši kalkulatorji. Ker vsebuje vse, kar potrebujemo za delo, in je lažji od dveh kilogramov, ga lahko prenašamo v torbi. Prodaja se praktično mikroračunalniškim uporabnikom vseh vrst: navdušencem za domačo uporabo, poslovnežem in inženirjem. Ker je izdelan na popolnoma novih osnovah, je za zdaj na voljo sorazmerno malo programske opreme. Vseeno pa je zelo primeren za učenje programiranja v basicu. Za Epson se lahko odločijo vsi tisti, katerih namen je napisati program za svoje posebne potrebe — zavarovalniški agent na primer za izračun premij, jadralec program za pomoč pri navigaciji, časnikar lahko sprogramira računalnik za zapiske.

Računalnik ima zaslon s tekočimi kristali, ki lahko prikaže štiri vrstice s po 20 znaki ali preprosto grafiko s 120 × 32 točkami. V večini primerov deluje zaslon kot okno, ki ga lahko s kurzorskimi tipkami (označene so s puščicami) premikamo tako, da nam pokaže katerikoli del dosti večjega besedila, ki ga računalnik obdeluje. Tiskalnik je vdelan in uporablja petcentimetrske zvitke običajnega papirja, na katerega lahko izpiše po 24 znakov v vrstico.

Mikrokasetno enoto je potrebno doplačati, čeprav se, ker jo skoraj vsakdo zahteva, prikazuje kot del osnovne naprave. Mesto, kjer je mikrokaseta, je namenjeno tudi za programsko opremo na vtičnih modulih, vendar jih za zdaj še ni na voljo. Rešitev z mikrokaseto je boljša od navadnega kasetofona, ker računalnik zaznava položaj traku in zato lahko pri iskanju programa ali potrebnega podatka hitro previja trak.

Cela vrsta vmesnikov kaže na možnost priključitve raznovrstne opreme, saj je na voljo celo priključek za bralnik črtne kode. Delovni pomnilnik s 16 kb se lahko z uporabo dodatnega modula razširi na 32 kb.

Strojna in programska oprema, ki jo lahko dokupimo, omogoča, da po telefonski liniji povežemo XH-20 s podobnimi ali večjimi glavnimi računalniki. Tako se dobi dostop do središčnega vira informacij.

Tipkovnica Epsona HX-20
Tipkovnica je takšna kot na pisalnih strojih in določa dimenzije računalnika. Čeprav so tipke drugačne kot pri pisalnem stroju, je bila tipkovnica pri uporabnikih dobro sprejeta. Poleg številk v zgornji vrstici se s pritiskom na »NUM« spremenijo tipke za U, I, O, J, K, L in M v tipke za številke; to olajšuje vpisovanje večje količine številčnih podatkov. Tipke za kurzorske premike in editiranje so zgoraj desno, skupaj s tipko SCRN, ki omogoča premikanje zaslonskega okna navzgor in navzdol po besedilu.
Pet programljivih funkcijskih tipk (PF1 in PF5) se tudi po videzu razlikuje od drugih. Dve se uporabljata tudi za krmiljenje mikrokasete in kopiranje vsebine zaslona na tiskalnik.

Vdelani tiskalnik
Majhni tiskalniki po sistemu matrike pik se že dolgo uporabljajo v registrskih blagajnah in celo kalkulatorjih, zato jih je Epson, ki je pričel kot izdelovalec tiskalnikov, hitro uporabil v svojih prenosnih HX-20 računalnikih. Ta tiskalnik matrike pik izpisuje grafiko prav tako dobro kakor tekst.

Električni priključek
Na tem mestu se HX-20 lahko preko transformatorja priključi na omrežje. Pri tem se polnijo baterije.

Reset tipka

Priključek za črtno kodo
Tu se lahko priključi pero za branje črtne kode.

Priključek za kasetofon
Uporablja se za navaden kasetofon in omogoča tudi nadzor motorja. Rešitev še vedno ni tako učinkovita kakor mikrokaseta.

Tiskano vezje
Zaradi jasnosti je na sliki tiskano vezje obrnjeno. Integrirana vezja so običajno obrnjena navzdol. Nekatera imajo podnožja, to omogoča lažjo menjavo.

Mikroprocesorji
Dva mikroprocesorja 6301 nadzirata računalnik in vmesnike. Vsak ima po 4 kb bralnega pomnilnika in 128 zlogov delovnega pomnilnika.

Delovni pomnilnik
HX-20 ima v osnovni izvedbi 16 kb delovnega pomnilnika v osmih 2 kb integrnih vezjih.

Hranjenje velike količine podatkov
Vdelani mikrokasetofon, ki uporablja kasete za žepne diktafone, je dosti boljši od običajnega kasetofona. Ker računalnik nadzira hitro previjanje, se bistveno skrajša čas dostopa do določenega mesta na traku. Poleg tega je mikrokasetofon vstavljen v mikroračunalnik tako, da niso potrebne dodatne žice.

Snetje modula
Ta gumb omogoča, da snamemo mikrokasetofon. Če hočemo računalnik uporabljati samo za en namen, lahko v to mesto vtisnemo vtični modul.

Nastavljanje zornega kota
Zasloni s tekočimi kristali se morajo običajno gledati pod natanko določenim kotom. Z obračanjem tega gumba si lahko uporabnik nastavi najprimernejši zorni kot.

Stikalo vklop/izklop

Serijski vmesnik
Čeprav je RS232 tudi serijski vmesnik, je ta kanal namenjen različnim perifernim napravam, ki jih bo Epson sčasoma še vpeljal.

Vrata RS232
To je vsesplošno sprejet serijski vmesnik, ki se uporablja za povezavo s tiskalnikom ali modemom, s katerim se preko telefona povezujemo z drugimi računalniki.

Vmesnik za razširitev pomnilnika
Temeljnih 16 kb pomnilnika se lahko podvoji, če dodamo Epsonov vtični modul, ki se oblikovno ujema z ostalo napravo.

Bralni pomnilnik
Štiri integrirana vezja dajo skupaj 32 kb lokacij. Prosto podnožje kaže na to, da se računalnik lahko, če se programska oprema zapiše v bralne pomnilnike, uporablja kot strogo uporabniški stroj.

Nov pogled
Velik del prostora v mikroračunalnikih zavzemajo katodne cevi, ki so jedro zaslonov. Epson se je teh problemov pri HX-20 rešil z uporabo zaslona s tekočimi kristali.
Taki zasloni se že dolgo uporabljajo v kalkulatorjih, toda v tem primeru je bil dosežen velik napredek v kvaliteti, saj lahko zorni kot na matriko 4 × 20 nastavljamo po želji.

Epson HX-20

VELIKOST
290 × 210 × 45 mm

TEŽA
1,7 kg

CPU
6301 Hitachi

TAKT
2,45 MHz

POMNILNIK
16 kb delovnega pomnilnika, z možnostjo razširitve na 32 kb. 32 delovnega pomnilnika, ki se lahko razširi na 64 kb.

ZASLON
Zaslon s tekočimi kristali (zorni kot je nastavljiv) s 4 vrstami po 20 znakov, ki deluje kot okno v področju do 255 × × 255 znakov. Vsaka od 120 × 32 točk na zaslonu je naslovljiva.

VMESNIKI
Vmesnik za razširitev spomina, RS232 (za zunanji tiskalnik), serijski vmesnik (za bodočo diskovno enoto) in priključek za bralnik črtne kode.

PROGRAMSKI JEZIK
Basic

DRUGI PROGRAMSKI JEZIKI
Forth interpreter

OPREMA
Lepo oblikovano plastično ohišje za prenašanje, napajalnik, priročnik z navodili.

TIPKOVNICA
60 tipk je podobnih tistim na pisalnem stroju, pet je programljivih funkcijskih tipk in štiri za določene namene.

DOKUMENTACIJA
Z računalnikom dobite dva priročnika: operacijskega in priročnik za basic. Oba sta zelo jasno pisana, vendar jima manjka kazalo. Epson si je izbral Microsoftov basic, ki je industrijski standard. Priročnik je dovolj dober, da se lahko uporabi za učno knjigo ali za obnovitev znanja. Operacijski priročnik vsebuje takšne podrobnosti, kakor so: celoten seznam vhodno-izhodnih priključkov za RS232 in serijski vmesnik; dovoljene vhodne nivoje ter pregledne tabele razdelitve pomnilnika.

3. poglavje

Osnovni pojmi

Podobno kot druga strokovna področja, ima tudi računalništvo svoje posebne pojme. Poskusimo se seznaniti z njimi.

Biti in zlogi

Računalnik ne razume nič drugega kot števila, ki pa imajo za nas precej čudno obliko

Besedi »bit« in »zlog« uporabljamo vedno, kadar govorimo o računalnikih. S tema izrazoma opisujemo način, kako računalniki hranijo in uporabljajo števila.

To počnejo precej drugače kakor ljudje. Mi predstavljamo števila z desetimi različnimi simboli (0 do 9), ki jih uporabljamo kot mnogokratnike števila 10 (to je znano kot desetiški sistem). Računalniki pa za vse svoje matematične čarovnije uporabljajo samo dva simbola: 0 in 1. Biti in zlogi so način predstavljanja kombinacij teh dveh simbolov.

Bit je najmanjši del informacije, s katerim računalnik še lahko operira. Računalnik »ve«, ali je na določenem mestu znak 0 ali 1. Skupino osmih bitov imenujemo zlog (angleški izraz je »byte«). Z zlogom lahko računalnik predstavi prav velika števila.

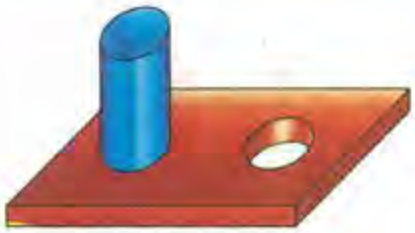
Najprej si poglejmo, kaj so biti in zakaj se tako imenujejo. Računalniki so elektronske naprave in zato delajo z električnimi signali. Električni signal v določeni žici je lahko priključen (v žici je električna napetost) ali pa izključen (ni napetosti). Stanje, ko je v žici napetost, predstavlja znak 1, stanje, ko te napetosti ni, pa predstavlja znak 0.

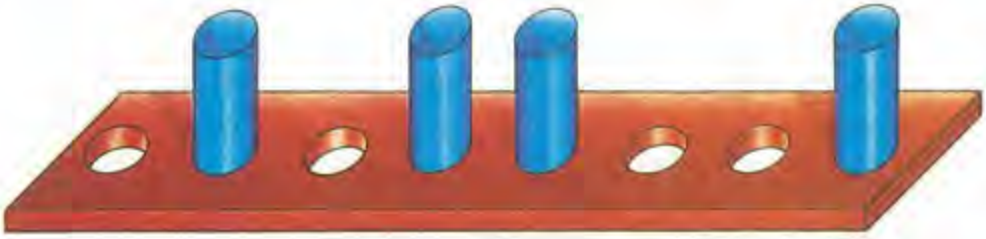
Ilustracija kaže deščico z luknjo, v katero lahko vstavimo zatič. Čeprav je luknja ena sama, lahko predstavlja dve stanji, to je podobno kakor pri žici v računalniku. Luknja je lahko prazna, to nam predstavlja znak 0, ali pa je v njej zatič, to predstavlja znak 1. Deščica z eno luknjo lahko torej predstavlja znaka 0 ali 1.

Kot smo že prej videli, lahko v računalniku dosežemo isto z električnim signalom v žici: ko je priključen, predstavlja 1, ko je izključen, pa 0. Poznavanje dejstva, da je na nekem mestu simbol nič ali ena, je najmanjši del informacije in se imenuje bit. Bit ima torej dve možni stanji: 0 ali 1. Beseda bit je okrajšava angleškega izraza **BI**nary digi**T**, kar pomeni dvojiški znak. Gledano drugače, bit lahko tudi šteje, vendar le od 0 do 1.

Biti in zlogi

Bit je najmanjša enota informacije, s katero lahko računalnik operira. Deščica z eno luknjo lahko predstavlja 0 ali 1, v odvisnosti od tega, ali je v luknjo vstavljen zatič ali ne. Računalnik dela isto z električnim signalom, ki je izključen (0) ali priključen (1). Deščica z dvema luknjama ima štiri možne razporeditve zatičev. Če pa uporabimo deščico z osmimi luknjami, število možnih kombinacij naraste na 256. Te kombinacije si lahko ogledamo v tabeli. Računalnik uporablja bite v skupinah po osem. Takšno skupino imenujemo zlog. Vsak zlog lahko predstavlja število od 0 do 255.

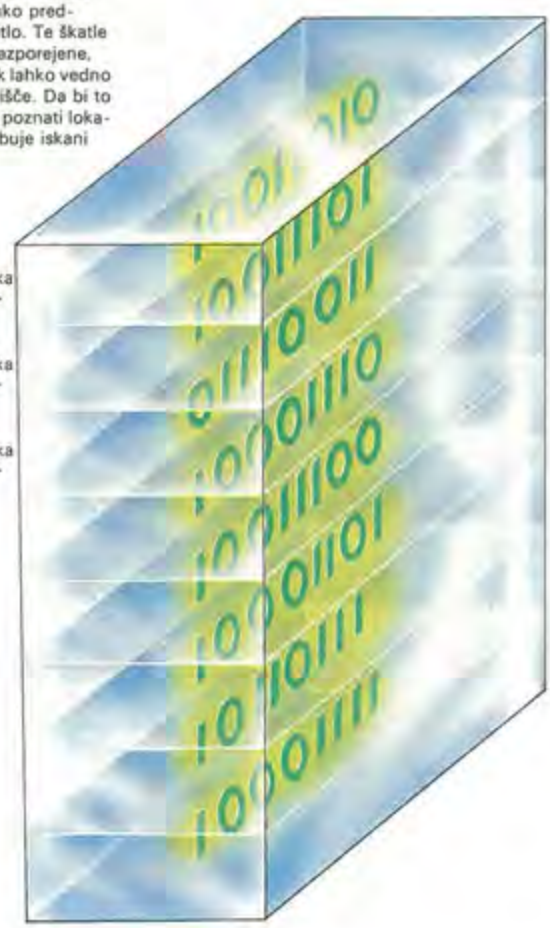




0	00000000	128	10000000
1	00000001	129	10000001
2	00000010	130	10000010
3	00000011	131	10000011
4	00000100	132	10000100
5	00000101	133	10000101
6	00000110	134	10000110
7	00000111	135	10000111
8	00001000	136	10001000
9	00001001	137	10001001
10	00001010	138	10001010
11	00001011	139	10001011
12	00001100	140	10001100
13	00001101	141	10001101
14	00001110	142	10001110
15	00001111	143	10001111
16	00010000	144	10010000
17	00010001	145	10010001
18	00010010	146	10010010
19	00010011	147	10010011
20	00010100	148	10010100
21	00010101	149	10010101
22	00010110	150	10010110
23	00010111	151	10010111
24	00011000	152	10011000
25	00011001	153	10011001
26	00011010	154	10011010
27	00011011	155	10011011
28	00011100	156	10011100
29	00011101	157	10011101
30	00011110	158	10011110
31	00011111	159	10011111
32	00100000	160	10100000
33	00100001	161	10100001
34	00100010	162	10100010
35	00100011	163	10100011
36	00100100	164	10100100
37	00100101	165	10100101
38	00100110	166	10100110
39	00100111	167	10100111
40	00101000	168	10101000
41	00101001	169	10101001
42	00101010	170	10101010
43	00101011	171	10101011
44	00101100	172	10101100
45	00101101	173	10101101
46	00101110	174	10101110
47	00101111	175	10101111
48	00110000	176	10110000
49	00110001	177	10110001
50	00110010	178	10110010
51	00110011	179	10110011
52	00110100	180	10110100
53	00110101	181	10110101
54	00110110	182	10110110
55	00110111	183	10110111
56	00111000	184	10111000
57	00111001	185	10111001
58	00111010	186	10111010
59	00111011	187	10111011
60	00111100	188	10111100
61	00111101	189	10111101
62	00111110	190	10111110
63	00111111	191	10111111
64	01000000	192	11000000
65	01000001	193	11000001
66	01000010	194	11000010
67	01000011	195	11000011
68	01000100	196	11000100
69	01000101	197	11000101
70	01000110	198	11000110
71	01000111	199	11000111
72	01001000	200	11001000
73	01001001	201	11001001
74	01001010	202	11001010
75	01001011	203	11001011
76	01001100	204	11001100
77	01001101	205	11001101
78	01001110	206	11001110
79	01001111	207	11001111
80	01010000	208	11010000
81	01010001	209	11010001
82	01010010	210	11010010
83	01010011	211	11010011
84	01010100	212	11010100
85	01010101	213	11010101
86	01010110	214	11010110
87	01010111	215	11010111
88	01011000	216	11011000
89	01011001	217	11011001
90	01011010	218	11011010
91	01011011	219	11011011
92	01011100	220	11011100
93	01011101	221	11011101
94	01011110	222	11011110
95	01011111	223	11011111
96	01100000	224	11100000
97	01100001	225	11100001
98	01100010	226	11100010
99	01100011	227	11100011
100	01100100	228	11100100
101	01100101	229	11100101
102	01100110	230	11100110
103	01100111	231	11100111
104	01101000	232	11101000
105	01101001	233	11101001
106	01101010	234	11101010
107	01101011	235	11101011
108	01101100	236	11101100
109	01101101	237	11101101
110	01101110	238	11101110
111	01101111	239	11101111
112	01110000	240	11110000
113	01110001	241	11110001
114	01110010	242	11110010
115	01110011	243	11110011
116	01110100	244	11110100
117	01110101	245	11110101
118	01110110	246	11110110
119	01110111	247	11110111
120	01111000	248	11111000
121	01111001	249	11111001
122	01111010	250	11111010
123	01111011	251	11111011
124	01111100	252	11111100
125	01111101	253	11111101
126	01111110	254	11111110
127	01111111	255	11111111

Zlogi v pomnilniku

Zlogi so skupine osmih bitov. Zloge, ki predstavljajo števila od 0 do 255, spravi računalnik v svoj pomnilnik. Vsak zlog spravi v ločeno pomnilniško celico, ki si jo lahko predstavljamo kot škatlo. Te škatle so sistematično razporejene, tako da računalnik lahko vedno najde zlog, ki ga išče. Da bi to lahko storil, mora poznati lokacijo škatle, ki vsebuje iskani zlog.



Deščica z dvema luknjama lahko predstavlja štiri različna stanja, torej lahko šteje od 0 do 3. Imamo štiri možnosti: obe luknji sta lahko prazni (0), desna ima zatič (1), leva ima zatič (2), obe imata zatič (3).

Na sliki vidimo tudi deščico z osmimi luknjami. Obstaja kar 256 načinov, kako lahko razporedimo čepe v teh osem lukenj. Tabela, ki uporablja ničle za prazne luknje, in enke za luknje z zatiči, nam kaže vse te možnosti.

Takšno skupino osmih bitov imenujemo zlog. Z zlogom lahko torej predstavimo kar 256 različnih stanj (šteje lahko od 0 do 255).

Ko rečemo, da računalnik »shrani« zlog, mislimo s tem, da se število v območju od 0 do 255 shrani na določeno mesto v pomnilniku računalnika. Pomnilnik računalnika je urejen kot skupina »škatel«, v katere spravlja zloge (na sliki so te škatle zložene ena vrh druge). Računalnik mora seveda vedeti, v katero škatlo je določen zlog spravil, če ga hoče najti, ko ga spet potrebuje.

Vsa števila od 0 do 255 lahko predstavimo s kombinacijo osmih bitov (tabela na levi). Bite računalnik spravlja in uporablja v skupinah po osem, ki se imenujejo zlogi (angl. byte).

Ko je 1 + 1 = 10

Računalniki dosegajo svoje velikanske računske zmožnosti z uporabo samo dveh znakov: 0 in 1

Spreminjanje iz dvojiškega sistema

Najlažje pretvorimo dvojiška števila v njihove desetiške ekvivalente tako, da si napišemo vrednost vsakega dvojiškega mesta na prste desne roke. Če dvojiško število ni daljše od štirih mest, moramo samo vzdigniti prst, ki ustreza določenemu mestu. Če je znak na tem mestu 0, prsta ne vzdignemo, če je ena, pa ga vzdignemo.
Ko vzdignemo prste za 1010, dobimo številki 8 in 2. Če ju seštejemo, dobimo desetiško število 10. Tretja ilustracija kaže, kako dekodiramo 0101. Dobimo 4 in 1, rezultat je 5 v desetiški obliki. Poskušajte določiti še desetiške ekvivalente števil 1110 in 0110.
Metodo lahko z uporabo obeh rok razširimo za izračun dvojiških števil do osem mest. Prste leve roke označimo s 16, 32, 64 in 128, tako da gledamo v dlan in na mezinec zapišemo 16, druga števila pa si sledijo po vrsti proti levi.

Večini ljudi se zdi naš številski sistem tako normalen in logičen, da tudi pomislijo ne, da je mogoče predstavljati števila še na druge načine.

Rimljani so na primer imeli sistem za predstavljanje števil s črkami: X so uporabili za 10, L za 50, C za 100, D za 500 in tako naprej. S tem sistemom so lahko zapisovali preprosta števila. Z računanjem pa so bile težave. Celo pri seštevanju rimskih številk so težave. Razlog je v tem, da rimske številke nimajo koncepta mestne »vrednosti«. Mesto, na katerem stoji znak v rimski številki, namreč nič ne pove o njegovi »vrednosti«.

Zdaj pa si poglejmo števili 506 in 56 v našem sistemu. Edina razlika je ničla med njima. Njena vloga v številu 506 je v tem, da pove, da v številu ni »desetic«. Ostala dva znaka povesta, da je v številu pet »stotic« in šest »enic«. Vsako mesto v številu ima torej svojo vrednost. Najbolj desno mesto je mesto enic, drugo z desne je mesto desetic, naslednje so stotice, pa tisočice in tako naprej. Številka na določenem mestu pove, kolikokratno vrednost tega mesta vsebuje ta številka.

Najbrž se že sprašujete, kaj ima to opraviti z računalniki. Računalniki so elektronske naprave, ki lahko delajo s števili, uporabljajoč različne napetosti v žicah. Pet voltov predstavlja enko, nič voltov pa ničlo. Kot smo se naučili v prejšnjem poglavju, lahko z enkami in ničlami predstavljamo števila.

Če želimo predstaviti večje število, vzamemo pač večje število bitov.

Denimo, da imamo palico z mnogimi zarezami. Ko jih preštejemo, ugotovimo, da jih je petsto šest. V nam domačem desetiškem (s tujko dekadnem) sistemu bi to

število zapisali takole: 506. V dvojiškem (binarnem) sistemu, ki ima za znake samo ničle in enice, pa bi bilo število teh zarez na palici takšnole: 111111010. Tako so v računalniku predstavljena števila.

V dvojiškem sistemu je mestna vrednost vsake številke drugačna kakor v desetiškem. V desetiškem sistemu raste mestna vrednost s potencami števila 10 (1, 10, 100, 1000, ...), v dvojiškem pa s potencami števila 2 (1, 2, 4, 8, 16, ...). Mesta povsem na desni so vedno enice. Ker pa imamo samo dva znaka (0 in 1), nam znakov za to mesto zmanjka takoj, ko eni enici prištejemo še drugo. V desetiškem sistemu pa nam zmanjka znakov šele pri devetici.

V desetiškem sistemu zavzemajo drugo mesto desetice (ker imamo deset znakov od 0 do 9), v dvojiškem pa so drugo mesto dvojice. Tako nam dvojiško število 10 predstavlja desetiško število 2.

Oglejmo si še enkrat število petsto šest, zapisano v obeh sistemih, desetiškem in dvojiškem.

stotice	desetice	enice	
5	0	6	
$= 5 \times 100 +$	$0 \times 10 +$	6×1	$(= 506)$

256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	0	1	0

$$= 1 \times 256 + 1 \times 128 + 1 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 \qquad (= 506)$$

Pravila računanja v dvojiškem sistemu so povsem enaka kakor vsem dobro znana pravila v desetiškem sistemu. Edina razlika je v tem, da nam zmanjka simbolov za določeno mesto že po enici, ne šele po devetici. Poskusimo sešteti dve števili. V oklepajih so desetiški ekvivalenti dvojiških števil.

$$\begin{array}{rr} (3) & 11 \\ + (5) & + 101 \\ \hline (8) & 1000 \end{array}$$

(1 + 1 = 0 prenos 1)
(1 [prenešena] + 1 = 0 prenos 1)
(1 [prenešena] + 1 = 0 prenos 1)
(1 [prenešena] + 0 = 1)

V dvojiškem sistemu nam seštevanje 1 in 1 pomeni, da nam je zmanjkalo znakov, ker sta 0 in 1 edina znaka, ki ju imamo. Zato rečemo: »Ena in ena je nič, eno prenesemo naprej.« Prav tako v desetiškem seštevanju 1 in 9 pomeni, da nam je zmanjkalo simbolov, ker pač ni simbola, večjega od 9.

Zato rečemo podobno: »Ena in devet je nič, eno prenesemo naprej.« Poskusimo, ali smo se zdaj naučili seštevati v dvojiškem sistemu. Prvi rezultat vam še podajamo, druga dva pa izračunajte sami.

$$\begin{array}{rr} (4) & 100 \\ + (6) & + 110 \\ \hline (10) & 1010 \end{array} \qquad \begin{array}{rrrr} (7) & 111 & (3) & 11 \\ + (2) & + 10 & + (12) & + 1100 \\ \hline (?) & ? & (?) & ? \end{array}$$

Do zdaj ste najbrž opazili, da so dvojiška števila veliko daljša od desetiških ekvivalentov. Za konec poskusite sešteti števili 11010110 in 1101101. Skušajte tudi ugotoviti, za katera desetiška števila gre. Pri seštevanju pazite, da bodo poravnana skrajna desna mesta, ker seštevate števili različnih dolžin.

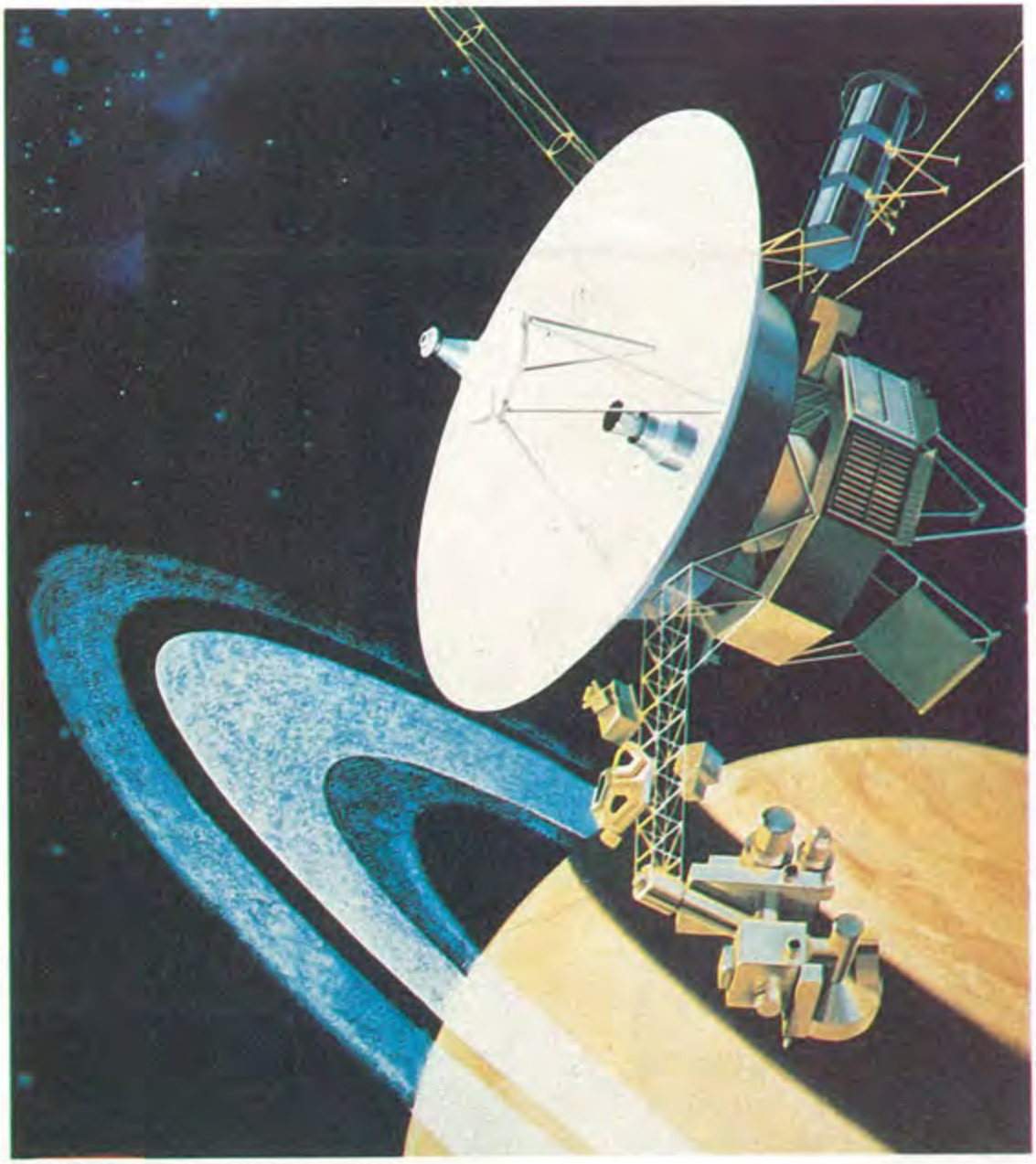
Zgodovina številk

Babilonske Rimske Hindujske Dvojiške

Stari Babilonci so imeli zelo razvit številski sistem, ki je imel za podlago 60 in ne 10 kakor naš. Gornja slika kaže klinopisni simbol za 59. Uporaba števila 60 za podlago številskega sistema je imela veliko prednosti. Ostanke tega sistema lahko zasledimo še danes. Minuta ima 60 sekund, ura ima 60 minut, krog ima šestkrat 60 stopinj. Vse to so ostanki matematičnega sistema, ki se je do vrhunca razvil pred 4000 leti.
Sistem, ki so ga imeli Rimljani, je bil precejšen korak nazaj. Za predstavitev števil so uporabili kar črke abecede, mesto vsakega znaka pa ni imelo nobene zveze z njegovo vrednostjo. Zato je bilo celo preprosto računanje nemogoče.

Hindujci v Indiji so uporabili devet znakov za števila od 1 do 9, kasneje pa so dodali še znak za nič. Poglavitna ideja v njihovem sistemu je bila zamisel mestne vrednosti. Za vsako mesto v številu je določeno, koliko je »vredno«. Tako je znak 3 v 30 »vreden« tri desetice. Hindujski sistem so prevzeli Arabci, od njih pa se je razširil v Evropo. Arabci so imeli nekaj zelo pomembnih matematikov, ki so s svojimi deli zelo vplivali na razvoj matematike.
Z razvojem računalnikov pa se je uveljavil dvojiški sistem, ker lahko vsa števila, ne glede na velikost, predstavimo s samimi enkami in ničlami.

Seštevanje in odštevanje?



Voyager 2
Slavni Voyager je bil prvo vesoljsko plovilo, ki je potovalo skozi naš sončni sistem. Med svojim potovanjem je zbiral fotografije in druge podatke, katere je njegov računalnik spreminjal v podatke v dvojiških številih. Tako spremenjene informacije je nato pošiljal na Zemljo s hitrostjo 116000 bitov v sekundi. Te informacije je nato obdeloval računalnik v Houstonu v Teksasu pri agenciji NASA.

Odštevanje računalnik prevede v seštevanje in si tako poenostavi problem

V prvem delu te serije smo ugotovili, da lahko vsako desetiško število prikažemo z binarnimi ali dvojiškimi števili. Dvojiška števila imajo slabost, da so daljša od ustreznih desetiških, kljub temu pa so primernejša za računalnik, ker le-ta razume ničlo kot nič in enko kot pozitivno napetost. Videli bomo, da lahko dvojiška števila seštevamo zelo preprosto.

Na papirju jih odštevamo enako lahko kakor desetiška, držeč se pravil, ki veljajo za desetiško odštevanje. Načrtovalci računalnikov pa so že zelo zgodaj spoznali, da elektronska seštevalna vezja zmorejo seštevanje in odštevanje, tako da ni treba odštevalnih vezij. Poglejmo, kako je to mogoče.

Dvojiški komplement

Metoda za ponazoritev negativnega števila v računalniku je znana kot »dvojiški komplement«. Tako proces odštevanja prevedemo v del seštevanja. Poglejmo si naslednji aritmetični problem:

$$16-12=4$$
$$\text{ali } 16+(-12)=4$$

Tu je bilo 12 odvzeto od 16, toda proces odštevanja smo prevedli v seštevanje: številu 16 smo prišteli negativno število 12. V obeh primerih smo dobili enak rezultat, edina razlika je v uporabi različnih aritmetičnih znakov in oklepaja. To neznatno spremembo uporabi računalnik, da izraža negativna števila in hkrati poenostavi problem odštevanja. Da bo stvar lažja, vzemimo, da lahko naš računalnik izvaja matematične operacije s petmestnimi dvojiškimi števili. Seveda resnični računalniki izvajajo matematične operacije z večtisočmestnimi dvojiškimi števili. Prikažimo metodo na našem petmestnem računalniku. Prvo število na skrajni levi je ločeno od ostalih štirih. Če je to številka »1«, pomeni negativni predznak in če je »0«, je to seveda pozitivni predznak. Preostala štiri števila pa nam predstavljajo število, zapisano v dvojiškem sestavu, ki smo se ga naučili že v prejšnjih sestavkih.

( )	( )	( )	( )	( )
−16 ali 0	8	4	2	1 ali 0

Za zgled vzemimo dvojiško število 01000, ki nam predstavlja desetiško 8, 10000 pa nam predstavlja desetiško −16. Kaj pa, če imamo 10100? To vsebuje −16 in +4, dobimo −12.

Koliko števil lahko prikažemo s petimi mesti, če uporabljamo to pretvorbo? Največje število je 01111 ali desetiško 15, največje negativno 10000 ali −16. S preskusi hitro ugotovimo, da lahko zapišemo katerokoli število med −16 in +15.

dvojiško	desetiško
10000	−16
10001	−15
10010	−14
10011	−13
10100	−12
itd.	
11111	− 1
00000	0
00001	1
00010	2
itd.	
01110	14
01111	15

Če bi povečali število mest v računalniku, bi lahko z njim seveda računali v širšem razponu števil.

V začetku razvoja dvojiške računalniške aritmetike je bil odkrit preprost trik za določitev dvojiškega komplementa oziroma negativne oblike kakega števila. Ta trik ima dve stopnji.

Prva stopnja je ta, da vsakemu mestu priredimo nasprotno vrednost. Če je na nekem mestu 1, jo spremenimo v 0, in če je nekje 0, jo spremenimo v 1. Na naslednji stopnji prištevamo temu nasprotnemu številu vrednost 1.

Sledimo sedaj tej metodi v naslednjem primeru. Imeli smo število +10, ki ima dvojiški ekvivalent 01100. (Prva ničla na skrajni levi strani ni obvezna, tako da lahko pišemo 1100, ker pa ima naš računalnik 5 mest, moramo uporabiti vsa.)

	01100 (= +12)
prva stopnja	10011
druga stopnja	00001 (+1)
	10100 (= −12)

Poglejmo sedaj, kako se računalnik loti odštevanja števil, recimo: 12 minus 4.

+ 12 je	01100	(uporabljen je
− 4 je	11100	dvojiški komplement)
12 + (−4)	101000	

Vidimo, da imamo v rezultatu uporabljenih 6 mest. Ker naš računalnik lahko zazna samo 5-mestno število, imenujemo skrajno levo število prekrmiljenje, ki ga naš računalnik ne zazna. Ostane torej 01000 ali desetiško 8, kar je pravilen rezultat! Bolj zapleten primer pa je 4 minus 12.

+ 4 je	00100
−12 je	10100
4 + (−12)	11000

Kot zadnji zgled si poglejmo še seštevanje negativnih števil: $-3-4=-3+(-4)=-7$

3 je	00011	
tako je −3	11101	(uporabljen je
in −4 je	11100	dvojiški komplement)
	111001	

Zopet imamo 6-mestno število. Ko prekrmiljenje izločimo, dobimo 11001 ali desetiško −7.

Videli smo, da izvedemo odštevanje s seštevanjem z uporabo »dvojiškega komplementa« (ki se sestoji iz zamenjave 0 v 1 in 1 v 0 ter iz prištevanja 1). Prednost računalnika je v tem, da lahko dvojiškim številom preprosto priredi obrnjeno vrednost z vrati NE. Vrata NE imajo en vhod in izhod. Vrednost na izhodu je vedno nasprotna vrednosti na vhodu. Če je na vhodu 0, je na izhodu 1, in če je na vhodu 1, je na izhodu 0. Ta lastnost, invertiranje, je natančno to, kar potrebujemo za prvo stopnjo trika za računanje dvojiškega komplementa. V naslednjem poglavju bomo videli, kako seštevanje v računalniku zlahka opravimo z uporabo kombinacije logičnih vrat.

Pike in črte
Morsova koda je ena zgodnjih oblik ponazoritve dvojiških števil v elektroniki. Leta 1837 je bila narejena prva električna brzojavna povezava v Londonu. Dolga je bila okrog 4 km in je povezovala Euston z železniško postajo Camden Town. Nekaj let pozneje je v Ameriki Samuel Morse prikazal svojo slavno kodo za prenos poročil. Vsaka črka in številka je bila kombinacija dveh vrst signalov: pik in črt.

Množenje v dvojiškem sestavu

Četudi računalniki hitro odgovarjajo na zapletene aritmetične programe, jih računajo nadvse preprosto

V prejšnjem poglavju smo ugotovili, kako računalnik sešteva. Oglejmo si sedaj še problem množenja.

Če želimo pomnožiti 14 in 12, bi bilo najbolj preprosto, da to naredimo z večkratnim seštevancem, na primer 14 + 14 + 14 + 14 + ... (12-krat). Množenje, prevedeno v obliko zaporednega prištevanja, so uporabili pri prvih realizacijah množenja z računalnikom. Ta metoda pa je zelo nerodna in časovno zamudna, zato so računalniški strokovnjaki razvili uspešnejšo metodo.

Kadar množimo dve števili, operacijo običajno na papirju napišemo v naslednji obliki:

```
  14
 ×12
 ---
  28
 +14
 ---
 168
```

Enako pa lahko računamo v kateremkoli številskem sestavu. Poglejmo zgled v dvojiškem sestavu:

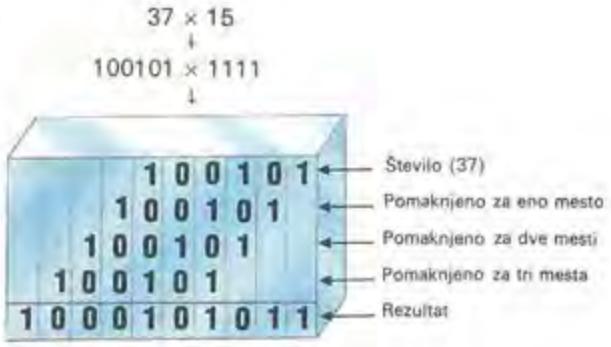
```
   101
  ×11
  ----
   101
  +101
  ----
  1111
```

Metoda je enaka tudi pri večjih številih. Povrnimo se torej k primeru 14 × 12 in opravimo množenje v dvojiškem sestavu:

```
    1110    (14)
   ×1100    (12)
   -----
    0000
   0000
  1110
 1110
 --------
 10101000   (168)
```

Množenje v dvojiškem sestavu je celo preprostejše kakor v desetiškem, ker nikoli ni števil, ki bi jih prenašali naprej. Če množimo z 1, je število nespremenjeno: 14 × 1 = 14, in če množimo z 0, je rezultat 0, kajti 14 × 0 = 0. To pravilo velja v dvojiškem, desetiškem ali kateremkoli drugem številskem sestavu.

Če pogledamo na to izračunavanje z matematičnega stališča, ugotovimo preprost vzorec dvojiškega množenja, ki ga sestavljata samo dve operaciji: pomik in seštevanje.



Tako množijo števila tudi računalniki. Računalnik pomakne kopijo števila iz gornje vrstice v vrstico pod njo na ustrezno mesto, ki ga določa številka 0 oziroma 1, in končno sešteje vse vrstice.

Računalnik potrebuje za množenje zmožnost računanja z velikomestnimi števili. Če medsebojno pomnožimo 4-mestni števili 1110 in 1110, je rezultat sestavljen iz 8-mestnega števila 10101000. V splošnem pa velja, da je rezultat množenja dveh števil lahko več ko dvakrat daljši od največjega izmed števil.

Preseneti pa nas spoznanje, da lahko z računalnikom zmnožimo napačno. Skoraj vse te napake lahko pojasnimo s tem, koliko prostora je konstruktor računalnika namenil za odgovor.

Če je prostor za rezultat premajhen, se pojavi prekrmiljenje. Tako izgubimo zadnjo številko in rezultat je napačen.

Pogled v množenje
Množenje v dvojiškem sestavu je dosti lažje kakor v desetiškem. Uporabljamo isto metodo kakor pri množenju desetiških števil. Poenostavitev je v dvojiškem sestavu mogoča zato, ker množimo samo z dvema vrednostma: 0 ali 1. Če množimo števila z 1, dobimo enak rezultat, kakor je število samo. Poglejmo primer, kjer 100101 (37) pomnožimo s 1111 (15). Spodaj zapišemo štirikrat prvo število, vsaka naslednja vrstica je pomaknjena za eno mesto v levo. Nato vse vrstice seštejemo in dobimo rezultat 1000101011 (555).

Gottfried Wilhelm Leibniz (1646—1716) je bil Newtonov sodobnik in ima veliko zaslug pri razvoju matematike, znanosti in filozofije. Iznašel je stroj za množenje in deljenje. Odkril je tudi možnost uporabe dvojiške aritmetike. (Dvojiška števila pa je prvi omenjal Francis Bacon leta 1623.) V svojih poznih letih je Leibniz prišel v spor z Newtonom glede tega, kdo je prvi iznašel diferencialni račun.

Pravilno ali napačno

Računalniki ne znajo misliti, vendar se ravnajo po zakonih logike

Centralno procesno enoto (CPU) pogosto opisujemo kot srce računalnika. V njej poteka računanje in logično odločanje. Kako pa to poteka?

Da bi to lahko razumeli, moramo poznati temelje dvojiške aritmetike in razumeti pomen logičnih vrat. Logična vrata so preprosta elektronska vezja, ki zmorejo opravljati logične odločitve in primerjave. Načela delovanja logičnih vrat so zelo preprosta. Poglejmo si jih ob nekaj zgledih iz vsakdanjega življenja.

Poznamo tri temeljne tipe logičnih vrat: vrata IN, vrata ALI in vrata NE. Pisali jih bomo z velikimi črkami, da jih ne bomo mešali z običajnimi besedami.

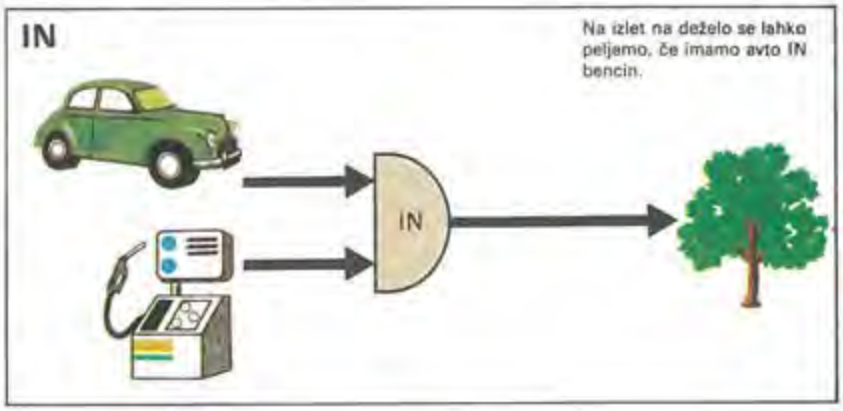
Logične zveze

Vrata IN so vezje, ki nam da »pravilen« izhod, če sta oba vhoda »pravilna«. Poglejmo si, kaj to pomeni. Vzemimo, da bi se radi peljali na izlet na deželo. Če imamo avto IN nekaj bencina, se lahko odpeljemo. Če imamo avto brez bencina, ali če imamo bencin, pa nimamo avta, se seveda ne moremo peljati. V vezju IN imamo torej dva vhodna pogoja, ki morata biti izpolnjena (pravilna), da je izhod pravilen. Trditev, da se bomo peljali na izlet, je pravilna, če imamo avto IN bencin.

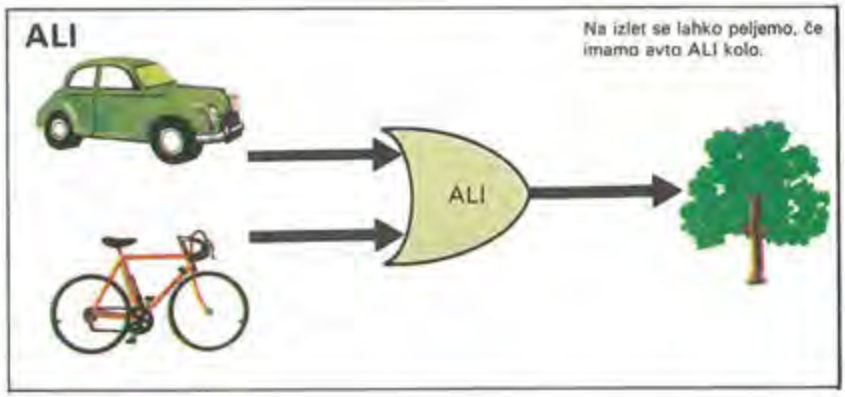
Zdaj pa si predstavljajmo malo drugačen položaj. Spet bi se radi peljali na izlet. To je mogoče, če imamo avto ALI kolo (vzemimo, da imamo bencin). Če imamo avto, lahko gremo. Prav tako lahko gremo, če imamo kolo. Doma moramo ostati le, če noben od teh dveh pogojev ni izpolnjen. Izhodna trditev vrat ALI je torej pravilna, če je pravilna vsaj ena vhodna trditev.

Tretja logična vrata, ki jih moramo še opisati, so vrata NE. Izhod teh vrat je nasprotje vhoda. Če je vhod pravilen, je izhod napačen, če pa je vhod napačen, je izhod pravilen. Vrnimo se k našemu izletu. Ne glede na to, ali bomo šli z avtomobilom ali s kolesom, se bomo lahko odpeljali le, če je napačna trditev, da je zračnica predrta. Če je trditev ob vhodu (predrta zračnica) pravilna, bo izhodna trditev (da se bomo peljali na izlet) napačna.

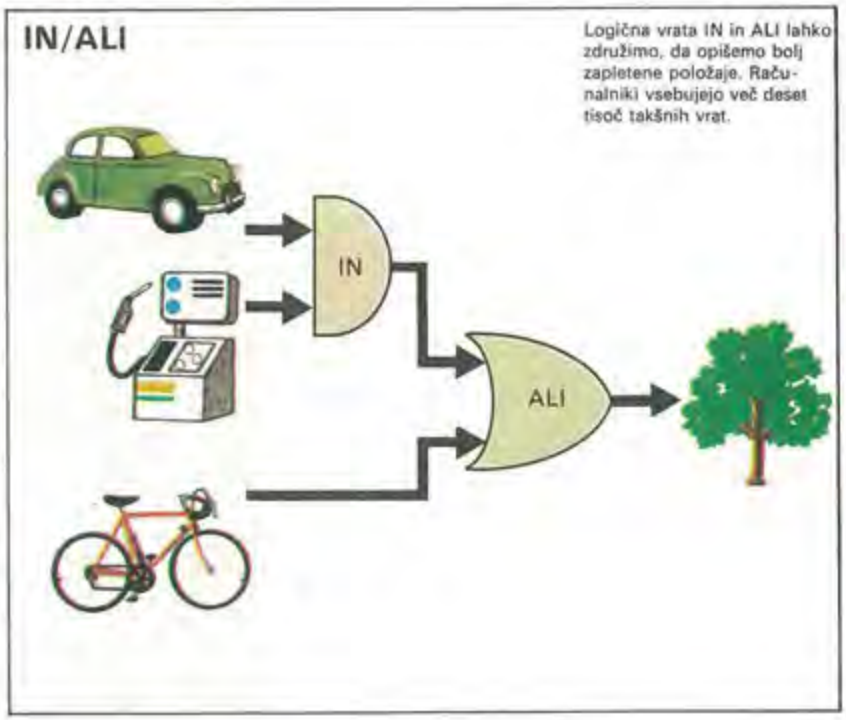
Logična vrata lahko tudi združujemo, da opišemo bolj zapletene položaje, kakor nam to kaže zgled za izlet na deželo na sliki. S kombinacijami vrat IN, ALI in NE lahko izvedemo tudi zelo zapletene odločitve.



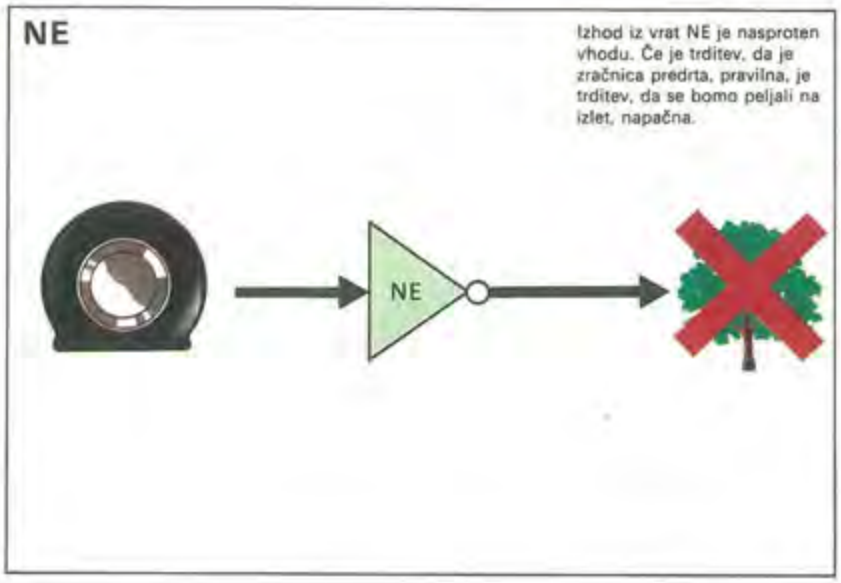
Na izlet na deželo se lahko peljemo, če imamo avto IN bencin.



Na izlet se lahko peljemo, če imamo avto ALI kolo.



Logična vrata IN in ALI lahko združimo, da opišemo bolj zapletene položaje. Računalniki vsebujejo več deset tisoč takšnih vrat.



Poskusite izdelati logične odločitve z vrati še za druge probleme. Kateri pogoji morajo biti izpolnjeni, da bomo lahko imeli piknik v naravi? Da bo izhodna trditev pravilna, moramo imeti denar ALI čekovno knjižico (da kupimo hrano in pijačo) IN prost dan IN lepo vreme IN rešetko IN oglje.

Pravilnostne tabele

Simboli, ki smo jih uporabili v ilustracijah, se uporabljajo tudi v diagramih za računalniška vezja. Odvisnost logičnih izhodov od vhodov lahko prikažemo v pravilnostnih tabelah. Za zgled vzemimo logična vrata IN. S črko a označimo vhodni pogoj »imamo avto«, s črko b označimo pogoj »imamo bencin«, izhodni pogoj »izlet na deželo« pa označimo s črko i. Črko P uporabimo za pravilno, črko N za napačno. Pravilnostna tabela nam kaže izhode za vse mogoče vhodne kombinacije.

Kot že vemo, uporabljajo računalniki le dvojiška znaka 0 in 1. 0 uporabljajo za napačno trditev, 1 za pravilno. Računalnik predstavi znak 1 s pozitivno napetostjo (npr. 5 voltov), 0 pa z napetostjo nič. Vezje IN je narejeno iz nekaj transistorjev tako, da bo na izhodu pozitivna napetost le, če bo pozitivna napetost na obeh vhodih. V vsakem drugem primeru bo izhodna napetost nič.

Elektronsko vezje ALI da na izhodu pozitivno napetost, če je vsaj en vhod na pozitivni napetosti. Oba vhoda morata biti torej nič, da dobimo nič na izhodu. Vezje NE pa da na izhodu nasprotno napetost, kakor je na vhodu. Če je na vhodu pozitivna napetost, je na izhodu nič, in narobe, če je na vhodu nič, je na izhodu pozitivna napetost.

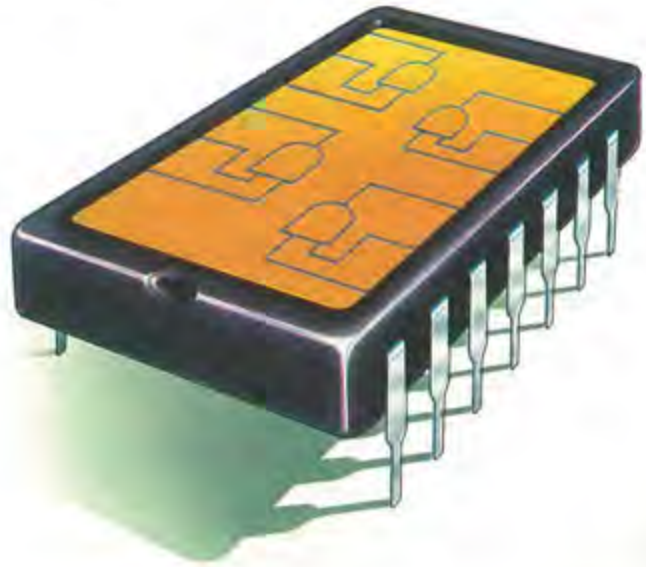
avto	N	●	N	●	(a)
bencin	N	N	●	●	(b)
izlet	N	N	N	●	(i)

Pravilnostna tabela za dvovhodna vrata IN.

avto	0	●	0	●	(a)
bencin	0	0	●	●	(b)
izlet	0	0	0	●	(i)

Ista pravilnostna tabela z uporabo znaka 0 za napačno in 1 za pravilno.

Integrirano vezje 7408
Velika integrirana vezja vsebujejo več tisoč logičnih vrat IN, ALI in NE. Integrirano vezje 7408 pa je majhno in vsebuje samo 4 elektronska vezja IN. Na sliki so ponazorjena z logičnimi simboli in ne z dejanskim vezjem.



Logična vrata in seštevalniki

Dvojiški (binarni) števili 1 in 0 lahko seštevamo z uporabo preprostih logičnih vrat, kakor so vrata IN, vrata ALI in vrata NE

Namizni seštevalni stroj
Še pred pojavom elektronskih kalkulatorjev so v prodajalnah in uradih uporabljali mehanske seštevalne stroje. Če izvzamemo nekaj izboljšav, jih poznamo nespremenjene skoraj 300 let. Njihovo delovanje je zasnovano na prenosnem razmerju zobatih koles. Prvi seštevalni stroj je izdelal Pascal za uporabo v očetovem uradu. Množenje in deljenje je tovrstnemu stroju dodal Leibniz. Tako je bil mehanski kalkulator pripravljen za množično uporabo.

Logična vrata IN, ALI in NE lahko izdelamo s sorazmerno preprostimi transistorskimi vezji. Presenča nas, da so ista logična vrata tudi sestavni deli aritmetične enote znotraj računalnika. Povedano z računalniško logiko sta vhoda vrat nič voltov, kar predstavlja »ničlo«, ali pa imata pozitivno napetost, kar predstavlja »enko«. Simbolično to zapišemo kot »0« in kot »1«. Kadar uporabljamo logična vrata za aritmetične operacije, uporabljamo ničle in enke, iste ničle in enke, vendar tokrat dejansko predstavljajo enke in ničle, ki se seštevajo, in v času, ko jih vpisujemo, se že seštevajo.

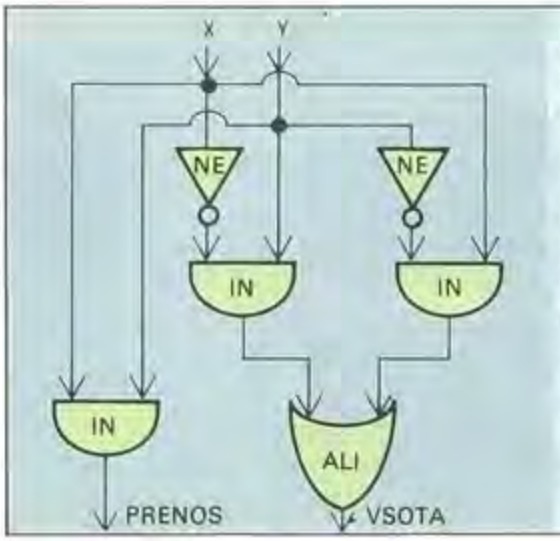
Če želimo sešteti dve binarni števili, potrebujemo samo dvovhodno seštevalno vezje, in možne bodo samo štiri kombinacije: 0 + 0, 0 + 1, 1 + 0 in 1 + 1. Spomnimo se, da smo se v binarni aritmetiki naučili, da je 0 in 0 enako 0 (tako kakor v desetiški aritmetiki). Vemo tudi, da je 0 in 1 (ali 1 in 0) enako 1. Razlika dvojiške aritmetike od desetiške je ta, da je tu 1 in 1 enako 0, dobimo pa še prenos 1. Poglejmo te štiri primere na spodnji tabeli:

x		y		z
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

Če uporabimo za seštevanje vrata ALI, dobimo izid 0, če sta oba vhoda 0 in dobimo izid 1, če je vsaj eden od vhodov 1 (0 + 1 ali 1 + 0).

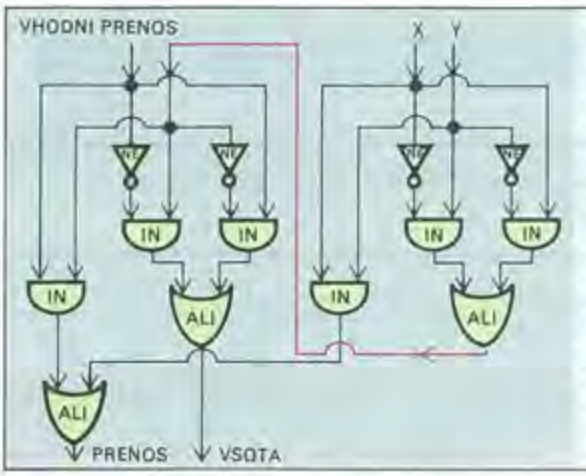
Tako je videti, da bi preprosta vrata ALI lahko lepo uporabili za dvojiško seštevanje dveh številk. Toda počakajmo. Če sta oba vhoda vrat ALI 1, je tudi izid 1, to pa bi bil napačen odgovor v dvojiški aritmetiki. Pravilno bi bilo, da bi bil izid 0 in prenos 1. Enostavna vrata ALI dajo pravilen rezultat za 3 od 4 mogočih vhodnih kombinacij, to pa nam seveda ne zadošča.

Potrebujemo vezje, ki bo imelo izid 0, če bosta oba vhoda 0, in bo imelo izid 1, če bo eden od vhodov 0, drugi pa 1; izid pa naj bo tudi 0, če bosta oba vhoda 1 (tako kakor kaže gornja pravilnostna tabela). Narediti takšno vezje ni tako težko, kakor je to videti na prvi pogled. Uresničitev le-tega je prikazana na spodnji sliki. Sestavljajo ga tri vrata IN, dve vrati NE in vrata



Polovični seštevalnik

To je elektronsko vezje, ki sešteva po dve dvojiški številki. Za to opravilo uporablja logična vrata. Imenujemo ga polovični seštevalnik (half adder), ker ne more računati s prenosom, ki ga pogosto dobimo, če seštejemo dve dvojiški številki. Poskusimo sešteti 1 in 1. Spomnimo se, da vrata NE spremene 1 v 0 in 0 v 1. Oba vhoda na vratih IN morata biti 1, da je na izhodu tudi 1. Pri vratih ALI pa mora biti vsaj eden od vhodov 1, da je tudi izhod 1. Na izhodu dobimo 0 le tedaj, ko sta oba vhoda 0.



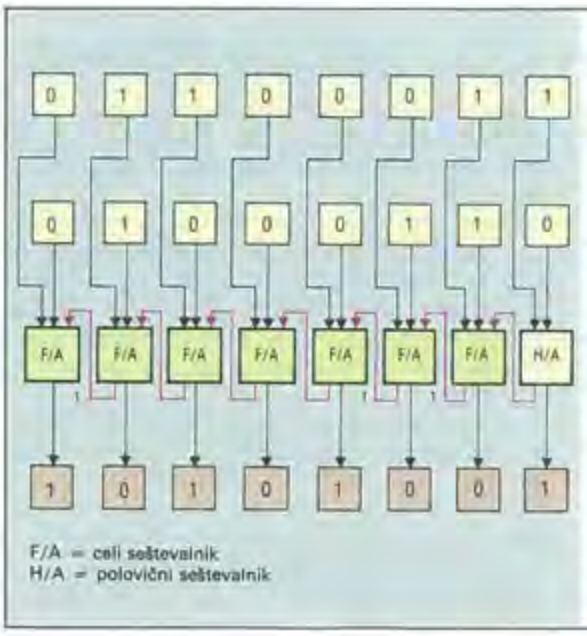
Celi seštevalnik

Celi seštevalnik (full adder) je vezje, ki sešteva številke v dvojiškem sestavu. Sestavljen je iz dveh polovičnih seštevalnikov ter vrat ALI. Računa lahko s po treme številkami — dve številki in morebitni prenos. Seštevanje se izvrši v dveh stopnjah. Najprej se seštejeta dve številki, nato se prišteje še prenos. Te dve stopnji sta omogočeni s povezavo vrat, kjer sta dva polovična seštevalnika združena z vrati ALI.

ALI. Dvoje IN vrat ima po en vhod, priključen direktno na X in Y, po en vhod pa je priključen na X in Y preko NE vrat. Izhoda teh dveh IN vrat vodimo na vhoda ALI vrat. Na izhodu vrat ALI dobimo 1 le takrat, če je X = 1 in Y = 0, ali pa če je Y = 1 in X = 0. Če sta X in Y enaka 0 oziroma 1, pa je izhod vrat ALI tudi 0.

Sedaj že skoraj imamo seštevalno vezje. Moti nas edino še to, da v primeru 1 na obeh vhodih dobimo »vsoto« 0, kar je pravilno, manjka pa nam signal za prenos. Dodati moramo še ena vrata IN, ki jih povežemo vzporedno na oba vhoda. Sedaj bomo dobili signal prenosa samo takrat, če bosta oba vhoda pravilna. Pravilnostna tabela za vezje na sliki, ki prikazuje polovični seštevalnik (half adder), je naslednja:

X (vhod ena)	Y (vhod dva)	C (izhod prenosa)	S (izhod vsote)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



8-bitni seštevalnik

Ko seštevamo velika števila, opravimo seštevanje za vsako številko posebej. Prav tako sešteva tudi računalnik. Poglejmo zgled seštevanja dveh 8-mestnih dvojiških števil. Gornja vrstica vsebuje prvo in spodnja drugo število. Začnimo seštevati na desni strani. Tu ni potrebno dodati nobenega prenosa, zato uporabimo polovični seštevalnik. V naslednjih stolpcih pa že dobivamo prenos, zato moramo uporabiti cele seštevalnike. Poskusimo izračunati vsoto še sami tako, kakor to dela računalnik, in preverimo rezultat v spodnji vrsti.

To vezje imenujemo »polovični seštevalnik« zato, ker je pravzaprav rezultat seštevanja polovičen. S tem vezjem lahko storimo le to, da seštejemo dve enomestni binarni števili in ničesar drugega. Običajno pa želimo sešteti dva zloga podatkov, vsak zlog pa vsebuje 8 bitov. Za seštevalnik, ki bi sešteval na skrajni desni strani stolpca števil, ne potrebujemo nič boljšega kakor polovični seštevalnik. Vsi preostali seštevalniki proti levi strani pa morajo imeti možnost treh vhodov — dveh številk iz »svojega« stolpca in priključek za prenos iz prejšnjega seštevalnika.
Poglejmo si sedaj naslednji primer:

$$\begin{array}{r} 011 \\ +\,111 \\ \hline 1010 \end{array}$$

Ko seštevamo enake stolpce, rečemo: 1 in 1 je 0 s prenosom 1, in zapišemo 0 pod enke stolpca. Ko seštejemo drugi stolpec, rečemo: 1 in 1 je 0 s prenosom 1. Prištejemo še prenos iz prejšnjega stolpca, ki je bil 1. Pod drugi stolpec zapišemo 1, novi prenos 1 pa prištejemo naslednjemu stolpcu. 0 in 1 v tretjem stolpcu da 1. Tej vrednosti prištejemo še prenos 1 iz prejšnjega stolpca, tako da dobimo 0 in prenos 1. 0 zapišemo pod tretji stolpec in ker v četrtem stolpcu nimamo več številk, zapišemo prenos na to mesto. Z drugimi besedami, pravilnostna tabela za cel seštevalnik (full adder) kaže, da lahko pravilno sešteva dve številki in prenos iz prejšnjega seštevalnika.

X	Y	C vhod	C izhod	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Cel seštevalnik lahko naredimo z uporabo dveh polovičnih seštevalnikov in z dodatnimi vrati ALI. Prenos (carry out) iz vsakega celega seštevalnika je povezan direktno na vhodni prenos (carry in) seštevalnika na levi strani. Tako sestavljamo cele seštevalnike, kolikor jih pač potrebujemo.

Moderni računalniki lahko poleg seštevanja opravljajo tudi druge aritmetične operacije z zelo velikimi števili. To opravijo z množico seštevalnih vezij, kakršna so opisana zgoraj. V večini primerov so seštevalna vezja vdelana v vezju, ki ga imenujemo CPU (centralna procesna enota). Pred leti, ko še ni bilo vezij z visoko stopnjo integracije, ki je omogočila izdelavo mikroprocesorja, so bila preprosta vezja, ki so vsebovala le po nekaj logičnih vrat. Običajno so jih imenovali vezja TTL (TTL je okrajšava za transistor-transistorska logika, kajti večino preklopne logike določajo transistorji, ki so medsebojno povezani). Znotraj tipičnega CPU je na silicijevi ploščici vdelana majhna površina RAM in ROM, preklopna logika ter del, ki se imenuje ALU ali aritmetična logična enota. ALU, ki je del CPU, vsebuje vsa logična vrata in seštevalnike, ki jih računalnik potrebuje za računanje in druge logične odločitve.

Popolno pomnjenje

Kako je organiziran pomnilnik računalnika in zakaj računalnik nikoli nič ne pozabi

Spomin je skladišče človekovega mišljenja. Je prostor, kamor se nalagajo drobci izkušenj, ki jih lahko kasneje koristno uporabimo. Za računalnik pomeni »spomin« nekaj podobnega, le da je računalnikov spomin veliko bolj omejen kakor človeški.

Slab spomin povzroča človeku neprijetnosti, včasih ga lahko spravi tudi v zadrego. Za računalnik pa pomeni slab pomnilnik katastrofo. Če ne bi imel pomnilnika, ne bi mogel ničesar narediti, ker spravlja v svoj pomnilnik vse programe, ki ga upravljajo, in podatke za izvajanje.

Beseda »spomin« je povezana z dvema dejavnostma: spravljanje informacij v spomin in njihovo ponovno branje. Spravljanje informacij, ki jih ne bi mogli več prebrati ali se jih spomniti, bi ne bilo posebno koristno, prav tako pa bi bilo nesmiselno, če bi skušali priklicati podatke, ki niso bili spravljeni.

Človeški in računalniški spomin sta si podobna tudi v tem, da sta dveh vrst: kratkoročni in dolgoročni. Moški, ki prečka cesto, si bo seveda zapomnil avto, ki mu prihaja nasproti, in bo počakal, da bo odpeljal mimo, preden bo prečkal cesto. Ko pa bo na drugi strani ceste, bo na ta avto pozabil in se niti ne bo spomnil, kakšen je bil. To je primer kratkoročnega spomina.

Če pa bi v tem avtomobilu zagledal dva maskirana moža na zadnjih sedežih in svojo ženo na prednjem, bi si gotovo zapomnil tip in barvo avtomobila, mogoče tudi registrsko številko. To bi bil dolgoročni spomin.

Tudi računalnik ima kratkoročni in dolgoročni pomnilnik. Dolgoročni vsebuje programe in podatke, ki jih uporabnik želi ohraniti tudi potem, ko računalnik izklopi. Takšen tip pomnilnika so magnetni posnetki na površini kasetnega traku, na disketi ali disku, in seveda tudi podatki, spravljeni v bralnih pomnilnikih (ROM).

Kratkoročni spomin, katerega vsebina se izgubi, ko ugasimo računalnik, pa je delovni pomnilnik (RAM) v računalniku. Vsebina pomnilnika RAM se izgubi, tudi če samo za delček sekunde prekinemo napajalno napetost.

Razlike med človeškim in računalniškim spominom pa so kljub tem podrobnostim velike. Da lahko računalnik sploh deluje, moramo programe in podatke prenesti iz dolgoročnega v delovni pomnilnik, tako da bo imel računalnik do njih ves čas dostop. Popolnoma različen je tudi način, kako se podatki spravljajo in berejo.

Še vedno ni povsem jasno, kako delujejo človeški možgani. Dogodek, ki si ga zapomnimo, se najbrž ne spravi v kakem majhnem, natančno določenem delu možgan. Ni nam treba vedeti, v kateri del možgan smo spravili neko informacijo, ko se je skušamo spomniti, prav tako nam te informacije, ko smo premišljevanje končali, ni treba vrniti v ta del.

Organiziran nered

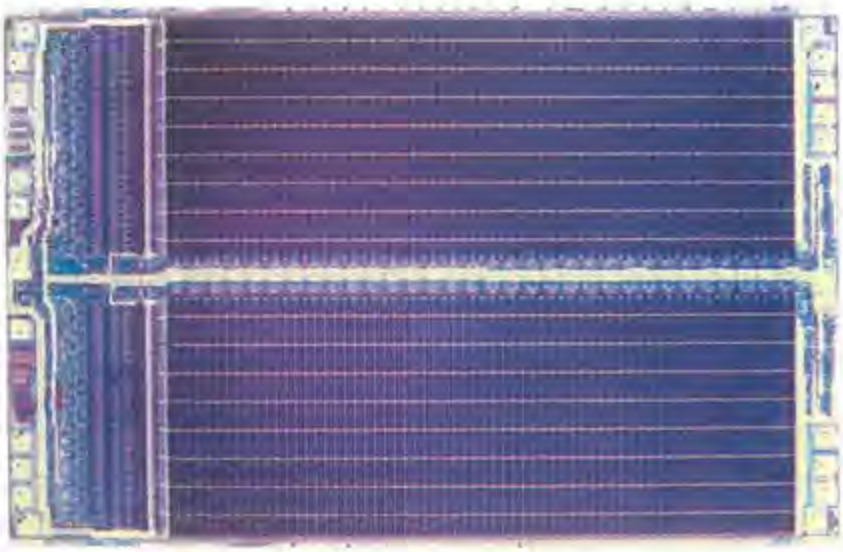
Za računalnik je bistvena lokacija vsake informacije v pomnilniku. Računalnik mora znati poiskati določen zlog informacije ne glede na to, ali je to program ali pa podatki, ki jih potrebuje med izvajanjem programa. Računalnik mora torej ves čas vedeti, kam je spravil kako informacijo.

Človeški spomin je kot škatla, natrpana z informacijami, vendar ne organizirana. Deli informacije se nalagajo naključno, se mešajo med seboj, menjujejo svoje mesto, ko prihajajo v spomin nove slike in izkušnje. Na neki sedaj še neznan način se znajo možgani v tem znajti in navadno tudi najdejo podatke, ki jih iščejo.

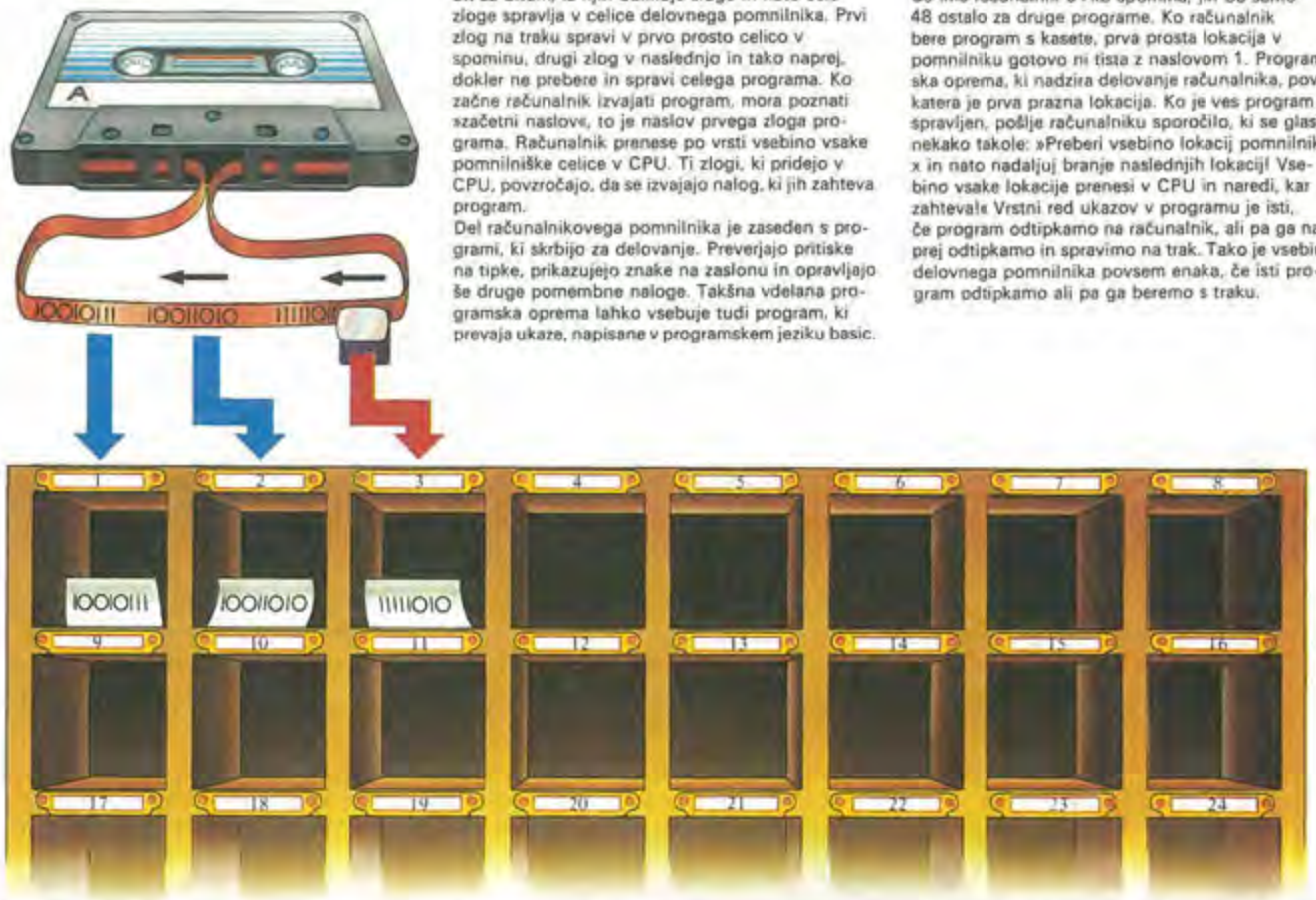
Računalnikov spomin pa bolj spominja na velikanski predalnik s številnimi predalčki. Vse je zelo urejeno, vsak predalček ima svojo številko (imenovano tudi naslov) in vsebuje nič več in nič manj ko en zlog. Računalnik najde določen zlog zato, ker pozna številko (naslov) predalčka, v katerem je spravljen.

Računalnik ni inteligenten in ne more sam organizirati svojega spomina. Zlog spravi na pravi naslov le

Bralni pomnilnik
Integrirano vezje RAM (na sliki spodaj) je eden najpomembnejših dosežkov računalniške tehnologije v zadnjem času. Integrirano vezje RAM predstavlja delovni pomnilnik, ki je del elektronskega pomnilnika računalnika. V to skupino sodi še bralni pomnilnik (ROM). Kasetni trakovi in diskete pa so zgledi za elektromagnetni spomin.
Integrirano vezje RAM je izdelano iz silicija. Z uporabo fotografskih procesov, jedkanja in drugih tehničnih postopkov nastane na majhni silicijevi tabletki več tisoč transistorjev. Pomnjenje vsakega bita zahteva najmanj en transistor v vezju pomnilnih celic.
Čas, potreben za vpis enega bita v eno od 16384 pomnilnih celic, je približno 200 nanosekund (petmilijoninka sekunde).



Pomnilnik



Programi, posneti na trak, so spravljeni serijsko. To pomeni, da so biti v zlogu posneti drug za drugim. Ko vključimo kasetofon, računalnik bere bit za bitom, iz njih oblikuje zloge in nato cele zloge spravlja v celice delovnega pomnilnika. Prvi zlog na traku spravi v prvo prosto celico v spominu, drugi zlog v naslednjo in tako naprej, dokler ne prebere in spravi celega programa. Ko začne računalnik izvajati program, mora poznati »začetni naslov«, to je naslov prvega zloga programa. Računalnik prenese po vrsti vsebino vsake pomnilniške celice v CPU. Ti zlogi, ki pridejo v CPU, povzročajo, da se izvajajo nalogi, ki jih zahteva program.

Del računalnikovega pomnilnika je zaseden s programi, ki skrbijo za delovanje. Preverjajo pritiske na tipke, prikazujejo znake na zaslonu in opravljajo še druge pomembne naloge. Takšna vdelana programska oprema lahko vsebuje tudi program, ki prevaja ukaze, napisane v programskem jeziku basic.

Ti notranji programi zasedajo del pomnilnika in puščajo manj prostora uporabnikovim programom. Nekatere verzije basica zasedajo kar 16 kb spomina. Če ima računalnik 64 kb spomina, jih bo samo 48 ostalo za druge programe. Ko računalnik bere program s kasete, prva prosta lokacija v pomnilniku gotovo ni tista z naslovom 1. Programska oprema, ki nadzira delovanje računalnika, pove, katera je prva prazna lokacija. Ko je ves program spravljen, pošlje računalniku sporočilo, ki se glasi nekako takole: »Preberi vsebino lokacij pomnilnika x in nato nadaljuj branje naslednjih lokacij! Vsebino vsake lokacije prenesi v CPU in naredi, kar zahteva!« Vrstni red ukazov v programu je isti, če program odtipkamo na računalnik, ali pa ga najprej odtipkamo in spravimo na trak. Tako je vsebina delovnega pomnilnika povsem enaka, če isti program odtipkamo ali pa ga beremo s traku.

zato, ker mu je pri tem nekaj pomagalo. Kako se to godi v tipičnem hišnem računalniku?

Ko vključimo svoj hišni računalnik, se običajno na zaslonu pokaže sporočilo, ki nam pove, da naprava deluje in da lahko začnemo pisati program. To sporočilo je skupaj s programi, ki nam omogočajo, da začnemo delati z računalnikom, spravljeno v dolgoročnem spominu v računalniku. Običajno je to bralni pomnilnik.

Ta del računalnikovega pomnilnika vsebuje programe, ki preverjajo, ali smo pritisnili kako tipko, ki »pišejo« na zaslon in opravljajo druga dela, ki omogočajo delovanje računalnika. Vsebuje tudi poseben program, ki prevaja ukaze, običajno napisane v basicu, v veliko preprostejši strojni jezik iz samih ničel in enk, ki ga računalnik razume.

Marsikateri računalnik nam ob vklopu pošlje sporočilo v obliki »x bytes free«, kjer je x nenavadno število, na primer 15797. S tem sporočilom nam pove število predalčkov v računalniškem pomnilniku, ki so prazni in je vanje mogoče spravljati zloge. Ko pritiskamo na tipke, začnemo te predalčke polniti. Pri tem je zelo pomembno, v kakšnem vrstnem redu se polnijo.

Ko pritisnemo tipko na tipkovnici, pošljemo en zlog (ki predstavlja pritisnjen znak) v pomnilnik. Ko na primer pritisnemo tipko »k«, se spravi dvojiška koda te črke v določen predalček v pomnilniku. Toda v kateri predalček gre ta zlog? Gre na prvo prazno mesto v delovnem pomnilniku. Če si predstavljamo prazen predalnik, gre zlog za »k« v predalček v levem gornjem kotu. Ko pritisnemo naslednjo tipko, na primer »e«, gre nov zlog, ki je binarna koda črke »e«, v drug prazen predalček, desno od tistega s kodo za »k«. Tretji znak gre v sosednji prazni predalček in tako naprej, dokler ne zapolnimo prve vrste.

Računalnik ima notranji števec, ki šteje zasedene naslove. Števec začne šteti pri naslovu, ki mu ga ob vklopu sporoči program, ki skrbi za delovanje računalnika.

Ob vsakem vpisu v pomnilnik se njegova vsebina poveča za enega. Tako s pomočjo števca računalnik vedno ve, kateri je naslednji prazni naslov.

Shranjevanje podatkov

Računalnik lahko shrani na tisoče zlogov informacij v svojem spominu in si zapomni, kje je vsaka izmed njih shranjena

Ko opisujemo računalniški pomnilnik, imamo v mislih trajno in začasno shranjevanje podatkov. Trajno shranjeni podatki ne izgubijo informacije tudi po zelo dolgem času, četudi smo vmes izključili napajalno napetost. V to kategorijo že spadajo magnetni trakovi in diskete.

Računalnik pa prav tako potrebuje hiter kratkoročen pomnilnik za začasno shranitev programov in podatkov.

Drug način za določitev računalniškega pomnilnika je delitev na notranji in zunanji pomnilnik. Notranji pomnilnik je znotraj računalnika in je običajno popolnoma elektronski, medtem ko je zunanji pomnilnik običajno zunaj računalnika. Zunanje pomnilniške enote so delno mehanske in obsegajo mehanizme kot kasete, diskete (floppy disk) ali celo luknjalnike in bralnike papirnih kartic.

Notranji elektronski pomnilnik običajno imenujemo delovni pomnilnik, medtem ko nam zunanje pomnilniške enote rabijo za shranitev programov in drugih podatkov. Sodobni računalniki imajo kot notranje pomnilniške enote delovne pomnilnike (RAM) in bralne pomnilnike (ROM).

EPROM
Problem pri naročanju bralnih pomnilnikov pri proizvajalcu je ta, da imajo ti vsebino vdelano že ob proizvodnji in je potem ni več mogoče spreminjati. EPROM (bralni pomnilnik z možnostjo brisanja stare vsebine in vpisa nove) pa je dosti prožnejši. Če nam njegova vsebina ne ustreza, ga lahko ponovno programiramo. Seveda moramo prej izbrisati prejšnjo vsebino. EPROM ima vdelano okence iz posebnega stekla, skozi katero lahko prihajajo ultravijolični žarki; ti povzročajo praznjenje majhnih kondenzatorjev, ki hranijo informacijo za vsak bit posebej. Ko tako zbrišemo staro vsebino, lahko v EPROM s posebno napravo vpišemo nove podatke.

Integrirana vezja za bralne in delovne pomnilnike so izdelana na silicijevi rezini in vdelana v pravokotnih plastičnih ohišjih s po dvema vrstama vzporednih priključkov. Naslavljanje in izbiranje posameznih bralnih oziroma delovnih pomnilnikov je izvedeno s centralno procesno enot (CPU) in je za obe vezji precej podobno.

Bralni pomnilnik nam rabi za shranjevanje trajnih programov v računalniku. Na vsaki lokaciji pomnilnika imamo trajen vzorec ničel in enk, ki je bil vdelan že ob izdelavi in ga pozneje ne moremo več spremeniti. Računalnik lahko jemlje podatke iz bralnega pomnilnika, ničesar pa ne more vanj vpisati.

Izraz bralni pomnilnik nam pove, da računalnik prebere vsebino, shranjeno na določeni lokaciji. Da pa je ta določena lokacija v pomnilniku dostopna računalniku, jo mora ta prej izbrati in priključiti na vodilo (BUS). Bralnih pomnilnikov imamo več vrst. Nekateri so takšni, da lahko shranjeni notranji program zbrišemo in ponovno programiramo drugega. Tipični bralni pomnilnik, ki ga izdeluje Intel, ima oznako 2364. To vezje ima 65536 bitov bralnega pomnilnika. Njegova organizacija je 8 kb po 8 bitov. To pomeni, da je vseh 64 k bitov zloženih v 8-bitne zloge. Vsak zlog 8 bitov je postavljen na svoji lokaciji. Ko naslovimo določeno lokacijo, lahko preberemo vsebino celotnega zloga. V matematiki pomeni $1\,k = 2^{10}$ (dve na deseto) ali 1024, tako da je $64\,k = 54 \times 1024$ ali 65536.

Računalnik mora biti zmožen izbrati katerokoli lokacijo, teh pa je 8 k ali 8192. Če pogledamo v proizvajalčev katalog podatke za to vezje, vidimo, da ima 28 priključnih nožic; en priključek je uporabljen za napajanje +5 V in eden za maso. Torej nam ostaja še 26 priključkov. Vsak zlog vsebuje 8 bitov. Kadar beremo informacijo iz bralnega pomnilnika, je vseh osem bitov povezanih s centralno procesno enoto. Torej imamo 8 povezav za osem bitov v zlogu, ki jih prebere centralna procesna enota. Te povezave imenujemo »podatkovno vodilo« (data bus). Osem priključkov tega integriranega vezja je namenjenih za priključitev na podatkovno vodilo — vsak bit ima torej svojo povezavo.

Ostane nam še 18 priključkov. Eden od njih je nepotreben in ni povezan. Imamo ga zato, ker je lažje izdelovati integrirana vezja s sodim številom nožic kakor pa z lihim. Štirje priključki so uporabljeni za izbiranje različnih načinov delovanja bralnega pomnilnika. Ti priključki so naslednji: z enim omogočimo priključitev bralnega pomnilnika na podatkovno vodilo, drugi nam omogoča delovanje tega vezja (bralnega pomnilnika), z dvema pa izbiramo posamezna integrirana vezja bralnega pomnilnika v računalniku. Ti priključki dobivajo signal od računalnika, in sicer samo takrat, ko izbere določeno vezje.

Preostalih 13 priključkov je naslovnih (»adresnih«). Vsak od teh je povezan na naslovno vodilo, ki prenaša naslov za vsak zlog, ki ga želimo dobiti iz bralnega pomnilnika. Povejmo še, da je naslov kodiran v dvojiškem sestavu. S trinajstmestnim dvojiškim številom lahko dobimo 2^{13} ali 8192 različnih kombinacij

Trajno shranjevanje podatkov
Bralni pomnilnik je podoben knjigi po tem, da vsebuje trajno shranjene podatke. Vsebine bralnega pomnilnika ne moremo spreminjati, kakor tudi ne moremo več spreminjati besed, ki so natisnjene v knjigi.

Začasno shranjevanje podatkov
Delovni pomnilnik je podoben kartoteki, v katerega lahko vstavimo podatke, pa jih lahko tudi spreminjamo. Shranjeni so začasno — ob izključitvi računalnika se zbriše tudi vsebina delovnega pomnilnika.

enk in ničel, tako da nam to zadošča za izbiranje kateregakoli od 8192 zlogov, shranjenih v ROM.

Delovni pomnilnik uporablja računalnik za začasno shranjevanje programov in podatkov med svojim delovanjem. Prav tako v delovni pomnilnik začasno shrani rezultate in druge podatke. Notranja zgradba delovnega pomnilnika je dosti bolj zapletena od tiste v bralnem pomnilniku, ker moramo pri delovnem pomnilniku imeti možnost, da z vpisovanjem spremenimo vsebino kateregakoli zloga. Tipično integrirano vezje, ki se uporablja kot bralni pomnilnik, je Intelov 2114. Vsako integrirano vezje 2114 ima 4096-bitni pomnilnik, ki je organiziran po 4-krat 1024 bitov. To pomeni, da imamo na vsaki naslovni lokaciji podatek, sestavljen iz štirih bitov. Da dobimo 1 kb podatkov, potrebujemo torej dve takšni integrirani vezji. Vsako integrirano vezje 2114 ima samo 18 priključkov. Dva od njih sta uporabljena za napajalno napetost, štirje pa za vhodno-izhodne podatkovne linije. Eden od priključkov je uporabljen za signal, ki pove našemu vezju, da je izbrano (chip select — CS), z drugimi besedami, da mora biti pripravljeno za vpisovanje ali za branje podatkov. En priključek imamo za to, da vezje ve, ali se bodo podatki brali ali vpisovali. Preostalih deset priključkov je uporabljenih za naslovno vodilo. Deset naslovnih linij nam predstavlja 2^{10} ali 1024 lokacij. Če ima računalnik 64 kb delovnega pomnilnika in če so uporabljena integrirana vezja 2114, jih potrebujemo 128, ker za vsak zlog potrebujemo po dve vezji. V sedanjem času pa že uporabljamo zmogljivejša vezja, ki imajo možnost shranjevanja večjih količin informacij. Kot primer takega vezja naj omenim integrirano vezje 4164, s katerim lahko naredimo 64 kb delovnega pomnilnika s samo 8 vezji.

Vezja za delovne in bralne pomnilnike postajajo vedno cenejša in imajo vedno večjo zmožnost shranjevanja večje količine podatkov. Izdelujejo že vezja, ki lahko shranijo 128 kb podatkov na enem samem integriranem vezju. Razvoj mikroelektronike pa prinaša vedno manjša in zmogljivejša integrirana vezja. Mikroelektronska vezja postajajo že tako majhna, da je optična fotolitografija komaj še primerna in zato že uvajajo nove tehnike, kjer pri proizvodnji integriranih vezij uporabljajo jedkanja z elektronskimi ali z X žarki. Za delovne pomnilnike uporabljamo dve vrsti vezij: statična in dinamična. Vsako izmed njih ima svoje prednosti in slabosti, vendar so se v sedanjem času bolj uveljavili dinamični delovni pomnilniki.

Oba tipa pa izgubita vsebino, če jima izključimo napajalno napetost. Dinamični pomnilnik potrebuje poseben signal, ki se pojavlja za osvežitev vpisane vsebine vsakih nekaj milisekund. Ta osvežitev podatkov mora biti opravljena za vsak bit podatkov, ne glede na to, kaj počne centralna procesna enota. V ta namen so izdelali posebna vezja, ki dajejo potrebne impulze v ustreznih časovnih zaporedjih.

Dinamični delovni pomnilniki imajo dve znatni prednosti pred statičnimi. Dinamična vezja potrebujejo za shranitev enega bita samo en transistor, medtem ko statična potrebujejo tri. Pri enaki površini integriranega vezja nam to omogoča večjo gostoto shranjenih podatkov. Večina dinamičnih pomnilnikov ima po 16 priključkov. Druga prednost dinamičnega pomnilnika je manjša poraba električne energije. Zato se ta vezja manj razgrevajo in potrebujejo manjša in cenejša napajalna vezja.

Prednost statičnega delovnega pomnilnika je preprosto načrtovanje celotnega računalnika. Ko so podatki vpisani v statični delovni pomnilnik, jih ni treba več obnavljati. Slabost pa je, ker so za vsak bit pomnilnika potrebni trije transistorji, in to pomeni manj zmogljiva vezja pri enaki velikosti. Zaradi večje porabe energije se ta vezja tudi bolj segrevajo, to pa pomeni, da moramo pri načrtovanju računalnika misliti na odvajanje odvečne toplote, zato je tak računalnik dosti dražji.

Razumem sporočilo

Ko pritisnemo na tipko, se aktivira programska oprema, ki dekodira naš ukaz, preiskuje pomnilnik in ves čas pregleduje tipkovnico, da bi zaznala naš poslednji ukaz

Računalnik je naprava, narejena iz raznih kovin, umetnih snovi in silicija, ki pa je brez programa v svojem pomnilniku nezmožna napraviti karkoli koristnega. Je kot gramofon brez gramofonske plošče. Proces, med katerim pripravimo računalnik do tega, da opravi želeno nalogo, imenujemo programiranje. Celo začetnik, ki še ni dosti programiral, ve, da ločimo pri tem dve fazi. Najprej moramo problem, ki ga želimo reševati z računalnikom, prevesti in zapisati v obliko, ki jo računalnik razume. V drugi fazi pa moramo program prenesti v računalnik in sprožiti njegovo izvajanje.

Želimo na primer napisati program, ki bo vodil finančno poslovanje. Najprej moramo popolnoma razumeti problem. Vedeti moramo, kakšen rezultat želimo in kakšne podatke potrebujemo za to.

Vsekakor morajo vhodni podatki zajemati vse, kar je v zvezi z našim finančnim poslovanjem. Če gre za podjetje, moramo upoštevati vse prihodke podjetja, način obračunavanja osebnih dohodkov, vse stroške poslovanja, plačila prispevkov in davščin itd. Če pišemo program, ki bo zasledoval družinski proračun, bo stvar seveda preprostejša.

Pri programu, ki bi ga pisali za podjetje, bi nam moral najbrž pomagati analitik, ki pozna vse probleme v zvezi s finančnim poslovanjem. Za svojo domačo rabo bi to storili sami.

Če bi računalniki razumeli slovenščino ali kak drug naravni jezik, bi nam ta »specifikacija programa«, ki smo jo do sedaj napravili, popolnoma zadoščala. Žal pa to še ni tako. Program moramo zapisati v enem od programskih jezikov, ki ga razume naš računalnik. Veliko začetnikov ima pri tem težave, ker skušajo napisati ves program naenkrat, kakor bi na primer prevajali članek iz slovenščine v angleščino. Izkušeni programerji razdelijo program na več delov in se ukvarjajo z vsakim posebej. Tako bi lahko razdelili naš program za finančno poslovanje na štiri dele, imenovane module: zapisovanje tekočih mesečnih podatkov, izračun novega stanja, sprememba podatkov, ki jih je treba zasledovati vse leto, in izpis stanja ob koncu meseca.

Vsak modul lahko še naprej delimo na manjše strukture. Takšen način programiranja je znan pod imenom »strukturno programiranje«. Vsak takšen delec vsebuje nekaj vrstic programa.

Ko napišemo cel program, ga odtipkamo v računalnik. Vsak dober programer si vsako stopnjo, preko katere se program razvija, zapiše, ker mu to zelo pomaga pri morebitnih kasnejših spremembah. Iz teh zapiskov se lepo vidi razvoj programa od začetnega programa, zapisanega v slovenščini, čez veliko vmesnih stopenj do programa v programskem jeziku, kot je npr. basic.

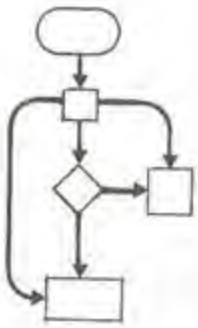
Vse, kar se zgodi od trenutka, ko poženemo izvajanje programa na računalniku z ukazom RUN, je popolnoma pod kontrolo računalnika in zopet obsega več stopenj. Vse to dogajanje pa je pred nami skrito. Mi

Od problema do programa:

Izvor računalniškega programa je spoznanje, da lahko neki problem rešimo z računalnikom. V našem primeru nas zanima, kako bi lahko ohranjali stalno temperaturo v rastlinjaku. Računalniški program je rezultat pretvorbe problema v obliko, ki jo razume računalnik. Pot od problema do programa pa gre seveda preko več stopenj.

Problem se kaže...

Dobimo idejo in jo skiciramo na papir



Problem analiziramo in razvijemo strukturo programa, toda da formuliramo diagram poteka...

Diagram poteka nato prevedemo v enega od programskih jezikov, npr. basic

samo damo vhodne podatke, če program to zahteva, in čakamo na rezultate.

Ker mikroprocesor ne razume programskih jezikov, je prva naloga računalnika, da prevede program v strojni jezik. V hišnih računalnikih opravlja to delo interpreter, ki je stalno spravljen v bralnem pomnilniku.

Interpreter je zelo zapleten kodni program v strojnem jeziku, ki ga opravlja mikroprocesor. Ko odtipkamo ukaz RUN, začne interpreter znak za znakom preiskovati naš program. Vsako besedo primerja s svojim slovarjem. Če odkrije besedo, ki je ne razume (to se lahko zgodi, če smo se zmotili pri tipkanju), se bo ustavil in nam sporočil, da je našel slovnično napako.

Če beseda obstaja v interpreterjevem slovarju (npr. ukaz PRINT), se v delo vključi drug del interpreterja, ki izvršuje ukaze. Določena rutina bo preiskala, kaj sledi v uporabnikovem programu besedi PRINT, da bo interpreter zvedel, kaj mora izpisati, nato pa bo te podatke poiskal v pomnilniku.

Tu nastopi druga stopnja v delovanju računalnika. V pomnilniku je tudi rutina, ki sprejema znake, ki jih je treba izpisati na zaslon. Spravi jih v drug del pomnilnika, rezerviran za pomnjenje slike, ki jo računalnik prikazuje na zaslonu. Ta rutina poskrbi, da se ti znaki spremenijo v signale, ki jih potrebujeta monitor ali televizor, da lahko pokažeta želeno sliko. To pretvarjanje mora potekati stalno, tudi takrat, ko se program ukvarja le z računanjem, ker mora zaslon vedno kaj prikazovati.

Enako je z drugim delom računalnika — tipkovnico. Posebna programska rutina mora pregledati tipkovnico in ugotoviti, ali smo pritisnili kakšno tipko, in če smo jo, katero. Nato mora poslati kodo, ki pripada tej tipki, v poseben del pomnilnika. Ker lahko natipkamo kadarkoli (želimo npr. z ukazom BREAK ustaviti izvajanje programa), mora ta rutina vedno, tudi med izvajanjem programov, pregledovati tipkovnico.

Večina mikroprocesorjev, ki jih imamo v hišnih računalnikih, lahko opravlja le eno delo hkrati. Zato mora učinkovito deliti čas med interpretiranjem uporabnikovega programa in notranjimi funkcijami, kot so pregledovanje tipkovnice in obnavljanje slike na zaslonu. En način za to je metoda prekinitev, pri kateri posebno elektronsko vezje približno 50-krat v sekundi prekine izvajanje programa in »spomni« mikroprocesor, da opravi svoje notranje funkcije. Ko te naloge opravi, nadaljuje izvajanje programa tam, kjer je prenehal.

Tako je moral računalnik potem, ko so natipkali program, izvajati več nalog na različnih stopnjah, preden smo dobili rezultat. Proces izvajanja je torej precej kompleksen, vendar na našo srečo za večino stvari skrbi računalnik sam.

Proizvajalci zdaj razvijajo računalnike, ki bi jih bilo čim lažje upravljati, da bi uporabnike razbremenili rutinskega dela.

Prav tako skušajo razviti nove programske jezike, ki so bližji naravnim. V bližnji prihodnosti bodo prišli na tržišče računalniki, ki bodo razumeli že specifikacije, pisane v naravnih jezikih.

Skrita programska oprema
V vsakem računalniku ves čas deluje hierarhično urejena skrita programska oprema. Nekatere od mnogih njenih nalog so nadziranje in preverjanje tipkovnice, skrb za sliko na zaslonu, nadzorovanje drugih perifernih enot in zasledovanje statusa in vsebine pomnilnika. Vse te funkcije opravlja stalno, tudi med izvajanjem programa. Načelo delovanja skrite programske opreme je enako v zelo izpopolnjenih poslovnih računalnikih (kakor je ta na sliki) in v relativno cenenih hišnih računalnikih.

Program s tipkovnico prenesemo v pomnilnik računalnika

Program v basicu gre skozi več integriranih vezij, ta ga prevedejo v strojno kodo, ki jo razume CPU

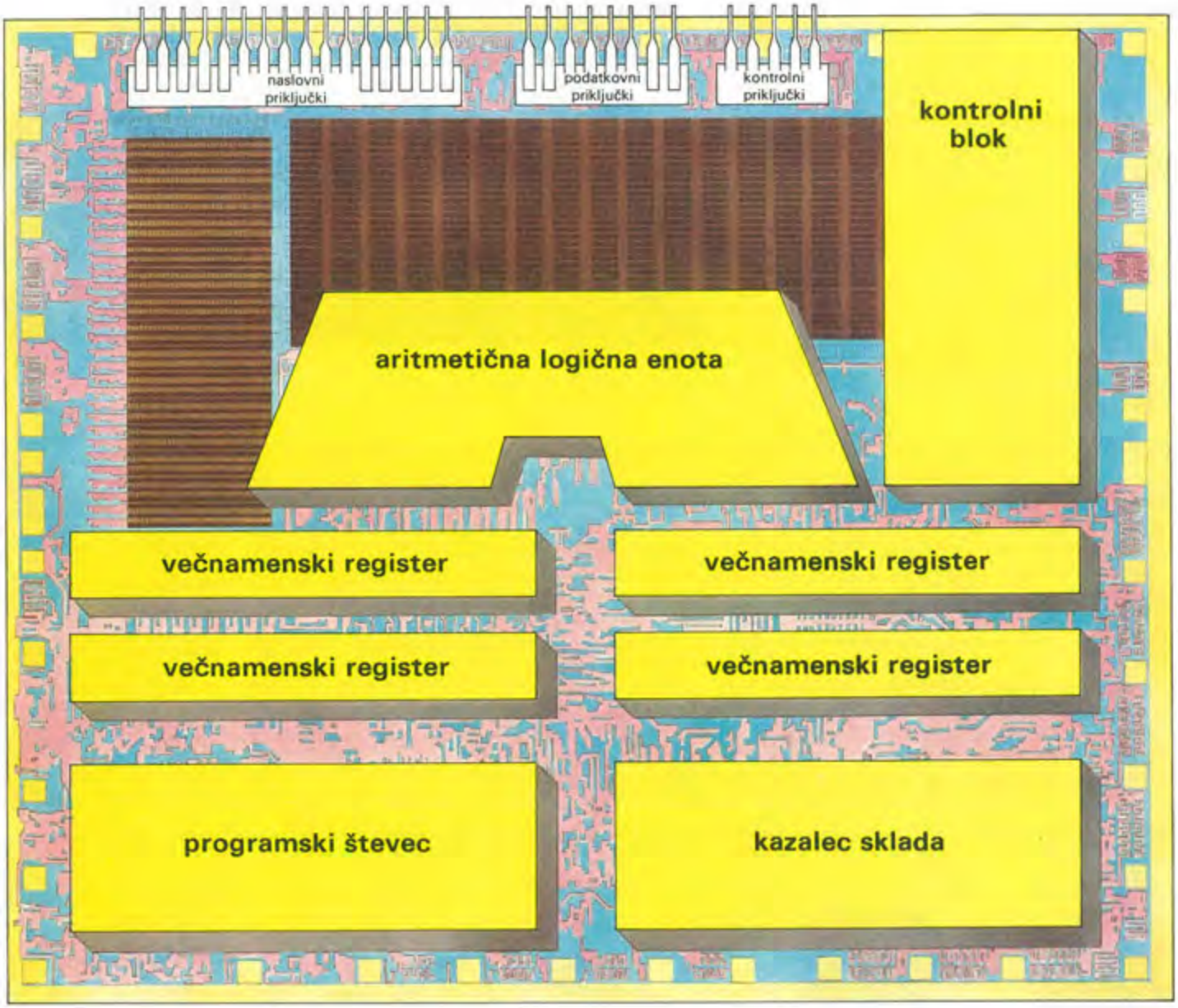
CPU opravi račune. Rezultati se prenesejo v periferno enoto (tiskalnik, monitor, diskovna enota...)

V našem primeru rezultate natiska tiskalnik

Če je bil program pravilen, smo dobili rešitev problema

Živčni center

Vse poti v računalniku se stekajo v centralni procesni enoti



Na sliki vidimo glavne dele CPU: registre, aritmetično logično enoto (ALU) in kontrolni blok. ALU je narejena iz več sto logičnih vrat, ki opravljajo aritmetične funkcije, kot so seštevanje, pomikanje bitov v zlogu levo ali desno, in logične funkcije, kot je ALI. Kontrolni blok dobi dvojiško kodirano navodilo in nato vodi njegovo izvrševanje v CPU. Če na primer navodilo zahteva, naj CPU spravi vsebino akumulatorja v določeno lokacijo pomnilnika, bo kontrolni blok poslal naslov te lokacije na naslovne priključke, s signalom na kontrolnem priključku pa bo obvestil pomnilnik, naj spravi zlog, ki ga nato pošlje po podatkovnem vodilu.

Vse funkcije računalnika nadzira centralna procesna enota (CPU). Že ime samo nam veliko pove o njeni vlogi: centralna, ker je osrednji del računalnika; procesna, ker opravlja delo; in enota, ker je sklenjena celota. Iz CPU, pomnilnika in nekaj vhodno-izhodnega vezja bi že lahko sestavili preprost računalnik (glej sliko). Računalnik potrebuje vhodno/izhodne sponke za stike z zunanjim svetom. V pralnem stroju na primer vhodno-izhodno vezje vključuje motorje in grelce. Pomnilnik potrebuje za to, da hrani navodila in podatke, ki jih obdeluje CPU. Podatki lahko vsebujejo števila in dvojiške kode, ki predstavljajo znake (črke, ločila in posebne znake).

Kako CPU ve, katere pomnilnike lokacije vsebujejo navodila, katere pa podatke? Če hočemo odgovoriti na to vprašanje, moramo zvedeti nekaj več o sestavi CPU.

CPU je v 8-bitnih mikroračunalnikih, ki jih ima večina hišnih računalnikov, navadno povezana v eno samo integrirano vezje, ki ima 40 priključkov, po 20 na vsaki strani. Vsak od teh priključkov (razen dveh, na katera priključimo napajanje 0 in 5 voltov) prenaša podatke med CPU in drugimi deli računalnika (pomnilnik in vhod/izhod).

Tipična CPU ima 16 naslovnih priključkov, ki so povezani z naslovnim vodilom. Po naslovnem vodilu, ki vodi do pomnilnikov, izbira CPU lokacijo, v katero bo shranila zlog, ali pa bo iz nje zlog prebrala. S 16 naslovnimi priključki lahko CPU naslavlja 65536 različnih lokacij pomnilnika.

Naslednjo skupino sestavlja 8 podatkovnih priključkov, ki so povezani s podatkovnim vodilom. Ti prenašajo podatke med CPU in pomnilnikom ter med CPU in vhodno-izhodnim vezjem.

Drugi priključki prenašajo kontrolne signale, od katerih je nekaj vhodnih, nekaj izhodnih. Z njimi CPU nadzira delovanje drugih delov računalnika.

CPU vsebuje nekaj 8- in 16-bitnih pomnilnih celic, ki jih imenujemo registri. Nekateri od njih so rezervirani za posebne namene, drugi so namenjeni za začasno spravljanje podatkov (večnamenski registri). CPU sestavljata še dva druga pomembna funkcionalna bloka, to sta aritmetična logična enota (ALU) in kontrolni blok.

ALU (kratica za Aritmetic and Logic Unit) opravlja vse aritmetične in logične operacije, kot so na primer seštevanje, premikanje bitov v zlogu za mesto levo ali desno, in logične operacije IN in ALI.

Kontrolni blok je specialno vezje, ki vodi delovanje CPU v skladu z navodili, ki jih dobi iz pomnilnika. Za zgled si oglejmo Z80 CPU, ki je zelo pogosta v hišnih računalnikih. Če dobi kontrolni blok iz pomnilnika kodirano navodilo 11000110, bo ALU seštela naslednji zlog pomnilnika z vsebino enega od registrov v CPU. Če želimo, da se rezultat seštevanja spravi na določeno lokacijo v pomnilniku, mora biti naslednje navodilo 00110010; sledita mu še dva zloga, ki povesta, v katero lokacijo pomnilnika naj spravi rezultat.

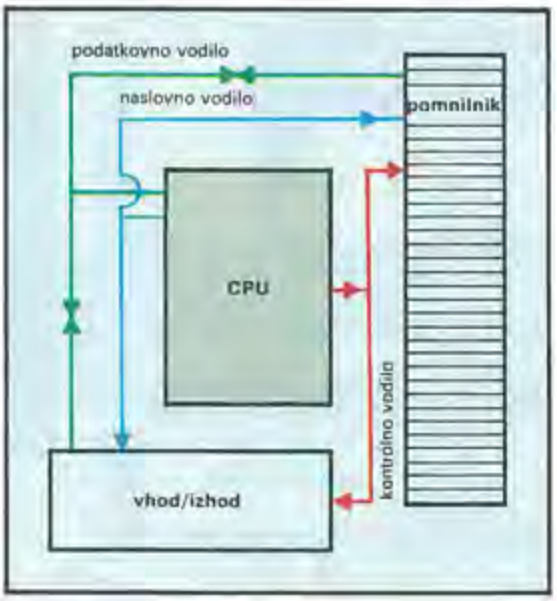
Vzemimo, da je bil rezultat seštevanja 37 (desetiško) in da sta sledila navodilu, specificirala adreso 33126 (tudi desetiško). Kontrolni blok bi dal na naslovne priključke dvojiško kodo za 33126 (1000000101100110). Nato bi po kontrolnih priključkih poslal pomnilniku signal, ki bi mu povedal, da bo moral spraviti podatke. Takoj za tem sporočilom bi dal na podatkovne priključke desetiški ekvivalent za 37 (00100101). Ti podatki bi šli po podatkovnem vodilu v pomnilnik in bi se spravili na lokaciji, ki jo določa naslovno vodilo. Če bi pozneje ta podatek spet potrebovali, bi moral CPU iz pomnilnika dobiti navodila za branje, njim pa bi sledila še dva zloga, ki bi povedala naslov lokacije, s katere je treba brati. Kontrolni blok bi tako naslovil lokacijo 33126, obvestil pomnilnik, naj pošlje zlog na tem naslovu na podatkovno vodilo, in poskrbel, da bi se podatek spravil v enem od registrov CPU.

Število in dolžina registrov sta pri raznih CPU različna. Specializirani registri imajo navadno posebna imena kot »kazalnik sklada«, »programski števec« ali »akumulator«. Večnamenske registre tudi poimenujejo, da jih ne mešamo med sabo (register X, register Y, register C).

Eden najbolj pomembnih registrov je programski števec, ki je 16-biten. Ta vedno vsebuje dvojiški naslov naslednjega navodila, ki ga je treba izvršiti. Ko se tekoče navodilo izvrši, se vsebina programskega števca prenese na naslovno vodilo, zlog, ki je v tej lokaciji pomnilnika, pa je naslednje navodilo.

Najpomembnejši 8-bitni register je akumulator. V ta register kontrolni blok začasno spravlja rezultate operacij, ki jih opravi ALU, prav tako pa lahko rabi kot vmesen pomnilnik za zloge, ki pridejo v CPU iz pomnilnika ali iz vhodno-izhodne enote, ali za zloge, ki jih bo CPU poslal v pomnilnik ali v periferijo.

V tem poglavju smo si ogledali, kako navodila iz pomnilnika vodijo delovanje CPU. Navodila povedo, v kateri lokaciji pomnilnika so spravljeni podatki, kaj naj CPU z njimi naredi, in v katero lokacijo pomnilnika naj potem spravi rezultate.



Sledenje ukazom
Preprost računalnik lahko naredimo samo iz CPU, pomnilnika in vhodno-izhodnega vezja. Pomnilnik vsebuje posebna navodila za delovanje CPU, vsebuje pa tudi podatke, ki jih CPU med izvajanjem navodil obdeluje. Z vhodno-izhodnim vezjem CPU komunicira z zunanjim svetom. Če CPU nadzira delovanje pralnega stroja, bo vhodno-izhodno vezje pošiljalo CPU signale z gumbov za upravljanje, CPU pa bo sporočal, kdaj naj vhodno-izhodno vezje vključi ali izključi motorje in grelce. Navodila so dvojiško kodirana. Vsak različen model CPU ima tudi svoj nabor kod za navodila.

Pravilno naslovljeno

CPU mora iskati navodila in podatke med tisoči zlogov, shranjenih v pomnilniku. V tem poglavju bomo zvedeli nekaj o dogajanju v CPU med izvajanjem navodil

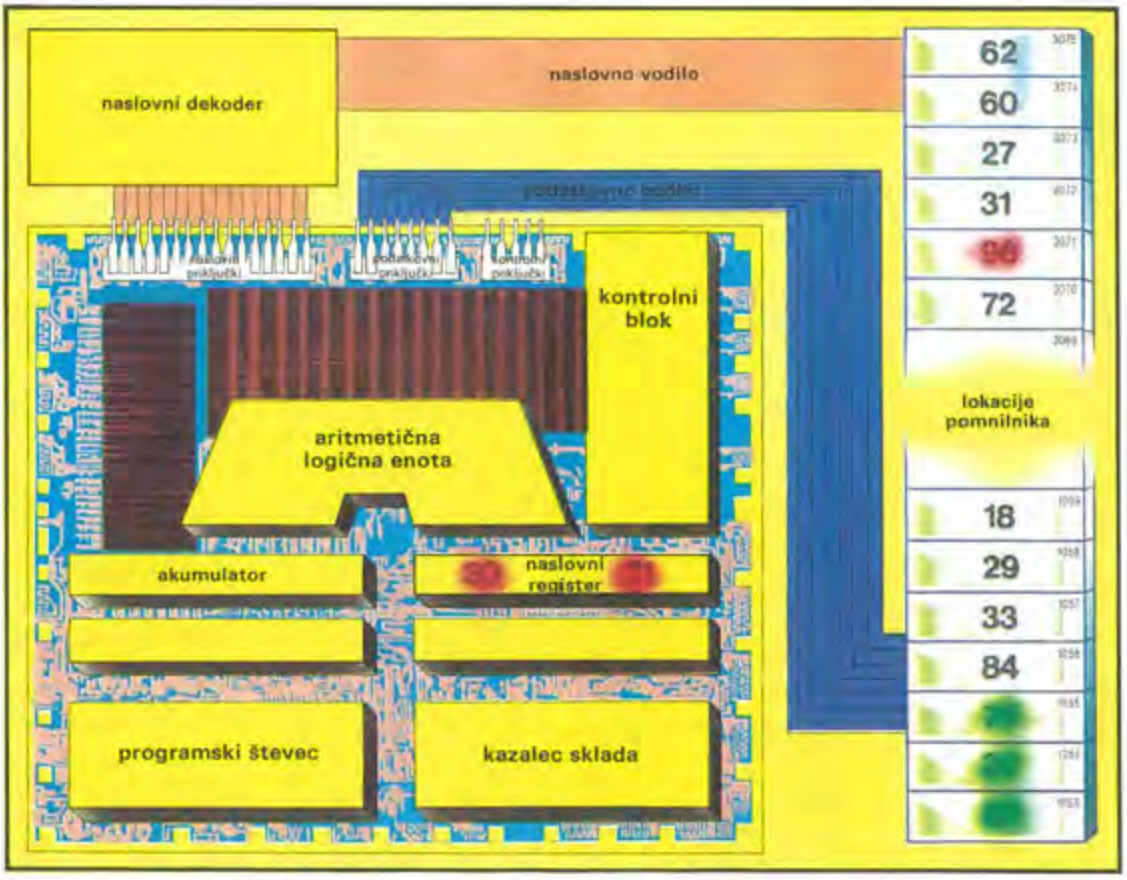
V prejšnjih poglavjih smo zvedeli, da bere CPU zloge iz pomnilnika tako, da postavi na naslovne priključke dvojiško kodo lokacije pomnilnika in nato shrani zlog, ki se pokaže na podatkovnem vodilu, v enega od registrov.

Vsaka od tisočero pomnilnih celic lahko vsebuje navodila, ki povedo CPU, kaj naj naredi, ali pa podatke, s katerimi CPU operira. Kako CPU zve, kateri zlogi so navodila in kateri podatki?

Spoznavanje kod

Najprej si razjasnimo, kaj je navodilo. To je dvojiška koda, ki povzroči, da CPU opravi specifično zaporedje operacij. Toda koda 00111010, če jo CPU spozna kot navodilo, povzroči, da CPU naslovi naslednja dva zloga pomnilnika, prebere podatke iz njiju, spravi te podatke v poseben »naslovni register«, postavi vsebino naslovnega registra na naslovne priključke, vsebino te lokacije prebere s podatkovnega vodila in jo spravi v akumulator.

To, kar smo zdaj opisali, je eden od načinov branja podatkov, ki ga uporablja Z80 CPU, ki je v hišnih računalnikih zelo pogostna. Celotni proces izvajanja navodila je prikazan na sliki. Vsa sicer dvojiška števila so predstavljena z desetiškimi ekvivalenti. Vzemimo, da CPU že ve, da bo naslednji zlog, ki ga bo prebrala iz pomnilnika, navodilo na lokaciji 1053. Ta naslov v dvojiški obliki (0000010000011101) postavi na naslov-



Veriga dogodkov
Tudi najpreprostejši ukaz za CPU je sestavljen iz več stopenj. CPU prebere ukaz iz pomnilnika, kontrolni blok ga spozna in vodi njegovo izvajanje. V tem primeru ukaz 58 iz lokacije 1053 povzroči naslednji niz dogodkov: Zlog iz naslednje lokacije pomnilnika (1054) CPU prebere in spravi v prvo polovico 16-bitnega naslovnega registra. Nato prebere še zlog iz naslednje lokacije in ga spravi v drugo polovico. Ta dva zloga skupaj sta zdaj naslov podatka v pomnilniku. Ta naslov gre na naslovno vodilo, pomnilnik postavi podatek na podatkovno vodilo, CPU ga prebere in shrani v akumulator. S tem je bil ukaz izvršen. Zato da CPU na naslovno vodilo naslov 1056, ker ve, da je na tej lokaciji naslednji ukaz. To je 84, kontrolni blok to interpretira kot invertiranje vsebine akumulatorja. Ker je to enozložni ukaz (zasede samo en zlog pomnilnika), CPU ve, da je na lokaciji 1057 že naslednji ukaz.

no vodilo. Linije, ki predstavljajo 0, so na napetosti nič, tiste, ki predstavljajo 1, pa na pozitivni napetosti. Ko dobi naslovni dekoder po naslovnem vodilu to kodo, jo dekodira in priključi pozitivno napetost na izbrano izhodno linijo, to je v našem primeru na linijo, ki vodi do lokacije 1053. Na naslednji stopnji pošlje pomnilnik zlog z lokacije 1053 na podatkovno vodilo. V našem primeru je to 58 (00111010). CPU ga prejme in ga, ker je pričakovala navodilo, pošlje kontrolnemu bloku, ki navodilo spozna in sproži natanko določeno zaporedje operacij. To navodilo določa, da naslednja dva zloga pomnilnika sestavljata 16-bitni naslov podatka, ki ga je treba prebrati in naložiti v akumulator. CPU ve, da bo dobila naslednje navodilo šele potem, ko opravi vse te operacije, in da je to navodilo na naslovu 1056.

Naslov na naslovnem vodilu se poveča za 1, na 1054. Vsebino te lokacije naloži CPU v prvo polovico naslovnega registra. Nato se naslov poveča na 1055, vsebina te lokacije pa gre v drugo polovico naslovnega registra.

Prenos številk

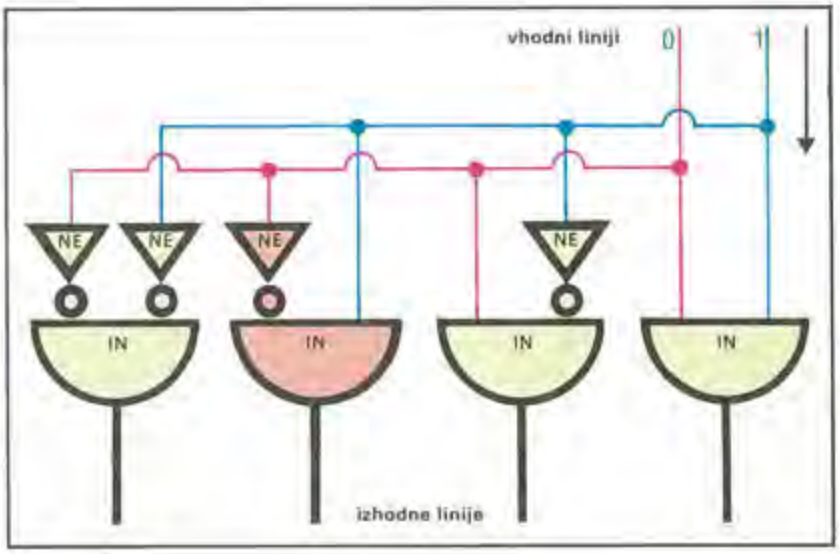
Na naslednji stopnji (vedeti moramo, da se vse to dogaja avtomatsko, kot rezultat branja navodila) se številke v naslovnem registru prenesejo na naslovno vodilo. Kakor vidimo, je to 3071 (dvojiško 0000101111111111), ki jo dekodira naslovni dekoder in izbere ustrezno pomnilno celico. Vsebina te lokacije, ki je 96 (dvojiško 01100000), gre preko podatkovnega vodila v CPU. Ta jo naloži v akumulator. Tako je CPU opravila vse operacije, ki jih je zahtevalo prejšnje navodilo. Zato postavi na naslovno vodilo naslov naslednjega navodila (1056).

Kakšno navodilo lahko pričakujemo zdaj, ko ima CPU podatek v akumulatorju? Različne CPU imajo od nekaj sto do več tisoč navodil in naslednje je lahko katerokoli izmed njih. Podatek pa smo naložili v akumulator z namenom, da napravimo kakšno operacijo. Recimo, da je ta operacija invertiranje, ki spremeni vse bite. Znake 0 zamenja z 1, znake 1 pa z 0. Navodilo za invertiranje mora torej biti na lokaciji 1056. V naši hipotetični CPU je koda za to operacijo 84. Ko CPU dobi to kodo, invertira vsebino akumulatorja. Številka v akumulatorju je bila 01100000, po invertiranju pa je 10011111. Navodilo za invertiranje zavzame samo en zlog pomnilnika, zato CPU ve, da je na lokaciji 1057 že novo navodilo.

Kode za navodila, ki smo jih uporabili na našem zgledu (58 za napolnitev akumulatorja in 84 za invertiranje), so navodila naše hipotetične CPU, vendar načelo delovanja, ki smo ga opisali, velja za vse mikroprocesorje. Edina razlika med različnimi tipi različnih proizvajalcev je nekoliko različen nabor navodil in različne kode teh navodil.

Vhodno-izhodne lokacije morajo prav tako imeti svoje naslove, načelo za njihovo naslavljanje pa je enako kakor za pomnilnik. V 8-bitnih mikroprocesorjih je na voljo le osem naslovnih linij za vhodno-izhodne lokacije, tako da je največje mogoče število teh lokacij 256. To število pa povsem zadošča za večino uporab malega računalnika.

Naslovno dekodiranje je vedno potrebno, ker lahko le tako zagotovimo, da je aktivna samo ena vhodno-izhodna ali pomnilniška lokacija, vse druge pa ostanejo nedejavne. Ta proces imenujemo »omogočanje«, ker integriranemu vezju, ki je izbrano, omogočimo, da komunicira s CPU. Kadar je potrebno dekodirati samo manjše število naslovnih linij, lahko izvedemo dekodiranje s preprosto povezavo nekaj logičnih vrat. Na sliki vidimo takšen dekoder, ki dekodira dve naslovni liniji in z njima izbere eno od štirih lokacij. Takšne dekoderje uporabljamo za dekodiranje vhodno-izhodnih lokacij, ker je njihovo število majhno. Ko pa začne število naslovnih linij rasti, se kompleksnost dekodiranega vezja zelo hitro povečuje. Dekoderji, ki izbirajo lokacije pomnilnika, so zato običajno že vključeni v samo pomnilniško integrirano vezje.



Naslovno dekodiranje
S 16 linijami, ki sestavljajo naslovno vodilo, lahko naslovimo 65536 različnih lokacij. Kombinacijo enk in ničel dekodira naslovni dekoder. Nekaj dekodiranja opravijo dekoderji, narejeni iz integriranih vezij, ki vsebujejo po nekaj logičnih vrat, dekodiranje v pomnilnikih pa opravijo posebna vezja v pomnilniških integriranih vezjih. Slika kaže, kako lahko dekodiramo dve naslovni liniji, da vsakič omogočimo delovanje enega od štirih integriranih vezij.

Digitalni dialog

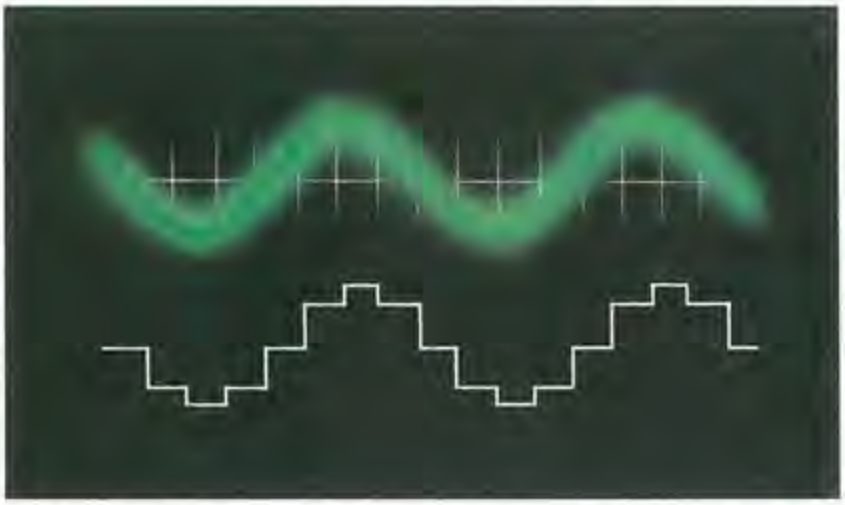
Vhod in izhod sta bistvena za delovanje vsakega računalniškega sistema

Vhod/izhod je izraz, s katerim opisujemo prenos informacij med CPU in »zunanjim svetom«. Za zunanji svet (večkrat uporabimo tudi izraz periferija) štejemo vse naprave, ki so povezane z računalnikom. Notranji svet je bistveni del računalnika, vanj štejemo CPU, bralni in delovni pomnilnik in vsa logična vezja, ki delujejo v tesni zvezi s CPU in pomnilnikom.

Zunanje naprave, ki uporabljajo vhod/izhod za komunikacijo z računalnikom, so tipkovnice, disketna enota, krmilna ročica, tiskalnik, zaslon in še bi lahko naštevali.

Ko želi CPU dobiti podatek iz pomnilnika, mora nasloviti lokacijo, kjer je ta zlog spravljen. Podobno mora pri spravljanju zloga v pomnilnik najprej nasloviti lokacijo, kamor bo zlog spravljen. Ta proces imenujemo naslavljanje pomnilnika. CPU pošlje dvojiško kodo, ki ustreza želeni lokaciji pomnilnika, na skupino 16 žic, ki jo imenujemo naslovno vodilo. Posebno vezje v pomnilniku dekodira to 16-mestno dvojiško kodo in izbere pravilno lokacijo pomnilnika. S 16 dvojiškimi znaki lahko naredimo 65536 različnih kombinacij enk in ničel, torej lahko s 16-mestno kodo naslovimo toliko različnih lokacij pomnilnika.

Če želi CPU komunicirati z neko zunanjo napravo, jo mora tudi na podoben način nasloviti. Vendar je za to na voljo le osem naslovnih linij, to omejuje število vhodno-izhodnih lokacij na 256. To je v primerjavi s številom lokacij pomnilnika majhno število, vendar povsem zadošča. Navadno namreč ni potrebno priključevati velikega števila zunanjih naprav.

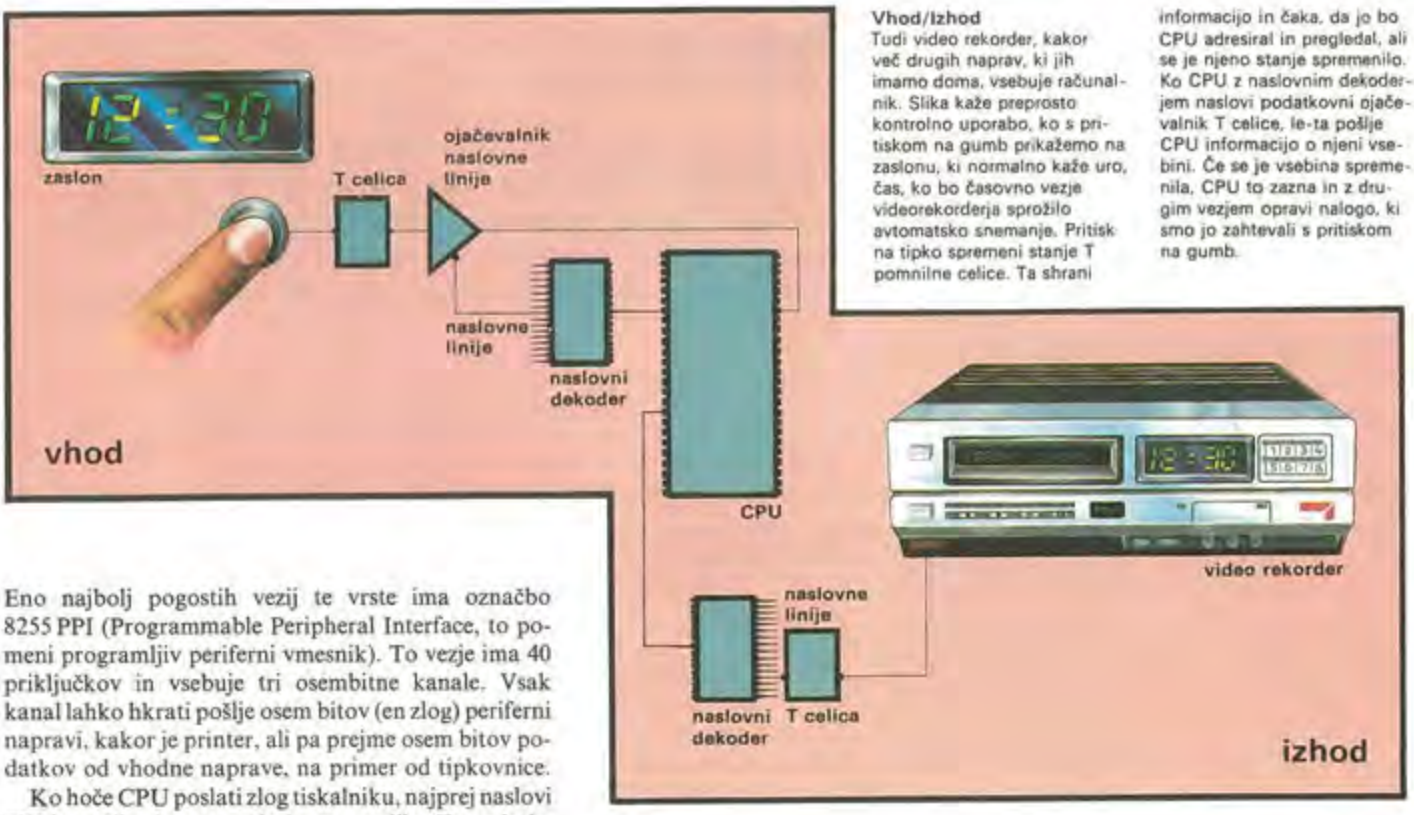


Analogno v digitalno
V realnem svetu je le malo informacij, ki bi se spreminjale v diskretnih (digitalnih) korakih. Navadno se spreminjajo zvezno kot na primer nivo morja ob plimi.
Če hočemo, da te podatke razume računalnik, jih moramo spremeniti v digitalno obliko. To napravimo z analogno-digitalnim (A/D) pretvornikom. A/D pretvornik jemlje v rednih časovnih presledkih (na primer stokrat na sekundo) vzorec analognega signala, ga spremeni v digitalno obliko in pošlje računalniku.
Ta spravi vsak vzorec v ločeno lokacijo pomnilnika, da lahko z njimi računa in jih primerja med sabo.
Digitalno-analogni (D/A) pretvorniki opravljajo nasprotno funkcijo: digitalne podatke, ki jih pošilja računalnik, spreminjajo v analogne.

Izbiranje naprav

Ob zgledu svetleče diode, ki je montirana na nekaterih računalnikih pod tipko »caps lock«, si bomo ogledali, kako računalnik izbere zunanjo napravo in ji pošlje podatke. S to tipko izbiramo, ali bomo pisali z malimi ali z velikimi črkami. Če so izbrane velike črke, svetleča dioda sveti, če so izbrane male, ne sveti. Računalnik mora sporočiti svetleči diodi, ali mora biti prižgana ali ugasnjena, za to mu zadošča en bit. Svetleča dioda, čeprav sprejema le en bit informacije, mora kot izhodna naprava imeti svojo adreso. CPU ne more ves čas naslavljati svetleče diode in ji sporočati, ali naj bo prižgana ali ne, saj bi za to porabila preveč časa. Naslovi jo le, če ji hoče sporočiti, da mora svoje stanje spremeniti (da mora ugasniti ali se prižgati). Recimo, da ima svetleča dioda naslov 32. Če jo želi CPU nasloviti, mora dati na naslovne linije dvojiški ekvivalent za 32, to je 00100000. Pred svetlečo diodo je posebno vezje (dekoder), ki spozna kodo 00100000 na svojem vhodu in da takrat na svojem izhodu pozitivno napetost. Svetleča dioda je priključena na vezje, ki se imenuje T pomnilna celica, ki deluje tako, da spremeni svojo vrednost, kadar se na izhodu dekoderja pojavi pozitivna napetost. Kadar je vrednost T celice 1, svetleča dioda sveti, ko pa je vrednot 0, ugasne. Tako se stanje svetleče diode spremeni vedno, kadar jo CPU adresira.

Večina zunanjih naprav, s katerimi komunicira CPU, je veliko bolj kompleksna od svetleče diode. Tiskalnik je tipična takšna naprava. Povsod tam, kjer se prenašajo velike količine informacij (kakor je to pri tiskalniku), vstavimo med CPU in periferno napravo posebno integrirano vezje vhodno-izhodni vmesnik.



Vhod/Izhod
Tudi video rekorder, kakor več drugih naprav, ki jih imamo doma, vsebuje računalnik. Slika kaže preprosto kontrolno uporabo, ko s pritiskom na gumb prikažemo na zaslonu, ki normalno kaže uro, čas, ko bo časovno vezje videorekorderja sprožilo avtomatsko snemanje. Pritisk na tipko spremeni stanje T pomnilne celice. Ta shrani informacijo in čaka, da jo bo CPU adresiral in pregledal, ali se je njeno stanje spremenilo. Ko CPU z naslovnim dekoderjem naslovi podatkovni ojačevalnik T celice, le-ta pošlje CPU informacijo o njeni vsebini. Če se je vsebina spremenila, CPU to zazna in z drugim vezjem opravi nalogo, ki smo jo zahtevali s pritiskom na gumb.

Eno najbolj pogostih vezij te vrste ima označbo 8255 PPI (Programmable Peripheral Interface, to pomeni programljiv periferni vmesnik). To vezje ima 40 priključkov in vsebuje tri osembitne kanale. Vsak kanal lahko hkrati pošlje osem bitov (en zlog) periferni napravi, kakor je printer, ali pa prejme osem bitov podatkov od vhodne naprave, na primer od tipkovnice.

Ko hoče CPU poslati zlog tiskalniku, najprej naslovi PPI in pošlje zlog na podatkovno vodilo. Te podatke PPI začasno spravi v svojem osembitnem pomnilniku, imenovanem register, in jih nato pošlje tiskalniku. Podobno se podatki, ki prihajajo od zunanje naprave, najprej spravijo v register PPI, ta pa jih da na podatkovno vodilo, kakor to zahteva CPU. Vhodne naprave ne morejo vedno pošiljati podatkov na podatkovno vodilo, ker računalnik podatkovno vodilo uporablja za prenos podatkov med CPU, pomnilnikom in drugimi perifernimi napravami. Zato PPI te podatke spravi in jih pošlje na podatkovno vodilo šele na zahtevo CPU.

Kako pa CPU zve, da poskuša periferna naprava poslati podatke? Za to sta na voljo dve tehniki.

CPU lahko v določenih časovnih presledkih preneha izvajati program in pregleda vse vhodne kanale. Če odkrije, da podatki čakajo na vpis, pošlje ustrezni PPI navodilo, da pošlje podatke.

Druga metoda uporablja »prekinitve«. Naprava, ki želi poslati podatke, pošlje CPU signal, ki prekine izvajanje tekočega programa. CPU prebere vhodne podatke in nadaljuje izvajanje programa. Prednosti in slabosti teh dveh metod bomo nadrobneje opisali kasneje.

Vhodno-izhodni vmesnik, ki smo ga opisali, je predstavnik vzporednih vmesnikov, ker pošilja in sprejema skupino osmih bitov (en zlog) hkrati. Poznamo pa še serijske vmesnike, ki sprejemajo in oddajajo bite drugega za drugim (le po enega). Serijske vmesnike uporabljajo nekateri tiskalniki, serijski so tudi izhodi iz modemov (glej stran 104). Glavna prednost serijskih vmesnikov je, da potrebujejo za komunikacijo le dve žici namesto osmih ali več, kakor jih zahtevajo vzporedni.

Serijski in vzporedni vmesniki
Večina sodobnih mikroračunalnikov ima oboje: serijske in vzporedne vmesnike. Serijski prenašajo po en bit, vzporedni pa kar cel zlog. Najbolj pogost tip serijskega vmesnika je znan pod označbo RS232C, ki navadno uporablja miniaturni konektor D (na levi vidimo tip s 25 priključki), redkeje pa DIN vtičnico, ki jo poznamo iz hi-fi sistemov. Vzporedni vmesniki uporabljajo standard IEEE488, ki ga je razvila firma Hewlett-Packard in ga je Institute of Electrical and Electronic Engineers sprejel kot industrijski standard v ZDA.

Čakalnica

Računalnik prenaša informacije z veliko večjo hitrostjo, kakor jo zmorejo mehanske naprave, recimo tiskalnik. Ta problem rešimo z uporabo vmesnih pomnilnikov

Zamislimo si program za urejanje teksta, ki lahko med drugim premakne del teksta iz enega dela besedila v drugega. To besedilo je v pomnilniku računalnika sestavljeno iz presledkov in znakov, ki jih bo tiskalnik natiskal, vmes pa so tudi znaki, ki jih ne tiska, kot npr. znak CR, ki pove tiskalniku, da mora besedilo od tega znaka naprej pisati v novo vrsto. Vsi ti znaki so v pomnilniku predstavljeni s svojimi kodami ASCII, pri čemer vsak znak zavzema en zlog. Celotno besedilo zavzema določen prostor pomnilnika, tako da si adrese znakov, ki si sledijo v tekstu, sledijo tudi v pomnilniku.

Če želimo zdaj del teksta prenesti na drug del celotnega besedila, moramo najprej nekje drugje v pomnilniku rezervirati prostor, kamor začasno spravimo ta tekst. Besedilo nato preuredimo tako, da napravimo prostor na mestu, kamor želimo vpisati naš tekst, pri tem pa seveda drugi deli besedila zasedejo lokacije, kjer je bil prej naš tekst. Nato lahko naš del teksta prenesemo nazaj na želeno mesto v besedilu. Takšno področje pomnilnika, ki ga uporabimo za začasno pomnjenje, imenujemo vmesni pomnilnik.

Naslednji zgled je tiskanje besedila, ki smo ga uredili z urejevalnikom teksta. Vzemimo, da je celotno besedilo sestavljeno iz 15000 znakov. Teh znakov ne moremo poslati tiskalniku hkrati, ker večina tiskalnikov ne more tiskati hitreje kakor 80 znakov na sekundo. Da rešimo ta problem, moramo rezervirati del pomnilnika, ki potem rabi kot vmesni pomnilnik za tiskalnik. Programska oprema napolni ta pomnilnik s tekstom in ga nato pošilja tiskalniku s primerno hitrostjo.

Načelo ostane enako, tudi če je ta vmesni pomnilnik majhen (128 ali 256 bytov). Najprej ga hitro napolnimo z »blokom« znakov ASCII, nato pa zloge počasi oddajamo tiskalniku. Prvi zlog, ki smo ga vpisali, bo tudi prvi odposlan, ker želimo, da tiskalnik natiska znake v istem vrstnem redu, kakor smo jih uredili. Tak tip organizacije pomnilnika imenujemo pomnilnik FIFO. FIFO je kratica angleškega izraza First In First Out, kar v dobesednem prevodu pomeni »prvi noter, prvi ven«. Ko tiskalnik dobi vse znake iz vmesnega pomnilnika, ga programska oprema spet napolni z naslednjim delom besedila in to potem ponavlja tako dolgo, da je ves tekst natisnjen.

Pomnilnik FIFO je zelo pogosta struktura v večini operacijskih sistemov računalnikov. Uporabljajo ga povsod tam, kjer nastajajo nesorazmerja v hitrostih. Do tega pride zelo pogosto, ne samo med računalnikom in tiskalnikom, tudi med računalnikom in disketno enoto, pa med računalnikom in tipkovnico. Zaradi izredne hitrosti delovanja lahko računalnik zaznava pritisk hitreje, kakor lahko mi tipkamo. Včasih pa se le zgodi, da ne more dovolj hitro spoznavati tipk in jih prikazovati na zaslonu. Lahko je, recimo, popolnoma zaseden s kakšno drugo dejavnostjo (npr. komuniciranje z diskovno enoto). V nekaterih računalnikih so ta problem rešili tako, da so v operacijski sistem vključili vmesni pomnilnik, ki si »zapomni« vse naše pritiske na tipke, računalnik pa jih prikaže na zaslonu takoj, ko to zmore. Navadno delovanja tega vmesnega pomnilnika niti ne opazimo, pri nekaterih sistemih, ki veliko

Začasni pomnilnik
Ena najbolj pogostih uporab vmesnega pomnilnika je »posredovanje« med računalnikom in tiskalnikom, ker tiskalnik ne more tiskati znakov tako hitro, kakor jih računalnik lahko pošilja. Zato računalnik pošilja znake, dokler mu tiskalnik ne sporoči, da je vmesni pomnilnik poln. Zloge nato tiskalnik natiska v istem vrstnem redu, vendar veliko počasneje, kakor so prišli v vmesni pomnilnik. Ko natiska vse znake iz vmesnega pomnilnika, se proces ponavlja, dokler ni natiskano celo besedilo.

Če z vedrom nalivamo vodo v posodo na sliki, bo skozi pipo počasi tekla z enakomernim curkom. Enako računalnik »nalivar« podatke v vmesni pomnilnik, ta pa jih nato spušča z veliko manjšo hitrostjo, kakršno zmore tiskalnik.

komunicirajo z diskom ali izvajajo programe, ki zahtevajo veliko procesiranja, pa lahko včasih le opazimo zakasnitev med trenutkom, ko pritisnemo na tipko, in ko se njen znak pokaže na zaslonu. Nekateri operacijski sistemi nam omogočajo, da se odločimo, ali bomo ta vmesni pomnilnik uporabljali ali ne. Drugi celo dopuščajo spreminjanje števila znakov, ki jih lahko vnaprej odtipkamo.

Način, kako programska oprema dejansko operira z vmesnimi pomnilniki, je odvisen od njihovega namena. Programska oprema mora vedno vedeti, koliko zlogov smo že prebrali iz polnega vmesnega pomnilnika, drugače se lahko zgodi, da pomembne podatke izgubimo.

Zunanji podatkovni vmesniki

Večina mikroračunalnikov ni zmožna opravljati drugih dejavnosti, dokler poteka tiskanje. Zato moramo ob takem sistemu lepo sedeti in čakati na konec dolgotrajnega tiskanja. Veliko proizvajalcev pa zdaj ponuja na tržišču vmesne pomnilnike, ki jih vključimo med računalnik in tiskalnik.

Ko mora računalnik natiskati določen tekst, začne pošiljati tiskalniku zloge, dokler mu ta s signalom »sem zaposlen« ne sporoči, da ne more več sprejemati znakov. Računalnik počaka, da ta signal postane »napačen«, kar pomeni, da tiskalnik spet lahko sprejema znake za tiskanje. Čeprav imajo tiskalniki navadno vdelane majhne podatkovne vmesnike, ti običajno niso večji od 2 kb. Zato mora računalnik pošiljati tekst v majhnih kosih in ves čas tiskanja komunicirati s tiskalnikom, to pa mu jemlje veliko časa.

Zunanji podatkovni vmesniki pa imajo velik pomnilnik (nekateri celo do 16 kb) z vdelano programsko opremo. Ta pomnilnik je tako velik, da običajno sprejme celotno besedilo hkrati, in to računalniku omogoči, da takoj nadaljuje delo. Vmesni pomnilnik nato namesto računalnika pod kontrolo svoje lastne programske opreme počasi pošilja podatke tiskalniku.

Do sedaj smo opisovali pomnilnike kot velikanske predalnike, v katere shranjujemo podatke in programe tako, da je vsak zlog dostopen kadarkoli. V tem poglavju pa smo že spoznali tudi drugačno pomnilniško strukturo, v kateri so najprej dostopni tisti podatki, ki smo jih prve shranili. Zadnji zlog, ki smo ga shranili, bomo lahko prebrali šele, ko bomo prebrali vse pred njim. Takšno strukturo smo imenovali FIFO.

Struktura, ki je tej podobna, je sklad. Podatke, shranjene v skladu, lahko beremo v obrnjenem vrstnem redu, kakor smo jih vpisovali. Prvi zlog, ki smo ga vpisali, je zadnji, ki ga lahko preberemo. Sklade lahko primerjamo s skladovnico desk. Desko, ki smo jo prvo položili na kup, bomo lahko zadnjo vzeli z njega. Sklad imenujemo tudi LIFO, kar je kratica angleškega izraza Last In First Out, kar pomeni »zadnji noter, prvi ven«.

Podobno kakor vmesne pomnilnike uporabljamo tudi sklade kot začasne pomnilnike. Strukturi sta si po svoji organizaciji zelo podobni, razlikujeta se le po vrstnem redu branja podatkov. Sklade uporabljajo interno tudi jeziki visokih ravni (npr. basic interpreter), ko je treba informacijo shraniti in jo ponovno prebrati. Poglejmo si tale del programa v basicu:

```
FOR X = 1 TO 10
PRINT "X = " ; X
FOR Y = 1 TO 10
GOSUB SCAN
NEXT Y
PRINT "C$ = " ; C$
NEXT X
```

To je zgled vgnezdenih zank FOR...NEXT. Ko pride basic interpreter do drugega stavka FOR, si mora za kasnejšo uporabo zapomniti spremenljivko prejšnjega (zunanjega) stavka FOR (v tem primeru X). To naredi tako, da »naloži« informacijo o prvem stavku FOR na sklad, ko pa konča delo v notranji zanki FOR, informacijo z vrha sklada spet vzame. Tako spet dobi informacijo o spremenljivki X. Zanke FOR...NEXT lahko večkrat gnezdimo, zato mora interpreter vsakič, ko preide na novo notranjo zanko, naložiti na sklad informacijo o prejšnjem stavku FOR. Ko se interpreter vrača v zunanje zanke, mora dobiti informacije v obrnjenem vrstnem redu, kakor jih je nalagal; to mu sklad tudi omogoča.

Strukture FIFO pogosto uporabljamo v vhodno-izhodnih rutinah kot »posrednike« med rutinami ali napravami, ki so ločene med sabo ali delujejo z različnimi hitrostmi. Vhodna rutina v basicu dela na primer s celimi vrsticami, interpreter pa nato pregleduje vsak znak posebej. Zato vhodna rutina naloži vrstico v vmesnik FIFO, interpreter pa nato pregleduje vsak znak posebej. Za delo z vmesnikom FIFO potrebujemo še »kazalec«, ki pove, kaj je v pomnilniku naslednji zlog vmesnika FIFO. Ta kazalec je v bistvu zlog ali nekaj zlogov, ki vsebujejo adreso naslednjega zloga v vmesniku FIFO. Ko interpreter pregleda en znak, se vsebina kazalca poveča za ena, to je adresa naslednjega znaka za obdelavo.

Dva načina prevajanja

Računalniki mislijo v strojnem jeziku: programerji raje pišejo programe v višjih jezikih, kakršen je npr. basic. Prevajalnik in interpreter sta dva načina za pretvorbo višjega jezika v strojni jezik

Prvi računalniki niso imeli tipkovnice. Programske ukaze so prenašali v računalnik korak za korakom, z nastavljanjem vsakega od osmih stikal »gor« ali »dol«.

Pozneje so tipkovnice zamenjale stikala, vzorce na stikalih pa so zamenjali z angleškimi besedami. Rezultat teh zamenjav so »višji jeziki«, kakor je na primer basic, ki so zamenjali strojne jezike. Računalniki pa se niso spremenili. Še naprej so delali (in še vedno delajo) z vzorci ničel in enk, zato so morali programerji v strojnem jeziku napisati program, ki prevaja programe, pisane v višjih jezikih, v strojni jezik. Ti programi so dveh vrst in se v skladu z načinom prevajanja imenujejo interpreterji ali prevajalniki.

V računalništvu (kakor tudi drugje) je treba vsako pridobitev v moči ali hitrosti plačati — z denarjem, s časom ali s svobodo delovanja. Tako je tudi z interpreterji in prevajalniki. Skupaj dajejo vse možnosti, ki jih programer potrebuje pri prevajanju. Interpreterji so močni na enih področjih, prevajalniki na drugih, toda vsak plačuje svoje prednosti z ustreznimi slabostmi. Interpreterji, običajno vdelani v hišne računalnike, so cenen način za prevajanje visokih jezikov v podatke, ki jih računalnik razume. Ne potrebujejo veliko lokacij v pomnilniku in tako puščajo več prostora programerju.

Skoraj vsi mikroračunalniki, ki stanejo približno 100000 dinarjev ali manj, imajo interpreter za basic. Vi napišete program v basicu, odtipkate RUN in program dela ali pa se ustavi in sporoči napako — na primer:

```
SYNTAX ERROR ON LINE 123
(SLOVNIČNA NAPAKA V VRSTICI 123)
```

Ko najdete napako, jo popravite, napišete RUN in program dela, ali pa se ponovno ustavi, in tako naprej. Boljši interpreterji za basic sproti preverjajo sintakso ukazov.

Podobne stvari ste mogoče delali že stokrat, ne da bi sploh pomislili na interpreter. Ravno to je njegova glavna vrlina. Interpreter je nevidno orodje, ki vam omogoča delo na programu, ne da bi vas pri tem skrbelo, v katerem delu pomnilnika je in kako dela.

Program je v vaših rokah in ga lahko sproti izvajate in popravljate. Interpreter se da zlahka uporabljati, vendar ni posebno dognan: vsakič, ko napišete RUN, mora interpreter poiskati program v pomnilniku in ga prevesti v strojni jezik vrstico za vrstico. Če program vsebuje zanko

```
400 LET N=0
500 PRINT N
600 LET N=N+1
700 IF N<100 THEN GOTO 500
```

mora interpreter prevesti in narediti vrstice od 500 do 700 stokrat, ko da jih še nikoli prej ni videl.

Prevajalniki so drugačni. So dragi, težko jih je pisati in zasedajo in uporabljajo precej lokacij v pomnilniku. Skoraj vedno so zapisani na disku, zato uporabnik potrebuje drag računalniški sistem. Vrline prevajalnikov so prilagodljivost, moč in hitrost. Zgornje štiri vrstice programa bi prevajalnik prevedel samo enkrat in jih potem izvedel stokrat. S tem si lahko prihranimo veliko časa — toda za precejšnjo ceno. Vzemimo, da imate prevajalnik za basic in želite napisati in izvajati program v basicu.

Najprej naložite in poženete program za pisanje datoteke (Editor), ki vam omogoča, da napišete program in ga shranite na disk kot »izvirnik«.

Datoteke morajo imeti imena, tako da jih lahko najdete, potem ko jih napišete, zato vas Editor vpraša za ime izvirnika. Imena v datoteki so običajno sestavljena iz dveh delov, prvi lahko vsebuje poljubno ime — recimo MOJPRG — drugi del pa po navadi sestavljajo tri črke, ki kažejo značilnost datoteke. To kodo imenujemo »podaljšek«. Datoteka, napisana v basicu, ima lahko podaljšek BAS. Vaš izvirnik je zdaj na disku in ima ime MOJPRG.BAS. Če zdaj natipkate:

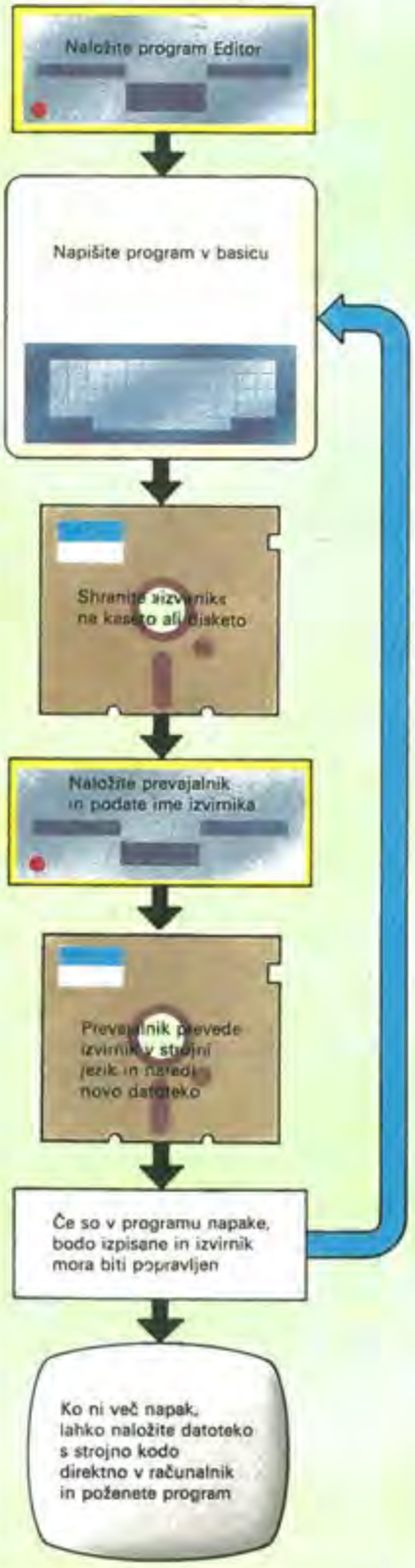
```
COMPILE MOJPRG.BAS
```

bo računalnik naložil in pognal prevajalnik za basic v vašem izvirniku, MOJPRG.BAS.

Potem je potrebno počakati nekaj sekund, čas je odvisen od dolžine datoteke, da prevajalnik prevede program v strojno kodo, ki jo shrani na disk pod imenom MOJPRG.OBJ — podaljšek OBJ kaže, da je to datoteka strojnih kod izvirnega programa.

Medtem ko prevajalnik prevaja vašo datoteko, preverja, ali so v njej slovnične napake. Če najde kakšno napako, potem boste dobili poročilo podobno temu:

Prevajanje programa
Pisanje programov za prevajalnike ni niti zdaleč tako preprosto kakor uporaba interpreterja na vašem računalniku. Ko pa je program napisan, se izvaja veliko hitreje. Tukaj je zapisan potek pisanja programa:




```
100 REED X:IF X=3(N+2) LET P=Q
     1                 2      3
FATAL ERROR:—
1) //REED// UNRECOGNISED COMMAND
2) //(// ILLEGAL OPERATOR HERE
3) ??'THEN' OR 'GOTO' EXPECTED HERE
```

Takšno sporočilo dobite za vsako vrstico, ki vsebuje napako. Z drugimi besedami, poročilo o napakah je dosti bolj obsežno kakor pri interpreterju. Zdaj morate naložiti in še enkrat pognati program Editor, poklicati izvirnik iz diska, ga popraviti in poizkusiti. Če ni več napak, lahko odtipkate:

```
RUN MOJPRG
```

in program dela tako, kakor pričakujete, ali pa tudi ne. Na tej stopnji ni več slovničnih napak, ker ste jih popravili. Če kljub temu želite spremeniti program, tedaj morate naložiti in pognati Editor in ponoviti vse drugo.

V času razvijanja programa je težko spoznati dobre strani prevajalnika, čeprav je natančen opis napak zelo pomemben. Njegove prednosti se uveljavljajo takrat, ko imate delujoč program in odtipkate RUN, ravno tam, kjer vas interpreter začne puščati na cedilu.

Prevedeni programi so hitri — od 5- do 50-krat hitrejši od interpretiranih programov (odvisno od zmogljivosti računalnika), toda hitrost izvajanja prevedenega programa je dosežena na račun počasnejšega razvoja programa.

Primerjanje prevajalnikov in interpreterjev na podlagi enakih programov ni objektivno, saj so prevajalniki pisani večidel za večje računalnike, katerih uporabniki lahko pišejo in izvajajo programe v različnih programskih jezikih.

Cobol (jezik za obdelavo obračunov, plačilnih list in zalog) je bil razvit za prevajalnik, medtem ko basic zares zahteva interpreter. Ko je program razvit, izvirnika ne potrebujete več, razen morda za dokumentacijo. Zato morajo biti izvirniki opremljeni s komentarjem in pisani čitljivo, medtem ko je datoteka strojne kode krajša in jemlje manj prostora v pomnilniku. Dejstvo, da se datoteke, ki jih ustvari prevajalnik, ne dajo brati, ker so v strojni kodi, ima svoje prednosti. Če prodajate programsko opremo, potem ne prodajate izvirnika, temveč samo strojno kodo, ki jo je veliko težje posneti, kopirati ali spremeniti.

Počasi
Pri pustolovskih igrah hitrost ni kritična in se večina programov sestoji iz obdelave nizov besedil. Zato so napisane v basicu in jih interpreter sproti prevaja.

Hitreje
Veliko poslovnih programov (posebno tabele) je težko pisati v strojnem jeziku, ker zahtevajo veliko matematičnih operacij. Interpreter bi bil prepočasen, zato so često pisani v basicu in potem prevedeni s prevajalnikom.

Najhitreje
Za hitre in razburljive arkadne igre, ki imajo tudi bogato grafiko, je celo program, preveden s prevajalnikom, prepočasen. Ti programi morajo biti pisani v strojnem jeziku, to pa je počasno in težko delo.

PEEK in POKE

Ukaza uporabljamo, kadar želimo napisati program, ki mu basic ni kos, vendar ju vsak računalnik uporablja drugače

Uporaba ukaza POKE
Ukaz POKE spreminja vsebino pomnilniške lokacije, zato morate biti pri delu z njim zelo previdni. Sprememba vsebine napačne lokacije lahko povzroči spremembe pri poteku programa. Večje škode ne morete povzročiti, lahko pa izgubite program. Tukaj je podanih nekaj avarnih zgledov uporabe ukaza POKE.
Na računalniku Atari 4000 ali 800 bo zapis vrednosti 1 na lokacijo 751 na zaslonu izbrisal kurzor. POKE 751,1. Na računalniku Commodore 64 poizkusite napisati ukaz POKE 1024,1. 1024 je naslov prve lokacije pomnilnika, ki se zapiše na zaslonu. Na računalniku Sinclair Spectrum poizkusite:

```
100 FOR N=0 TO 6 STEP 2
110 POKE USR"A"+
    N,BIN01010101
120 POKE USR"A"+N+
    1,BIN10101010
130 NEXT N
140 PRINT "AAAAAAAA"
```

Pri tipkanju A-jev mora biti računalnik v grafičnem načinu delovanja. Program bo na zaslon računalnika narisal nekaj nenavadnih vzorcev simbolov.

PEEK in POKE sta dva ukaza iz basica, ki se uporabljata v programih, kjer je potrebno delati s posameznimi biti in zlogi v pomnilniku računalnika. Ukaz PEEK se uporablja za pregled vsebine določenega naslova (lokacije) v pomnilniku, ukaz POKE pa za shranjevanje števil (od 0 do 255) v določeni lokaciji pomnilnika računalnika.

Ukaza PEEK in POKE omogočata programerju dostop do podatkov v računalniku na način, ki ni običajen. Normalno basic sam skrbi za lokacije, kjer so shranjene spremenljivke in podatki, ki določajo znake, ki bodo zapisani na zaslon. Čeprav nas običajno ne zanima, kje v pomnilniku so ti podatki shranjeni, moramo včasih vedeti tudi to. V takih primerih uporabimo ukaz PEEK.

Za pregled vsebine poljubne lokacije pomnilnika računalnika lahko napišemo kratek program:

```
10 REM POGLED V POMNILNIK RAČUNAL-
   NIKA
20 PRINT "ZAPIŠI LOKACIJO POMNILNIKA V
   DESETIŠKI OBLIKI"
30 INPUT M
40 P=PEEK (M)
50 PRINT "VSEBINA LOKACIJE"; M; "JE" ; P
60 GOTO 20
70 END
```

Program bo izpisal vsebino naslova, določenega v desetiški obliki (računalnik naslov shrani v binarni obliki). Basic vsebuje funkcijo za spremembo decimalnega števila v ustrezni znak. To je funkcija CHR$ in sprememba vrstice 50 bo povzročila tiskanje znakov, pripadajočih vsebini pomnilnika:

```
50 PRINT "VSEBINA LOKACIJE"; M;":";
   CHR$(P)
```

Za pregled celotnega pomnilnika lahko dodamo FOR...NEXT zanko, izbrišemo vrstico 30, spremenimo vrstico 20 v FOR X = 0 TO 65535 ter zamenjamo vrstico 60 z NEXT X.

Če želimo počasnejše prikazovanje znakov, lahko dodamo kasnilno zanko za PRINT in pred NEXT X. Zgornja meja FOR...NEXT zanke predvideva 64 kb lokacij. Pri manjših pomnilnikih je treba zgornjo mejo spremeniti. Za 32 kb je meja 32767, za 48 kb je 49151.

Celotni izpis programa je takšen:

```
10 REM PREGLED IN TISKANJE VSEH
   LOKACIJ POMNILNIKA
20 FOR X = 0 TO 65535
30 LET Y = PEEK(X)
40 PRINT "LOKACIJA"; X;" = ";"Y"; "=";
50 PRINT CHR$(Y)
60 FOR D = 1 TO 200
70 NEXT D
80 NEXT X
90 END
```

Čeprav funkcija CHR$ prevaja decimalna števila v ustrezne znake, so znaki, ki jih je moč tipkati, predstavljeni s števili od 32 do 128. Večina računalnikov uporablja števila od 128 do 255 (največje število, ki je lahko shranjeno v enem zlogu) za posebne grafične znake. Števila od 0 do 31 so večinoma uporabljena za kontrolne znake. Ko program pride do takšnega znaka, jih CHR$ prevede v funkcijo, ki povzroči določene spremembe na zaslonu, na primer zatemnitev celotnega zaslona ali pomik kurzorja v zgornji levi kot zaslona.

Ukaz POKE deluje nasprotno kakor PEEK. Omogoča nam, da »zapišemo« zlog informacije (število med 0 in 255) v poljubno lokacijo pomnilnika. Pri tem pa moramo biti zelo previdni. Če zapišemo število v napačno lokacijo, lahko računalnik »pokvarimo«, ker zmedemo kak del bistvenega programa. V tem primeru moramo računalnik resetirati (ga izklopiti in ponovno vklopiti, če nima za to posebne tipke), pri tem pa lahko uničimo program. Preden začnemo pisati POKE v pomnilnik računalnika, preverimo, kateri deli pomnilnika so namenjeni uporabniku. Večina računalnikov omogoča dostop do video pomnilnika (dela pomnilnika, kjer so shranjeni podatki, ki so zapisani na zaslonu). Običajno računalnik bere oblike znakov, ki jih prikazuje na zaslon iz posebnega bralnega pomnilnika, kjer so shranjeni vzorci grafičnih točk za vsak znak. Po navadi pa je mogoče uporabljati tudi delovni pomnilnik. Ko so kode vzorcev shranjene v delovnem pomnilniku, lahko z uporabo ukaza POKE zapišemo nove kode, definirane kot decimalna števila, in tako dobimo popolnoma nove oblike znakov.

Parno pravilo

»Parno pravilo« zagotavlja, da je število enic v zlogu vedno parno. Tako laže odkrijemo napako pri prenosu podatkov

Ena glavnih prednosti digitalnih računalnikov pred analognimi je ta, da se napake in netočnosti, ki nastopajo v elektronskih vezjih, ne seštevajo, kadar signal preide več stopenj. Vendar je pri prenosu podatkov na večjo razdaljo — s serijskim vmesnikom in žičnim parom ali po telefonski liniji — električni »šum« na liniji včasih dovolj velik, da posamezen bit prevrže z ničle na enico ali narobe.

Računalnik pri sprejemu navadno ne more ugotoviti, da se je to zgodilo, in bo sprejete podatke obravnaval kot pravilne.

Poglejmo, kaj se zgodi, če se en bit v kodi ASCII za črko Q pokvari:

() 1 0 1 0 0 0 1 (Oddana koda ASCII za Q)
() 1 0 0 0 0 0 1 (Sprejeta koda ASCII za A)

Taka napaka je pri prenosu vsaj nadležna, je pa lahko tudi katastrofalna. Spomnimo pa se, da je koda ASCII definirana le za vrednosti od 0 do 127, kar zahteva 7 bitov (označenih od 0 do 6). Najtežji bit (bit 7) zato dostikrat uporabljamo kot »parni« bit, da lahko ugotovimo, ali se je pri prenosu zgodila napaka.

Imamo dva dogovora za uporabo parnih bitov: parno pravilo in neparno pravilo. Poglejmo si prvo! Parno pravilo pravi, da postavimo parni bit (bit 7 v kodi ASCII) tako, da je celotno število enic v zlogu vedno parno. Takšnile bi bili kodi za A in Q s parnim bitom:

(0) 1 0 0 0 0 0 1 (koda ASCII za A s parnim bitom)
(1) 1 0 1 0 0 0 1 (koda ASCII za Q s parnim bitom)

V kodi ASCII za A sta dve enici, zato postavimo parni bit 0, vsota vseh bitov je parna. V kodi ASCII za Q so tri enice, zato postavimo parni bit 1. Tako dobimo vsoto bitov 4, kar je parno število.

Zdaj pa si poglejmo, kaj se zgodi, če se bit 4 v kodi ASCII Q spremeni kakor v gornjem primeru:

(1) 1 0 0 0 0 0 1 (napačna koda ASCII za Q)

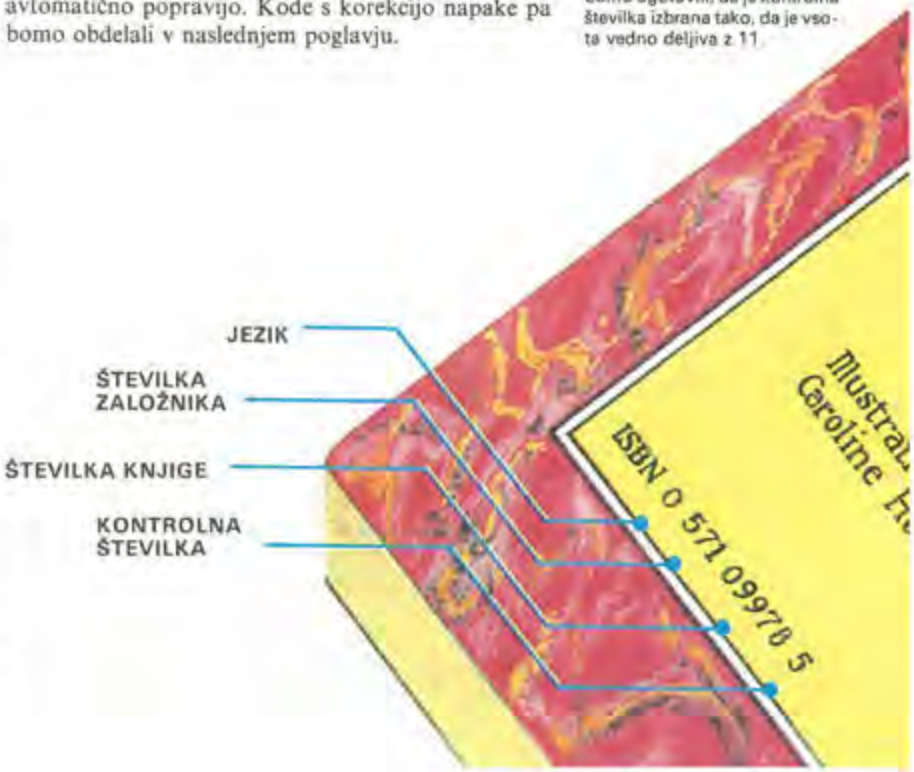
Ko preverimo parnost zloga (programsko ali s posebnimi vezji), vidimo, da ima pravilna koda za Q parno število enic (z vštetim parnim bitom). Nasprotno pa ima napačna koda za Q bit 4 spremenjen z 1 na 0, originalni parni bit (bit 7) pa je še vedno ostal 1. Pri prevajanju parnosti takega zloga bomo ugotovili, da ima neparno število enic. Po tem bomo vedeli, da je z zlogom nekaj narobe in ga lahko zavržemo. Če dobro premislimo, bomo videli, da tudi v primeru, ko se zgodi napaka na parnem bitu, postopek preverjanja parnosti pokaže, da se je pri prenosu zgodila napaka, in napačni zlog bo zavržen.

Če si ogledate kode ASCII v svojem računalniku, boste ugotovili, da se sedmi bit v resnici uporablja za dodaten set znakov (običajno za grafične znake), ker so napake pri prenosu podatkov znotraj računalnika zelo redke. Parnost navadno uporabljamo samo pri prenosu podatkov na večje razdalje ali pri zapisu na magnetni medij (trak ali disk), ki je zelo dovzeten za »bitne napake«.

Preverjanje parnosti nam pove, da se je pri prenosu zloga zgodila napaka, ne pove pa nam, kateri bit v zlogu je prenesen napačno, zato na sprejemni strani te napake ne moremo popraviti. Še več, če se pri prenosu zloga zgodita dve napaki, sploh ne bomo vedeli, da se je zgodila napaka in bomo napačno sprejeti zlog obdržali kot pravilen. Zato v primeru napačno sprejetega zloga sprejemna naprava odpošlje nazaj obvestilo o napaki, in program, ki oddaja podatke, še enkrat odpošlje napačno sprejeti zlog.

Izdelali so tudi že bolj zapletene postopke za ugotavljanje in popravljanje napak pri prenosu, ki natanko določijo, pri katerem bitu se je zgodila napaka, in jo avtomatično popravijo. Kode s korekcijo napake pa bomo obdelali v naslednjem poglavju.

Le za kontrolo
Zadnja številka v mednarodni standardni knjižni številki (International Standard Book Number — ISBN) je samo številka za kontrolo — enakovredna parnemu bitu v računalniku. Če pomnožimo prvo številko (v našem primeru 0) z 10, drugo (5) z 9 in tako naprej in seštejemo rezultate, bomo ugotovili, da je kontrolna številka izbrana tako, da je vsota vedno deljiva z 11.



Detektivski posli

Kadar prenašamo podatke od enega računalnika na drugega, vedno obstaja nevarnost, da nastane napaka. Hammingova koda zna take napake odkriti in popraviti

Vsi smo že slišali zgodbe o računalnikih, o hudih napakah, ki jih delajo — na primer 500 kopij istega dokumenta so poslali enemu naslovniku. V resnici seveda za to ne moremo kriviti računalnika, običajno se v takih primerih zmoti človek, mogoče kar pri tipkanju. Računalnik pa tak problem samo še poveča. Včasih sicer nastane napaka, ker uporabniški program ne upošteva vseh možnosti — kot v primeru, ko pošlje opomin pred tožbo za neplačan račun v vrednosti 0,00 dinarjev.

Od časa do časa pa računalnik le naredi napako, ki je ne moremo pripisati človeku in se kaže v obliki »bitnih napak«. Bitna napaka nastopi, kadar se posamezen bit v bloku podatkov spremeni iz 1 v 0 ali narobe. Bitna napaka se lahko zgodi, kadar se pokvari kak del opreme, na primer vezje pomnilnika. Zato mnogi domači računalniki ob vklopu izvedejo programsko rutino za diagnostično preverjanje.

Večina bitnih napak pa so »mehke napake« — biti se sprevržejo, čeprav je pomnilnik prestal diagnostični preskus. Hišni računalniki so zasnovani za delovanje v domačem okolju, med poletno vročino pa je povsem mogoče, da temperatura preseže dovoljeno stopnjo delovne temperature za posamezne komponente. Okvara navadno ni stalna, bitna napaka pa se lahko pokaže kot nenadna sprememba znaka na zaslonu, če pa je spremenjeni bit del pomembnega kazalca v programu, se lahko program »podre«, tako da moramo računalnik vnovič naravnati.

Bitne napake se dogajajo tudi v času močnega delovanja sonca, ko subatomski delci vstopajo v atmosfero in motijo pretok elektronov v miniaturnih elektronskih vezjih. Pri uporabi računalnikov v vojski v industrijskem krmiljenju, pri znanstvenih preskusih ali v mednarodnem bančništvu imajo lahko take napake katastrofalne posledice, zato uporabljamo vrsto postopkov za njih odkrivanje.

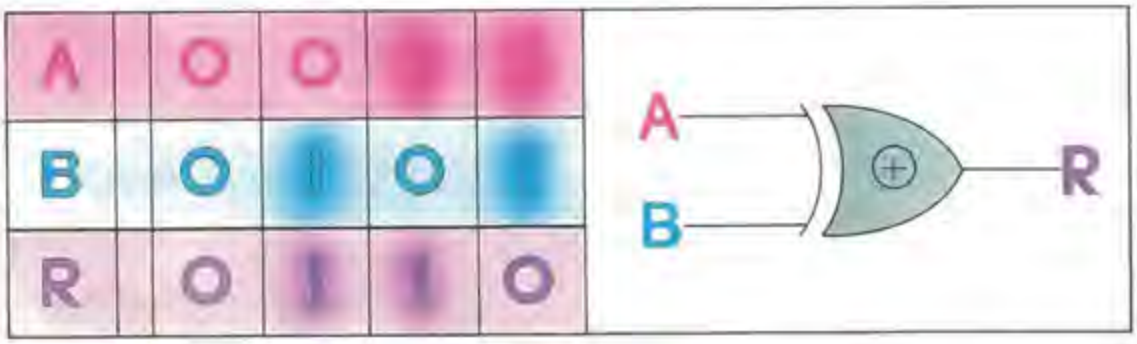
Najpreprostejši postopek je parno preverjanje (glej stran 77). Drugi postopek je preverjanje vsote (checksum), ki se na široko uporablja pri zapisu podatkov na magnetni trak ali disk. Podatke običajno obdelujemo v blokih po 128 zlogov; pri tem je zadnji, ki ga preberemo ali vpišemo, zlog za preverjanje vsote. Ta zlog predstavlja vsoto vseh drugih zlogov (od katerih ima vsak vrednost od 0 do 255) po modulu 256 — to pomeni ostanek vsote, če jo delimo z 256. Poglejmo si zgled:

Podatki: 114, 67, 83... (121 drugih vrednosti)... 36, 154, 198
Vsota teh 127 zlogov = 16673
Vsota, deljena z 256 = 65, ostane 33
Torej je zlog za preverjanje vsote 33

Vsota vseh zlogov (16673) je enaka 65 delov po 256 plus ostanek 33 — vrednost, ki se vpiše v 128. zlog kot vsota za preverjanje. Ko računalnik bere blok podatkov, hkrati računa vsoto vseh podatkov, in če na koncu ta vsota ni enaka 33, ve, da se je pri zapisu ali branju zgodila bitna napaka.

Tako pri parnem preverjanju kakor pri preverjanju vsote pa računalnik ne more vedeti, kateri bit podatkov je napačen. Če se je napaka zgodila pri prenosu, lahko računalnik, ki sprejema podatke, zahteva ponovitev oddaje določenega bloka podatkov: ob napaki pri zapisu na trak pa prav lahko ni več načina, da bi ponovno dobili pravilen podatek.

Tam, kjer so napake nedopustne, pa mora postopek, ki ga uporabljamo, omogočiti tako odkrivanje kakor



Ekskluzivni ALI
Preprosta ekskluzivna vrata ALI imajo dva vhoda in en izhod. Če je na obeh vhodih logična 0, je tudi na izhodu logična 0. Če je na kateremkoli vhodu logična 1, je tudi na izhodu 1. Če pa je logična 1 na obeh vhodih, je na izhodu 0. To zadnje stanje loči vrata ALI od (na kratko) vrat EX-ALI. Delovanje lahko predstavimo s pravilnostno tabelo. Kadar ima EX-ALI več vhodov, bo izhod na 1, kadar bo na vhodih liho število enic. S takimi vezji navadno generiramo parne bite in bite za korekcijo napak.

popravljanje napak. To nalogo opravljajo Hammingove kode, imenovane po svojem izumitelju R. W. Hammingu iz Bell Telephone Laboratories.

Vsi načini za korekcijo napake delujejo po načelu redundance. Človeški jezik vsebuje visoko stopnjo redundance — če pride do napake pri tipkanju ali če pok izbriše besedo pri telefonskem pogovoru, bomo često lahko ugotovili pomen teh besed iz sobesedila. Včasih sami posebej dodamo redundanco za uporabo v »šumnem« okolju: v radiotelefoniji uporabljamo, na primer, besede »Avala«, »Beograd«, »Cetinje« namesto črk »a«, »b« in »c«.

Vzemimo, da z našim računalnikom pošiljamo besedo dolžine x bitov, ki se sestoji iz y bitov pravih podatkov in iz redundatnih bitov (torej je $x = y + z$). V razlagi o parnosti smo imeli vrednost sedem za y in ena za z. Za Hammingovo kodo mora biti z sorazmerno večji. Vzemimo zdaj, da se lahko zgodi napaka v kateremkoli od x bitov (naših z redundantnih bitov je seveda ravno tako izpostavljenih napaki, kot y podatkovnih bitov). Če je verjetnost napake v besedi, recimo, ena proti milijon, potem je verjetnost dveh napak v besedi ena proti milijon milijonov, zato bomo to možnost zanemarili.

Ko bomo na drugi strani podatke sprejeli, bomo imeli $x + 1$ možnosti sprejema. Ali ne bomo imeli nobene napake, ali bo napaka v prvem bitu in tako naprej do zadnjega bita. Z z redundantnimi biti lahko predstavimo 2^z kombinacij, zato mora za besedo, ki je zaščitena pred napako na enem mestu, veljati:

$$2^z \geqslant y + z + 1$$

Če je y sedem (za kodo ASCII), mora biti z štiri. Če je y štiri (kot v našem zgledu ob strani), mora biti z tri. Če pa je y 16, se mora z povečati na 5. Iz tega sledi, da so Hammingove kode dosti bolj učinkovite za daljše kakor za krajše besede.

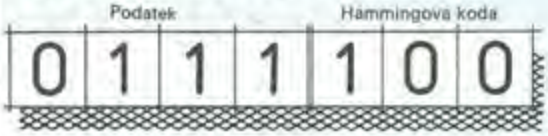
V Hammingovi kodi vsak redundantni bit deluje kot parni kontrolni bit za različno kombinacijo bitov v besedi. Če se pri prenosu katerikoli bit sprevrže, bo več ali manj kontrolnih bitov napačnih, in iz kombinacije teh bitov bomo lahko ugotovili napačni bit v besedi (glej zgled). Sprejemni računalnik lahko ta bit kratko malo sprevrže nazaj v pravilno vrednost.

Ključ do načina, kako delujejo Hammingove kode, leži v različnih kombinacijah bitov, pri katerih vsak kontrolni Hammingov bit rabi za parno preverjanje. Celotno število bitov je učinkovito razdeljeno na več različnih skupin, ki se prekrivajo in so sestavljene tako, da se noben par bitov ne pokaže v isti kombinaciji skupin. Sprejemni računalnik opravlja parno preverjanje na istih skupinah bitov, s katerimi je oddajni računalnik sestavil Hammingovo kodo. Če se je med prenosom katerikoli bit, vštevši Hammingove bite, sprevrgel, potem bo v eni ali več skupinah parna napaka. Kombinacija teh neuspelih preskusov nam pokaže na napačni bit.

Nekateri računalniki uporabljajo Hammingove kode celo za svoje interne operacije. V takem primeru je mogoče odstraniti integrirano vezje s pomnilnikom, pa bo računalnik še vedno deloval. Nekateri vojaški računalniki so prignali načelo redundance do takega ekstrema, da so vse sestavine v računalniku podvojene in se po vsaki operaciji primerjajo rezultati iz vsake polovice.

Kako deluje Hammingova koda

Recimo, da želimo prenesti te štiri bite podatkov



Dodati jim moramo tri bite Hammingove kode, enoumen vzorec bitov, ki jih izračuna računalnik tako, da izpolnjujejo naslednje pogoje:

Če pogledamo te štiri od sedmih bitov, moramo videti parno število enic

Podobno moramo tudi med temi štirimi biti videti parno število enic

In v tej skupini mora biti spet parno število enic. Da bi računalnik izračunal tri bite, ki bodo zadostili tem trem pogojem, mora rešiti sistem treh enačb.

Pa vzemimo, da se je pri prenosu tretji bit z leve pokvaril, torej preskočil iz 1 v 0

Ko računalnik pri sprejemu izvede prvega od treh preskusov, ugotovi, da je preskus negativen, saj je vidno liho število enic. To nam pove, da se je zgodila napaka, ne pa tudi, pri katerem bitu.

Tudi drugi preskus nam da negativen izid

Pri tretjem preskusu je rezultat pozitiven — vidno je parno število enic

Napačen bit nam pokaže kombinacija pozitivnih in negativnih preskusov. Če izrazimo negativen test z 1 in pozitivnega z 0 in napišemo izide preskusov v obrnjeni smeri, dobimo binarno število (v našem primeru 3), ki nam pove, kateri bit je napačen, in lahko ga popravimo tako, da ga še enkrat sprevržemo (0 in 1)

Ta postopek deluje tudi, če je napačno sprejet kateri od Hammingovih bitov. Če imajo vsi trije preskusi negativen izid, bo številka 111 pokazala, da je napačen skrajni desni bit, če pa so vsi trije pozitivni, pomeni, da napake ni bilo. Postopek odpove samo, če se zgodi napaka več ko enkrat v besedi.

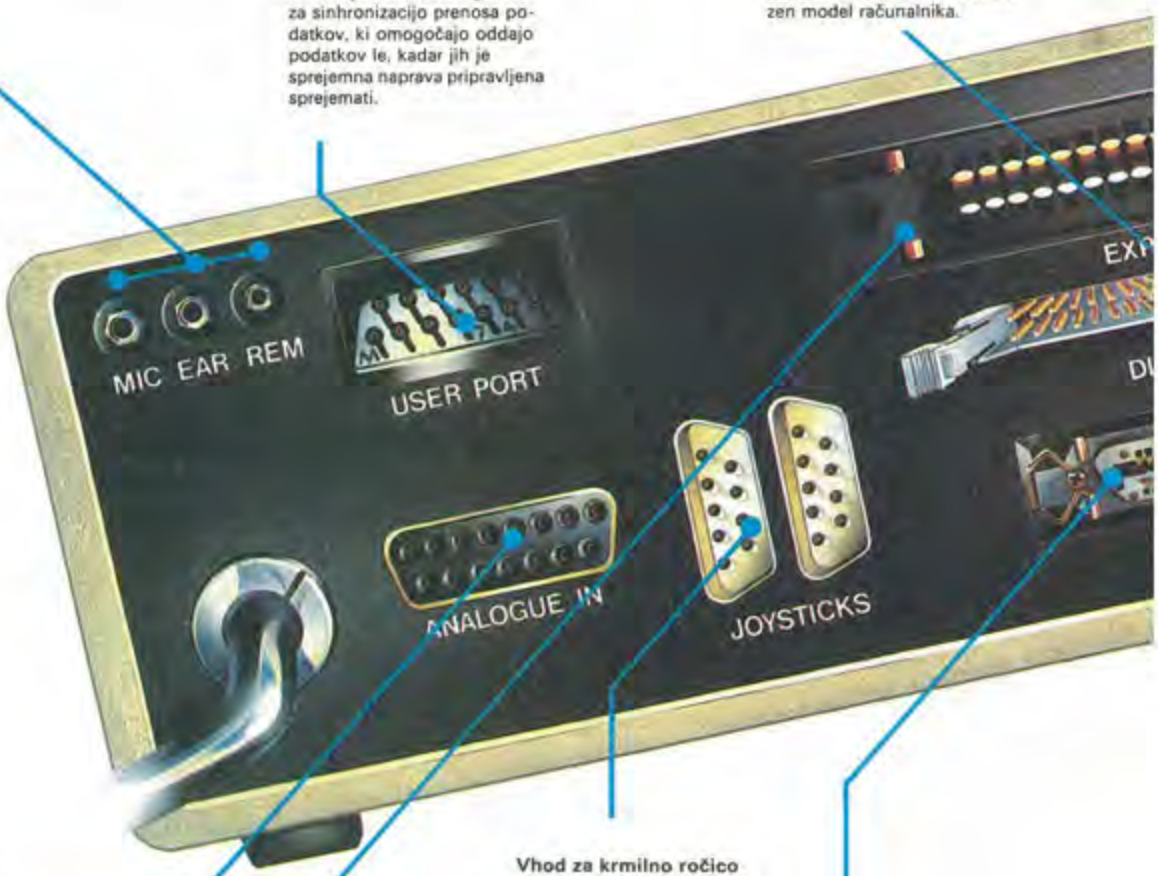
Vplivne zveze

Kakor ne moremo vtakniti kocke v okroglo luknjo, ne moremo zvezati dveh računalniških naprav, če nimata ujemajočih se vmesnikov

Vzporedna vrata
To je splošen vzporedni vmesnik za priključitev raznih perifernih naprav na računalnik. Vseh osem bitov zloga prenaša hkrati (vzporedno) preko osmih linij. Dodani so signali za sinhronizacijo prenosa podatkov, ki omogočajo oddajo podatkov le, kadar jih je sprejemna naprava pripravljena sprejemati.

Vmesnik za diskovno enoto
Diskovne enote so običajno priključene na računalnik preko vzporednega vmesnika. Splošnih standardov ni, velja pa, da je mogoče priključiti diskovno enoto samo na ustrezen model računalnika.

Kasetni vmesnik
Kasetni kanal ali vmesnik je na večini hišnih računalnikov pravi serijski vmesnik. Ker moramo podatke zapisati na navadni kasetofon, ne moremo uporabljati visokih bitnih hitrosti. Vmesnik bere iz pomnilnika podatkovne zloge, ki jih želimo zapisati na trak, in jih spreminja v serijo bitov. Ničle in enice spreminja v dva različna tona. Ko beremo podatke s traku, vmesnik dekodira ta dva tona in dobljene ničle in enice uredi v 8-bitne zloge, ki jih spravlja v pomnilnik. Kasetni vmesniki na večini hišnih računalnikov so univerzalni v tem smislu, da jih lahko priključimo na kakršenkoli kasetofon. Konektor ni standardiziran, običajno pa se uporabljajo DIN priključki in mini vtičnice.



Analogni vhod
Običajno ga najdemo samo na dražjih modelih, ki so namenjeni izobraževanju. Na analogni vhod lahko priključimo razne laboratorijske instrumente, na primer merilnike temperature ali svetlobne senzorje. Vmesnik ima običajno po enega ali več vhodov, na katere lahko pripeljemo in merimo napetosti v predpisanem območju. Uporabnik mora sam paziti, da na tak vhod ne priključi prevelike napetosti, ki lahko povzroči resno okvaro.

Priključek za dodatni pomnilnik
Ta priključek običajno vsebuje vse linije, ki prihajajo direktno od mikroprocesorja — tj. adresno, podatkovno in krmilno vodilo. Nanj lahko priključimo dodaten pomnilnik, na nekaterih računalnikih pa tudi originalne periferne enote. Ta priključek je navadno izveden kot konektor na tiskanem vezju, včasih pa kot gnezdo za konektor, kakršen je na enotah z igrami (ki so v bistvu samo dodaten bralni pomnilnik).

Vhod za krmilno ročico
Za krmilne ročice ne obstaja standardni vmesnik, čeprav so mnogi proizvajalci privzeli Atarijev standard, da jim ni treba izdelovati lastnih krmilnih ročic. Večina takih vmesnikov ima po pet aktivnih linij — po eno za vsako smer na krmilni palici in eno za sprožilec. Analogne krmilne ročice potrebujejo drugačen vmesnik, ki lahko glede na velikost napetosti določi natančen položaj krmilne ročice. Večina računalnikov ima več ko en vhod za krmilno ročico, včasih pa si več vhodov deli skupni konektor.

Vmesnik za tiskalnik
Vmesniki za tiskalnike so razmeroma dobro standardizirani po sistemu, ki ga je razvila firma Centronics Corporation, zato običajno ni težko priključiti tiskalnika s Centronicsovim vmesnikom na večino računalnikov. Tako nivoji kot funkcije signalov so standardizirani (0 V in 5 V za logično ničlo in enico). Konektorji in razpored njihovih priključkov niso standardizirani, zato je včasih treba narediti poseben kabel s konektorjem za povezavo določenega tiskalnika in računalnika.

Video vmesnik (VF)
Vsi hišni računalniki imajo priključek za video zaslon in v večini primerov je to kar črno-beli ali barvni televizor. Če ima televizor samo antenski vhod, moramo z video signalom v računalniku najprej modulirati visokofrekvenčni (VF) nosilni signal, da dobimo signal, kot ga oddajajo tv oddajniki. Ta signal televizor nato normalno sprejme, demodulira in pokaže sliko na zaslonu.

Vmesnik RGB (red, green, blue)
Video monitor potrebuje tri barvne signale — rdečega, zelenega in modrega. Če te tri signale pripeljemo v monitor ločeno, dosežemo najboljšo možno ločljivost in kakovost slike. Vmesnik RGB sporoča monitorju vse tri ločene barvne signale in še signale za vodoravno in navpično sinhronizacijo.

Video vmesnik (kompozitni)
Nekateri televizorji in monitorji imajo kompozitni ali svideoe vhod, ki se ogne sprejemniku in demodulatorju in tako omogoča boljšo kakovost slike. Na kompozitnem video izhodu v računalniku so pomešani vsi standardni video signali (luminanca, krominanca, sinhronizacijski signali itd.), vendar jih ne peljemo na video VF modulator, ampak naravnost v televizor ali monitor.

Serijski vmesnik
Drugače kakor mnogi drugi vmesniki je RS232 serijski vmesnik, določen po standardu Electrical Industries Association. Ta standard predpisuje tip konektorja (»miniaturni D« konektor s 25 priključki), signale na posameznih priključkih in njihove nivoje. Na žalost se le redki proizvajalci natanko držijo standarda in včasih je težko priključiti neko napravo z vmesnikom RS232 na računalnik. Od mnogih priključkov na konektorju se navadno uporabljajo le naslednji: priključek 2 za oddajo serijskih podatkov iz računalnika, priključek 3 za sprejem serijskih podatkov iz periferne naprave in priključek 7 kot masa (ozemljitev). Tako oddajna kakor sprejemna stran morata biti nastavljeni na isto baudno hitrost in isti format podatkov.

Vmesnik IEEE
Vmesnik IEEE je univerzalni vzporedni vmesnik, ki sloni na Hewlett-Packard Interface Busu (HPIB) in je sprejet kot standard pri ameriškem Institute of Electrical and Electronic Engineers. Standard je dobro opredeljen, tako fizikalno kakor električno. Drugače kakor pri drugih podatkovnih vmesnikih (npr. Centronics ali RS232), na katere lahko priključimo hkrati samo po eno periferno napravo, lahko na IEEE vmesnik priključimo do 15 naprav (z računalnikom vred) hkrati. Med napravami, ki imajo vmesnik IEEE, nahajamo tiskalnike, diskovne enote, risalnike, signalne generatorje, voltmetre in druge merilne instrumente. Zaradi svoje prikladnosti pri uporabi merilnih in preskusnih naprav je vmesnik IEEE zelo priljubljen v laboratorijih in industrijskih kompleksih. Le nekaj današnjih hišnih računalnikov ima vdelan vmesnik IEEE in nekaj od teh ima namesto standardnega kar robni konektor na tiskanem vezju.

Površno gledano se izraz »vmesnik« (interface) uporablja za konektor ali vtičnico, kamor priključimo na računalnik razne dodatne naprave. Dejansko pa se ta izraz nanaša na celotno vezje in pripadajočo programsko opremo, ki omogoča zvezo med dvema računalniškima napravama.

V samem računalniku se vrši pretok podatkov preko »vodil« — skupine vzporednih vodnikov, od katerih vsak prenaša en sam bit. Večina mikroračunalnikov ima po tri vodila: 8-bitno podatkovno vodilo, 16-bitno adresno ali naslovno vodilo in kontrolno vodilo, ki običajno prenaša 5 do 12 bitov in označuje trenutno stanje CPU. Nekateri kontrolni signali povedo pomnilniku in perifernim vezjem, ali želi CPU podatke vzeti (brati) ali odložiti (vpisati). Drugi spremenijo signale, ki prihajajo v CPU od zunaj, in ji na primer povedo, da ima periferna naprava pripravljene podatke in zahteva od CPU pripravljenost.

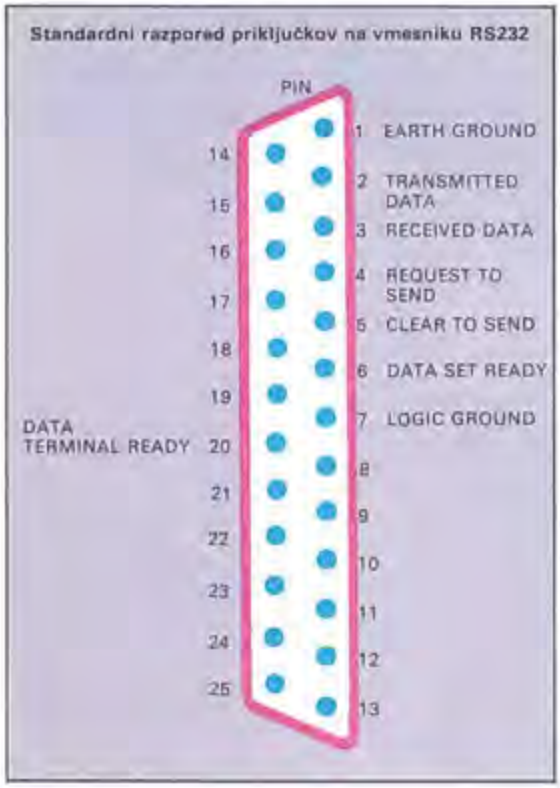
Interno računalnik običajno obdeluje podatke, ki se sestojijo iz 8 ali 16 bitov. Tako na primer CPU, če želi prebrati podatek iz pomnilniške lokacije 65535 (ali FFFF, izraženo heksadecimalno), postavi na vse linije

naslovnega vodila logično enico in s tem označi to lokacijo. Če je vsebina pomnilnika na tej lokaciji na primer 128 (heksadecimalno B6), se bo na podatkovnem vodilu znašlo osem bitov 10110110.

Kadar prenašamo podatke na opisani način, imenujemo prenos »vzporeden«. Tudi veliko perifernih naprav oddaja in sprejema podatke v vzporedni obliki. Vmesniki, predvideni za periferne naprave tega tipa, se imenujejo »vzporedni vmesniki« in večina računalnikov ima vsaj po eno vtičnico ali konektor, namenjen prav »vzporednim« napravam.

Vendar vse periferne naprave ne morejo sprejemati in oddajati podatkov v vzporedni obliki. Nekatere uporabljajo eno samo linijo, s katero komunicirajo z računalnikom. Interno podatke še vedno uporabljajo kot 8- ali 16-bitne besede, vendar sprejemajo ali oddajajo hkrati samo po en bit iz besede. Začnejo z »najmanj pomembnim« bitom in končajo z »najbolj pomembnim«. Vsaka beseda se tako v vzporedno-serijskem pretvorniku razdeli na vrsto bitov, ki jih naprava pošilja drugega za drugim, in se na drugi strani spet sestavijo v zlog, uporabljajoč vzporedno besedo v serijsko-vzporednem pretvorniku.

Tako serijski kakor vzporedni vmesniki lahko prenašajo podatke v računalnik ali iz njega. Računalniki



imajo običajno še druge vrste vmesnikov, ki pošiljajo podatke vedno le navzven (na primer tv vmesnik), ali pa jih vedno samo sprejemajo (na primer vhod za krmilno ročico).

Serijski vmesniki

Obstaja serijski vmesnik, znan kot vmesnik RS232, ki ima natanko določene vse podrobnosti v zvezi z nivoji signalov in razporedom priključkov. Določen je celo tip konektorja. Na žalost se standarda vsi natančno ne držijo, zato je včasih izvedba serijske povezave nekoliko težavna. RS232 je »dvosmerni« serijski vmesnik z enim oddajnim in enim sprejemnim vodnikom.

Podatke pošiljamo bit za bitom preko oddajnega (transmitted data) priključka in sprejemamo preko sprejemnega (received data) priključka. Podatke lahko sprejemamo in oddajamo v raznih standardnih formatih, seveda pa morata obe napravi uporabljati isti format.

Ker oddajamo besede bit za bitom, mora program, ki oddaja te podatke, imeti možnost označiti, kdaj je začel oddajati prvi bit in kdaj je prenehal oddajati zadnjega. Po najbolj razširjenem dogovoru uporabljamo en »startni bit« (logična ničla), ki mu sledi osem bitov podatka in na koncu en »stop bit« (logična enica).

Hitrost oddaje moramo določiti vnaprej, sicer lahko vzorec enic in ničel, ki predstavljajo podatke, napačno sprejmemo. Ta hitrost oddaje je znana kot »baudna hitrost« in ima ime po francoskem izumitelju iz 19. stoletja Baudotu. Baudne hitrosti se gibljejo med 75 baudi in 19200 baudi. Te številke se nanašajo na število bitov, prenesenih v eni sekundi, in ker običajno prenašamo 10 bitov za vsak znak (skupaj s startnim in stop bitom), je hitrost prenosa znakov desetinka baudne hitrosti.

Vzporedni vmesniki

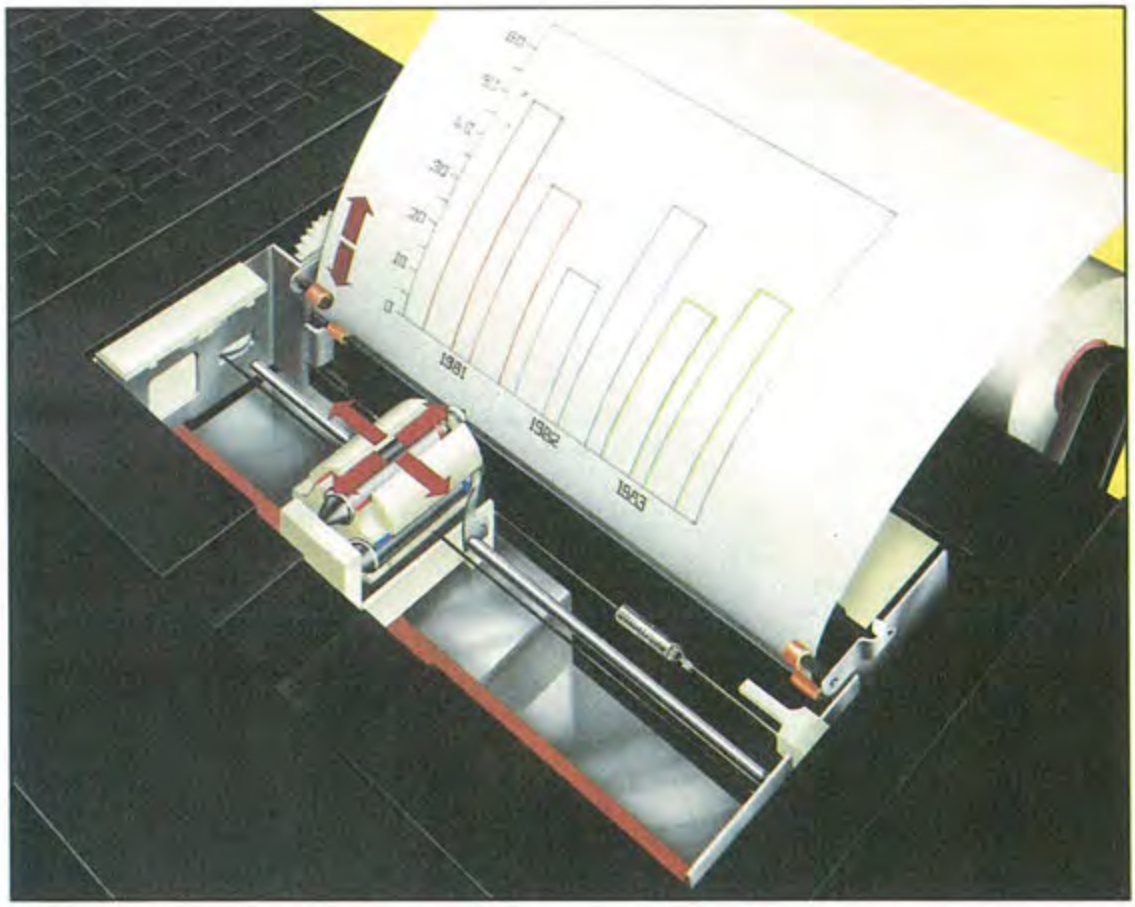
Vzporedni vmesniki sprejemajo ali oddajajo hkrati celo besedo, potrebujejo pa poleg osmih »podatkovnih linij« še druge signale, ki računalniku in perifernim napravam povedo, kdaj je mogoča oddaja podatkov in kdaj ne. Najbolj znan tip vzporednega vmesnika je vmesnik »Centronics« (imenovan po ameriškem proizvajalcu tiskalnikov Centronics Corporation), vendar se tega standarda večina ne drži natančno. Proizvajalci uporabljajo zelo različne tipe konektorjev in razpored priključkov. Večina vmesnikov Centronics pa ima vsaj naslednje signale:

DATA 0 do DATA 7	Osem linij za prenos osmih bitov zloga, ki ga prenašamo.
ADK	Vhodni signal na računalniku, ki pove, da je sprejemnik pripravljen za sprejem podatkov.
GND	Ozemljitvena žila, ki je skupna referenčna točka za 0 V za računalnik in periferno napravo.
BUSY	Signal iz periferne naprave, ki pove računalniku, da periferna naprava ni pripravljena na sprejem.
STROBE	Izhodni signal iz računalnika, ki pove periferni napravi, da je podatek iz računalnika pripravljen in ga lahko prebere.

Veliko drugih naprav poleg tiskalnikov ima kvazistandardne vmesnike Centronics in za njihovo priključitev na računalnik običajno ne potrebujemo nič drugega kakor poseben kabel za povezavo, ki si ga lahko naredimo tudi sami. Velja pa pravilo, da za krmiljenje perifernih naprav v tem primeru niso potrebne nobene spremembe programske opreme.

4. poglavje

Računalnik in njegova oprema



S samim računalnikom si ne moremo pomagati. Potrebujemo opremo, s katero računalnik spremenimo v delovno orodje.

Tesen dotik

Tipkovnice so si na prvi pogled močno podobne. Vendar so nekatere dosti boljše od drugih in delujejo na različne načine

Računalnikova tipkovnica je pomemben del sistema. Konec koncev je to pot, po kateri komuniciramo z računalnikom. Tipkovnici moramo posvetiti ravno toliko pozornosti kakor velikosti pomnilnika ali kvaliteti grafike.

Mikroračunalniki so podedovali od pisalnih strojev Qwerty tipkovnico, ki se tako imenuje zato, ker prvih šest črk na njej daje QWERTY. V začetku tega stoletja so bile črke na tipkah razporejene tako, da so zavirale hitro tipkanje, da se ne bi poškodoval občutljivi mehanizem stroja!

V zgodnjih petdesetih letih, ko so računalniki prvič prišli v komercialno uporabo, je bila razporeditev QWERTY standardna za strojepiske in je postala standardna tudi za računalniške vhodne enote. Danes se lastnik mikroračunalnika zaplete s sistemom QWERTY, ki je dober za izurjene strojepiske, novinec pa ga včasih težko obvlada. Ko so računalniki stali še stotisoče funtov, je bila cena mehaničnih delov zanemarljiva. Razvoj mikroprocesorskih tehnologij pa je dramatično znižal ceno mikroračunalniških sestavin.

V času, ko se je pojavil Sinclair ZX81, je lahko bila tipkovnica, podobna tistim na pisalnem stroju, velik del proizvodne cene mikroračunalnika. Mehanična premična tipkovnica, kakor na primer na Dragonu ali BBC Microcomputerju, uporablja prava stikala pod tipkami (glej sliko na nasprotni strani). Ko pritisnemo tipko, sklenemo kontakte v tipki. Taka stikala imajo veliko sestavnih delov in občutno povečujejo ceno tipkovnice. BBC Microcomputer ima 74 tipk in nekateri modeli še več.

Ena od rešitev problema je bil nov, cenen tip tipkovnice. »Senzorsko« tipkovnico na ZX81 je porodil premislek, da bo večina ljudi, ki bodo kupili računalnik, predvsem igrala in pisala kratke programe. Te dejavnosti zahtevajo zelo malo dela na tipkovnici in bilo je videti logično, da bodo morebitni uporabniki računalnika pripravljeni sprijazniti se z nižjo kvaliteto tipkovnice. Ko so izdelovalci žrtvovali prednosti prave tipkovnice, so lahko precej prihranili.

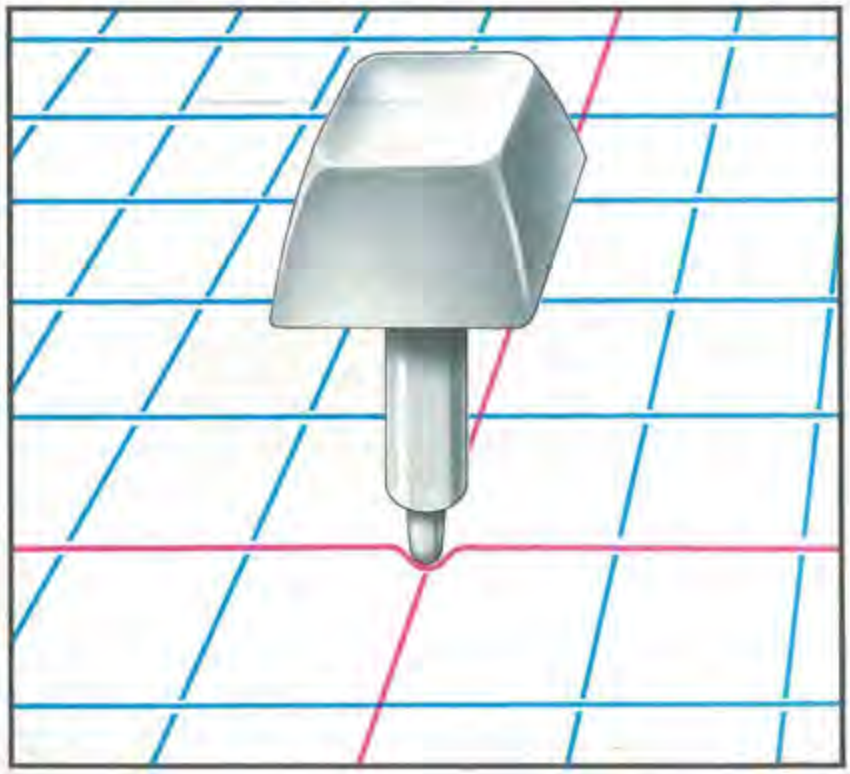
ZX81 je bil razvit s senzorsko tipkovnico, pri kateri so se znebili večine drobnih sestavnih delov. To je znižalo ceno modela, ni pa prineslo končne rešitve. Problem pri senzorskih tipkovnicah je v tem, da ne dajejo pravega občutka pri tipkanju (nikoli nismo prepričani, da je računalnik sprejel to, kar smo ravnokar pritisnili, razen če gledamo na zaslon).

Sinclair je pri svojem naslednjem izdelku, Spectrumu, vpeljal membransko tipkovnico (glej sliko). Ta vrsta tipkovnice pomeni izboljšanje, še vedno pa ne daje pravega občutka pri tipkanju.

Precej sorazmerno cenenih računalnikov (vštevši BBC, Dragon in Lynx) ima »poklicne« tipkovnice. Prednosti take tipkovnice postanejo jasne, kadar uporabljamo računalnik za tekstovno procesiranje. Znani občutek pisalnega stroja omogoča, da hitro opravimo veliko dela. Še ena vrsta tipkovnic leži med popolnoma pomično tipkovnico in Spectrumovo membransko izvedbo. Često jim pravimo »kalkulatorske tipkovnice« in jih najdemo pri New Brainu in Oric-1. Tipke dajejo boljši občutek, so pa majhne, stisnjene in manj primerne za slepo tipkanje kakor popolnoma gibajoče se tipkovnice. Eden od načinov, kako nadomestiti občutek na senzorskih in membranskih tipkovnicah, je uvedba kratkega piska, kadarkoli pritisnemo tipko. Ta pisk uporabniku potrdi, da je tipko v resnici pritisnil in da jo je računalnik spoznal.

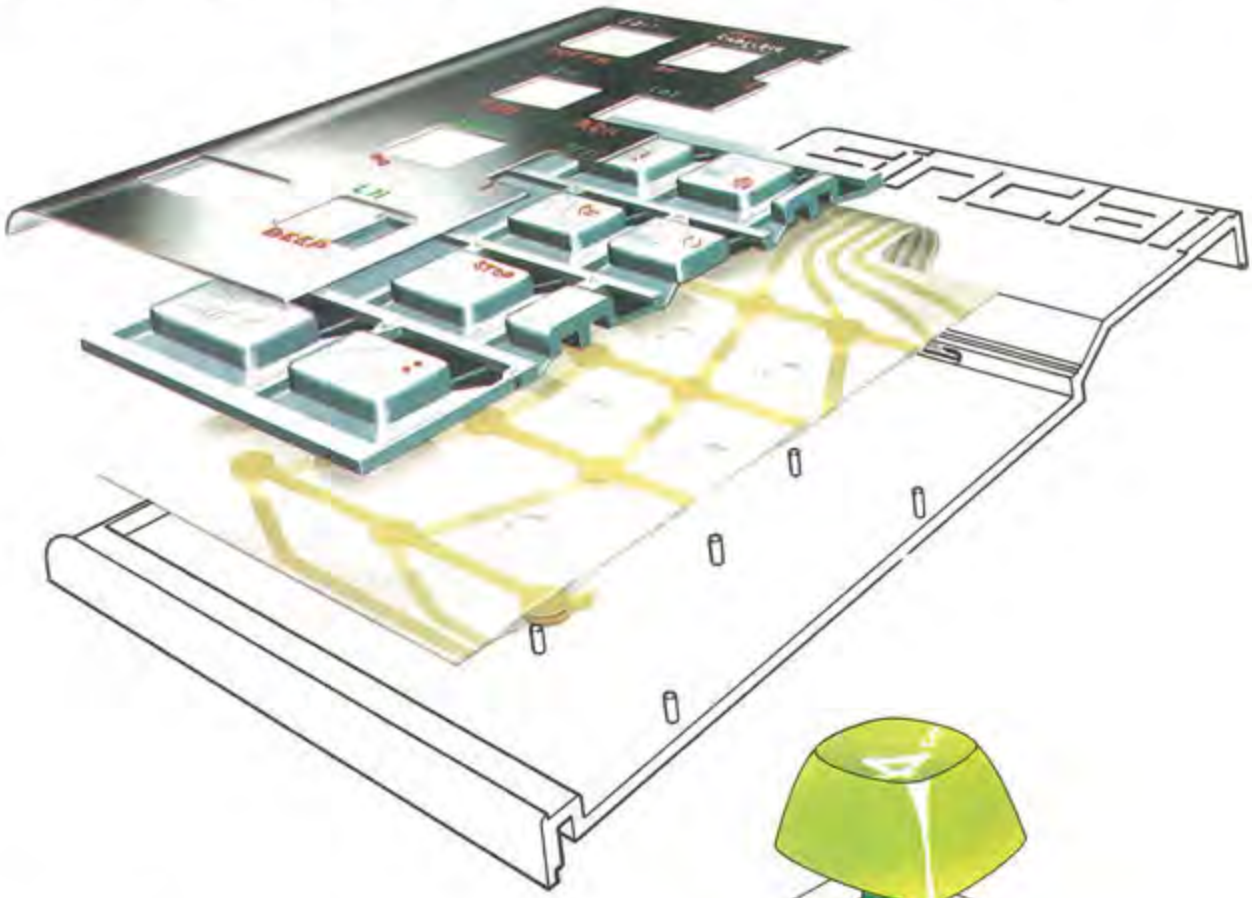
Matrika tipkovnice

Tipke na računalniku so dejansko stikala, priključena na mrežo žic. Slika prikazuje, kako se ob pritisku na tipko stakneta dve žici v mreži. Za vsako tipko se sklene samo en par žic. Vsaka tipka tako naredi enkraten stik na mreži in omogoči računalniku, da ugotovi, katera tipka je bila pritisnjena.



Sinclairova tipkovnica

Sinclair je pri Spectrumu nadaljeval umetnost izdelave tipkovnic z »membransko« tipkovnico. Preko folije s kontakti, ki predstavlja mrežo ali matriko, je položena gumijasta membrana, ulita iz enega kosa, v kateri so že izoblikovane izbokline za tipke. Ko pritisnemo na tipko, pritisne nosek pod tipko na kontakt in ga sklene. Računalnik preverja, kateri par kontaktov je bil sklenjen in kateri tipki pripada. Kontakte drži narazen mehurček zraka, zaprt med dvema plastičnima folijama. Tipko vrne v mirujočo lego elastični gumijasti rob tipke. Ta izvirni način izdelave tipkovnice ima gotovo svoje pomanjkljivosti, omogoča pa nizko proizvodno ceno tipkovnice in kupci imajo od tega direktno korist.



Tipka s stikalom

Tipka s stikalom ima običajno par kontaktov. Normalno sta razklenjena in tok ne teče preko njiju. Ko pritisnemo na tipko, se plastična zagozda med njima (na sliki v vijoličasti barvi) pomakne navzdol in omogoči sklenitev kontaktov. Vzmet v tipki omogoča vrnitev tipke v mirujoč položaj. Ko se kontakta skleneta, steče skoznju tok in računalnik to lahko ugotovi. Priključene žice za tipke so izvedene v obliki mreže (matrike). Računalnik ob pritisku na tipko preveri, katera navpična in katera vodoravna žica prevaja tok, in tako ugotovi, katero tipko smo pritisnili. Take vrste tipk so mehanično zapletene in stroški izdelave so visoki. So pa zelo zanesljive in dajejo boljši občutek kakor gumijaste membranske tipke. Ta občutek daje odpor vzmeti, ki je vdelana v tipko. Dobro izdelana tipka daje tako dober občutek pri dotiku, da nagonsko vemo, kdaj smo jo pravilno pritisnili. Vrh tipke je tudi rahlo vdrt, da omogoča udobno tipkanje. Če nameravamo računalnik dosti uporabljati, je tak tip tipkovnice nedvomno boljša izbira.

Razvijalci ZX81 in Spectruma so vpeljali tudi nov in uporaben način, kako zmanjšati število potrebnih pritiskov na tipkovnico pri programiranju v basicu. Vsaka tipka predstavlja več ko samo črko ali številko. Z uporabo posebne funkcijske tipke v povezavi z običajnimi tipkami lahko prikličemo na zaslon celotne ukaze v basicu. Ukaz PRINT v basicu lahko na primer pokličemo s tem, da hkrati pritisnemo posebno funkcijsko tipko in tipko P. Računalnik sam ugotavlja, kdaj pomeni pritisk na določeno tipko črko in kdaj ukaz. Podobno idejo je uporabil tudi Sord pri modelu M5.

Konec tipkanja

Še do nedavnega smo morali vse ukaze vtipkati s tipkovnico. To je bil sam po sebi pogosto dolgočasen posel, posebno, če nismo spretno tipkali. Spričo dejstva, da so te ovire ljudi odvračale od uporabe (in s tem od nakupa) računalnikov, so proizvajalci našli genialno preprosto rešitev – »miško«. Miško lahko pomikamo po katerikoli ravni površini, in ko se premika, se na zaslonu računalnika premika kurzor. Tako lahko zelo hitro premaknemo kurzor na katerokoli točko zaslona, pritisnemo na gumb na miški in tako začnemo želeno operacijo. Miške lahko uporabljamo tudi v grafične namene in za barvanje površin na zaslonu.

V arhivu

Mora vsakega programerja je, da bi izgubil mojstrovino, ki jo je s trudom izdelal na računalniku. Ta problem je rešil spomin na kaseti

Zvočnik
V večini kasetofonov se zvočnik izklopi, kadar napravo priključimo na računalnik ali hi-fi sistem.

Brisalna glava
Briše vse signale, ki so bili na traku. Deluje pri snemanju.

Števec traku
Koristen pripomoček, če imamo na traku več programov.

Glava za zapis in branje
Ta glava zapisuje podatke na trak in jih tudi bere s traku.

Hobbit
Ikonov Hobbit je kasetofon, izdelan izključno za zapis in branje podatkov in programov. Je precej boljši od navadnega kasetofona, ker ga lahko upravljamo z računalnikom. Ni ga treba ročno previjati in iskati programov, Hobbit dela vse to sam. Če želimo prebrati program, napišemo samo LOAD in ime programa, in Hobbit bo v svojem seznamu poiskal to ime, previl trak na določeno mesto in prebral program.

Pogonska os
To je natančno obdelana os, ki se vrti z natanko določeno hitrostjo, kar zagotavlja stanovitno hitrost premikanja traku mimo glave.

Jakost
S tem gumbom naravnamo j kost signala pri branju in mo biti nastavljen dokaj natančr sicer bomo imeli težave.

Kadarkoli vtipkamo v računalnik program, iz knjige ali revije ali po svoji zamisli, spravljamo podatke v delovni pomnilnik (RAM) računalnika. Dokler je računalnik vključen, je program varno spravljen, ko pa ga izključimo, cel program izgine. To je precej zoprno, saj nam je vse delo pri programiranju izginilo za vedno! Če bi želeli program znova uporabiti, bi ga morali na novo vtipkati. Da bi se temu ognili, izdelovalci hišnih računalnikov v računalnik vdelajo možnost bolj trajnega shranjevanja programov.

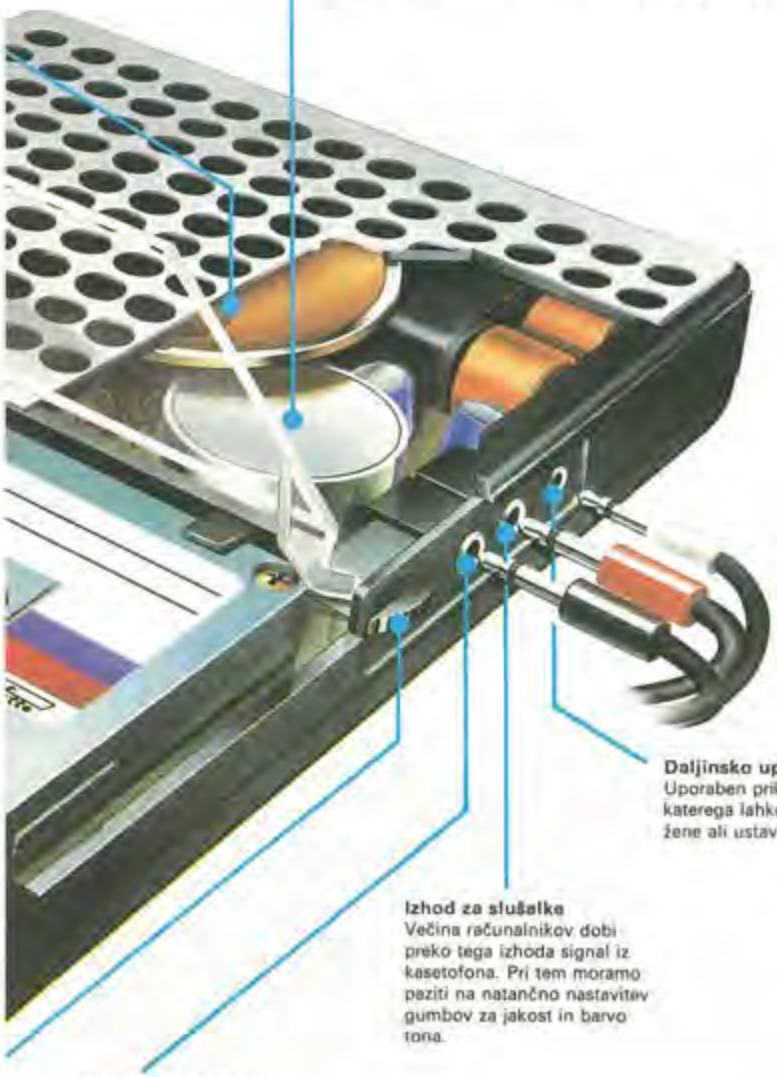
Najbolj splošen način shranjevanja programov je na magnetni trak v kaseti. Ta način so v začetku izbrali zato, ker je bil pri roki in poceni, danes pa ga najdemo praktično na vseh domačih računalnikih. Način, kako računalnik spravlja podatke na trak, je pri vsakem računalniku drugačen. Programa, ki smo ga napisali in spravili na trak s Commodorjem, ne bomo mogli naložiti v Spectrum. Postopek za trajno spravilo podatkov na trak pa je vseeno skoraj univerzalen.

Kasetofon, ki ga uporabljamo pri domačem računalniku, je najbolje prilagojen za snemanje glasbe, vemo pa, da je program v računalniku shranjen v obliki binarnih števil. Ta števila moramo spremeniti v zvočni signal tako, da bo računalnik lahko spoznal razliko med logično enico in ničlo. Najlažji način za to je, da enici priredimo en ton, ničli pa drugačnega. Običajno izberemo za logično enico ton s frekvenco 2400 Hz, za logično ničlo pa ton s frekvenco 1200 Hz.

Ko v računalnik vtipkamo ukaz SAVE, bo ta posnel na trak najprej nekaj sekund stalnega tona. Tako lahko pri reprodukciji loči med praznim trakom in delom, kjer je zapisan program. Prvi podatek za tem tonom je serija bitov, ki predstavljajo znake imena, ki smo ga dali programu. Vsak znak se sestoji iz enega zloga — osmih bitov — torej potrebujemo za zapis vsakega znaka osem tonov. Za označitev začetka in konca vsakega znaka doda računalnik na koncu vsakega znaka še en ton. Ti toni se imenujejo startni in stop biti in so vedno enaki: ali nič ali ena, odvisno od računalnika.

Tudi program je zapisan na podoben način, le da je pogosto razdeljen v segmente. Ti so običajno dolgi po 256 zlogov in često vsebujejo še dodatne informacije, ki računalniku zagotavljajo, da nalaga pravilne podatke. Način, ki se pri tem uporablja, je zelo preprost in se imenuje preverjanje vsote (checksum). Prvi zlog v segmentu vsebuje podatek o številu zlogov v segmentu, zadnji zlog pa vsebuje posebej izračunano številko, ki predstavlja vsoto vseh zlogov v segmentu po modulu 256. Ko računalnik bere podatke s kasete, primerja podatek, ki ga je prebral, s podatkom, ki ga je sam izračunal, in če se ne ujemata, nas obvesti o napaki.

Nekateri sistemi s kasetami, na primer na BBC Micro, razširijo to preverjanje tako, da imenujejo in preverjajo vsak segment posebej. Če se pri branju zgodi napaka, zadostuje, da previjemo trak za nekaj cm nazaj in poskusimo še enkrat. Nekateri drugi sistemi pa ne pokažejo niti imena programa, ki ga nalagajo.



Baudna hitrost

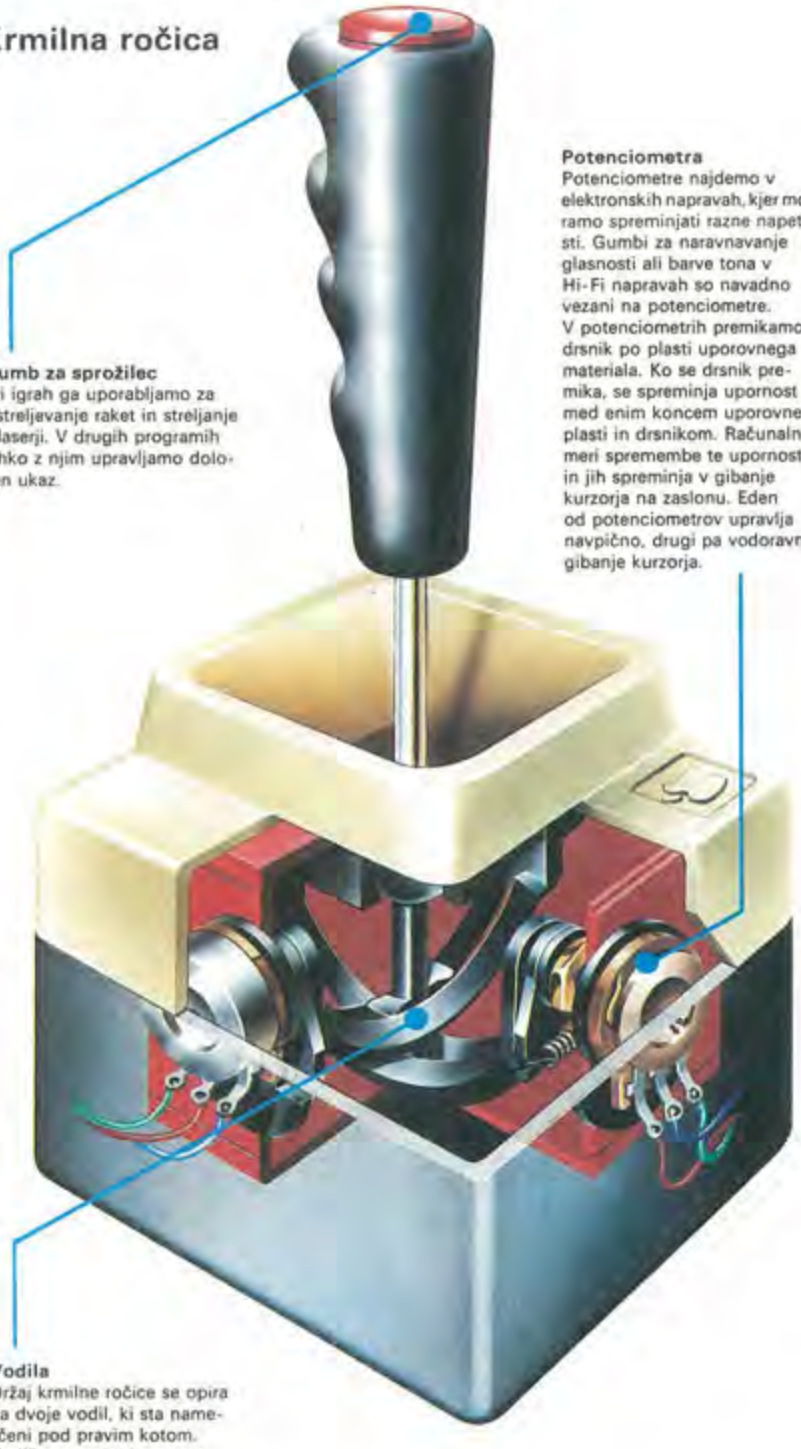
Hitrost, s katero računalnik generira in zapisuje tone na kaseto, običajno (in nepravilno) imenujemo baudna hitrost. Ime izvira iz Baudotove kode, ki so jo uporabljali pri prvih oblikah električnega brzojava in se v resnici nanaša na število sprememb signala v sekundi. Natančnejše merilo za hitrost bi bilo število bitov, ki jih posnamemo v sekundi. Čim večja je nazivna hitrost — hitrosti se gibljejo med 300 in 1200 biti na sekundo — toliko hitreje bomo program zapisali na trak in hitreje ga bomo spet nalagali v računalnik. Na žalost postajajo sistemi z višanjem hitrosti nezanesljivi; hitrost 1200 bitov na sekundo je hkrati zanesljiva in dovolj velika, da nas ne ovira pri delu. Nekateri sistemi ponujajo dve hitrosti, običajno posebej zanesljivo počasno hitrost 300 bitov na sekundo in večjo hitrost 1200 ali 2400 bitov na sekundo. Tako lahko naredimo kopije dragocenih programov za vsak primer v obeh hitrostih.

Sam kasetofon naj bi bil dobre kakovosti. Nič ni narobe, če uporabljamo audio kasete namesto posebnih kaset, malo pa moramo paziti na to, da kupujemo kasete priznanih proizvajalcev. Poleg tega se ogibajmo kaset, daljših od 60 min. Kapaciteto kasete lahko približno ocenimo tako, da delimo hitrost kasetnega vmesnika z 10. Tako dobimo število zlogov, ki jih lahko zapišemo na kaseto v eni sekundi. Tako lahko na kaseto C-60, ki traja na vsaki strani po 30 minut, zapišemo z vmesnikom, ki teče s hitrostjo 1200 bitov na sekundo, okoli 432 kb programa.

Hitri reaktorji

Nekaj dodatkov, ki bodo omogočili navdušencem za računalniške igre hitrejše delovanje

Krmilna ročica



Gumb za sprožilec
Pri igrah ga uporabljamo za izstreljevanje raket in streljanje z laserji. V drugih programih lahko z njim upravljamo določen ukaz.

Potenciometra
Potenciometre najdemo v elektronskih napravah, kjer moramo spreminjati razne napetosti. Gumbi za naravnavanje glasnosti ali barve tona v Hi-Fi napravah so navadno vezani na potenciometre. V potenciometrih premikamo drsnik po plasti uporovnega materiala. Ko se drsnik premika, se spreminja upornost med enim koncem uporovne plasti in drsnikom. Računalnik meri spremembe te upornosti in jih spreminja v gibanje kurzorja na zaslonu. Eden od potenciometrov upravlja navpično, drugi pa vodoravno gibanje kurzorja.

Vodila
Držaj krmilne ročice se opira na dvoje vodil, ki sta nameščeni pod pravim kotom. Vodili sta zvezani s potenciometroma. Če premaknemo krmilno ročico, se premakneta tudi drsnika na potenciometrih in tako spremenita električno upornost.

V računalniški igri moramo včasih pilotirati vesoljsko ladjo skozi sovražnikove vrste in z izstrelki uničiti tarčo. V tem primeru lahko prenesemo ukaze za vesoljsko ladjo z nerodne tipkovnice na krmilno ročico (joystick), ki jo držimo v rokah. Krmilna ročica je narejena po zgledu pilotske krmilne ročice v letalih.

Krmilno ročico lahko priključimo na računalnik in jo navadno uporabljamo pri arkadnih igrah. Vesoljska ladja, ali karkoli že z ročico upravljamo, se bo premikala v isti smeri kakor krmilna ročica. Običajno lahko krmilno ročico premikamo v vseh štirih smereh. Kadar premikamo ročico naprej, se bo ladja na zaslonu pomikala navzgor.

Električno gledano so v krmilni ročici nameščena štiri stikala tako, da se vključujejo le takrat, kadar pomaknemo ročico v smeri, v kateri so nameščena stikala. Tako vsako stikalo ločeno sporoča računalniku, kam smo pomaknili ročico: levo, desno, gor ali dol.

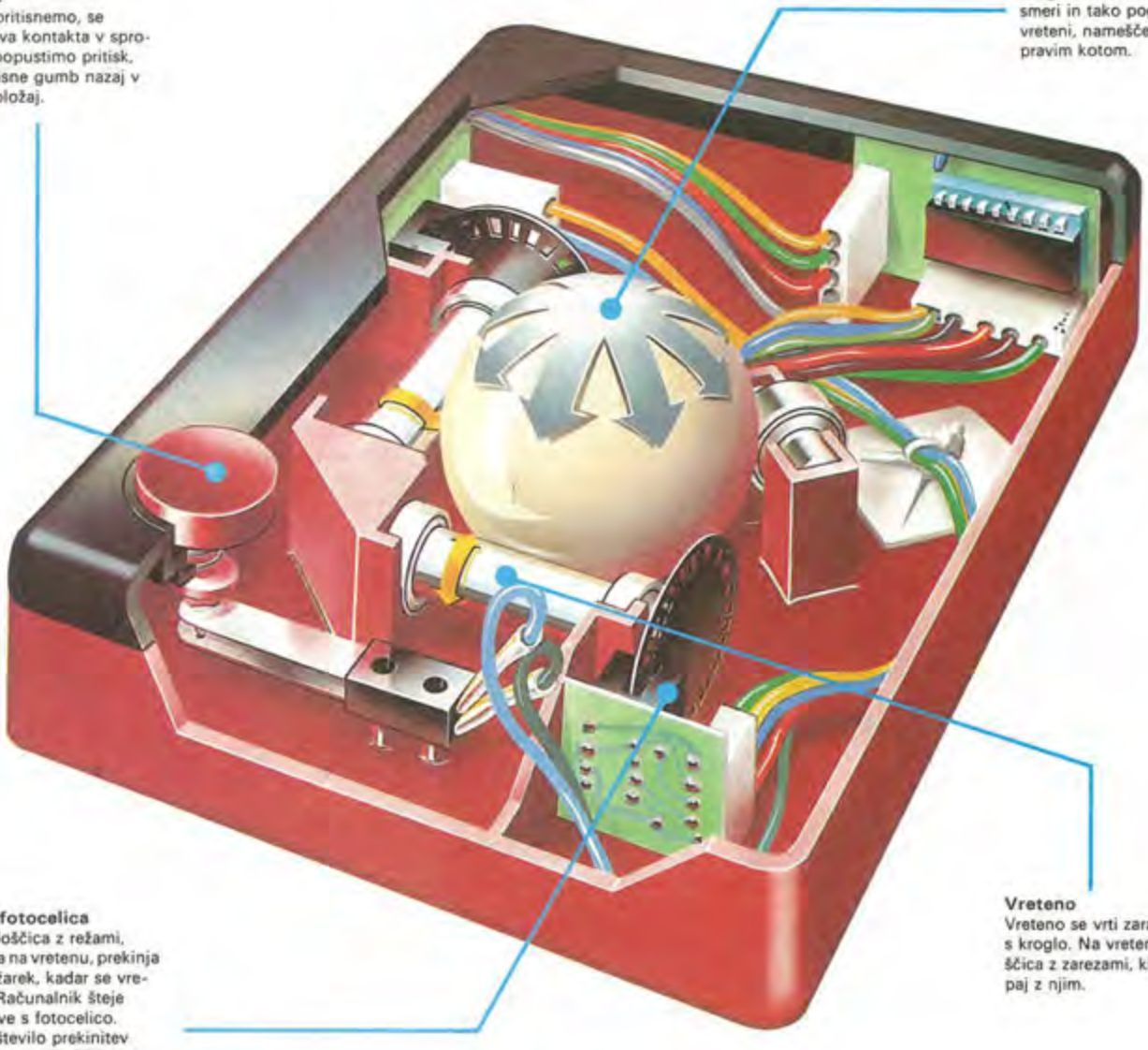
Nekatere krmilne ročice imajo tudi tipke za sprožilec. Ta tipka je lahko nameščena na ohišju ročice ali pa na vrhu same ročice in jo prožimo s palcem.

Cenejši mikroračunalniki, na primer Sinclair ZX81 in Spectrum, nimajo vedno priključka za krmilno ročico. V tem primeru moramo pač usmerjati vesoljsko ladjo s tipkovnico, ali pa dokupimo vmesnik za krmilno ročico.

Vmesnik je posebno vezje, ki omogoča priključitev krmilne ročice na računalnik. Nekateri neodvisni proizvajalci so naredili vmesnike za krmilne ročice, pa tudi v tem primeru morajo biti njihovi programi napisani tako, da je mogoče uporabljati krmilno ročico ali pa tipkovnico.

Sprožilec
Kadar ga pritisnemo, se skleneta dva kontakta v sprožilcu. Ko popustimo pritisk, vzmet potisne gumb nazaj v mirujoč položaj.

Vrteča se krogla
Kroglo lahko vrtimo v vse smeri in tako poganjamo dve vreteni, nameščeni pod pravim kotom.



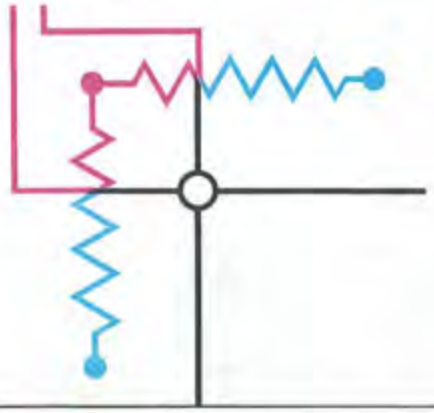
Lučka in fotocelica
Okrogla ploščica z režami, nameščena na vretenu, prekinja svetlobni žarek, kadar se vreteno vrti. Računalnik šteje te prekinitve s fotocelico. Glede na število prekinitev premika kurzor ali objekt na zaslonu. V napravi sta dva kompleta ploščic, lučk in fotocelic, eden za navpično in drugi za vodoravno gibanje.

Vreteno
Vreteno se vrti zaradi trenja s kroglo. Na vretenu je ploščica z zarezami, ki se vrti skupaj z njim.

Krmilna krogla

Predstavljajmo si, da moramo voditi kurzor na zaslonu skozi labirint. Imeti moramo možnost pomikati kurzor in ga voditi skozi hodnike labirinta. Krmilna krogla je narejena ravno za reševanje takih problemov. To je krogla v velikosti žoge za biljard, ki jo vrtimo z dlanjo. Ko vrtimo kroglo, se pomika tudi naš predmet na zaslonu v isti smeri in s hitrostjo, s katero vrtimo kroglo. V sami napravi sta pod pravim kotom nameščeni dve vreteni, na kateri se naslanja krogla. Ko vrtimo kroglo z dlanjo, prestreza eno vreteno navpično, drugo pa vodoravno vrtenje krogle. Računalnik združi obe informaciji o gibanju in tako prenese vrtenje krogle v gibanje predmeta na zaslonu.

Vezje s potenciometrom

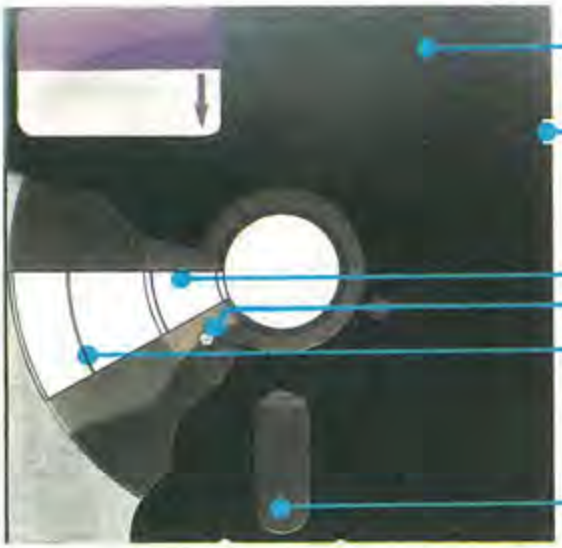


Držaj krmilne ročice je povezan z dvema spremenljivima uporoma (potenciometroma). Mehanična povezava premika kontaktno točko (drsnik) vzdolž uporovne sledi (cikcak črta na sliki). Položaj krmilne ročice tako določa upornost potenciometrov. Računalnik meri napetost na teh upornostih in tako določa položaj krmilne ročice. Nato spremeni to informacijo v položaj kurzorja na zaslonu.

Ploščati pomnilnik

Magnetni diski se vrtijo v diskovnih enotah z veliko hitrostjo in nosijo podatke, ki jih lahko beremo z računalnikom

Analogno vezje
To vezje obdeluje signale, ki prihajajo iz glave ali gredo v glavo. Digitalne signale spreminja v analogne in narobe.



Gibki disk (disketa)
Površina diskete je razdeljena na ločene pasove, imenovane sledi. Te sledi so še dodatno razdeljene na sektorje. Pri računalniku Apple II je na primer vsaka sled razdeljena na 16 sektorjev. Vsak sektor ima adresno in podatkovno polje.
Diskovni operacijski sistem doseže posamezne sektorje na disketi z adresnim poljem, ki vsebuje številko sledi in sektorja in identifikator (za kontrolo, ali uporabljamo pravo disketo). Tako lahko doseže podatek podobno kakor pri pomnilniku (z uporabo njegove adrese).

Indikator
Svetleča dioda pokaže, kdaj računalnik uporablja disketno enoto.

Pogonsko pesto
Pesto oprime plastično disketo in ju vrti v ovitku.

Hišni računalniki pozabijo vse, kar smo sprogramirali, v trenutku, ko jih izključimo. V najboljšem primeru je to lahko malo neprijetno, v najslabšem pa katastrofa, ko nam delo celotnega večera izgine v nič. Zato proizvajalci hišnih računalnikov skrbijo za možnost zapisa podatkov v trajnejši obliki. Ta je navadno v obliki kasete, ki smo jo spoznali v prejšnjem poglavju (glej stran 98).

Kadar imamo opraviti z dolgimi programi ali z izbiro kratkih programov, ki jih pogosto uporabljamo, se z iskanjem in nalaganjem programov v računalnik precej zamudimo. Za to sta dva vzroka. Prvi je ta, da moramo kasetofon pognati od začetka, če želimo najti neki program, čeprav nam pri tem kasetofoni s števcem precej pomagajo.

Drugi problem izvira iz načina zapisa podatkov na kaseti. Ti se zapisujejo v obliki tonov (višji ton za logično enico in nižji ton za ničlo). Najvišja praktična hitrost zapisa je okoli 150 zlogov na sekundo (1200 bitov na sekundo). Pri večji hitrosti začne upadati zanesljivost zapisa.

Pri kaseti C-10 potrebujemo pet minut na vsaki strani, da najdemo in naložimo program, če ne vemo, kje je. Zato resnično potrebujemo sistem, ki bo lahko našel in naložil program v nekaj sekundah. Tak pomnilni sistem je gibki disk (floppy disk) ali disketa in se lahko uporablja pri večini hišnih računalnikov. Če si predstavljamo metre magnetnega traku iz kasete, kako ga navijamo v obliki spirale na disketo, ki meri v premeru pičlih 13 cm, nam bo jasno, kako hitro lahko najdemo določeni program na disketi. Disketa je vložena v zaščitno ovojnico in tako zavarovano jo tudi vlagamo v disketno enoto.

Naloga disketne enote je, da vrti disketo in omogoča prenos podatkov na disketo in z nje v računalnik. To dela z glavo za branje in zapisovanje, ki je podobna tisti v kasetofonu, vendar je precej manjša. Ta glava se lahko pomika prečno po površini vrteče diskete, medtem ko se v kaseti trak pomika mimo glave.

Drugače kakor trak, ki vsebuje le dolg niz podatkov, je disketa »izoblikovana« v vrsto koncentričnih krogov, ki jih sistem obravnava kot majhne bloke, dolge

NE ZVIJAJ!

NE ZLAGAJ V KUPE!

ZAŠČITI PRED MAGNETI!

PAZLJIVO SHRANI!

HRANI PRI SOBNI TEMPERATURI!

Vzdrževanje diskete
Diskete so občutljive in moramo z njimi ravnati previdno. Natančno moramo upoštevati navodila proizvajalca.

Ploščati konektor
Ta konektor omogoča zanesljivo povezavo s ploščatim kablom, ki jo je mogoče odklopiti.

Ploščati kabel
Podatke pošiljamo proti disketni enoti po ploščatem kablu. Ta vsebuje žile za osembitne podatke in krmilne signale.

Pogonski motor
Motor vrti pogonsko pesto.

Glava za zapis in branje

Koračni motor in polž
Zelo natančen motor, ki pomika glavo po površini diskete.

Nosilec in mehanizem za vlaganje
Nosilec je povezan z mehanizmom na reži za vlaganje in omogoča natančno namestitev diskete na vretenu.

Glava za zapis in branje
To je močno povečana slika glave, ki zapisuje in bere podatke na površini diskete. Podobna je glavi pri kasetofonih, je pa s prostim očesom skoraj ne vidimo.

po 256 zlogov. Vsakega od teh »sektorjev« lahko posebej naslovimo.

Kadar želimo program zapisati na disketo, se glava najprej pomakne na seznam, posebno datoteko, ki deluje kot kazalo za celo disketo, in pregleda, kam lahko zapiše naš program. Če vpisuje nov program preko starega, poišče prvi sektor starega programa in začenja zapisovati na tem mestu. Če pa vpisujemo program na novo, v seznamu še ni imena zanj, zato se v seznam vpiše najprej ime, nato pa se podatki zapišejo na prvo prosto mesto na disketi.

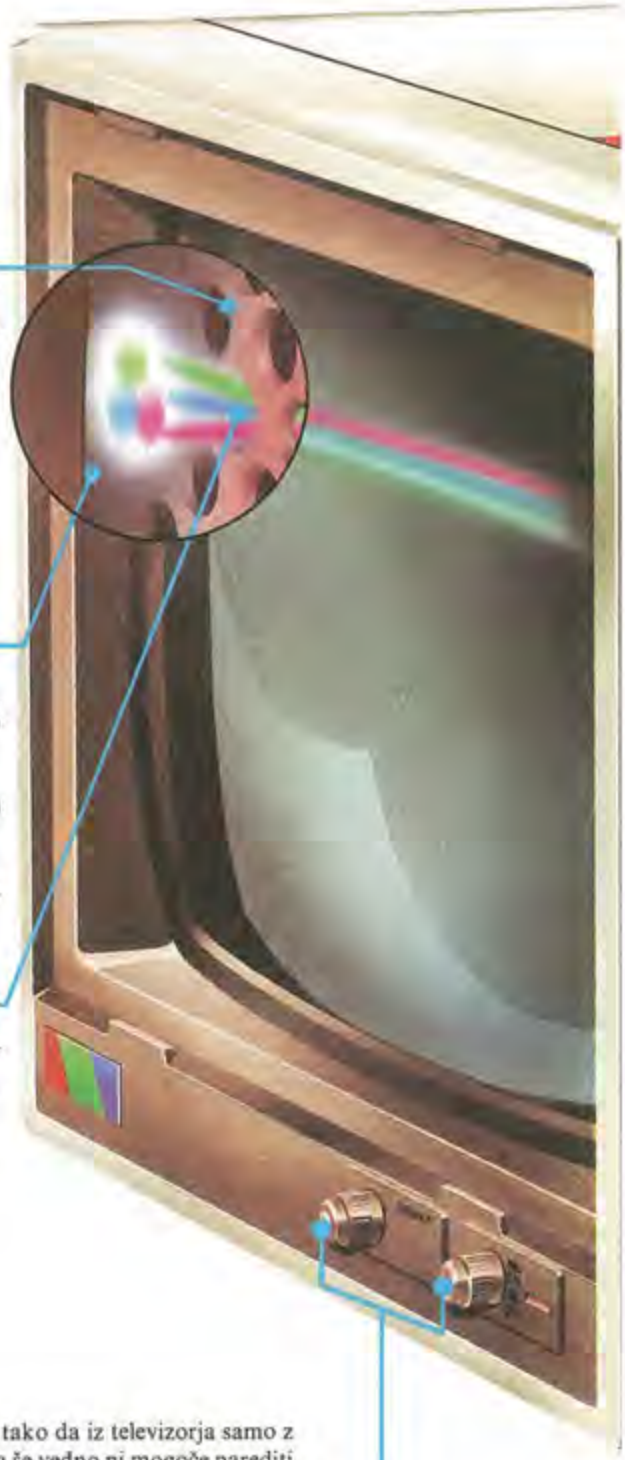
Prednosti, kakor sta velika hitrost in velika kapaciteta pomnjenja, ki jo omogoča disketa, so tudi razlog za veliko razliko v ceni med disketnim in kasetnim sistemom. Disketne enote stanejo okoli 120 funtov, kasetofon pa navadno ni dražji od 20 funtov. Poglavitni razlog za visoko ceno je velika natančnost izdelave, ki se zahteva pri disketni enoti.

Bralno-zapisovalno glavo s prostim očesom komaj vidimo, nameščena pa mora biti z natančnostjo nekaj desetink milimetra.

Podlaga mehanizma, ki premika to malo glavo, je električni koračni motor, ki se lahko zavrti za kot, ki je natančen na nekaj stopinj. Motor je s polžastim prenosom zvezan z gredjo, ki nosi glavo, in jo pomika po disketi v natančno preračunanih korakih. Disketa sama se vrti z zelo stalno hitrostjo, to dosežemo z uporabo zapletene elektronike in z montažo vseh delov na ogrodje, ki je ulito v enem kosu in zmanjšuje učinke tresljajev in temperaturnih nihanj.

V žarišču

Za domači računalnik lahko danes kupimo zaslon, ki je zanj posebej prirejen in daje kakovostno sliko za grafiko in igre



Maska
Pred zaslonom je maska z luknjicami, ki usmerjajo elektronske žarke posameznih barv natančno na luminiforje teh barv na zaslonu.

Luminifor
Barvna slika je sestavljena iz treh barv (kakor je videti na sliki). Na notranjo stran zaslona so nanesene točke kemičnih substanc (luminiforjev). Če luminiforje osvetlimo z elektronskim žarkom, se zasvetijo vsak v svoji barvi (rdeči, zeleni in modri) in tako iz barvnih točk sestavijo sliko na zaslonu. Svetlost luminiforja je odvisna od jakosti elektronskega žarka, ki ga je osvetlil.

Elektronski žarki
V slikovni cevi so trije elektronski žarki. Vsak izmed njih zbuja enega od barvnih luminiforjev in riše sliko v svoji barvi na zaslonu. Vsota vseh treh enobarvnih slik (rdeče, zelene in modre) nam da barvno sliko.

Komande
Na monitorju je več komand. Vertikalno in horizontalno sinhronizacijo lahko nastavljamo od zunaj, ravno tako svetlost slike. Regulatorji barvnega zasičenja in drugih parametrov slike pa so običajno skriti pod pokrovčkom.

Danes, ko čedalje več ljudi uporablja računalnike in druge naprave, ki potrebujejo zaslon, je začela cena specialnih monitorjev padati.

Včasih je dober barvni monitor stal šeststo ali sedemsto funtov, danes ga je mogoče kupiti že za okoli 350 funtov ali celo manj. Črno-beli monitorji so precej cenejši in stanejo nekje med 100 in 150 funti.

V času naraščajočih grafičnih zmožnosti mikroračunalnikov, med katerimi ima večina tudi barve, je zelo pametno kupiti barvni monitor.

Obstajata dve vrsti barvnih monitorjev. Prvi so znani pod imenom monitorji RGB (kratica pomeni Red-Green-Blue, to je rdeča-zelena-modra), drugi pa kot monitorji s kompozitnim video vhodom.

Pri monitorjih RGB krmili računalnik vsako od treh barv posebej; pri tem direktno vklaplja in izklaplja tri elektronske topove za tri barve. Računalnik tudi generira signale, ki skrbijo za sinhronizacijo monitorja z računalnikom.

Imamo dve vrsti sinhronizacijskih impulzov, ene za sinhronizacijo posameznih vrstic (vodoravni sinhronizacijski impulzi) in druge za sinhronizacijo celotne slike. Ko monitor nariše celotno sliko, dobi navpični sinhronizacijski impulz, ki mu pove, da mora vrniti žarek na levi gornji konec zaslona in začeti risati novo sliko.

Podoben postopek se ponavlja na koncu vsake vrstice, kjer vodoravni sinhronizacijski impulz pove monitorju, da je risanje vrstice končano in da mora premakniti žarek na levo stran zaslona in začeti risati novo vrstico.

Pri monitorju RGB vse naštete signale (rdeči, zeleni in modri signal ter vodoravne in navpične sinhronizacijske impulze) pošilja računalnik po ločenih linijah.

Kompozitni monitor je bolj soroden televizorju, ker pri njem vse signale združimo v enega samega in pošiljamo po koaksialnem kablu. Monitor vse barvne in sinhronizacijske signale spet loči in uporabi za risanje slike na zaslonu.

Monitor je v bistvu televizor brez sprejemnega dela. Monitor lahko spremenimo v televizor tako, da mu dodamo sprejemni del, in televizor v monitor tako, da mu odvzamemo sprejemnik.

Taki postopki seveda niso splošno priporočljivi, saj se v vsaki napravi, ki vsebuje televizijski zaslon, srečujemo z nevarno visokimi napetostmi. Celo profesionalci se lotevajo takih problemov z nemirom. Drugi razlog je v tem, da se vezja v televizorju in monitorju vseeno malo razlikujejo, tako da iz televizorja samo z odstranitvijo sprejemnika še vedno ni mogoče narediti dobrega monitorja.

Razlog, zakaj je bolje uporabljati monitor kakor pa televizor, je ta, da moramo za televizor s kompozitnim signalom najprej modulirati nosilni visokofrekvenčni signal, ki ga nato pripeljemo v televizor. Televizor mora ta signal najprej demodulirati, šele nato spet dobi kompozitni signal. Pri vseh teh postopkih kakovost signala precej pade in rezultat je medla in neostra slika na televizorju.

Nasprotno pa se monitor ogne modulaciji in demodulaciji in zato daje lepšo in ostrejšo sliko. Taka slika je dosti prijetnejša za oko in daje programom bolj profesionalen videz.

Visokonapetostna vezja
Ker potrebuje slikovna cev za svoje delovanje zelo visoke napetosti, potrebujemo posebna vezja, ki spreminjajo omrežno napetost 220 V na visoko napetost (od 15000 do 20000 V)

Glavna plošča z vezji
Tukaj najdemo vsa vezja, ki krmilijo tokove za jakost in odklanjanje elektronskih žarkov. Del vezja za odklanjanje žarka, ki dela z visoko frekvenco (15625 Hz), služi tudi kot napajalnik za visoko napetost za slikovno cev.

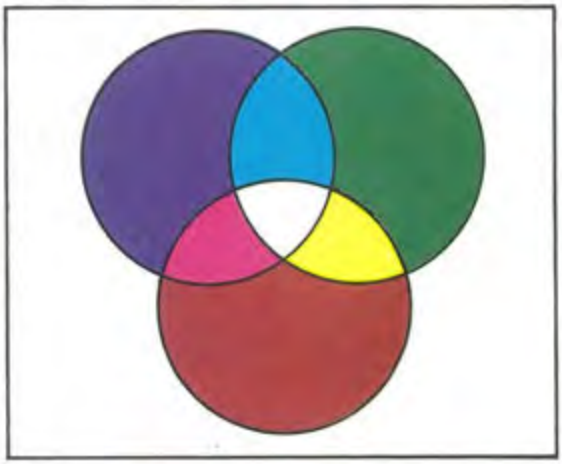
Elektronski topovi
Barvni monitor ima tako kot barvni televizor tri elektronske topove: za rdečo, zeleno in modro barvo. Ti so nameščeni v ozadju slikovne cevi tako, da elektronski žarek iz vsakega topa pada skozi luknjice v maski pred zaslonom samo na luminifor svoje barve.

Napajalnik
Slikovna cev zahteva zelo visoke napetosti in precej velike tokove, zato je omrežni transformator precej velik.

Odklonske tuljave
Odklonske tuljave so narejene iz več velikih navitij, ki proizvajajo močno magnetno polje. To polje se hitro spreminja in tako odklanja žarek po zaslonu, da riše sliko vrstico za vrstico.

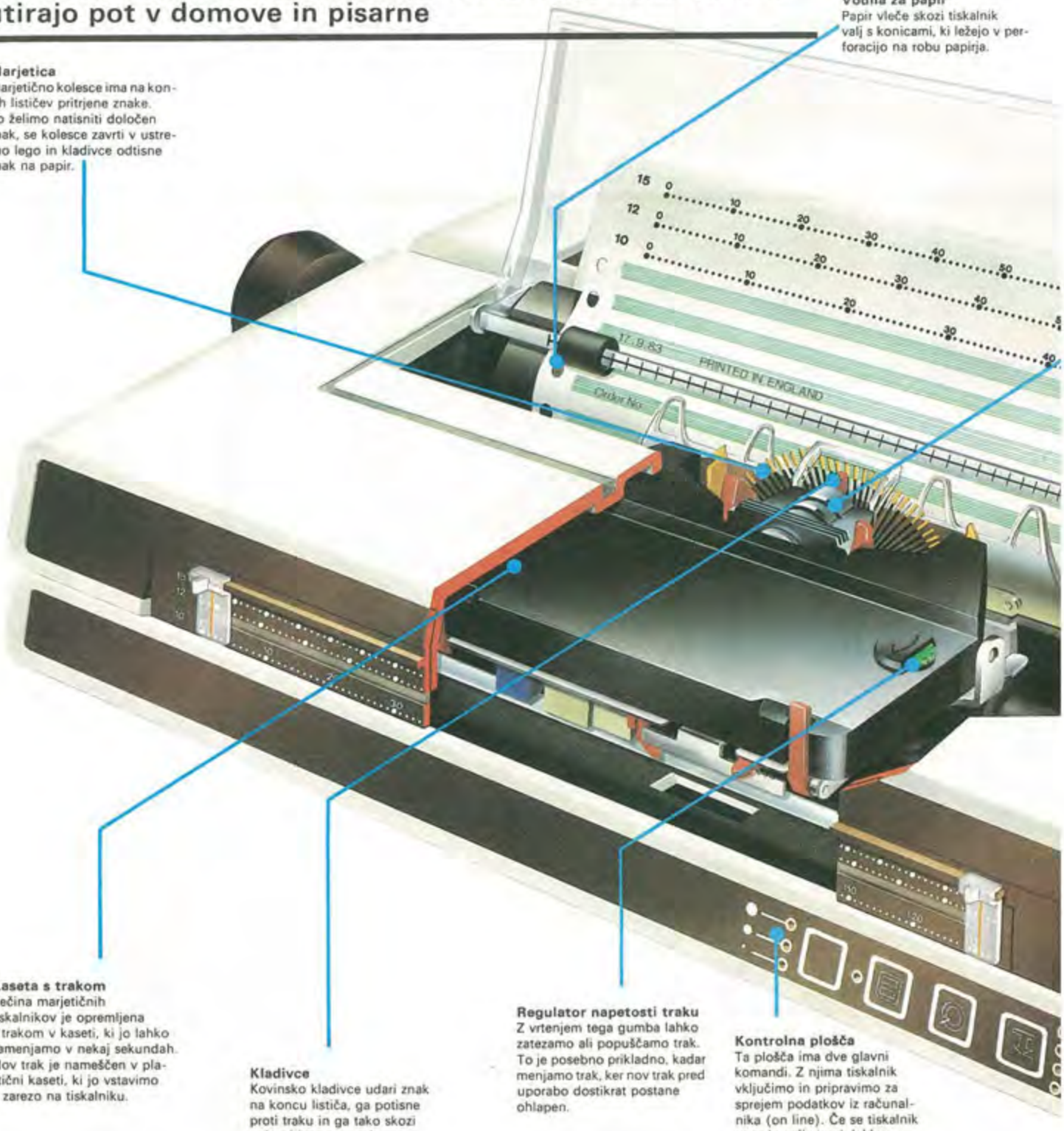
Priključek za anodo
Elektronski žarek, ki prileti iz topa, se pospešuje v električnem polju visoke napetosti. To napetost pripeljemo na cev preko oklopljenega kabla, ki ima na koncu veliko izolirano zaščitno ploščo.

Barvna lestvica
Sončno svetlobo lahko s prizmo razklonimo v mavrične barve ali spekter, ki se razteza od rdeče preko zelene do modro vijoličaste barve. Če te barve združimo z drugo prizmo, dobimo spet sončno svetlobo bele barve. Ta postopek seštevanja uporabljamo tudi pri barvnih monitorjih. S seštevanjem različnih jakosti treh glavnih barv (rdeče, zelene in modre) lahko dobimo poljubne barve.



Opremljen za tiskanje

»Marjetice«, curki črnila in matrični tiskalniki sodijo med najnovejše dosežke na področju tehnologije tiskanja, ki si utirajo pot v domove in pisarne



Vodila za papir
Papir vleče skozi tiskalnik valj s konicami, ki ležejo v perforacijo na robu papirja.

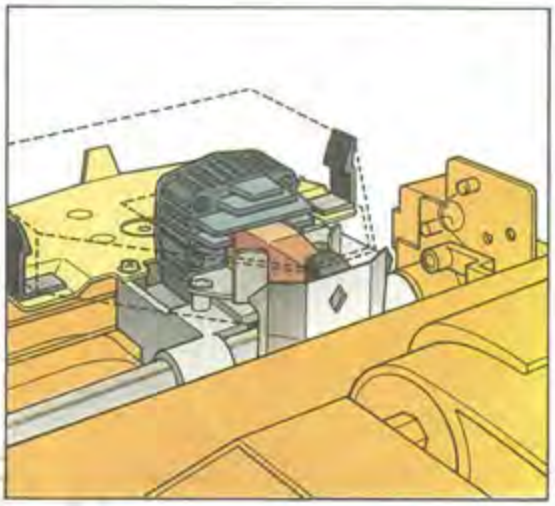
Marjetica
Marjetično kolesce ima na koncih lističev pritrjene znake. Ko želimo natisniti določen znak, se kolesce zavrti v ustrezno lego in kladivce odtisne znak na papir.

Kaseta s trakom
Večina marjetičnih tiskalnikov je opremljena s trakom v kaseti, ki jo lahko zamenjamo v nekaj sekundah. Nov trak je nameščen v plastični kaseti, ki jo vstavimo v zarezo na tiskalniku.

Kladivce
Kovinsko kladivce udari znak na koncu lističa, ga potisne proti traku in ga tako skozi trak odtisne na papir.

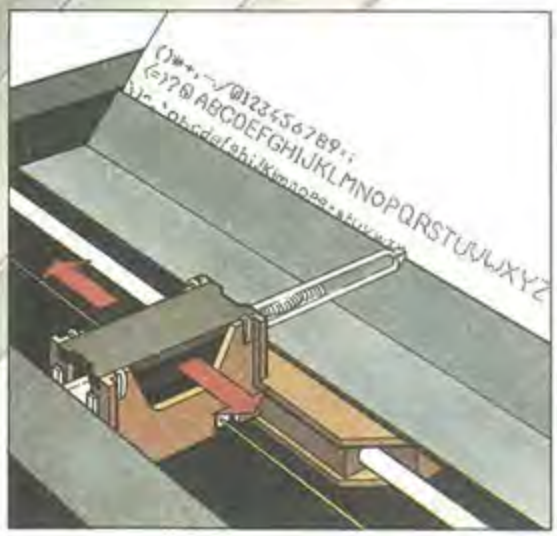
Regulator napetosti traku
Z vrtenjem tega gumba lahko zatezamo ali popuščamo trak. To je posebno prikladno, kadar menjamo trak, ker nov trak pred uporabo dostikrat postane ohlapen.

Kontrolna plošča
Ta plošča ima dve glavni komandi. Z njima tiskalnik vključimo in pripravimo za sprejem podatkov iz računalnika (on line). Če se tiskalnik ustavi sredi strani, lahko s posebno komando postavimo papir na novo stran. Z drugo komando pomaknemo papir za eno vrstico naprej.



Motor za marjetico
Ta motor obrača marjetico in postavlja znake na lističih pred kladivce, ki jih potem odtisne na papir.

Matrični tiskalnik
Matrični tiskalnik uporablja mrežo (matriko) točk, iz katerih oblikuje znake. Tiskalna glava vsebuje skupino igel, ki tolčejo po traku in na papirju puščajo sled v obliki točk. Medtem ko se glava pomika vzdolž papirja, električni impulzi prožijo igle proti papirju in oblikujejo znake.



Tiskalnik s curkom črnila
Ta tiskalnik potiska črnilo skozi drobno šobo, ki ga razprši v kapljice. Kapljice se naelektrijo in potem odklanjajo z električnim poljem med odklonskimi ploščami. Naboj zagotavlja, da kapljice oblikujejo na papirju pravi vzorec.

Do sedaj mogoče niti niste dosti premišljevali o tem, da potrebujete tiskalnik. Če ste čisto srečni, da uporabljate hišni računalnik za igranje ali za izračun domačih financ, niti ne potrebujete tiskane kopije tistega, kar delate na zaslonu.

Ko pa postajate spretnejši v delu z računalnikom, so omejitve dela brez tiskalnika očitnejše. Če pišete lastne programe, si boste gotovo želeli imeti kopije programskih seznamov. Če uporabljate računalnik za vodenje domačih računov, vam bo potreben zapis kalkulacij na papirju.

Izbira pravega tiskalnika je precej zapletena reč. Cena zanj je odvisna predvsem od hitrosti, s katero tiskalnik tiska, in kakovosti rezultatov.

Izbira tiskalnika

Za domačo uporabo prihajajo v poštev predvsem tri vrste tiskalnikov: matrični, marjetični in termični.

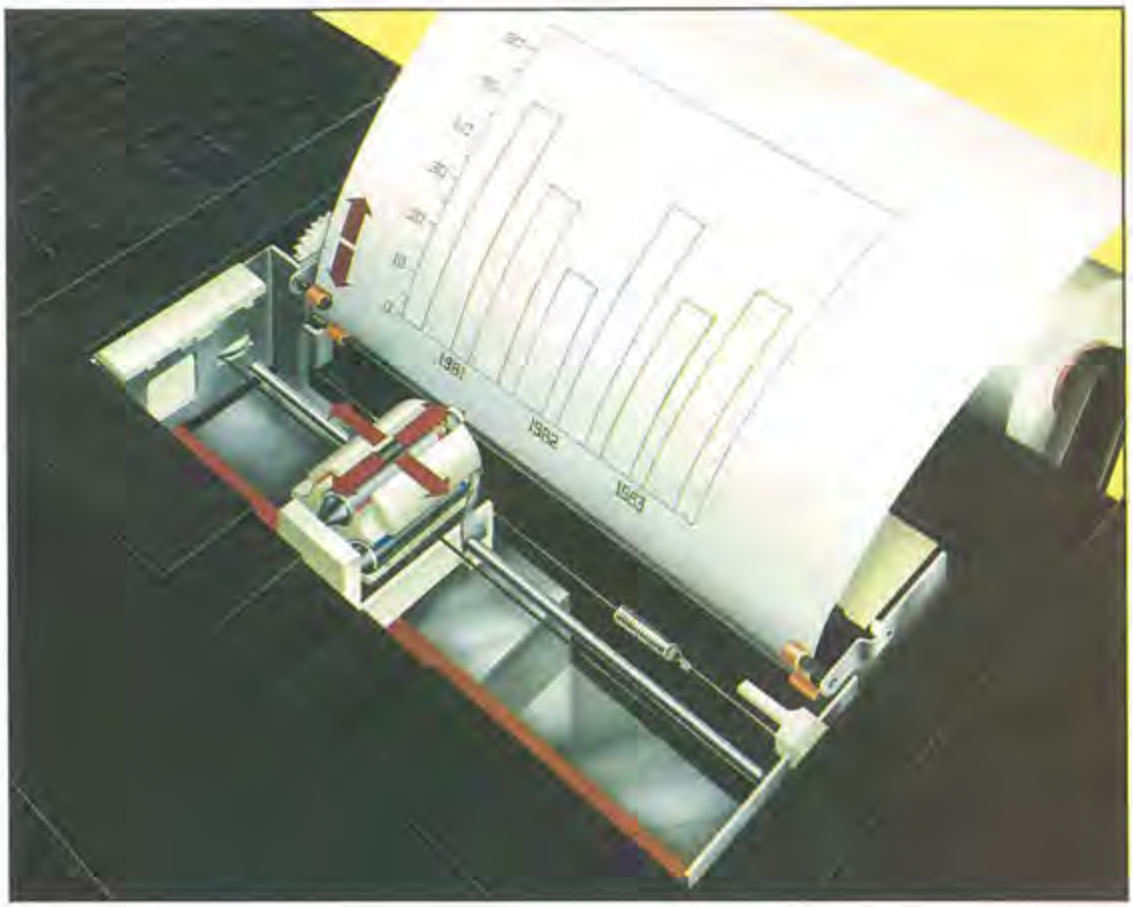
Najbolj splošna metoda tiskanja je matrična. Deluje s tiskalno glavo, ki vsebuje skupino tiskalnih igel. Znake oblikujemo z različnimi kombinacijami igel, ki udarjajo preko traku na papir. Prednost matričnih tiskalnikov je njihova velika hitrost in relativno nizka cena (okoli 300 funtov). Seveda pa je kakovost znakov, ker so sestavljeni iz točk, razmeroma slaba. Tiskalnik je tudi precej glasen.

Nekateri tiskalniki odpravljajo slabo kakovost tiska tako, da vsako točko natisnejo dvakrat ali trikrat. Medtem se tiskalna glava malce premakne, in to zapolni prostor med točkami, ki so bile najprej natisnjene.

Matrični tiskalniki so sprejemljivi, če smo zadovoljni s surovimi kopijami tistega, kar nam je računalnik naredil. Z njimi lahko rišemo tudi diagrame in grafe, ker lahko tiskalna glava tiska tako znake kakor vzorce. Seveda pa za kakovostno natisnjene dokumente, na primer za poslovna pisma, potrebujemo drugačen tiskalnik.

Za tisk, ki bo tako kakovosten kakor na pisalnem stroju, moramo uporabiti marjetični tiskalnik. Tako se imenuje zato, ker tiskalnik uporablja za tiskanje kolesce z dolgimi »lističi«, ki spominjajo na marjetico. Na koncu vsakega lističa je nameščena črka, znak ali številka. Pri tiskanju se kolesce zasuče tako, da namesti listič z ustreznim znakom pred kladivce, ki udari konec lističa z znakom preko traku na papir. Kolesce je lahko kovinsko ali plastično. Kolesca lahko tudi menjamo in tako dobimo različne vrste črk, podobno kakor pri pisalnem stroju s pisalno glavo.

Tiskalnik s kemičnim svinčnikom
Je najnovejše dognanje v tehniki tiskanja. Tiskalna glava drži štiri posebno oblikovane kemične svinčnike. Pri tiskanju se papir pomika gor in dol, glava pa levo in desno. Prednost takega sistema je ta, da ga lahko uporabljamo tudi za risanje barvnih tabel in grafov. Tudi kakovost tiskanja je boljša kakor pri matričnem tiskalniku, ker piše znake v eni potezi. Je pa ta sistem razmeroma počasen in pri dolgem tiskanju moramo pogosto menjati kemične svinčnike.



Težava pri marjetičnih tiskalnikih je ta, da so precej počasnejši od matričnih tiskalnikov in tudi precej dražji. Ravno tako niso primerni za risanje tabel in grafov, saj bi za to potrebovali preveč kolesc z množico grafičnih oblik. Marjetični tiskalnik stane najmanj 400 funtov.

Še više na cenovni lestvici so tiskalniki s curkom črnila. Slednji brizgajo na papir kapljice črnila, ki oblikujejo znake, ki jih želimo natisniti. Črnilo potiskajo skozi drobno šobo, da se razprši v drobne kapljice. Te kapljice nato letijo mimo elektrod in se naelektrijo. Končno naelektrene kapljice odklanjamo v električnem polju tako, da oblikujejo znak na papirju. Ti tiskalniki so tako hitri, da lahko natisnejo do 20 m znakov na sekundo.

Termični tiskalniki uporabljajo toplotno občutljiv papir. Tiskalna glava na določenih mestih segreje papir, ki se tam obarva črno in tako oblikuje znak. Termični tiskalniki so zelo tihi in razmeroma hitri. Eden najbolj popularnih je Applov ustrezno imenovani Silent Type model. Termični tiskalniki imajo zmerno ceno okoli 300 funtov, vendar uporabljajo samo poseben toplotno občutljivi papir, ki je dražji od navadnega, pa tudi kakovost tiskanja je slabša kakor pri marjetičnih tiskalnikih.

Pravi vmesnik

Ko se odločite za nakup tiskalnika in tudi veste, kaj od njega pričakujete v pogledu hitrosti in kakovosti, morate tudi vedeti, ali ga boste lahko priključili na svoj računalnik. Priključek na tiskalniku se mora ujemati z računalnikom. Priključek za tiskalnik običajno najdemo na zadnji strani računalnika in se imenuje »vmesnik«.

Trije najbolj znani tipi vmesnikov so Centronics, IEEE488 in RS232. Centronics je znan tudi kot vzporedni vmesnik. Vaš računalnik mora imeti vsaj eno odprtino z enim od naštetih imen.

Ker pa boleha računalniška industrija za kronično neubranostjo, boste kaj lahko ugotovili, da se računalnik in tiskalnik, ki imata isti vmesnik, ne ujemata. V tem primeru običajno vmesnika na računalniku in tiskalniku nista nastavljena na isto hitrost prenosa podatkov. Ta hitrost je znana kot baudna hitrost in je določena s hitrostjo prenosa v bitih na sekundo med računalnikom in tiskalnikom. Podatke lahko pošiljamo tiskalniku na dva načina: ali si sledijo posamezni biti po isti liniji, kot npr. pri vmesniku RS232 (serijski prenos), ali pa jih prenašamo po več linijah (za vsak bit v zlogu ena linija), kot pri vmesnikih IEEE488 in Centronics (vzporedni prenos).

Papir lahko teče skozi tiskalnike predvsem na dva načina. Pri prvem vlagamo v tiskalnik vsak list posebej kakor pri pisalnem stroju in ga skozi tiskalnik potiskajo valji. Lahko pa ima papir ob straneh luknjice, v katere nalegajo čepki na valju tiskalnika in tako pomikajo papir podobno, kakor filmska kamera pomika perforiran film. Slednji način je primernejši, kadar želimo, da se tiskalnik sam oskrbuje s papirjem. Seveda pa moramo liste z že natisnjeno glavo vlagati vsakega posebej.

Kadar torej kupujete tiskalnik, dobro premislite, za kaj ga želite uporabljati, in izberite najboljši model, ki si ga lahko privoščite za svoj računalnik.

Na dveh kolesih

Robotska tehnika se hitro razvija. Talne robote, kakršen je BBC Buggy, je mogoče sprogramirati tako, da odkrivajo ovire

Roboti na kolesih in želve niso samo učila, ampak tudi prav zabavne igrače. Načela za upravljanje naprave, kakor je BBC Buggy, so enaka kakor pri pravih industrijskih robotih. Čeprav za zdaj še ne opravljajo gospodinjskih del in drugih hišnih opravil, bi lahko postali naslednja generacija gospodinjskih pomočnic.

Roboti morajo biti zmožni zavzeti natančno določeno lego glede na svojo okolico. Navadno jih poganjajo koračni motorji, ki se — v tem se razlikujejo od običajnih motorjev — ne zavrtijo, ko jih priključimo na električno napetost. Os koračnega motorja se vsakič, ko pošljemo v motor električni impulz, zavrti za vnaprej določen delček celega obrata. Število impulzov, potrebnih za cel obrat motorja, je odvisno od njegove zgradbe. Tudi smer vrtenja moramo določati. Talnega robota ali želvo je z računalniškim vodenjem posameznih motorjev mogoče tako premakniti v poljubno smer za natančno določeno razdaljo. Talnega robota na primer lahko zasučemo na mestu preprosto s tem, da zavrtimo njegovi kolesi v nasprotnih smereh.

Za gibanje robota pa je seveda enako pomembno tudi to, da nadzornemu računalniku poroča o ovirah, na katere naleti. Zadevanje obnje običajno »odkriva« z odbijači, ki jih ima nameščene po obodu in ki so z mikrostikali priključeni na računalnikova vhodna vrata. Sklenitev ali razklenitev stikala povzroči nastavitev ustreznega bita v vhodnih vratih na »1« ali »0«.

Od robota pa včasih zahtevamo tudi druge informacije. Pogosto mora biti zmožen voziti vzdolž bele črte na temnih tleh. Ta zmožnost je zasnovana na zaznavanju odbite svetlobe, ki jo robotova žarnica oddaja v tla. Količina odbite svetlobe se spreminja glede na velikost bele črte na površini, ki jo svetlobno tipalo robota zaznava. Ta količina pa ni digitalna, temveč zvezna. Mikroračunalnik BBC je opremljen z analognim vhodom, ki omogoča neposredno priključitev zveznega tipala. Za večino drugih sistemov pa je treba zvezni signal najprej spremeniti v digitalno obliko, šele potem ga lahko pošljemo nazaj računalniku.

Posebno zanimiva robotska uporaba tipala te vrste je bralnik ročičnih kod (bar code reader). Robot bi lahko z njim med iskanjem zahtevanega predmeta pre-

Na poti

Ta želvica ZEAKER je v resnici deloma talni robot, deloma pa za določeno nalogo razvita želva, ki ji je zaradi vdelanih tipal za trke mogoče dati neko stopnjo »pametic. ZEAKER je zmožen gibanja naprej, nazaj, levo in desno ter lahko vzdiguje in spušča pisalo. Vodenje te želve je močno poenostavljeno z jezikom »snail logo«, ki je izpeljan iz jezika logo in omogoča neposredno uporabo ukazov NAPREJ, NAZAJ in podobnih. Želvo lahko priključimo na računalnik BBC Micro neposredno, za druge mikroračunalnike pa je potreben vmesnik.

biral kode predmetov v skladišču. Sistem BBC Buggy ima demonstracijsko programsko opremo, ki uporablja bralnik ročične kode za izbor glasbe.

Drugi analogni signali, ki bi jim robot morda hotel slediti, so svetloba, zvok in magnetna polja. Posebno magnetna polja pogosto uporabljajo v industrijski robotiki, na primer za vodenje robota po vnaprej določeni stalni poti skozi skladišče ali proizvodno dvorano. Magnetno polje za vodenje robota ustvarja električni tok po vodnikih tik pod površino tal.

Talni robot je običajno uporabljen z vrsto posebnih programskih rutin. Te pošiljajo računalniku informacije in jih od njega sprejemajo po kablu, s katerim je robot priključen na vrata računalnika. Robot BBC Buggy ima štiri kable za kontrolo motorjev. Po istem kablu Buggy sporoča podatke računalniku. Analogna izhoda iz svetlobnega tipala in čitalnika črtaste kode sta priključena na analogna vrata računalnika, dve tipali trakov pa sta priključeni na dva preostala vhoda na uporabniških vratih računalnika.

Stanje vhodno-izhodnih vrat — na primer uporabniških vrat računalnika — je mogoče ugotoviti s pregledom vsebine na določenih naslovih v pomnilniku računalnika. V basicu navadno prebiramo vsebino na zahtevanem pomnilniškem naslovu z ukazom PEEK. S spremembo vrednosti posameznih bitov na določenem naslovu — na primer z ukazom POKE — programer lahko spremeni smer vrtenja motorjev in tako robota požene v drugo smer.

Podobno lahko pregledamo tudi analogno informacijo — če je računalniku vdelan analogno-digitalni pretvornik. Če ga računalnik nima, potrebuje robot dodatno vmesno enoto, ki skrbi za pretvarjanje analognih signalov v digitalne, preden jih pošlje računalniku.

Želva je v resnici neke vrste talni robot. Namenjena je uporabi jezika logo. Meja med njo in talnim robotom tako v resnici ni ostra (glej stran 112). Najnovejše želve so opremljene s tipali za trke, mnogi talni roboti pa imajo držala za pisalo, tako da jih lahko uporabljamo kot želve. Zamisel o risanju velikih risb s premikanjem peresa na kolesih po risalni površini se je porodila pri proučevanju razmerja med razdaljo, kotom in obliko. Če človek stopi deset korakov naprej, se obrne na levo, stopi deset korakov naprej in se ponovno obrne na levo, pa še enkrat stopi deset korakov naprej in se obrne na levo ter napravi še deset korakov naprej, bo prehojena pot kvadrat. Podobne povezave med gibanjem in oblikami prehojenih poti laže raziskujemo in ponazarjamo, če premikamo želvo, ki pot zariše na papir.

BBC Buggy je mogoče kupiti po delih. Najprej ga moramo sestaviti v želvo, šele potem lahko z njim raziskujemo svet robotike. Ker je Buggy zasnovan kakor sistem Fischer, ga brez težav širimo in izpopolnjujemo.

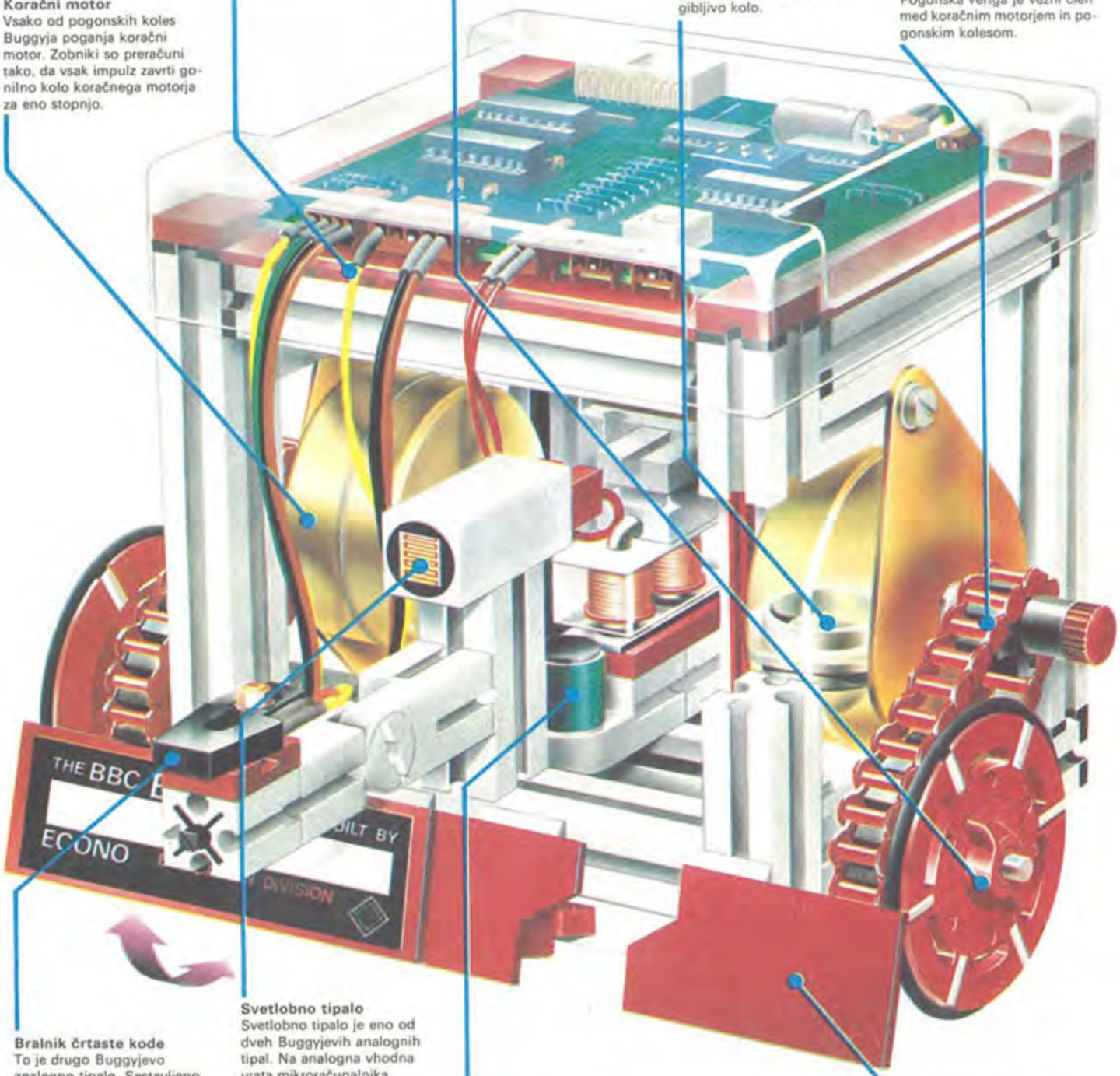
Sestavljanje napravice Buggy pa je zelo poučno. Marsikaj se lahko naučimo že iz raziskovanja povezave posameznih delov v sistemu. Seveda pa se resnično učenje prične šele takrat, ko poskušamo svojo novo »igračko« voditi, usmerjati. Mnogi ne predragi talni roboti so sicer že opremljeni s krmilnimi programi, vendar je pisanje lastnih programov veliko bolj zabavno in zahteva popolnoma nov način programiranja — namreč nadziranje.

Med delovanjem robota mora program stalno nadzirati njegova tipala, da ugotavlja ali je našel črto na tleh, odkril močan svetlobni vir, ali zadel ob stol in podobno. Brž ko detektor nekaj sporoči, se mora računalnik odzvati, da robota (ali pa njegovo okolje) obvaruje pred morebitno poškodbo. Programom te vrste zaradi njihovega takojšnjega odzivanja pravimo programi v realnem času (real-time programs).

Teoretično ni dosti razlike med programom, ki varno vodi robota po sobi, ne da bi se zaletaval ob pohištvo, in takim, ki upravlja elektrarno. Metode vodenja, ki se jih naučimo pri igri z napravo, kakor je BBC Buggy, lahko tudi pripomorejo k razumevanju umetne inteligence. Sestavimo lahko program, ki vodi talnega robota pri opravljanju njegove naloge, dokler robotovo tipalo ne zazna, da so baterije že skoraj prazne. Robot takrat poišče primeren električni vir, da si napolne baterije, in potem delo nadaljuje.

Naslednja generacija talnih robotov bo imela še večje zmožnosti. Verjetno bodo opremljeni z rokami, da bodo lahko prijemali in prenašali manjša bremena. Svetlobna tipala bodo morda nadomestile polvodniške kamere, s katerimi bo robot lahko »videl«, kje hodi. Sintetične govorne enote bodo robotu omogočale komuniciranje z upravljalcem in dojemanje govorjenih ukazov bo odprlo novo pot za nadziranje delovanja robota.

Želva je posebna oblika robota, ki pod nadzorstvom računalnika riše po tleh z odmičnim peresom. Želve so običajno povezane s šolskim jezikom logo (glej stran 112), programiramo pa jih lahko tudi v drugih jezikih (na primer v basicu).



Pogonsko kolo
Pogonski kolesi na obeh straneh Buggyja imata gumijast plašč, ki preprečuje drsenje. Zato morajo biti premiki želve natančni.

Krmilno vezje
Vse signale, poslane računalniku, in tiste, ki jih računalnik pošlje nazaj, to vmesniško vezje obdela.

Zadnje kolo
Na robu Buggyjevega trikotnega podvozja je tole prostogibljivo kolo.

Veriga
Pogonska veriga je vezni člen med koračnim motorjem in pogonskim kolesom.

Koračni motor
Vsako od pogonskih koles Buggyja poganja koračni motor. Zobniki so preračuni tako, da vsak impulz zavrti gonilno kolo koračnega motorja za eno stopnjo.

Bralnik črtaste kode
To je drugo Buggyjevo analogno tipalo. Sestavljeno je iz svetilne diode, ki svetlobo oddaja, in iz fotodiode, ki zaznava odbito svetlobo. Količina odbite svetlobe je odvisna od barve in drugih lastnosti površine tal. Tipalo moremo uporabiti za odbiranje palične kode, pa tudi za gibanje po začrtani poti.

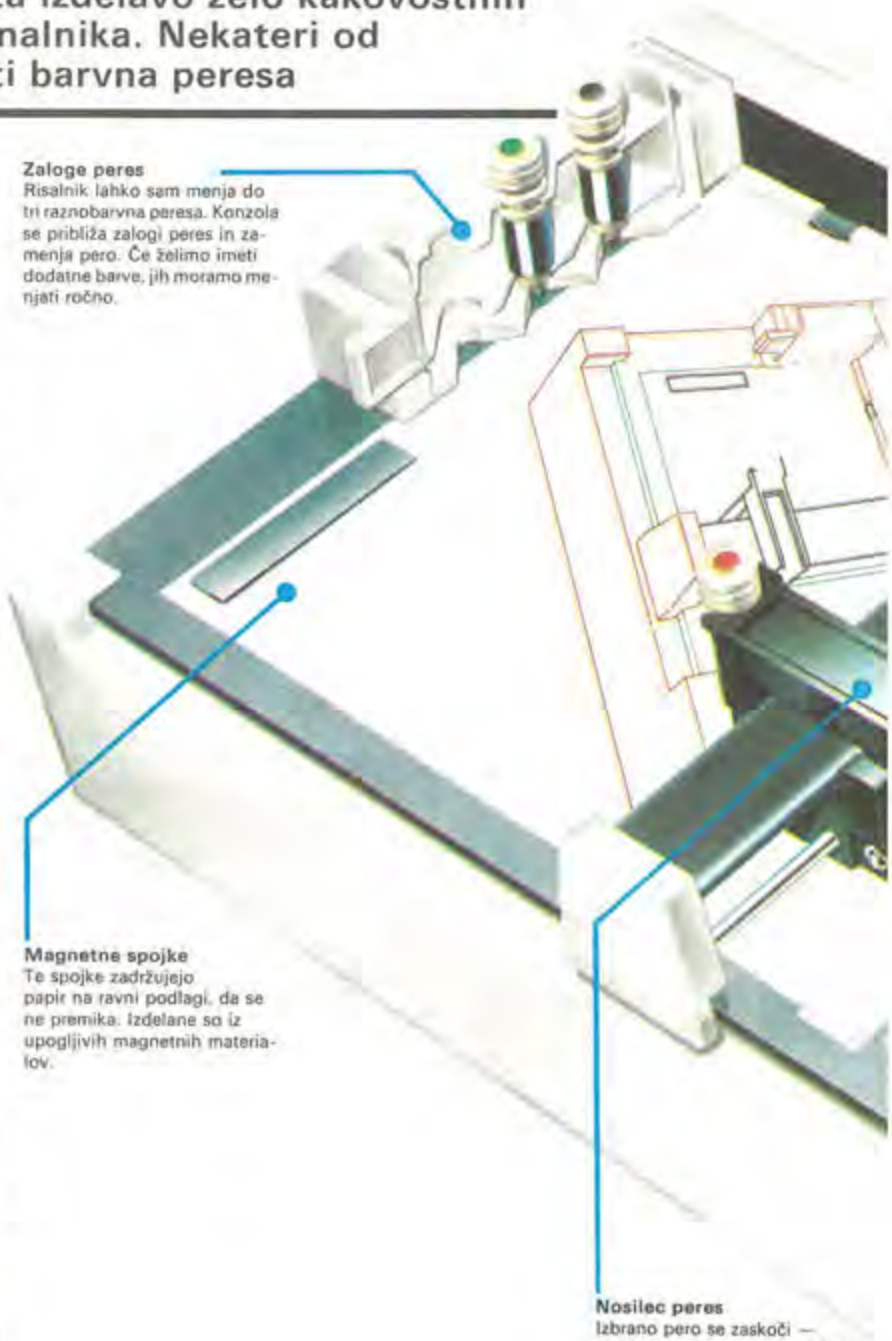
Svetlobno tipalo
Svetlobno tipalo je eno od dveh Buggyjevih analognih tipal. Na analogna vhodna vrata mikroračunalnika pošlje signal, ki je sorazmeren otipani svetlobni jakosti — čim močnejša je svetloba, tem močnejši je signal.

Držalo za pero
Na BBC Buggy je mogoče med kolesi namestiti držalo za pero, da more med premikanjem risati po prevoženi površini črto. Držalo lahko pod nadzorstvom računalnika premaknemo zaradi risanja k tlom ali pa ga od tal odmaknemo.

Prednje tipalo za trke
Na vsaki strani razpolovljenega odbijača je mikrostikalo, ki ob vklopu opozori računalnik na oviro.

Hitro risanje

Risalnik je najboljša naprava za izdelavo zelo kakovostnih risb in grafike z uporabo računalnika. Nekateri od modelov znajo že sami menjati barvna peresa



Mnogi resni uporabniki računalnika zahtevajo od njega izdelavo kopij diagramov na zaslonu. Inženirji, znanstveniki, industrijski oblikovalci in poslovneži potrebujejo natančne diagrame in tabele, ki jih ne morejo dobiti s tiskalnikom. Edina naprava, ki lahko dela take slike, je risalnik, ki je bil do nedavnega še predrag za domačo uporabo.

Odkar pa so prišli na tržišče modeli, kakor sta Tandy/CGP-115 in Oric MCP-40, je prišla ta grafična oprema tudi v doseg plitvejših žepov. V zadnjem času se je na tržišču pojavila vrsta risalnikov, opremljenih z možnostmi, ki smo jih lahko prej našli samo v zelo dragih napravah.

Potreba po risalniku je običajno odvisna od vrste izdelka, ki ga želimo od risalnika. Inženir ali risar bo potreboval natančno risbo naprave ali vezave, poslovnež pa bo mogoče želel grafe in razpredelnice, ki bodo kazali prodajne podatke. Izdelava teh je na navadnem tiskalniku zelo težavna in risba bo samo črno-bela. Edina cenena rešitev je barvna fotografija zaslona in če bo ta že dovolj dobra za poslovneža, gotovo ne bo dovolj natančna za inženirja ali arhitekta.

Risalniki delujejo popolnoma drugače kakor tiskalniki: rišejo črte od točke do točke, namesto da bi izdelovali risbo s tiskanjem že pripravljenih znakov ali točk. Večina risalnikov temelji na koordinatnem sistemu X-Y. Kadar lahko graf narišemo tako, da določimo koordinate točk, skozi katere graf poteka, lahko narišemo tudi kakršnokoli drugo obliko tako, da jo razdrobimo na posamezne koordinate. Te koordinate mora risalnik samo še povezati med seboj. Zato je pero risalnika pritrjeno na konzolo, ki se lahko pomika v smeri X (levo — desno), samo pero pa se lahko pomika po konzoli v smeri Y (gor in dol).

Klasični risalniki so znani kot risalniki z »ravnim ležiščem«, ker je pri njih papir pritrjen na ravno ploščo, čez katero se giblje konzola s peresom — ta tip je prikazan na sliki desno. Njegova pomanjkljivost je v tem, da mora biti risalnik vsaj tako velik kakor papir.

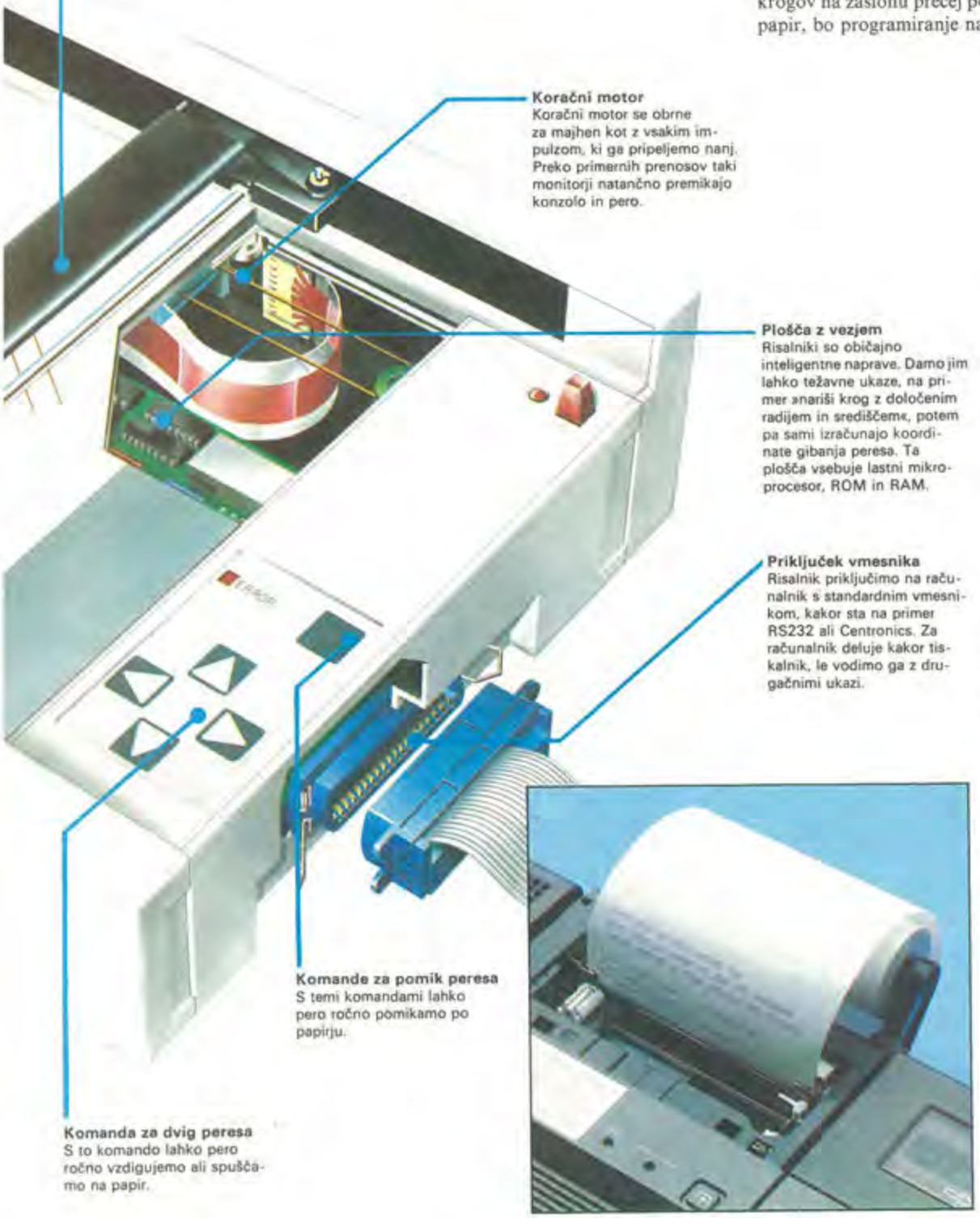
Eden od načinov za zmanjšanje risalnika je ta, da uporabimo izvedbo, ki je narisana na sliki: v smeri Y se pomika pero, v smeri X pa papir. Tako so recimo narejeni risalniki Strobe 100 in Hewlett Packard Sweetlips. Premikanje papirja mora biti enako natančno kakor gibanje konzole pri klasičnem risalniku; to dosežemo z uporabo koračnega motorja. Koračni motor je poseben motor, ki se zavrti za delček obrata, kadarkoli pripeljemo nanj električni impulz. Koračne motorje imamo tudi v diskovnih enotah, tam določajo pozicijo glave na disku, in v industrijskih robotih.

Risalnik priključimo na računalnik enako kakor tiskalnik, vsaj kar se tiče vmesnika. Večina risalnikov je opremljena s serijskim (RS232) ali paralelnim (Centronics ali IEEE488) vmesnikom, ki ga lahko priključimo na izhod na računalniku, navadno namenjen za tiskalnik. Programiranje pa je često bolj zapleteno, saj moramo risalniku poleg rezultatov izračuna poslati še način, kako naj te rezultate nariše. To delamo precej podobno, kakor če rišemo diagram na zaslonu.

Zaradi zapletenega načina risanja je večina risalnikov »inteligentna«. To pomeni, da imajo vdelane mikroprocesorje, ki spreminjajo znake iz računalnika v serijo koordinat, ki jih risalnik nato nariše. Mnogi boljši risalniki znajo narisati tudi bolj zapletene oblike, recimo kroge in razne krivulje, dodamo jim samo izhodiščne točke. Nekateri risalniki tudi sami označijo diagrame in pobarvajo histograme ali krožne diagrame in nam tako olajšajo delo pri programiranju.

Mnogim risalnikom je priložen že program, ki nam omogoča uporabo risalnika v uporabniškem programu, kot na primer zaslona COPY. Če takega programa nimamo, bomo morali podatke z zaslona ustrezno prirediti in jih poslati risalniku. Nekateri risalniki nimajo vdelanega nabora znakov, v tem primeru bomo morali risalniku poslati tudi kode za risanje znakov. To pa vsaj omogoča uporabniku, da sam izdela svoj nabor znakov in njihovih tipov. Ko smo določili obliko znaka, ga lahko narišemo kamorkoli na papir in v poljubni orientaciji ali velikosti. Tudi rutine za risanje krogov in krivulj so često zelo koristne, posebno na polju poslovne grafike. Ker so rutine za risanje krogov na zaslonu precej podobne tistim za risanje na papir, bo programiranje navadno prav preprosto.

Konzola
Konzolo lahko postavimo na katerokoli mesto v smeri X, pero na konzoli pa kamorkoli v smeri Y. S kombinacijo dveh gibanj lahko dosežemo vse točke na papirju.



Koračni motor
Koračni motor se obrne za majhen kot z vsakim impulzom, ki ga pripeljemo nanj. Preko primernih prenosov taki monitorji natančno premikajo konzolo in pero.

Plošča z vezjem
Risalniki so običajno inteligentne naprave. Damo jim lahko težavne ukaze, na primer »nariši krog z določenim radijem in središčem«, potem pa sami izračunajo koordinate gibanja peresa. Ta plošča vsebuje lastni mikroprocesor, ROM in RAM.

Priključek vmesnika
Risalnik priključimo na računalnik s standardnim vmesnikom, kakor sta na primer RS232 ali Centronics. Za računalnik deluje kakor tiskalnik, le vodimo ga z drugačnimi ukazi.

Komande za pomik peresa
S temi komandami lahko pero ročno pomikamo po papirju.

Komanda za dvig peresa
S to komando lahko pero ročno vzdigujemo ali spuščamo na papir.

Tiskalnik-risalnik s štirimi peresi

Ta mehanizem je prvič zbudil pozornost, ko se je pojavil v Sharpovem tiskalniku CE-150. Njegova večja brata Tandy CGP-115 in Oric MCP-40 pa sta približala ceneno tiskanje in risanje tudi uporabnikom hišnih računalnikov.
Kakor vse dobre ideje je tudi zamisel za ta sistem neverjetno preprosta. Valj z iglicami vleče papir skozi mehanizem, in sicer naprej in nazaj v natančno določenih stopnjah, držalo peres s štirimi miniaturnimi peresi pa se pomika vodoravno.
Slika na papirju nastane tako, da se nosilec peres najprej zavrti na določeno barvo peresa. Vodoravne linije riše risalnik s pomiki nosilca, navpične pa s pomiki papirja. Kombinacija teh dveh gibanj nam da poljubne oblike krivulj ali znakov. Slike so zelo kakovostne, čeprav risalnika zaradi ozkega papirja ne moremo uporabljati za tekstovno procesiranje in podobne resne namene.

Halo, računalnik...

Akustični spojnik spreminja digitalne podatke v slišne tone in narobe. Če ga spojimo s telefonsko slušalko, ga lahko uporabljamo za komunikacijo z drugimi računalniki

Priključitev hišnega računalnika na tiskalnik ali disketno enoto nam običajno ne dela težav, saj so vse naprave v isti sobi, če ne celo na isti mizi. Priključitev računalnika na njegovega večjega brata v pisarni ali na drugi strani sveta pa je že večji problem. Na srečo že imamo sistem za komunikacijo po celem svetu — to je telefonsko omrežje. Potrebujemo le še napravo, s katero bomo nanj priključili računalnik.

Ker se telefonsko omrežje že na široko uporablja za velike računalniške sistema (na primer za rezervacije letalskih vozovnic), je tehnologija na tem področju že dognana. Za svoj hišni računalnik potrebujemo samo še poenostavljeno različico že delujočih naprav. Običajno potrebujemo za priključitev računalnika na telefonsko omrežje napravo, ki se imenuje modem. To čudno ime je le okrajšava za modulator-demodulator.

Naprava deluje podobno kakor kasetni vmesnik v računalniku. Vzorec binarnih številk 1 in 0 spremeni v serijo zvočnih signalov z dvema frekvencama (modulacija) in jih nato pošlje po telefonski liniji. Na nasprotni strani modem te signale spremeni nazaj v enice in ničle (demodulacija). Tudi kadar računalnik ne pošilja podatkov, modem oddaja stalen ton (nosilni ton), da računalnik na nasprotni strani ve, da je zveza še vedno postavljena.

Glavna pomanjkljivost modema je ta, da je fiksno priključen na telefonsko linijo in jo s tem stalno zaseda, to pa je za domačega uporabnika včasih neprijetno.

Drug način povezave se imenuje akustični spojnik. Ker sistem uporablja slišne tone, jih lahko pošiljamo na zvočnik. Če ta zvočnik spojimo s telefonsko slušalko, lahko tone pošiljamo po telefonski liniji. Na drugi strani spojimo slušalko z mikrofonom, s katerim sprejemamo tone. Za tak način prenosa je bil akustični spojnik tudi izdelan. Od modema se loči po tem, da ni stalno priključen na telefonsko linijo.

Na voljo je velika izbira akustičnih spojnikov, od takih, kakor je prikazan na sliki, dovolj majhnih, da jih lahko uporabljamo s prenosnimi računalniki, pa do velikih namiznih modelov. Boljši modeli znajo sami odgovoriti na klic, ne da bi bil operater pri računalniku, in sicer tako, da kontrolirajo nosilni ton.

Tako kakor kasetni vmesniki tudi akustični spojniki delajo z različnimi hitrostmi. Področje hitrosti pa je tu strogo omejeno. Prenosne karakteristike telefonskih prenosnih sistemov omogočajo dovolj zanesljiv prenos le do hitrosti 1200 znakov na sekundo.

Cenene naprave lahko prenašajo celo samo po 30 znakov na sekundo, pri dražjih modelih pa je na voljo vrsta hitrosti. Seveda je pomembno, da morata napravi na obeh straneh linije delovati z isto hitrostjo, sicer ne bo prenosa.

Široka uporaba hišnih računalnikov je spodbudila razvoj in izdelavo velikega števila novih izdelkov na področju prenosa podatkov. Naprave, kakor je Sendata in njeni sorodniki, omogočajo prenosnim računalnikom (kakor sta na primer Tandy Model 100 in Epson HX-20), da delujejo kot računalniški terminali za vrsto uporabnikov od poslovnežev do časnikarjev. Naročila, članke in dopise lahko pripravijo na računalniku in jih potem po telefonu pošljejo v glavno pisarno.

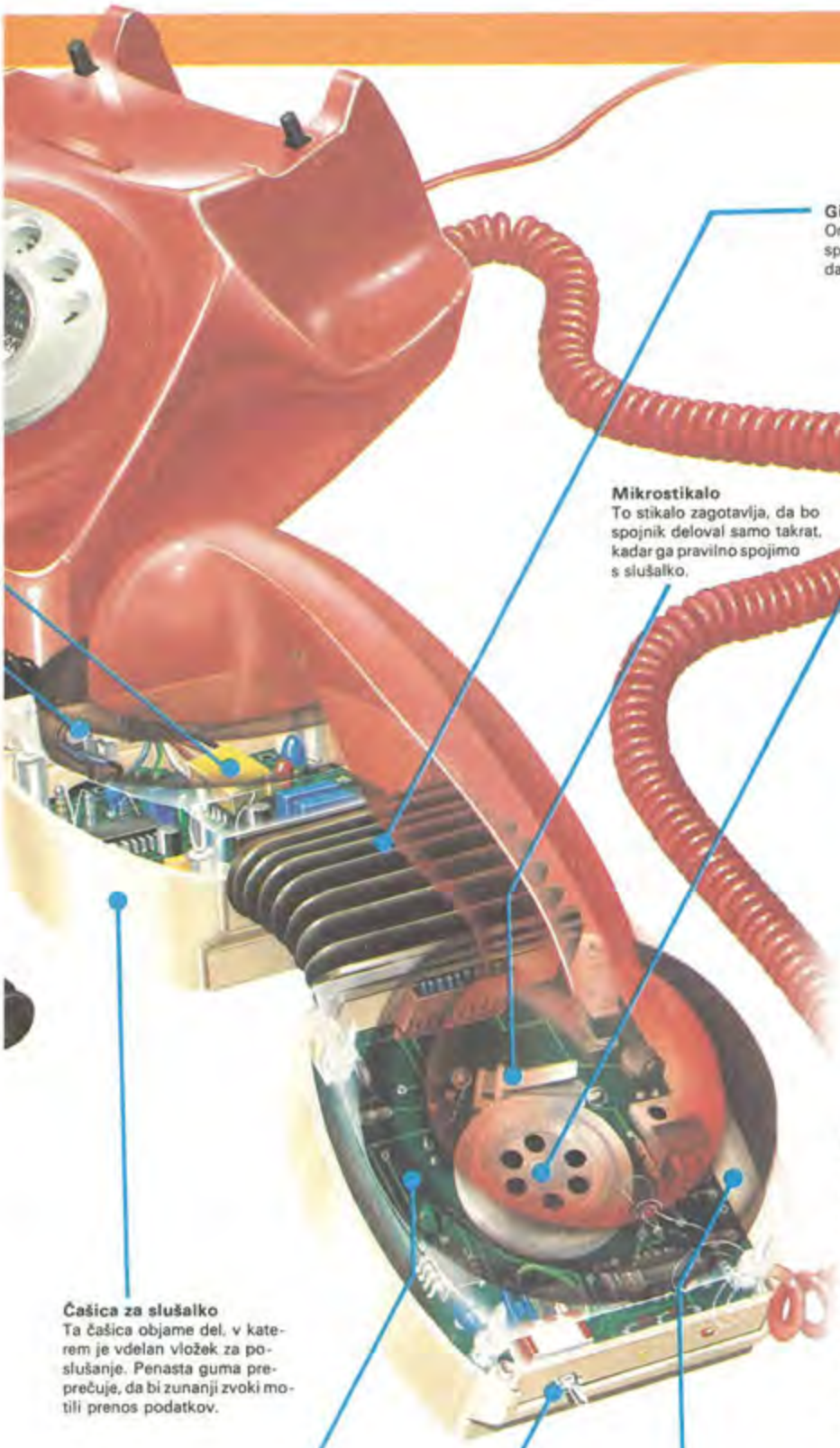
Poslovnežem omogočata akustični spojnik in računalniški terminal trenuten dostop do velikega števila informacij in računalniških birojev. Naročanje blaga v velikih trgovinah že teče preko računalnikov tako, da osebje vstavi v računalnik podatke o vrsti in količini blaga, nato pa jih pošlje v glavni računalnik podjetja. Pri hišni uporabi ima akustični sklopnik prednost pred modemom, ker ga ni treba za stalno priključiti na

Mikrofon
Mikrofon spreminja zvočne signale iz slušalke v električne in jih pošilja v elektronsko vezje.

Priključek za napajanje
Preko tega priključka napajamo spojnik in hkrati polnimo vdelane Ni-Cd akumulatorje.

Kabel za vmesnik
Ta kabel priključimo na serijski vmesnik (RS232) računalnika.

Telefonska govorilnica
Lahki prenosni akustični spojniki omogočajo potujočemu uporabniku računalnika preko telefonskega omrežja zvezo s katerimkoli računalnikom na svetu.



linijo. Poslovnež, ki dela doma pri hišnem računalniku, se lahko vsak trenutek poveže s svojo pisarno in pošlje ali sprejme podatke, ne da bi imel stalno zaseden telefon.

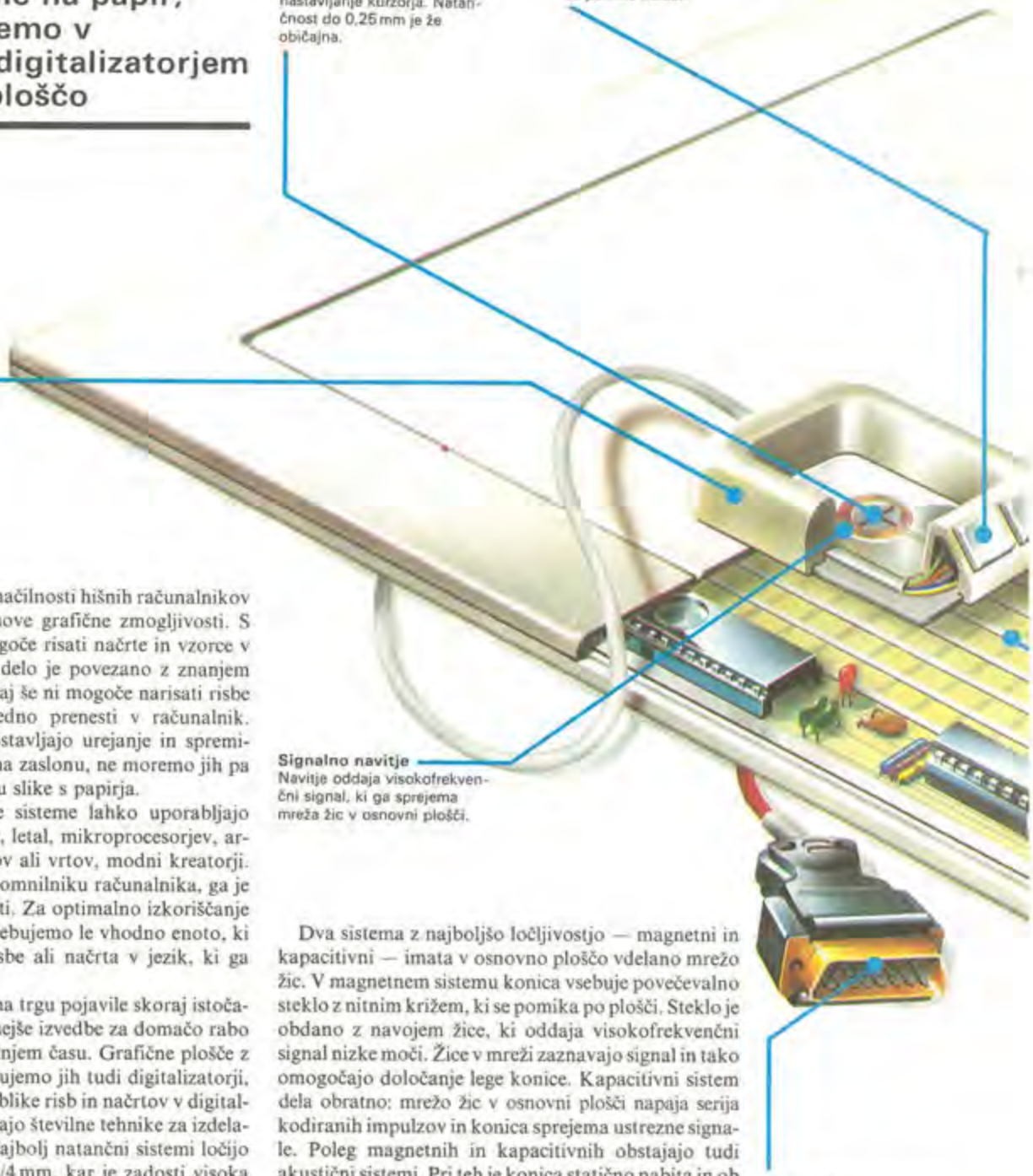
Na tržišču hišnih računalnikov je akustični spojnik cenena in pripravna alternativa modemu pri vključevanju v javne podatkovne baze, kakor sta Prestel in Micronet 800. Spojnik je tudi dosti zanesljivejši način za sporočanje programov prijateljem kakor pa pošiljanje kaset po pošti. Elektronska pošta, ki uporablja storitve, kakor je »Telecom Gold« British Telecoma, ponuja lastnikom hišnih računalnikov akustični spojnik z možnostjo takojšnje povezave, kakor so jo pred kratkim imela le velika podjetja.

Po drugi strani pa se evropski akustični spojnik ne more povezati z ameriškim. Ameriška računalniška industrija dela po standardu Bell 103, evropska pa po CCITT V21. Najbrž ni potrebno dodajati, da se standarda ne ujemata.

Dodatne težave prinaša nova generacija telefonskih aparatov, katerih slušalke se ne prilegajo več v mnoge akustične spojnike (glej sliko). Ker deluje sistem tako, da oddaja zvok, se moramo ogniti vsakemu zunanjemu šumu. Če se spojnik ne prilega natančno slušalki, se lahko zgodi, da zunanji šum močno popači podatke, ki jih oddajamo.

Branje grafičnih podatkov

Slike, napisane na papir, lahko prenesemo v računalnik z digitalizatorjem ali grafično ploščo



Ena najpomembnejših značilnosti hišnih računalnikov četrte generacije so njihove grafične zmogljivosti. S preprostimi ukazi je mogoče risati načrte in vzorce v različnih barvah. Tako delo je povezano z znanjem programiranja, saj za zdaj še ni mogoče narisati risbe na papir in jo neposredno prenesti v računalnik. Svetlobna peresa poenostavljajo urejanje in spreminjanje slike, ko je ta že na zaslonu, ne moremo jih pa uporabljati pri kopiranju slike s papirja.

Grafične računalniške sisteme lahko uporabljajo načrtovalci avtomobilov, letal, mikroprocesorjev, arhitekti bivalnih prostorov ali vrtov, modni kreatorji. Ko je načrt shranjen v pomnilniku računalnika, ga je moč poljubno spreminjati. Za optimalno izkoriščanje grafike računalnika potrebujemo le vhodno enoto, ki bo prevedla oblike z risbe ali načrta v jezik, ki ga računalnik razume.

Grafične plošče so se na trgu pojavile skoraj istočasno kot računalniki. Cenejše izvedbe za domačo rabo pa so na voljo šele v zadnjem času. Grafične plošče z visoko ločljivostjo, imenujemo jih tudi digitalizatorji, ker prevajajo analogne oblike risb in načrtov v digitalno informacijo, uporabljajo številne tehnike za izdelavo želene informacije. Najbolj natančni sistemi ločijo točke, ki so oddaljene 1/4 mm, kar je zadosti visoka ločljivost za inženirje in risarje. Za digitalizatorje je značilna ravna osnovna plošča, na katero se polagajo načrti ali risbe. Za branje podatkov se uporablja konica, ki je lahko navadno pero ali zapletena elektronska naprava. Organizator ugotovi lego konice na osnovni plošči in jo prenese v računalnik kot par koordinat.

Dva sistema z najboljšo ločljivostjo — magnetni in kapacitivni — imata v osnovno ploščo vdelano mrežo žic. V magnetnem sistemu konica vsebuje povečevalno steklo z nitnim križem, ki se pomika po plošči. Steklo je obdano z navojem žice, ki oddaja visokofrekvenčni signal nizke moči. Žice v mreži zaznavajo signal in tako omogočajo določanje lege konice. Kapacitivni sistem dela obratno: mrežo žic v osnovni plošči napaja serija kodiranih impulzov in konica sprejema ustrezne signale. Poleg magnetnih in kapacitivnih obstajajo tudi akustični sistemi. Pri teh je konica statično nabita in ob dotiku z osnovno ploščo ustvari majhno iskro in zvočni val. Čas, ki ga val porabi za pot do mikrofona, daje informacijo o legi konice. Med drugim ta sistem omogoča digitaliziranje tretje dimenzije s pošiljanjem signala skozi različne plasti.

Na repu lestvice so grafične plošče, ki so občutljive na pritisk. Papir se položi na ploščo in potem se s

Osnovna plošča
Risbo, ki jo bomo digitalizirali, položimo na ploščo. Pri nekaterih sistemih je plošča elektrostatično nabita, tako da se papir »prilepi« nanjo. Zelo je pomembno, da slike ne pomikamo po plošči.

Branje zemljevidov
Ena od najbolj razširjenih poslovnih uporab digitalizatorjev je zbiranje podatkov z zemljevidov in geodetskih načrtov. Zgled kaže uporabo računalnika za napovedovanje lege novih naftnih polj iz digitaliziranih geoloških podatkov.

Delovna plošča
Delovna plošča vsebuje mikroprocesor ter delovni in bralni pomnilnik. Tako je izdelana zato, da lahko računalniku sporoča podatke v obliki parov koordinat x in y.

Sprejemna mreža
V osnovno ploščo je vdelana mreža žic, ki sprejema signale, ki jih oddaja tuljava. Mreža je redkejša, kakor bi lahko sklepali iz natančnosti digitalizatorja, ker procesno vezje interpolira pravo lego iz moči sprejetih signalov v sosednjih žicah.

konico prekrijejo vsi podatki. Sistem zahteva močnejši pritisk na ploščo. Dve električno prevodni plošči sta ločeni s staničastim izolatorjem. Plošči napajata visokofrekvenčna signala različnih frekvenc. Signal, ki ga sprejme konica, ko povzroči električni stik med ploščama, določi njen položaj. Tipičen problem, ki ga srečujemo pri tem sistemu, je spreminjanje površinske upornosti plošče zaradi poškodb ali dotikanja z rokami med digitaliziranjem. Zaradi manjše zmogljivosti hišnih računalnikov pa je tudi ta metoda več ko primerna za delo z njimi. Najcenejši in najpreprostejši digitalizatorji pa so »pantografi«, ki delujejo po načelu starega risarskega pripomočka »upognjene roke«. Uporabljajo geometrijo koordinat, ki omogoča neposredno določanje lege konice. Spremenljiva upornost v dveh sklepih daje napetost, sorazmerno kotom v »rami« in »komolcu« roke. Natančnost pantografa je omejena z natančnostjo spremenljivih uporov in mehanskih povezav. Tipična natančnost je samo pet odstotkov.

Izboljšani pantografi, ki slonijo na načelu optičnega merjenja zasukov sklepov, dajejo veliko boljše rezultate, čeprav imajo še vedno slabše lastnosti od magnetnih in kapacitivnih sistemov. Optične plošče uporabljajo za določanje lege konice mrežo infrardečih žarkov. Njihova natančnost ni primerljiva z natančnostjo drugih sistemov, omogoča pa uporabo prsta za izbiro posameznih polj na plošči.

Pri nekaterih izvedbah so infrardeči viri in sprejemniki razporejeni po robu enote in tako ustvarjajo interaktiven zaslon, po katerem se da risati kar s prstom.

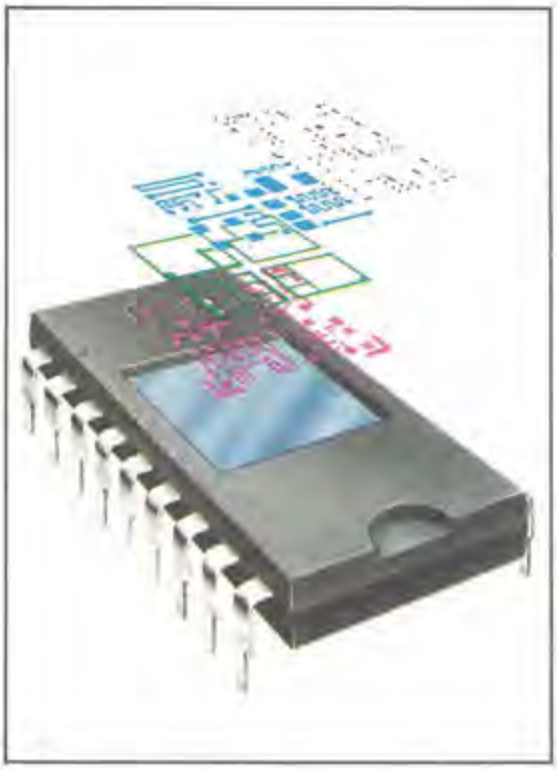
Podatki, ki jih daje grafična plošča ali digitalizator, morajo biti prevedeni v obliko, primerno za prikaz na zaslonu. Večina izdelkov vsebuje vso potrebno opremo za to. Vsekakor pa vpis podatkov ni edini način uporabe grafičnih plošč. Ko so podatki shranjeni v računalniku, se plošča lahko uporablja za vpis popravkov in sprememb. Površina plošče je lahko programirana tako, da rabi kot izvirnik standardnih podpornih programov in se tipkovnica uporablja le za določitev glavnih funkcij. Vsi sistemi za računalniško animacijo imajo visokokvalitetne grafične plošče, ki se uporabljajo za vpis podatkov.

Načrtovanje po naročilu

Logične mreže (ULA) lahko izvajajo vse funkcije hišnih računalnikov, razen funkcij centralne procesne enote ter delovnega in bralnega pomnilnika

Ena najpomembnejših novosti v elektronskem načrtovanju, ki so posledica hitrega razvoja mikroračunalnikov, je razvoj integriranega vezja (čipa), ki se imenuje logična mreža. Čeprav razvoj logičnih mrež v širši javnosti ni zbudil večjega zanimanja, je v zadnjih letih prišel do take stopnje, da je že mogoče delati zelo zmogljive računalnike in druge naprave iz štirih glavnih komponent: centralne procesne enote (CPU), delovnega pomnilnika (RAM), bralnega pomnilnika (ROM) in — za povezavo med temi enotami — logične mreže (ULA).

Kaj je logična mreža? Kot nam pove že samo ime, je to veliko število (polje) logičnih vrat, ki med sabo niso povezana, lahko pa se povežejo tako, da opravijo skoraj vsako funkcijo, ki jo načrtovalec potrebuje. Logična mreža se lahko tolmači tudi kot izboljšava bralnega pomnilnika, saj lahko vsebino obeh komponent določi le proizvajalec integriranih vezij, uporabnik pa ne.



Zadnja plast
Vsa polprevodniška integrirana vezja so narejena iz polprevodniških plasti, ki so individualno jedkane tako, da sestavljajo elemente vezja. Zadnja plast je prevodna in določa povezave med elementi. Logična mreža vsebuje polje logičnih elementov, ki so lahko povezani v kompleksno vezje.

Preden bralni pomnilnik ali logično mrežo »sprogramiramo«, se sestoji samo iz velikega števila preprostih elektronskih vezij ali celic, ki med sabo niso povezane in je zato neuporabna. Vsa integrirana vezja so narejena z nanašanjem plasti polprevodniških materialov. Zadnja plast je po navadi narejena iz prevodnega materiala in ustvarja povezave med različnimi celicami. Veliko število možnih povezav dela logično mrežo zelo prilagodljivo zahtevam načrtovalca. Čeprav so celice precej preproste, sestoje se lahko iz nekaj tranzistorjev ali enega samega upora, jih z zadnjo plastjo lahko med sabo povežemo tako, da sestavljajo kompleksna vezja, kot so registri, števci in vezja s časovnimi funkcijami. Za funkcije, ki jih lahko izvajajo logične mreže, v hišnih računalnikih po navadi uporabijo celo vrsto standardnih integriranih vezij.

Logična mreža je lahko programirana tako, da opravlja zelo različne funkcije. Uporabi se lahko kot sintetizator zvoka, nastavljalec osvetlitve in ostrine pri fotografskih aparatih, ali pa lahko opravi večino dela pri digitalnem termometru. Poleg tega logična mreža skoraj ne potrebuje zunanjih vezij — razen za napajanje, za stikala ter za nekatera tipala ali kontrolne tipke.

Pri načrtovanju plasti, ki povezuje celice v logični mreži, uporabljajo računalnike. Mini računalnik, kot na primer DEC PCP11/23, ki se uporablja pri računalniško podprtem načrtovanju, najprej nariše načrt planirane sestavnice mask. Ko je načrt narisan, se prenese v večji računalnik, ki preveri, ali je sprejemljiv, ga primerja z osnovnim logičnim načrtom in popravi vse napake. Potem načrt obdela še drugi program, ki simulira vezje na podlagi testnih podatkov, ki jih zagotovi naročnik. Ko je načrtovanje končano, računalnik naredi predlogo za masko, ki se uporabi pri nanašanju zadnje plasti.

Kako daleč lahko gredo logične mreže? Zamisel o postavljanju velikega števila preprostih celic v silicij in možnost uporabnika, da se odloči, kako bodo celice povezane, je tako privlačna, da lahko postane pomembna metoda za načrtovanje večine vezij. Uporaba logičnih mrež v današnjih tehnologijah je upravičena le tedaj, ko potrebujemo nekaj tisoč enakih vezij. PROM, EPROM, EEPROM in EAROM so izpeljanke bralnega pomnilnika, ki jih lahko uporabnik, če ima ustrezno opremo, programira sam. Mogoče se bodo v kratkem pojavile tudi podobne različice logičnih mrež.

5. poglavje

Računalnik pri delu in igri

V službi, v šoli, doma — računalnik počasi toda zanesljivo prodira v vse pore življenja.

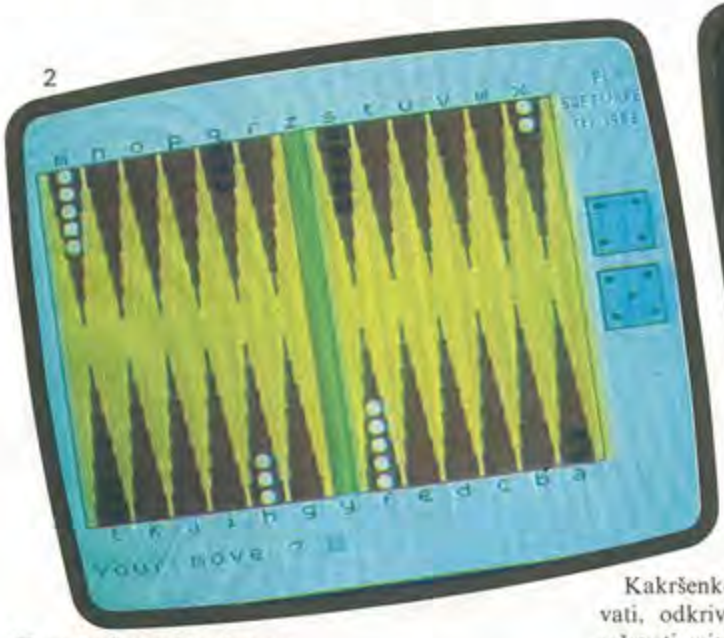
Ljudje in igre

Ječe in zmaji, borze in vesoljski poleti, razvedrilo in izobraževanje — vse to je v računalniških igrah

1

Večina lastnikov računalnikov igra z njimi tudi igre in svet računalniških iger je zanimiv kalejdoskop razvnetosti, osupljivih ugank in novih izzivov.

Ceneni in močni hišni računalniki hitro prehitevajo naprave za video igre, tako za uporabo v igralnicah kot tudi za domačo uporabo. Poleg tega, da vam dajejo več raznolikosti in zabave, vam tudi ne poberejo vseh prihrankov, kakor vam jih je vzelo učenje nove igre v »Mini casinu«. Računalniške igre so pripeljale v svet računalnikov več ljudi, kakor jih je ali jih kdaj bo kakršenkoli knjigovodski program. Ne imejte občutka, da računalnika ne uporabljate pravilno, če se z njim igrate. Igre so pač za to, da v njih uživate.


2

3

4

Pustolovske igre

Ni nujno, da igra človeka zasvoji z barvami in zvokom. Skupaj s cenenimi računalniki se je pojavila nova vrsta iger, ki vzbujajo domišljijo z besedami, kakor so jo vedno doslej knjige.

Te igre imenujemo pustolovske v spomin na prvi program, ki so si ga programerji naredili, da bi se z njim igrali na svojem velikem računalniku v prostem času. Ideja je v tem, da program pričara neki svet, po katerem igralec vodi svoj »drugi jaz«, vodi pa ga z vtipkavanjem ukazov na tipkovnici.

»Možička« vodimo po računalniškem svetu z vtipkavanjem smeri, kakor so »sever« ali »jug« ali »gor«, računalnik pa daje opise okolice in predmetov v njej. Igralčev drugi jaz lahko te predmete vzame (TAKE), spusti (DROP), obrne (TURN) ali razbije (BREAK), odloči pa se lahko za kakršnokoli početje, ki se zdi igralcu obetavno. Edina omejitev je domišljija.

Svet v računalniku je lahko blodnjak podzemeljskih jam in ječ, poln pošasti in zakladov, kakor na primer v priljubljeni igri Ječe in zmaji (Dungeons and Dragons). Lahko je tudi zapuščena marsovska vesoljska ladja, ali morda celo zapuščena podeželska hiša, v kateri je treba razjasniti umor.

Kakršenkoli je že scenarij igre, igralec mora raziskovati, odkrivati uporabne predmete in zaklade ter reševati miselne uganke. Računalniški svet je treba dodobra preiskati in polno število točk v rezultatu prinašajo le rešitve vseh ugank ter vsi najdeni zakladi. Najboljše pustolovske igre so kakor napeti romani, vendar lahko traja dlje, da dokončamo igro, kakor pa preberemo roman.

6

Namizne igre

Naravno je, da so se v računalniški obliki pokazale vse najbolj priljubljene namizne igre, brž ko jih je računalniška tehnika zmogla. Tako vam ni treba več iskati soigralca, če lahko to opravi računalnik, in če naredite

napačno potezo, jo lahko popravite, ne da bi vas računalnik obtožil goljufije. Ker vas računalnik opozarja na storjene napake in jih popravlja, svojo igro lahko tudi izboljšate.

Računalniški šah je dosegel že zelo visoko raven. Grafika hišnih računalnikov zmore pokazati položaj na šahovnici popolno in podrobno, z zveznimi premiki figur. Dama je igra, ki je proučena do vseh podrobnosti, in računalniški program bi bil v tej igri lahko svetovni prvak. Dostopni so tudi programi za backgammom, bridge, othello, pontoon in go, ki se dajo igrati na vrsti hišnih računalnikov. Računalniki so močni nasprotniki na plošči, mizi ali zaslonu.

Učenje ob igri

Nekateri proizvajalci programske opreme se posebej posvečajo poučnim igram, pri katerih računalnik spra-

5

1. Čarovništvo: Vstopite v skrivnostni svet. Tovariši vam lahko pomagajo. Toda koga izbrati? Bojevnika, dekle ali znanstvenika?

2. Backgammon: Vaš nasprotnik v tej igri ima za seboj velikansko moč — nezmotljivo logiko računalnika.

3. ABC Dragon: Učna igra za otroke.

4. Zaxxon: Igro so najprej igrali v igralnicah. Zaslon postane pilotovo okno. Med raketami in boji do cilja!

5. Vsaka poteza v strateških igrah zahteva temeljit premislek.

6. Simulacija letala: Pilotirajte ali strmoglavite!

šuje in postavlja uganke ter tudi ponudi nagrado, če so odgovori pravilni. Kakšna igra se zdi samo igra, v resnici pa uči seštevati, pisati ali kake fizikalne zakone.

Zelo je priljubljena igra, pri kateri otroci vodijo po papirju robota s peresom (zaradi oblike robota imenujemo »želva«). Tako se hkrati zabavajo in učijo geometrijo.

Mini kazino (arkadne igre)

Igre, ki jih običajno vidimo v igralnicah, zahtevajo hitre reakcije, skozi reže na video napravah so zvabile že milijarde kovancev. Na hišnih računalnikih lahko igrate vse uspešnice igralnic, boj proti letečim osvajalcem, skakajočim žabam, rudarjem in gorilam.

Proizvajalci programske opreme pa so šli še dlje in danes je mogoče dobiti igre, ki so po zanimivosti in spektakularni grafiki enake igralniškim, vendar se igrajo na hišnih računalnikih. Te igre so napete, hitre, vedno pripravljene za igranje, in nobene lačne reže ni, ki bi čakala na kovanec. Pri takih igrah programerji in računalniki pokažejo vse, kar zmorejo.

Strateške igre

Igre, ki zahtevajo mišljenje in sposobnost načrtovanja, zvitost in zagrizenost velikega generala, so tudi prispele do mikroračunalnikov.

Igralec je lahko katerikoli general v katerikoli vojni, razvršča armade in skuša prelisičiti svojega računalniškega nasprotnika. Računalnik izvrstno deluje kot sodnik in kontrolor, nadomeščajoč gosto tipkana navodila in zlahka izgubljajoče se igralne figure, zaradi katerih je bilo vojskovanje na deski poprej omejeno na ozek krog ljubiteljev.

Druga oblika igre je, da je igralec kralj majhne dežele in mora skrbeti za njeno gospodarstvo ter zaklade tako, da so podložniki zadovoljni in siti, polja pa varna pred roparji.

Igralec je lahko odgovoren za energetsko bilanco »svoje« dežele in išče ravnotežje med prednostmi, dolgoročnimi nevarnostmi in ceno premoga, nafte ter jedrske energije. Računalnik vam lahko tudi pomaga gledati dolgoročno in izpopolniti vaše načrte za prevzem sveta.

Simulacijske igre

Računalniški programi vas lahko posadijo v pilotski sedež majhnega letala, z vsemi instrumenti in potrebnimi kontrolami, tako da lahko z njim letite, pristajate in vzletate s posnetkov resničnih letališč. Podobno lahko postanete pilot vesoljskega taksija in skozi njegove line opazujete Zemljo. Na svetovni borzi ste lahko finančnik Rockefellerjeva kova, ki ustvarja in uničuje velika podjetja.

V simulacijskih igrah je natančnost odločilnega pomena. Če sledite pravilom resničnega sveta, vam igra pokaže, kaj bi se zgodilo v svetu. Naredite napako in znašli se boste ujeti v razbitinah ali boste prisiljeni skočiti z nebotičnika na Wall Streetu. Vendar računalniki odpuščajo več kakor resnični svet!

Elektronski vzgojitelj

Tudi najmlajši član družine se lahko nauči delati z računalnikom. Tu je opisan najboljši način, kako začeti

Eden najmočnejših računalniških pripomočkov za otroke v osnovni šoli je »želva«. Ta robot je povezan z računalnikom in voden s programom, imenovanim logo. Želva je zelo priročna za poučevanje matematičnih pojmov, kakor so liki, razdalje in odnosi med liki, saj lahko otroci z njo rišejo po papirju. »Želva« je tudi zelo zabavna.

Veliko staršev se sprašuje, ali bo računalnik koristen za njihove otroke. Večina jih ve, da je dobro, če doma in v šoli z računalniki delajo najstniki, toda ali se ob računalniku lahko česa nauči tudi najmlajši otrok? Da! Odgovor je pritrdilen, toda obstajajo različni načini, kako se da otroka vpeljati v svet računalnikov, in eni načini so boljši od drugih. Računalniki niso uporabni le za računanje. Z dobrim programom — za zdaj je takih programov za otroke premalo — lahko računalnik otrokom pomaga pri učenju glasbe, zemljepisa, tujih jezikov in seveda temeljev aritmetike in geometrije.

Obstajata dva glavna načina, kako lahko računalnik pomaga majhnim otrokom. Otrok lahko uporabi računalnik za širjenje svojega obzorja, drugi način pa je, da računalnik uči otroka.

Šest let starega otroka nima smisla učiti programiranja v basicu. Pred dvanajstim letom otrok ne more dojeti abstraktnih pojmov takega jezika. Nekateri otroci lahko pišejo programe v basicu pri devetih ali desetih letih ali celo prej, toda francoski psiholog Jean Piaget pravi, da ima večina otrok, mlajših od dvanajst, trinajst let, težave z dojemanjem abstraktnih pojmov.

Raziskovalci so, zavedajoč se težav, našli način, kako lahko otrok kontrolira in programira računalnik, ne da bi pri tem imel opraviti z abstraktnimi pojmi (glej opis jezika logo). Po navadi učitelji seznanijo otroka z računalnikom po poti, ki je mešanica obeh metod.

Potujoča želva

Že zelo majhni otroci lahko uporabljajo računalnik kot pripomoček pri učenju. Slika na levi strani kaže majhnega otroka, kako se igra z želvo, majhnim robotom, ki je povezan z mikroračunalnikom. Želve so drage in namenjene uporabi v šoli, toda načelo je preprosto: želva ima dve kolesi in pero. Otrok želvi pove, kako naj se pomika po papirju in ali naj pri pomikanju pušča za sabo sled ali ne. Tako otrok lahko riše. Želvi daje navodila, kako naj se obrača in katere točke naj povezuje. Otroci morajo premisliti, kako se mora želva pomikati, če želijo narisati določen lik, in se tako spoznavajo s temelji geometrije. Tak način samopomoči je bistvo metode logo. Med pedagogi vlada prepričanje, da se otrok več navadi hevristično (s poskušanjem in popravljanjem napak) kakor pa, če mu pokažemo, kako naj dela.

Dve šoli uporabe računalnikov se razlikujeta v načinu, kako uporabljamo računalnike pri delu z otroki. Pri devetih, mogoče desetih letih začnejo otroci delati z želvo na zaslonu računalnika. Rišejo zapletene like in učijo želvo, kako si lahko zapomni posamezne postopke. Ko otrok »uči« želvo, kako naj se giblje po papirju, v bistvu programira računalnik. Logo je jezik, ki otrokom omogoča pisanje programov, še preden so dovolj razviti za abstraktno premišljanje, ki ga zahteva večina računalniških jezikov. Igranje z želvo navaja otroke na delo z računalniki in jim pomaga spoznavati okolje.

Drugi način uporabe računalnikov pri delu z otroki izkorišča izrazito »potrpežljivost« računalnika pri učenju otrok. Otrokom, ki imajo težave pri razumevanju določene snovi, običajno pomagajo programi, ki delajo po načelu »vaja naredi mojstra«. Ti programi sprašujejo otroka in zbirajo podatke o tem, kako dobro otroku gre. Večina teh programov je zelo privlačnih, z dobro grafiko in zvočnimi učinki. Računalnik se nikoli ne utrudi in ne naveliča, tudi če daje otrok nepravilne odgovore. Ta potrpežljivost se je pokazala za zelo pomembno pri delu z otroki, ki jim učenje dela težave. Toda tak način uporabe računalnika ga postavlja v vlogo učitelja, in to nas pripelje do pomembne ugotovitve: računalnik ne more zamenjati človeka. Človeški stik je najpomembnejša prvina pri učenju, in čeprav je računalnik najmočnejši izobraževalni pripomoček, ne more nadomestiti učitelja.

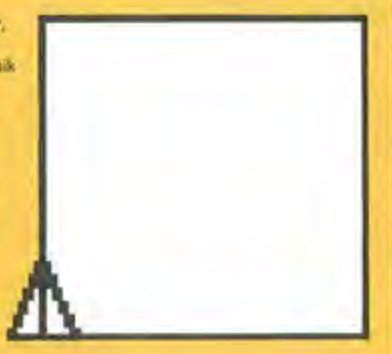
Računalniki so zabavni in prav je, da dovolimo otrokom, da se igrajo z njimi. Precej staršev skrbi, da bi nekatere igre negativno vplivale na razvoj otroka, toda večina igric je zelo zanimiva in za zdaj ni nič slišati o kakih negativnih posledicah.

Programski jezik logo

Tukaj bomo pokazali, kako se da jezik logo uporabiti za risanje likov na zaslon.
Logo je računalniški jezik, razvit, da lahko majhni otroci — stari štiri ali pet let — pišejo programe za računalnike. Program so razvili na Massachusettskem Institutu za tehnologijo (Boston) leta 1960 pod vodstvom matematika Sevmourja Peperta, ki je delal skupaj s svetovno znanim pedagogom Jeanom Piagetom.
Logo uporablja želvo, ki je majhen robot, ali pa svetlobni trikotnik na zaslonu računalnika. Ukaz FORWARD 10 (NAPREJ 10) povzroči pomik želve za deset enot naprej. Ukaz RIGHT 90 (DESNO 90) obrne želvo pod pravim kotom v desno. Ukazi se lahko združujejo tako, da želva nariše kvadrat, trikotnik, krog ali kak neobičajen lik. Želva se da naučiti, da si ukaze »zapomni«. Ne da bi se tega zavedal, otrok, ki vodi želvo, v bistvu programira računalnik.

Eden od načinov, kako se da narisati pravokotnik

```
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
```



Sestava ukaza ŠKATLA

```
TO BOX
REPEAT 4 [FORWARD 50 RIGHT 90]
END
```

Ukaz ZVEZDA

```
TO STAR
REPEAT 12 [BOX RIGHT 30]
END
```

Druge slike se dajo preprosto narediti z uporabo ukaza ŠKATLA in podobnih ukazov.

Prenapenjanje oči

Nekateri odrasli ljudje, ki veliko delajo z računalniki, trpijo zaradi prenapenjanja in utrujenosti oči. Otroci ne zdržijo toliko časa za zaslonom, da bi lahko imeli podobne težave. Problem pri odraslih je pojav, da se oči ustalijo na določeni žariščni razdalji in potrebujejo nekaj časa za prilagajanje novim razmeram. Če mislite, da bo vaš otrok tako vnet za računalništvo, da bo sedel za zaslonom nepretrgano dalj časa, potem se težavam lahko ognete s tem, da mu svetujete krajše odmore vsakih petnajst minut.
Potrebno je omeniti še tehnični problem, ki zadeva televizijske zaslone. Ugotovljeno je, da nekateri barvni aparati, narejeni pred letom 1970, oddajajo nizko dozo sevanja, in to je lahko nevarno, če jih uporabljate za redno delo. Če uporabljate za delo barvni televizijski zaslon, se splača prepričati, če ni bil narejen pred letom 1970.

Otroci, mlajši od sedmih let, potrebujejo pomoč in nadzor pri vklopu računalnika in televizijskega aparata ter pri nalaganju programov. Če je program dovolj dober, lahko pustimo otroka, da dela sam, čeprav je samostojno delo otroka odvisno tudi od tega, kako dobro zna brati in odgovarjati na računalnikova vprašanja. Kar se tiče opreme, so zahteve precej preproste. Mikroračunalnik mora biti čvrst. Otroci s pestmi tolčejo po tipkah, vlečejo za električne povezave in packajo po zaslonu. Če je sistem netrpežen, slabo sestavljen in pretežak za uporabo, za otroke ne bo zanimiv. Nekateri strokovnjaki menijo, da mora imeti tipkovnica za majhne otroke velike, razločno označene tipke. Po sedmem letu starosti pa je otrokova motorika že tako razvita, da lahko dela na normalnih tipkovnicah. Tipkovnice, občutljive za dotik, ki jih imajo najcenejši računalniki Sinclair, niso priporočljive za otroke mlajše od devetih, desetih let, medtem ko so izpeljanke tiskanih tipkovnic, na primer za računalnik Philips Videopac 700, primerne že za štiriletne otroke.

Mladi programerji

Izbira programov za majhne otroke ni lahka. Če nameravate uporabljati sisteme, ki delajo s kasetami, boste morali nadzorovati vsa nalaganja programov in shranjevanja podatkov. Če imate sistem z diskom, boste opazili, da so otroci zmožni prav dobro ravnati z njim. Ena najboljših oblik shranjevanja programov za otroke do sedmega leta so ROM vtični moduli, plastične plošče, ki vsebujejo integrirano vezje z vpisanim programom. Slaba stran teh sistemov je, da ne omogočajo shranjevanja podatkov, toda vtični moduli so skoraj neuničljivi in omogočajo majhnim otrokom uporabo računalnikov brez nadzora staršev.

Če se boste odločili in svojemu otroku kupili računalnik, skušajte poiskati stalno mesto zanj. Prestavljanje računalnika iz sobe v sobo ter stalno vklapljanje in izklapljanje računalniku sicer ne bo škodilo (če vam ne pade na tla), vendar lahko med tem časom otrokovo pozornost pritegne kaj bolj zanimivega recimo televizija.

Vzorno delovno mesto

Najbolje bi bilo, če bi otrok imel računalnik in televizijski zaslon v svoji sobi. Če želite otroku vzbuditi pravilen odnos do računalnikov, mu morate zagotoviti televizijski zaslon izključno za njegovo uporabo. (Računalnik postavite v sobo najstarejšega otroka, tako da bo lahko delal, ko bodo šli drugi otroci spat.) Črno-beli televizijski zasloni so kar primerni za delo z računalniki. Veliko so razpravljali o vrednosti barv na zaslonu takrat, ko z računalnikom delajo otroci. Nekateri poznavalci trdijo, da so barve izjemnega pomena, drugi pravijo, da so barve dobrodošle, niso pa nepogrešljive. Zdi se očitno, da je bolje, če je računalnik stalno priključen na črno-bel televizor, kakor da ga občasno priklapljate na barvni zaslon v dnevni sobi.

Če imate možnost za ureditev stalnega ali skoraj stalnega mesta za računalnik v eni od otroških sob, potem je pametno, če lahko stvari uredite tako, da se bo dal računalnik prestavljati, ne da bi bilo treba razdirati povezave in priključke. Ko pripravljate mizo ali klop, je najbolje, da pričvrstite vsa vodila in povezave tako, da jih otroci ne bodo mogli po nesreči trgati. Zagotoviti morate, da bodo vsi glavni priključki varni in pričvrščeni. Televizijskega zaslona ne priklapljajte na anteno, saj se lahko zgodi, da »pomotoma« na zaslon zaide film, namenjen odraslim v času, ko bi morali otroci že spati. Pomembno je, da je mikroračunalnik stabilen in da ne pleše sem ter tja. Za zelo lahki računalnik Sinclair je moč dobiti poseben podstavek. Če so vaši otroci živahni, si lahko omislite sponke ali kak drug način pričvrstitve računalnika. Seveda bodo računalnik, ki ste ga kupili svojemu otroku, uporabljali tudi drugi družinski člani, zato se izplača kupiti dvojna vodila in (če je potrebno) drug napajalnik za računalnik. Ti so precej poceni in vam bodo omogočili, da pošljete otroke spat, izklopite računalnik in kasetni snemalec (ali diskovno enoto) in oboje odnesete v svojo sobo. Če pa se bo pogosto dogajalo, da boste otrokom vzeli računalnik takrat, ko bodo želeli delati oni, se lahko zgodi, da bo njihova želja za delo z računalnikom v prostem času zamrla, še preden bodo začeli resnično delati.

Big Trak

Videti je kakor igrača, v resnici pa je močan vzgojni pripomoček. BIG TRAK je igrača, ki se da programirati. Otroku omogoča programiranje pomikov, ki jih naredi. Tank si lahko zapomni 16 korakov in je lahko programiran tako, da se sprehaja po sobah in se potem vrne v »bazo«. Otrok se zabava, računalnik pa mu pomaga spoznavati svet in snovati korake, ki so potrebni za pisanje manjših programov. Kljub njegovemu napadalnemu videzu imajo dekleta BIG TRAK rada prav toliko kot fantje.

Mikrosvetovi

Računalnike res že na široko uporabljamo v šolah, vendar so računalniški programi za pouk večinoma le učbeniki elektronike. Logo je drugačen — z uporabo računalnika ustvarja novo vrsto »okolja za učenje«

Želva, ki hodi

Želva je bila zasnovana, da bi pomagala pri premišljanju — posebno v procesu učenja geometrije in odnosov v prostoru. Kadar otroci niso prepričani, kako želvi opisati gibanje, ki ga zahtevajo, se navadno sami postavijo v njeno vlogo, tako da med hojo ubogajo ukaze logo. To daje pri učenju veliko večji občutek resničnosti.

Odkar se je l. 1977 pojavil prvi mikroračunalnik, daljnovidni vzgojitelji odkrivajo možnosti njegove uporabe pri pouku. V razvitem svetu ima že skoraj vsaka šola vsaj po en mikroračunalnik, mnoge od njih pa so v učne programe uvrstile tudi računalništvo. Tudi v Sloveniji smo razmeroma zgodaj — že leta 1971 in prej kot marsikatera druga evropska država — pričeli z veliko vnemo uvajati pouk računalništva v srednje šole, izobraževati učitelje ter pisati učbenike, vendar je bila ta dejavnost zaradi pomanjkanja opreme do danes na večini šol omejena na klasične metode pouka o računalnikih.

Vendar pa ni mikroračunalnik za zdaj tudi v razvitem svetu bistveno spremenil tradicionalne metode pouka. Najmočnejši dokaz za to je množica mikroračunalniških šolskih programov za hišne računalnike, ki jih v splošnem »odlikuje« presenetljivo pomanjkanje domišljije. Večinoma jih lahko opišemo kot »elektronske knjige«, s katerimi računalnik najprej predstavi učencu zaporedje »okvirjev« (ki ponazarjajo strani v učbeniku iz papirja) na zaslonu, potem pa z zaporedjem izbirnih vprašanj preizkusi, v kakšni meri je učenec snov dojel. Seveda lahko računalnik odgovore tudi oceni.

Na hišnem računalniku je take programske pakete lahko napisati in besedilo opremiti z barvitimi in morda celo animiranimi — grafičnimi ilustracijami. To pa je le avtomatizacija obstoječega načina učenja, ne pa bistveno nova, drugačna uporaba zmožnosti mikroračunalnika.

Logo je drugačen. Profesor Seymour Papert, ki ga je razvil na Inštitutu za tehnologijo v Bostonu (MIT), ga je označil kot filozofijo šolanja in kot družino računalniških programskih jezikov za izvajanje te filozofije.

Mnogi ljudje logo napačno razumejo le kot programski jezik. Njegove ukaze in zmožnosti primerjajo z basicom in ugotavljajo, da je logo za začetnike veliko boljši jezik. V tem primeru zgrešijo bistvo tega jezika. Papert s svojim sistemom nikoli ni nameraval razviti metode za poučevanje programiranja, temveč je hotel ustvariti okolje, v katerem bi se otroci lahko učili različne predmete — pravzaprav okolje, v katerem bi se otroci lahko učili, kako naj se učijo.

Ta filozofija v precejšnji meri izvira iz trditve uglednega švicarskega filozofa Jeana Piageta, da se v primernem okolju otroci vsega lahko sami naučijo, kakor se sami naučijo hoditi in govoriti. Seveda je Piaget postavil le teorijo. Papert pa se je odločil ustvariti praktično okolje za piagetovsko učenje.

Da tega ni mogoče doseči s tradicionalnimi metodami šolanja, očitno potrjuje dejstvo, da se odrasli učenja večinoma bojijo in ne uživajo ob misli, da bi se morali naučiti novih spretnosti ali novih področij znanja. Najpogostejši — tako trdi Papert — je splošen strah pred matematiko ali »matofobija«, kakor temu pravi.

Eden od razlogov za strah pred učenjem je, da večino predmetov poučujemo na enak način, medtem ko so njihove uporabe popolnoma različne. Otroke učimo, na primer, množenje na enak način kot glavna mesta sveta — z učenjem na pamet. Proces učenja se je ločil od predmeta učenja, namesto da bi ostal z njim neločljivo povezan.

Papert sam šteje učenje vsake nove spretnosti, pa naj bo to letenje, kuhanje ali znanje tujega jezika — za konjička. Ta svoj odnos pripisuje izkušnji iz svojega otroštva, ko je že zelo mlad odkril, kako delujejo zobniki, in ko je potem ta koncept uporabljal pri vsakem novem problemu.

Tudi Albert Einstein je pravil, da se je nerazumljivih reči lotil tako, da jih je razdrobil v tiste pojme, ki jih je dojel do svojega petega leta.

Kako so ti močni pojmi vdelani v logo, spoznamo, ko si ogledamo njegovo delovanje. Prva pomembna reč za logo je želva, ki so jo zasnovali kot »pripomoček za razmišljanje« na enak način, kot je Papert uporabljal zobato kolesje. Želva ima za majhne otroke obliko

Kako se učimo sukanja

Ta namišljeni, vendar tipični primer kaže, kako logo spodbuja skupino otrok k reševanju doslej neznanih problemov.

```
ZASUKATI
PONOVI 80
  NAPREJ 1
  DESNO 1
KONEC
```

»Če želimo list, moramo narisati krivuljo.«
»Ampak želva se vedno premika naravnost!«
»Kaj pa, če bi jo pognali malo naprej, jo samo malo zasukali, jo spet pognali malo naprej, pa spet malo zasukali in to ponavljali! — Dobili bi krivuljo.«
»V redu, najmanjša razdalja je ena in najmanjši kot je ena. Poskusimo to napraviti osemdesetkrat.«

```
ZASUK
ZASUK
```

»Tako, natanko to želimo.«
»Dve krivini pa bosta dali ravno list — poskusimo!«

```
CVETETI
  ZASUK
  DESNO 90
  ZASUK
KONEC
CVET
```

»To pa ni noben list. Kaj se je zgodilo?«
»Želva je kar nadaljevala od tam, kjer je nehala — morali bi ji ukazati, naj gre v drugo smer.«
»Ampak za koliko naj jo zasučemo?«
»Poskusimo za devetdeset — to je navadno dobro.«
»In naj bo to nova beseda — LIST — da bomo prihranili čas.«

```
CVETETI
  ZASUK
  DESNO 100
  ZASUK
KONEC
CVET
```

»Tole je boljše, ampak devetdeset ni bilo dovolj. S čim naj poskusimo zdaj?«
»Najprej raje premislimo, namesto da bi ugibali.«
»Aha — naučili smo se, da če želva pride okrog česa, se potem obrne za tristo šestdeset enot.«
»No, vemo, da se zasuče za osemdeset stopinj na prvi krivini, zato se mora tudi med vračanjem zasukati za osemdeset — torej skupno za sto šestdeset.«
»To pomeni, da se mora zavrteti za dvesto stopinj na konici lista.«
»Ne, ker se mora obrniti tudi na drugem koncu, da bi prišla nazaj.«
»Torej bi morali poskusiti s polovico od dvesto.«
»Imenitno — štiri take bodo ravno dovolj za cvet.«

```
LIST
LIST
LIST
LIST
```

```
CVETETI
PONOVI 4
  LIST
  DESNO 10
KONEC
CVET
```

»To pa ni dobro, cvet je postrani.«
»Pozabili smo vstaviti zasuke med listi.«
»Zakaj pa jih potem ni narisala drugega nad drugim?«
»Ker je želva po risanju enega lista obrnjena za sto enot na levo od začetka smeri.«
»Listi so torej vsakič zasukani za sto na levo.«
»To je videti skoraj v redu — le da želimo samo devetdeset; zato dodajamo med vsakim listom zasuk za deset v desno.«

```
CVET
DESNO 180
NAPREJ 100
DESNO 180
LIST
```

»Končno! Želvo moramo zasukati na desno, da narišemo steblo.«
»Sto bi moralo biti dovolj dolgo.«
»Na koncu stebla dodajmo še list — lahko je kar enak cvetnemu.«
»Tokrat pa ne smemo pozabiti na kot — želvo je treba zasukati na desno.«

posebnega talnega robota, ki je povezan z mikroračunalnikom in ga lahko premikamo po tleh z vtipkavanjem ukazov logo. Želva običajno nosi pisalo za risanje poti, opremljena pa je lahko tudi z majhnim zvočnikom in detektorji za trke, s katerimi jo je mogoče voditi skozi ovire.

Otroci običajno napredujejo od uporabe talnih želv k zaslonskim, ki jih je mogoče premikati po zaslonu računalnika. Želva je močan pripomoček, s katerim se otroci najprej naučijo temeljnih pojmov prostorskih odnosov, potem pa jih razvijejo do geometrije.

Vodenje želve je sicer samo delček uporabe sistema logo, ker pa je vizualno najbolj zanimiva, se o njej tudi največ piše. Pomembnejši je koncept gradítve preprostih zamisli v bolj zahtevne in narobe: koncept drobljenja večjih problemov v manjše, in sicer takšne, ki smo jih že obravnavali.

Te procese jasno vidimo pri zamišljenem pogovoru skupine otrok, ki se učijo voditi želvo tako, da bi narisala cvetico (glej sliko v okviru). Otroci najprej poznajo samo tri ukaze: NAPREJ, ki želvo premakne naprej za določeno razdaljo; DESNO, ki želvo zasuče za določen kot na desno; ter ukaz PONOVI za večkratno ponovitev izvrševanja ukazov, ki so zapisani v program.

Iz teh osnovnih idej si otroci najprej naredijo »orodje«, program, za risanje krive črte (ZASUKATI...KONEC). Vse to zaporedje lahko sedaj pokličejo preprosto z vtipkavanjem ukaza ZASUK. Podobno lahko otroci po poskušanju in nadaljnjem učenju razvijejo ukaz LIST, ki se opira na ukaz ZASUK. Nazadnje pridejo v tem razvoju do ukaza CVET, ki nariše celotno cvetico.

Logo sicer ni edini jezik, ki omogoča tako snovanje ukazov (drugi je na primer forth), je pa edini, ki je namenjen majhnim otrokom. Očiščen je mnogih formalnosti in postopkov, ki jih zahteva programiranje v drugih jezikih. Cilj opisanega postopka pri logu je pač v tem, da naj se otrok ne zaveda, da programira računalnik, temveč samo, da rešuje problem.

V nekaterih učnih situacijah učenja učenec niti ni udeležen na tej stopnji. Učitelj uporabi logo, da zasnuje splet močnih orodij, ki se vsa nanašajo na določen učni predmet ali področje znanja. Nato lahko učenec z uporabo teh orodij raziskuje ta predmet in sam odkriva njegove zakonitosti. Takim področjem pravimo »mikrosvetovi« — omejena okolja, v katerih z računalnikom upodobimo kaj iz resničnega sveta ali kako področje znanja.

Najboljši primer mikrosvetov je verjetno logov model Newtonove fizike. Čeprav Newtonov prvi zakon pravi, da se telo brez vpliva zunanjih sil giblje v premi črti in z enakomerno hitrostjo, pa mladi misleci hitro ugotovijo, da se v resničnem svetu vse upočasni. To povzroča zavoro pri učenju. S sistemom logo pa lahko zgradimo mikrosvet, v katerem se vse vede v skladu z Newtonovim zakonom. In z uporabo orodij za premikanje objektov po zaslonu se otroci kmalu sami naučijo vseh treh Newtonovih zakonov.

Logo je močna zamisel, ki se je je vsekakor vredno učiti ob hišnem računalniku. Talne želve je sicer na Zahodu mogoče kupiti, niso pa poceni. Zato so tembolj dobrodošle verzije sistema logo z zaslonskimi želvami, ki jih v zadnjem času imajo nekateri priljubljeni hišni računalniki.

Prenosni urejevalnik
Urejanje besedila hitro postaja eden najbolj priljubljenih načinov uporabe računalnikov. Novi računalniki imajo lastnosti posebej prilagojene tej funkciji. Med njimi so zaslon z 80 stolpci (da lahko pokažejo vso širino tipkanega pisma), vdelane disketne enote (včasih je disketa s programom za urejanje besedila celo všteta v ceno) in tipke, ki jih lahko programiramo. Nekatere naprave, kakor je Ajile na sliki, so prenosne — idealne za novinarje in poslovne ljudi, ki veliko potujejo.

Lepa beseda

S pravo programsko opremo se naš računalnik spremeni v urejevalnik besedila, ki ureja, organizira in shranjuje besede in celo popravlja nepravilno črkovanje

Urejanje besedila je ena najbolj koristnih nalog, ki jih lahko opravlja računalnik.

Mogoče bi bilo boljše programe in opremo, ki jo zajema izraz »urejanje besedila«, imenovati »strojepis s pomočjo računalnika«. Z dodatkom tiskalnika lahko namreč na večini hišnih računalnikov izvajamo neko obliko urejanja besedila. Lastnik računalnika bo lahko običajno šele po poskusnem delu ugotovil, kako zelo uporabni so taki programi.

Kadar računalnik uporabljamo kot urejevalnik besedila, slednji izpisuje besede na zaslonu tako, kakor jih vpisujemo, podobno kot bi jih pisalni stroj pisal na papir. Večji mikroračunalniki lahko na zaslonu, ki predstavlja »stran«, napišejo po 80 znakov v eni vrstici. Pri manjših računalnikih mora biti uporabnik bolj potrpežljiv. Zadovoljiti se mora z dosti ožjim zaslonom in nekateri modeli ga prikrajšajo za male črke. Upoštevati mora tudi dejstvo, da lahko manjša naprava shrani le omejeno količino besedila.

Programi za urejanje besedil ponujajo vrsto dognanih pripomočkov. Vsi programi zaznajo konec vrstice in samodejno skočijo v novo vrstico, pri tem pa

prenesejo celotno zadnjo besedo na začetek nove vrstice. To pomeni, da pri tipkanju ni treba več paziti na vračanje valja na koncu vsake vrstice. Namesto tega lahko tipkamo v nedogled, program pa skrbi za urejanje teksta v vrstice. Seveda pa moramo pred začetkom novega odstavka pritisniti tipko RETURN.

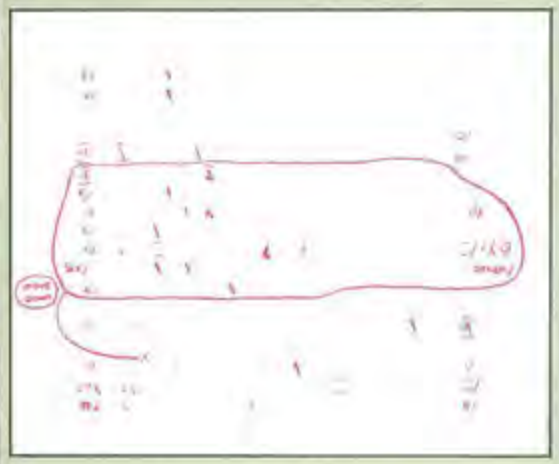
Pri običajnem pisalnem stroju moramo biti pripravljeni na mehanično popravljanje napak, običajno tako, da prebarvamo napako in ponovno natipkamo besedo. To pa je videti precej nemarno. Pri več napakah imamo le dve možnosti: da odpošljemo popackano pismo, ali da začnemo znova. Z urejevalnikom besedila tega problema ni več. Utripajoči kurzor na zaslonu, kakor običajno, kaže našo trenutno pozicijo. Pomikamo ga lahko nazaj preko teksta, dokler ne kaže na mesto napake. Zatem lahko napako izbrišemo in napišemo pravilen tekst.

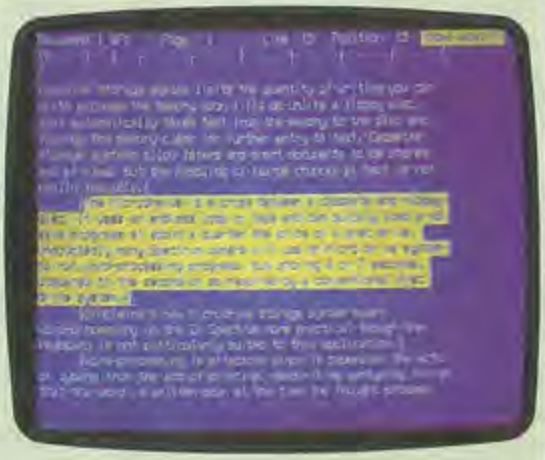
Ko spoznamo moč urejevalnika besedila, postanemo bolj premišljeni pri pripravi teksta. Z ukazom INSERT lahko na primer dodamo besedo, cel stavek ali odstavek ravno tako preprosto kakor eno samo črko. To nas opogumi, da še enkrat pretehtamo, kaj smo napisali v pismu ali dokumentu, in ga po potrebi brez težav spremenimo ali dopolnimo. Tudi brisanje besedila je zelo preprosto. Ukaz kar izbriše nezaželene črke ali besede z zaslona, preostalo besedilo pa prestavi tako, da dobimo lepo urejeno stran. Veliko poklicnih piscev in časnikarjev danes uporablja urejevalnike besedila in večina zatrjuje, da sta se izboljšali tako kakovost kakor količina njihovega dela.

Pogled v menu
Fotografija prikazuje »menu« kakovostnega urejevalnika besedila. Menu se pokaže na zaslonu v trenutku, ko naložimo v računalnik program, ki nas vodi do posameznih funkcij, ki jih urejevalnik opravlja. Primeri takih funkcij so: tabulator, urejanje robov, razmik med vrsticami, števec besed v dokumentu, preurejanje odstavkov in izdelava kazala.

količino teksta, ki ga lahko obdelujemo. Težava pri ZX81 je tudi njegova senzorska tastatura, ki onemogoča vsak resen poskus hitrega tipkanja. Čeprav je tipkovnica na ZX Spectrumu znatno boljša, še vedno ni mehanična, na kakršno so navajene strojepiske. Če razmišljate o nakupu računalnika za urejanje besedila, dobro preglejte tipkovnice naprav, ki prihajajo v poštev, ker te močno vplivajo na hitrost in udobnost tipkanja (glej stran 86).

V angleščini ločijo programe po zmogljivosti in jih tudi imenujejo različno. Enostavni programi, kot npr. za ZX81, se imenujejo »text editing programs«, zapleteni programi pa se imenujejo »word processors«.





Elektronsko urejanje

Glavna prednost računalniških urejevalnikov pred pisalnim strojem je njihova neverjetna prilagodljivost in hitrost. Tam, kjer ima napaka v tipkanem tekstu za posledico mukotrpno urejanje in popravljanje, opravi urejevalnik delo tako rekoč z elektronsko hitrostjo. Poleg tega lahko vse operacije izvajamo na tekstu, ki je prikazan na zaslonu. Kakovostni sistem zna: iskati besede v besedilu in jih menjati, prestavljati vrstice, avtomatsko popravljati napake v celotnem besedilu in celo popravljati napačno črkovane besede. V resnici nam urejevalnik lahko prihrani toliko časa, da je začel postopoma nadomeščati običajni pisalni stroj. Primerjajte zmešnjavo, ki jo povzroči popravljanje besedila na levi s čistim elektronskim urejanjem.

Celo najmanjši mikroračunalniki lahko nudijo neko stopnjo urejanja besedila. Sinclair ZX81 lahko izvaja preprost program za urejanje besedila, ki nam omogoča, da na zaslonu napišemo pismo in popravimo napake. Vendar so možnosti tega programa omejene, ker lahko obdeluje le eno ali dve strani besedila, ne pa tudi daljših dokumentov. Majhnost pomnilnika v ZX81 in podobnih malih računalnikih močno omejuje

Računalnik s pomnilnikom 16 kb ali 32 kb omogoča že kar uporabno obdelavo besedila. Potem ko si omislimo tiskalnik, postane glavni problem opreme velikost pomnilnika, v katerega shranjujemo besedilo. Mogoče je tudi izvajati program za obdelavo oz. urejanje besedila, ki uporablja kasete za dodatno shranjevanje teksta. Vendar uporaba kaset omejuje količino teksta, ker se pomnilnik računalnika hitro

napolni. Nasprotno se pri uporabi diskete tekst avtomatsko prenaša iz spomina na disketo in s tem briše pomnilnik računalnika ter ga pripravlja za vpis nadaljnjega teksta. Uporaba kaset omogoča shranjevanje in tiskanje pisem in krajših dokumentov, obdelava velikih tekstov pa z njihovo uporabo ni mogoča.

Z uporabo novega sistema z Microdrivi je postalo urejanje besedil na ZX Spectrumu bolj priročno. Microdrive je križanec med kaseto in disketo. Uporablja neskončno zanko traku in lahko hitro shranjuje in nalaga programe in podatke, pri tem pa stane le četrtino cene disketne enote. Nedvomno bodo mnogi lastniki Spectrumov uporabljali Microdrive za izvajanje programov za urejanje besedil, pa čeprav shranjevanje in nalaganje strani traja okoli šest sekund v primerjavi z eno sekundo pri običajni disketni enoti. Urejanje besedila je učinkovito, ker ločuje sestavljanje besedila od tipkanja. Tako pri rokopisu kakor pri tipkanju pišemo besede na papir medtem, ko jih miselno oblikujemo. Pri računalniškem urejanju besedila pa se na papirju ne prikaže niti beseda, dokler teksta na zaslonu popolnoma ne uredimo. Seveda pa za to, da spravimo tekst na papir, potrebujemo tiskalnik, in ceneni tiskalnik, ki ga uporabljamo za izpis programov, običajno ni dovolj kakovosten, da bi izdelal zadovoljivo kopijo pisma ali pesmi. (Glej lastnosti tiskalnikov.)

Proizvajalci si prizadevajo, da bi naredili cenene sisteme za urejanje besedil in danes lahko za nekatere hišne računalnike že dobimo programe, ki so shranjeni v integriranih vezjih, ki jih kar vtaknemo v podnožja na tiskanem vezju računalnika. Ti programi so zelo priročni, kadar nimamo na voljo disketne enote. Njihova prednost je v tem, da so v trenutku na razpolago in jih ni treba nalagati v računalnik s kasete ali diskete. Če ima računalnik dovolj velik pomnilnik (od 32 kb navzgor), bomo lahko s tem programom izdelali dokument, ki bo vseboval do 5000 besed in ga urejali po mili volji.

Če želimo urejeni tekst, potem ko smo ga natisnili, spraviti, ga moramo zapisati na kaseto: postopek nam bo vzel nekaj minut. Vezje, ki vsebuje program za procesiranje teksta, si ne more zapomniti teksta. Če pa želimo z našim urejevalnikom besedila napisati novelo, moramo vedeti, kako dobro zna program obdelovati velike kose teksta.

Nekateri dognani programi znajo opravljati vrsto posebno uporabnih nalog. Med najbolj popularnimi so avtomatski slovar, preskušanje črkovanja in iskanje ter zamenjava besed. Seveda pa za to potrebujemo disketni sistem. Slovar preskuša besede v dokumentu tako, da jih primerja z besedami v pomnilniku ali na disketi. Če naleti na besedo, ki je ne spozna, jo označi in nas opozori, da jo popravimo. Program zna tudi poiskati določeno besedo v tekstu. Pri tem postavi na besedo kurzor in nam s tem omogoči popravke in spremembe besede. Če nam v besedilu kaka beseda ni všeč, bo program to besedo v celem besedilu zamenjal z drugo in potem besedilo lepo uredil.

Večina urejevalnikov besedila skrbi tudi za lepo obliko tiskanega besedila tako, da vanj vstavlja dodatne presledke in tako besede na levem in desnem robu lepo poravna.

Uporaba urejevalnikov se hitro širi in zelo verjetno je, da se bo poznavanje njihove uporabe uveljavilo kot bistveno pisarniško znanje.

MIKRORAČUNALNIK	CENA	PROGRAMSKA OPREMA	NA KASETI	NA DISKETI	PREMIČNA TIPKOVNICA	CENA
Sinclair Spectrum	£125	Wordprocessor	●			£9.95
Commodore Vic 20	£130	Vicwriter	●		●	£19.95
Dragon 32	£175	Gemini	●		●	£19.95
New Brain	£269	Propen	●			£40.24
Commodore 64	£299	Visawrite		●	●	£79
Atari 800	£300	Keyword		●	●	£115
BBC Micro	£399	View		●	●	£59.80
Epson HX20	£486	Deckmaster	●		●	£34.50

Cena obdelave besedil

Gornja tabela prikazuje izbor paketov za urejanje besedil za vrsto mikroračunalnikov vseh razredov. Če nameravate dolgo uporabljati urejevalnik besedila, pazite na to, da boste udobno tipkali na tipkovnico. Kalkulatorske in membranske tipkovnice so bile izdelane zato, da bi imele nižjo ceno, in so primerne le za igro in pisanje kratkih programov. Prednost imajo tudi računalniki s tipkami, ki jih lahko definiramo. Te se večinoma uporabljajo na boljših sistemih in zmanjšujejo število ukazov, ki jih moramo natipkati. Cene v splošnem ustrezajo tistemu, kar lahko pričakujete, da boste za program plačali. Očitno je tiskalnik nujno potreben, če želimo imeti tiskane kopije teksta. Tiskalniki se močno razlikujejo po hitrosti in kvaliteti tiskanja. Poskrbite, da se bosta urejevalnik in tiskalnik ujemala za vse naloge, ki jih želite izvajati. Če želite samo pisati pisma, bosta zadostovala urejevalnik besedila in tiskalnik za zmerno ceno. Za daljše tekste pa bo seveda potrebna dražja kombinacija.

Preglednica

	A	B		N
1	JANUAR	FEBRUAR	1	SKUPNO
2	42.41	18.75	2	388.4
3	160.35	149.89	3	1732.7
4			4	

Področje uporabe
Preglednica je razdeljena na stolpce in vrstice. Kjer se stolpec in vrstica križata, dobimo polje, ki ga imenujemo celica. Celice adresiramo z njihovimi koordinatami (A1, B3 itd.). Vsaka celica lahko vsebuje naslov (npr. januar), številko (npr. 149.89) ali pa obrazec. Celica N2 ima obrazec SUM (A2:M2), kar pomeni vsoto zgornje vrste številk od januarja do decembra. Rezultat tega računa je prikazan: 388.4. V našem primeru je preglednica razdeljena v dve okenci, tako da podatkov za mesece od marca do decembra ne vidimo.

S programom »preglednica« je naš računalnik zelo koristna naprava, saj nam pomaga napovedovati in predvidevati, kako bomo razpolagali z denarjem pri raznih finančnih odločitvah

Ugotovljeno je, da porabijo poslovneži okrog 30 odstotkov svojega časa za urejanje denarnih zadev. Največkrat uporabljajo kos papirja, ki je približno tako velik kakor dve strani te knjige. Razdelijo ga na celo vrsto stolpcev. Vsak stolpec označijo s številko posameznega meseca, medtem ko vpisujejo vsakovrstne izdatke po levi strani navzdol. V vsak stolpec vpisujejo mesečne stroške za posamezne kategorije izdatkov. Tako lahko seštejejo izdatke v določenem stolpcu in dobijo celoten mesečni izdatek, ali pa seštejejo vrstice za posamezne izdatke, da dobijo znesek, ki so ga porabili za določeno vrsto izdatkov v vsem letu.

Težave nastajajo, če kdo načrtuje preveliko ali pa, to je še slabše, premajhno porabo. Tedaj se mora vračati v tabelo in spreminjati številke za razne izdatke v različnih mesecih toliko časa, da dobi zadovoljiv končni rezultat.

S preglednico pa lahko takoj preračunamo celo stran številk, če se je spremenil eden od temeljnih podatkov. Vzemimo za zgled, da so se nam v mesecu januarju podražili prevozni stroški. Zaradi te spremembe so se nam povečali celotni izdatki za ta mesec, kakor tudi vsi izdatki za prevoz. Vse to lahko izračunamo le s pritiskom na tipko! Zato ne preseneča, da so preglednice najbolje prodajana programska oprema na svetu!

Tako kakor večina druge programske opreme je tudi preglednica običajno pisana s programskimi moduli (overlay), kar pomeni, da je med izvajanjem v delovnem pomnilniku računalnika le del programa. Če gledamo na program kot na množico podprogramov, to pomeni, da so med izvajanjem programa določeni podprogrami shranjeni na eni od zunanjih spominskih enot, kakor je to disketa ali morda magnetni trak. Računalnik med izvajanjem programa sam kliče ustrezne podprograme iz zunanje spominske enote ter jih vpiše v delovni pomnilnik. Računalnik prekrije določene podatke, ki jih ne potrebuje več v delovnem pomnilniku, s podprogramom, ki ga je dobil od zunanje pomnilniške enote. Hiter premislek nam pove, da lahko tako izvajamo precej obsežne programe na računalniku, ki ima sorazmerno skromno zmogljivost delovnega pomnilnika. Slabost takšnega izvajanja programa pa je ta, da je nekoliko bolj zamudno, ker je prenos podatkov iz zunanjega v delovni pomnilnik precej počasen.

Programski paketi, ki jih običajno srečujemo pod imenom »preglednica«, so prirejeni za veliko večino hišnih in poslovnih računalnikov. Eden najuspešnejših programov te vrste glede na število prodanih je program Visicalc, ki je bil v izvirniku pisan za računalnik Apple II in ga je bilo mogoče kupiti že sredi leta 1979. V svetu mikroračunalniške programske opreme se nikoli počasi ne odzivajo ob uspehu kakšnega programa. Tudi ta program ni bil izjema. Kmalu je bilo mogoče kupiti najrazličnejše programe za različne tipe računalnikov pod skupnim imenom »preglednice«, pa naj so bili uporabni ali ne.

Za uspešno uporabo mora preglednica imeti dve lastnosti: primerno velikost in pa ustrezno oblikovanje z dobrimi kontrolnimi ukazi. Naj glede velikosti poudarimo, da vidimo na zaslonu samo določeno »okno« celotne tabele. Pomembna pa je celotna velikost tabele.

To pa pomeni, da vsak stroj ni primeren za dobro uporabo preglednice. Za poslovne namene so minimalne zahteve za izvajanje preglednice, da ima računalnik vsaj 32 kb delovnega pomnilnika in zaslon z najmanj 80 znaki, medtem ko za domačo rabo verjetno zadošča že 40 znakov.

Uporabniki hišnih računalnikov lahko kupijo tovrstno programsko opremo na kasetah za različne računalnike, kakor so Spectrum, VIC-20 itd. Čeprav so ti računalniki nekoliko manj zmogljivi, so dovolj uporabni.

Ker nam »preglednice« lahko odgovarjajo na vprašanja »Kaj če...?«, jih lahko uporabimo za postavitev preprostih računalniških modelov ali simulacij. Tam je že opisano, kako dela sistemski analitik. Za podobno natančno načrtovanje lahko uporabimo preglednico. Baza podatkov, ki smo jo omenili, je sestavljena iz množice podatkov, ki so urejeni tako, kakor želi uporabnik. Program za urejanje besedila (»word

Kurzor označuje izbrano polje N2. Podatki, odtipkani na tipkovnici računalnika, se bodo prikazali na označenem polju. Celotno vsebino tega polja pa lahko prikažemo s posebnim ukazom, ki ga v tem primeru vidimo v spodnjem desnem kotu te strani.

processor«), ki je tudi eden od zelo dobro prodajanih programov, nam omogoča pomikanje ene besede ali cele skupine besed med izvajanjem programa. Preglednica pa je nekoliko drugačna, saj zahteva od uporabnika, da sledi celotnemu izvajanju.

Če, recimo, želimo uporabiti preglednico za analiziranje naših življenjskih stroškov, lahko zberemo vse stroške bivanja: najemnino, davek, zavarovanje itd., jih seštejemo in rezultate vpišemo v večjo tabelo.

Vsaka posamezna lokacija v preglednici je določena kot celica. Za njeno določitev moramo adresirati njeni koordinati X in Y. V vodoravni smeri uporabljamo črke A — Z, AA — AZ ali pa morda še BA — BM, kar nam omogoča maksimalno dolžino preglednice — 65 celic v popularnih verzijah. Navpično pa lahko uporabljamo števila od 0 do 256. Vsaka celica lahko vsebuje naslov (npr. »prihodek« ali »dohodek«), vrednost, ki jo vpišemo ali dobimo iz računov (npr. 1000), ali pa matematično formulo za računanje te vrednosti (kot primer B4 + B6*B5). Ker zavzemajo formule v veliki večini primerov večji prostor, kakor pa ga lahko vidimo v našem kvadratku na zaslonu, jih običajno prikazujemo v ločeni vrstici na vrhu zaslona. Ob vpisovanju novih podatkov v preglednico bo vnaprej določena velikost celice, morda na 8 ali 9 digitov ali znakov. To je znano kot privzeta (osnovna) velikost, kajti običajno je dovoljeno skrajšanje ali podaljšanje celice, da bo primerna za račune, ki jih izvajamo. Nekateri tovrstni programi dopuščajo, da je širina levega stolpca večja od drugih, ker ga običajno uporabljamo za naslove oziroma opise. Ni se nam treba takoj odločiti, kako velike celice želimo imeti. Večina tovrstnih programov nam omogoča razširitev ali zmanjšanje le-teh tudi še takrat, ko imamo vpisane podatke. Če boste velikost zmanjšali, da bo manjša od dolžine, del, ki ga na zaslonu ne vidimo, ne bo zbrisan, le prikazan ne bo.

Zadnja pomembna sestavina preglednice so povezave za ukaze, s katerimi lahko pomikamo »okence« po celotni tabeli. Tako lahko vidimo na zaslonu katerikoli izrez iz tabele. S temi ukazi pa lahko tudi urejujemo in upravljamo izpis podatkov na zaslon, prav tako pa lahko opazujemo tudi njihov potek nastajanja. Velika večina takšne programske opreme dovoljuje široke možnosti za posege v podatkovno bazo: lahko brišemo, premikamo ali kopiramo cele stolpce ali vrstice, lahko izberemo posamezna »okenca« v tabeli ter jih združimo in obenem prikažemo na zaslonu. To je ugodno pri velikih tabelah, kjer so podatki, ki nas zanimajo, medsebojno precej oddaljeni. Prav tako pa posamezna okenca lahko neodvisno pomikamo po tabeli. S kurzorjem se lahko pomikamo po raznih celicah v preglednici. Obstaja pa še možnost, da s posebnim ukazom skočimo direktno v želeno celico. Tipke, s katerimi dajemo ukaze računalniku, so zlasti pomembne pri shranjevanju, brisanju in zaščiti podatkov, ker lahko te funkcije izvršujemo preprosto. Zato je izvajanje preglednice nekoliko počasnejše, vendar je to v primerjavi z vpisovanjem podatkov zanemarljivo. Tu velja pravilo, da pred vpisovanjem oziroma spreminjanjem podatkov vedno prej shranimo staro tabelo, kajti pri vpisovanju podatkov lahko naredimo napako.

Rezultate naše tabele lahko izpišemo s tiskalnikom. Prej pa moramo določiti, kateri del tabele želimo izpisati. S tiskalnikom lahko izpišemo samo »okence«, ki ga vidimo na zaslonu ali pa del celotne tabele. Za natis lista, ki je širši od vašega tiskalnika, je najbolje, da tiskate dvakrat in potem strani sestavite.

Uporaba več »okenc«, ki smo jo že opisali, nam omogoča hkraten prikaz dveh različnih delov tabele. Taka okenca lahko tudi natisnemo. To je posebno koristno pri vpisovanju podatkov, ker se lahko oziramo na prejšnje vpise. Večina preglednic omogoča

Kaj če...?
Če se spremeni vsebina katerekoli celice, preglednica avtomatsko preračuna vse druge celice, ki so na kakršenkoli način odvisne od te številke. Hitrost in lahkotnost tega procesa omogočata uporabniku hiter preskus pravilnosti celotnega stanja. Uporabnik lahko vnaprej predvidi svoj dohodek, če se spremenijo posamezni stroški. Oglejmo si spodnji primer prodajalca sadja.

Če se gorivo podraži za X %. potem se mesečni prevozni stroški povečajo za Y %. To povzroči zvišanje cene jabolk na veliko, ki se bo pokazala pri strankah v obliki višje cene...

uporabniku, da »drži« gornje ali prve (ali oboje) vrstice, kjer imamo običajno imena ali naslove.

Do sedaj smo imeli »preglednico« za tabelo, kjer lahko dobimo podatke zaporedno drugega za drugim, v vrstici ali v stolpcu. Če pa se odpovemo seštevanju posameznih vrstic ali stolpcev, prav lahko uredimo list v kakršnemkoli vzorcu, če ta vzorec uporabniku olajšuje premišljevanje. Ne smemo le pozabiti, da bo takšna umetelnost pri urejanju baze podatkov terjala še bolj temeljito sistemsko analizo kakor običajna.

Poslovno orientirane preglednice, ki jih dobimo pod komercialnimi imeni Visicalc, Supercalc in Masterplan, omogočajo uporabniku, da informacije iz »preglednice« s posebnim urejevalnim programom spreminja v tako obliko, ki nam da hitrejši pregled nad podatki. To so različni grafični prikazi, kot na primer grafikoni ali pa diagrami.

Prej smo že omenili možnost uporabe preglednic za uporabnike hišnih računalnikov za analizo domačih stroškov in načrtovanja. Druga zelo zanimiva uporaba je napeljevanje centralne kurjave. Pri le-tej je veliko spremenljivk, ki jih moramo upoštevati pri naših odločitvah: vrsta goriva, ki naj ga uporabljamo, število in vrsta radiatorjev, temperatura tople vode, poraba vode itd. Pri odločanju, kako optimalno ogrevati stanovanje ter predvideti še druge možnosti, nam lahko preglednice veliko pomagajo.

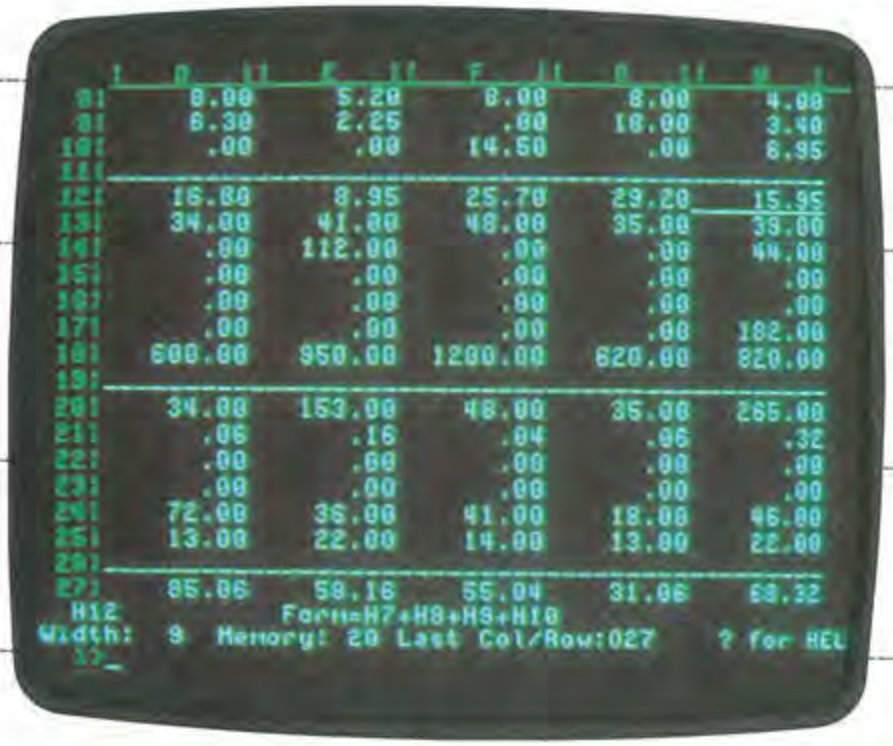
»Preglednice« za poslovne namene, ki se izvajajo na ustreznih računalnikih, zbujajo morda največ občudovanja s svojo hitrostjo delovanja. To prihaja od tega, ker je program zanje napisan v strojni kodi. Za male hišne računalnike pa so ti programi napisani največkrat v basicu, tako da smo včasih manj zadovoljni s hitrostjo. Morda je vredno pogledati, na kakšne težave bi zadel, kdor bi želel napisati podoben program v basicu, da bomo imeli vsaj približno predstavo o zapletenosti naloge.

Začnimo s tem, da moramo vsako celico opredeliti na tri načine. Imeti mora možnost shranjevanja »spremenljivih opisnih« podatkov, kakor so to januar, obroki itd.; možnost shranjevanja številčnih podatkov za uporabo v aritmetičnih operacijah, na primer znesek obresti za mesec januar; in možnost shranjevanja matematičnih obrazcev, na primer »letne obresti/12«, da dobimo mesečne obresti. Vsaka celica mora imeti možnost širitve ali oženja, vendar pri tem ne sme izgubiti nobenega pomembnega podatka. Zato morajo biti vse podvojene: prva, ki jo gledamo na zaslonu, in druga, ki ima ves čas shranjene vse podatke in je skrita globoko znotraj programa.

Kakor vidimo, je urejanje podatkov zelo zamotana stvar. Pri tem pa se moramo zavedati, da imajo bolj dognane preglednice do 16000 posameznih celic! Tehnika pisanja take programske opreme nas spominja na pisanje prevajalnikov za računalniške jezike, kakor sta basic ali forth.

Sedaj nam je nekoliko bolj razumljiva visoka cena takšne poslovne programske opreme. Programi, kakot sta Visicalc ali Supercalc, imajo na zahodnem tržišču izredno visoko ceno, vendar bi pri tem ne smeli pozabiti prihranka, ki ga lahko uporaba takšnega programa prinese. Že prej omenjeni primer uporabe, ko poslovnež primerja stroške, lahko omogoči prihranke, ki se gibljejo v obsegu 15 do 20 odstotkov na leto. Stroški za programsko opremo so v primerjavi s prihranki ob boljšem gospodarjenju zanemarljivi, zato se nam, tudi če ne upoštevamo prihranjenega časa in energije, takšna investicija hitro povrne.

Pri preglednici lahko pomikamo okence, ki ga gledamo na zaslonu, po celotni tabeli. Vidno področje izbiramo s posebnimi tipkami. Tako lahko dobimo večje področje, kakor pa ga lahko prikažemo na zaslonu.



Doba prenosnih računalnikov

Medtem ko računalniki postajajo bolj dognani, se dajo stisniti v manjša ohišja, tako da so zdaj mogoči resnično prenosni mikroračunalniki

Ni igrača
Prenosni računalniki, kakor na primer Epson HX-20 s pomnilnikom 32 kb in vdelanimi perifernimi enotami, so poslovnežem močno olajšali dostop do informacij, ne glede na to, kje se nahajajo. Na mikrokaseto lahko naložite 230 kb informacij (to ustreza okoli 40000 besedam), standardni urejevalnik besedila pa omogoča, da lahko delate kjerkoli, tudi v taksiju!

Razvoj prenosnega mikroračunalnika, kakršnega poznamo danes, je prišel iz dveh smeri. Ena je bila povečanje zmogljivosti žepnih računal, kakor sta na primer Sharp PC 1251 in Casio FX700P. Druga je bil proces miniaturizacije, ki ga predstavljajo računalniki, kakor sta Tandy TRS80 Model 100 Personal Computer in Epson HX-20.

Ta napredek je rezultat razvoja integriranih vezij z velikim številom elementov, ki omogočajo, da na enak prostor spravimo veliko več informacij.

Ko se je leta 1972 prvič pojavil mikroprocesor na enem integriranem vezju, je teoretsko postalo mogoče vdelati računalnik v škatlico, majhno kakor zavojček cigaret. Vendar je velikost prikazovalnika in temeljnega dostopa do računalnika — tipkovnice — določila praktične meje zmanjševanja.

Žepna računala pa so v resnici postala manjša in danes je mogoče dobiti ročno uro, ki je hkrati tudi računalo; za pritiskanje njegovih »tipk« je treba uporabljati vžigalice. Toda z njimi je že dovolj težko izvajati preproste aritmetične operacije in celo najbolj vnet navdušenec bi se ustrašil vpisovanja 50 vrstic programa v basicu v tak računalnik. Zato je malo verjetno, da bodo osebni računalniki postali majhni kakor zapestne ure. Pač pa so že tako majhni, kakor so bila svoj čas žepna računala.

Ko se je v žepna računala dalo vpisovati programe, pa čeprav le v njihovih lastnih programskih jezikih, je bil le še majhen korak do tega, da pričnejo uporabljati tudi višje programske jezike, in za to se je kar ponujal basic. Približno isti čas so proizvajalci računalnikov pričeli uporabljati »trajne« pomnilnike, ki pri izklopu napajanja naprave zadržijo majhen električni naboj in zato pri tem ne izgubijo vsebine. Boljši generatorji znakov pa so omogočili prikazovati tudi črke.

Ne predraga žepna računala, kakor na primer Sharp in Casio, ki so po številu ukazov v basicu in po velikosti pomnilnika podobna hišnim računalnikom, hitro prevzemajo tržišče programljivih računal. Dovolj so majhna, da jih je mogoče spraviti v žep — celo pri srajci — in nekatere je mogoče priključiti direktno na tiskalnik in kasetofon. Brez dvoma bo kmalu mogoče priključiti računalo na telekomunikacijsko mrežo, če bo le zanimanje za to dovolj veliko. Tak priključek bo omogočil prenos informacij iz računala po telefonskih linijah. Modeli Casia so še posebno privlačni za znanstvenike in inženirje, ker ohranjajo paleto matematičnih in znanstvenih funkcij, zaradi katerih so postali priljubljeni.

Mogoče je kupiti druge modele, ki imajo že vdelan tiskalnik in skupaj s kasetofonskim vmesnikom stanejo okoli 125 funtov. Sharpov PC 1251 ima približno podobne lastnosti in ceno.

Naslednji velik korak predstavljajo modeli »za v naročje«, kakor na primer Epson HX-20, Tandy 100 in NEC 8220. Ti mikroračunalniki imajo celotno opremo za basic (v vseh omenjenih primerih je to Microsoft), 16 do 24 kb delovnega pomnilnika in vdelan kazalnik na tekoče kristale (LCD) sprejemljive velikosti (20 krat 4 pri Epsonu, 40 krat 8 pri drugih dveh) ter možnost priključka na velik izbor standardnih periferih enot.

Zares, če jih priključite na monitor (morda morate za to kupiti poseben vmesnik), dobite mikroračunalnik, ki je po velikosti, zmožnostih in ceni primerljiv z navadnim hišnim računalnikom. Bistvena razlika med njima je v tem, da opisane naprave uporabljajo svoj lastni, notranji vir energije, ki omogoča popolno prenosnost.

Epson, na primer, ima vdelane nikelj-kadmijeve akumulatorje, ki z enim polnjenjem napojijo računalnik tudi za 50 ur. Sistem je narejen tako, da se sam izklopi, ko so akumulatorji že skoraj prazni, in tako ohrani bistvene podatke, shranjene v pomnilniku.

Nasprotno pa NEC in Tandy uporabljata integrirana vezja, izdelana v tehnologiji komplementarnih metal-oksid polprevodnikov (CMOS), ki v primeri z drugimi porabijo veliko manj energije in jih zato lahko napajamo z običajnimi baterijami.

Epson je bil prvi mikroračunalnik, ki so zanj delali množično reklamo in je bil sprva mišljen kot »pomočnik poslovnežem«, bolj verjetno pa ga boste srečali zunaj pisarne kakor v njej. Odlično je prilagojen za zbiranje podatkov v tovarniških dvoranah, na odročnih mestih ali celo med hojo po ulici. Te podatke nato lahko obdelamo v pisarni.

Epson ponuja različne načine prenašanja podatkov. Poseben dodatek omogoča, da ga uporabljamo kot brzojavni terminal; prek modema ga lahko priključimo po telefonski liniji na drug računalnik; najbolj preprosto in najcenejše pa je, če podatke spravimo na kasetni trak in nato trak pošljemo po pošti.

Zaradi take vsestranosti in prilagodljivosti taki računalniki privabljajo ljudi, ki poprej niso niti pomislili, da bi računalnike uporabljali pri vsakodnevnem delu. Trgovski potnik, na primer, ki opravi 20 ali več obiskov na dan zato, da pri svojih kupcih potrdi običajna naročila, bi lahko uporabil, namesto da izpolnjuje naročilnice, prenosen računalnik take velikosti in zmožnosti. Program za obdelavo naročil bi na ukaz prikazal seznam izdelkov, ki jih potnik prodaja, in naročilo bi kar vtipkal v računalnik. Ob koncu vsakega obiska bi podatke o naročilih spravil na kasetni trak, zvečer pa bi potnik lahko poslal kaseto z vsemi dnevnimi naročili v svojo centralo ali pa prenesel podatke naravnost v glavni računalnik svojega podjetja.

Preko tiskalnika v mikroračunalniku pa bi trgovski potnik dobil kopijo svojih dnevnih naročil ter takojšnje potrdilo za naročnika. Le malo večji računalnik bi lahko zasledoval popis zalog v skladišču in opozarjal, ko bi se te nevarno zmanjšale. Seveda, če bi več potnikov prodajalo te zaloge istočasno, bi se morali pri vsakem naročilu s svojimi mikroračunalniki priključiti na glavni računalnik. S cenenim in lahkim akustičnim sklopnikom to ne bi bilo težko. S tem bi naš ceneni prenosni mikroračunalnik postal že pravi interaktivni terminal, ki bi omogočal trgovskemu potniku na terenu takojšen dostop do celotne podatkovne baze svoje firme.

Poleg očitnih prednosti takojšnjega dostopa do najnovejših informacij lahko prihranki časa in stroškov, ki jih omogoča takšna izmenjava podatkov, v nekaj mesecih poplačajo cel prenosni sistem. Kakor mali žepni računalniki izpodrivajo programljiva računala, tako računalniki »za v naročje« izpodrivajo naprave za vpisovanje in zbiranje podatkov. Ti »topoumni terminali« na baterije — ne dajo se namreč programirati — so na tržišču že nekaj časa, vendar zaradi visoke cene in težavne uporabe nikoli niso bili posebno priljubljeni ali uspešni.

S primerno programsko opremo, možnostjo, da se priključijo na dobre monitorje in tiskalnike, ter morda s hitrejšimi kasetnimi in diskovnimi enotami se bo uporaba prenosnih računalnikov »za v naročje« verjetno zelo razširila. Prenosne naprave te vrste so prvo resno tržišče za magnetno-mehurčne pomnilnike. Ti slonijo na povsem drugačni vrsti integriranih vezij in pri izklopu naprave ne izgubijo vsebine, omogočajo pa orjaško velikost pomnilnika v majhnem ohišju. Tipični primeri, kakor Sharp PC 5000, omogočajo dodajanje 128 kb pomnilnikov v vtični enoti, s časi dostopa, ki so nekoliko krajši kakor pri disku. Zelo vabljiva je misel na majhne računalnike z milijonom zlogov spomina.

Taka velikost pomnilnika je za zdaj dostopna na diskih pri tretji vrsti prenosnih računalnikov, kakor so na primer Osborne Executive, Ajile Hyperion in Portico Miracle. Ti stanejo pet- do šestkrat toliko kakor HX-20 ali TRS 100 in bi jih bolj pravilno opisali kot prenosljive, saj potrebujejo električno energijo iz vtičnice. Nekateri od njih imajo tudi baterijsko napa-

Računalnik za vsako rabo
(V smeri urnega kazalca od desne zgoraj.) Prvi med novimi računalniki »za v naročje« je bil Epson HX-20. Ima 32 kb pomnilnika in velik izbor periferih enot.
Najcenejša in najšibkejša izmed novih prenosnih računalnikov sta Sharp PC 1251, na sliki predstavljen s tiskalnikom in mikrokaseto, ter Casio FX700P, ki ga lahko uporabljamo skupaj z navadnim kasetofonom.
Potem ko je podjetje Osborne Computers osvojilo velik del trga, je naredilo nov računalnik z večjim zaslonom in drugimi izboljšavami. Ime mu je Executive.

Stenopis
Z izumom Microwriterja smo se že za korak približali elektronski beležnici. Microwriter je urejevalnik besedila, ki je prirejen za delo z eno roko. Šest tipk pritiskamo v različnih kombinacijah, ki pomenijo različne črke abecede. Zapisane podatke lahko izpišemo na tiskalniku ali pa jih spravimo v večji računalnik.

Poklicno orodje

Cenene prenosne računalnike uporabljamo tudi tam, kamor informacijska revolucija doslej še ni segla.

Lekarnarji lahko preprosto in čitljivo označujejo izdana zdravila. Podatki v računalniku nato pomagajo obnavljati zaloge v skladišču.

Kontrola zalog v skladišču je pomembna celo za majhna podjetja, za velika pa je sploh nepogrešljiva. Prenosni računalniki z 32 ali 64 kb pomnilnika lahko pri takem delu zelo pomagajo.

Novinar, ki se pripravlja na intervju, lahko v svoj prenosni računalnik vtipka svoja vprašanja in nato zraven njih še odgovore svoje »žrtve«. Na poti nazaj v uredništvo lahko svoj prispevek uredi in pripravi za objavo.

Trgovski potnik lahko na mestu da predračun tudi za obsežna dela, kakor je napeljava centralnega ogrevanja.

janje, vendar to redko omogoča več ko dve uri uporabe.

Običajno imajo ti računalniki vdelano dvojno enoto za diskete, televizijski monitor, ločeno tipkovnico in operacijski sistem, kakor je na primer CP/M.

Od računalnikov te vrste je bil prvi na tržišču Adam Osborne model 1. Ponujal je popoln spekter programske opreme, vštevši basic, Wordstar, Supercalc in CP/M, vse to za manj ko 2000 funtov.

Šibka točka tega računalnika je bil majhen zaslon — 100 mm krat 85 mm — tako da so bili prikazani znaki precej majhni in je bilo zato potrebno pri programih, ki so zahtevali večje znake (na primer tekstovni procesor Wordstar), premikati vrstice sem ter tja po zaslonu, da bi se videle cele. Uporabniki pa so se na to hitro navadili in Osborne 1 je postal najbolje prodajani računalnik svojega razreda.

Tekmeci so potem ponudili več zmožnosti, posebno glede mikroprocesorja in hitrosti, velikosti diskovne enote, dokler ni ta razred računalnikov postal v vsem razen v zunanjosti povsem enak večjim pisarniško-poslovnim računalniškim sistemom. V mnogih primerih so ti računalniki združljivi z drugimi, na primer IBM Personal Computer, in to mogoče pospešuje njihovo uporabo kot »drugi računalnik«.

Prenosni računalniki in pisarniški računalniški sistemi se resnično ločijo le po ohišjih. Program, ki je namenjen uporabi na običajnih pisarniških sistemih, prav tako lahko uporabljamo na prenosnem računalniku. Verjetno bi bila edina omejitev velikost zaslona in s tem velikost posameznega znaka.

Vse to je omogočil razvoj mikroprocesorjev na enem integriranem vezju. Omejitve pri dostopnosti vhodov in izhodov računalnika so posledica uporabe tipkovnic in televizijskih zaslonov za vhodne in izhodne enote. Toda zamislimo si, da bi ju zamenjali z napravami za spoznavanje glasu in umetno govorico! Ni razloga, da mikroračunalnik z enakimi zmožnostmi, kakor jih ima danes Ajile Hyperion, potem ne bi bil enako majhen kakor Sony Walkman. Verjetno bi mu bil enak tudi na pogled, z dodanim majhnim mikrofonom, če bi le imeli tehnologijo, s katero bi ga naredili.

V dobi bliskovitega napredovanja tehnologije pa bo razvoj mikroračunalnikov velik, tudi če ne zaidemo v domišljijski svet upravljanja računalnikov z glasom. Predstavljajte si, na primer, da združite Sinclair ZX Spectrum, Sinclairovo televizijo s ploščatim zaslonom in Microdrive, ter tak sistem napajate z nikelj-kadmijevimi akumulatorji. Ni razloga, da si takega sistema ne bi mogli sestaviti sami, čeprav bo morda to storil za nas Sinclair.

Ker je ZX Spectrum majhen, je kakor nalašč za tako uporabo, toda večina računalnikov potrebuje za delovanje le napetosti in tokove, ki jih zlahka dobimo z združevanjem običajnih baterij. Skoraj ni razloga, da ne bi po tej poti mogli narediti prenosnega praktično kateregakoli mikroračunalnika.

Zanimiv razvoj kaže računalnik ACT Apricot, ki je sicer običajen pisarniški sistem, čeprav manjše velikosti. Odložite njegov monitor in Apricot postane prenosen. Prenosnost pa je seveda slabo definiran pojem. Mnogi računalniki, ki jih prodajajo kot prenosne, imajo vdelan le ročaj za nošenje, nikoli pa niso bili mišljeni kot samostojni in sklenjeni sistemi.

S širokim spektrom prenosnih mikroračunalnikov so proizvajalci naredili velikanski korak proti svojim resničnim zmožnostim — proizvodnji cenenih računalnikov za vsakogar, takšnih, ki ne bodo za uporabo zahtevali nobenega znanja ali učenja. Res bo minilo še nekaj časa, da se bo to zgodilo, vendar se cilj že vidi.

Komunikacija človek–računalnik

Razvoj gre v smeri, ki bo človeku omogočila enakopravno uporabljati različne načine komunikacije z računalnikom

V prvih desetletjih razvoja industrije obdelave podatkov v splošnem ni skrbel učinkovit dialog človek – računalnik, saj je bila prvenstvena skrb posvečena učinkoviti uporabi strojne opreme: sistemi so bili snovani od znotraj navzven (začenši z notranjostjo stroja) in pri tem zaporedju je človek prišel zadnji na vrsto.

Danes morajo biti računalniški sistemi snovani, če naj bodo učinkoviti, od zunaj navznoter; operater na terminalu ali konzoli postaja osnovno žarišče snovanja sistemov. K takemu stanju je pripomoglo več dejavnikov: cena računalniške moči stalno pada; računalniki postajajo shrambe človekovega znanja, saj smo priče pravi eksploziji spoznanj in informacij, ki jih ni mogoče več obvladovati s tradicionalnimi metodami; vrste človekovih dejavnosti danes ni več mogoče opravljati brez pomoči računalnika. Skratka, računalnik čedalje bolj postaja naše (edino) *informacijsko okno* v svet in za »gledanje« skozi to okno moramo razviti ustrezne metode komunikacije človeka z računalnikom.

Pri tehnični realizaciji te t.i. »interaktivne« komunikacije sta dolgo časa prevladovala *tipkovnica*, s katero je človek dajal navodila računalniku, in *zaslon*, ki je kazal računalnikove reakcije. Ta dialog je potekal le s pomočjo simbolov (črk in številk), ti simboli pa človeku predstavljajo samo enega od več komunikacijskih načinov. Izboljšanje oziroma olajšave v izmenjavi informacij med človekom in računalnikom sta prinesla svetlobno pero in grafika na računalniku.

Pri omenjenih načinih komuniciranja je človek omejen na svoj sicer najmočnejši senzorni kanal, na vid. Na poti k bolj naravni komunikaciji med človekom in računalnikom so nato vpeljali *zvočnik*, ki posreduje računalnikova sporočila, in *mikrofon*, ki je odprl pot za človeku najbolj naravno izrazno sredstvo — govorni jezik. To sta dve preprosti napravi, ampak njuna uporaba v povezavi z računalniki je daleč od enostavne, saj moramo le-tega pripraviti do tega, da bo *razumel* človekova sporočila na eni strani, na drugi pa dosegel, da ga bo človek (prek zvočnika) razumel.

Tok glasu

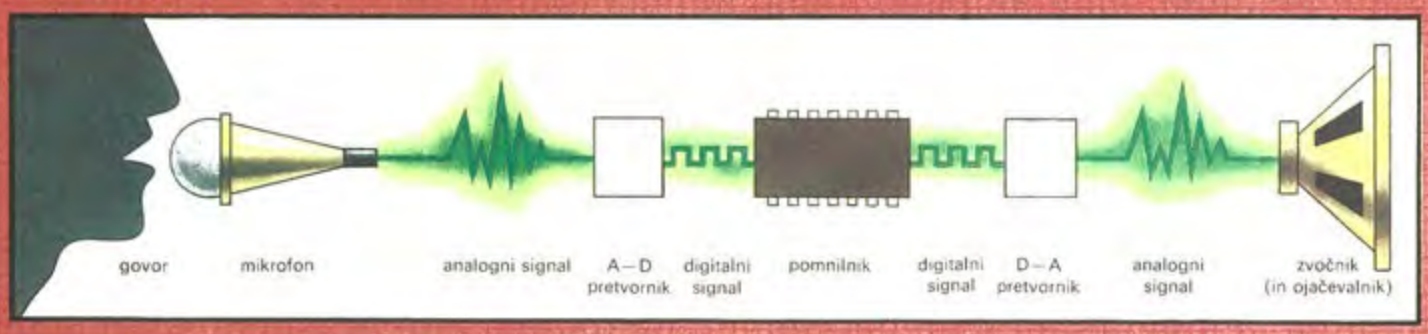
Govor je mogoče digitalizirati in shraniti v pomnilniku bodisi RAM ali ROM. Električni izhod iz mikrofona gre skozi analogno-digitalni pretvornik. Izhod iz tega integriranega vezja je digitalni vzorec 0 in 1. Tako shranjeni govor lahko reproduciramo z uporabo digitalno-analognega pretvornika, ojačevalnika in zvočnika.

Generiranje govora

Pa se prav posvetimo govorni komunikaciji v smeri računalnik – človek. Danes namreč že obvladamo generiranje elektronskega govora — vsaj splošne matematične in tehnične osnove ter uporabne oziroma praktične podrobnosti za nekatere bolj razširjene jezike. Do pred kratkim je bilo za generiranje človeku podobnega govora potrebno precej računalniške moči in pomnilniških kapacitet. Danes pa je s pomočjo primernih dodatkov skoraj vsak hišni računalnik (in tudi elektronska igrača) sposoben odgovarjati v naravnem govorjenem jeziku. Hiter napredek v tehnologiji in padajoča cena računalniških komponent sta omogočila, da je govoreči računalnik nekaj vsakdanjega.

Ljudje pri govorjenju proizvajamo tri bistveno različne vrste glasov*. Glasovi prve vrste so zveneči (zapornik) ali glasovi samoglasniške vrste

* Opisana razdelitev načinov generiranja različnih vrst glasov je poenostavljena, kar hitro dokažemo s tremi različnimi vrstami (istega) samoglasniškega glasu »e«: če smo pozorni, ugotovimo, da z glasilkami resda dosežemo ustrezno višino tona oziroma vrsto samoglasniškega glasu (»e«), z ustnicami in jezikom pa njegovo kvaliteto (tri vrste »e«). Pri bolj natančnem opisu človeškega generiranja glasov tako govorimo o dveh hkratnih dimenzijah (mestih) nastajanja glasov — zveneči in frikativni, kar da 4 vrste glasov: 1. zveneči-frikativni — »z«, »ž«; 2. nezveneči-frikativni — »f«, »h«, »s«, »š«; 3. zveneči-nefrikativni — samoglasniki, »l«, »n«; 4. nezveneči-nefrikativni — zev.



— »o«, »a«, »i« itd. Te glasove proizvaja nihanje glasilk v grlu, pri čemer frekvenca tega nihanja določa samoglasniški glas. Glasovi druge vrste so nezveneči (priporniki, frikativi), kot so »s«, »š«, »h« in »f«. Pri teh glasovih gre zrak iz pljuč nemoteno mimo glasilk in frekvenco glasu določa položaj ustnic in jezika. Tretji »glas« je zev (tišina), če hočemo biti bolj natančni, presledki, ki se pojavljajo znotraj besed, kot so »ata«, »ako«, »Koper« (nasploh pred glasovi »p«, »t« in »k«). Ni nujno, da se zavedate, da pri izgovoru takih besed obstajajo presledki, toda če jih poskušate počasi izgovoriti, boste ugotovili, da je gladek prehod od glasu »a« v »t« ali »k« nemogoč.

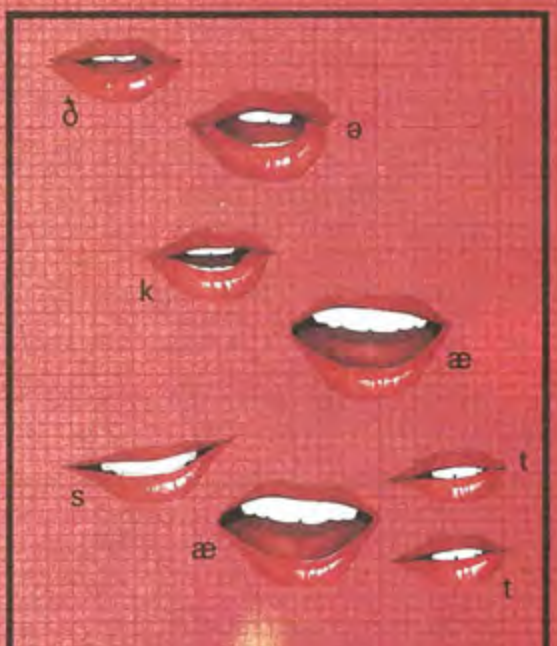
Sestavni deli glasov

Za elektronsko generiranje govoru podobnih glasov obstajata dva načina. Prvi način, ki je bil do nedavnega najbolj pogost, je s t.i. »sintezo delov besed«. Z analizo frekvenc, ki jih vsebuje govor, je mogoče zasnovati sistem pravil, ki nam omogočajo kreiranje kateregakoli danega glasu iz njegovih komponent. Na primer besedo »so« lahko definiramo kot določeno število milisekund mešanice frekvenc, ki sestavljajo glas »s«, ki ji takoj sledijo frekvence glasu »o«.

Posamezne besede lahko razdelimo na sestavne dele glasov različnih velikosti. Najmanjše imenujemo »fonemi«, večje »(fonetični) polzlogi« in še večje »(fonetični) zlogi«. Z njihovo uporabo v različnih kombinacijah lahko skonstruiramo katerokoli besedo. Kadar generiramo govor na tak način, se individualne značilnosti posameznega človeškega govorca izgubijo, toda besede je mogoče prepoznati in razumeti. Kjer so pravila za generiranje npr. fonemov določenega jezika vgrajena v samo strojno opremo, lahko uporabnik vnese zaporedje fonemov v sistem. Fonemi se reproducirajo skozi majhen zvočnik. Z malo prakse je mogoče hipoma generirati celotne stavke s priklicem zaporedja fonemov, ki jih običajno lahko shranimo v basicove nize znakov. Pri presojanju naravnosti (kvalitete) tako zloženih besed oziroma stavkov se moramo zavedati, da je tukaj opisano enostavno »zlaganje« govornih LEGO kock zdaleč nezadostno, ker je za ustrezno povezavo posameznih zidakov potrebno storiti še marsikaj; recimo temu malta, ki gladko veže zidake med seboj.

Pri takem pristopu je potrebno opozorilo, da si bo vsak narod svoj računalniški govor generiral sam. To velja tako za vgraditev za določeni jezik ustreznih sestavnih delov besed v strojno opremo (sicer lahko pri nas pričakujemo npr. ameriško zveneče slovenske besede — pomislimo samo na ameriško izgovarjavo glasov »o« in »r«) kot za poglobljene raziskave povezav med takimi sestavnimi deli (če nočemo, da bi nam računalnik »go-vo-ril kot da bi dr-va se-kal«). Končno, čeprav ne najmanj pomembno, je pa treba upoštevati tudi višje nivoje jezika od posameznih besed (npr. različna izgovarjava besede »je« v pomenih »biti« in »jesti«).

Druga metoda za sintezo govora se zanaša na sposobnost človekovih ušes in možganov, da zapolnijo praznine (v širšem smislu besede). Na primer: frekvenčni obseg (širina), ki ga lahko prenesemo prek telefonske linije, daje samo petino kvalitete, ki jo pričakujemo od hi-fi sistema srednje kvalitete, toda



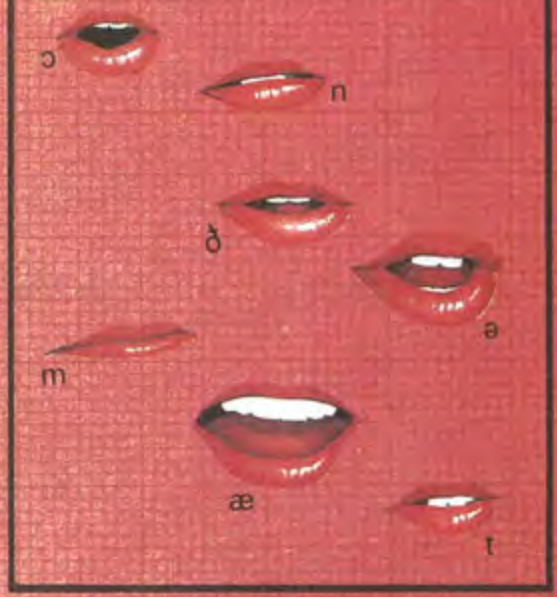
Elementi govora

Posnetki v tem stolpcu kažejo izgovarjavo fonemov, iz katerih je sestavljen preprost angleški stavek »The cat sat on the mat« (Mačka je sedela na preprogi). Ti fonemi so — v uveljavljeni slovenski fonetski transkripciji — naslednji:

ðə kæt sæt ɔn ðə mæt.

Če te foneme sintetiziramo v pravem zaporedju, dobimo približek človeškega govora. Integrirano vezje Votrax na primer lahko generira 80 fonemov in hrani tudi pravila za njihovo povezavo. Vsak fonem je dosegljiv z ustreznim številom. Če hočemo torej slišati neki stavek, odtipkamo zaporedje fonemov oziroma ustreznih številk. Sintetizatorja Braid Speech Synthesiser in Votraxov Personal Speech System pa sta opremljena še s posebnim mikroprocesorjem in programi, tako da lahko odtipkamo angleško besedilo (v običajnem zapisu) in slišimo izgovorjeni tekst.
Med posnetki izgovarjave angleških fonemov lahko poiščete tiste, ki bodo ustrezali slovenskim fonemom za stavek. Mi smo doma. Primerjava je naslednja (slovenski fonem — angleški fonem):

m = m, i = s, s = n, m = m, o = o, d = t, o = o, m = m, a = æ.



kljub temu je govor, ki ga slišimo skozi slušalko, popolnoma razumljiv in ne samo to, zaznavamo tudi značilnosti govorca — in to zaradi tega, ker naši možgani zapolnijo praznine (v tem primeru praznine v prenašanem frekvenčnem obsegu).

Druga metoda sinteze, imenovana »digitalizirani govor« — bolje bi bilo reči »konzervirani ali kodirani govor« — izkorišča opisani pojav. Z nižanjem cene računalniškega spomina je danes mogoče pretvoriti govor v digitalno informacijo s pomočjo analogno-digitalnega pretvornika. Rezultirajoče podatke nato večstokrat komprimiramo (kar ima za posledico zmanjšanje tako zajete informacije) in shranimo v ROM.

Če želimo povzročiti izgovor katerekoli od tako shranjenih besed, preprosto damo računalniku naslov te besede v ROM, odkoder sistem vzame digitalno informacijo in jo pretvori nazaj v glas. Osebne značilnosti takih besed ostanejo, ker so shranjene besede originalnega govorca. Na primer v Acornovih govornih integriranih vezjih za BBC Micro lahko jasno prepoznamo glas napovedovalca Kennetha Kendalla.

Zopet je potrebno opozorilo, da pri različnih jezikih ne moremo mehanično presajati izkušenj drugih. Če samo pomislimo na bogastvo pregibnih oblik slovenščine, kar z drugimi besedami pomeni veliko število različnih besednih oblik iste izhodiščne besede (»miza«, »mize«, »mizi«, ...), nam postane jasno, da tega pristopa ne bomo mogli uporabiti pri bolj zahtevnih projektih računalniškega generiranja slovenskega glasu.

Nekateri računalniki, zlasti npr. Sirius 1, imajo vgrajeno strojno in programsko opremo na disku, in omogočajo uporabniku digitalizacijo lastnega glasu z uporabo mikrofona. Rezultirajoči podatki so shranjeni na disku — ena sekunda govora zavzame približno 1 kbyte pomnilnika — od koder jih uporabniški program lahko prikliče kot osnovo za besedna sporočila in opozorila.

Uporabe sintetizatorjev govora so številne in raznolike, tako da je skoraj nemogoče navesti vse. Sinteza govora lahko npr. nadomesti na trak posneta obvestila na železniških in avtobusnih postajah, letališčih in drugih terminalih; v ZDA jo na široko uporabljajo v letalskih sistemih za informiranje klicajočih o napačno izbranih in zasedenih številkah ali ukinjenih uslugah. Številni avtomatizirani sistemi za naročanje že vsebujejo »sintetizirane« odgovore; ko vtipkamo številko naročila v računalnik, ta izgovori opis, ki rabi za dvojno preverjanje. Računalnik lahko tudi obvesti stranko o trenutnem stanju zalog ali pa o verjetni dobi čakanja, tako da lahko stranka spremeni naročilo ob samem času naročanja.

Enote za sintezo govora so danes vgrajene v avtomobile — npr. BL Maestro — kot del standardne instrumentacije. Taki sintetizatorji so več kot prodajalski trik, saj dajejo opozorila, ki jih voznik lahko sliši in reagira nanje, ne da bi umaknil oči s ceste.

V hišnih računalnikih in elektronskih igrah sintezo govora uporabljajo za izboljšanje kvalitete iger: opozorila o sovražnikovih napadih so npr. podana govorno, kar omogoča igralcu, da se bolj koncentrira na taktiko igre, saj mu ni treba pregledovati sporočil, izpisanih na dnu zaslona.

Končno, obstajajo tudi učni pripomočki, kot je Speak'n' Spell firme Texas Instruments, ki izrecitira besedo, ki jo je nato treba pravilno črkovati, in slovarji tujih jezikov, ki izgovorijo besede istočasno, ko jih prikažejo.

Posebno področje generiranje govora je »sinteza govora iz teksta«, pri kateri računalnik pretvori poljubni tekst v govor; beseda »poljuben« pomeni, da lahko vse odgovore računalnika, ki so bili podani z običajnimi simboli (črkami, številkami, besedami), pretvorimo v ustrezno govorjeno obliko. Poseben primer tega pristopa je pretvorba različnih merilnih podatkov (npr. iz vodovodnega omrežja) ali sporočil (npr. kode napak iz avtomatskih telefonskih central) v tako smiselno obliko sporočila, ki omogoča sintezo govora.

Prepoznavanje govora

Če naj bo računalnik resnično uporaben, nam mora omogočiti, da ga »hranimo« z ukazi in informacijami. »Vmesnik«, ki ga običajno uporabljamo za komuniciranje s hišnim računalnikom, je tipkovnica (čeprav sta miška in krmilna ročica — v povezavi z menuji izbir možni alternativi). Pri uporabi tipkovnice pa odkrijemo, da smo z računalnikom prisiljeni komunicirati s

Deli govora

Pri prvi tehniki prepoznavanja govora preprosto digitaliziramo govor in izvajamo izčrpno analizo z metodo »prepoznavanja vzorcev«. Po bolj učinkoviti metodi uporabljamo predhodno obdelavo s strojno opremo (elektronika), pri kateri neodvisna vezja merijo signal glede na zveneče glasove (npr. samoglasniki), nezveneče glasove (s, š, h itd.) in kratka obdobja tišine (npr. med zlogi). Izhod iz vsake od teh filtrirnih naprav je zaporedje ničel in enic, ki ga računalnik primerja s knjižnico shranjenih primerov in izbere najbližje ujemanje kot besedo, ki jo je prepoznal.

pomočjo umetnega jezika. Ukazi, kot so CLS, DIRECTORY, RUN, LOAD in SAVE, imajo morda smisel za operacijski sistem, toda niso »naravni«.

Naravna komunikacija za ljudi je govor, ne pa vtipkavanje sporočil na tipkovnicah in opazovanje odgovorov na televizijskih zaslonih. Če bi lahko pripravili računalnik do tega, da bi razumel izgovorjene ukaze — pa čeprav bi bili oblikovani na enak način kot tisti, ki jih vnesemo prek tipkovnice — bi ga bilo dosti lažje uporabljati; še zlasti pa fizično prizadetim ljudem. Vendar preden lahko katerikoli računalniški sistem »razume« izgovorjene besede, mora najprej obdelati glasovni vhod; analogne signale je treba analizirati in pretvoriti v digitalno obliko, šele nato se jih lahko računalnik zares loti. Čeprav je videti, da je govor mogoče preprosto generirati na elektronski način, je govor zelo zapletena kombinacija glasov.

Sanje o takojšnji in popolni prepoznavi govora (kar je npr. zmogel računalnik HAL v filmu »2001 Vesoljska odiseja«) verjetno ne bodo izpolnjene še mnogo let, če sploh bodo kdaj. Pisalni stroj z glasovnim vhodom je enako oddaljen; vendar pa tehnologija za to oboje in tudi za »razumevajoči« računalnik že obstaja. Razen visoke cene tehnologije pa obstaja še ena velika ovira na poti k sistemom za prepoznavanje govora: besede lahko zvenijo enako, vendar imajo različen pomen, ki je odvisen od konteksta (sobesedila), v katerem se nahajajo. Računalniške moči, ki je potrebna za rešitev tega problema, ni na voljo po razumni ceni.

Čeprav so raziskovalci ustvarili sisteme, ki se približujejo temu cilju, so odkrili, da z večanjem števila ljudi, ki jih računalnik lahko prepozna, pada število besed, ki jih sistem lahko prepozna. 20 do 30 je število besed, ki jih zmore naenkrat prepoznati sistem za prepoznavanje več govorcev, in to z mero uspešnosti okoli 85 do 90 odstotkov.

Možnosti uporabe sistemov za prepoznavanje govora so upoštevanja vredne. Tako nemška poštna služba že uporablja sistem za pomoč pri sortiranju pošte; v letalstvu — tako civilnem kot vojaškem — je široko področje uporabe takih sistemov, saj piloti dobesedno nimajo dovolj rok in nog za vodenje letal. Pri vseh teh uporabah je število besed, ki jih je mogoče prepoznati naenkrat, omejeno na okoli 20. To pa seveda ne pomeni, da je celoten sistem tako omejen. Uporabnik izbira eno od 20 besed iz »menuja« in vsak prepoznani ukaz prikliče nadaljnji menu besed, med katerimi je mogoče izbirati v naslednjem koraku. Računalnik se bo lotil določene akcije šele tedaj, ko bo uspešno prepoznal celotno zaporedje besed. V primeru sortirne službe bi bil lahko prvi nivo sortiranja po državi; ko je enkrat izbrana prava država, gre naslednje sortiranje po mestu, nato po občini (vasi) itd. Šele na najnižjem nivoju bo pošiljka dokončno odposlana, kar zagotavlja največjo zanesljivost takega postopka.

Analiza glasu

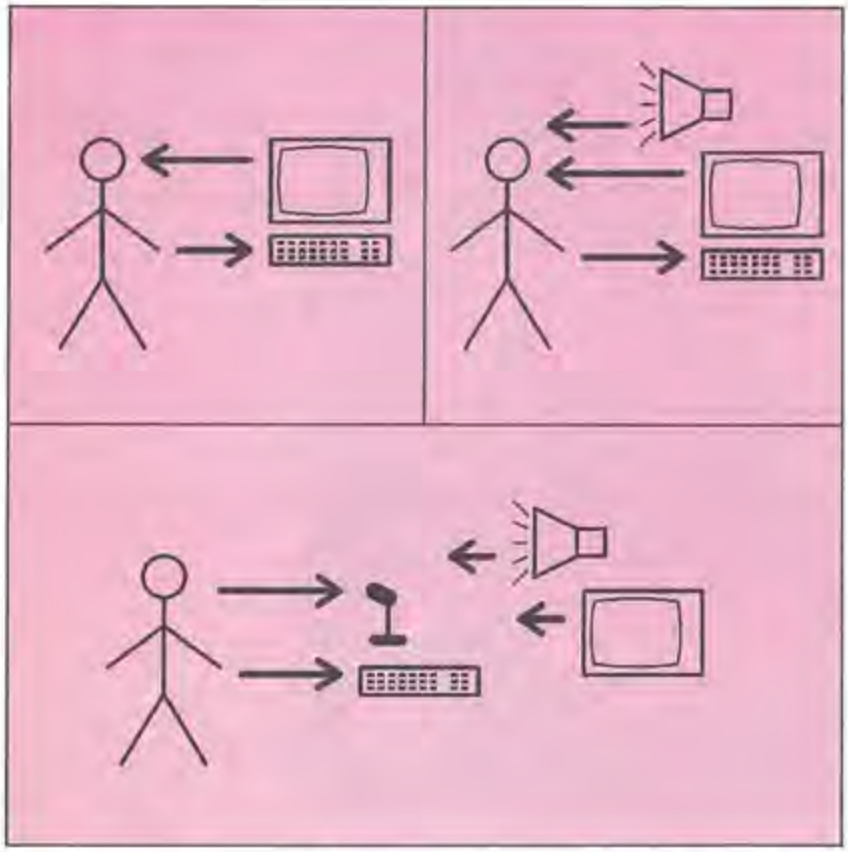
Prepoznavanja govora se navadno lotimo na enega od dveh načinov. Po »hitri« poti preprosto vnesemo prek analogno-digitalnega pretvornika ves govor v računalnik in uporabimo računalnikovo moč za izvajanje celotne analize. Ta metoda ima na žalost vrsto slabih strani, med katerimi je najbolj opazna čas, ki je potreben za dokončanje analize. Sistemi, ki uporabljajo to metodo, porabijo dve do tri sekunde za prepoznavo ene vhodne besede. Če naj prepoznavanje govora postane kakorkoli zares uporabno, mora računalnik »razumeti« govor tako hitro kot običajen človek.

Pri drugi metodi uporabljamo predprocesiranje; namesto da bi govorni signal analizirali matematično, je mogoče opraviti dosti dela s standardno elektroniko. Računalniku nato predamo informacije o govornem vhodu: frekvenčno vsebino, intonacijo, energijo itd. Frekvence lahko merimo s filtriranjem signala in zaznavanjem nivoja v vsakem frekvenčnem območju, kar je precej podobno uporabi tonskih kontrol na hi-fi aparaturi, kjer želimo npr. poudariti basovski boben. Na ta način opravljena analiza je skoraj trenutna, ker se elektronsko procesiranje opravi v hipu, ko pripeljemo originalni govorni signal na vezja. Če bi želeli podobno operacijo opraviti na digitalnih podatkih iz A/D pretvornika, bi to zahtevalo več računalnikov, ki bi obdelovali številke hkrati. Ta metoda predprocesiranja je še vedno v raziskovalni fazi — za zdaj takšnih sistemov še ni v prodaji — vendar lahko z gotovostjo rečemo, da ima večje možnosti kot prva.

Ko enkrat iz originalnega signala izločimo (ne glede na uporabljeno metodo) informacije o frekvenčni vsebini, intonaciji, energiji itd., se dejansko prepoznavanje opravi s primerjanjem trenutne množice števil z modeli, ki so shranjeni v računalnikovem pomnilniku. Te modele smo ustvarili s »treniranjem« sistema za prepoznavanje. Besede, ki naj bi jih sistem prepoznaval, govorimo sistemu eno po eno. Rezultirajoče informacije se shranijo v digitalno »knjižnico« zgledov. Celotno množico teh besed nato znova izgovarja nekdo drug, in računalnik primerja ta vhod s svojim trenutnim modelom. Če se ujemata, se druga množica informacij doda k prvi in tako oblikuje bolj popolna verzija modela. To je trajen proces, s katerim dodajamo nove informacije v knjižnico za več in več govorcev.

Nadzor nad okolico

Primer take uporabe vsebuje računalnik, robotsko roko in določeno število preprostih objektov, ki jih roka lahko manipulira. Uporabnik lahko z ukazom, izgovorjenim v mikrofon, ukaže roki, naj »POLOŽI JAJCE V SKODELICO ZA JAJCE«. Računalnik bo moral interpretirati ukaz, pregledati položaje objektov v svojem spominu, narediti plan akcije in sporočiti ustrezno zaporedje ukazov svoji periferiji, tj. roki.



Razvoj komunikacije človek – računalnik

Da bi prepoznal izgovorjeno besedo, mora računalnik primerjati vzorec vhodne informacije z enim ali več modeli, ki so shranjeni v trenutni knjižnici. V številnih primerih bo računalnik našel več možnih ujemanj, ker se bodo deli drugih besed ujemali z vhodnim vzorcem. Prva dva zloga besede »internacionalen« sta enaka kot v besedi »interpret«. Na koncu iskanja mora ena beseda izstopati z bolj popolnim ujemanjem kot vse druge možnosti, in to je tista, ki jo bo računalnik interpretiral kot vhod.

Pripomočki za prepoznavanje govora bodo v bodočnosti gotovo uporabni na številnih področjih, toda zelo verjetno jih bomo najbolj uporabljali kot »čelne vmesnike« za kompleksne programske pakete, kot so npr. baze podatkov, pri katerih bo mogoče izbirati ukaze iz menuja na zaslonu. Ta vrsta uporabe bo odstranila največjo oviro na poti uporabe računalnikov s strani nestrokovnjakov – tipkovnico. V sistemih za interaktivni dostop do podatkov, kot je npr. angleški Prestel, so skrčili vhodno napravo na enostavno številčno tipkovnico, toda to bistveno zmanjša obseg interakcije, ki ga uporabnik lahko tako doseže z računalnikom. Vmesnik z govornim krmiljenjem, ki bi bil zmožen prepoznati tako standardno množico ukazov za preiskovanje baze podatkov kot tudi številske simbole in črke abecede, bi bil pripomoček, ki bi zahteval malo ali skoraj nič običajne računalniške vadbe za uporabo.

Danes že obstajajo komercialno dosegljive enote za prepoznavanje, ki jih je mogoče priključiti na hišne računalnike, toda to so zelo preproste naprave. Sistemi, kot so »Big Ears« ali »Speech Lab« firme Heuristics Inc., uporabljajo dosti računalniške moči za prepoznavanje samo nekaj besed, ki jih izgovori ena sama oseba. Zmožnost prepoznavanja besed, ki jih izgovori *katerakoli* oseba, ne glede na narečje ali naglas, je tisto, kar potrebujemo, če naj prepoznavanje govora postane resnično uporabno. Omejitveni faktor na tej stopnji je količina pomnilnika, ki je na voljo za hranjenje modelov. Zanimiva možnost bi bila uporaba videodiska za hranjenje standardne množice modelov; tak pristop bi uporabljal zelo malo internega pomnilnika in zmanjšanje hitrosti bi bilo komaj opazno.

Računalniško razumevanje naravnega jezika

Če bi se lahko pogovarjali z računalnikom kot s kakim drugim človeškim bitjem, bi bila to najbolj idealna oblika komunikacije. Medtem ko ni težav pripraviti računalnik do *odgovorov* v naravnem jeziku, ga je izredno težko sprogramirati tako, da bi *razumel* naravni jezik. (Ne bomo se spuščali v filozofske in psihološke razprave o pomenu besede »razumeti« in še zlasti ne o konstruktu »računalnik razume«, ampak bomo shajali z intuitivno vednostjo o njunem pomenu.) Naravnega jezika po mnenju nekaterih ne bo nikoli mogoče popolnoma razvozlati s pomočjo računalnika, vendar v razvitem svetu temu problemu posvečajo veliko truda in sredstev, saj bi možnost bolj naravnega komuniciranja človek – računalnik izredno razširila uporabnost računalnikov na številnih področjih.

Jezik, ki ga govorimo, je izredno neurejen; njegove besede so dvoumne, sintaksa (skladnja) zmedena. Številni stavki so nepopolni izrazi misli, saj je jezik le deloma racionalen. Mehanična analiza stavkov je težavna, saj so številne besede lahko enkrat glagol ter drugič samostalnik in jezik je poln nepravilnosti ter izjem. Prepoznavanje pomena stavkov često zahteva pravo množico predhodnega znanja in zelo težko je vprogramirati to običajno človeško funkcijo v današnje serijske stroje.

Zaradi različnih alternativnih pomenov besed je zelo težko sprogramirati skladenjsko in pomensko (sintaktično in semantično) razčlenitev (analizo) določenih stavkov. Primer v slovenščini je stavek »Gori na gori gori«, kjer ima besedna oblika »gori« tri slovnične lasnosti – besedne vrste (prislov, samostalnik, glagol), primer v angleščini pa sloviti stavek »Time flies like an arrow«, kjer so možna tolmačenja za prve tri besede po vrsti: »time« (pridevnik, samostalnik, glagol), »flies« (samostalnik, glagol) in »like« (pridevnik, prislov, glagol). Celo številni stavki, ki sicer izgledajo nedvoumni, lahko vodijo do pretanjenih problemov, ki jih je izredno težko rešiti z računalnikom.

Še težji problem za sprogramirati so t.i. »predpostavke«, kajti mentalni proces poslušalca mora biti v razumni meri blizu mentalnega procesa govorca, če naj mu omogoča pravilen pomen tistega, kar sliši. Izjave, ki smo jih podali, in prepričanja, ki smo jih imeli pred izgovoritvijo tekočega stavka, bodo vplivali na njegovo interpretacijo. Naravni jezik, ki ga govorimo, predpostavlja izredno zapleteno mrežo povezav med besedami, ki jih uporabljamo. Človeška bitja dojemajo pomen stavkov, ker imajo notranjo enciklopedijo, s katero so povezani izgovorjeni stavki. To pa ima za posledico, da ni nujno, da so stavki slovnično pravilni ali pa celo popolni v nekaterih primerih. Oseba primerja tak delno slišan stavek s svojim pogledom na svet in tako vzpostavi njegov pomen. Izredno težko je dati računalniku tako notranjo enciklopedijo, razen v primerih, ko je vsebinsko področje zelo ozko.

Oglejmo si nekaj nazornih primerov, iz katerih je razvidna težavnost komunikacije človek—računalnik v neomejenem naravnem jeziku.

»Prispodoba« (metafora), katere dva primera srečamo v stavku »Stari lisjak je končno nasedel na povsem običajen trik«, pomeni drugačne pomene ustaljenih besed, saj v našem primeru ne gre niti za žival, niti za fizično dogajanje med dvema fizičnima telesoma, kot je to sicer primer, ko ladja nasede na čer.

Pri »nanašanju« (referenca) mora računalnik pravilno povezati (usmeriti) dela stavkov, ki sodita skupaj. V stavku »Občinski možje stavkujočim niso dovolili demonstracije, ker so

a) se bali nasilja,

b) spodbujali k revoluciji«

so v primeru a) občinski možje nosilec glagolskega dejanja, v primeru b) pa jim tega skoraj ne moremo pripisati. V stavku »Videl sem moža na hribu s teleskopom« se fraza »s teleskopom« lahko po vrsti nanaša na »mene«, »moža« in na »hrib«.

»Izpuščanje« enega ali več stavčnih členov (elipsa) omogoča hitrejšo komunikacijo, s stališča računalniške obdelave pa ni preprost problem. Od sistema za preiskovanje baze podatkov si želimo tako zmožnost, ki je povsem običajna v človeškem pogovoru:

1. Kakšna je povprečna plača tajnic v podjetju?
2. Pa snažilk?

Osnovna ugotovitev večletnih raziskav računalniškega obravnavanja naravnega jezika je, da je *predstavljanje znanja* središčni problem bolj naravnega komuniciranja človek—računalnik. Izkaže se, da je v dialog med dvema osebama vpleteno dosti več kot le dobesedni pomen besed, ki jih uporabljata v takem dialogu. Omenili smo že predpostavke, dodajmo jim še »namene« in »cilje«. Za primer vzemimo vsakdanji pogovor med možem in ženo:

— mož: »Greva danes ven?«
— žena: »Lahko.«
— mož: »Kje so ključi?«
— žena: »Na omarici.«

V tem dialogu večina prave informacije sploh ni bila izrečena:

— Ker je večer in sta oba lačna, pomeni »iti ven« v resnici večerjati v gostilni.
— Ker stanuje naš par v enem od spalnih naselij, kjer ni nobene gostilne v bližini, mož sprašuje za avtomobilske in ne za katere druge ključe.
— Ključi so na omarici v predsobi, kjer jih vedno puščata, in ne na kateri drugi omarici v stanovanju.

Zato samo na osnovi stavkov prikazanega dialoga računalnik (in niti mi sami) ne more odgovoriti na naslednja vprašanja:

— Kam zares gresta?
— Kako gresta?
— Na kateri omarici?

medtem ko so odgovori za naš par sami po sebi umevni.

V naslednji zgodbi sta za razumevanje potrebna znanje in *sklepanje*, ki ju moramo na nek način spraviti tudi v računalnik, če naj ta »razume« dogajanje:

1. Peter ima rojstni dan.
2. Suna vzame prašička in ga strese.
3. Ker nič ne zasliši, je žalostna.

Za odgovor na vprašanje »Zakaj je Suna žalostna?« moramo vedeti še naslednje:

a) Suna je Petrova hči.

b) Suna hoče kupiti Petru darilo za rojstni dan.

c) Prašiček pomeni hranilnik.

č) Prazen (tih) prašiček pomeni, da ni denarja.

d) Brez denarja ni nakupa darila.

e) Brez darila ni obdarovanja.

f) Brez obdarovanja ...

Tudi pri naslednji kratki zgodbi vidimo, da so stavki, ki jo opisujejo, premalo za razumevanje dogajanja:

1. Janez si kupi mleko in ovsene kosmiče.
2. Doma ugotovi, da je mleko skisano.

Za odgovor na vprašanje »Ali je Janez sit?« moramo vedeti še naslednje:

a) Ovsene kosmiče običajno jemo z mlekom.

b) Skisano mleko (še posebno po naše — »industrijsko« kar v embalaži) ni primerno za pripravo ovsenih kosmičev.

Tudi če se omejimo na bolj ozko področje baz podatkov, nam podobni problemi ne »uidejo«, kar si ilustrirajmo s sledečimi starimi (obstoječimi) dejstvi:

1. Marjan stanuje v Šiški.
2. Marjan ljubi Apolonijo.
3. Marjan je poročen z Apolonijo.
4. Marjan je šef.

katerim želimo dodati oziroma jih zamenjati s sledečimi novimi dejstvi:

a) Marjan stanuje v Mostah.

b) Marjan ljubi Anico.

c) Marjan je poročen z Anico.

č) Peter je šef.

Pri tem se postavljajo sledeča vprašanja:

— Katera stara dejstva so med seboj neodvisna? (Npr., ali mora človek biti poročen s tisto osebo, ki jo tudi ljubi?)
— Katera nova dejstva nadomestijo stara dejstva in katera jih le dopolnijo? Dejstvo a) zelo verjetno izpodrine dejstvo 1; pri dejstvih b) in 2 tega ne moremo tako prepričano trditi; sožitje dejstev c) in 3 je v naši družbi kaznivo dejanje, na Bližnjem vzhodu pa ne itd.

Taka in podobna vprašanja so vodila do združevanja tako različnih ved, kot so psihologija, filozofija, lingvistika, računalništvo in umetna inteligenca v t.i. *kognitivno znanost*, ki proučuje spoznavanje in komunikacijske procese človeka z uporabo formalnih ter računalniških metod in katere spoznanja so neposredno uporabna pri snovanju rešitev za komunikacijo človek—računalnik.

Težave z naravnim jezikom so skrajni primer bolj splošne dileme pri snovanju dialoga človek—računalnik. Strukturo dialoga moramo zasnovati tako, da bo čim enostavnejši za človeško uporabo. Stopnja, do katere bomo izpolnili ta pogoj, bo določala sprejemljivost sistemov, ki jih ustvarjamo. Seveda pa velja, da bo za večjo svobodo, ki jo omogočimo uporabniku pri oblikovanju njegovih sporočil sistemu, potrebna večja kompleksnost programiranja za interpretiranje tistega, kar bo rekel. V praksi zelo hitro odkrijemo, da zahteve za tak interpretativni program presežejo sprejemljive komercialne nivoje izdatkov za programsko opremo in/ali računalniški čas. Nadalje, cilj napraviti dialog čimbolj tekoč, je na splošno v nasprotju s ciljem napraviti vhod (input) čimbolj natančen. To nas ne sme začuditi, kajti če tekoč dialog pomeni obračanje na računalnik na običajen pogovorni način, to hkrati pomeni težave, saj je naš jezik tako nenatančen in dvoumen.

Miške

Načrtovalci računalnikov želijo nadomestiti tipkovnico z nečim, kar se bo laže uporabljalo. Ena izmed možnosti je miška

Ni še dolgo tega, kar je bilo treba pri računalnikih uporabljati samo velike elektromehanske pisalnike — teleprinterje. Te naprave so bile hrupne, okorne in nezanesljive in jih je medtem nadomestil hitri in tihi videoterminal s tipkovnico. Videoterminal je odpravil mnoge probleme, povezane s teleprinterji, ne nazadnje tudi velike količine papirnatega traku, ki so ga teleprinterji »proizvajali« med vtipkavanjem informacije. Seveda pa sta tako mehanski terminal kakor videoterminal omejena zaradi svoje znakovne oziroma vrstične narave. Uporabnik se ne more hitro premikati po zaslonu — da bi tu izbral ukaz za menu, tam spremenil podatek, ali da bi spreminjal datoteke ali programe — ne da bi se moral srečevati z omejitvami vrstičnega formata. S krmilno ročico ali kroglo se je sicer mogoče ogniti tipkovnici pri določanju lege na grafičnih terminalih in pri video igrah, toda kako naj to koristi resnemu uporabniku?

Večina sodobnih hišnih računalnikov je opremljena s smernimi tipkami za premikanje po programu ali besedilu na zaslonu v štirih smereh do položaja, kjer je treba kaj spremeniti. Vendar pa se da kurzor premikati samo po znakih ali vrsticah in ga uporabnik ne more premakniti neposredno na želeno mesto na zaslonu. Če bi tudi črkovni kurzor lahko premikali po zaslonu tako kakor grafični, bi lahko znatno hitreje premikali podatke po zaslonu.

Tri slepe miške
Mnogi novejši poslovni mikroračunalniki imajo miško kot standardni del opreme, nekatera podjetja pa ponujajo miške kot dodatek k obstoječim računalnikom. Večina mišk deluje na podlagi vrtenja krogle na njihovi spodnji strani, na hrbtu pa imajo po eno, dve ali tri »IZBIRNE« tipke.

Glavna krogla
Velik jeklen krogelni ležaj počiva na površini, po kateri se giblje miška. Na nekaterih miškah je krogla narejena iz trde gume, da ne bi drsela.

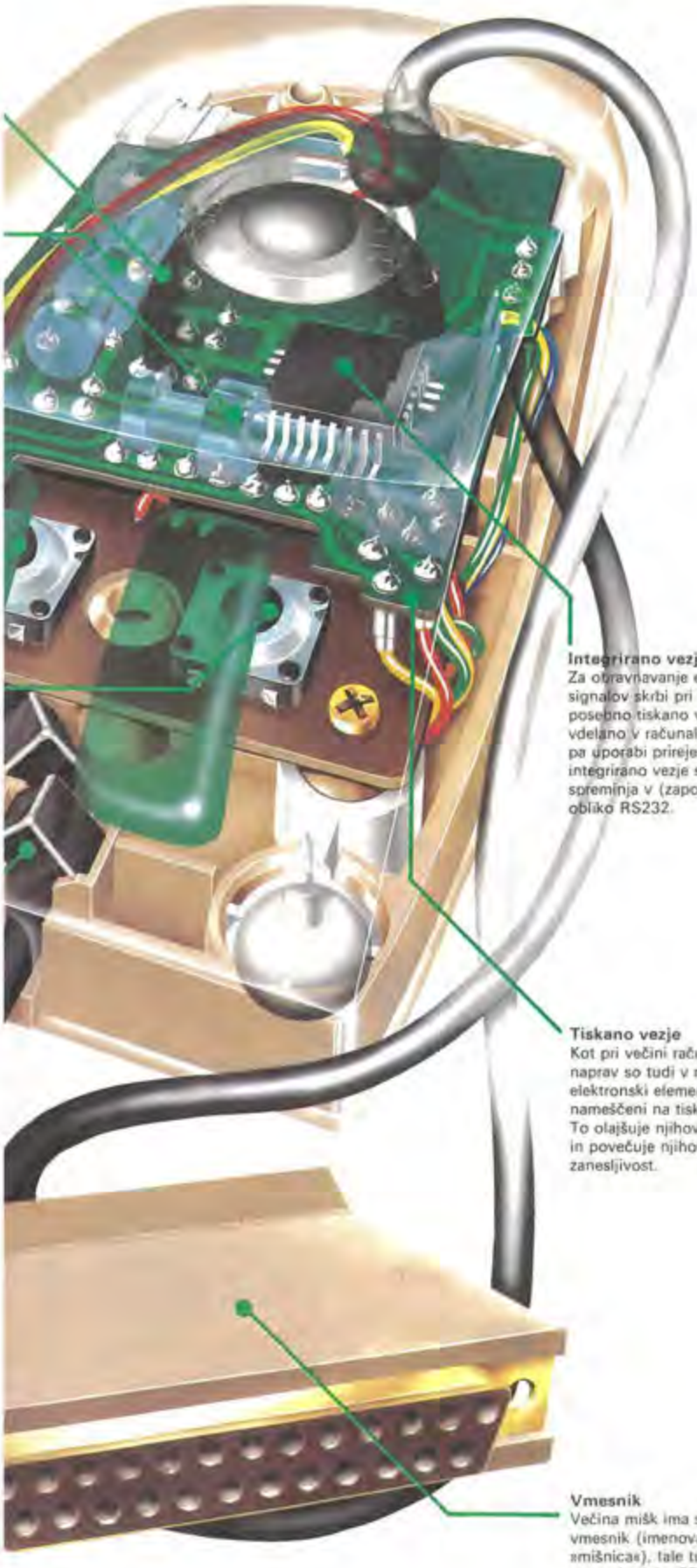
Kodirana kolesca
Ti dve kolesci se stalno dotikata krogle in spremljata njeno gibanje v dveh smereh. Kolesci sta pritrjeni na dveh oseh; kodirni napravi na konceh obeh osi spreminjata vrtenje osi v električne signale.

Tipki
Vloga teh dveh tipk je odvisna od programske opreme. Ena tipka se običajno rabi za izbiranje podatkov, druga pa za premikanje predmetov po zaslonu.

Mikrostikala
Ta so nameščena na tiskanem vezju pod tipkami in potrebujejo le majhen premik tipke za sklenitev ali prekinitev tokovnega kroga.

Gumijast moznik
Miška mora imeti možnost, da se prosto premika po mizi. Gumijast moznik je posebno pomemben za preprečevanje mehanskih obremenitev na spoju med kablom in tiskanim vezjem.

Rešitev tega problema so v šestdesetih letih najprej raziskovali na Stanfordskem raziskovalnem institutu v Kaliforniji in prvo »miško« (kakor so poimenovali novo napravo) so patentirali leta 1970. Naprava je dobila tako ime zaradi svojega videza; dovolj majhna je, da nam lepo leže v dlan, njen priključni kabel je nekak »rep«, prve naprave te vrste pa so imele tudi dve krmilni tipki kot »ušeci«. Krmilne krogle in ročice se ne uporabljajo, ker natančnost, s katero usmerjajo kurzorje po zaslonu, ni potrebno.



Integrirano vezje
Za obravnavanje elektronskih signalov skrbi pri večini mišk posebno tiskano vezje, ki je vdelano v računalnik. Tukaj pa uporabi prirejeno integrirano vezje signale spreminja v (zaporedno) obliko RS232.

Tiskano vezje
Kot pri večini računalniških naprav so tudi v miškah elektronski elementi nameščeni na tiskano vezje. To olajšuje njihovo izdelavo in povečuje njihovo zanesljivost.

Vmesnik
Večina mišk ima svoj poseben vmesnik (imenovan »mišnica«), tale tukaj pa lahko s standardnim 25-polnim povezovalnikom priključimo na vsaka vrata tipa RS232.

Delovanje miške sloni na zaznavanju njenega gibanja po ravni površini v smereh gor – dol in levo – desno. Ta gibanja se neposredno prenašajo na gibanje kurzorja po zaslonu.

Obstajata dva glavna načina zbujanja električnih signalnov ob gibanju miške. V obeh primerih je na njeni spodnji strani krogla, ki se dotika površine, po kateri miško premikamo.

Gibanje krogle pod miško se prenaša na notranje valje. Po enem sistemu so na koncih teh valjev pritrjena kodirna kolesca, na katera so izmenoma naneseni pasovi iz prevodne in neprevodne snovi. Operacijski program miške šteje dobljene električne sunke in ji omogoča določanje lege kurzorja na zaslonu. V drugem sistemu sta valja opremljena s preluknjanima ploščama. Nanju je stalno usmerjen svetlobni snop in na nasprotni strani plošč ga zaznava fotocelica. Utripe svetlobe na poti skozi zareze fotocelica spreminja v električne signale, ki jih potem obravnavamo enako kakor signale v elektromehanskih sistemih.

Obstajajo pa tudi drugačni sistemi. Eden je na primer tak, da k miški sodi tudi posebna podlaga, ki je pokrita z vzorcem pik. Lučka znotraj miške osvetljuje površino pod njo, vzorec osvetljenih pik pa razpoznava posebno optično integrirano vezje. Vsak premik miške povzroči spremembo teh vzorcev. Iz teh sprememb pa integrirano vezje sproti izračunava, za koliko in v katero smer se je miška premaknila. Prednost takih mišk je ta, da nimajo gibljivih delov, so pa dosti dražje od mehanskih.

Ko kurzor s pomočjo miške premaknemo na želeno mesto na zaslonu, lahko s pritiskom na eno od »ušesc« oziroma tipk vpišemo njegov položaj v računalnik. Število tipk je pri raznih proizvajalcih različno. Nekateri sistemi uporabljajo kar tri tipke; Microsoft se je odločil za dve, medtem ko ima Applova Lisa samo eno. Tipke se lahko uporabljajo tudi za izbiro ene od možnosti v seznamu – tak je Microsoftov program MultiTool – in prepuščajo miški nadzor nad premikanjem kurzorja. Miške pa lahko uporabljamo tudi z zelo dognanimi programi, kakršne ima na primer Applova Lisa. Tukaj s prvim pritiskom tipke izberemo eno od »ikon« v menuju, z drugim pa poženemo izbrani program.

Glavna prednost vseh mišk in pripadajoče programske opreme je ta, da jih zlahka uporabljajo tudi tisti, ki ne znajo tipkati. Namesto vtipkavanja imena programa ali izbiranja funkcij s pritiskom na določeno črko ali številko uporabnik preprosto premakne miško tako, da kurzor pokaže na zahtevano uporabo ali smer gibanja, in potem pritisne tipko za izvršitev ukaza.

Na žalost miška ne more v celoti nadomestiti potrebe po tipkovnici; nova besedila ali števila je še vedno treba vtipkavati v računalnik, ravnanje s temi informacijami pa je dosti lažje. Preskusi, ki so jih opravili pri Applu med razvojem Lise, so pokazali, da se celo uporabnik, ki računalnikov ne pozna, lahko v manj kot 15 minutah priuči uporabi programske opreme podprte z miško. V običajnem sistemu traja privajanje podobnim programom skoraj 20 ur, predvsem zaradi težav pri tipkanju in ker si je treba zapomniti dolge in zamotane ukaze. Elektronske miške bodo kmalu nepogrešljiv del hišnih računalnikov. So učinkovite in pri uporabi preproste ter omahljivcev ne strašijo tako močno kakor pogled na običajno tipkovnico.

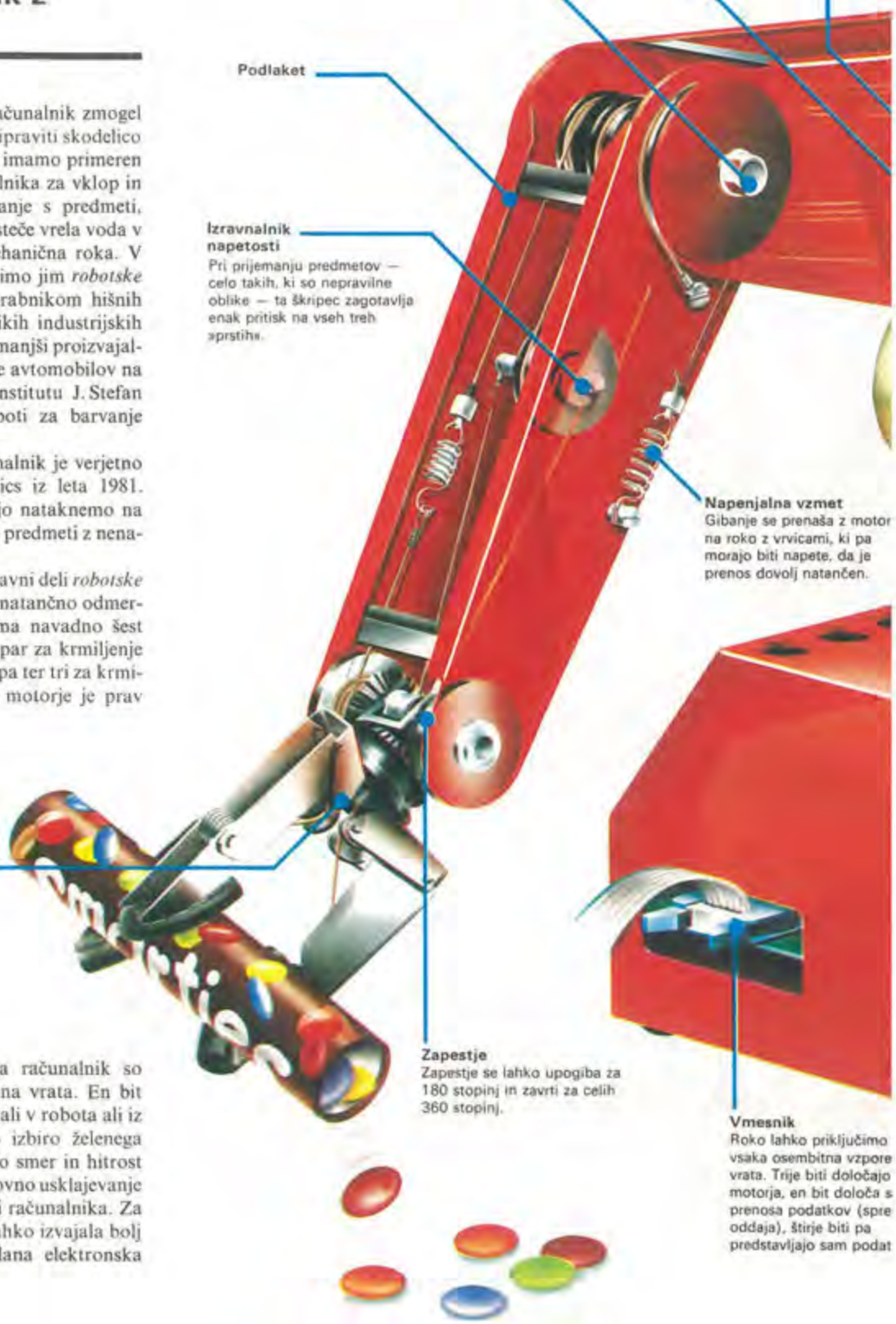
Enoroki rokovnjači

Roke majhnih robotov omogočajo vpogled v programiranje upravljanja. Priključiti jih je mogoče na vsak hišni računalnik z vzporednimi vrati

Ali ne bi bilo imenitno, če bi tvoj računalnik zmogel tudi tako preprosto nalogo, kot je pripraviti skodelico čaja? Nobene težave ni — seveda, če imamo primeren vmesnik — v programiranju računalnika za vklop in izklop kuhalnika. Za fizično ravnanje s predmeti, recimo, ko je treba nagniti lonec, da steče vrela voda v čajno skodelico, pa je potrebna mehanična roka. V zadnjem času so te naprave — pravimo jim *robotske roke* — postale dostopne tudi uporabnikom hišnih računalnikov in so pomanjšave velikih industrijskih rok, ki jih Volvo, Fiat in zdaj tudi že manjši proizvajalci uporabljajo za varjenje in barvanje avtomobilov na tekočem traku. V Gorenju in na Institutu J. Stefan nastajajo tudi že prvi domači roboti za barvanje izdelkov bele tehnike in za kovanje.

Prva *robotska roka* za hišni računalnik je verjetno »armdroid« podjetja Colne Robotics iz leta 1981. Čeprav je roka pritrjena (razen če jo nataknemo na talnega robota), omogoča ravnanje s predmeti z nenavadno visoko stopnjo natančnosti.

Razen kovinskih delov samih so glavni deli *robotske roke* koračni motorji, ki omogočajo natančno odmerjeno gibanje njenih delov. Roka ima navadno šest motorjev: enega za vrtenje roke, en par za krmiljenje »ramenskega« in »komolčnega« sklepa ter tri za krmiljenje gibanja v »zapestju«. Vse te motorje je prav lahko krmiliti z računalnikom.



Za priključitev *robotske roke* na računalnik so potrebna le ena osembitna vzporedna vrata. En bit določa smer pretakanja informacije, ali v robota ali iz njega. Trije naslovni biti določajo izbiro želenega motorja, ostali štirje biti pa določajo smer in hitrost gibanja. Urni signali omogočajo časovno usklajevanje (sinhronizacijo) gibanja roke z ukazi računalnika. Za pospešitev dogajanja in da bi roka lahko izvajala bolj zahtevne gibe, pa so v vezja vdelana elektronska

Zobniški mehanizem
Nazobčani gumijasti pogonski trak in večji zobniki omogočajo zobniško redukcijo, tako da lahko roko premikamo s ponovljivostjo do natančnosti 2 mm.

držala, ki omogočajo sočasno delovanje poljubne kombinacije motorjev tako, da zadržijo ukaz enemu motorju, medtem ko so drugi v pogonu.

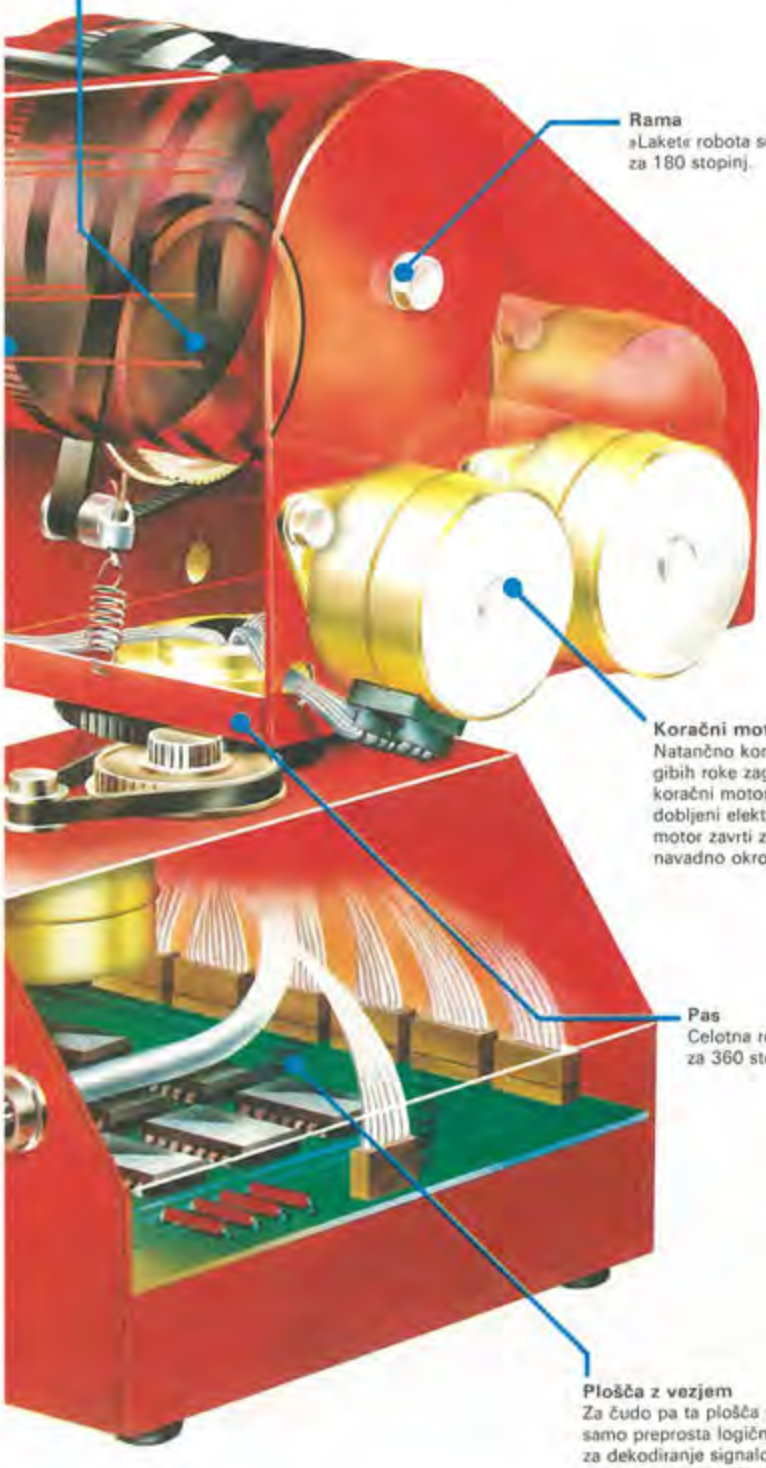
Da bi se roka premaknila v želen položaj in prijela predmet, je treba najprej razdeliti celotni gib v zaporedje preprostih korakov. Vsak motor potrebuje ločeno navodilo, ki natančno določa njegovo gibanje, tako da bo skupni učinek vseh motorjev ravno želeni celotni gib *robotske roke*. To informacijo je mogoče hraniti v računalnikovem pomnilniku, roka pa lahko poljubno mnogokrat ponovi gib. Večina *robotskih rok* je opremljena s pogonskimi programi, ki vsebujejo tudi programe za »učenje« zaporedij gibov.

Če robot ravna s krhkimi objekti — običajen preizkusni kamen je jajce — mora računalnik nadzorovati moč prijema. Če je prijem prešibak, bo jajce ušlo iz roke; če je premočan, bo zdrobil lupino. Uporabljajo se različni načini prenosa te informacije od roke do računalnika. Najbolj preprosti so zasnovani na preprostih mikrostikalih. Ta so lahko nameščena tako, da določajo meje gibanja roke (večina cenenih rok nima tipal), lahko pa so zaradi zaznavanja vnaprej določenih mejnih pritiskov tudi vdelana v dlan.

Na večjih rokah pa deluje nadzorni sistem povečini brez mikrostikal in je zasnovan na zveznem zaznavanju pritiska. Nekatere snovi spremenijo svojo električno upornost, kadar jih podvržemo spremembi pritiska, in te spremembe lahko merimo. Ta metoda je sicer dražja, daje pa zelo natančne rezultate.

Če program ne upošteva povratne informacije od tipal na roki do računalnika, pravimo, da krmili roko z odprto zanko ali deterministično. V našem gornjem primeru bi bil učinek takega programa brez dvoma zdrobljeno jajce. Če pa obstaja kakšna povratna zveza, ki gibanje pod kontrolo programa prilagaja v odvisnosti od podatkov s tipal, tedaj sistem krmilimo z zaprto zanko in pravimo, da je sistem stohastičen. V tem primeru z mikrostikali ali tipali pritiska omejimo zapiranje dlani takrat, ko je jajce že čvrsto v dlani, ni pa še zdrobljeno.

Robotski sistemi z visoko zmogljivostjo imajo običajno več tipalnih sistemov za merjenje svetlobe, toplote in drugih spremenljivk. S temi tipali je mogoče zasledovati dogajanje med gibanjem roke in poročati računalniku, če gre kaj narobe: na primer, da si robotski varilnik veselo žge luknje vase!



Rama
»Laket« robota se lahko zavrti za 180 stopinj.

Koračni motorji
Natančno kontrolo pri vseh gibih roke zagotavljajo koračni motorji. Za vsak dobljeni električni sunek se motor zavrti za en korak — navadno okrog 7 stopinj.

Pas
Celotna roka se lahko zavrti za 360 stopinj.

Plošča z vezjem
Za čudo pa ta plošča vsebuje samo preprosta logična vezja za dekodiranje signalov iz mikroračunalnika. Na njej ni nobenega mikroprocesorskega ali pomnilniškega vezja.

Robot nas uboga
Pisanje programov za krmiljenje *robotske roke* je razmeroma preprosta naloga. V jeziku basic bi bila glavna naloga programiranja v tem, da bi računalnik usposobili za sprejem krmilnih ukazov s tipkovnice in za njihovo sporočanje roki skozi vzporedna vrata z ukazom POKE. Na istih vratih lahko računalnik prebere podatke od *robotske roke* z ukazom PEEK. Kadar pa je pomembna predvsem hitrost krmiljenja, se programiranju v strojnem jeziku ni mogoče ogniti. Forth je jezik, ki omogoča lahkotno programiranje kakor basic, vendar tudi večino hitrosti, običajne pri strojni kodi. Ta jezik postaja dostopen za vse večje število hišnih računalnikov. Segmentom programa, ki so podobni podprogramom ali proceduram, so dana imena, ki jih lahko včlenimo v množico ukazov tega jezika. To omogoča izjemno učinkovito programiranje tudi za take posebne uporabe, kakor je krmiljenje robotske roke. Celotno operacijo prijemanja bi v tem primeru lahko uresničili z enim samim ukazom PRIMI.

Gibajoče se slike

Veliki pomnilniki so omogočili, da lahko hišni računalniki prikazujejo hitro premikajoče se barvne slike

Ena najbolj osupljivih lastnosti hišnih računalnikov je njihova zmožnost, da nam z barvno grafiko prikazujejo barvne slike, ki se lahko celo gibljejo. Nekateri to imenujejo z izrazom računalniška animacija. S skoraj vsakim hišnim računalnikom lahko rišemo točke, črte, kroge in druge bolj zapletene oblike in spreminjamo barve znakov in ozadja.

Za akcijske igre in simulacije pa potrebujemo zmožnost simuliranja hitrega gibanja. Najpreprostejši način za to je, da prikažemo serijo mirujočih slik, ki se med sabo le malo razlikujejo. To metodo uporabljajo skoraj vsi sistemi, ki prikazujejo gibajoče se slike (tudi film in televizija).

Hitrost akcije

Na zaslonu lahko prikažemo znak, ga izbrišemo in takoj zatem spet prikažemo na rahlo spremenjenem mestu. Če hočemo doseči gladko gibanje, mora biti premik minimalen. Minimalen mora biti tudi čas nastanka in izgina slike.

Gibajočo sliko lahko naredimo z grafičnimi ukazi v basicu. Vendar je tako gibanje navadno prepočasno. Lahko je precej hitrejše, če pišemo program v zbirnem jeziku. Tako pisanje pa zahteva zelo veliko prakse in pazljivosti, če nočemo dobiti »utripajoče« slike. Poleg tega je to delo zelo težavno, ker moramo kontrolirati vsak znak posebej.

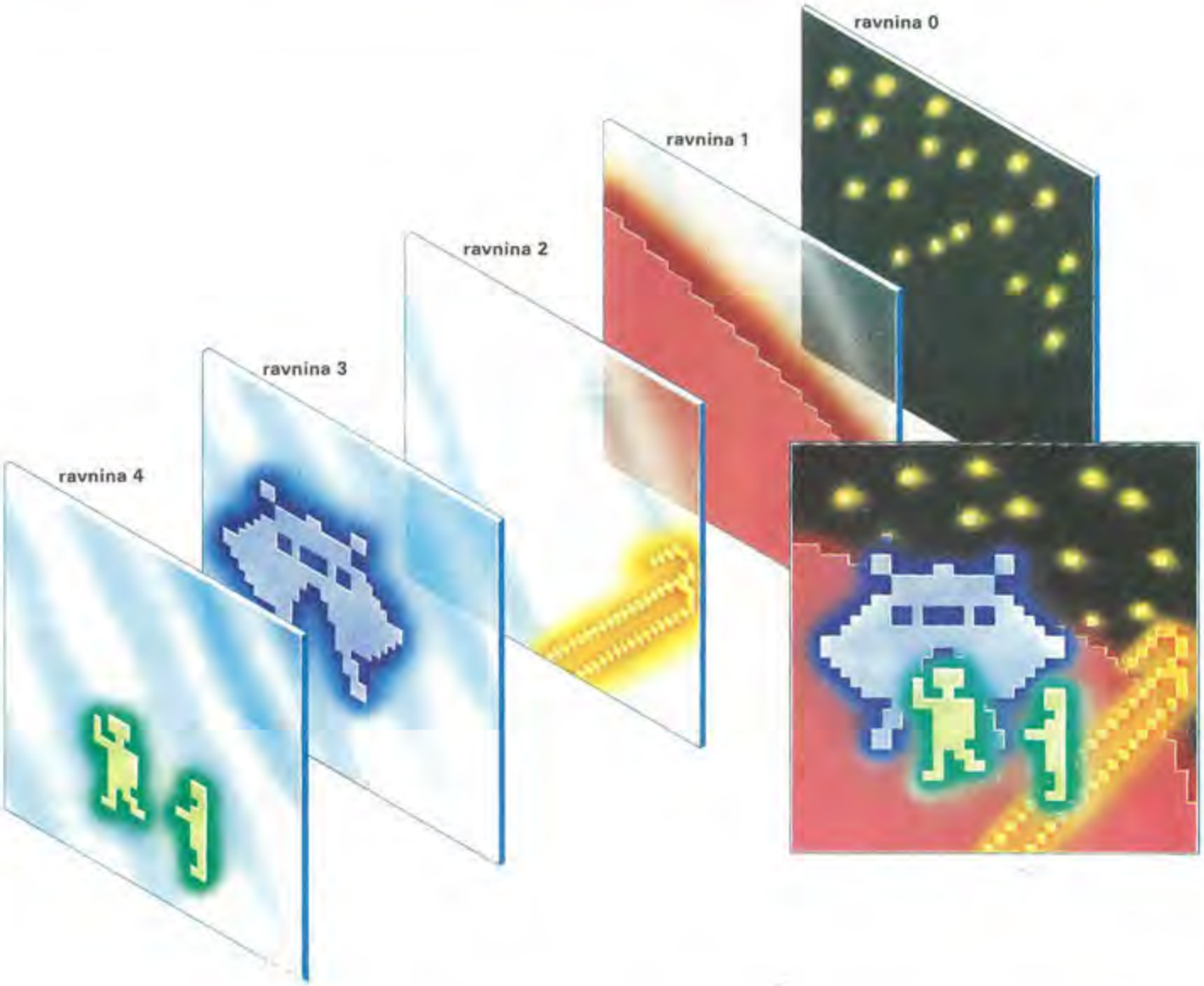
Več računalnikov, med njimi so najbolj znani Commodore 64, Sord M5, Texas Instruments TI99/4A in vrsta računalnikov firme Atari, rešuje ta problem podobno kakor veliki arkadni igralni avtomati. Ta tehnika animacije temelji na uporabi figur (sprite).

Grafični liki so »objekti« ali oblike, ki jih lahko premikamo po zaslonu neodvisno drugega od drugega. To naredimo samo s spreminjanjem vsebine dveh lokacij pomnilnika, ki določata koordinati X in Y (pozicije levo — desno in gor — dol) grafičnega lika. Navadno se X spreminja od 0 do 255, Y pa od 0 do 191. Pri nekaterih računalnikih moramo podati le smer in hitrost gibanja grafičnega lika, drugo pa opravi računalnik sam.

Računalniki, ki »znajo« uporabljati grafične like, imajo posebej za to dodana specialna integrirana vezja. Računalniki, ki teh vezij nimajo, ne zmorejo delati z grafičnimi liki, lahko pa zanje kupimo posebne programe, s katerimi dosežemo podobne, vendar manj zadovoljive rezultate.

Vlak

Vlak na sliki je narejen iz treh grafičnih likov (lokomotiva in dva vagona). Naredili smo jih s programskim paketom Spritemaker na računalniku Commodore 64. Grafične like smo izdelali na zaslonu v velikem merilu (slika desno) in jih nato spravili na kaseto skupaj z liki hiše in drevesa. Nato smo jih zložili nazaj v pomnilnik računalnika in z ukazi POKE opredelili njihove položaje, barve in hitrost na zaslonu. Grafične ravnine smo izbrali tako, da gre vlak za hišo in pred drevesom.



Vsak grafični lik se giblje na svoji tako imenovani grafični ravnini. Grafična slika je sestavljena iz več ravnin, ki so zložene ena vrh druge, tako da jih oko vidi kot eno sliko.

S temi ravninami lahko dosežemo učinek tridimenzionalnosti. Če se namreč dva grafična lika, ki sta na dveh različnih ravninah, srečata, bo tisti, ki je na gornji, prekril spodnjega. Tako lahko vlak prekrije drevo, ko zapelje mimo njega (če je drevo na ravnini za vlakom), hiša, ki je na ravnimi pred vlakom, pa zakrije vlak. Število grafičnih likov je omejeno in je pri vsakem računalniku drugačno (Sord M5 jih ima npr. 32).

Grafični liki so lahko različnih barv. Barve se dajo spreminjati, prav tako lahko grafične like s spreminjanjem nekaj lokacij pomnilnika večamo ali manjšamo.

Grafični liki seveda pokažejo svojo pravo vrednost v igrah, kjer nastopa veliko število hitro se gibajočih likov. Posebno koristna lastnost, ki zelo olajša programiranje iger, je »odkritje trčenja«. Računalnik namreč odkrije, kdaj se dva grafična lika prekrijeta, to informacijo pa lahko izkoristimo, da tedaj skočimo v kak drug del programa. Ko npr. naš izstrelek zadene sovražno vesoljsko ladjo, lahko skočimo v del programa, ki prikaže eksplozijo.

Preden lahko grafične like začnemo uporabljati, jih moramo na neki način ustvariti. Generiramo jih podobno kakor grafične znake.

Črke, številke, posebni znaki in grafični znaki so shranjeni v posebnem vezju, ki se imenuje »znakovni generator«. Kot smo že spoznali, zavzemajo znaki polje (matriko) osem krat osem grafičnih točk. Grafični lik pa lahko zavzame precej večje polje, maksimalna velikost pa je pri različnih računalnikih različna. Na računalniku Commodore 64 lahko npr. kreiramo grafične like, ki zavzemajo največ 21 krat 21 grafičnih točk.

Najprej si narišemo kvadratno mrežo s toliko kvadratki, kolikor je maksimalno število grafičnih točk. Nato počrnimo kvadratke, ki naj predstavljajo naš

Na različnih ravninah
Z grafičnimi liki lahko naredimo kompleksne slike tako, da jih spravimo na različne ravnine, ki so postavljene druga pred drugo. Na zaslonu se te ravnine združijo, tako da vidimo eno samo sliko. Bistvena prednost sistema z več ravninami je v tem, da lahko premikamo vsako ravnino neodvisno od drugih. Ravninam predpišemo prednosti, tako da grafični lik z ravnine z višjo prednostjo prekrije tistega z nižjo, in to nam da učinek tridimenzionalnosti. Večina računalnikov, ki uporabljajo tehniko grafičnih likov, lahko ustavi izvajanje programa in skoči na drugo mesto, brž ko se dva grafična lika dotakneta. Takrat lahko prikažemo drugo sliko (npr. eksplozijo) in povečamo števec, ki spreminja izid igre.

Razsežnosti grafičnih likov
V tabeli si lahko ogledamo nekatere lastnosti štirih najbolj pogostih hišnih računalnikov z možnostjo generacije grafičnih likov. To so: Commodore 64, Sord M5, TI99/4A in Atari 800.
Vsi imajo na voljo po 16 različnih barv, Atari pa doseže število 256, ker lahko vsako barvo spreminjamo v 16 odtenkih. Pri programiranju iger je zelo koristno, če lahko računalniki zaznajo, kdaj se dva grafična lika dotakneta (npr. ko trčita dve vesoljski ladji). To zmorejo vsi računalniki v tabeli. Učinek tridimenzionalnosti lahko dosežemo tako, da spravimo grafične like na različne ravnine. Seveda je boljši računalnik z več ravninami. Maksimalna velikost grafičnega lika je izražena z grafičnimi točkami. Za nekatere računalnike lahko kupimo posebne programske pakete, ki nam olajšajo ustvarjanje grafičnih likov.

mikroračunalnik	število barv	odkritje trčenja	ravnine	velikost lika	programski paket
Commodore 64	16	Da	8	24 × 21	Da
Atari 800	256	Da	16	16 × 16	Da
Texas TI 99/4A	16	Da	28	32 × 32	Da
Sord M5	16	Da	32	8 × 8	Ne

grafični lik. Počrnjene kvadratke označimo z 1, prazne pa z 0. Ker računalnik razume samo ta dva znaka, spravi grafični lik v pomnilnik na enak način. 1 mu pomeni barvno grafično točko, kjer pa je 0, ni grafične točke. Računalnik pa shranjuje podatke po zlogih, zato moramo ničle in enice, ki smo jih dobili, organizirati v skupine po osem. V priročniku, ki ga dobimo z računalnikom, lahko najdemo natančen opis načina, kako mrežo z enkami in ničlami razbijemo v zloge. Grafični lik opišemo računalniku s posebnim programom v basicu. Ker uporablja basic na nekaterih računalnikih le desetiška števila, moramo vsak zlog pretvoriti v desetiški ekvivalent. Tako dobimo števila od 0 do 255. Pretvorbo izvedemo tako (glej poglavje Ko je 1 + 1 = 10), da shranjeno levo mesto pomnožimo s 128, naslednje s 64 in tako vse do zadnjega mesta, ki ga pomnožimo z ena. Nato vse te zmnožke seštejemo.

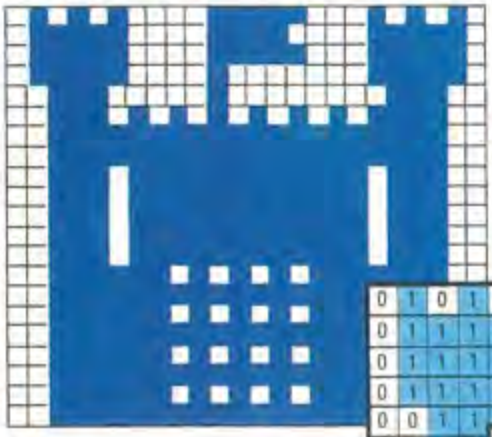
Skupina desetiških števil, ki jih tako dobimo, definira grafični lik. Te številke prenesemo v računalnik s programom v basicu. Pri tem so podrobnosti pri vsakem računalniku drugačne. Ko je grafični lik shranjen v pomnilniku računalnika, je delo z njim zelo lahko. S preprostimi ukazi določimo trenutno pozicijo vsakega grafičnega lika na zaslonu, spreminjamo barve, jih večamo in manjšamo, zaznavamo trenutek, ko se dva grafična lika prekrijeta, in počnemo še druge vragolije.

Ustvarjanje grafičnih likov pa si lahko zelo olajšamo, če kupimo poseben program, ki nam pri tem pomaga. Ta program nam na zaslonu nariše mrežo v velikem merilu, grafični lik naredimo s premikanjem kurzorja po mreži. Vse preračunavanje, ki smo ga prej opisali, opravi pod kontrolo tega programa računalnik sam. Ko končamo oblikovanje, mreža izgine, na zaslonu pa se pokaže grafični lik v pravi velikosti. Če nam ni všeč, lahko pokličemo mrežo nazaj in lik spremenimo.

Možnost uporabe grafičnih likov pomeni pravo revolucijo v grafiki hišnih računalnikov. Zdaj lahko vsakdo preprosto in učinkovito ustvarja hitro gibajoče se barvne slike.

Nastanek grafičnega lika

Najboljše je, če začnemo risati nov grafični lik na kosu papirja s kvadratno mrežo. Lik narišemo tako, da pobarvamo želene kvadratke. Lik na sliki je enostaven, na nekaterih računalnikih pa lahko naredimo tudi večbarvne.
Sedaj uporabimo karo papir, na katerem vsak pobarvan kvadratek ponazorimo z 1, praznega pa z 0. Ker je računalnikov pomnilnik razdeljen v zlog (po 8 bitov), moramo celotno mrežo razbiti v skupine po osem kvadratkov. Vsak zlog, ki smo ga tako dobili, spremenimo v desetiško število. Skrajni levi bit pomnožimo s 128, naslednjega s 64 in tako naprej. Rezultate množenja seštejemo in dobimo desetiško število med 0 in 255. To nato uporabi program v basicu.



0	1	0	1	0	1	0	0
0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	0
0	0	1	1				
0	0	1	1				

0	0	1	1	1	1	1	0
128	64	32	16	8	4	2	1
↓	↓	↓	↓	↓	↓	↓	↓
0	0	32	16	8	4	2	0 = 62

Vojna in mir

V tem namišljenem scenariju Britanci pričakujejo bombni napad na letališče v Lutonu, ki ga varuje eskadra lovskih letal.

LUTON

Nemška letala se zberejo nad Londonom, ki ga varujejo le manjše britanske letalske enote. Kljub temu Nemcem zadajajo hude izgube.

LONDON

DUNKIRK

Medtem ko se vračajo v Francijo, Nemci uspešno bombardirajo radarsko postajo Dunkirk v Kentu, ki je ne varujejo britanski lovci.

DIEPPE

CHERBOURG

Bitka za Anglijo
Lovsko poveljstvo (Fighter Command firme Strategic Simulations Inc. za računalnik Apple) je primer zelo zahtevne vojne igre, ki se igra na računalniku. Preden se igra prične, se mora igralec odločiti, kakšna letala bo uporabljal, kakšno bo vreme itd. Igra se s simboli na zemljevidu, dodatne informacije pa daje besedilo na zaslonu. Skupaj z igro dobite tudi tiskana navodila in zemljevid s papirnatimi figurami za pomoč pri igranju.

Kupite lahko računalniške igre, kjer se lahko izkažete kot vojaški strateg in taktik v zgodovinskih ali izmišljenih bitkah

Taktično tankovsko poveljstvo TAC
TAC (firme Avon Hill) je mogoče kupiti za mikroračunalnike Atari, Apple, IBM PC in Commodore 64. Igra predstavlja tankovske spopade v drugi svetovni vojni. TAC lahko igra eden ali dva igralca, ki imata na izbiro pet različnih scenarijev, poveljujeta pa britanskim, ameriškim, sovjetskim in nemškim tankovskim enotam.

Moderni generali posvečajo veliko pozornost vojnim igram, ki jih igrajo, da bi preizkusili svoje odgovore na zamišljene sovražne napade ali »sovražne scenarije«, kakor pravijo temu. Da bi lahko igrali te umetelne igre, so razvili velike sisteme strojne in programske opreme, ki simulirajo vsa znana dejstva mogočega spopada, na primer začetne položaje zavezniških in sovražnih sil, stanje zalog in rezerv itd. Ti sistemi upoštevajo tudi slabo vreme, spremembe sovražnikove taktike, učinke delovanja pete kolone ali kakršnegakoli števila še drugih dejavnikov, ki bi utegnili vplivati na uspeh izvajanja vojaške operacije. Ena glavnih nalog ameriškega radarskega obrambnega sistema NORAD, postavljenega v gorah Cheyenne v zvezni državi Wyoming, je, da z najnovejšimi podatki stalno ocenjuje vojaško zmogljivost ZDA in ZSSR ter pomaga pripravljati vojaški odgovor na vsako novo grožnjo.

Seveda, igranje vojnih iger je bilo doslej za amaterskega generala nekoliko manj umetelno. Da bi bile igre dovolj zapletene, je moral uporabljati številne tabele, debele knjige pravil in brezštevilne mete kock. Že samo zaradi količine trdega dela, ki ga je bilo treba opraviti za igranje vojnih iger, so bile privlačne le za sorazmerno majhno skupino navdušencev. S prihodom hišnih računalnikov in dostopnostjo programov za igranje vojnih iger pa je vse dolgočasno »štabno delo« izginilo in namesto njega so nastale napete igre, ki so po razburljivosti in izzivu enake katerikoli zdaj priljubljeni računalniški igri na tržišču.

Na voljo je neznansko število različnih vojnih računalniških iger. Mogoče je poustvariti ali simulirati praktično kakršenkoli način vojskovanja, od antičnih Grkov do teoretičnega spopada sil NATO in Varšavskega pakta kdaj v prihodnosti. Lahko bojujete zračne in pomorske bitke, vojne v vesolju in celo vojne med mitološkimi imeriji. Obseg takega vojskovanja nima meja.

Legionar
Ta simulacija vojne med Cezarjevimi enotami (vodite jih vi) in barbari (ki jih vodi mikroračunalnik Apple ali Atari) se igra v realnem času. Pehoto, konjenico in druge sile predstavljajo simboli, ki jih izberete in premikate s kontrolno ročico. Beli kvadrat na sliki je kurzor. Igro izdeluje podjetje Avalon Hill.

Zgodovinske vojne igre vam dajejo priložnost, da ugotovite, kje je Napoleon naredil napako pri Waterlooju, ali da premagate Hitlerja pri njegovem napadu na SZ v letu 1941. Vesoljske vojne igre dajejo še več možnosti vaši domiselnosti. Ne samo da lahko manevrirate z ladjevji vesoljskih ladij po galaksijah, izbirate lahko tudi modele ladij, ki jih potrebujete. Seveda morate pri tem narediti nekaj kompromisov – če hočete hitrejše ladje, morate morda žrtvovati nekaj orožja na njih in boljši oklepi lahko zmanjšajo preskrbo z gorivom. Izbrati morate tak kompromis, ki najbolj ustreza vašemu načinu vojskovanja.

Drugače kakor njihove starejše predhodnice računalniške igre ne zahtevajo posebnih spretnosti ali znanja in večina jih je opremljena le s kratkimi navodili in namigi za začetnike. Vredno pa je vedeti, da so nekatere igre označene kot začetniške, srednje težke ali težke. Če nameravate pričeti z igranjem strateških iger, boste morda storili bolje, če začnete na začetniškem nivoju in se najprej naučite nekaj temeljnih zasnov vojnih iger in strateške simulacije, preden pričnete igrati igre na višji ravni.

Oblika neke igre je prilagojena potrebam načina vojskovanja, ki ga predstavlja. V splošnem pa se igrajo te igre na velikih zemljevidih. Če je zemljevid prevelik, da bi ga lahko naenkrat prikazali na zaslonu, običajno zaslon predstavlja »okno«, skozi katero gledamo zemljevid in s krmilno ročico premikamo po zemljevidu. Pri zgodovinskih simulacijah načrtovalci iger poskušajo kolikor mogoče natančno posneti teren, na katerem je bitka potekala. V igri »Računalniški Bismarck« (ime bojne ladje v 2. svetovni vojni) je kraj dogajanja Severni Atlantik. To okolje načrtovalcem igre ni postavljalo hudih zahtev glede grafične predstavitve. Te zahteve pa so pri igri »Bitka za Normandijo« veliko hujše. Ne samo, da mora biti teren dogajanja zvest posnetek pravega, prave morajo biti tudi druge zemljepisne značilnosti, peščine in obrežje, vasi, mesta, reke. Pri izmišljenih bitkah ima načrtovalec več možnosti, da igralcu omogoči čim boljše izkoriščanje svojih vojaških sil. Vendar je v tem primeru pri načrtovanju treba postaviti dovolj ovir in protitež, da igra ne bi postala prelahka za eno ali drugo stran.

Zemljevid ima narisano mrežo, ki ga deli enako, kakor kvadrati delijo šahovnico. Pri vojnih igrah je ta mreža pogosto šesterokotna namesto kvadratna. Vsak kvadrat ali šesterokotnik ima določeno vrednost, odvisno od terena, ki ga omejuje. Ta vrednost predstavlja stopnjo težavnosti za neko vojaško enoto, da bi v ta del bojišča prodrla ali ga prešla. Napor pri premikanju skozi področje zmanjša gibljivost enote za določeno vrednost. Če postane gibljivost enote enaka nič, ali manjša od vrednosti na področju, na katero hoče prodreti, mora za nadaljnje prodiranje počakati na naslednjo potezo svojega »poveljnika«.

Igre so običajno razdeljene na poteze, te pa predstavljajo pretečeni čas od pričetka igre. Vsak igralec ima nekaj ciljev, ki jih mora v določenem času doseči, če hoče zmagati. V večini primerov ni potrebno ali celo mogoče doseči vseh ciljev. Torej mora igralec najprej oceniti svoje možnosti in glede nanje določiti, čemu bo dal v svoji strategiji prednost. V scenarijih računalniških vojnih iger je pogosto nasprotnikova naloga, da prepreči napadalcu doseči njegove cilje. Tudi v tem primeru verjetno ni mogoče varovati vsega in se mora branilec odločiti, kdaj naj zapusti brezupne položaje, kako dolgo naj brani svoje utrdbe, in ali naj se odloči za protinapad, da bi ponovno osvojil izgubljene položaje ali pa onemogočil nasprotniku priprave za nov napad.

Igralec komunicira s programom z grafično in besedno predstavitvijo svojih sil na zemljevidu. Grafični prikaz vsebuje položaj posameznih enot na bojišču, besedni prikaz pa daje podatke o njihovi bojni učinkovitosti in gibljivosti.

Igralec premika svoje enote tako, da jih imenuje s kurzorjem, ali da mu jih računalnik prikazuje drugo za drugo in pri izbrani pritisne na tipko. Ko je enota imenovana, je treba izdati ukaz za njen premik. Na zemljevidu s šesterokotno mrežo bi ukaz 1 poslal enoto proti severu, 2 proti severovzhodu, in tako naprej okoli po straneh neba. Vse več iger igramo s krmilno ročico ali s kroglo in v tem primeru igralec lahko enoto kar »pobere« in premakne na zaželeno mesto. Konec premikanja neke enote pogosto določa ukaz FINISH, ali skrajšano F. Pa še tedaj nekatere igre dovoljujejo igralcu, da enoto preimenuje in jo spet premika, dokler njena gibljivost ni izčrpana. Ko so opravljeni vsi premiki, igralec to sporoči računalniku z ukazom EXECUTE ali E. Računalnik nato prične bojno fazo igre.

Vzhodna fronta
Pri tej igri igralec prevzame vlogo nemške vojske, ki skuša leta 1941 doseči Moskvo, računalnik pa vodi sovjetske obrambne sile. Igro je napisal Chris Crawford, prodaja pa jo Atari. Med novostmi, ki jih prinaša igra, je »povečava« dela bojišča. Zemljevid s spreminjanjem letnih časov spreminja barvo. Jeseni listje odpade, pozimi tla prekrije sneg in reke zamrznejo. Ugotavljajo, da je v tej igri izredno težko priti do Moskve.

Med bojno fazo računalnik pokaže enote, ki se lahko spopadejo s sovražnikom, in daje podatke o relativni moči posameznih vmešanih enot. Na podlagi teh podatkov lahko igralec sprejme ali zavrne vsak boj, ki mu je ponujen. Ko so končani vsi napadi in so njihovi učinki izračunani ter prikazani, dobi potezo nasprotnik.

Za mnoge ljudi je glavna privlačnost strateških iger v tem, da ni nobene »prave« rešitve problemov, ki jih igra postavlja. Igralec uživa ob tem, da premaguje fizične in logistične probleme terena, na katerem deluje, ter da sprejme intelektualni izziv in s sredstvi, ki jih ima na voljo, skuša premagati nasprotnika. Seveda, vsak strateg bi rad zmagal z najbolj drznimi zamislimi in skrbno nastavljenimi pastmi, predvsem pa si želi zmagati!

Šola na zaslonu

Računalniki so našli pot tudi v učilnice in na voljo je veliko zanimivih izobraževalnih programov

Računalnik v osnovni šoli
Namen računalnika v osnovni šoli ni le seznanjanje otrok z novimi tehnologijami, temveč tudi uporaba računalnika pri različnih predmetih, od biologije do tujih jezikov. Računalnik je idealen, potrpežljiv učitelj, ki menja področje šele potem, ko so vsi problemi rešeni, in omogoča slabim in dobrim učencem, da si sami izberejo hitrost dela. Zato ima velik pomen pri izobraževanju. Z uporabo računalnika se otrok uči, kako je treba problem analizirati in rešiti.

Računalniki se ne uporabljajo več samo pri predmetu »računalništvo«, temveč jih pedagogi uporabljajo tudi pri poučevanju slovnice, matematike, tujih jezikov in za pomoč učencem, ki se teže učijo.

Na trgu je veliko izobraževalnih programov za hišne računalnike, toda pedagogi se pogosto pritožujejo zaradi njihove slabe kvalitete. Razlog za to je dejstvo, da je še malo programov, ki združujejo izobraževanje in poznavanje računalnikov. Programerji nimajo izkušenj pri učenju, učitelji pa večinoma niso vešči programerji. Zato prihaja do začetniških napak pri izobraževalnih programih. Program, ki je shranjen na disketi ali disku, po navadi še ni zadosten za pravilno uporabo, zato sta dober opis in navodilo sestavni del programa. Brez tega je lahko program za učence neuporaben. Za novince v svetu računalništva so pomembne vse podrobnosti, ki se zdijo poznavalcu umevne same po sebi, zato morajo biti navodila pisana zelo skrbno.

Dobro programiranje zahteva poznavanje napak, ki jih lahko naredi začetnik. Ta lastnost programa je enako pomembna kakor njegova izobraževalna plat. Dobro programiranje pomeni več ko samo čiščenje programa, dokler ne dela, kar naj bi delal, ko je pritisnjena prava tipka. Pomeni tudi zagotovilo, da program ne naredi ničesar, česar ne bi smel narediti, če je pritisnjena napačna tipka. To je najtežji del pisanja programov. Program mora delovati pravilno kljub še tako hudim napakam, ki jih lahko naredijo otroci, hkrati pa jim mora ohraniti prepričanje, da je delo z računalnikom preprosto in zabavno. Kljub temu obstaja veliko število izobraževalnih programov, primernih za uporabo doma in v šoli. Računalniki so čudovito orodje za poučevanje in prav je, če pri izbiri programa za svojega otroka poznate razne načine uporabe računalnika za poučevanje.

Računalnik lahko uporabimo tako, da poučuje otroka. Če je program dober, bo otrok verjetno prevzet

Tajni agent

Na trgu je veliko izobraževalnih programov. Ta se imenuje Tajni agent. Ujeti morate znanega sovražnikovega vohuna, ki je nekje v Evropi, preden on uniči vse vaše agente. Preden pa se lahko spoprimete z njim, morate razvozlati klobčič neznank o tem, kje je. Lov se pričenja.

Glavnega vohuna lahko ujamete le v večjem mestu. Vohun je na istem kraju le nekaj ur. Ko mislite, da veste, kje je, imate na izbiro potovanje z vlakom ali z letalom. Sami se morate odločiti, kaj je pomembnejše, čas ali denar.

Vsakič, ko vam eden vaših agentov pošlje sporočilo, se prižge na zemljevidu lučka v mestu, od koder je sporočilo poslano. Če je vaš agent izločen iz igre, bo sporočilo prestreženo, preden ga sprejmete. Potem lahko najamete novega agenta, ki pa ga morate plačati.

Ovaduhi vam z veseljem ponujajo naprodaj svoje informacije, ki pa so zakodirane, zato morate prositi za pomoč center v Londonu. Na koncu igre si boste zapomnili imena in lego vseh večjih mest v Evropi, navadili pa se boste tudi razumno razporejati čas in denar.

in motiviran za učenje. Običajni tip izobraževalnega programa se da opisati kot »učenje in vaja«. Računalnik otroku pokaže zgled in potem zahteva rešitve podobnih problemov. Običajno program hrani podatke o uspešnosti učenca. Če učenec odgovori pravilno, ga računalnik pohvali, če pa stori napako, mu prijazno svetuje, naj poskusi znova.

Odločitev o tem, kateri program je pravi za vašega otroka, je odvisna od otrokove starosti, računalnika in snovi, ki se jo otrok uči v šoli.

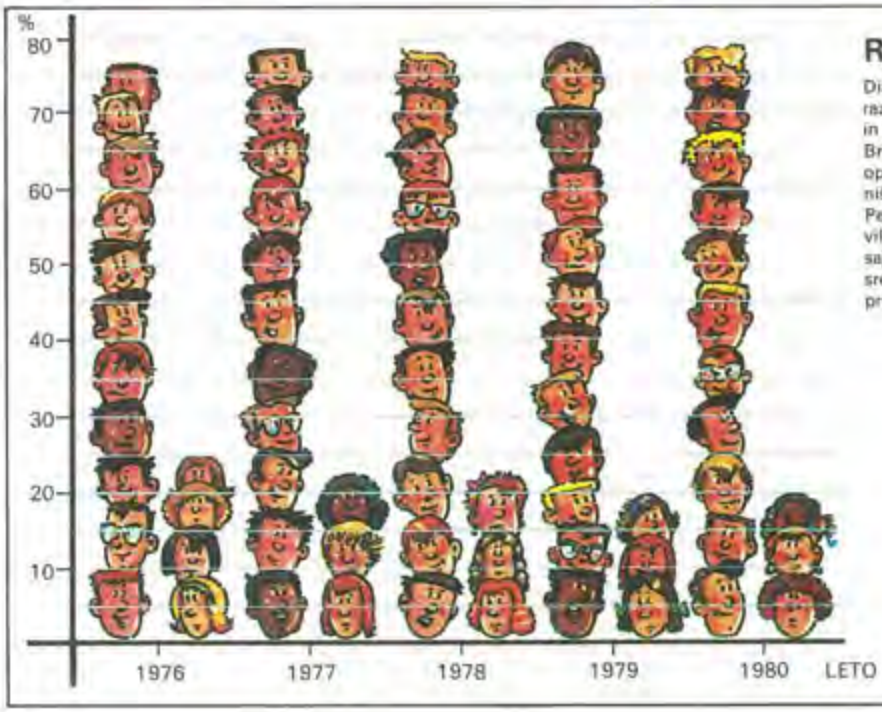
Razumljivo je, da je za boljše računalnike na voljo več izobraževalnih programov. Nekateri proizvajalci pa so dali izobraževanju še poseben pomen. Zelo velik izbor izobraževalnih programov je na voljo za računalnike Apple, Commodore PET, Tandy, BBC Micro, Sinclair in Texas Instruments. Izobraževalne programe je moč dobiti pri proizvajalcih hišnih računalnikov (za njihove računalnike) ter pri različnih neodvisnih proizvajalcih, ki pišejo programe za vse priljubljene hišne računalnike.

tor glasu. TI ima tudi verzijo programa logo, ki pa spada v drugo kategorijo programov.

Za starejše otroke je na voljo še več raznovrstnih programov. Za otroke od osmega do enajstega leta obstaja več programov, ki se razlikujejo po kvaliteti in zahtevnosti, večina pa se jih usmerja v krepitev otrokovih osnovnih spretnosti in veščin in v razširjanje njegovega obzorja. Otroci v tej starosti kažejo zanimanje za razna področja, od glasbe do tujih jezikov. Izobraževalni programi so temu prirejeni.

Za srednješolce pa je že kar preveč programov. Pomembno je, da študij vašega otroka doma ni v nasprotju z delom v šoli.

Naslednja kategorija izobraževalnih programov je namenjena predvsem otrokom do trinajstega leta. V teh letih otroci še vedno iščejo način, kako se bodo učili, zato morajo biti programi, ki jih usmerjajo v uporabo računalnikov pri učenju, zares dobri. Najbolj je znan program logo in izpeljanke tega programa so dostopne na računalnikih Atari, Tandy, Apple, Texas



Računalništvo

Diagram na levi kaže razmerje med študenti moškega in ženskega spola v Veliki Britaniji, ki so uspešno opravili izpit iz osnov računalništva v letih od 1976 do 1980. Pedagogi upajo, da se bo število deklet bistveno povečalo, saj se zdaj otroci obeh spolov srečujejo z računalniki že v prvih letih šolanja.

Pravilna izbira

Za otroke do osmega leta starosti večina programov zajema temeljno znanje pisanja, branja in računanja. Pri proizvajalcu Texas Instruments (TI) so razvili zelo privlačne izobraževalne programe za majhne otroke. Prodaja hišnega računalnika TI-99/4A v Evropi hitro narašča. Večina staršev je navdušena nad izobraževalnimi programi, ki so jih izdali pri TI in Scott, Foresman & Co. v ZDA. 99/4A je v bistvu 16-bitni računalnik, kar pomeni, da so programi, pisani v strojnem jeziku, najbrž veliko boljši od tistih, pisanih na bolj običajnih, 8-bitnih hišnih računalnikih.

To dokazujejo TI programi »Slovnica za začetnike«, »Seštevanje in odštevanje« (1 in 2) in »Čarobna števila«. Ti programi so shranjeni na plastičnih vtičnih modulih za TI-99/4A in jih lahko uporablja vsak majhen otrok. Na računalnik se da priklopiti sintetiza-

Instruments, Research Machines, Commodore in IBM. Podobne programe obetata tudi BBC in Sinclair, vendar za zdaj še niso na voljo. Program logo spodbuja otroke med šestim in dvanajstim letom k spoznavanju računalnikovih risarskih zmogljivosti (geometrije) z uporabo »želve«. Otrok odkriva, kako si želva lahko zapomni posamezne dele programa, in se ob zgledih uči toliko časa, da lahko riše like iz sveta svoje domišljije. Pri delu s tem programom otroci spoznavajo osnove matematike.

Dobro izobraževalno programsko opremo je težko izbrati. Obiski na razstavah računalniške in programske opreme vam lahko omogočijo primerjanje in izbiranje najboljših programov. Pomanjkanje dobrih programov, ki bi ustrezali hkrati izobraževalnim in računalniškim zahtevam, pa ne bo trajalo dolgo. Vsak mesec se pojavljajo novi programi, ki so lahko pomembna spodbuda pri razvoju vašega otroka.

Ekspertni sistemi

Pomen komunikativnosti računalniških programov

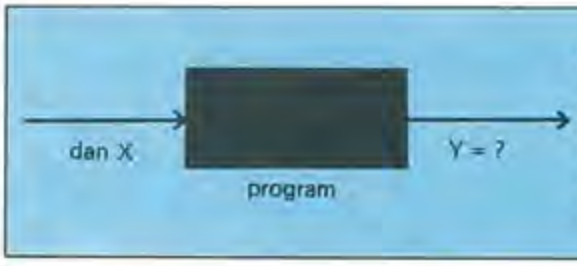
Večinoma se obstoječi računalniški programi vedejo kot *črne škatle*, ki za dane vhodne podatke izračunajo pripadajoče rezultate. Tako vedenje programov je ustrezno za situacije, ko program vedno izračuna pravilno rešitev, uporabnika pa za dane podatke zanimajo zgolj rezultati brez kakršnekoli razlage. Za primere, ko je vedno mogoče izračunati pravilno rešitev, pravimo, da obstaja »zanesljiv klasičen model«.

Pogosto pa uporabnik potrebuje poleg golega rezultata tudi odgovor na vprašanje: Kako si prišel do rezultata? Tako pojasnilo je potrebno npr. za to, da uporabnik dojame »miselnost« programa in se prepriča, ali je rezultat pravilen ali ne. Vedenje sistema bi moralo biti v tem smislu transparentno oziroma uporabniku razumljivo. Program naj bi se vedel kot *prozorna škatla*.

Oglejmo si nekaj situacij, ko črne škatle ne zadoščajo zahtevam uporabnika.

a) Situacija, ko za rešitev problema obstaja zanesljivi klasični model, vendar uporabnika zanima tudi razlaga, kako je sistem prišel do rezultata. Primeri takih situacij so sistemi za poučevanje z računalnikom, npr. sistem, ki naj bi na podlagi modela dane naprave in ustrenega simulatorja razložil, kako naprava deluje. Tu klasična črna škatla ne zadošča, potrebujemo prozorno. Primer je program, ki analizira električna vezja s tem, da izračuna časovni potek veličin v vezju. Čeprav lahko tak sistem izračuna časovni potek napetosti in tokov npr. v vezju astabilnega multivibratorja in take poteke prikaže npr. v obliki diagrama, prikaz takih diagramov ne zadošča, da bi učenec razumel bistvo mehanizma osciliranja v takem multivibratorju. Za razumevanje samega mehanizma bi potrebovali kvalitativni opis dogajanj v vezju, sestavljen iz stavkov kot npr. »napetost na kondenzatorju C1 raste, dokler ne doseže kritične vrednosti; s tem raste tudi potencial na bazi transistorja T2; ob kritični vrednosti potenciala na bazi T2 se T2 odpre...«. Ena možnost, da sistem sestavi tako razlago, je ta, da spremeni kvantitativne rezultate, ki jih dobimo iz klasičnega numeričnega modela, v kvalitativni opis. Vendar ni mogoče generirati smiselnega in jedrnatega kvalitativnega opisa, če sistemu niso znane tudi »kvalitativne« oz. vzročne relacije med elementi v modelu.

b) Situacije, ko za rešitev problema obstaja zanesljivi klasični model, vendar uporabnika zanimajo tudi odgovori na vprašanja, kakor so: Kakšen mora biti vhod X, da bo izhod Y? Kako bodo na (kvalitativno) vedenje sistema vplivale spremembe v modelu (spremembe relacij med elementi modela)? V takem primeru klasične črne škatle, ki računajo v smeri vhod—izhod (izračunaj Y za dani X), ne zadoščajo. Za zgled navedimo zahteve uporabnika, ki razvija sistem oz. napravo in ga zanima, s kakšnimi spremembami v modelu bi se dalo doseči zahtevano vedenje sistema oz. naprave.



Slika 1: Tipično vedenje klasičnih računalniških programov. Program se vede kot črna škatla: iz danih vhodnih podatkov X izračuna pripadajoče rezultate Y. Program pa ni zmožen pojasniti, na kakšen način je prišel do rezultata.

c) Položaji, kjer klasični model ni zanesljiv. V tem primeru je že tudi pri odgovarjanju na najbolj osnovno vprašanje, kakšen je Y za dani X, potrebna razlaga, zato da uporabnik vidi, ali naj rezultatu zaupa ali ne. Take položaje srečujemo v vsaki zahtevnejši uporabi računalnikov v vodenju kompleksnejših procesov, ali pa v prognostiki in diagnostiki na zahtevnejših področjih, kakor so ekonomija, demografija, medicina itd.

Za zgled neustrezno izvedene avtomatizacije kompleksnega proizvodnega procesa omenimo primer jeklarne v Hoogovensu, ki je postal zaradi svoje publicitete že skoraj klasičen (Voysey 1977). Ko so v jeklarni zamenjali polavtomatski sistem za vodenje procesa s popolnoma avtomatskim, računalniškim vodenjem, je proizvodnja nepričakovano padla. Raziskava vzrokov je pokazala, da je bil glavni razlog slaba komunikativnost računalniškega sistema do operaterjev, ki naj bi opazovali proces in posegli vmes v primerih, ko proces kljub računalnikovemu upravljanju ne teče v želeni smeri. Ker operaterji niso razumeli računalnikovih odločitev oz. niso imeli vpogleda v delovanje krmilnih programov, so intervencijo odlagali tako dolgo, da je bilo za pravočasno preusmeritev procesa že prepozno.

Podobno je pri računalniškem vodenju nuklearnih elektrarn. Ukrepe, ki jih predlaga računalnik, je nujno treba preveriti. Preiskave nekaterih dejanskih nesreč v nuklearnih elektrarnah so pokazale, da so operaterji ukrepali neprimerno ravno zaradi premajhne komunikativnosti obstoječih kontrolnih programov (Kopec 80).

V obeh primerih, pri jeklarni in pri nuklearki, je problem v tem, da model, na podlagi katerega delujejo kontrolno-krmilni programi, ni zanesljiv. Zaradi za-

pletenosti procesov in nezanesljivosti sprotnih vhodnih podatkov, dobljenih z meritvami, ni mogoče pričakovati povsem zanesljivih rezultatov modela in je večja komunikativnost edini izhod.

Do podobnega absurda prihaja v makroekonomskih modelih, kjer je prozornost programov, ki izpolnjujejo te modele, navadno popolnoma prezrta. Model, tipično sestavljen iz nekaj sto ali tisoč enačb, je seveda samo (nezanesljiv) približek dejanskim dogajanjem. Da lahko model poženemo na računalniku, je treba vstaviti podatke o tisočih parametrov, ki navadno niso vsi znani ali so spet nezanesljivi.

Postopek od podatkov do rezultatov je npr. reševanje sistema nekaj sto ali tisoč enačb; ta numerični postopek je tako obsežen, da je za uporabnika seveda tipična črna škatla. Ko uporabnik dobi rezultate iz modela, ki se ne ujemajo z njegovimi intuitivnimi predstavami, nima praktično nobene možnosti, da bi preveril, kje je morebitna napaka; v modelu samem, v podatkih, ali pa je rezultat na podlagi modela nazadnje le pravilen. Kadar se izkaže, da daje model nepravilne rezultate, razvijalci modela navadno premišljajo takole: model ni dovolj natančen, zato je treba natančneje opisati relacije med veličinami v sistemu. V modelu se poveča število enačb in parametrov. Rezultat takega napihovanja modela pa se kaže v še večji nerazumljivosti in nepreglednosti modela.

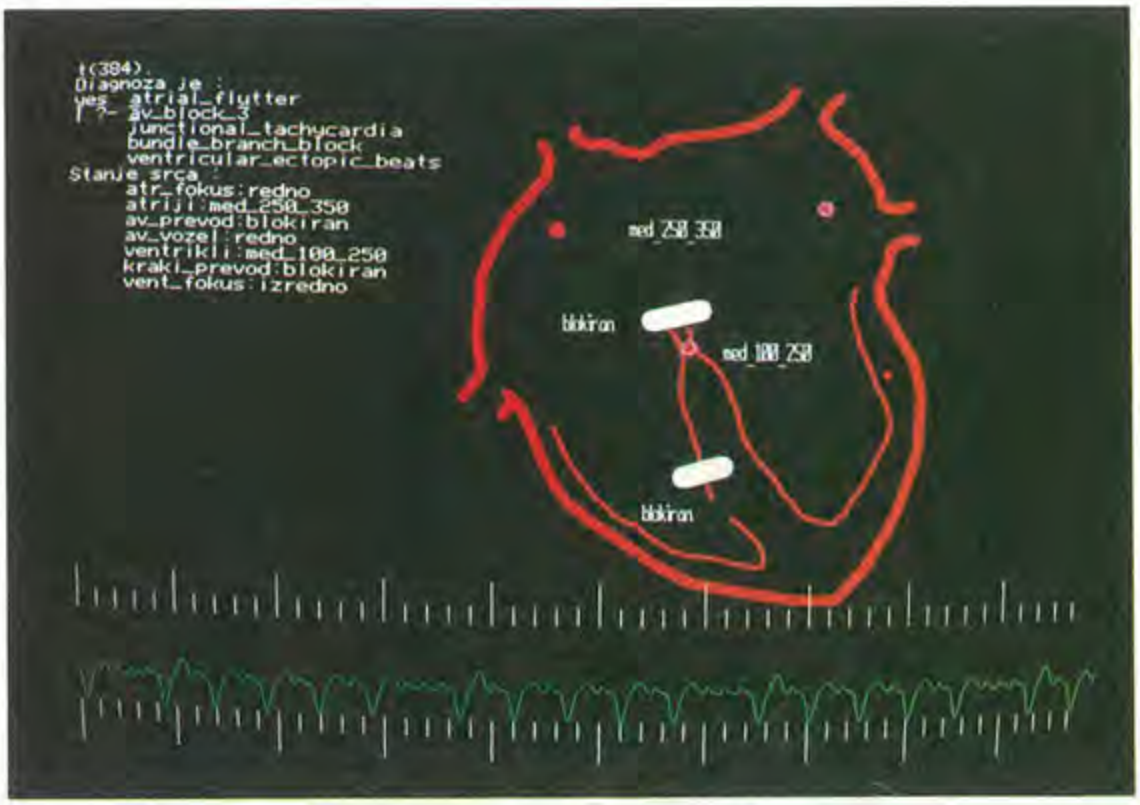
Metodologija ekspertnih sistemov

Neustreznosti črnih škatel se je prvič resno zavedela filozofija ekspertnih sistemov. V najširšem smislu razumemo pod pojmom *ekspertni sistemi* računalniške programe, uresničene z različnimi metodami umetne inteligence, ki podobno kakor ljudje, izvedenci na svojem področju, inteligentno rešujejo probleme (Feigenbaum 79, Gams 79). Ime »ekspertni sistemi« izhaja ravno iz zahteve, naj se računalniški program vede podobno kakor človek — strokovnjak (ekspert), ki zna na svojem specializiranem problemskem področju pametno sklepati, svetovati in na zahtevo tudi razložiti svoje odločitve oz. rešitve problema.

Ekspertni sistemi torej modelirajo tiste elemente človekovega reševanja problemov, za katere navadno menimo, da sestavljajo inteligenco: sklepanje, presojo, odločitve ob nezanesljivih in nepopolnih informacijah ter zmožnost pojasnjevanja svojega razmišljanja. Prav zadnja lastnost, ki poleg golega reševanja problema omogoča interaktivno posvetovanje o problemu med sistemom in uporabnikom, je izredno pomembna novost, ki jo prinaša metodologija ekspertnih sistemov.

Metodologija ekspertnih sistemov je razširila spekter uporabe računalnikov tudi na področja, ki (še) niso dosegla zadostne stopnje formalizacije, katera bi omogočala natančne algoritmične rešitve. Na teh področjih (kot je npr. medicinska diagnostika) strokovnjaki rešujejo probleme v veliki meri z intuicijo in po izkušnjah, kar običajno pomeni tudi uporabo obsežnega *znanja*. Znanje, ki ga na takih področjih uporablja človek strokovnjak, se pogosto imenuje »mehko«, saj ni dobro formalizirano, sistematizirano in dopušča veliko možnosti za subjektivno presojanje. Taka je npr. zdravnikova izkušnja: če bolnik kaže neke simptome, potem bi morda utegnila biti koristna informacija o boleznih v njegovi družini.

Moč ekspertnih sistemov zato temelji na kompleksnih *bazah znanja*, specifičnih za konkretna problemska področja. Pristop z metodami ekspertnih sistemov je kvalitativno nov tudi v tem smislu, da se problemi praviloma rešujejo s simbolnim sklepanjem, ne pa z vnaprej določenim zaporedjem numeričnih računov.



Slika 2:
Shematična slika okvar v srcu ter ustreznega elektrokardiograma, prikazana na grafičnem terminalu Chromatics. Na podlagi kvalitativnega modela, ki simulira električno delovanje srca, sistem KARDIO za dano diagnozo (motnjo srčnega ritma) na ekranu prikaže vzrok te motnje (shematično sliko nepravilnosti delovanja nekega dela srca) ter nariše ustrezni elektrokardiogram.

Primeri ekspertnih sistemov

Katera so področja uporabe metodologije ekspertnih sistemov in kakšno znanje mora sistem vsebovati, da bo na teh področjih zmožen reševati probleme tako kakor najboljši strokovnjaki? Na ti vprašanji si bomo odgovorili s površnim opisom nekaterih ekspertnih sistemov.

Ekspertni sistem DENDRAL (Buchanan 78) v interaktivni komunikaciji z uporabnikom (kemikom) ugotovi kemijsko strukturno formulo neznane snovi. Za to uporablja svoje znanje o valencah, o stabilnih in nestabilnih konfiguracijah atomov, pravila o razpadu snovi v masnem spektrometru, o magnetni resonanci, pravila za planiranje in ocenitev hipotez o neznani snovi ter dodatno uporabnikovo znanje o neznani snovi. DENDRAL je na ravni najboljših strokovnjakov za snovi, za katere ima vloženo še specialno znanje. Sistem je v ZDA dostopen preko računalniške mreže TYMNET.

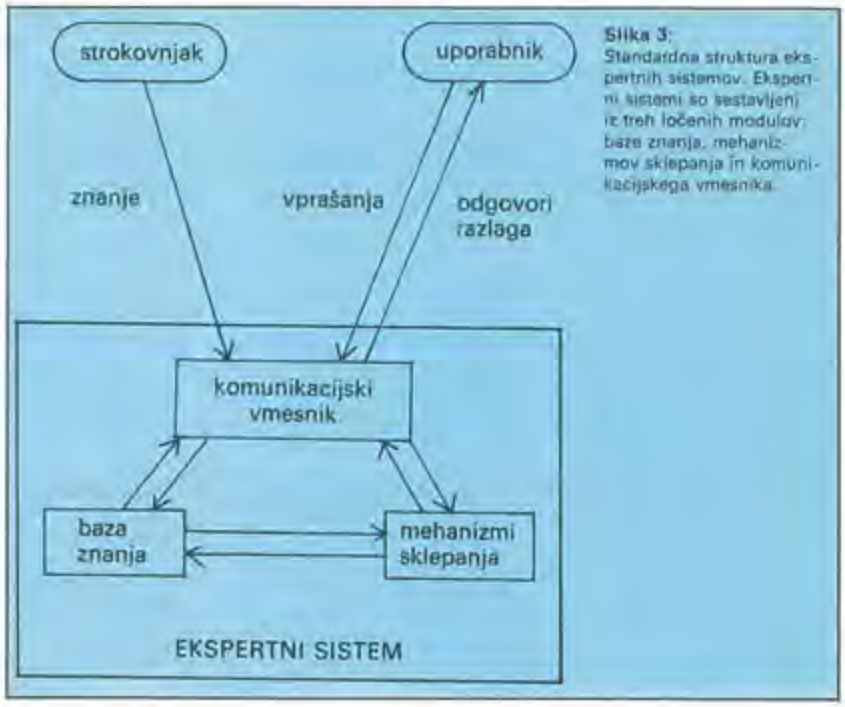
PROSPECTOR (Duda 80) je ekspertni sistem za pomoč pri odkrivanju nahajališč rud in nafte. Geološko znanje je v sistemu predstavljeno z logično-verjetnostnimi pravili, ki so povezana v semantično mrežo. V tipični uporabi da PROSPECTOR kot rezultat geografsko karto, na kateri so označene verjetnosti nahajališč rude. Pred nedavnim je sistem pomagal odkriti nahajališče molibdena, vredno 100 milijonov dolarjev.

Ekspertni sistem MYCIN (Shortliffe 76) svetuje zdravniku pri diagnostiki nalezljivih bolezni in izboru ustreznega zdravljenja (doziranju zdravil). Znanje o infekcijskih boleznih sestoji iz okrog 500 pravil oblike »če potem«, npr.:

če barvanje organizma je po Gramu pozitivno, in morfologija organizma je kok, in oblika rasti organizma je veriga
potem identiteta organizma je streptokok (s faktorjem zaupanja 0,7).

V svetu je bilo doslej razvitih že na stotine bolj ali manj uspešnih ekspertnih sistemov. Naštejmo jih le nekaj. INTERNIST je izredno obširen diagnostični sistem za področje interne medicine. SU/X je sistem za razpoznavanje objektov in njihovega položaja v prostoru na podlagi zveznih signalov z merilnih naprav. MOLGEN pomaga genetikom z nasveti, kateri poskusi z DNA so najbolj obetavni. RAFFLES vodi vzdrževalca računalniške opreme pri iskanju aparaturnih in programskih napak v sistemu. AM skuša odkriti nove zanimive matematične koncepte. MACSYMA rešuje nekatere matematične probleme (reševanje diferencialnih enačb, simbolično integriranje itd.). Zanimivi so tudi sistemi AL/X, AGE, EMYCIN, EXPERT in podobni, ki so neodvisni od problemskega področja in rabijo kot orodje za izdelavo ekspertnih sistemov za konkretne aplikacijske domene. Pregledna tabela nekaterih najbolj znanih sistemov je podana v (Bramer 82).

V našem okolju je doslej do najboljših rezultatov pripeljal razvoj ekspertnega konzultacijskega sistema KARDIO za diagnostiko in zdravljenje motenj srčnega ritma. Sistem razvijajo sodelavci Laboratorija za umetno inteligenco Odseka za računalništvo in informatiko na Institutu Jožef Stefan skupaj z zdravniki Centra za intenzivno interno medicino Univerzitetnega kliničnega centra v Ljubljani.



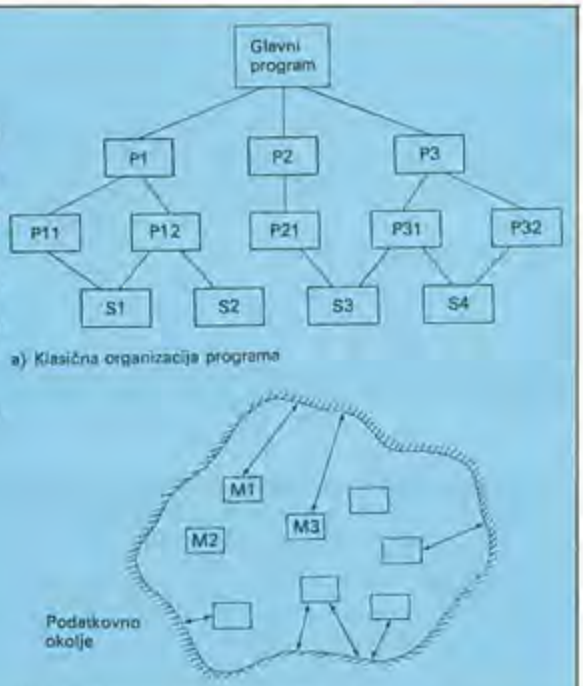
Slika 3: Standardna struktura ekspertnih sistemov. Ekspertni sistemi so sestavljeni iz treh ločenih modulov: baze znanja, mehanizmov sklepanja in komunikacijskega vmesnika.

Najpomembnejši del tega sistema je sistem za elektrokardiografsko diagnostiko motenj srčnega ritma na podlagi podatkov, dobljenih z analizo bolnikovega elektrokardiograma. Sistem vsebuje znanje o elektrokardiografskih značilnostih vseh mogočih kombinacij motenj srčnega ritma (sistem pozna 2400 kombinacij motenj ritma, ki jim ustreza okrog 141000 kvalitativnih elektrokardiografskih opisov). To obsežno znanje smo avtomatsko generirali z uporabo kvalitativnega modela srca, ki simulira električno delovanje srca (Mozetič 83, Mozetič 84). To znanje zadošča za pravilno diagnosticiranje 95% bolnikov z motnjami srčnega ritma. Slika 2 prikazuje srce, na katerem so shematično predstavljene okvare, ki so pripeljale do motnje ritma, ter ustrezni elektrokardiogram.

Struktura ekspertnih sistemov

Praviloma so ekspertni sistemi sestavljeni iz treh modulov: baze znanja, mehanizmov sklepanja in komunikacijskega vmesnika. *Baza znanja* vsebuje dejstva, relacije med dejstvi in metode za reševanje problemov v izbrani aplikacijski domeni. *Mehanizmi sklepanja* implementirajo algoritme za reševanje problemov, tako da kratko malo pokličejo neko dejstvo iz baze ali pa da izpeljejo nova dejstva iz dejstev, ki so eksplicitno shranjena v bazi znanja. Za izpeljevanje novih dejstev se lahko uporabljajo tudi splošne metode, ki so shranjene v bazi znanja. *Komunikacijski vmesnik* pa omogoča uporabniku udobno interakcijo s sistemom. Strukturo ekspertnih sistemov prikazuje slika 3.

Pri načrtovanju ekspertnih sistemov se je pokazala nujnost spremembe pri organizaciji programov. V konvencionalni organizaciji programskih sistemov programski moduli kličejo drug drugega po strogo določeni shemi. S tem je v veliki meri vnaprej določeno zaporedje operacij, ki se morajo opraviti na vnaprej



Slika 4: Primerjava klasične (a) in vzorčno vodene (b) organizacije programov. V klasični organizaciji programskih sistemov se moduli kličejo po stalno določeni shemi. Vsak podprogram se s klicem naslednjega podprograma odloča za vrstni red izvajanja programskih modulov. Izvajanje poteka pretežno deterministično. V vzorčno vodeni organizaciji sistemov pa so moduli definirani z vzorcem, ki določa pogoj, pod katerim se modul aktivira, in z akcijo, ki naj se izvrši, če se podatki ujemajo z vzorcem v pogoju.

predvidenih podatkih. Ekspertni sistemi pa imajo praviloma drugačno organizacijo programa. Posamezni moduli ne kličejo direktno drugih, temveč izvajanje modula sproži »vzorec« v podatkovnem okolju (bazi) sistema. Sistem torej pregleduje podatke, da bi ugotovil, kateri vzorci iz modulov ustrezajo danemu vzorcu v podatkih. Ko sistem najde ustrezni modul, se opravi akcija oziroma zaporedje akcij tega modula. Tako organizirani programski sistemi se imenujejo *vzorčno vodeni sistemi* (Watermann 78). Primerjavo med klasično in vzorčno vodeno organizacijo programa ilustrira slika 4.

Klasična organizacija programov ima značilnosti, ki v splošnem veljajo za dobre: je urejena, hierarhična in omogoča dobro kontrolo in preglednost. Organizacija vzorčno vodenih sistemov pa je povsem drugačna. Med moduli ni hierarhije, prav tako ni razvidno, kateri moduli kličejo druge. Moduli komunicirajo s podatkovnim okoljem, ne z drugimi moduli. Struktura sama v principu omogoča istočasno izvajanje več modulov.

Ta organizacija je s stališča načel strukturiranega programiranja povsem nesprejemljiva, vendar ima tudi izrazite prednosti. Ni nujno, da so vse povezave med moduli vnaprej skrbno predvidene in določene. Posledica te lastnosti je, da je mogoče načrtati in implementirati vsak modul relativno neodvisno od drugih, to pa omogoča visoko stopnjo modularnosti. Pogosto npr. odstranitev enega modula iz sistema, priključitev drugega ali spreminjanje nekega modula ne povzroča nobenih nevšečnosti. Spremembe nastanejo le v načinu reševanja problema, ni pa treba spreminjati programa ali vsaj klicev podprogramov kakor v klasični organizaciji.

Tako visoka stopnja modularnosti je zaželena predvsem v sistemih s kompleksnimi bazami znanja, saj je težko predvideti vse interakcije med posameznimi deli znanja v bazi. Vzorčno vodena arhitektura ponuja naravno rešitev tega problema: znanje je lahko predstavljeno s posameznimi »kosi znanja«, imenovanimi *produkcijska pravila* (glej podpoglavje o predstavitvi znanja v ekspertnih sistemih). Ena od mogočih interpretacij produkcijskega pravila oblike »če—potem« je naslednja:

če situacija S potem akcija A

Slika 5 ilustrira, kako lahko pravila take oblike štejemo za vzorčno vodene module, ki se sprožijo na podlagi vzorcev v podatkovnem okolju.

Za zgled si oglejmo, kako lahko vzorčno vodeni programi rešujejo tudi čisto »navadne« programerske naloge. Implementirali bomo Euklidov algoritem za iskanje največjega skupnega delitelja dveh naravnih števil A in B. V proceduralnem jeziku, kakor je npr. pascal, je program takle:

```
begin
while A =/= B do
if A > B then A := A-B
else B := B-A
write (A)
end
```

Ekvivalentni vzorčno vodeni program pa sestoji iz dveh pravil:

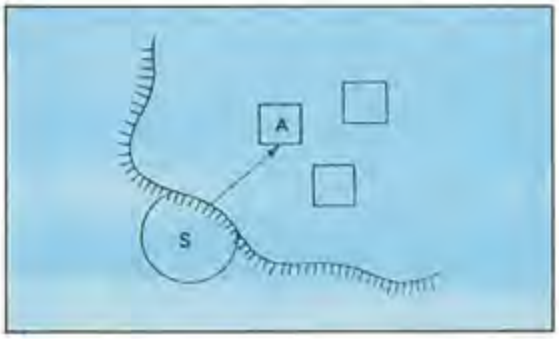
Pravilo 1:

```
[m(C1, A), m(C2, B), A > B] - - - >
[replace m(C1, A), m(C1, A-B)]
```

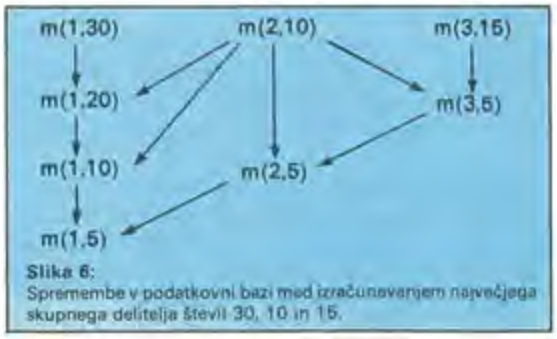
Pravilo 2:

```
[m(C, X)] - - - > [write(X), stop]
```

Imena A, B, C1, C2 in X je treba razumeti kot univerzalno kvantificirane spremenljivke. Pravilo 1 se lahko prebere kot:



Slika 5: Pravilo oblike »če situacija S potem akcija A« ustreza enemu modulu v organizaciji vzorčno vodenih programov.



Slika 6: Spremembe v podatkovni bazi med izračunavanjem največjega skupnega delitelja števil 30, 10 in 15.

če obstaja spominska celica C1 z vsebino A
in
obstaja spominska celica z vsebino B
in
A > B
potem zamenjaj vsebino C1 z A – B.

V začetku vsebuje podatkovna baza števili, za kateri hočemo izračunati največji skupni delitelj, če sta to števili 25 in 15, bi baza vsebovala podatka:

m(1,25), m(2,15)

Končno stanje podatkovne baze pa bi podajalo največji skupni delitelj obeh števil v naslednji obliki:

m(1,5), m(2,5)

Pri primerjavi s pascalovim programom vidimo (v tem primeru zelo majhno) prednost vzorčno vodenega programa v tem, da ni potrebna posebna akcija za vsakega od dveh primerov: A > B in B > A. Vzorec v pogoju pravila 1 namreč vedno poimenuje spominski celici C1 in C2 tako, da je A > B.

Z istim programom lahko poiščemo tudi največji skupni delitelj treh števil. Če vsebuje podatkovna baza v začetku tri števila, npr. 30, 10 in 15, tj. podatke:

m(1,30), m(2,10), m(3,15)

je izhod iz programa največji skupni delitelj vseh treh števil. Sledenje spremembam v podatkovni bazi prikazuje slika 6. Ta splošnost vzorčno vodenih programov izhaja iz vzorca v pogoju pravila 1, ki zahteva le obstoj dveh neenakih števil v podatkovni bazi, neodvisno od drugih podatkov. Razširitev pascalovega programa na iskanje največjega skupnega delitelja treh števil bi zahtevala večje spremembe.

Predstavitev znanja v ekspertnih sistemih

V ekspertnih sistemih je znanih več različnih shem za predstavitev znanja, tako semantične mreže, okviri in predikatni račun. Najpogosteje pa je znanje predstavljeno v obliki *pravil* (Bratko 82). Po navadi eno pravilo ustreza enemu modulu v shemi vzorčno vodenih sistemov. Pravila so običajno oblike

če – potem

imajo pa lahko različne interpretacije. Primeri takih interpretacij so:

če pogoj P *potem* sklep S.
če situacija S *potem* akcija A.
če veljata pogoja P1 in P2 *potem* pogoj P ne velja.

Pogosto se pravila zapišejo v naslednji notaciji:

S = = > A

Pomen takega zapisa je seveda odvisen od interpretacije pravila, npr. če položaj S, potem akcija A. Položaj določa pogoje (definira vzorec), pod katerimi je treba izvesti akcijo. Akcija pa je navadno neka trditev ali procedura, ki spremeni podatkovno bazo ali pa npr. usmeri kontrolne mehanizme sistema v aktiviranje določenega dela znanja. Navedene interpretacije podajajo *logične zveze* med elementi problemskega prostora.

Mogoče pa so tudi naslednje interpretacije pravil oblike »če—potem«:

če pogoj P *potem* sklep S s faktorjem zaupanja G.
če pogoja P1 in P2 *potem* je pogoj R navadno izključen.

Pravila take oblike ne podajajo le logičnih zvez med elementi problemskega prostora, temveč tudi *verjetnostne zveze*, zato jih imenujemo *logično-verjetnostna pravila*. Tako pravila omogočajo tudi obdelavo nezanesljivih oz. »mehkih« informacij.

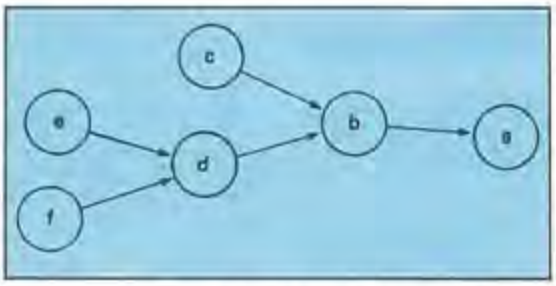
Ekspertne sisteme, ki imajo znanje predstavljeno v obliki pravil, imenujemo *produkcijski sistemi*. Posamezno pravilo vsebuje sklenjen del znanja o problemskem področju. Delovanje sistema si lahko predstavljamo kot zaporedje aktiviranja temeljnih delov (»kosov«) znanja. Zasledovanje aktiviranja pravil (vezava kosov znanja, zaporedje sklepov) nam daje dober pregled nad delovanjem sistema. Tako zgrajen sistem je torej prozoren in omogoča naravno razlago svojih odgovorov.

Ob zgledu si oglejmo, kako predstavitev znanja v obliki produkcijskih pravil omogoča naraven način odgovarjanja na uporabnikova vprašanja »Kako?« in »Zakaj?«.

Imejmo majhno bazo znanja, ki sestoji iz naslednjih pravil:

b — — — > a (kar pomeni če b, potem a)
c — — — > b
d — — — > b
e — — — > d
f — — — > d

Ta sistem pravil si lahko predstavimo s sliko 7.



Slika 7: Majhna baza znanja.

Če je uporabnik vprašal: »Kako veš, da je a res?«, lahko sistem generira odgovor tako, da sledi puščicam na sliki 7 v smeri nazaj. Od a se sistem premakne na b in odgovori:

Ker vem, da:
1. b je res.
2. a sledi iz b.

Po istem principu sistem upraviči tudi resničnost dejstva b na podlagi resničnosti dejstev c in d ter pravil, ki povezujejo c in d z dejstvom b.

Na vprašanje »Zakaj te zanima c?« pa sistem odgovori s sledenjem puščicam v smeri naprej. Sistem lahko odgovori:

Ker me zanima, ali je a res, in ker vem, da:
1. a sledi iz b.
2. b sledi iz c.

V navedenem primeru so nastopala »trda« logična pravila. Sistem pa podobno generira razlago svojega delovanja tudi v primeru, ko so v bazi znanja »mehka« logično verjetnostna pravila.

Programski jezik prolog kot orodje za načrtovanje ekspertnih sistemov

Poleg navedenih tehnik za načrtovanje ekspertnih sistemov (vzorčno vodeni sistemi in produkcijski sistemi) želimo omeniti tudi pomembno področje umetne inteligence, imenovano *logično programiranje*. Načela logičnega programiranja in konkretno uresničenje teh načel v programskem jeziku prolog (PROgramming in LOGic) omogočajo reševanje problemov s simbolnim sklepanjem, kakršno zahteva narava problemov ekspertnih sistemov.

Prolog je preprost, vendar presenetljivo močan programski jezik. Njegova matematična podlaga je matematična logika, v podobnem smislu, kakor so aritmetični izrazi matematična podlaga fortrana in njemu sorodnih jezikov. Podlaga prologa je bila razvita l. 1975 na univerzi v Marseillu kot praktična realizacija ideje, da je mogoče matematično logiko uporabiti tudi kot programski jezik (Kowalski 79).

Prolog implementira šibkejši, a izrazno dovolj naraven in uporaben podjezik predikatnega računa prvega reda, ki je omejen na t.i. Hornove stavke. Hornovi stavki izražajo pogojne trditve (implikacije) tipa

če P1 in P2 in ... in Pn **potem** P

kar se v sintaksi edinburške implementacije prologa za DEC-10 (Pereira 78) zapiše kot

P :— P1, P2, ..., Pn.

Predikatni račun prvega reda, katerega podjezik daje sintakso prologu, omogoča preprosto definiranje *relacij* med podatki in strukturiranje podatkov. Zato je prolog posebno močan pri simboličnem, nenumeričnem procesiranju in pri delu z bogatimi podatkovnimi strukturami.

Prologove programe lahko štejemo za vzorčno vodene sisteme (Bratko 82), saj že posamezni stavki oblike

a :— b. (a)

označujejo relacijo, da »b implicira a«, ali pa da »a je res, če je b res«, to ustreza posameznemu pravilu vzorčno vodenega sistema. Prologov program pa je množica takih stavkov in stavkov oblike

b. (b)

ki označujejo dejstvo, da »b je res«. Pravilo (a) se aktivira, če ga sproži vzorec (b) v podatkovni bazi sistema. Na vprašanje »ali je a res?«, ki se v prologovi sintaksi zapiše kot

?— a,

prolog odgovori z »yes«, saj sistem logično izpelje a iz dejstva b. Na prologov interpreter, ki sam poišče zaporedje operacij, ki pripeljejo do odgovora na zastavljeno vprašanje, lahko gledamo kot na implementacijo mehanizmov sklepanja vzorčno vodenega sistema.

Bazo znanja na sliki 7 zlahka prepišemo v prolog:

```
a :— b.
b :— c.
b :— d.
a :— e.
d :— f.
```

Recimo, da velja e:

```
e.
```

Zdaj vprašamo:

```
?— a.
```

in prolog izpelje odgovor »yes«.

Znanih je več implementacij prologa. Za bralce te knjige je zlasti zanimiva implementacija prologovega interpreterja za mikroprocesor Z-80 in s tem tudi za osebni računalnik Sinclair ZX Spectrum. Interpreter so razvili na Imperial College, London (Clark 83).

Literatura

Opomba: Reference, označene z zvezdico (*), so delno ali v celoti natisnjene v brošuri »Sistemi in tehnike umetne inteligence«, urednik Ivan Bratko, izdala in založila Fakulteta za elektrotehniko v Ljubljani, 1983.

Bramer M. A.: A Survey and Critical Rewiew of Expert Systems Research. V Expert Systems (EDITOR MICHIE D.), Gordon & Breach 1982. (*)

Bratko I.: Expert systems and Prolog. State of the art report: Supercomputer systems technology (editor Sumner F.), Series 10, No. 6, str. 187—211, Pergamon Infotech Ltd., 1982. (*)

Buchanan B. G., Feigenbaum E. A.: DENDRAL and Meta-DENDRAL, Artificial Intelligence, Vol. 11, No. 1—2, 1978.

Clark K. L., McCabe F. G., Ennals J. R.: Sinclair ZX Spectrum micro-PROLOG primer, Sinclair Research Ltd., 1983.

Duda R., Gaschnig J., Hart P.: Model design in the Prospector consultation system for mineral exploration. V Expert systems in the microelectronic age (editor Michie D.), str. 153—167, Edinburgh University Press, 1979.

Feigenbaum E. A.: The art of artificial intelligence: Themes and case studies of knowledge engineering. V Expert systems in the microelectronic age (editor Michie D.), str. 3—25, Edinburgh University Press, 1979.

Gams M., Lavrač N., Bratko I.: Osnovni koncepti in struktura ekspertnih sistemov. Informatica 4, str. 11—16, 1979.

Kopec D.: Problems of the human window. Research report, University of Edinburgh: Machine Intelligence Research Unit, 1980.

Kowalski R.: Logic for problem solving. North Holland Publ. Co., 1979.

Mozetič I., Bratko I., Lavrač N.: An experiment in automatic synthesis of expert knowledge through qualitative modelling. Logic programming workshop '83, Albufeira, Portugal, 1983. (*)

Mozetič I., Bratko I., Lavrač N.: The derivation of medical knowledge from a qualitative model of the heart. ISSEK Workshop '84.

Pereira F., Pereira L. M., Warren D. H. D.: User's guide to DEC-system 10 Prolog. Dept. of artificial intelligence, University of Edinburgh, 1978.

Shortliffe E. A.: Computer-based medical consultation: MYCIN. Elsevier Publ. Co., 1976.

Voysey H.: New Scientist, avg. 1977.

Watermann D. A., Hayes-Roth F.: An overview of pattern — directed inference systems. V Pattern — directed inference systems (editors Watermann D. A., Hayes-Roth F.), str. 577—601, Academic Press, 1978.

6. poglavje

Umetnost programiranja

Če vas mika kaj več kot zgolj uporaba kupljenih programov, potem se morate naučiti, kako boste v računalniškem programu uporabljali ukaze in tako obvladali pretok podatkov.

Diagram poteka

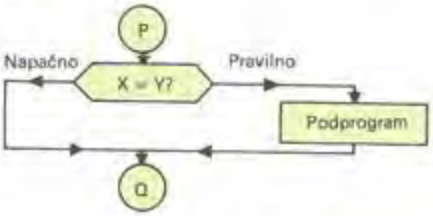
Natančna uporaba diagrama poteka vodi k učinkovitemu in dobro pripravljenemu programu

Tek podatkov
Pravi namen diagrama poteka je preprosto in zgoščeno pokazati tek podatkov skozi računalniški program.
Zelo pomembne so »preskusne« točke, kjer je izvajanje programa ponazorjeno drugače, kakor se dejansko izvaja.
Preprosto grafično ponazoritev razumemo veliko laže od podobne ugotovitve, ki je zapisana — slika je resnično lahko vredna tisoč besed!
»Preskusni« simbol je lahko predstavljen s ploščatim šesterokotnikom, kakršen je prikazan tukaj, ali pa s podolžnim rombom.

Problem lahko predstavimo slikovno preprosto z risanjem diagramov poteka, ki pokažejo korake programa in povezave med njimi. Koristen je za razumevanje problema in nam lajša pisanje programov.

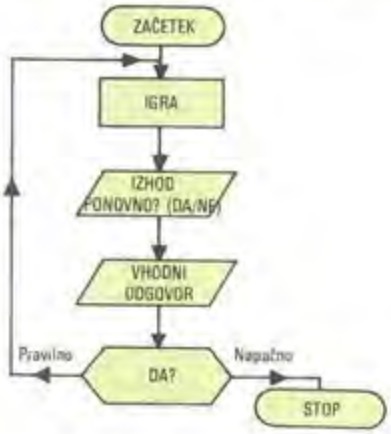
Vsak četverokotni simbol poteka prikazuje proces ali dejanje, linije, ki povezujejo te simbole, pa predstavljajo mogoče poti skoznje. Poti so enosmerne, puščice pa uporabljamo za označbo smeri, običajno puščice kažejo navzdol ali pa od leve proti desni.

Kadar se je treba odločiti, uporabimo šesterokotnik ali znak v obliki romba, ki pomeni funkcijo odločanja. Program vstopa po eni poti, ven pa lahko gre v eno od dveh smeri, odvisno od rezultata preskusa. Če naj preskus določi, ali naj se posamičen proces izvede ali ne, imamo pri odločitvi samo eno izhodno pot. Tukaj je zgled preskusa, ki odloča o tem ali se podprogram izvrši ali ne.




```
120 IF X = Y THEN GOSUB 300
```

Znak za funkcijo odločanja lahko uporabimo za označbo preskusa, ki se konča z zanko. V spodnjem zgledu je prikazano, da se program vrne na začetek, če dobimo na vprašanje »PONOVNO?« pritrdilen odgovor.




```
 90 REM "ZAČETEK IGRE"
100
800
810 PRINT "PONOVNO ? (DA/NE)"
820 INPUT R$
830 IG R$ = "Y" THEN GOTO 100
840 END
```

Lahko se želimo odločiti med dvema popolnoma ločenima potekoma. V spodnjem zgledu primerjamo igralčev rezultat z najvišjim prejšnjim.

```
1200 IF TA ZADETEK > NAJVEČJI REZULTAT
     THEN GOTO 1230
1210 PRINT "SMOLA. ŠE POSKUSI!"
1220 GOTO 1250
1230 LET NAJVEČJI REZULTAT = TA REZULTAT
1240 PRINT "ČESTITAMO. PONOVNO VELIK
     REZULTAT!"
1250 PRINT VELIK REZULTAT
```

Vidimo, da je vrednost višjega rezultata prikazana v obeh primerih in da se oba mogoča poteka združita in postaneta enojen vhod v operacijo na izpis.

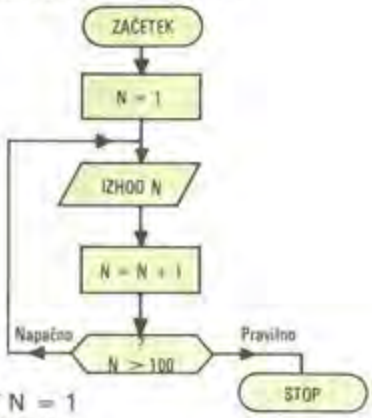
Vse odločitve so sprejete kot izid preskusov podobnih temu, ki dajejo pozitiven ali negativen, pravilen ali napačen rezultat. Kakor vidimo, nam ta popolnoma dvojiški proces odločanja ne dopušča odgovora »morda«. Lahko uporabimo katerikoli izraz, vedno pa moramo primerno označiti oba mogoča poteka.

Vsi programski jeziki imajo možnost pogojnih stavkov. Če je odločitev »PRAVILNO«, gre izvrševanje programa v pogojno zanko, če pa je »NAPAČNO«, se program izvršuje naprej. V primeru programskega jezika basic, ki omogoča samo preprost pogojni stavek IF–THEN (ČE–POTEM), uporabimo pogojno zanko s stavkom GOTO. V prejšnjem primeru smo s stavkom GOTO skočili v stavek 1200. Stavek 1210 se bo izvršil, če bo rezultat preizkušanja v vrstici 1200 »NAPAČNO«.

Poglejmo si še drugačno uporabo ukaza GOTO v vrstici 1220. Uporabimo ga na koncu preskusa za rešitev problema pogojne znake. S to metodo ponovno združimo oba mogoča poteka, kakor na primer v vrstici 1250.

Uporaba diagrama poteka po navadi pospešuje uporabo stavka GOTO, ki sledi grafični ponazoritvi programa od točke do točke. V splošnem pa je uporaba brezpogojnih stavkov na preskakovanje precej nevarna. Če moramo v basicu uporabiti tak stavek, potem je diagram poteka odlična metoda za oceno poti, po kateri se kontrola prenese na drugi del programa.

Uporabimo še zadnji zgled za preskus, kako nam uporaba diagrama poteka omogoča, da natanko ponazorimo potrebne korake za izvršitev preproste naloge: tiskanje vseh števil od ene do sto.




```
10 LET N = 1
20 PRINT N
30 LET N = N + 1
40 IF N > 100 THEN END
50 GOTO 20
```

Diagram poteka nas tako spodbuja k pisanju programov korak za korakom, to pa pri velikih programih pogosto vodi k precej neelegantnemu izidu. Zato si poglejmo uporabo zanke FOR-NEXT v basicu ob naslednjem zgledu:

```
10 FOR N = 1 TO 100
20 PRINT N
30 NEXT N
40 END
```

Diagram poteka ne more prikazati te »stenografije« v basicu, in če bi se ga natančno držali, bi nas to pripeljalo k manj učinkoviti poti za reševanje problema. Daje pa nam nekaj informacij o sestavi zanke FOR-NEXT in je zato koristen, ko raziskujemo to in druge funkcije basica, da bi ugotovili njihovo konstrukcijo.

Diagrami poteka so še posebej koristni pri pripravljanju ali konceptiranju programa, še posebno pri težavnih delih. Izkušeni programerji jih uporabljajo redkeje kakor začetniki in se k diagramom poteka velikokrat zatečejo za analizo in dokumentacijo delov programa, ki jih ne poznajo.

Naj bo diagram poteka narisan na papirju ali pa obstaja le v programerjevi podzavesti, risanje poteka informacij in kontrol je pomemben pripomoček računalnikov za reševanje problemov.

Urejanje podatkov

Računalnik lahko izbira podatke in nabira sezname iz velikega števila informacij, shranjenih v podatkovni bazi

Zbirko podatkov, ki so dosegljivi računalniku, imenujemo podatkovno bazo. V svojem vsakdanjem življenju vsi uporabljamo razne neračunalniške podatkovne baze.

Za zgled take neračunalniške baze lahko vzamemo telefonski imenik. Vendar pa ni nujno, da so podatki v podatkovni bazi urejeni na poseben način. Pri računalnikih urejanje določajo nekatere omejitve. Program za delo s podatkovno bazo se sestoji iz podprogramov, ki omogočajo izbiranje med podatki. Takšni programi segajo od preprostega oštevilčevanja podatkov do popolnih jezikov.

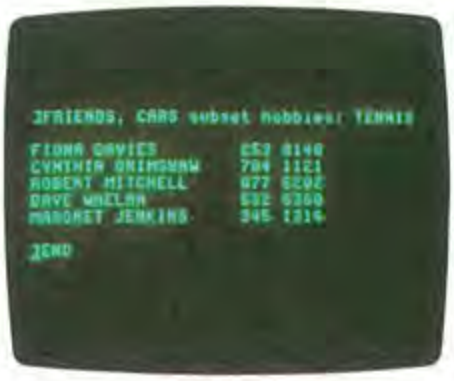
Tipična računalniška podatkovna baza je velika in vsebuje razne vrste podatkov. To pa ne pomeni, da bi morali imeti orjaški računalnik. Vsak računalnik lahko dela z uporabno podatkovno bazo. Edina omejitev je velikost in hitrost pomnilniških zmogljivosti.

Za zgled lahko izdelamo seznam, ki ga imenujemo »ljudje« in bo vseboval podatke o različnih osebah. Podatke v seznamu razvrstimo v več skupin. Večina od teh so preproste besede ali pa števila. Za dve vrsti podatkov imamo omejene možnosti: spol je lahko moški ali ženski, stan pa je lahko: poročen, samski, ločen ali ovdovel.

Včasih je koristno, če nekatere podatke združimo v vrste besed in številk. Za zgled vzemimo: poklic, ime podjetja, kjer je nekdo zaposlen, naslov v službi, telefon v službi in ime predstojnika. Te podatke lahko razvrstimo v rubriko »služba«. Avtomobil, njegovo barvo, starost itd. pa lahko vpišemo v rubriko »avtomobil«.

Če razširimo to zamisel, lahko razvrstimo vse naslove v obliko seznama. To je bolje, kakor če imamo naslov kot posebno enoto, saj nas lahko zanima samo kraj bivališča neke osebe, ne pa tudi ime ulice.

Kdo bi igral tenis?
Uporabili smo Rubikovo kocko za ponazoritev vrste podatkovnih baz. Na njej so vse informacije, ki jih potrebujemo, niso pa še urejene. V našem primeru iščemo partnerja za igranje tenisa (simbol je lopar), ki je lastnik avtomobila (avtomobil) in je prost določeni dan (rdeči kvadrati).



Postavi pravilno vprašanje!

Večina komercialnih baz uporablja kode, ki so že skoraj programski jeziki. V našem primeru smo uporabili tipične kode. »ACCESS« ali dostop nam označuje program, ki ga želimo povprašati po datoteki, ki smo jo prej sestavili. »Use« ali uporaba nam označuje, da želimo samo del podatkov datoteke pod imenom »FRIENDS« (prijatelji) in kadar je pogoj »while« (dokler) izpolnjen, dobimo izpis, ki se nanaša na »CAR« (avtomobil) ali TENNIS (tenis). Rezultat je seznam prijateljev z njihovimi telefonskimi številkami, ki so izbrani po načelu, da igrajo tenis in vozijo avto.

Organizacija podatkov
Hierarhična organizacija podatkovne baze vodi uporabnika od ene informacije k drugi in mu omogoča sprotno izbiranje podatkov, ne da bi pred tem poznal zgradbo datoteke.

Podatke pod nazivom »stan« lahko razširimo tako, da dodamo še ime drugega zakonca. To ime je lahko samo dodatna beseda, bolje pa je, če naredimo kar novo rubriko s podatki o zakoncu.

Vsak zapis v datoteki ima svoj prostor in je označen s številko. Za referenčne podatke o neki osebi lahko uporabljamo kar to številko namesto imena.

Tovrstne podatke, kjer so zapisi označeni s številkami, imenujemo »kazalčni« zapisi. Če želimo tako označiti neke predpostavljene osebe, dobimo strukturo, ki je podobna strukturi »zapis«.

Razlika med kartoteko in računalniško bazo podatkov je v tem, da imamo v kartoteki podatke urejene samo na en način — običajno po abecedi.

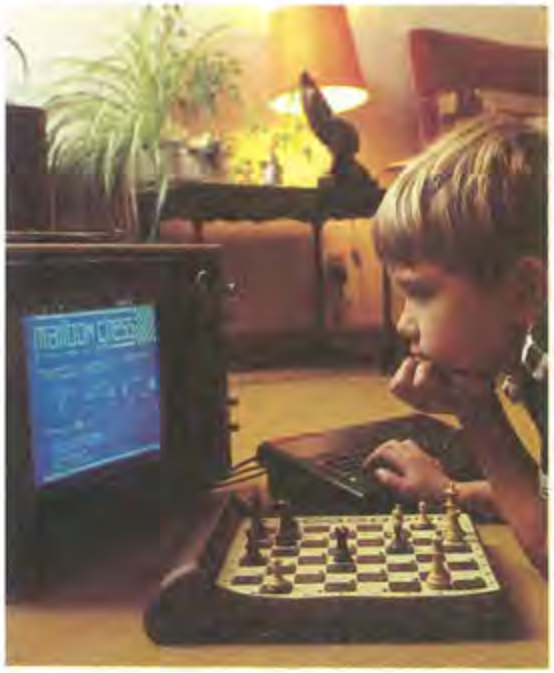
Ureditev podatkov v obliki kartoteke nam zadošča, če želimo na primer poiskati, v katerem podjetju je zaposlen kak človek. Kaj pa, če bomo želeli zvedeti imena vseh zaposlenih v podjetju?

Pri uporabi kartoteke bi morali pregledati vsako kartico posebej, da bi lahko poiskali želene podatke. Takšno delo je počasno, obenem pa je mogoče, da naredimo napako.

Pri uporabi računalniškega sistema pa lahko računalniku ukažemo, da pregleda vse podatke in izpiše imena oseb, ki so zaposlene v podjetju, ki nas zanima.

Lahko pa z računalnikom preuredimo datoteko tako, da bo področje z imenom podjetja najpomembnejši del. To bo povzročilo popolno preureditev podatkov, ki bo razvrstila vse podatke za določeno podjetje v eno skupino in iz tega dela datoteke bomo dobili imena vseh zaposlenih. V kartoteki imamo samo eno glavno rubriko, v kateri so shranjeni priimki, medtem ko je v računalniškem sistemu lahko vsaka rubrika glavna.

Z upoštevanjem tega načela lahko popolnoma preuredimo podatkovno bazo. Za zgled navedimo ureditev podatkov ljudi po avtomobilih, ki jih imajo, ali pa po krajih, kjer živijo. To je velika prednost računalniško urejene podatkovne baze.



Današnja enciklopedija
Z velikimi računalniki je mogoče uporabljati velikansko bazo podatkov, ki je dostopna posameznikom kar v njihovem stanovanju. Za prikaz podatkov je uporabljen domači televizor, medtem ko je posebna tipkovnica priključena na centralni računalnik z običajno telefonsko povezavo. Dostop do baze podatkov je razdeljen na več področij, ki jih uporabnik vidi na zaslonu. Če sporoči računalniku številko svoje kreditne kartice, lahko opravi nakup kar doma.

Razvrščanje informacij

Kadar načrtujemo programe, ki shranjujejo informacije ali manipuliramo z njimi, je pomembno, da pazimo na sestavo le-teh in jih ne podvajamo

Kratka pot
Neka tovarna ima svoje obrate v šestih mestih in jim vsak mesec pošilja surovine. Računalnik lahko izpiše optimalno pot te dostave.

Mesto (1)	Ajdovščina
Mesto (2)	Cerknica
Mesto (3)	Maribor
Mesto (4)	Novo mesto
Mesto (5)	Postojna
Mesto (6)	Tržič

Imena mest so shranjena po abecedi v sekvenčni datoteki. S traku se berejo po tem vrstnem redu in se razporejajo v tabelo Mesta ( ). Računalnik ugotovi najboljšo razvrstitev:

Novo mesto	Mesto (4)
Postojna	Mesto (5)
Cerknica	Mesto (2)
Ajdovščina	Mesto (1)
Tržič	Mesto (6)
Maribor	Mesto (3)

Za takšno shranitev mest bi porabil računalnik več prostora v pomnilniku, zato si zapomni samo položajne številke mest v tabeli Mesta ( ). Tako je:

Seznam številk je kazalo k tabeli Mesta ( ). Druga podobna kazala so lahko urejena tako, da vsako da različno razvrstitev mest. Prednost urejanja podatkov s kazalci je v tem, da prvotne baze podatkov ni treba preurejati ali podvajati, temveč je treba urediti in shraniti samo kazalce.

Vsak, ki je kdaj uporabljal sistem kazala kartic (npr. knjižnični katalog), ve, kako uporaben je. Če smo kdaj stresli škatlo kartic, smo videli, da postanejo neurejene kartice čisto neuporabne. Knjižnica vsebuje veliko število informacij. Vendar je informacijski sistem skoraj brez vrednosti, če informacije niso razvrščene po namenih.

Bistvo informacijskega sistema niso informacije same, temveč njihova organizacija. Poglejmo si zgled informacije:

Super market Titova 10, Ljubljana

Sama zase je ta informacija omejena. Če vemo, da gre za oddelek s tekstilom, dobimo nov pomen, ki je določen s strukturo podatka.

Najpreprostejša podatkovna sestava ima obliko datoteke, ki nam je zbirka podatkov s skupnimi značilnostmi. Ime datoteke pove nekaj o sami informaciji in združitev podatkov pod istim imenom omogoča enostavnejše delo z datotekami. Za lažjo uporabo obravnavamo datoteke kot večjo enoto informacij, ali pa kot posebno skupino manjših enot. Knjiga je kot datoteka. Vzemimo za zgled avtorico G. Beeton. Njen življenjepis običajno beremo kot celoto, njeno kuharsko knjigo pa kot zbirko posameznih receptov.

Če je datoteka obsežna, moramo preveriti vsak njen podatek, dokler ne najdemo pravega. To je zaporedno iskanje ali serijski dostop do informacij, zato tako urejeno datoteko imenujemo zaporedno. Tudi televizijski spored je zaporedna datoteka informacij.

Zaporedne datoteke se pogosto uporabljajo zato, ker so koristne in uporabne in na številne načine odsevajo človekov način mišljenja. Postanejo pa nepripravne in počasne za uporabo, zato so interno razdeljene na poddatoteke, ki jih lahko najdemo direktno brez preiskovanja cele datoteke. Knjige so bile nekoč zaporedno datotečno urejene, vendar jih je vpeljava poglavij, oštevilčevanje strani in kazalo preuredila. Poglavja so poddatoteka knjige in strani so poddatoteka obojega: poglavja in knjige.

Datoteko, ki ne potrebuje zaporednega iskanja, imenujemo direktno dostopno datoteko. Album pesmi na magnetnem traku je zaporedna datoteka, medtem ko je isti album na veliki plošči direktno dostopna datoteka: poiskati pesem na traku zahteva, da začnemo na začetku in previjamo, medtem ko imamo do kateregakoli odlomka na plošči direkten dostop s premikom ročice.

Pri direktnem dostopu moramo vedeti, kje so podatki: v knjigi pove kazalo, kje je kaj. Natančno določiti, kje so podatki, pomeni delo. Opremiti knjigo z imenskim ali stvarnim kazalom je posebno delo za avtorja ali založnika in informacije v njej morda niso vredne tega truda in izdatka; romani npr. nimajo takega kazala, medtem ko ga učbeniki ponavadi imajo.

Računalnike uporabljamo za hiter dostop do množice informacij za različne podatkovne strukture. Podatki, ki so shranjeni na magnetnem traku ali disketi v določenem zaporedju — po navadi v sekvenčnih datotekah — so v računalnikovem pomnilniku shranjeni v precej drugačni obliki.

Vzemimo, da ima pekarna naslove vseh svojih prodajaln v zaporedni datoteki na traku in želi, da računalnik izpiše načrt dostave kruha tem prodajalnam. Seznam na traku bo takšenle:

ABC Pomurka, Miklošičeva
Delikatesa, Titova
Emona, Celovška
Maximarket, Trg revolucije
Merkator, Slovenčeva
Nama, Tomšičeva

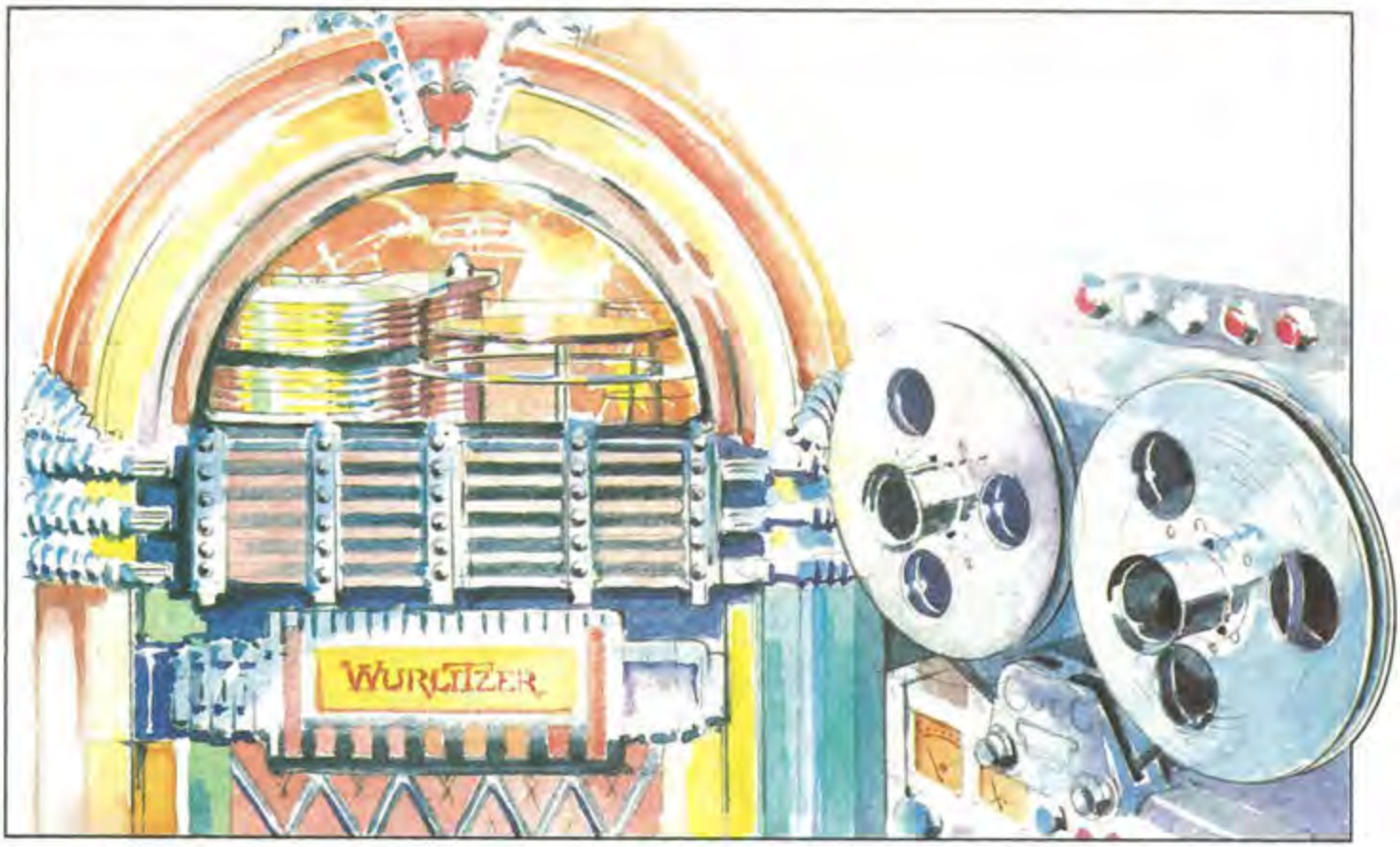
Ko je seznam s traku shranjen v glavnem pomnilniku, bo vsako ime in naslov shranjeno v svoji lastni oštevilčeni lokaciji. Vse te lokacije skupaj določajo prostor v pomnilniku, ki ima svoje ime, tako da je seznam takle:

Skupinsko ime: Trgovine
1. ABC Pomurka, Miklošičeva
2. Delikatesa, Titova
3. Emona, Celovška
4. Maksimarket, Trg revolucije
5. Merkator, Slovenčeva
6. Nama, Tomšičeva

Zdaj so podatki dostopni individualno, z določitvijo imena bloka in lokacije znotraj bloka. Trgovine (4) npr. pomeni Maksimarket, Trg revolucije. Tako ureditev imenujemo tabela in je zelo primerna za računalniško obdelavo. Ima videz knjige s po nekaj informacijami na vsaki strani. Kakor vidite, je ta preprosta ureditev mahoma spremenila naše gledanje na podatke.

Datoteka znanih imen in naslovov je postala skupek anonimnih podatkov. Računalniku ni potrebno vedeti, kaj podatki pomenijo, ampak le, kje so in kaj naj počne z njimi.

Podatki v tabeli Trgovine ( ) so razvrščeni po abecednem redu, vendar je malo verjetno, da bi bil to



najbolj gospodaren način za obisk trgovin. Vzemimo, da bo računalnik ugotovil, da je najboljša razdelitev takšnale:

1. Maximarket, Trg revolucije
2. ABC Pomurka, Miklošičeva
3. Emona, Celovška
4. Delikatesa, Titova
5. Nama, Tomšičeva
6. Merkator, Slovenčeva

Ta seznam je lahko shranjen v drugem polju, vendar bi to pomenilo, da so isti podatki dvakrat shranjeni v spominu. Lastniki manjših računalnikov pa vedo, da je delovni pomnilnik omejen, takšno podvajanje podatkov je lahko nerodno ali nemogoče, zato je potrebna druga metoda.

Če so pravi podatki zamenjani z njihovimi položajnimi številkami v tabeli trgovin, potem je razdelilni seznam takle:

Skupinsko ime: Dostave
1. 4
2. 1
3. 3
4. 2
5. 6
6. 5

Ta razvrstitev pomeni: Najprej pojdi v trgovino, katere podatki so shranjeni v tabeli Trgovine (4), potem v trgovino (1), nato (3)... itd. Edina pomembna informacija v seznamu je navodilo, po katerem naj obiskujemo trgovine. To pa je tudi vse, kar moramo shraniti v novi tabeli z imenom Dostave.

Dostave so sedaj kazalo k tabeli Trgovine ( ) za dostavljanje. Ko bo računalnik tiskal ta seznam, bo uporabil številke iz tabele Dostave ( ), da natisne imena in naslove iz tabele Trgovine ( ) v pravem zaporedju.

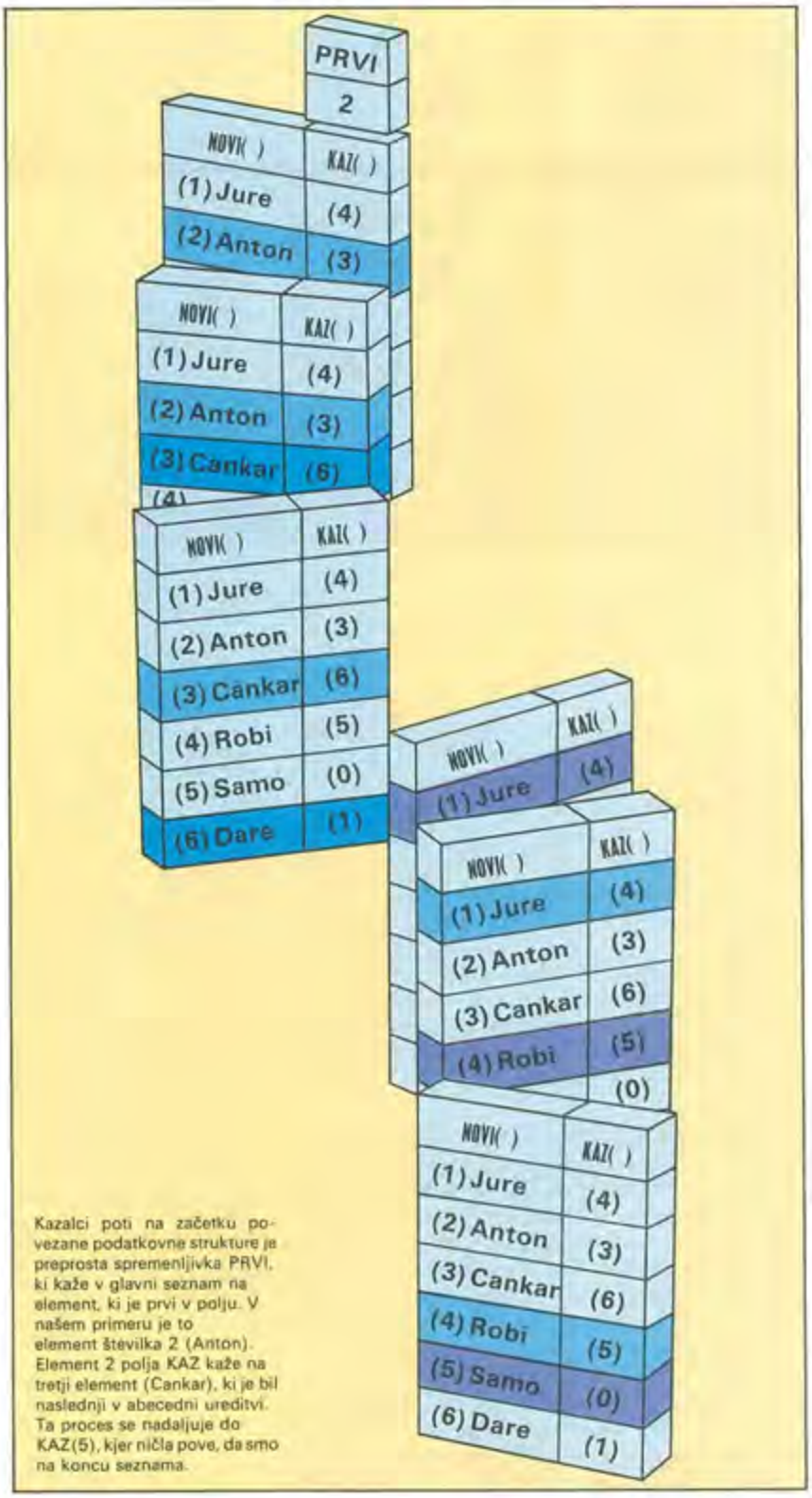
V tej preprosti nalogi so informacije — obdelali smo imena trgovin in naslove, vendar jih nismo spreminjali z različnimi ureditvami podatkov. Ureditev podatkov ne spreminja vsebine podatkov, daje pa jim različen pomen, ko jih na urejen način združuje z drugimi podatki.

Kakor lahko spremenimo seznam trgovin s tem, da jih na novo razvrstimo po seznamu skladno s tabelo Dostave ( ), lahko izdelamo druge sezname za druge namene. Ko smo razpravljali o bazi podatkov (glej stran 154), smo povedali, da nekatere podatke lahko izbiramo po kazalcih, ki jih vsebuje vsak posamični zapis. Po tej poti lahko v vsak zapis v seznamu Trgovina ( ) »vložimo« kazalec, ki bo pokazal njegovo mesto v dostavnem seznamu. Nadalje bi lahko razširili zapise še s kazalcem za seznam stalnih naročnikov. Mercator npr. vedno naroča 48 hlebov belega kruha, 12 hlebov črnega itd. Proizvodni oddelek bi lahko potem pregledal seznam samo glede na podatke, ki zadevajo samo število hlebov, ki jih morajo speči.

Na pravi poti
Igralni avtomat vsebuje 200 pesmi na 100 ploščah. Je precej drag in tudi težak. Za izbiro katerekoli pesmi pritisnemo tri tipke. Normalen čas od izbire do igre je 15 sekund. Kolut magnetnega traku na magnetofonu lahko vsebuje 200 pesmi. Magnetofon je precej cenejši in lažji. Za izbiro katerekoli pesmi previjemo trak, pritisnemo IGRA in počakamo. Normalen čas od izbire do igre je 1500 sekund.
Igralni avtomat je naprava z direktnim dostopom do podatkov, je hiter, precej specializiran in drag. Magnetofon je naprava s sekvenčnim dostopom do podatkov, počasen, manj specializiran, vendar poceni. Kasetofon, povezan z mikroračunalnikom je naprava s sekvenčnim dostopom, medtem ko ima disketa direkten dostop, čeprav jo lahko uporabljamo za shranjevanje sekvenčnih datotek.

Verižna sporočila

Indeksiranje je eden od načinov razvrščanja velikih podatkovnih baz, kakor so alfanumerični zapisi (naslovi, imena). Sklenjen seznam ali veriga (chain) je alternativen način z izrazitimi prednostmi



Kazalci poti na začetku povezane podatkovne strukture je preprosta spremenljivka PRVI, ki kaže v glavni seznam na element, ki je prvi v polju. V našem primeru je to element številka 2 (Anton). Element 2 polja KAZ kaže na tretji element (Cankar), ki je bil naslednji v abecedni ureditvi. Ta proces se nadaljuje do KAZ(5), kjer ničla pove, da smo na koncu seznama.

V računalniškem pomnilniku so podatki shranjeni kot množica napetostnih vzorcev. V digitalni računalniški tehniki poznamo le dve stopnji: to sta logična enica (pomeni, da napetost je) in logična ničla (napetosti ni). Eno je visoko stanje, drugo nizko. Kaj vzorci pomenijo, je odvisno od tega, kako jih računalnik razlaga. Tako lahko kaka beseda pomeni del ukaza, lahko je operacijska koda ali numerični podatek.

S stališča uporabnika so nekatere vrste podatkov dejansko povezane s strukturo računalnika. Programski jeziki običajno zahtevajo omejeno število različnih podatkovnih struktur. Programski jezik basic je zasnovan na numeričnih in podatkovnih nizih in vsebuje spremenljivke in pomnilniška polja za delo s tema tipoma podatkov. Število in raznovrstnost podatkovnih tipov sta določena za moč posameznega jezika.

Z uporabo spremenljivk in polj — basicovih podatkovnih struktur — lahko simuliramo nekatere druge načine obdelave podatkov.

Naslovljiva (indeksirana) polja so zelo koristen podatkovni sestav, ki se hkrati zlahka uresničuje v basicu. Vendar tudi tu opažamo omejitve, še zlasti, če se vsebina sestavov menja pogosto ali nenapovedljivo. Oglejmo si to na zgledu telefonskega imenika s podatkovno bazo o naročnikih. Baza se nenapovedljivo povečuje z novimi naročniki. Začetna baza naročnikov — imenujemo jo NOVI ( ) — v indeksnem polju je takšna:

NOVI( )	INDEX( )
(1) Jure	(2)
(2) Anton	(3)
(3) Cankar	(6)
(4) Robi	(1)
(5) Samo	(4)
(6) Dare	(5)

Indeks polja pove, kako je treba brati polje NOVI ( ), da bodo vhodi urejeni po abecedi. Prvi naročnik ali člen verige je NOVI (2), naslednji NOVI (3). Na zgledu smo pokazali le imena, vendar tak seznam vsebuje še ime in naslov naročnika in ima običajno kakih 60 znakov.

Takoj lahko ugotovimo, da bi bilo urejanje baze s 60 znaki prepočasno in bi povrhu zahtevalo veliko odvečnega pomnilniškega prostora. Zato je boljše pustiti bazo NOVI ( ) neurejeno in namesto tega izdelati indeks. Poglejmo podatkovno polje, ko smo vstavili novega naročnika:

NOVI( )	INDEX( )
(1) Jure	(2)
(2) Anton	(7)
(3) Cankar	(3)
(4) Robi	(6)
(5) Samo	(1)
(6) Dare	(4)
(7) Brane	(5)

Vsebina INDEKS ( ) nad novim naročnikom v tabeli je nespremenjena, vsebina pod njim pa je ostala v enakem vrstnem redu kakor pred vstavitvijo. Opažamo le premik za eno vrsto navzdol. Potrebovali smo torej naslednje operacije:

– poiskati mesto novega elementa (i)
– premik vseh elementov od mesta (i) do konca polja za eno mesto navzdol
– vpis novega elementa – indeksa

Vendar so lahko tudi gornje operacije pri velikem številu elementov počasne.

PRVI (2)

NOVI( )	KAZ( )	INDEX( )
(1) Jure	(4)	(2)
(2) Anton	(3)	(3)
(3) Cankar	(6)	(6)
(4) Robi	(5)	(1)
(5) Samo	(0)	(4)
(6) Dare	(1)	(5)

Vzemimo sedaj, da uredimo podatke drugače. Pustimo polje NOVI ( ) neurejeno in vpeljimo vzporedno polje KAZ ( ). Imenujemo ga kazalec. Sedaj potrebujemo novo spremenljivko PRVI ( ), ki kaže v polje NOVI (2) – na element, ki bi bil prvi v abecedni ureditvi. Uporabili smo tudi številko 0 v polju KAZ (5), kar pomeni, da bi bil ta element zadnji v abecedni ureditvi. Naš drugi zgled se od prvega opisanega razlikuje tudi po vsebini polja KAZ ( ). INDEKS ( ) pomeni: prvi element je v NOVI (2), drugi je v NOVI (3), tretji v NOVI (6) itd. PRVI (2) pove, da je prvi element abecedne ureditve v NOVI (2); KAZ (3) pove, da je naslednji v NOVI (6) in tako naprej. KAZ (5) pove, da je NOVI (5) zadnji element.

INDEKS ( ) torej podaja absolutni položaj elementov v bazi; KAZ ( ) pa pove le relativni položaj – kaže na naslednji element in ne pove ničesar o dejanskem položaju. Številka v INDEKS (4) kaže na četrti člen v urejeni bazi, medtem ko številka v KAZ (4) kaže na člen, ki je v urejeni bazi za elementom NOVI (4). Tako oblikovani podatki so v računalniški literaturi znani pod imenom sklenjeni seznami (angl. »Linked list«). Največja prednost takih ureditev je njihova prožnost.

Nadaljujemo z našim zgledom in vstavimo novega naročnika v podatkovno bazo. Polje KAZ ( ) se bo spremenilo le na dveh mestih:

PRVI (2)

NOVI( )	KAZ( )
(1) Jure	(4)
(2) Anton	(7)
(3) Cankar	(6)
(4) Robi	(5)
(5) Samo	(0)
(6) Dare	(3)
(7) Brane	(3)

a) KAZ (2) je kazal na NOVI (3) kot naslednji element za NOVI (2); sedaj kaže na NOVI (7), ki sledi elementu NOVI (2).
b) KAZ (7) je bil neuporabljen, sedaj pa kaže na NOVI (3) kot element po NOVI (7), ki sledi elementu NOVI (2).

Za vstavitev novega elementa v sklenjeno polje moramo najti element tik pred novim elementom in ga spremenimo tako, da kaže na novi element, novi element pa kaže na element, pred katerega se je vrinil. Vidimo, da je vstavitev novega elementa v seznam podobna vstavitvi novega člena v verigo: verigo pretrgamo, priključimo novi člen na prejšnjega in naslednjega v prekinitvi. Zato sklenjeni seznam včasih imenujemo tudi verižni seznam. Številke v KAZ ( ) – členi verige – pa so kazalci.

Najvidnejša lastnost opisanega načina je njegova velika povezanost. Element lahko najdemo le, če začnemo iskanje na začetku in pregledamo vse elemente pred njim. Polja, ki smo jih uporabili, so strukture z direktnim dostopom, vendar smo jih tu spremenili v sekvenčne datoteke. Povejmo še, da imajo nekateri jeziki, kakor sta lisp in pascal, zgoraj opisano lastnost verižnih seznamov že vdelano.

Te strukture lahko koristno uporabimo pri analizi naravnih jezikov (spoznavanju glasu) ali pri umetnih jezikih (sestavljanje programov), kjer podatki sami po sebi sestavljajo seznam elementov.

Urejanje kart — pravila igre

Možnost sortiranja informacij v neko ureditev je bistvena za večino programov. Obstaja veliko načinov urejanja

Urejanje je ena najpogostejših računalniških operacij, vendar tudi zelo neučinkovita operacija. Urejanje vzame računalniku od 30 do 40 % časa, in če dodamo še pridružene operacije, kakor sta združevanje in kopiranje, potem vse to zajame včasih tudi čez 50 %.

Seveda porabi približno toliko časa tudi programer, ko išče ustrezen algoritem. Boljše metode sortiranja je težko analizirati, preprostejše računalniške metode pa se dajo precej lahko razumeti. Za pomoč si vzamemo primer urejanja igralnih kart.

Vzemimo iz igre 13 igralnih kart iste barve in jih položimo na mizo. Zložimo jih v vrsto in pazimo, da enica in dvojka ne bosta na skrajni desni. Karte bomo uredili v padajočem zaporedju, to pomeni, da bo na skrajni levi kralj, nato dama in tako naprej. Problem je videti preprost, vendar ga bomo težko natančno opisali. Pravila naj bodo naslednja:

a) hkrati premikamo le po eno karto,
b) nobena karta ne more biti nad drugo,
c) karte naj zavzemajo čimmanj prostora.

Hkrati premišljajmo, kako bi to delal računalnik. Karte so njegova podatkovna baza, površina, na kateri so karte, je pomnilnik, mi pa smo program. Začnimo ga izvajati:

1. Položimo kovanec pod skrajno levo karto. Kovanec je kazalec, ki kaže na trenutno stanje ureditve. Primerjajmo označeno karto z desno sosedo. Sta padajoče urejeni? Ne! Potem naj zamenjata mesti. Kovanec pa naj ostane na svojem mestu, ob upoštevanju pravila, da lahko zamenjamo samo eno karto in da ne smemo polagati kart eno na drugo.
2. Ko sta karti urejeni, premaknimo kovanec za eno mesto v desno in ponovimo točko 1. Sedaj smo v zanki, iz katere bomo izstopili, ko bo kovanec pod skrajno desno karto. Ko smo izstopili, smo naredili en prehod.
3. Po prvem prehodu je najmanjša karta že na skrajni desni strani, tu je njeno pravo mesto. Ko bomo naredili drugi prehod, bosta urejeni dve karti s skrajne desne. S ponavljanjem prve in druge točke pridemo do zadnjega prehoda — do urejenosti.

Metoda ima nekaj pomanjkljivosti, je zelo monotona, neekonomična, saj zamenjava dveh kart zahteva tri različne operacije, in, kar je najpomembnejše, veliko primerjanj je nepotrebnih. Recimo, po prvem obhodu je as na prvem mestu in nima smisla postavljati kazalca na pozicijo 13 (tam sploh ni mogočih primerjav). V drugem obhodu je karta na desni že urejena in nima pomena postavljati kazalec na pozicijo 12. V splošnem se bo za vsak obhod element pomaknil za eno mesto v levo. Naslednje vprašanje, ki se postavlja, je, kdaj algoritem končamo oziroma izstopimo iz zanke. Urejanje končamo po prehodu, v katerem nismo zamenjali nobene karte, drugače povedano, če v zanki ni bil zamenjan noben podatek, je urejanje končano.

Metoda, ki smo jo pregledali, je v literaturi znana kot sortiranje z menjavo (angl. »Bubble Sort«). Pozitivne lastnosti te metode so preprosta programska tehnika, majhna uporaba dodatnega pomnilnika in precejšnja učinkovitost pri delno urejenih manjših množinah podatkov. Ker ne uporabljamo dodatnega pomnilnika, ampak urejamo na temeljni podatkovni bazi, nekoliko podaljšamo čas sortiranja. Prednost smo dali gospodarnosti. Tudi v nadaljevanju se bomo ognili algoritmov, ki prenašajo podatke iz temeljne baze in jih sortirajo v drugo pomnilniško polje.

1. Ponovno razložimo zmešane karte na mizo. Položimo dinarski kovanec pod drugo karto z leve. To karto imenujemo KD1.
2. Vzemimo KD1 iz vrste in pod prvo levo karto od izpraznjenega mesta položimo petdinarski kovanec. Imenujemo pripadajočo karto KD5.
3. Primerjajmo KD1 in KD5. Če sta urejeni, vstavimo KD1 nazaj na izpraznjeno mesto in nadaljujemo s točko 4. Če nista urejeni, premaknemo karto KD5 na izpraznjeno mesto, petdinarski kovanec pa za eno mesto v levo. Tako dobimo novo karto KD5. (Če smo že na skrajni levi, začnemo z izvajanjem točke 4.) Spet primerjajmo KD1 in KD5. Postopek 3 ponavljamo, dokler ne najdemo pravega mesta za KD5.
4. Premaknimo dinarski kovanec za eno mesto v desno in ponovimo koraka 2 in 3. Ko dinarja ne moremo več premikati, so karte urejene.

Opisani algoritem poznamo kot sortiranje z vstavljanjem. Sprogramirati ga je nekoliko težje kot sortiranje z menjavo, vendar je tak program učinkovitejši. Kasneje bomo spoznali še nekatere druge, bolj sestavljene algoritme.

Urejanje (sortiranje) z menjavo
Diagram ilustrira urejanje devetih kart (T je desetica). 1 in 2 nad vrstico kart označuje karti, ki se trenutno primerjata.

```
Začetek urejanja
2 8 9 3 T 5 K 6 7  začetek
1 2                1 prehoda
8 2 9 3 T 5 K 6 7
  1 2
8 9 2 3 T 5 K 6 7
    1 2
8 9 3 2 T 5 K 6 7
      1 2
8 9 3 T 2 5 K 6 7
        1 2
8 9 3 T 5 2 K 6 7
          1 2
8 9 3 T 5 K 2 6 7
            1 2
8 9 3 T 5 K 6 2 7
              1 2
8 9 3 T 5 K 6 7 2  po 1. prehodu
9 8 T 5 K 6 7 3 2  po 2. prehodu
9 T 8 K 6 7 5 3 2  po 3. prehodu
T 9 K 8 7 6 5 3 2  po 4. prehodu
T K 9 8 7 6 5 3 2  po 5. prehodu
K T 9 8 7 6 5 3 2  po 6. prehodu
konec urejanja
```

Urejanje (sortiranje) z vstavljanjem
Pri tem algoritmu se karte neposredno pomikajo na prave položaje. Urejanje se delno končuje z leve strani.

```
Začetek urejanja
2 8 9 3 T 5 K 6 7
2 1
8 2 9 3 T 5 K 6 7
2   1
9 8 2 3 T 5 K 6 7
    2 1
9 8 3 2 T 5 K 6 7
2       1
T 9 8 3 2 5 K 6 7
        2   1
T 9 8 5 3 2 K 6 7
2           1
K T 9 8 5 3 2 6 7
            2   1
K T 9 8 6 5 3 2 7
              2   1
K T 9 8 7 6 5 3 2
konec urejanja
```

Optimizacija hitrosti

Izvajanje kateregakoli programa v basicu lahko pospešimo, če posvetimo dovolj pozornosti spremenljivkam, njihovi urejenosti in strukturi programa

Basic je vsestranski in močan učni pripomoček. V njem je mogoče sprogramirati kakršenkoli algoritem, če je le na voljo dovolj pomnilniškega prostora. Seveda ne smemo biti prezahtevni glede časa izvajanja.

Urejanje na primer je proces, kjer se nekateri deli programa velikokrat ponavljajo. V večjih zankah so manjše in izvajanje je časovno neoptimalno. Če na primer urejamo 100 členov, potrebujemo od 2500 do 5000 ponovitev v zanki. Programi v basicu se interpretirajo in so običajno počasni, vendar s pravilnim kodiranjem dosežemo občutno izboljšanje. Če se mora kak ukaz ponoviti 5000-krat, in če ga lahko zapišemo tako, da prihranimo stotinko sekunde ob vsaki ponovitvi, to pomeni prihranek 50 sekund in veliko izboljšanje programa. Za opazovanje razlike med dobrim in slabim programiranjem rabimo uro in testni program. Uporabnik računalnika Commodore si bo lahko v preskusnem programu pomagal s sistemsko uro in s spremenljivkama TI in TI$. V preskusni program lahko vstavimo tudi zvočni signal na začetek in konec zanke in si pomagamo z navadno merilno uro. Preskusni program bi bil takle:

```
1000 L 500
2000 PRINT"*****GO*****": REM"BEEP"
     programski ukazi
2100 TI$ = "000000"
2200 FOR K=1 TO L
.....................................
2900 NEXT K:T9=TI
2950 REM"BEEP" programski ukazi
3000 PRINT "*****STOP*****"
3100 PRINT "Preizkušanje traja";(T9/60);"
     sekund"
```

Vrstici 2100 in 3100 sta za uporabnike računalnikov Commodore. Pri drugih računalnikih ju je treba izbrisati ali zamenjati z ustreznimi ukazi. V prostor med vrsticama 2500 in 2900 vstavimo kode, katerih čas izvajanja želimo meriti. Merimo zanko z L ponovitvami, ker bi bilo merjenje enega samega prehoda preveč nenatančno.

Navedli bomo nekaj pravil za pisanje boljših basicovih programov:

1. Ogibati se je treba vse aritmetike v zankah.

Za zgled vzemimo x^3 (tretja potenca) in trigonometrično funkcijo cos y. Izvajanje teh je razmeroma počasno. Vemo, da je tudi množenje in deljenje počasnejše od seštevanja in odštevanja. Celo najhitrejša od teh operacij (seštevanje) je počasna. V preskusni program vstavimo vrstici:

```
 900 Z=1.1
2300 X=Z↑3
```

in program izvedemo. Na našem preskusnem računalniku smo za 500 ponovitev potrebovali 27,95 sekunde. Sedaj zamenjamo vrstico 2300 in izvedemo program:

```
2300 X=Z*Z*Z
```

Porabili smo le 3,55 sekunde — silna razlika! V nadaljevanju bomo videli, pri katerem eksponentu se izplača zamenjava množenja. Na našem računalniku je to potenca 18 (X=Z↑18). Potenciranje lahko zamenjamo z množenjem le, če je eksponent celo število. Za realne in negativne eksponente moramo uporabiti temeljno funkcijo. Sami lahko preizkusimo, ali je hitrejše množenje z 0,5 ali deljenje z 2.

2. Uporabljati spremenljivke namesto numeričnih konstant.

Kadarkoli se v programu pokaže numerična konstanta, jo je treba prevesti v uporabno obliko. Preizkusimo naslednjo vrstico:

```
2300 X=X+7280
```

Porabili smo 4,63 sekunde za 500 ponovitev. Medtem ko

```
 900 C=7280
2300 X=X+C
```

porabi le 2,75 sekunde za enako število ponovitev.

3. Če je v programu ukaz GOTO, je bolje, da se program v nizu ukazov vejí naprej kakor pa nazaj. Pri vejitvi nazaj je najboljša vejitev na začetek programa.

Enako velja za ukaz GOSUB. Ko basicov interpreter naleti na enega od omenjenih ukazov, primerja tekoči naslov s ciljnim naslovom. Če je ciljni naslov večji, interpreter išče cilj — vrstico za vrstico, dokler ne najde ciljnega naslova. Če pa je cilj pred tekočim ukazom, začne iskati od prve programske vrstice naprej. Dodajmo našemu programu na začetek vrstico 56 REM in preskusimo naslednje:

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

Porabili smo 2,33 sekunde za 500 ponovitev. Medtem

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

porabi 4,85 sekunde.

4. Opredeliti vse spremenljivke v programu v vrstnem redu glede na pogostost uporabe.

Interpreter shrani imena spremenljivk v tabelo simbolov v vrstnem redu, kakor so se pokazale v programu. Zato potrebuje za spremenljivke, ki se pojavijo v programu pozneje, več časa, da jih nadje. Iz istega razloga se ogibljemo vpeljevanju novih spremenljivkv program, če lahko uporabimo že obstoječo, ki trenutno ni uporabljena. Če je uporabljena znotraj zanke, kakor pri urejanju, jo pogosto naslavljamo in jo zato definiramo na začetku pred drugimi spremenljivkami. Zaznamujemo jo lahko s poljubno vrednostjo:

```
1000 L=500:C=7280:X=0:Z=1.1
2300 A=0
```

za kar smo porabili 2,2 sekunde pri 500 ponovitvah, medtem ko

```
1000 A=0:L=500:C=7280:X=0:Z=1.1
2300 A=0
```

vzame le 2,06 sekunde.

5. Ogibati se nizom.

Operacije na nizih zahtevajo veliko pomnilnika in sprotno urejanje. Zato interpreter pogosto kliče sistemski program, imenovan »Garbage Collector« (»Smetar«), da počisti vsebino pomnilnika. Ta postopek pa zahteva veliko časa. Težko ga je popisati, ker imajo razni računalniki različno sistemsko organizacijo pomnilnika. Napolniti moramo večino uporabniškega dela pomnilnika. Poskusimo izvajati operacije na nizu tako, da se bo izvajal tudi »Smetar«. Na našem računalniku uporabimo

```
40 POKE 52,32:POKE 56,32:CLR
```

da močno zmanjšamo vsebino pomnilnika za program v basicu, in nato vstavimo:

```
1000 L=500:DIMT$(L)
1100 FOR K=1 TO L
1200 T$(K)="A"+"B"
1300 PRINT K
1400 NEXT K
```

ki uporabi velik niz (string) in oskrbi pomnilnik za kasnejšo uporabo. Ukaz PRINT (tiskanje) se izvaja v vsaki ponovitvi in izpisuje vrednost zančnega števca. Če izvajamo omenjeno verzijo s preskusnim programom, opažamo ponovne premore, ko »Smetar« preureja pomnilnik. Ti premori so včasih daljši od treh sekund. Nadaljujmo program z

```
2300 A$=LEFT$(T$(L),1):B$=A$+RIGHT$
     (T$(L),1)
```

in to traja 30,03 sekunde za 500 ponovitev. Ko bi izvajali enak program in razpolagali z več pomnilniškega prostora, preurejanje ne bi bilo opazno. Izvajanje bi trajalo tedaj le 8,66 sekunde.

Inteligentne igre

Program za igranje šaha je težko napisati. Tudi začetniki pa lahko napišejo preprost »inteligenten« program za kako igro

Nevidna roka
Naprave za igranje šaha imajo vdelane podobne sestavine kakor hišni računalnik: CPU, RAM in program v ROM, od njih se razlikujejo le po načinu predstavljanja vhodnih in izhodnih podatkov. »Phantom«, ki ga vidimo na sliki, uporablja servomehanizme in magnete, ki omogočajo računalniku avtomatsko premikanje šahovskih figur. Za skok konja čez druge figure je napisan poseben algoritem, ki odstrani vsako oviro in jo po potezi vrne na prejšnje mesto.

Veliko ljudi začne ob začetku pisanja lastnih računalniških programov sanjati o dnevu, ko bodo zmožni napisati program za igranje šaha. Ta želja pa se ne oglaša zaradi pomanjkanja teh programov na tržišču, saj jih obstaja več vrst. Primerni so tako za uporabo na hišnih računalnikih kakor tudi za računalnike, namenjene samo za igranje šaha. Toda pisanje šahovskih programov lahko postane neke vrste obsedenost mnogih programerjev, ki niso ravno strastni privrženci te kraljeve igre. Razlog za to bi lahko bilo mnenje, da je šah igra na visoki intelektualni stopnji, in da je računalnik, ki obvlada to igro, bliže stopnji, ki jo imenujemo »inteligentni stroji«. Ampak pisanje programa za igranje šaha je zahtevno delo in ga je težko opisati samo z nekaj besedami. Lahko pa razložimo nekaj glavnih zakonitosti, katerih se držimo pri pisanju tako imenovanih računalniških inteligentnih iger. To si bomo ogledali na takšni ravni, da boste lahko pisali že precej dognane programe v basicu.

Zavedati pa se moramo, da se v teh nekaj vrsticah ne bomo naučili pisanja programov za arkadne ali pustolovske igre, kakor tudi ne posnemanja različnih programskih tehnik, ki jih avtorji iger uporabljajo pri svojem delu. Svoje premišljanje o inteligentnih igrah

bomo začeli ob preprostem zgledu, ki nam bo kljub temu pokazal mnoge temeljne zakonitosti pisanja teh programov.

Mnogi otroci (kakor tudi »veliki otroci«) poznajo igro škarje — papir — kamen. Pravila igre so preprosta: igralca si morata zamisliti eno od teh treh reči in potem hkrati z roko pokažeta obliko, ki naj predstavlja izbrano reč. Zmagovalca določajo naslednja tri pravila: škarje premagajo papir (z rezanjem), papir premaga kamen (s prekritjem) in kamen premaga škarje (ker jih otopi).

Vsakdo, kdor pozna temelje programiranja, bi moral znati zlahka napisati računalniški program za ta primer. Ukaz RND nam izbere enega od treh elementov, ki so shranjeni v poljih »ŠKARJE«, »PAPIR« in »KAMEN«. Ko pritisnemo tipko za presledek, se izbrani element izpiše. Igralec odtipka svoj izbrani predmet, računalnik pa izračuna zmagovalca, prikaže rezultat na zaslonu ter shrani podatke zase in za svojega nasprotnika. Če je tipka RND resnično takšna, da izbere naključno število, bodo tudi rezultati sledili temu pravilu in ne bodo odvisni od strategije, za katero se bo igralec odločil. Sedaj moramo določiti še, kako naj izboljšamo računalnikovo strategijo, da bo zanesljivo zmagal po več kolih igranja.

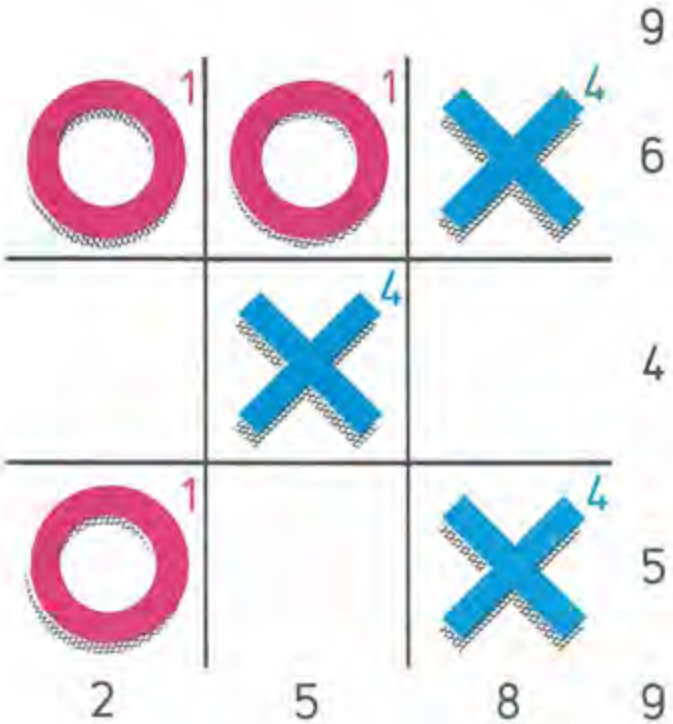
Ko smo si ogledali naključno funkcijo, smo ugotovili, da je generiranje zaporedja popolnoma naključnih funkcij nemogoča naloga tako za računalnik kakor za človeka, čeprav se ji računalnik dosti bolj približa. Po več kolih naše igre bo človek pri izbiri neogibno dajal prednost enemu od treh predmetov. Lahko napišemo podprogram, ki bo zasledoval igralčeve odločitve in si jih tudi zapomnil, uporabljajoč razporeditev teh treh reči, imenovanih npr. 1. izbira, 2. izbira in 3. izbira. Vsakokrat, ko igralec nekaj izbere, se doda ena k vsoti ustreznega predmeta v razporeditvi. Na podlagi takega sledenja računalnik ve, katerega od predmetov si je igralec pogosteje izbral. Računalnik nato lahko izbere predmet, ki bo premagal igralčevo najljubšo izbiro.

Pri tej metodi pa nastajajo trije problemi. Če izbira računalnik dosledno isti predmet, bo njegov nasprotnik to hitro izkoristil. Zato moramo narediti tako, da izbira z uporabo funkcije RND, podprogram, ki mu je dodan, pa omogoča bolj pogosto izbiro predmeta, s katerim bo porazil igralčevo najljubšo izbiro.

Naslednji problem je ta, da igralec v teku igre rad spremeni strategijo — izbere si drug predmet, da mu daje prednost. Zato je bolje, če si računalnik zapomni samo nekaj zadnjih izbir nasprotnika, recimo 20. V ta namen računalnik potrebuje razporeditev 20-krat po tri predmete, in bolj dognan podprogram, da z njim po izračunu teh treh stolpcev izbere svojo najboljšo naslednjo potezo.

Najhujša slabost tega algoritma pa se pokaže takrat, ko igralec ugane računalnikovo strategijo. Če se to zgodi, bo zlahka igral tako, da bo računalnik zanesljivo izgubil več ko polovico iger. Igralec lahko, recimo, vztrajno igra isti predmet, potem nepričakovano preskoči k drugemu in tako naprej. Zato v programu potrebujemo drugačen algoritem, ki bo odpravil te probleme. Vredno bi bilo razvijati takšne programe, ki uporabljajo naključno kakor tudi modificirano naključno metodo, pa opazovati dosežke, kakršne imajo igralci, ki jim stvari niso nič jasne.

Ljudje nismo zmožni popolnoma nespametnih ali



Zmagovalni položaj

Ocenitev položaja je bistvena za program vsake igre — celo v preprosti igri s križci in krogci. V našem primeru imamo desko predstavljeno s trikrat tremi polji. Igralčevi krogci imajo vrednost 1, računalnikovi križci pa 4. Z uporabo teh vrednosti lahko določimo katerokoli polje s seštevanjem za vsako vrstico, stolpec ali diagonalo. Vsota 12 kaže, da je zmagal računalnik, 3 pomeni zmago igralca. 8 pomeni, da sta že dva križca v liniji in da lahko računalnik zmaga. Vrednosti 1 in 4 sta uporabljeni zato, da dobimo z vsako kombinacijo krožcev in križcev različen rezultat.

naključnih odločitev, zato je vsaka izmed njih odvisna od prejšnjih. Ta povezava je lahko izredno zapletena in se je tudi sami največkrat ne zavedamo, če pa se računalnik močno približa tej funkciji, bi moral biti zmožen, da bo precej stanovitno zmagoval. Vsak igralec ima namreč svoj podzavestni obrazec igranja, v teku dolge igre pa ga bo verjetno spremenil. Zato mora program ta obrazec interpretirati med igro. Programe, ki so se zmožni naučiti tega, imenujemo »hevristične«.

Hevristični program omogoča računalniku zaznavanje sprememb v nasprotnikovi strategiji in hkratno prilagajanje svojega algoritma. Takšen program si shrani npr. pregled zadnjih 50 nasprotnikovih odločitev. Ves čas igre jih pregleduje, uporabljajoč statistično tehniko, imenovano »korelacija«.

Ta omogoča računalniku na stotine primerjav med sedanjo in igralčevo prejšnjo odločitvijo, ali eno pred njo ali z odločitvijo pred nekaj potezami. Ta primerjanja opravi računalnik po svoji izbiri. Za zgled si oglejmo povezavo med igralčevo sedanjo in prejšnjo potezo. Izberimo si naslednje oznake: škarje — lik 1, papir — lik 2 in kamen — lik 3. Najprej postavimo 3 × 3 polja in jih imenujemo CORR 1, ker nam predstavljajo prvi korelacijski preskus. Sedaj pa preglejmo zadnjih 50 nasprotnikovih potez. Vsakokrat, ko je škarjam (1) sledil kamen (3), smo dodali 1 h korelaciji CORR 1 (1,3); če pa je kamnu (3) sledil papir (2), smo eno dodali korelaciji CORR 1 (3,2). Tako smo preverili zadnjih 50 potez.

Če bi se igralec resnično odločil za poteze povsem naključno, bi morali imeti približno enake vrednosti posameznih korelacij CORR 1, to pa je v praksi zelo neverjetno. Če je igralec nazadnje izbral papir, potem lahko na podlagi največjega elementa v vrstici 2 (papir) od CORR 1 domnevamo, kakšna bo njegova naslednja odločitev. Čim večje so razlike med elementi v posameznih vrsticah, tem boljša je korelacija in s tem večja verjetnost, da računalnik ugane naslednjo igralčevo potezo. Mogoče pa imamo le majhno povezavo med igralčevo zadnjo in predzadnjo izbiro. V tem primeru moramo hitro izračunati tudi korelacijo do predpredzadnje izbire, ali med igralčevo in računalnikovo prejšnjo potezo. Problem nastane, če različne korelacije napovedujejo različne rezultate za naslednjo igralčevo potezo. Program mora v tem primeru izbrati najbolj zanesljivo odločitev. V tej preprosti igri mora samo pogledati, kateri preskus ima najbolj izrazito korelacijo. Na primer, CORR 1 utegne napovedovati naslednje verjetnosti: škarje 51 %, papir 29 % in kamen 20 %, medtem ko CORR 2 (ki primerja igralčevo in računalnikovo zadnjo potezo) daje drugačne podatke: škarje 24 %, papir 60 %, kamen 16 %. Jasno je, da je korelacija CORR 2 boljša, zato bo računalnik izbral njeno napoved. Program za inteligentne igre je lahko sestavljen iz večjega števila podprogramov, ki so pisani na podlagi različnih strategij. Vsak od njih izračuna najboljšo potezo in jo sporoči glavnemu programu, ta se nato odloči za potezo, ki jo je izračunala večina podprogramov. V nadaljnjem poteku igre lahko vsakemu podprogramu da oceno glede na to, ali je bil njegov nasvet dober ali ne.

Če ugotovimo nekatere korelacije med igralčevimi potezami ali izbirami in prejšnjimi potezami računalnika, lahko vsebuje program neke vrste »slepilen« faktor, s katerim namenoma zapelje igralca. To se najbolje uporablja pri hazardnih igrah, kjer nagrada narašča med potekom igre, tako da se splača izgubiti prejšnje runde v zameno za zmago v poznejših.

```
  5 CLS
 10 DIM C1(3,3),C2(3,3),C3(3,3)
 20 CR=0
 30 FOR I=1 TO 3
 40 IF C1(PL,I)>CR THEN BG=I:CR=C1
    (PL,I)
 50 IF C2(PP,I)>CR THEN BG=I:CR=C2
    (PP,I)
 60 IF C3(P3,I)>CR THEN BG=I:CR=C3
    (P3,I)
 70 NEXT I
 80 CT=BG-1
 90 IF BG=1 THEN CT=3
100 GET PT: IF PT=0 THEN 100
110 REM VRSTICA 100 ČAKA NA
120 REM ODTIPKANO ŠTEVILKO
130 IF CT=PT-1 THEN CS=CS+1
140 IF CT=PT-2 THEN PS=PS+1
150 IF CT=PT+1 THEN PS=PS+1
160 IF CT=PT+2 THEN CS=CS+1
170 CLS
180 PRINT "TVOJA IZBIRA:"; PT
190 PRINT "MOJA IZBIRA:"; CT
200 PRINT "TVOJ SEŠTEVEK ZNAŠA"; PS
210 PRINT "MOJ SEŠTEVEK ZNAŠA"; CS
220 C1(PL,PT)=C1(PL,PT)+1
230 C2(PP,PT)=C2(PP,PT)+1
240 C3(P3,PT)=C3(P3,PT)+1
250 P3=PP
260 PP=PL
270 PL=PT
280 GOTO 20
```

Ta program je zasnovan na igri škarje — papir — kamen in kaže, kako se lahko program med igro »uči«. Računalnik izbira med tremi številkami, 1, 2 in 3, primerja svojo izbiro s številko, ki jo vtipka igralec, in uredi izračun. Ukaz GET je uporabljen zato, da lahko kar hitro pritisnemo tipko za tri številke. Če izbiramo predmete naključno, bomo opazili, da bo po nekaj sto igrah računalnik povedel. Ta program se da prelisičiti in še naprej zmagovati, toda z dodajanjem boljših podprogramov lahko postane tudi program boljši.

Na državni univerzi države New York v Buffalu so pripravili tekmovanje med programi, ki znajo igrati poker, da so se spoprijeli med seboj v nekaj tisoč igrah (poročilo je iz revije *Scientific American* iz julija 1978). Vsi ti programi so imeli tudi možnost učenja. Skupni zmagovalec tega tekmovanja je bil program pod imenom Adaptive Evaluator of Opponents (Prilagodljivi ocenjevalec nasprotnikov), ki je najprej presodil moč svojega nasprotnika, v teku igre pa je to oceno spreminjal. Presenetljivo slabo se je izkazal program »Sells and Buys Images« (Prodaja in kupuje podobe) — njegova tehnika se sestoji iz tega, da hoče na neki način z blefiranjem »prodati« nasprotniku napačno predstavo oziroma hoče »kupiti« igralni stil svojega nasprotnika. Program »Bayesian Player« si je skušal zapomniti povzetek iger in izboljšati svoj način igranja, primerjajoč posledice prejšnjih odločitev. Program »Adaptive Aspiration Level« pa je skušal posnemati lastnost, ki naj bi jo imel človeški način igranja: prilagajati raven pričakovanja (to je stopnjo, koliko si upa tvegati) glede na prejšnje dosežke in trenutno stanje.

Na kratko smo si ogledali nekatere temeljne zamisli, ki so jih avtorji uporabili pri pisanju »inteligentnih programov«. Obstaja cela vrsta načinov za pisanje takšnih programov. Na tej podlagi pa strokovnjaki snujejo še bolj napredne in komplicirane igre, ki nam bodo kmalu dosegljive za uporabo na naših hišnih računalnikih.

Sam svoj avtor

Obstajajo že programi, ki omogočajo računalniku pisanje novih programov ali odkrivajo napake, ki smo jih vstavili ob programiranju

»Zakaj moramo še vedno pisati računalniške programe, če so računalniki že tako ‚bistri'?« Izkušeni uporabniki računalnikov ob takšnem vprašanju začetnikov samo skomignejo z rameni, toda vprašanje ni tako neumno, kakor se zdi. V svetu so bile opravljene mnoge raziskave o možnosti pisanja programov, ki bi mogli porajati druge programe, in tudi o izdelavi operativnih sistemov, ki bi znali popravljati napake, vložene ob našem programiranju.

Poslovni programi so pisani glede na specifične zahteve datotek, ki jih obdelujejo, in z definicijo vseh prenosov in sporočil potem uporabnik določi povezavo med različnimi skupinami podatkov. Ta diagram prikazuje prve korake snovanja preprostega knjigovodskega sistema.

Hišni računalniki največkrat sporočajo napake v programu kot BESEDNA NAPAKA (SYNTAX ERROR). Marsikoga takšno sporočilo jezi, ker informacija v takšni obliki pove zelo malo. V primerjavi s hišnimi računalniki pa nam večji računalniki že dajejo bolj podrobne podatke o vrsti napake v programu. Za zgled si poglejmo sporočilo o napaki.

```
1090 A = (C*2 + F$)*(FG - C)*TH + 1)
                ↑                ↑
NAPAKE: 1) SPREMENLJIVKA F$ NI DOVOLJENA
        2) ZADNJI OKLEPAJ JE ODVEČ
```

Za uporabo podobnega načina opisovanja napak pri hišnih računalnikih ni tehničnih ovir, stroški za dodaten ROM bi bili minimalni. Nekateri hišni računalniki imajo procedure za bežno preverjanje napak, večina pa ne preverja niti vpisanih ukazov. Kljub temu je dostikrat mogoče kupiti še dodaten ROM oziroma vstaviti program za odkrivanje napak na kak drug način (diskete, kasete). Tako razširimo obseg ukazov v basicu, še zlasti tiste, ki se nanašajo na razvoj in čiščenje (debugging) programov.

Ti ukazi v basicu vsebujejo naslednje:

HELP — izpiše tisto vrstico programa, kjer je napaka. Tisti del stavka, kjer se je ustavilo izvrševanje programa, pa je močneje osvetljen. Za identifikacijo izvora besedne napake v stavku je običajno to dovolj, vendar ne zadošča v vseh primerih.

DUMP — izpiše imena vseh spremenljivk in njihovo trenutno vrednost pri izvrševanju določenega programa. To nam pomaga ugotoviti, kako daleč se je program izvajal, preden se je zgodila napaka.

TRACE — v kotu zaslona prikaže številko vrstice ali številke več vrstic, ki se tokrat izvršujejo v programu. S tem ukazom lahko uporabnik sledi izvajanju programa. Tako se lahko prepriča, ali se podprogrami izvajajo v želenem zaporedju.

Pisanje programov, ki omogočajo računalniku popravljanje programskih napak človeka, v splošnem ni lahko opravilo. Za neko vrsto napak pa je to zelo preprosto. Vzemimo zgled iz programiranja v basicu, kjer se mora vsaka vrstica začeti z veljavnim ukazom. Če na primer začnemo vrstio s PRUNT ali PRONT, se da sorazmerno lahko ugotoviti, da smo želeli napisati PRINT. V tečaju programiranja v basicu smo že govorili o približnem ujemanju — to je o algoritmih za izbiranje besedil, ki so najbližje kakemu stavku, in to bi

se lahko uporabilo tudi pri programskih geslih. Ali pa naj bi prevajalni program vseboval seznam običajnih tipkovnih napak in njihove pravilne ekvivalente. Za večjo varnost bi bilo zaželeno, da računalnik skupaj z uporabnikom preveri vse spremembe, ki jih naredi.

Onstran teh temeljnih postopkov pa postane popravljanje avtomatsko. V prejšnjem primeru smo videli, da se je pojavila tiskarska napaka pri FS, ki lahko pomeni F ali pa F4. Ali pa kaj povsem drugega? Kadar napišemo program in ga pokažemo nekomu drugemu, ki je tudi vešč programiranja, bo lažje odkril in popravil naše napake, kot pa če bi jih iskali sami. Merili, ki ju bo uporabil pri kontroli našega programa, sta sobesedilo, v katerem se je pokazala programska vrstica, in pa njegova lastna izkušenost.

Tak način pregledovanja programov se že dolgo uporablja za popravke angleških besedil. Obstajajo namreč programi za pravilno pisanje angleških besed, ki jih podamo računalniku v fonetični obliki. Za vsako tako besedo poišče računalnik v svojem slovarju ustrezen ekvivalent. Računalniki imajo za te namene vpisane slovarje že z več ko 50000 besedami. Večina teh programov ima možnost učenja novih besed (npr. izgovarjava raznih imen krajev, podjetij itd.), ki jih dodaja svojemu slovarju. Bolj dognani programi nam sugerirajo pravilnejšo izgovarjavo za nekatere besede. Eksperimentalni programi za urejanje besedila imajo vdelana nekatera slovnična kakor tudi oblikovna pravila, ki opozarjajo na nepravilno rabo ločil, besedila brez odstavkov, mešane prispodobe in neprimerne pridevnike. Seveda ti programi upoštevajo tudi sobesedilo vsakega stavka in primerjave s celo knjižnico prej rabljenih zgledov.

Več truda je bilo vloženo v razvoj sistemov, ki bi znali sami ustvarjati programe in ne samo popravljati že obstoječe. V letu 1981 se je na tržišču na Zahodu pojavil program s skromnim imenom »Poslednji« (The Last One). Gre za program, ki naj bi znal pisati kakršnekoli druge programe. V reklami zanj so zatrjevali, da je to poslednji program, ki ga bo uporabnik računalnika kupil, kajti vse nadaljnje programe bo izdeloval že »Poslednji«. To sicer ni bilo res, toda ta program je kljub temu imenitno orodje pri ustvarjanju nekaterih programov, zlasti na poslovnem področju. Na zahodnem tržišču obstaja že več podobnih programov za poslovne, kakor tudi za hišne računalnike pod skupnim imenom »generatorji programov«.

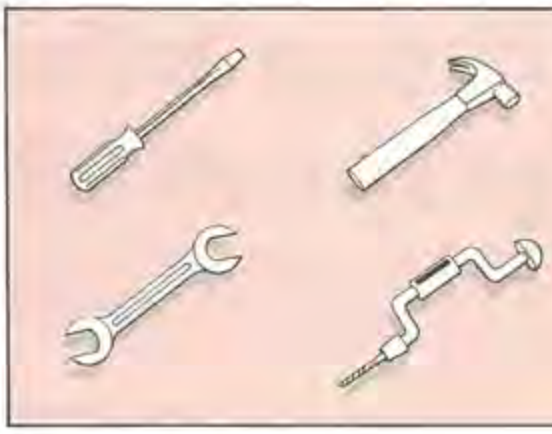
Ob preprostem zgledu si oglejmo temeljno zamisel programa z možnostjo generiranja novih programov.

```
10 PRINT "KAJ ŽELIŠ S PROGRAMOM
   PRIKAZATI NA ZASLONU?"
20 INPUT A$
30 PRINT "PROGRAM JE:"
40 PRINT "10 PRINT"; CHR$ (34); A$;
   CHR$ (34)
```

Če temu programu, ki ga lahko izvajamo na večini hišnih računalnikov, na njegovo vprašanje odgovorimo ZDRAVO, bo izpisal:

```
   PROGRAM JE
10 PRINT "ZDRAVO"
```

Če isto tehniko uporabimo za vpis podatkov o programu, ki ga imamo v mislih, potem lahko napišemo preprost generator programov. Če se da na vsa vprašanja preprosto odgovoriti, potem bo z našim generatorjem pisal enostavne programe tudi nekdo, ki še ni pisal programov.



Poklicno orodje

»Programsko orodje« za večino uporabnikov hišnih računalnikov ima obliko integriranih vezij ROM ali pa obliko vstavnih modulov. Tako običajno razširimo obseg ukazov v basicu, zlasti tistih za pisanje programov in odpravljanje napak.

Komercialno uspešni generatorji programov so napisani podobno. Večina poslovnih aplikacij se sestoji iz petih temeljnih postopkov: vhod podatkov, prikaz na zaslonu ali izpis na tiskalniku, shranjevanje podatkov v datoteke ter razvrščanje in preračunavanje podatkov. Generator programov ima za te namene standardne in zelo prilagodljive podprograme. Tak program nas vpraša, kakšno strukturo podatkov bomo uporabljali, kako naj preračuna podatke in kakšen izpis želimo na zaslonu in na tiskalniku. Glede na naše odgovore generator programov uredi spremenljivke v podprogramih, jih združi in sestavi program, ki ga želimo.

Generatorji programov sicer postajajo vse bolj dognani, vendar je malo verjetno, da bodo v bližnji prihodnosti zamenjali programerje, saj imajo razne omejitve. Kot prvo naj omenimo, da je njihova tehnika pisanja programov za poslovno rabo, kjer jih uporabljajo za obračunavanje, seznam zalog itd., še kar dobra, odpovedo pa pri pisanju iger, tehničnih programov, pri programih za urejanje besedila in še na mnogih drugih področjih.

Kot drugo slabost omenimo počasno izvajanje (zaradi obsežnih podprogramov) in pa veliko porabo računalnikovega pomnilnika. Za določen problem lahko programer napiše program, ki je hitrejši in porabi manj računalnikovega pomnilnika.

Tretja slabost pa je, da tako napisani programi niso tako preprosti za uporabo kakor programi, ki jih pišejo programerji. Vzemimo za zgled grafično zmogljivost sodobnih računalnikov, ki jih je težko uporabljati z večino programov, pisanih z generatorjem programov. Generatorji programov, ki so zdaj dosegljivi na zahodnem tržišču, nam lahko v resnici nadomestijo samo zadnjo fazo programiranja, to je pisanje ukazov. Uporabnik mora še vedno podati podatke v natanko predpisani obliki, določiti pa mora tudi želeno obliko izhodnih podatkov.

V začetnih fazah programiranja je potrebno kar precej spretnosti, da zahtevani primer pravilno opišemo. Velika podjetja, ki se ukvarjajo z razvojem programske opreme, imajo zaposlene strokovnjake, ki se ukvarjajo s sistemsko analizo. Ti poiščejo globalno rešitev problema in svoje rezultate sporočijo programerjem, da izdelajo posamezne dele programov. Generatorji programov pa se bodo morali vseh teh spretnosti še naučiti.

Programski pripomočki

Podporni programi so programski paketi, ki širijo omejene zmogljivosti basica in ponujajo možnosti za lažje čiščenje programov

Prvi hišni računalniki, kot na primer Apple II in Commodore PET, so imeli omejene zmogljivosti in so bili načrtovani predvsem za obdelavo številk in besedil. Temu primerne so bile tudi lastnosti takratnega basica. Zato je bilo napisanih veliko podpornih programov, večinoma v strojni kodi, ki so delovali zunaj basica. To so bili programski pripomočki v obliki dodatnih ukazov, ki so omogočali pisanje zahtevnejših programov in njihovo čiščenje.

Pripomočki
Tukaj je nekaj pripomočkov in izboljšav basica, ki so na voljo za nekatere najbolj razširjene hišne računalnike. Programski paketi, ki omogočajo uporabo grafike na računalnikih, ki nimajo takih zmogljivosti, so vse bolj priljubljeni.

Podporni programi	
SUPER TOOL KIT	Na voljo za 16 kb in 48 kb Spectrum — Nectarine
SPECTRUM EXPANDED BASIC	Za 48 kb Spectrum, — CP Software
SPECTRUM KEYDEFINE	Za 48 kb Spectrum — Scientific Software
PROGRAMMER'S AID	Za VIC-20 — Commodore
BUTI	Za VIC-20 — Audiogenic
TOOL BOX	Za BBC Model A in B — BBC Software
SPRITE MAGIC	Za Dragon 32 — Merlin Micro Systems
SPRITE GRAPHICS	Za 48 kb Spectrum — B Sides Software
SPRITE MASTER	Za BBC Model B — Micro Dealer UK

Od takrat je zaradi velikega zanimanja za arkadne igre prišlo do hitrega razvoja hišnih računalnikov. Vsak novi model ima nove lastnosti, ki so kmalu vključene v standardno opremo. Kljub temu pa vdelani basic, z nekaj izjemami, omogoča le majhne izboljšave osnovne verzije. Zato mora uporabnik dostikrat pisati programe z množico ukazov PEEK in POKE, s katerimi skuša obogatiti program. Zato je zdaj za večino najbolj priljubljenih računalnikov na voljo obilo podpornih programov v basicu.

Take razširitve so lahko shranjene v RAM, v notranjem ROM ali v ROM na vtičnem modulu. Uporabniki imajo raje programe, shranjene v ROM zato, ker ne jemljejo prostora v pomnilniku računalnika in se ne dajo pomotoma izbrisati. V splošnem bo program, napisan s takimi podpornimi programi, moč poganjati le na računalnikih s podobno opremo. Obstajajo tudi podporni programi, s katerimi se dajo napisati taki programi, da bodo lahko tekli tudi na računalnikih, ki niso dodatno opremljeni. To so večinoma grafični in nekateri zvočni editorji.

Zelo uporabne izboljšave basica so ukazi, ki poenostavljajo delo z grafiko (PAINT, DRAW, CIRCLE), ukazi za delo z zvokom (SOUND, PLAY, MUSIC, ENVELOPE ali besede, ki opisujejo zvočne učinke, kakor so BANG ali ZAP). Uporabni so tudi ukazi, ki omogočajo strukturno programiranje: REPEAT... UNTIL in IF... THEN... ELSE. Ti ukazi omogočajo uporabniku pisanje preglednih in razumljivih programov, za razliko od težko razumljivih programov, ki slonijo le na uporabi ukaza GOTO.

Simonov basic

Trenutno je najpopolnejša izboljšava programskega jezika basic Simonov basic, dostopen na hišnem računalniku Commodore 64 v obliki ROM na vtičnem modulu. Standarden Commodorjev basic, vdelan v 64, je precej staromoden in ne omogoča strukturnega programiranja in uporabe novejših ukazov. Čeprav je računalnik sodobno opremljen, saj ima sintetizator glasu, grafiko z visoko ločljivostjo in modularno grafiko, njegov basic omogoča programiranje teh enot z uporabo ukazov PEEK in POKE. Simonov basic je precejšnja izboljšava za Commodore, saj ima naslednje lastnosti:

1. Obširen nabor programskih pripomočkov, tako tudi funkcij, ki omogočajo dodaten nadzor nad izpi-

Programerska orodja

Opisani ukazi predstavljajo karakteristične lastnosti vsake boljše izpeljanke basica

Različni ukazi za delo z diskom, ki omogočajo formatiranje novih diskov, kopiranje in brisanje posameznih datotek in shranjevanje vseh informacij z diska na druge enote

DISK

Dodatno k ukazom za risanje ravnih črt lahko obstaja tudi ukaz DRAW, ki riše povezave med točkami in omogoča preprosto risanje likov po zaslonu računalnika

DRAW

PAINT omogoča barvanje določenih območij na ekranu z izbrano barvo. Na nekaterih računalnikih postavimo kurzor na sredino lika in računalnik potem lik barva od kurzorja proti robovom

PAINT

Namen tovrstnih grafičnih ukazov je risanje krogov, nekateri programi omogočajo tudi risanje lokov in elips

CIRCLE

Čeprav so ukazi za delo z zvokom za različne računalnike precej različni, ukaz MUSIC po navadi zaznamuje možnost igranja zaporedja not, ki morajo biti prej pripravljene v določenem nizu

MUSIC

Seznam vsebuje listo imen, tipov in velikosti vseh datotek, ki so shranjene na disku. Ukaz DIR po navadi omogoča prikaz seznama na zaslonu, ne da bi pri tem izgubili program, ki se nahaja v RAM

DIR

Z ukazoma PEEK in POKE je težko slediti gibanju krmilne ročice. Ukaz JOY pa zapiše informacijo o položaju krmilne ročice direktno v par spremenljivk, določenih z basicom

JOY

som programa, čiščenjem programa in zaščitami (zaščite pred nedovoljenim kopiranjem programa).

2. Dodatne ukaze za delo z nizi in z besedilom.

3. Posebne aritmetične operacije in ukaze za številčne pretvorbe.

4. Poenostavljene komande za ravnanje z diskom.

5. Grafiko z visoko ločljivostjo, ki omogoča kombiniranje besedila z risanjem točk in linij. Vdelana je tudi možnost risanja in pisanja v različnih barvah.

6. Grafiko z nizko ločljivostjo in ukaze s preprosto obdelavo podatkov na zaslonu. Vsebina zaslona se da shraniti na disk ali tiskalnik.

7. Preprost editor za delo z modularno grafiko.

8. Strukturno programiranje, kjer se dajo uporabiti ukazi PROC, CALL, EXEC, ukazi za zanke in vejitve REPEAT... UNTIL, LOOP... EXIT, IF... END LOOP ter IF... THEN... ELSE, ki popolnoma izriva ukaza GOTO in GOSUB.

9. Podprograme za izdelavo zvoka, ki z uporabo preprostih ukazov omogočajo popolno izkoriščenost lastnosti računalnika.

10. Preproste ukaze za delo s svetlobnim peresom, krmilno ročico in loparjem.

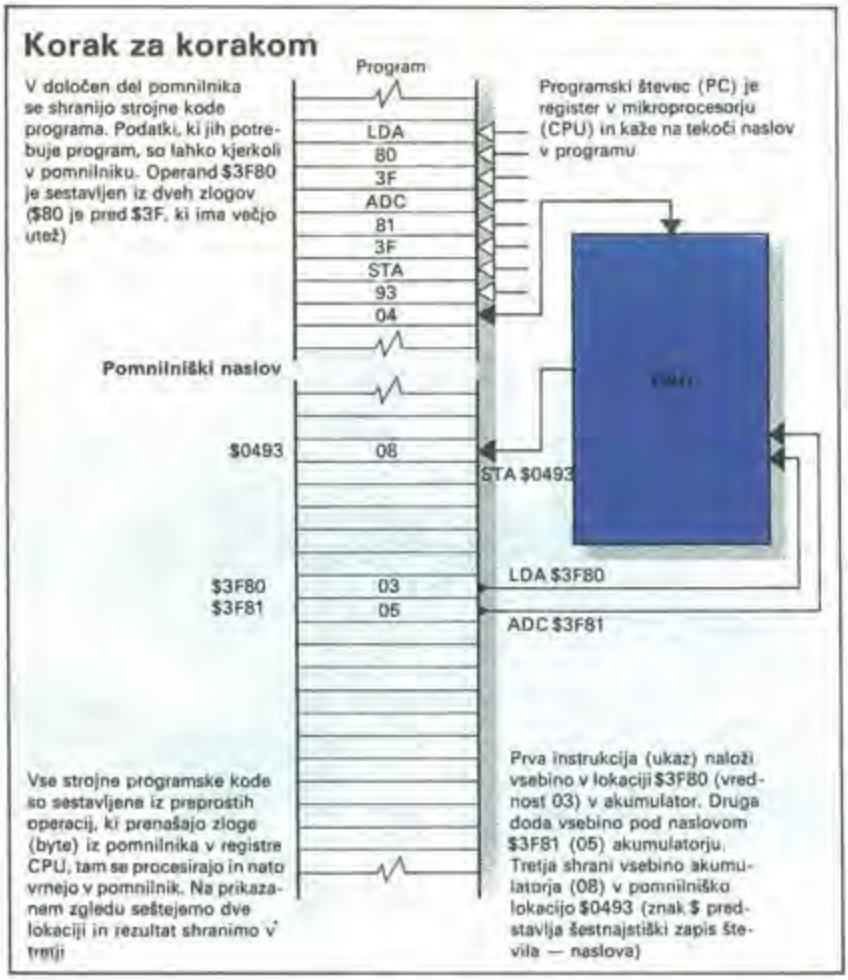
Običajno izboljšave niso tako popolne. Večina programskih paketov ponuja izboljšave za določeno področje programiranja. Na primer, vtični modul Super Expander, napisan za VIC-20, vsebuje le ukaze za delo z grafiko in zvokom. Najbolj iskane pa so tiste izboljšave, ki vsebujejo pripomočke za delo s programom. Te po navadi omogočajo izvajanje ukazov s pritiskom na eno samo tipko ter vsebujejo različne podprograme za poenostavljeno oštevilčevanje vrstic, pisanje in čiščenje programov. Podporni programi in programske izboljšave povečini omogočajo preprostejše izkoriščanje zmogljivosti hišnega računalnika. Redki pa so programi, ki razširjajo zmogljivosti računalnika. Na primer številne modularne grafike (grafike figur – sprite graphics) so navdihnile nekatera podjetja k pisanju podpornih programov, ki omogočajo uporabo te grafike na računalnikih, kjer to prej ni bilo mogoče.

Podporni programi in izboljšave, ki smo jih navedli, so le majhen del pripomočkov, ki so na voljo. Čeprav proizvajalci težijo k univerzalnemu basicu, bo vedno obstajala potreba po programskih pripomočkih, ki bodo prispevali k temu, da bo programiranje kreativen užitek, ne pa težaško delo.

Strojna koda

Uporabljati strojne kode pri programiranju pomeni velik odmik od višjih in visokih programskih jezikov, hkrati pa največje povečanje hitrosti in učinkovitosti

Korak za korakom



Doslej smo v vseh svojih izvajanjih uporabljali basic, ker je dovolj pregleden in lahek. V nadaljevanju bomo spoznavali strojne kode. Videli bomo, da lahko tako veliko bolje izrabimo lastnosti našega računalnika.

Poznati strojno kodo in znati jo uporabljati pomeni vedeti, kaj dela temeljni sestavni del našega računalnika — mikroprocesor oziroma centralna procesna enota CPU. Naj takoj povemo, da zna CPU le temeljne operacije, kakor so: premikanje temeljnih informacijskih gradnikov (bitov), njihovo primerjanje, seštevanje in razne prenose. Vse druge operacije so sestavljene iz prej omenjenih. Vsaka operacija se izvede sinhrono z osnovnim taktom. Da se ukaz ali instrukcija konča, je potrebnih več osnovnih taktov (odvisno od načina naslavljanja ukazov oziroma od vrste operanda). Povprečni ukazi, kakor seštevanje in odštevanje, se izvršijo v dveh ali treh ciklih. Kako hitro potekajo operacije, nam pove podatek, da je hitrost takta oziroma osnovne ure do 4 Mhz. Na strojnem nivoju lahko izvajamo operacije med posameznimi biti. Tako ni težko generirati temeljnih logičnih funkcij OR, AND, NOT in drugih.

Pisanje strojnih kod pa je zelo zamuden posel, še zlasti, ko je treba računati vejitvene ukaze (naslove). Zato uporabljamo zbirni jezik (assembler) in mnemonike, konstante in spremenljivke. Pregled nad delovanjem CPU ne bo zato nič manjši. Podobno kakor v basicu tudi tu opredelimo na začetku programa spremenljivke in njihova mesta v pomnilniku. V zbirnem jeziku so to stavki RMB in EQU. Zgledi, ki jih bomo navajali, se bodo nanašali na 8-bitne mikroprocesorje. Pri 16-bitnih so operacije podobne, le da je podatkovna beseda (byte) daljša.

Če pogledamo, kakšna je zgradba procesorja in kakšne so možnosti komuniciranja z zunanjim svetom (periferijo), ugotovimo naslednje: Osembitno podatkovno vodilo je uporabno za prenos podatkov in kodnih ukazov v zunanji pomnilnik (ali vhodno/izhodno komunikacijsko enoto) in iz njega. Šestnajstbitno naslovno vodilo naslavlja 65536 podatkovnih besed oziroma naslovov. Prav tolikšna je lahko osnovna kapaciteta pomnilnika.

Kakšna je organizacija pomnilnika? Nekaj naslovov uporablja mikroprocesor za naslavljanje notranjih registrov, ki so dosegljivi uporabniku. Takšni registri so na primer ura, vhodno/izhodni registri (6303). Del tega — spodnjega področja uporablja notranji pomnilnik RAM in je zato lahko neposredno naslovljiv (krajši ukazi). Preostalo področje uporabnik poljubno porazdeli med programski pomnilnik, podatkovni pomnilnik ali vhodno/izhodne enote. To doseže s tretjim vodilom, ki ga imenujemo kontrolno vodilo mikroračunalnika. Seveda je potrebno narediti še dodatno kontrolno logiko. Sestavljajo ga bralno-pisalna linija R/W, računalniški takt in še nekatere (odvisno od mikroprocesorja). Naslovno vodilo skupaj s kontrolnim krmili nadzorno logično vezje hišnega računalnika.

Notranjost mikroprocesorja sestavlja množica registrov oziroma pomnilnikov dolžine ene besede ali manj. Programski števec je 16-bitni register. Njegova vsebina kaže na trenutno stanje programa. Vmesne operacije na podatkih in ukazih potekajo na eno-, štiri- in osembitnih lokacijah. Takšen je akumulator — aritmetični register. Teh je lahko tudi več. Kazalec na sklad (stack) SP je 16-bitni register. Enako dolžino ima tudi indeksni register, ki omogoča indeksno naslavljanje ukazov. Oba omenjena registra sta dostopna uporabniku. Sklad je sestavni del notranjega pomnilnika RAM. Vanj se shrani stanje notranjih registrov ob vejitvi programa na podrutine ali ob prekinitvah (interrupt), ko se izvaja prekinitvena rutina. Shranjeni registri omogočajo vračanje programa na ukaz, ki sledi prekinitveni točki (break point).

Naj na tem mestu omenimo, da obstaja pri nas nekaj literature s področja mikroračunalnikov. Pa tudi iz vsakega boljšega priročnika lahko dobimo informacije o zbirnem jeziku.

Zbirni jezik

Omenili smo že, da je mogoče uporabiti različne prijeme pri programiranju. Učinkovitost programa in njegova dolžina sta odvisni od načina naslavljanja ukazov in pomnilnika. Različni mikroprocesorji omogočajo različne načine, ki jih uporabljamo v zbirnem jeziku (assemblerju). Mikroračunalnik 6800 je bil doslej pri nas najbolj razširjen. Danes se vse bolj uveljavlja 6809, ki je dostopen tudi že v tehnologiji CMOS. Načini naslavljanja, ki jih bomo omenili, se nanašajo na mikroprocesorja 6809 in 6303. Slednji se je uveljavil v nekaterih domačih izdelkih, ker ga mnoge lastnosti uvrščajo pred druge mikroprocesorje.

Nekatere lastnosti procesorja 6809:
- 16-bitna aritmetika
- množenje brez predznaka
- dva 16-bitna indeksna registra
- dva 16-bitna kazalca na sklad
- možnost uporabe indeksnega registra kot kazalca na sklad
- načini naslavljanja, ki jih poznamo pri 6800
- dodatni načini naslavljanja
- možnost neposrednega naslavljanja kjerkoli v pomnilniku
- razširjeno indeksno naslavljanje

Možnosti, ki jih daje materialna oprema mikroprocesorja:
- hitre prekinitve (v sklad se shrani le programski števec in pogojno-kodni register)
- krmiljenje zakasnitve na naslovnem in podatkovnem vodilu, kar omogoča uporabo počasnih in dinamičnih pomnilnikov
- blokada maskirne prekinitve (NMI) po brisanju (do definiranja kazalca na sklad)

Na kratko podajmo nekatere načine naslavljanja.

1. Naslavljanje aritmetičnih registrov (akumulatorjev). Operand je specificiran kot akumulator. Taki ukazi so na primer:
   ASL ... premik v levo
   ASR ... premik v desno (premik vsebine akumulatorja)
   CLR ... brisanje vsebine
   COM ... komplement
   Zbirnik prevede te ukaze v enozložne.
2. Neposredno in razširjeno naslavljanje. Neposredno naslovljivi ukazi se prevedejo v dvozložne strojne kode. Prvi je koda instrukcije, drugi pa koda operanda ali operand. Pri razširjenem naslavljanju zavzema operand dva zloga (byte).
3. Indeksno naslavljanje. Vrednost v indeksnem registru se prišteje k naslovu, ki je specificiran ob ukazu. Zgled:
   LDAA LOC, X k naslovu, ki ga določa spremenljivka LOC, se prišteje vsebina registra X*
   LDX O, X vsebina lokacije, katere naslov je v registru X, se prenese v register X
4. Relativno naslavljanje. Nanaša se na vejitvene ukaze. Mikroprocesor 6809 omogoča tudi razširjeno relativno naslavljanje vejitvenih ukazov (skoki, daljši od 127 lokacij). Zgled:
   BEQ 17 preskoči naslednjih 15 lokacij programa, če je rezultat prejšnje operacije nič**
   BEQ LOOP-5 skok 5 lokacij od naslova LOOP nazaj

Daljši skoki uporabljajo mnemonike JMP (skok) ali JSR (skok v podprogram).

Omenili smo že, da nekateri mikroprocesorji omogočajo 16-mestno aritmetiko. Ko govorimo o mestih, imamo v mislih dvojiški sistem. Ob koncu tega poglavja bomo prikazali v tabeli razmerje med dvojiškimi, desetiškimi in šestnajstiškimi števili.

Spojitev dveh artimetičnih registrov daje dolžino 16 mest. Včasih ju staknjena imenujemo akumulator D. Za zgled si oglejmo 16-mestno seštevanje brez prenosa v tabelaričnem zapisu:

ACCD ← – – (ACCD) + (M:M1)
Prišteje vsebino pomnilnika M in M+1 k vsebini akumulatorja D.

Vpliv na pogojno-kodni register

H : ni vpliva
I : ni vpliva
N : N = D15
Z : $\overline{D15}.\overline{D14}.\overline{D13} \ldots \overline{DO}$
V : AB15.M15.$\overline{D15}$. $\overline{AB15}.\overline{M15}$.D15
C : AB15.M15 . M15.$\overline{D15}$. $\overline{D15}$.AB15

D izraža rezultat, AB je akumulator, D in M pomnilnik. Če je kateri od zgoraj napisanih logičnih pogojev pravilno izpolnjen, se pripadajoče mesto spremeni v logično enico.

Načini naslavljanja in potrebno število taktov

Naslavljanje	Mnemonik	Operand	Koda ukaza 1.	Koda ukaza 2.	Koda ukaza 3.	Število zlogov	MPU takt 6303	MPU takt 6809
takojšnje	ADDD	Imm	C3	ImmH	ImmL	3	3	4
neposredno	ADDD	M	D3	M		2	4	6
razširjeno	ADDD	M	F3	MH	ML	3	5	7
indeksno	ADDD	disp, X	E3	disp		2	5	6

Prvo mesto kode je šestnajstiška strojna koda operacije ADDD. Iz gornje tabele je razvidno, da potrebuje mikroprocesor 6303 manj taktov za izvršitev ukaza ADDD.

Uporabili smo pogojno-kodni register (CC). Vsebina tega registra je indikator rezultata aritmetičnih in nekaterih drugih operacij. Status CC registra preskušamo z vejitvenimi ukazi. Posamezna mesta predstavljajo:

	H	I	N	Z	V	C

H ... polovični prenos s tretjega mesta
I ... maskirni bit za prekinitev IRQ
N ... negativen rezultat
Z ... ničelni rezultat
V ... prekoračitev
C ... prenos

Lahko bi še veliko napisali o programiranju in uporabi mikroračunalnikov, vendar to ni namen te knjige.

DESETIŠKO	ŠESTNAJSTIŠKO	DVOJIŠKO
0	0	0000
1	1	0001
2	2	0010
3	3	0110
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

V tabeli je prikazana 4-mestna predstavitev števil. Postopek je podoben na višjih mestih. Šestnajstiško število D3 na primer predstavimo v dvojiški obliki kot 1101 0011 in v decimalni kot 211.

* Vsebina tako dobljene lokacije se prenese v akumulator A.
** Mesto Z v pogojno-kodnem registru ima visoko stanje (logična enica).

Branje zemljevida

Visoki programski jeziki, kakor je basic, sami urejajo pomnilnik; sicer pa potrebujemo natančen načrt pomnilnika, da obvladamo računalnik

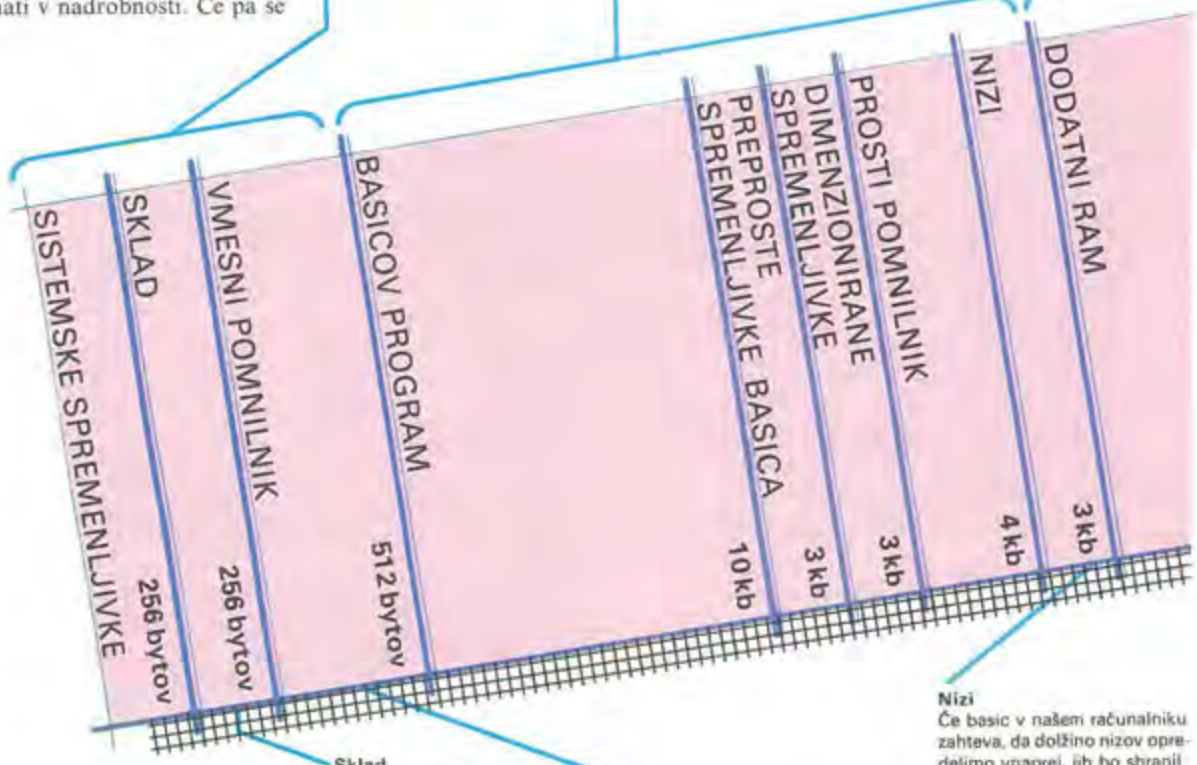
CPU v srcu računalnika ima adresno polje, ki določa maksimalno število pomnilniških lokacij, ki jih lahko doseže; v večini hišnih računalnikov je to polje veliko 64 kb. To polje pokriva ves RAM in ROM v računalniku, dodatni RAM in ROM in vse vmesnike, ki jih računalnik obravnava kot pomnilniške lokacije. Eden poglavitnih vidikov pri načrtovanju računalnika je »tabela pomnilnika« – seznam ali diagram, ki določa, kateri del pomnilnika se bo uporabljal za določene funkcije. Če programiramo samo v basicu, nam tabele pomnilnika ni treba poznati v nadrobnosti. Če pa se spustimo do strojne kode ali če sami skušamo priključiti na računalnik razne dodatke, postane poznavanje tabele pomnilnika nadvse pomembno.

Na teh straneh vidimo, kaj vsebuje tipična tabela pomnilnika. Naš zgled je bližji sistemu 6502 kot Z80, glavne značilnosti pa so skupne obema. Nekateri proizvajalci v priročniku natisnejo celotno tabelo pomnilnika, medtem ko drugi o tem molčijo. Vendar v takem primeru kaka navdušena skupina uporabnikov s preskusi kaj kmalu izdela tabelo.

Prazno polje
V tabeli pomnilnika mora biti rezervirano prazno polje za dodatne pomnilnike. Nekateri sistemi omogočajo priključitev več ko 64 kb dodatnega pomnilnika, vendar ga v tem primeru navadno vključuje in izključuje (bank switching). Posebno vezje po potrebi vključuje dodatni pomnilnik v tabelo in ga po potrebi spet izključuje.

Vrhnji sistem
Računalnik s 4 kb RAM omogoča uporabniku dostop le do 3 kb. Razlika je v vrhnjem sistemu, delu pomnilnika, ki je rezerviran za operacijski sistem. Ta del se uporablja za sistemske spremenljivke, na primer za začasne vrednosti pri računanju kompleksnih izrazov in kazalce, ki povedo računalniku, kje so posamezni podatki v pomnilniku.

Uporabniški pomnilnik
Z velikostjo tega dela pomnilnika je določena dolžina programov, ki jih lahko naložimo v računalnik. Ta podatek je eden poglavitnih pri nakupu računalnika.



Sklad
Ta del pomnilnika je namenjen izključni uporabi CPU in je organiziran kot podatkovna struktura LIFO (Last In / First Out – zadnji noter / prvi ven). Podatke lahko nalagamo na vrh sklada ali pa jih z vrha spet pobiramo. Kadar na primer v basicu izvajamo rutino GOSUB, CPU spravi na vrh sklada lokacijo v pomnilniku, na katero se mora vrniti, ko *zadene* na ukaz RETURN na koncu podprograma. CPU na veliko uporablja sklad, kadar izračunava aritmetične izraze in v primeru zank FOR...NEXT.

Vmesni pomnilnik
Če vtipkavamo podatke hitreje, kakor jih lahko računalnik obdeluje, jih shranjujemo v vmesnem pomnilniku, da jih ne izgubimo. Tudi za kasetni vmesnik potrebujemo vmesni pomnilnik, ker večina operacijskih sistemov zapisuje podatke na kaseto v blokih.

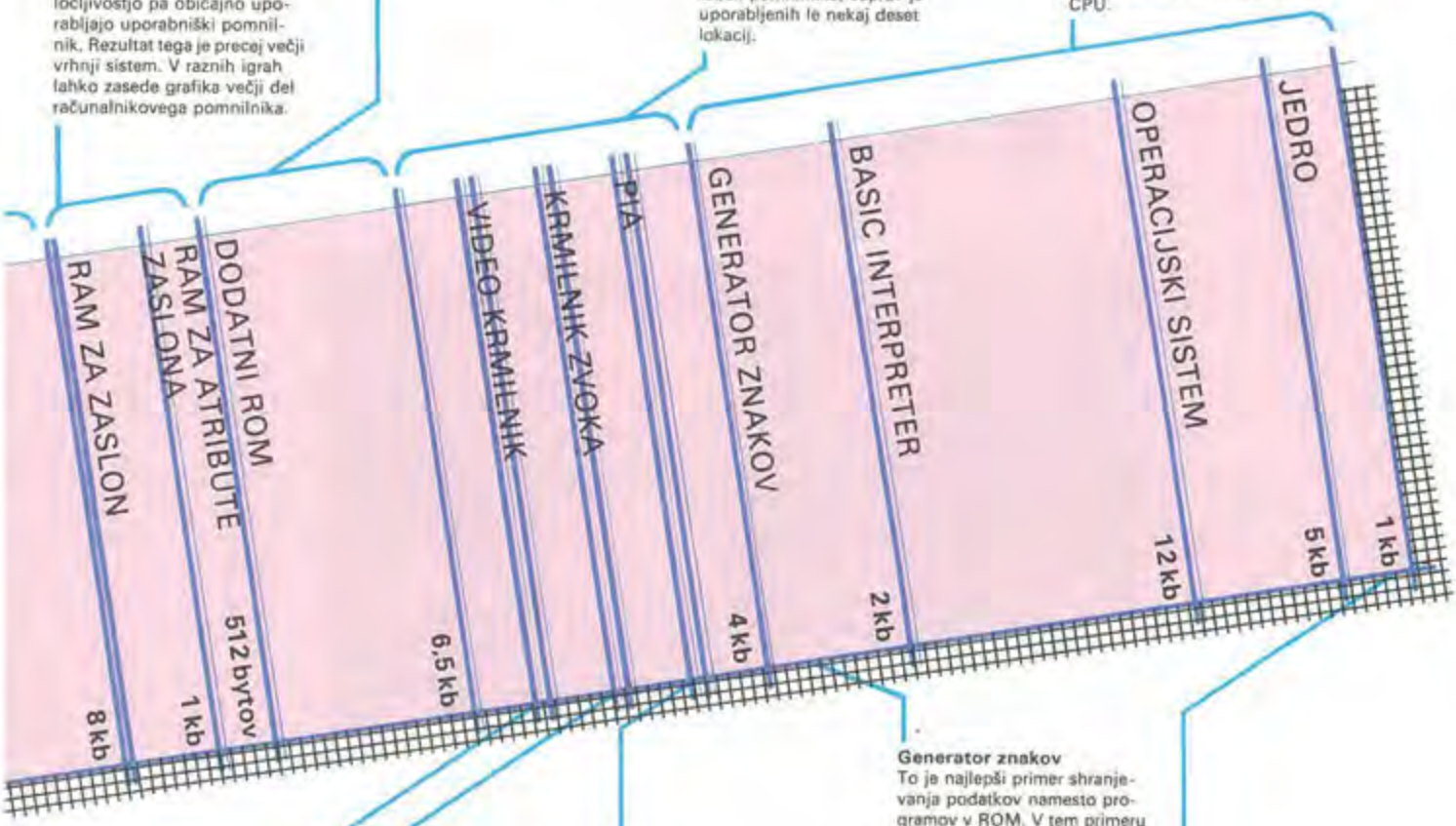
Nizi
Če basic v našem računalniku zahteva, da dolžino nizov opredelimo vnaprej, jih bo shranil v tabeli kot dimenzionirane spremenljivke. Če pa uporablja dinamične nize, katerih dolžina se spreminja, bo dejanski podatke shranil v ločenem delu pomnilnika, katerega velikost se bo prilagajala dolžini niza. V določenih časovnih presledkih bo računalnik »pometel« pomnilnik, s tem bo kratko malo počistil prostor za nize in brisal nerabne podatke.

Sistemski RAM
Nekateri računalniki imajo sistemski RAM, ki ni označen kot uporabniški RAM. Običajno se uporablja za shranjevanje vsebine in atributov (barve, utripanje itd.) zaslona. Računalniki z velikim številom grafičnih možnosti in visoko ločljivostjo pa običajno uporabljajo uporabniški pomnilnik. Rezultat tega je precej večji vrhnji sistem. V raznih igrah lahko zasede grafika večji del računalnikovega pomnilnika.

Prazno polje
Kadar uporabljamo programe, zapisane v bralnih pomnilnikih, ki jih vključimo v računalnik, se ta pomnilnik pokaže v tabeli kot dodatni ROM. Nekateri računalniki imajo konektorje za dodatni ROM kar na tiskanem vezju, kamor lahko vstavimo ROM z drugim jezikom ali kakim drugim programom. Tudi tak ROM ima rezerviran prostor v tabeli.

Vhodno/izhodna vezja
CPU lahko komunicira samo z vezji, ki se pojavljajo kot lokacije v tabeli pomnilnika, zato mora tabela vsebovati vse vmesnike in vezja, na primer vmesnik za tipkovnico, kasetofon, video vmesnik in vmesnike za periferne naprave, kakor so tiskalniki in risalniki. CPU navadno naslavlja pomnilnik v blokih (navadno velikosti 4 kb). Zato lahko vhodno/izhodna vezja zasedajo 4 kb v tabeli pomnilnika, čeprav je uporabljenih le nekaj deset lokacij.

Sistemski ROM
V hišnih računalnikih uporabljamo ROM za shranjevanje podatkov, ki jih vedno potrebujemo in se nikoli ne menjajo. Najbolj temeljna vsebina ROM je operacijski sistem, ki je v bistvu skupina programov v strojnem jeziku, ki nadzirajo delovanje računalnika. Ti programi na primer skanirajo tipkovnico in zapisujejo ter berejo podatke na kaseti. Drugi del ROM vsebuje interpreter za basic, ki prevaja ukaze v basicu v nižje ukaze, ki jih razume CPU.



Video krmilnik
Najbolj zamotane grafične operacije (npr. grafične like — sprites) raje izvajamo s strojno kakor s programsko opremo. V ta namen je v računalnik vdelana vrsta registrov (kak ducat), ki se pojavijo tudi v tabeli pomnilnika. Ti registri določajo vse komponente grafičnega lika, od barve podlage do položaja na zaslonu.

Krmilnik zvoka
S programsko opremo lahko proizvajamo samo grobe zvočne učinke, zato imajo računalniki z več tonskimi generatorji ali ADSR regulacijo zvoka (nastavljanje vzpona, padca, trajanja, iznihovanja zvočnega signala) vdelano posebno vezje za generiranje zvoka, katerega izhod je vezan na majhen ojačevalnik.

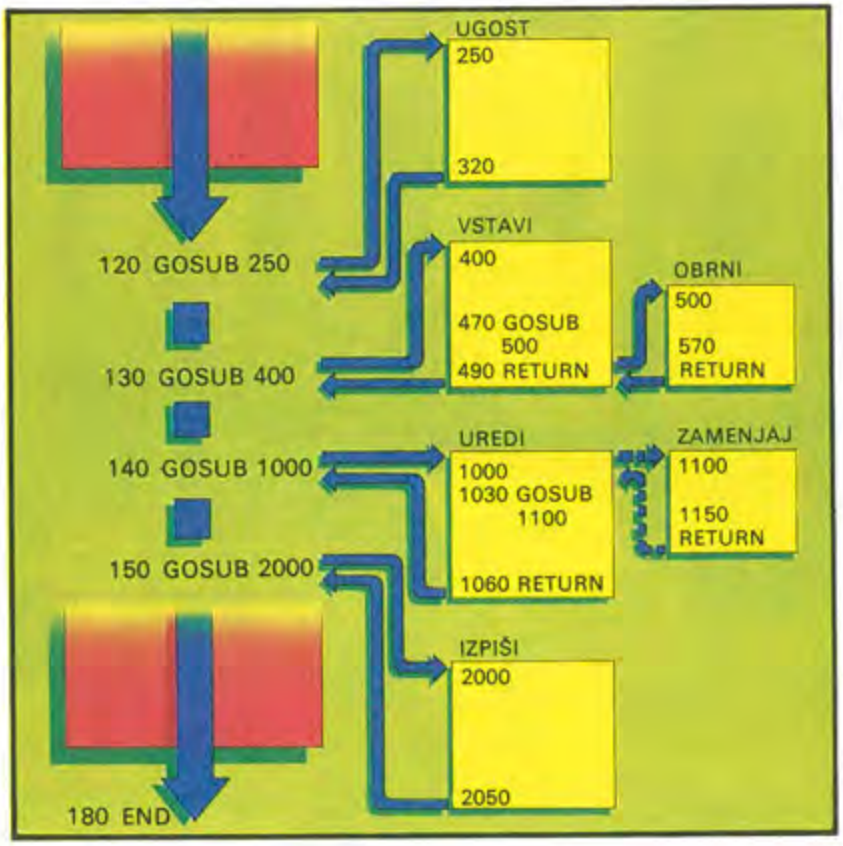
PIA
Periferne vmesniške adapterje uporabljamo za večino vmesniških opravil za priključitev tipkovnic, kasetofonov, krmilnih ročic in tiskalnikov. Najbolj obsežna integrirana vezja te vrste (kot na primer 6522 Versatile Interface Adaptor) spreminjajo vzporedne podatke v serijske in narobe ter imajo vdelana časovna vezja, s katerimi lahko programsko določimo hitrost prenosa podatkov.

Generator znakov
To je najlepši primer shranjevanja podatkov namesto programov v ROM. V tem primeru ROM shrani podatke o obliki znakov, kakor se bodo kazali na zaslonu. Pri nekaterih računalnikih lahko del ali celoten set znakov prepišemo v RAM, to nam omogoča, da sami določimo nove oblike znakov ali nove znake (na primer č, š, ž).

Jedro (kernel)
»Jedro« (ki ima pri vsakem računalniku drugačno ime) je srce operacijskega sistema. Ko računalnik vključimo, CPU avtomatično skoči na lokacijo začetka jedra in ga začne izvajati. Jedro pregleda RAM in določi, koliko pomnilnika je na voljo, postavi vse podatke v vrhnjem sistemu in preveri, ali je na računalnik priključen dodaten ROM. Jedro tudi obvladuje najpreprostejše vhodno/izhodne funkcije.

7. poglavje

Tečaj programiranja v basicu



Temeljit tečaj programiranja v basicu, najbolj razširjenem programskem jeziku, vas bo naučil pisati programe.

Izvajamo ukaze

Vaš računalnik bo naredil točno to, kar želite, če se boste z njim pravilno »pogovarjali«

Drugi jeziki

Kar se tiče mikroračunalnikov, je basic bolj v rabi kot drugi programski jeziki. Teh je pa veliko. V dobi, ko še ni bilo mikroračunalnikov, ko so bili samo velikanski računalniki, ki so napolnili cele dvorane, so znanstveniki in inženirji uporabljali jezik *fortran*. Pri mikroračunalnikih pa so poleg basica veliko v rabi še *pascal*, *forth* in *logo*. Oglejmo si nekaj njihovih značilnosti.

Pascal
Podobno kot basic so tudi pascal razvili kot učni programski jezik za bodoče programerje. Pedagogi in strokovnjaki ga zelo cenijo, kar omogoča pisanje dobrih (strukturiranih) programov. Pascal je običajno na voljo na disketah in je drag. Cenena inačica je dostopna za Spectrum za približno 30 funtov. S pascalom lahko pišemo dolge in zahtevne programe.

Forth
Programi, napisani v forthu, so veliko manj podobni angleščini kot programi v basicu ali pascalu. Fortha se tudi težje naučimo. Je pa zelo zmogljiv jezik, ker lahko zelo kompleksne programe napišemo v majhnem številu vrstic. V forthu lahko tudi sami določimo pomen ukaza, ki so npr. v basicu določeni vnaprej.

Logo
Logo je dokaj nov jezik, ki postaja vedno bolj zanimiv za šolstvo. Tako enostaven je namreč, da se ga lahko naučijo tudi majhni otroci. Pomaga spoznavati pravila programiranja in nas že od začetka uči logičnega pristopa k programiranju. S pomočjo »želve« je mogoče z njim enostavno risati po zaslonu. Na računalnik pa lahko priključimo še mehanično želvo. Želvo vodimo z enostavnimi ukazi in njena sled so črte.

Računalnike lahko uporabljamo ne da bi vedeli, kako delujejo. Toda slejkoprej kupljeni ali od prijateljev sposojeni programi in igre postanejo dolgočasni in takrat pomislimo, mar ne bi sami naredili program, čisto po naših željah in zahtevah. Ta tečaj programiranja v basicu nam bo v nekaj nadaljevanjih omogočil prav to, tj. pisanje računalniških programov.

Računalniški program je seznam navodil ali ukazov, ki računalniku zelo natančno razloži, kaj mora narediti in kako se mora naloge lotiti. Sestavljanju programa rečemo *programiranje*. Programiranje ni težko. Tudi poglobljeno znanje matematike ni potrebno, razen če želimo napisati program, ki je s področja matematike. Poznati pa moramo vsaj en programski jezik — v našem primeru se bomo seznanili z basicom.

Naš prvi jezik

Večina hišnih računalnikov vsebuje programski jezik basic. Že ime (beginner's all-purpose symbolic instruction code = večnamenska simbolna ukazna koda za začetnike) pove, da je ta jezik namenjen začetnikom, ki se bodo z njim hitro in enostavno naučili osnov programiranja. Podobno kot pravi človeški jeziki ima tudi basic svoja slovnična pravila in slovar, ki pa je precej manjši od angleščine, od koder so vzeti tudi njegovi izrazi. Basic namreč uporablja vrsto kratkih angleških besed, katerih pomen je jasen. Kot večnamenski jezik je basic primeren tako za začetnika kot izkušenega programerja.

Težava pri basicu pa je v tem, da so z leti različni proizvajalci računalnikov ta jezik spreminjali. Tako imamo danes veliko različic basica; razlike so največje pri ukazih, ki nadzorujejo tiste vidike zmogljivosti strojev, ki so se uveljavili šele v zadnjem času — to so ukazi za barve, grafiko in zvok. Nekaj teh razlik omenjamo posebej v okvirčkih.

Prav zaradi teh razlik ni mogoče pisati programov v basicu, ki bi bili primerni za vse računalnike. K sreči pa je osnova ali jedro jezika ostala ista v vseh basicih. Zato bomo začeli s tem jedrom in postopoma poskusili napisati zahtevnejše programe.

Prvi koraki

Napišimo za začetek programček in si oglejmo nekaj zakonitosti basica. Računalnik, ki bo delal po tem programu, bo delal navidezne napake. Vključite računalnik in vtipkajte spodnji program natanko tako, kot je prikazan, in upoštevajte vsa prazna mesta! Oznaka <CR> na koncu opominja, da je treba vsako vrstico še vnesti v pomnilnik s tipko Carriage Return ali Return ali Enter (odvisno od modela računalnika).

```
10 REM RAČUNALNIKI SE NIKOLI NE
   ZMOTIJO <CR>
20 PRINT "VTIPKAJ ŠTEVILO" <CR>
30 INPUT A <CR>
40 LET A = A + 1 <CR>
50 PRINT "MISLIM, DA SI VTIPKAL ŠTEVILO
   "; <CR>
60 PRINT A <CR>
70 END<CR>
```

Ko smo vtipkali program, pritisnemo tipko LIST <CR>. Program se bo pojavil na zaslonu v celoti. LIST je namreč ukaz, s katerim računalniku naročimo, naj izpiše program, ki je v pomnilniku. Program, ki se je pojavil na zaslonu, pregledamo, če ni kaj napak, nato pa sprožimo njegovo izvajanje s tipko RUN. Če pa odkrijemo napako v programu, smo lahko brez skrbi. Enostavno izpišemo vrstico, ki ima napako, znova. Spredaj moramo obvezno napisati še številko vrstice. Malo se še poigrajmo. Napišimo

```
25 REM TU JE ŠE ENA VRSTICA REM<CR>
```

in znova pritisnemo LIST. Če se želimo znebiti te ali pa kakšne druge vrstice, vtipkamo številko vrstice in <CR>. Ko smo s programom čisto zadovoljni, ga sprožimo z RUN<CR>. Pri našem programu se bo na zaslonu pojavil napis:

```
VTIPKAJ ŠTEVILO
```

Torej, vtipkajmo neko število. Na primer 7. Pritisniti pa moramo tipko z arabsko številko, kajti računalnik ne bo spoznal besede »sedem« kot 7, razen če nismo prej napisali programa, ki mu bo to omogočil. Če vtipkamo 7, se bo na zaslonu pojavil napis:

```
MISLIM, DA SI VTIPKAL ŠTEVILO 8
```

Se je računalnik zmotil ali enostavno izvaja naše ukaze? Če si ogledamo program vrstico za vrstico, bomo ugotovili, kaj je računalnik naredil pri vsaki vrstici. Prva vrstica:

10 REM RAČUNALNIKI SE NIKOLI NE ZMOTIJO

REM pomeni remark, tj. angleško za »opomba«. Za to, kar je napisano za REM, se računalnik enostavno ne zmeni. To so namreč programerjeve opombe, da on ve, kaj je naloga dela programa ali pa programa v celoti, ki sledi. V našem primeru pa je vrstica REM enostaven naslov in nam ne pove nič o programu. Kasneje bomo spoznali, kako koristne so REM vrstice, v katerih resnično opišemo, kaj dela program ali njegov del, ki sledi. Oglejmo si sedaj:

20 PRINT "VTIPKAJ ŠTEVILO"

Ko pride basic do ukaza PRINT, se del vrstice, ki sledi ukazu, izpiše ali »iztiska« na zaslonu (beseda print v angleščini pomeni »tisk« ali »tiskaj«). Kot vidimo, je besedilo izpisano med narekovajema. Pravilo basica veleva, da se bodo znaki (črke) med dvema narekovajema izpisali natančno tako, kot smo jih vtipkali. Drugačno uporabe ukaza PRINT pa bomo videli v vrstici 60. Sledi:

30 INPUT A

To vrstico bomo sedaj preskočili in se k njej vrnili po vrstici 40:

40 LET A = A + 1

Črka A je tukaj uporabljena kot spremenljivka. Spremenljivko si predstavljamo kot škatlo, označeno z etiketo, ki lahko vsebuje bodisi neko število ali pa znake. Namesto da si zapomnimo, kaj je v škatli, nas zanima samo, kako je škatla označena. To je tako, kot če bi rekli: »Dodaj mi škatlo B« namesto: »Dodaj mi škatlo, v kateri so 5-milimetrski lesni vijaki z ravno glavo«.

V tej vrstici imamo »škatlo«, označeno z A. Škatli rečemo spremenljivka, ker se lahko tisto (ali vrednost tistega), kar damo vanjo, spreminja. Spremenljivki lahko namreč pripišemo (priredimo) katerokoli vrednost. Spremenljivki A smo priredili neko vrednost v vrstici 30. Pa si oglejmo, kako smo to naredili:

30 INPUT A

Ukaz INPUT (vložiti, vstaviti) je eden od načinov, s katerimi v basicu spremenljivki prirejamo vrednost. Ko pride program v basicu do vrstice, ki se začenja z INPUT, počaka, da nekaj vtipkamo. INPUT A sporoči računalniku, da imamo spremenljivko A in da bo tisto, kar bomo vtipkali, postala njena vrednost. Če torej odtipkamo 7<CR>, bomo vstavili 7 v škatlo A. V računalniškem jeziku rečemo, da spremenljivki A priredimo vrednost 7. Sedaj ko vemo, kaj je spremenljivka in ko tudi poznamo enega od načinov, s katerim ji priredimo neko vrednost, si ponovno oglejmo vrstico 40:

40 LET A = A + 1

Ime spremenljivke, ki ji prirejamo vrednost, stoji vedno pred enačajem. Tukaj spremenljivki prirejamo novo vrednost. Naš stavek pomeni: »Naj bo (angl. LET) nova vrednost spremnljivke A stara vrednost, povečana za 1«. Ker je bila stara vrednost A 7, bomo sedaj dobili 7 + 1, tj. 8.

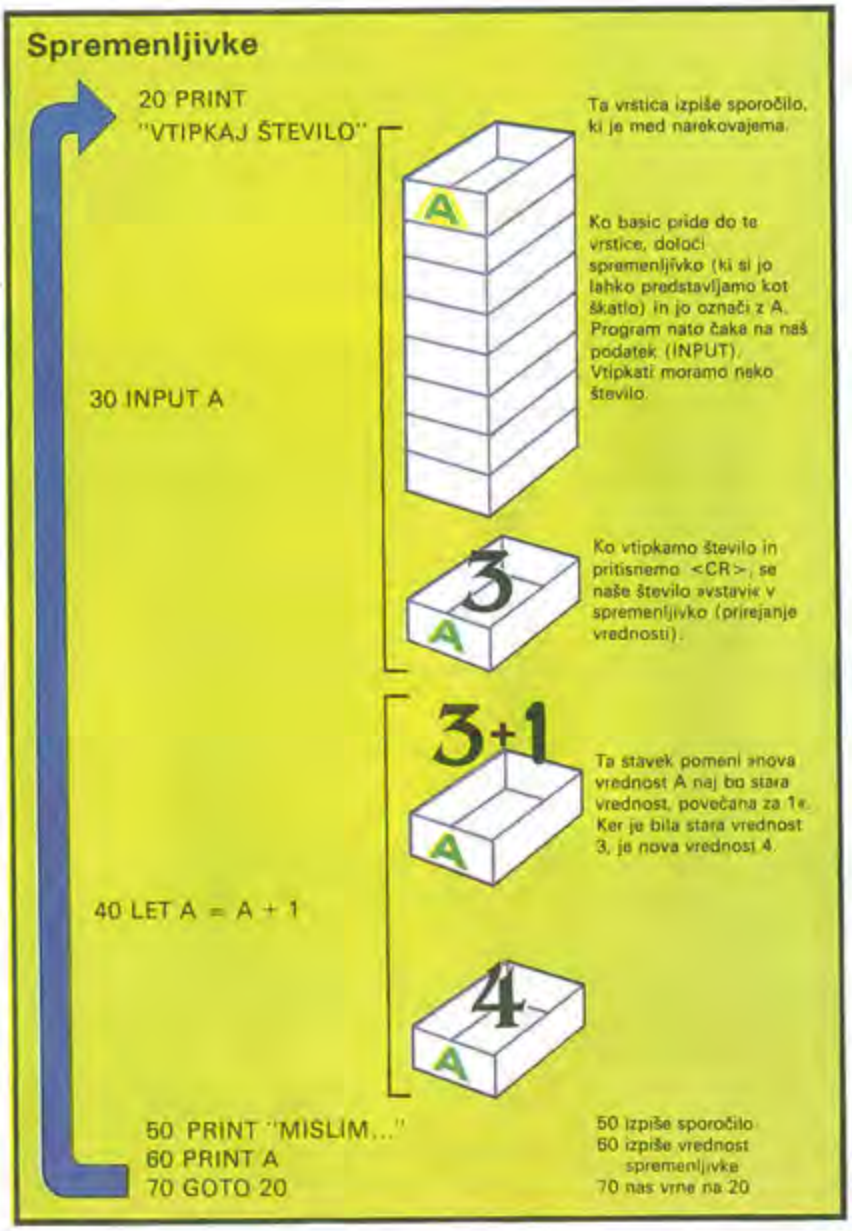
50 PRINT "MISLIM, DA SI VTIPKAL ŠTEVILO";

Spet imamo stavek s ukazom za izpis. Zato se bo na zaslonu izpisalo besedilo med narekovajema. Na koncu vrstice je podpičje. Z njim odrejamo, kje na zaslonu bo izpisano besedilo. Kasneje si bomo še podrobneje ogledali, kako uporabljamo podpičje. Oglejmo si sedaj:

60 PRINT A

Tu imamo še eno vrstico z ukazom za izpis, toda tokrat brez narekovajev. Sedaj že vemo, da program ne bo izpisal A na zaslonu, ker ta pač ni znotraj narekovajev. Zato bo basic poiskal spremenljivko, ki je označena z vsebino vrstice, ki sledi ukazu PRINT. Če jo najde, bo

Okvir spodaj nam kaže, kako v basicu uporabljamo spremenljivke. Kaže nam tudi, kako z ukazom GOTO naredimo zanko.



na zaslonu izpisal vrednost te spremenljivke. (Če je pa ne najde, bo javil napako). Ker naš program vsebuje spremenljivko A, bo basic izpisal njeno vrednost — ki je koliko?

Pravilni odgovor ni 7, kajti rekli smo, da gre basic skozi program vrstico za vrstico, sledeč številkam vrstic. Ko pride do vrstice 60, se vrednost spremenljivke poveča na 8, zato se bo na zaslonu pojavila osmica. In končno pridemo do:

```
70 END
```

Ta ukaz pove, da smo prišli do konca programa. Nekatere različice basica zahtevajo, da se vsi programi končajo z END, pri drugih pa to ni potrebno (primerjaj okvirček zgoraj).

Opazili smo tudi, da program, ko ga sprožimo, steče le enkrat. Če želimo z njim ponovno delati, moramo znova vtipkati RUN<CR>. Sedaj si bomo ogledali, kako bomo s stavkom GOTO dosegli, da se bo »zavrtel« večkrat.

Stavek GOTO

Spodaj smo izpisali isti program, ki smo mu pa dodali še eno vrstico. Če računalnika medtem niste izključili, vtipkajte novi vrstici 70 in 80, ki ju lahko preberete na dnu programa:

```
10 REM RAČUNALNIKI SE NIKOLI NE
   ZMOTIJO  <CR>
20 PRINT "VTIPKAJ ŠTEVILO"<CR>
30 INPUT A<CR>
40 LET A = A + 1 <CR>
50 PRINT "MISLIM, DA SI VTIPKAL
   ŠTEVILO ";  <CR>
60 PRINT A  <CR>
70 GOTO 20<CR>
80 END <CR>
```

Ko ste vtipkali program in ga tudi izpisali (LIST), poskusite odgovoriti na vprašanje, kaj se bo zgodilo (počakajte z izvajanjem programa RUN). No, sedaj pa sprožimo izvajanje programa (RUN<CR>). Podobno kot pri prvi inačici programa, bomo zagledali:

```
VTIPKAJ ŠTEVILO
```

Vtipkamo neko število, naj bo tokrat 8, in pritisnemo <CR>. Računalnik bo našemu številu prištel 1 in izpisal sporočilo:

```
MISLIM, DA SI VTIPKAL ŠTEVILO 9
```

Takoj za tem sporočilom pa se bo znova pojavilo sporočilo VTIPKAJ ŠTEVILO. Če vtipkamo novo število in pritisnemo <CR>, dobimo spet novo sporočilo. Razlog za to je vrstica 70:

```
70 GOTO 20
```

Ko basic pride do stavka GOTO, ne gre naprej na naslednjo vrstico, ampak se vrne na zahtevano vrstico. (GO TO v angleščini pomeni »pojdi k, pojdi na«.) V našem primeru se program vrne na vrstico 20 in se ponovi. Program se vrti v zanki. Ni načina, kako bi ga ustavili.

Seveda je program mogoče napisati tudi tako, da lahko izstopimo iz zanke, ko želimo. Enega od načinov si bomo ogledali v nadaljevanju. Sedaj pa poskusimo ustaviti program, ki ga imamo. Če ima računalnik tipko BREAK (»prelomi«), lahko z njo ustavimo program. Če potem spet vtipkamo RUN<CR>, se bo program znova sprožil.

Na koncu programa še vedno stoji stavek z END, čeprav zaradi stavka GOTO 20 dobimo neskončno zanko, zaradi katere nikoli ne pridemo do konca. Nekatere različice basica namreč zahtevajo ta stavek na koncu vsakega programa!

Če ne morete drugače ustaviti izvajanja programa, pritisnite tipko RESET. Z njo boste skoraj gotovo ustavili izvajanje programa. Nato pa spet pritisnite LIST. Če dobite na zaslonu izpis programa, boste lahko takoj prešli na vaje. Če pa tipka RESET v vašem računalniku izbriše program iz pomnilnika, ga boste morali napisati znova.

Vaje

■ Prepišite program tako, da bo računalnik izpisal število, ki ga vtipkate. (Pomoč: treba je izbrisati eno vrstico; to dosežete, če vtipkate njeno številko, nato pa pritisnete <CR>.)

■ Predelajte vrstico 70 tako, da bo program prešel na vrstico 80. V čem je razlika?

■ Spremenite vrstico 60 tako, da bo računalnik izpisal črko A in ne vrednost spremenljivke.

■ Prepišite program tako, da bo število množil z 10. Spremeniti morate tudi besedilo v vrstici 50. Znak za množenje je *.

Posebnosti računalnikov

Samo na Spectrumu moramo pritisniti LET. Na drugih računalnikih se ta ukaz razume sam po sebi. Na teh bi vrstico 20 izpisali kar A = A + 1 in ne LET A = A + 1

Ga ni na Spectrumu. Zadnja vrstica velja kot konec programa. Lahko pa dodamo STOP.

Ta ukaz se pojavi v dveh besedah (GO TO) pri Spectrumu, čeprav pritisnemo le eno tipko. Večina drugih računalnikov, razen BBC, sprejme ukaz tudi v dveh besedah

Pomembne zanke

V drugem delu tečaja si bomo ogledali, kako izstopimo iz zanke oz. kako določimo, kolikokrat se bo ponovila, in kako oštevilčimo vrstice

Prvo nadaljevanje tečaja smo končali s programom, ki je prikazan spodaj. Z njim smo bili zadovoljni, toda motilo nas je, da se je program vračal na začetek in ga pri tem nismo mogli prekiniti. Izvajanje programa smo ustavili le s tipko BREAK ali RESET.

Sedaj si bomo ogledali, kako se da izstopiti iz takšne zanke s pomočjo preizkusa. Ponavadi to naredimo tako, da uporabimo kot kriterij število, ki sicer za naše delo ne pride v poštev. Program nam je omogočil, da smo vtipkali neko število, računalnik pa je nato na zaslonu izpisal to število, povečano za 1. Odločimo se npr., da ne bomo uporabljali števila, večjega od 999. Kaj se zgodi, če v tem primeru odtipkamo število, večje od 999. Če smo že vtipkali program, dodamo še vrstico:

```
35 IF A > 999 THEN GOTO 80 <CR>
```

Sprožimo izvajanje programa in ugotovimo, da deluje tako kot prej, razen če ne vtipkamo število, večje od 999. Vtipkajmo 1000 <CR>. Kaj se zgodi?

Zakaj se je tokrat program vstavil? To smo dosegli z IF (»če«) v vrstici 35. Ko basic pride do stavka IF, ve, da sledi logični preizkus. Znak > pomeni »večji«. Vrstica 35 torej pomeni: Če je (IF) spremenljivka A večja od 999, potem (THEN) pojdi (GOTO) na vrstico 80. Če vtipkamo 1000, dobi spremenljivka vrednost, ki presega 999, zato program preide na vrstico 80, ki pomeni konec programa. Če pa seveda vtipkamo število, ki je manjše od 999, preide program na naslednjo vrstico.

```
10 REM RAČUNALNIKI SE NIKOLI NE ZMOTIJO
20 PRINT "VTIPKAJ ŠTEVILO"
30 INPUT A
40 LET A = A + 1
50 PRINT "MISLIM, DA SI VTIPKAL ŠTEVILO ";
60 PRINT A
70 GOTO 20
80 END
```

Pri tem programu lahko torej vstavljamo vedno nova števila — dokler ne presežemo 999. Stavek IF — THEN torej preizkuša naša števila in ukrepa ustrezno. Ko se program ustavi, se na zaslonu pojavi oznaka za »pripravljen«, tj., računalnik je pripravljen za drugačno delo. Ta oznaka se od računalnika do računalnika razlikuje. Na BBC se pojavi znak >, na Dragonu OK ind. S tem znakom basic sporoči, da ne izvaja nobenega programa in da čaka nova navodila.

Različni basici različno uporabljajo logični preizkus z THEN. Drugi logični preizkusu v basicu pa so: < (manj kot), = (enako), > = (večje ali enako), < = (manj ali enako) in < > (ni enako). Te preizkuse oziroma primerjave bomo srečali v nadaljevanju tečaja.

Pa si pred nadaljevanjem tečaja še sami preizkusimo v teh primerjavah:

Vaje

■ Spremeni eno od vrstic, da se bo program ustavil, če je A = 1000.

■ Spremeni eno od vrstic, da se bo program ustavil, če je število manjše od ničle.

■ Spremeni vrstico GOTO, tako da se bo program vračal na začetek, če je A enak ali manjši od 500. Pomoč: ni treba ločevati IF-THEN od GOTO.

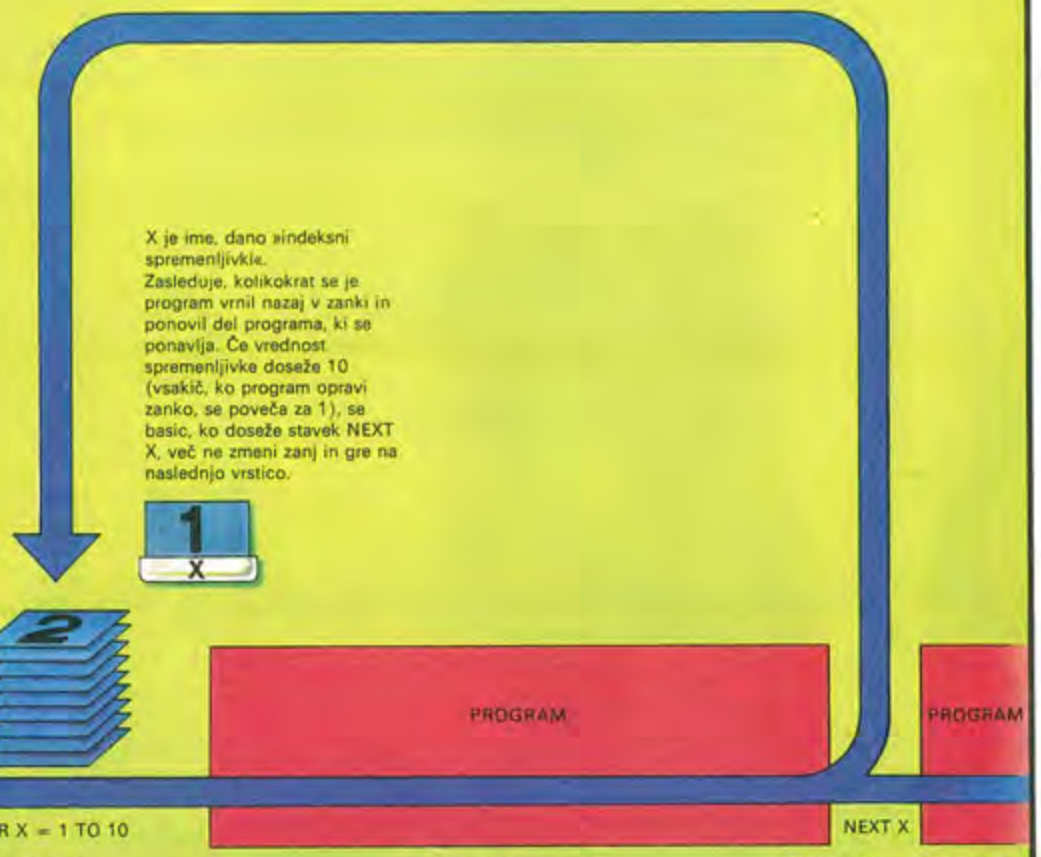
Odkrijemo še FOR-NEXT

Včasih želimo, da bi se nekateri deli programa izvedli večkrat, pri čemer naj bi bilo število ponovitev vnaprej določeno. GOTO v vrstici 70 je omogočil, da se je zanka ponovila v poljubnem številu. Kasneje smo dodali stavek IF-THEN v vrstici 35, ki nam je omogočil, da smo z »neprimernim« številom izstopili iz zanke. Vendar rešitev z GOTO marsikdaj ni najbolj primerna.

Vrnimo se k našemu staremu programu, ki smo ga priredili, da pove število, ki smo ga vtipkali, da to število pomnoži z 10 in da to opravi osemkrat.

Zanka FOR-NEXT v basicu

Stavek FOR-NEXT v basicu ustvari zanko, ki omogoča, da se del programa ponovi večkrat, število ponovitev določimo. Ko se basic sreča z besedo FOR, ve, da se bo del programa, ki sledi, ponavljal. Določi spremenljivko (v našem primeru X) in ji priredi vrednost 1. Nato izvede naslednji del programa (del, ki se bo ponavljal). Ko pride basic do stavka NEXT X, ne gre na naslednjo vrstico, ampak se v zanki vrne do stavka FOR X, poveča vrednost spremenljivke X (kar je na sliki prikazano kot prestavljanje nove karte v predal, označen z X), nato pa spet izvede del programa, ki je v zanki. Ta postopek se ponovi desetkrat.




```
10 REM MNOŽI Z 10
20 FOR X = 1 TO 8
30 PRINT »VTIPKAJ ŠTEVILO«
40 INPUT A
50 LET A = A*10
60 PRINT "TVOJE ŠTEVILO, POMNOŽENO Z
   10, JE ";
70 PRINT A
80 NEXT X
90 END
```

Vtipkajmo program, si ga oglejmo z LIST, če ni kaj napak, in nato sprožimo izvajanje z RUN. Program nas bo povabil, da vtipkamo število samo osemkrat. Po tem se bo enostavno ustavil. Razlog za to je vrstica 20.

```
20 FOR X = 1 TO 8
```

To je del zanke FOR-NEXT. (Slovensko bi ji rekli »zanaslednji.«) To je ena najboljših rešitev (struktur), ki jih ponuja basic. Posvetimo se ji.

V tej zanki smo v našem primeru začeli tako, da smo določili spremenljivko X. Lahko bi jo pa poimenovali kakorkoli, razen A, ker smo A že uporabili na drugem mestu. FOR zahteva NEXT, ki pa se pojavi kasneje v programu — po delu, ki se ponavlja (v našem primeru osemkrat). FOR vrstica v zanki FOR-NEXT ima vedno naslednjo obliko:

FOR spremenljivka = začetna vrednost TO (do) končna vrednost.

V našem primeru, FOR X = 1 TO 8, smo spremenljivko označili z X in ji dali začetno vrednost 1. Računalnik nato izvaja del programa, ki sledi: število, ki ga vtipkamo, množi z 10 in ga izpiše na zaslonu. Nato pride do vrstice 80, ki pravi, da je treba preiti na naslednji X (NEXT X). Program se vrne na vrstico 20, kjer je navodilo za spremenljivko X. Pri vrnitvi na 20 poveča X za 1, tako da dobi spremenljivka X vrednost 2. Nato se del programa med dvema vrsticama zanke FOR-NEXT izvede drugič. Ko pride program znova do NEXT v vrstici 80, so spet povrne na 20 in poveča vrednost X na 3.

Program ponavlja zanko, dokler se vrednost X ne poveča na 8. S tem je izvajanje zanke končano. Vrstica NEXT X več ne vrne programa na FOR X in program preide na naslednjo vrstico.

Druge uporabe zanke FOR-NEXT

Zanko FOR-NEXT lahko uporabimo tudi, da dosežemo zastoje v izvajanju programa. Včasih namreč ne želimo, da bi vse potekalo z maksimalno hitrostjo, zato delovanje navidezno upočasnimo. Sami ste ugotovili, da so se odgovori v programu množenja z 10 pojavili v hipu. Naredimo sedaj tak program, da bomo imeli vtis, da računalnik nekaj časa razmišlja. Z zanko FOR-NEXT bomo dosegli zastoj v izvajanju programa. Dodamo vrstici 52 in 54.

```
10 REM MNOŽI Z 10
20 FOR X = 1 TO 8
30 PRINT "VTIPKAJ ŠTEVILO"
40 INPUT A
50 LET A = A * 10
52 FOR D = 1 TO 1000
54 NEXT D
60 PRINT "TVOJE ŠTEVILO, POMNOŽENO
   Z 10, JE ";
70 PRINT A
80 NEXT X
90 END
```

Novi vrstici smo dodali znotraj prvotne FOR-NEXT zanke. Oglejmo si ju posebej:

```
52 FOR D = 1 TO 1000
54 NEXT D
```

D zavzame vrednost 1 in program gre na naslednjo vrstico. Tu je ustrezna vrstica NEXT. Znotraj te zanke se pravzaprav nič ni zgodilo, program je le skočil nazaj na vrstico 52 in povečal D na 2. To se zgodi tisočkrat in šele nato gre program na naslednjo vrstico in izpiše odgovor. Računalniki so sicer hitri, vendar tudi takšne reči trajajo nekaj časa. Različni računalniki za takšne zanke potrebujejo različno veliko časa. Epson HX-20 je za to zanko potreboval 2,9 sekunde, Spectrum pa 4,5 sekunde. Poskusite spremeniti zgornjo mejo v vrstici 52.

Če pa naj bo računalnik še bolj podoben človeku, dodajte naslednje tri vrstice:

```
56 PRINT "DA POMISLIM..."
57 FOR E = 1 TO 1000
58 NEXT E
```

Izpišite program (LIST) in sprožite izvajanje (RUN). Sedaj imamo že dva zastoja, katerih namen je zapravljanje časa.

Dodajte še naslednji vrstici:

```
51 REM TA ZANKA ZAPRAVLJA ČAS
55 REM TUDI TA ZANKA ZAPRAVLJA ČAS
```

Sedaj pa izpišite program (LIST) in si ga dobro oglejte. Vse vrstice, ki smo jih dodajali pozneje, imajo svoje pravo mesto. Sedaj se bomo pogovorili prav o tem — številčenju vrstic.

Naš prvi program smo začeli z vrstico 10 in vsaka naslednja vrstica je bila označena z okroglimi deseticami. Končali smo z vrstico 90. Lahko bi izbrali tudi druge številke, npr. 1, 2, 3... 9. Toda če bi takrat tako postopali, kako bi kasneje vrinili nove vrstice? Programerji morajo vedno nekaj popraviti, zato pri oštevilčenju preskakujejo števila. Zato bi lahko začeli tudi s številom 100 in nadaljevali v »skokih« po 50.

V nekaterih basicih imamo koristen ukaz AUTO. Z računalnikom, ki ima tak basic, z lahkoto oštevilčimo vrstice programov. Ali ima vaš računalnik ukaz AUTO, ugotovite, če odtipkate:

```
AUTO 100, 10 <CR>
```

Če računalnik ima ta ukaz, bo na zaslonu izpisal:

```
100
```

Za številko 100 je prazen prostor in za njim kurzor. Kurzor je znak na zaslonu (včasih črta, včasih kvadratek), ki kaže, kje se bo pojavil naslednji znak. Tukaj napišemo prvo vrstico. Nato pritisnemo <CR> in kurzor se preseli na naslednjo vrstico, ki pa že nosi številko 110. Ukaz AUTO lahko uporabljamo le sam po sebi ali pa skupaj z enim ali dvema »argumentoma«. *Argument* je matematični pojem. V izrazu 2 + 3 = 5 sta argumenta 2 in 3. Ukaz AUTO lahko tako uporabljamo sam (tj. AUTO<CR>) ali z enim ali dvema argumentoma (npr. AUTO 100<CR> ali AUTO 300,50<CR>). Sam AUTO ponavadi doseže, da se številke vrstic večajo za 10. Če uporabimo en argument npr. AUTO 100 <CR>, bo prva številka 100 (v našem primeru), naslednje se bodo pa večale v koraku po 10 ali drugače, odvisno od računalnika. Če pa podamo dva argumenta, potem prva številka določi številko prve vrstice v programu, druga pa korak. AUTO 250,50<CR> bo dal kot številko prve vrstice v programu 250, naslednja številka bo 300 in tako naprej, vedno za 50 več. Celo na najenostavnejšem računalniku vam ne bo zmanjkalo ne novih vrstic, ne vrstic med starimi vrsticami.

Posebnosti računalnikov

IF — Večina mikroračunalnikov uporablja ta ukaz v obliki IF A >999 THEN 80 ali IF A >999 THEN GOTO 80. (Slednjo obliko ima tudi Spectrum.)

AUTO — Tega ukaza ni na Commodore Vic 20, Dragon 32 in Sinclair Spectrumu.

Preidimo k stvari

Zakaj je pri pisanju računalniškega programa treba paziti na ločila?

V izpisu programa smo v prvem delu tečaja programiranja lahko opazili, da je na koncu vrstice 50 podpičje. Tedaj ni bila pojasnjena vloga tega ločila v basicu, čeprav je zelo pomembna. Uporablja se v skoraj vseh različicah basica za združevanje podatkov, ki so sestavni deli stavkov PRINT. Tu združevanje pomeni povezavo v serijo (drug za drugim). Vrstici 50 in 60 sta se glasili:

```
50 PRINT "MISLIM, DA SI VTIPKAL
   ŠTEVILO";
60 PRINT A
```

Vrstica 50 izpiše besedilo znotraj dvojnih narekovajev. Vrstica 60 pa izpiše vrednost spremenljivke A. Z vstavitvijo podpičja na koncu vrstice 50 dosežemo, da se vrednost spremenljivke A izpiše takoj za besedilom iz vrstice 50. Če podpičja ne bi bilo, bi se vrednost spremenljivke A izpisala v vrstico, ki bi sledila izpisu besedila.

Naslednji program nam bo pojasnil še nekaj lastnosti podpičij, uporabljenih v basicu. Vtipkamo program ter ga poženemo z ukazom RUN. Od tod naprej bomo izpuščali opozorilo < CR > na koncu vsake vrstice, ki je opozorilo, da je potrebno pritisniti tipko RETURN. Ta program spreminja temperaturne enote. Vpišemo temperaturo v stopinjah Celzija (°C), računalnik pa jo sam avtomatsko izpiše v enotah Fahrenheit (F).

Naložimo ta program, ga izpišemo z ukazom LIST, da se prepričamo, ali je bil pravilno vpisan, ter ga poženemo z ukazom RUN. Računalnik bo najprej zahteval, da mu vpišemo najnižjo temperaturo. Vpišemo npr. −5. Nato bo zahteval, da mu vpišemo najvišjo temperaturo. Vpišemo npr. 10. Program bo nato spremenil vse temperature med −5 in 10 stopinjami Celzija, po eno °C, v istovredne temperature, izražene v Fahrenheitih. Izpis na zaslonu bo nekako takšen:

```
-6 V STOPINJAH C JE 21.2 V FAHRENHEITIH
-5 V STOPINJAH C JE 23 V FAHRENHEITIH
-4 V STOPINJAH C JE 24.8 V FAHRENHEITIH
```

Opaziti je, da vrstice niso izenačene po dolžini (zaradi decimalnih števil), vendar je vsaka vrednost (v °C in ekvivalent v F) napisana v eni vrsti. Ko program nekajkrat poženemo, mu spremenimo vrstico 80 tako, da le zamenjamo podpičja z vejicami in nato program ponovno poženemo z ukazom RUN. Kakor lahko vidimo, postane izpis zelo zmeden.

Naslednji program nam bo še bolj pojasnil pomen vejice oziroma podpičja v basicu. Napišemo NEW < CR >. Sedaj pa vpišemo:

```
10 REM PRIMERJAJ ; Z ,
20 PRINT "TA VRSTICA UPORABLJA
   PODPIČJA"
30 PRINT "M";"A";"R";"K";"O"
40 PRINT "TA VRSTICA UPORABLJA
   VEJICE"
50 PRINT "M","A","R","K","O"
60 END
```

Računalnik bo izpisal vrstico 30 kot MARKO, medtem ko bo vrstica 50 izpisana kot M A R K O. Računalniki se tu med seboj lahko tudi razlikujejo (glej sestavek Posebnosti računalnikov). Vejica se v basicu uporablja v različne namene. V stavku PRINT ima ta učinek, da se med posameznimi deli informacije izpiše na zaslonu običajno od 8 do 16 presledkov, odvisno od zvrsti basica. V primeru, ko v stavku PRINT ne uporabimo ne vejic ne podpičij, se bodo posamezni deli informacije izpisali v različnih vrsticah.

Naš program za spreminjanje temperatur nam poleg pojasnjevanja uporabe podpičij nudi še pregled med ukazi, ki so bili že opisani v prvih dveh poglavjih tečaja programiranja. Vrstici 30 in 50 pripišeta spremenljivkama L in H vrednosti za najnižjo in najvišjo temperaturo. Vrstica 60 je prvi del zanke FOR...NEXT. Čeprav je videti, da se ta zanka razlikuje od zank FOR...NEXT, ki smo jih doslej že srečali, ker uporablja črke namesto številk, ni med njimi v bistvu nobene razlike. Med L in H, ki ju tu uporabljamo, sta spremenljivki s številčnima vrednostma, ki smo ju

vpisali skladno s stavkoma INPUT L oz. INPUT H v programu. Če vpišemo, kakor v prejšnjem primeru, temperaturi −5 in 10, je stavek FOR X = L TO H istovreden stavku FOR X = −5 TO 10.

Vrstica 80 v bistvu pove: napiši (PRINT) vrednost spremenljivke X (ki se začne pri vrednosti najnižje temperature ter se povečuje po 1° do najvišje temperature), kateri sledi direktno v isti vrsti (zato smo tudi uporabili podpičje) besedilo v narekovajih, kateremu ponovno sledi (še eno podpičje) vrednost spremenljivke F. Če pazljivo pogledamo spremenljivko F, vidimo, da je to temperatura v °C, ki smo jo množili z 9, delili s 5 ter ji prišteli 32, da smo dobili temperaturo v Fahrenheitih. Vrstica s stavkom NEXT X skrbi, da se spreminjanje temperatur vrši, dokler ne dosežemo gornje meje v zanki FOR...NEXT. Preden si gremo pogledat še nekaj zahtevnejših primerov uporabe stavka PRINT, se ustavimo za trenutek še pri vrstici 70 našega programa:

```
70 LET F = X*9 / 5 + 32
```

Ta vrstica pripiše vrednost spremenljivki F (kar pomeni Fahrenheit). Program najprej vzame vrednost X (temperatura v °C), ki jo množi z 9, deli s 5 ter ji prišteje 32. V običajnih matematičnih knjigah bi bila ta formula napisana v obliki: $F = C \times 9 \div 5 + 32$. Basic uporablja znak * za množenje, / za deljenje, + za seštevanje in − za odštevanje.

Kot v običajni matematiki je tudi tu v basicu pomemben vrstni red, po katerem potekajo matematične operacije. Na prvem mestu je množenje, kateremu sledi deljenje, nato seštevanje in na koncu odštevanje. V primeru, ko so deli številčnega izraza obdani z oklepaji, se opravijo najprej. Če hočemo opraviti najprej seštevanje in šele v drugi fazi množenje, se mora seštevalni del izraza obdati z oklepaji. Vzemimo za zgled, da želimo izračunati znesek, ki ga moramo vplačati za trimesečno naročnino za rtv, in sicer za televizijo (tv) in avtoradio (ar).

```
N = (TV + AR) * 3
```

Če je enomesečna naročnina za televizijo (tv) 470 din in za avtoradio (ar) 109 din, si najprej izračunamo, koliko znese enomesečna naročnina za televizijo in avtoradio skupaj. Šele nato množimo s 3, da dobimo znesek, ki ga je potrebno vplačati za trimesečno naročnino rtv, in sicer za televizijo in avtoradio skupaj. Izračunani znesek je 1737 din. Če oklepajev ne bi bilo, bi bil izračunani znesek 797 din, kar je seveda narobe. Vedno se moramo prepričati, ali bodo aritmetične operacije potekale v želenem vrstnem redu.

Print using

Z namenom, da si ogledamo še zadnjo izboljšavo v našem programu za spreminjanje temperatur, program ponovno vtipkamo in ga z ukazom RUN poženemo. Nato vpišemo npr. −10 za najnižjo temperaturo in 10 kot najvišjo temperaturo. Kakor smo že videli, je izpis na zaslonu zelo pomanjkljiv, to pa zaradi podpičij, ki smo jih uporabili v vrstici 80 z namenom, da povežemo vse dele stavka PRINT pri izpisu v eno vrsto, namesto da jih izpišemo v različnih vrstah. S podpičji smo uredili izpis v vodoravni smeri, nismo ga pa v navpični smeri. Ker so lahko številke, ki označujejo temperaturo v °C in F, različno dolge, se nam pokvari navpična urejenost izpisa.

Skoraj vse verzije basica imajo posebno obliko stavka PRINT, ki se glasi PRINT USING. Le-ta omogoča, da se števila ali besede izpišejo v urejeni obliki. Če hočemo izpisati vrednosti spremenljivke X, za katero že vnaprej vemo, da se spreminja od −99 do 99, lahko dosežemo z uporabo stavka PRINT USING »# # #«; X, da bodo kolone izpisa številk lepo poravnane. Trije znaki »#« dovoljujejo izpis spremenljivke X v obliki treh številk ali v obliki dveh številk, pred katerima je znak minus (−). Če vzamemo za X več ko tri številke, bo izpis napačen (vseboval bo le tri številke). Če pa vpišemo za X le dve ali eno številko, bo le-ta izpisana pravilno in na pravem mestu. V primeru, da je potrebna decimalna pika, jo lahko uvrstimo med znake #. Stavek ima tako lahko npr. tudi tako obliko: PRINT USING »# # #. # #«; X. Uporabiti moramo en znak »#« za vsako številko. Decimalne pike pa se bodo pojavile avtomatsko.

Spremenimo sedaj program tako, da spremenimo vrstico 80 ter dodamo vrstice 82, 84 in 86.

```
80 PRINT USING "# # #";X;
82 PRINT "V STOPINJAH C JE";
84 PRINT USING "# # #.# #";F;
86 PRINT "V FAHRENHEITIH"
```

Pregledamo še enkrat program tako, da ga izpišemo z ukazom LIST ter ga nato poženemo. Vse kolone bi morale biti sedaj lepo poravnane. V enem od naslednjih poglavij si bomo ogledali, kako lahko shranimo program, da nam ga ni treba vsakič ponovno vtipkati.

Naloge:

- Poskusite vpisati −1000 kot vrednost za najnižjo temperaturo. Zakaj program sedaj ne deluje? Kako bi spremenili stavek PRINT USING v vrstici 80, da bi program deloval?
- Spremenite vrstico 84 tako, da se bodo izpisovala samo cela števila (brez decimalnega dela).
- Napišite program, ki bo spreminjal dinarje v dolarje pri menjalnem tečaju: 1 $ = 180 din.

Posebnosti računalnikov

PRINT USING

Tega pripomočka ni pri računalnikih Commodore 64, Oric, Spectrum, ZX 81 in BBC Micro. Vendar lahko pri računalniku BBC omejimo število decimalnih mest pri izpisu z uporabo stavka:
@%=131594

COMMA

Pri uporabi vejice med posameznimi deli informacije v stavku PRINT se bodo le-ti izpisali ločeno, tako da bo med njimi niz presledkov, katerih število je odvisno od tipa računalnika. Pri računalnikih Dragon 32, Spectrum in ZX 81 je ta niz dolg 16 presledkov, pri BBC in Commodore 64 pa 10 presledkov, medtem ko ima Oric le 4 presledke.

Pomen rutinskih programov

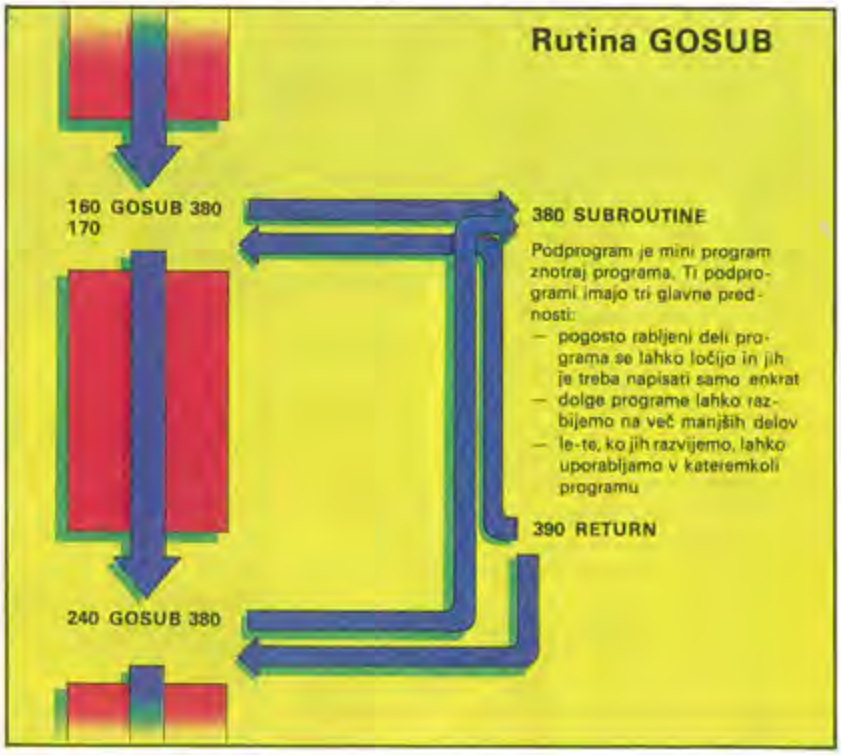
Programi znotraj programov: predstavljamo vam novo odliko basica, s katero bomo naše programe laže pisali

V prejšnjih poglavjih našega tečaja programiranja smo običajno s programi ravnali tako, da smo jih najprej vtipkali, nato pognali, jim vpisovali spremembe oz. izboljšave ter jih na kraju izbrisali iz pomnilnika z ukazom NEW, ker smo hoteli vtipkati nov program. Če smo hoteli star program ponovno pognati, smo ga morali spet vtipkati. Da se temu ponavljajočemu se in zamudnemu delu ognemo, vsebujejo vse verzije basica ukaz, s katerim program shranimo na navadno kaseto. Naslednji program se da shraniti na kaseto z uporabo preprostega ukaza SAVE, kateremu sledi ime datoteke. Ta program računa število stenskih ploščic, ki so potrebne za polaganje v sobi.

```
 10 REM TA PROGRAM RAČUNA ŠTEVILO
    PLOŠČIC
 20 REM KI SO POTREBNE ZA OBLOŽITEV
    SOBE
 30 PRINT "VSTAVI STRANICO PLOŠČICE
    V MM"
 40 INPUT A1
 45 LET T = 0
 50 REM VRSTICA 60 IZRAČUNA POVRŠINO
    PLOŠČICE
 60 LET A2  = A1*A1
 70 PRINT "VPIŠI ŠTEVILO STEN"
 80 REM W POSTAVI MEJO ZANKE
 90 INPUT W
100 FOR X = 1 TO W
110 PRINT "DOLŽINA STENE ŠT.";X;
    "V METRIH"
120 REM D JE MERA STENE
130 INPUT D
140 REM PRETVORI SE V MM
150 REM V PODPROGRAMU
160 GOSUB 380
170 REM VRSTICA 190 PRIPIŠE L
180 REM DOLŽINO STENE V MM
190 LET L = D2
200 PRINT "VIŠINA STENE ŠT.";X;
    "V METRIH"
210 REM VRSTICE 230 DO 250 PRIPIŠEJO H
220 REM VIŠINO STENE V MM
230 INPUT D
240 GOSUB 380
250 LET H = D2
260 REM VRSTICA 270 PRIPIŠE A3
    POVRŠINO STENE
270 LET A3=L*H
280 REM S (DELNI REZULTAT) JE POVRŠINA
    STENE
290 REM DELJENO S POVRŠINO PLOŠČICE
300 LET S = A3/A2
310 REM T (REZULTAT) SE PRIPIŠE DELNI
    REZULTAT
320 REM V VSAKEM KROGU ZANKE
330 LET T=T+S
340 NEXT X
350 REM IZPIŠE REZULTAT
360 PRINT T
370 END
380 LET D2=D*1000
390 RETURN
```

Ko program vtipkamo v računalnik, je treba samo uporabiti ukaz SAVE in program se shrani na kaseto. Še prej je seveda treba pripraviti kasetofon po navodilih, ki jih dobimo v priročniku za računalnik. Uporaba ukaza SAVE je zelo preprosta. Natipkamo SAVE in dodamo v narekovajih ime datoteke. Datoteka pomeni v računalniškem izrazju program ali množico podatkov, ki se lahko shrani ali ponovno uredi, ko je potrebno. Najprimernejše je uporabiti za ime datoteke kar ime, ki nas bo spomnilo na funkcijo programa. V našem primeru, ko računamo število ploščic za sobo, bomo datoteko imenovali kar »PLOŠČICE«. Ko je kasetofon nastavljen, vstavimo kaseto s praznim trakom, na katero želimo shraniti oz. posneti program.

Nekateri magnetofoni imajo tudi vhod (vtičnico) za daljinsko vodenje (remote control). V tem primeru lahko delovanje motorja (pogon, ustavitev) uravnavamo direktno z računalnikom. V nasprotnem primeru nastavimo kasetofon za snemanje in pritisnemo gumb



za odmor. Vtipkamo ukaz SAVE in ime datoteke. Sprostimo zavoro motorja s tem, da ponovno pritisnemo na gumb »odmor« ter takoj zatem še na tipko RETURN.

Če hočemo preskusiti, ali je bil program pravilno shranjen, izbrišemo računalnikov pomnilnik z ukazom NEW<CR>, previjemo kaseto ter poženemo kasetofon s tipko »igraj« (PLAY). Program bomo poklicali in prenesli v računalnik z ukazom LOAD, kateremu sledi ime želenega programa — datoteke. V našem primeru bomo natipkali LOAD »PLOŠČICE« in takoj pritisnili na tipko RETURN.

Ko se program prenese v računalnik, se pokaže na zaslonu sporočilo kot npr. READY ali OK, ki pravi, da je prenos končan. Nato izpišemo program z ukazom LIST na zaslon ter preskusimo, ali je enak tistemu, katerega smo vtipkali.

GOSUB

GOSUB je stavek, ki preusmeri potek programa k podprogramu. Podprogram je nekak ločen mini program ali program znotraj programa. V našem programu je podprogram zelo preprost. Rabi samo kot prikaz uporabe podprogramov in bi ga lahko mirno izpustili ter uporabili drugačno pot za računanje istega rezultata.

Naš program izračuna število ploščic, potrebnih za oblogo sten v sobi, tako da najprej ugotovi površino ene ploščice. Nato zahteva vpis podatkov za dolžino in višino vsake stene v metrih. Te mere sten v metrih nato spremeni v milimetre in izračuna površino vsake stene. Do števila potrebnih ploščic pride tako, da površino vsake stene deli s površino ene ploščice in nato te delne rezultate sešteje. Sprememba dolžin in višin sten v milimetre se opravlja v podprogramu, ki samo pomnoži višine oz. dolžine v metrih s 1000, da dobi istovredne mere v milimetrih.

Podprogrami imajo tri prednosti. Pogosto rabljeni deli programa se lahko ločijo in jih je treba napisati samo enkrat, ne glede na to, kolikokrat bodo potrebni. Dolgi in kompleksni programi se dajo razbiti na več delov, ki so razumljivejši in preprostejši. In ne navsezadnje lahko podprograme prenesemo tudi v druge programe, če so le njihove lastnosti ustrezne.

V našem programu sestavlja podprogram, ki se začne v vrstici 380, samo en stavek: LET D2 = D*1000. Le-ta množi mero stene D (dolžino ali višino) s 1000 in tako spremeni metre v milimetre. Rezultat je pripisan spremenljivki D2.

Ukaz, ki prisili program, da skoči na podprogram, je stavek GOSUB. Najprej se pokaže v vrstici 160. V vrstici 130 je dolžina stene pripisana spremenljivki D. Vrstica 160 prisili program, da skoči na podprogram, kjer se spremenljivki D2 pripiše vrednost spremenljivke D, pomnožene s 1000. Ukaz RETURN v vrstici 390 je potreben zato, da program vrne iz podprograma v glavni program. Podprogrami se vedno vrnejo v vrstico, ki sledi navodilu GOSUB, v našem primeru v vrstico 170.

Naslednji ukaz GOSUB se pokaže v vrstici 240 in se nanaša na isti podprogram. V tem primeru se podprogram z ukazom RETURN vrne v vrstico 250. Čeprav naš program uporablja samo en podprogram, bi jih lahko tudi več, če bi bilo potrebno. V vsakem primeru mora navodilo GOSUB vsebovati tudi številko vrstice, kjer se določeni podprogram začne. Opazimo lahko, da se stavek END pokaže v vrstici 370, to je pred podprogramom. Stavek END določa konec glavnega programa in v našem primeru preprečuje, da bi program, ko bi bil že izvršen, tekel še naprej čez podprogram.

Čeprav je ta program malo daljši od prejšnjih programov v tem tečaju, ni zato nič bolj zapleten. Poskusimo ga sedaj še enkrat proučiti tako, da bomo pregledali vsako vrstico posebej, da bomo vedeli, kaj se dogaja v njej. Razen ukaza GOSUB in podprogramov nam ta program predstavlja samo en nov pojem, in sicer daljša imena spremenljivk.

Pri proučevanju programa nam lahko pomaga tudi tabela z imeni spremenljivk na vrhu, v katero bi vpisali vrednosti spremenljivk na vsaki stopnji.

Vrstica 300: LET S = A3/A2 bo včasih dala kot rezultat decimalno število. Poskusimo pognati program in vstaviti mero ploščice 110 mm ter dolžino in višino ene stene 2,3 in 1,8 metra. Rezultat bi se moral glasiti 342,149 ploščice. Ker se loščice prodajajo samo kot cele enote, ta rezultat ni popolnoma primeren. V naslednjem poglavju si bomo pogledali, kako dobiti rezultate v obliki celih števil.

Naloge:

- Poglejte, kaj se zgodi, če vpišete za mero ploščice 0 mm. Ob koncu poteka bi morali dobiti sporočilo o napaki (ERROR MESSAGE). Zakaj to? Zakaj ne dobite podobnega sporočila o napaki, če vpišete za dolžino stene 0 metrov? Pomislite: deljenje z 0 ali množenje z 0 ni isto — preskusite to na žepnem računalniku!
- Program bo deloval pravilno samo, če gre za kvadratne ploščice. Poglejte, ali lahko spremenite vrstice od 30 do 60 tako, da bo program računal površino pravokotne ploščice. Nasvet: oglejte si, kako program izračuna površino stene.
- Dodajte v vrstici 355 stavek, ki bo povečal končno število ploščic za 5 % zaradi izmeta. Množenje števila s 105/100 da za 5 % večje število.

Posebnosti računalnikov

END — Tega ni pri računalnikih Spectrum, ZX 81, Oric-1 in Dragon 32. Uporabljati je treba stavek STOP.

LET — Vrstica 45 je potrebna pri Spectrumu in računalniku BBC. Pri večini drugih računalnikov se lahko izpusti.

GOSUB — Pokaže se v obliki dveh besed pri računalnikih Spectrum in ZX 81, čeprav je treba le enkrat pritisniti na tipko.

Novo leto v basicu

Predstavljamo vam nove ukaze za ravnanje s podatki in napisali bomo program, ki nam bo računal število dni do Novega leta

V tem programu bomo pregledali skoraj vse do sedaj opisane teme v tem tečaju programiranja in spoznali nekaj novih, zelo pomembnih stavkov v basicu. Naloga programa je, da nam izračuna število dni, ki ostajajo do konca leta.

Če pogledamo izpis programa, vidimo, da se začne s seznamom uporabljenih spremenljivk. To sicer ni nujno potrebno, je pa priporočljivo, ker naredi program veliko bolj pregleden in razumljiv, če ga čez dolgo časa znova pregledujemo. Nekatere oblike dovoljujejo, da imajo spremenljivke dolga imena, npr. DAN namesto ene same črke, kakor smo jo uporabljali doslej. Če imamo srečo in imamo v računalniku tak basic, da so dovoljena dolga imena, si bomo vsekakor izbrali imena s kakim pomenom: npr. DAN, MESEC. DANST je veliko boljše kakor A, X ali D. Če pa nimamo tako zmogljivega basica, nam tak seznam spremenljivk na začetku programa to pomanjkljivost ublaži.

Ko bomo program pognali, se bodo na zaslonu najprej izpisali stavki PRINT, ki se začnejo v vrstici 230. Ti stavki na kratko razložijo, kaj bo program delal, ter zahtevajo od uporabnika, da vtipka datum v podani obliki, uporabljajoč vejice za ločitev dneva, meseca in leta.

Prvi neznan stavek se bo pokazal v vrstici 300. To je stavek DIM (dimension). Uporablja se za določanje števila elementov polja X. Polje je kot vsaka druga spremenljivka, le da vsebuje več elementov. V vrstici 300 opredelimo spremenljivko X, kateri dodelimo 13 elementov. K temi polje in stavek DIM se bomo še povrnili v nadaljevanju tečaja, tam ju bomo tudi podrobneje opisali.

```
310 INPUT D, M$, L
```

Ta vrstica je navaden INPUT stavek, le da pričakuje tri vhode. D je številčna spremenljivka, ki bo vsebovala številko dneva v mesecu, L pa tekoče leto. M$ pa je nekaj drugega. Imenuje se spremenljivka niza ter se označuje z znakom $. Spremenljivka niza lahko sprejme znake iz tipkovnice, kakor tudi številčne znake. Če vpišemo npr. datum 23, JANUAR, 1983, bo spremenljivki D pripisana vrednost 23, spremenljivki niza M bo pripisan niz JANUAR, spremenljivki L pa bo pripisana vrednost 1983.

```
330 GOSUB 560 REM ŠT. RUTINE ZA MESEC
```

Ta vstavek pove programu, da naj skoči na podprogram z začetkom v vrstici 560. Ne prezrite, da je opomba REM vstavljena v isti vrstici. Če je prostor v vrstici, opombe REM ni vedno treba pisati v novo vrsto. Ta podprogram se uporabi samo enkrat in bi ga lahko torej mirno vključili tudi v glavni program. Ker je podprogram, je ločen od ostalega programa.

Ko je bil program prvič napisan, smo uporabljali za označbo meseca številko in ta del programa ni bil potreben. Kasneje smo se odločili, da bomo mesece vpisovali s celimi besedami. Ker pa je treba besedo spremeniti v ustrezno številko, smo ta dodatni program, ki sedaj sestavlja podprogram, napisali posebej. Edina sprememba, ki jo je bilo treba vnesti v glavni program, je bil dodatek enega stavka GOSUB. Ta podprogram lepo prikazuje, kako se lahko program razvija po delih, ki jih na koncu združimo skupaj s stavki GOSUB in RETURN. Podprogram sam je zelo preprost, vendar pojasnjuje, kako zelo je basic zmožen manipulirati z znakovnimi nizi. Vzemimo, da smo vpisali JANUAR kot mesec pri INPUT stavku. Spremenljivki M$ bo pripisan znakovni niz JANUAR. Prva vrstica podprograma se glasi:

```
560 IF M$ = "JANUAR" THEN LET M = 1
```

Ta stavek primerja vsebino spremenljivke niza M$ z znaki znotraj narekovajev. Če je vsebina enaka znakom (kakor v našem primeru), se vrstica nadaljuje do tam, kjer se spremenljivki M$ pripiše neka vrednost. Tu ne smemo zamenjati spremenljivke M s spremenljivko M$, ker se razlikujeta. Samo ena od teh dveh spremenljivk vsebuje spremenljivko niza, in sicer ta z znakom $! Po pregledu, ali je M$ enak nizu JANUAR, se program nadaljuje v naslednjo vrstico, tam ponovno pregleda vsebino M$ in jo primerja s FEBRUAR. Ker ni enaka, se spremenljivki M ne pripiše vrednost 2. Samo tam, kjer bo primerjava pravilna, se bo spremenljivki M pripisala neka vrednost. Ta vrednost pa bo enaka zaporedni številki meseca: 1 za JANUAR, 3 za MAREC itd.

Ko program pride do vrstice 680 oz. stavka RETURN, se vrne v glavni program, v vrstico, ki sledi stavku GOSUB. To je vrstica 340. Vsebuje stavek REM, vendar brez komentarja. Vložen je le zato, da razmakne program, ki postane tako bolj pregleden.

Vrstice 350 do 370 vsebujejo zanko FOR... NEXT. Ta povečuje vrednost spremenljivke I, začenši z 1 pa

do 13. Uporablja se za naslavljanje polja X v vrstici 360. No, to je treba pogledati malo podrobneje.

```
360 READ X(I)
```

READ je nov stavek in ga še nismo srečali. Stavek READ se vedno uporablja v povezavi s stavkom DATA. Stavek DATA za to vrstico je v vrstici 510:

```
DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
     31, 0
```

Te številke, razen zadnje, predstavljajo števila dnevov v vsakem mesecu v letu. Te dve vrstici sta istovredni trinajstim ločenim stavkom LET:

```
LET X(1) = 31
LET X(2) = 28
LET X(3) = 31
LET X(4) = 30
LET X(5) = 31
LET X(6) = 30
LET X(7) = 31
LET X(8) = 31
LET X(9) = 30
LET X(10) = 31
LET X(11) = 30
LET X(12) = 31
LET X(13) = 0
```

Zanka, začenši v vrstici 350, povečuje spremenljivko I od 1 do 13, zato smo lahko zamenjali X(1), X(2), X(3) itd. z X(I).

Preden se povrnemo k programu, si poglejmo ta dosti preprostejši mali program:

```
10 READ A, B, C,
20 LET D = A + B + C
30 PRINT D
40 DATA 5, 10, 20
```

Stavek READ, v vrstici 10 tega programa, prebere prvi del podatka stavka DATA iz vrstice 40 ter ga pripiše prvi spremenljivki. Z drugimi besedami, pripiše vrednost 5 spremenljivki A. Stavek READ nato prebere drugi del podatka in ga pripiše naslednji spremenljivki. Ta program naredi to, da velja: A = 5, B = 10 in C = 20. Te vrednosti nato sešteje ter pripiše rezultat spremenljivki D. Ta rezultat, 35, je nato izpisan s stavkom PRINT iz vrstice 30.

Povrnimo se sedaj k našemu programu. V prvem krogu zanke iz vrstice 350 se pripiše vrednost 1 spremenljivki I. Vrstica 360 je zato istovredna stavku READ X(1). Ustrezni podatek iz stavka DATA v vrstici 510 je 31 (prvi podatek). Torej se vrednost 31 pripiše spremenljivki X(1).

V drugem krogu se spremenljivki I pripiše vrednost 2, zato je tudi vrstica 360 istovredna stavku READ X(2). Naslednji podatek v vrstici DATA je 28. To pomeni, da se vrednost 28 pripiše spremenljivki X(2). Tako se vseh 13 oddelkov v polju X napolni s števili dni v vsakem mesecu: razen trinajstega oddelka, ki je napolnjen z vrednostjo 0.

```
390 GOSUB 750 REM RUTINA ZA
    PRESTOPNO LETO
```

Ta vrstica preusmeri program k podprogramu, ki ugotovi, ali je vstavljeno leto prestopno ali ne. Prestopno leto se opredeli kot leto, ki je deljivo s 4. Če gre za

```
100 REM SPISEK SPREMENLJIVK
110 REM
120 REM D = DANASNJI DATUM (DAN)
130 REM M$ = IME MESECA
140 REM L = LETO
150 REM I = INDEKS 1
160 REM X = POLJE ZA DNEVE V VSAKEM MESECU
170 REM R = STEVILO PREOSTALIH DNI
180 REM M = STEVILKA MESECA
190 REM K = INDEKS 2
200 REM Z = CELA VREDNOST OD L/4
210 REM
220 REM
230 PRINT "TA PROGRAM RACUNA"
240 PRINT "STEVILO PREOSTALIH"
250 PRINT "DNI DO NOVEGA LETA"
260 PRINT
270 PRINT "VSTAVI DANASNJI DAN, MESEC, LETO"
280 PRINT "NPR. 3,JUNIJ,1984"
290 PRINT
300 DIM X(13)
310 INPUT D,M$,L
320 REM
330 GO SUB 560: REM ST. RUTINE ZA MESEC
340 REM
350 FOR I = 1 TO 13
360 READ X(I)
370 NEXT I
380 REM
390 GO SUB 750: REM RUTINA ZA PRESTOPNA LETA
400 REM
410 LET R = X(M) - D
420 FOR K = M TO 11
430 LET M = M + 1
440 LET R = R + X(M)
450 NEXT K
460 REM
470 IF R = 1 THEN  GO TO 500
480 PRINT "PREOSTALO JE ";R;" DNI DO NOVEGA LETA"
490 GO TO 520
500 PRINT "PREOSTAL JE LE 1 DAN DO NOVEGA LETA"
510 DATA 31,28,31,30,31,30,31,31,30,31,30,31,0
520 STOP
530 REM
540 REM
550 REM
560 IF M$ = "JANUAR" THEN  LET M = 1
570 IF M$ = "FEBRUAR" THEN  LET M = 2
580 IF M$ = "MAREC" THEN  LET M = 3
590 IF M$ = "APRIL" THEN  LET M = 4
600 IF M$ = "MAJ" THEN  LET M = 5
610 IF M$ = "JUNIJ" THEN  LET M = 6
620 IF M$ = "JULIJ" THEN  LET M = 7
630 IF M$ = "AVGUST" THEN  LET M = 8
640 IF M$ = "SEPTEMBER" THEN  LET M = 9
650 IF M$ = "OKTOBER" THEN  LET M = 10
660 IF M$ = "NOVEMBER" THEN  LET M = 11
670 IF M$ = "DECEMBER" THEN  LET M = 12
680 RETURN
690 REM
700 REM
710 REM
720 REM OPOZORILO - TA PODPROGRAM NE PREGLEDA
730 REM POGOJA ZA PRESTOPNO LETO PRI
740 REM KONCU VSAKEGA STOLETJA
750 LET L = L/4
760 LET Z = INT (L)
770 IF L - Z = 0 THEN  GO TO 790
780 RETURN
790 LET X(2) = X(2) + 1
800 RETURN
```

letnico stoletja, mora biti deljiva tudi s 400, da bo veljala za prestopno leto. Zaradi preprostosti smo vstavili samo pogoj o deljivosti s 4.

```
750 LET L = L/4
760 LET Z = INT(L)
770 IF L – Z = 0 THEN GOTO 790
780 RETURN
790 LET X(2) = X(2) + 1
800 RETURN
```

Vrstica 750 pripiše spremenljivki L staro vrednost spremenljivke L, deljeno s 4. Nova vrednost L bo celo število, če je leto deljivo s 4. V nasprotnem primeru bo L decimalno število.

Vrstica 760 uporablja funkcijo INT, da poišče celo vrednost spremenljivke L. INT (integer) pomeni celo število. Medtem ko pri celih številih INT nima učinka, pa zato necele številke zaokroži na najbližje manjše celo število. Število, ki ga je treba zaokrožiti, se napiše med oklepaje, ki sledijo INT. Med oklepaje se lahko napiše tudi samo ime spremenljivke. Tako bo stavek LET Z = INT (496,25) pripisal spremenljivki Z vrednost 496.

Vrstica 770 odšteje Z od L ter preskusi, ali je rezultat enak 0. Če je to res, pomeni, da je leto prestopno (ker ni bilo decimalnega dela nove spremenljivke L). V tem primeru skoči program zaradi ukaza GOTO v vrstico 790. Vrstica 790 prišteje 1 drugemu podatku v polju (drugi podatek je bil 28, to je število dni v navadnem februarju).

Če je rezultat v vrstici 770 različen od 0, se spremenljivka X(2) ne spremeni, program pa se s stavkom RETURN povrne iz podprograma v glavni program, v vrstico 400.

Vrstica 400 vsebuje naslednji stavek REM, ki rabi za razmaknitev programa. Naslednja vrstica, v kateri se nekaj res zgodi, je 410: tam je R spremenljivka, ki vsebuje število dni, preostalih do novega leta. Tu se ji najprej pripiše število vpisanih dni (datum). Če smo vpisali npr. 12, FEBRUAR, 1983, bo spremenljivka D enaka 12 in M enaka 2. Torej bo X(M) enako kot X(2), drugi podatek v polju X pa je 28 (ker leto 1983 ni prestopno leto, se X(2) ne prišteje 1). Zato bo spremenljivki R pripisana vrednost (28 – 12), to je 16, to pomeni število preostalih dni v tekočem mesecu februarju.

V vrstici 420 se začenja naslednja zanka. Namen te zanke je povečevati vrednosti spremenljivke M. Zakaj smo tu uporabili stavek FOR K = M TO 11 namesto stavka FOR K = M TO 12? Če je bil M enak 2, ker smo vstavili mesec februar, bo vrstica 430 povečala vrednost spremenljivke M na 3. Vrstica 440 nato pripiše spremenljivki R novo vrednost tako, da ji prišteje vrednost spremenljivke X(M). Le-ta je sedaj enaka X(3), ker je bil M povečan za 1. Vrednost spremenljivke X(3) je 31, to je število dni v marcu. Torej pripiše vrstica 440 novo vrednost spremenljivki R, in sicer 16 + 31 (16 je rezultat 28 – 12). V naslednjem krogu se M poveča na 4, število dni v aprilu, X(4), se pa prišteje stari vrednosti spremenljivke R. Vrednost spremenljivke R je torej že enaka 16 + 31 + 30.

Vrednost spremenljivke X(12) = 31 se še zadnjič prišteje vrednosti spremenljivke R v zadnjem krogu zanke, ko je K = 11.

Kaj se zgodi v zanki, če je vhodni mesec december? Zaradi različnosti računalnikov bodo nekateri zanko kar preskočili, medtem ko bodo nekateri naredili še en krog v zanki, kjer bo vrednost spremenljivke X(13) prišteta vrednosti spremenljivke R. Da pa bi bil rezultat kljub temu pravilen, je bila spremenljivki X(13) pripisana vrednost 0.

```
470 IF R = 1 THEN GOTO 500
```

Ta vrstica samo preskusi, ali je samo en dan preostal do novega leta, da lahko rezultat slovnično pravilno izpiše na zaslon. Če pa R ni enak 1, je preostalo več dni do novega leta, zatorej bo stavek PRINT v vrstici 480 slovnično pravilen.

No, to bi bilo vse o tem programu. Zvrst basica, katero smo mi uporabili, bi morala ustrezati večini računalnikov (glej rubriko Posebnosti računalnikov), razen mogoče zanke, ki ugotavlja prestopna leta. Basic je zelo neskladen pri uporabi stavka LET. Če vrstice, kakor npr. IF M$ = »SEPTEMBER« THEN LET M = 9, na vašem računalniku ne delujejo, se lahko podprogram spremeni v obliko:

```
560 IF M$ = "JANUAR" THEN GOSUB 900
570 IF M$ = "FEBRUAR" THEN GOSUB 910
580 IF M$ = "MAREC" THEN GOSUB 920
.
.
.
900 LET M = 1
905 RETURN
910 LET M = 2
915 RETURN
920 LET M = 3
925 RETURN
(...itd.)
```

Ta rešitev potrebuje več prostora in je manj pregledna zaradi vseh teh stavkov GOSUB in RETURN. Vendar lepo izpričuje, kako lahko do rešitve problema pridemo po več poteh.

Posebnosti računalnikov

READ DATA

Teh ukazov ne dobite pri ZX 81, zato izbrišete vrstice 300, 350–370 in 510. Dodajte:

```
10 DIM X(13)
20 FOR K = 1 TO 13
30 PRINT "VSTAVI PODATEK ŠTEVILKA"; K
40 INPUT X(K)
50 NEXT K
60 STOP
```

Poženite program in vstavite podatke. Izbrišite vrstice 10–60 ter shranite program, ki obdrži vrednosti polja. Po shranitvi programa (LOAD) v prihodnosti uporabljajte GOTO 100 raje kot RUN, zaradi ohranitve spremenljivk.

INPUT

Računalnik ZX 81 zahteva ukaz INPUT za vsak podatek, zato dodajte vrstico
285 PRINT "KO SI PRIPRAVLJEN" in
310 PRINT "VSTAVI DAN"
312 INPUT D
ter podobno za mesec in leto.

REM

Pri računalnikih BBC Micro, Commodore in Vic-20 mora stati pred stavkom REM na koncu programske vrstice dvopiče (:).

Izziv elementom

Naslovljive spremenljivke lahko, drugače kakor njihove dvojnice, vsebujejo katerokoli število elementov

V prejšnjem programu, kjer smo računali število preostalih dni do novega leta, smo srečali spremenljivko, kateri pravimo naslovljiva (indeksirana). Le-ta se razlikuje od navadnih ali preprostih spremenljivk v tem, da vsebuje poljubno število oddelkov ali elementov. Navadne spremenljivke lahko označujemo s po dvema črkama, ali s črko in številko od 0 do 9 (nekatere zvrsti basica dovoljujejo tudi daljša imena spremenljivk). A, B, B1, C3 in R2 so same navadne spremenljivke. Naslovljive spremenljivke pa so takšnele: A(6), B(12) ali X(20). Naslov je številka v oklepaju. Ti primeri bi se brali takole: »A od šest«, »B od dvanajst« in »X od dvajset«.

Če si predstavljamo navadno spremenljivko kot škatlo z imenom na vrhu, si lahko naslovljivo spremenljivko predstavljamo kot škatlo, ki vsebuje določeno število elementov oz. oddelkov. Če potrebujemo spremenljivko z 12 elementi, jo lahko naredimo z uporabo stavka DIM, npr. DIM A(12). Uporabijo se lahko vse črke v abecedi.

Navedenim spremenljivkam se pripisujejo vrednosti z uporabo stavkov LET ali INPUT, npr. LET A = 35, LET B1 = 356 ali INPUT C3. Enako se lahko pripišejo vrednosti elementom naslovljive spremenljivke. Pa si poglejmo, kako bi pripisali vrednosti naslovljivemu polju. Polje je alternativno ime za vrsto naslovljivih spremenljivk. Na primer:

```
10 DIM A(5)
```

Naredi naslovljivo spremenljivko s petimi elementi. Sedaj bomo pripisali vrednost vsakemu od teh elementov:

```
20 LET A(1) = 5
30 LET A(2) = 10
40 LET A(3) = 15
50 LET A(4) = 20
60 LET A(5) = 100
```

Da ugotovimo, kako se te spremenljivke razlikujejo od navadnih, bomo pripisali vrednosti nekaj navadnim spremenljivkam:

```
70 LET X = 5
80 LET Y = 6
90 LET Z = 7
```

Vpišimo sedaj vse to v računalnik in preglejmo vsebino vsake spremenljivke s stavkom PRINT. Veliko stavkov v basicu lahko nastopa tudi kot ukaz. Ko smo to vpisali, naredimo še pregled, uporabljajoč ukaz LIST, nato poženemo program z ukazom RUN. Sedaj lahko napišemo PRINT X< CR>. Na zaslonu se bo takoj pokazala številka 5. Nato napišemo PRINT Y. Računalnik bo odgovoril na ta ukaz s tem, da bo na zaslonu izpisal številko 6. Če želimo pregledati elemente v naslovljivi spremenljivki, napišemo PRINT A(1), da ugotovimo vrednost prvega elementa v polju. Računalnik bo na zaslonu izpisal številko 5. Poskusimo ugotoviti še vrednosti A(3) in A(5).

Pomembna razlika med navadno in naslovljivo spremenljivko je ta, da je lahko naslov sam spremenljivka. Da bomo videli, kaj to pomeni, napišemo PRINT A(X). Na zaslonu se bo pokazala številka 100. Zakaj? Če pregledamo vsebino programa, vidimo, da smo spremenljivki X pripisali vrednost 5. Se pravi, da je A(X) istovreden A (vrednost spremenljivke X) in to je istovredno A(5). Kakšno vrednost lahko pričakujemo, če vpišemo PRINT A(Y—X)? Preden si bomo to pogledali, poskusite odgovoriti na vprašanja.

Pripisovanje vrednosti

Če imamo samo nekaj navadnih spremenljivk, jim bomo najpreprosteje pripisali vrednosti s stavkom LET. Naslovljive spremenljivke lahko imajo tudi veliko število elementov v polju, zato si poglejmo, kakšne so metode za pripisovanje vrednosti tem elementom:

```
10 DIM A(5)
20 PRINT "VPIŠI SPREMENLJIVKE"
30 INPUT A(1)
40 INPUT A(2)
50 INPUT A(3)
60 INPUT A(4)
70 INPUT A(5)
```

Ta metoda je utrudljiva zaradi dolgega tipkanja, čeprav bo zanesljivo delovala. Če natančno poznamo število vseh spremenljivk (v našem primeru 5), bo preprostejše, če uporabimo zanko FOR...NEXT, npr:

```
10 DIM A(5)
20 FOR X = 1 TO 5
30 INPUT A(X)
40 NEXT X
```

Ta program bo pričakoval pet vrednosti, ki jih bo treba vtipkati med delovanjem programa. Tipko RETURN

je treba pritisniti po vsaki vpisani vrednosti. Če poznamo pred tem vse vrednosti spremenljivk, je še lažje, če uporabimo stavek READ v povezavi s stavkom DATA:

```
10 DIM A(5)
20 FOR X = 1 TO 5
30 READ A(X)
40 NEXT X
50 DATA 5, 10, 15, 20, 100
```

Preskusimo ta kratki program in preglejmo vsebino polja z uporabo ukaza PRINT, to je, uporabimo stavek PRINT po izvršitvi programa. Na primer PRINT A(1) < CR > in PRINT A(5). Lako pa tudi dodamo nekaj vrstic v program, da bo ta avtomatsko izpisal vrednosti vseh elementov polja:

```
60 FOR L = 1 TO 5
70 PRINT A(L)
80 NEXT L
90 END
```

Poženimo ta program z ukazom RUN in preglejmo izpis na zaslonu. Nato spremenimo vrstico 50 s tem, da uporabimo pet drugih podatkov. Ne smemo pozabiti, da moramo podatke v stavku DATA ločiti z vejico; te pa ne sme biti pred prvim ali za zadnjim podatkom.

Najpreprostejši način pripisovanja vrednosti je uporaba stavkov DATA in READ. Če bodo spremenljivke vsakič, ko bomo na novo pognali program, drugačne, je ugodnejše, da uporabimo stavek INPUT znotraj zanke FOR...NEXT. Če je maksimalno število elementov v polju znano, lahko uporabimo to število za zgornjo mejo stavka FOR.

Sedaj bomo uporabili vse svoje pridobljeno znanje in napisali kratek, vendar zmogljiv program. Vzemimo, da želimo nekaj številk urediti v rastočem zaporedju. Preden bomo začeli pisati program, si bomo naredili logični načrt rešitve. Ko bo pot do rešitve jasna, si jo bomo napisali korak za korakom v kratkih in jedrnatih stavkih.

Recimo, da začnemo s petimi številkami: 4, 9, 2, 8, 3. Urediti te številke v rastočem zaporedju res ni velik problem. Natančno pregledamo vrstico, poiščemo najmanjšo številko, jo pomaknemo na levo in nato postopek ponovimo z drugimi številkami.

Računalnik pa potrebuje zelo natančna navodila, zato bomo morali premisliti, kateri koraki so potrebni. Tu bomo opisali en mogoč način obdelave: primerjaj prvo številko z drugo, in če je prva večja od druge, ju zamenjaj. Če pa je prva številka manjša od druge, ju pusti na istih mestih.

Primerjaj drugo številko s tretjo. Če je druga manjša od tretje, ju pusti na istih mestih.

Ponovi ta postopek do zadnjega para številk.

Če ni bilo nobenih zamenjav, so številke že urejene. Če so bile zamenjave, se vrni na začetek in ponovi postopek še enkrat.

Če pogledamo ta postopek, vidimo, da resnično lahko uredimo vsako skupino številk v rastočem zaporedju. Poglejmo, kaj se zgodi z našimi številkami, če primerjamo vsak par številk:

```
4 9 2 8 3
4 2 9 8 3
4 2 9 9 3
4 2 8 3 9
```

Vsi pari so bili primerjani ter zamenjani, kjer je bilo potrebno. Ker je bila potrebna najmanj ena zamenjava, se bomo vrnili na začetek in postopek ponovili:

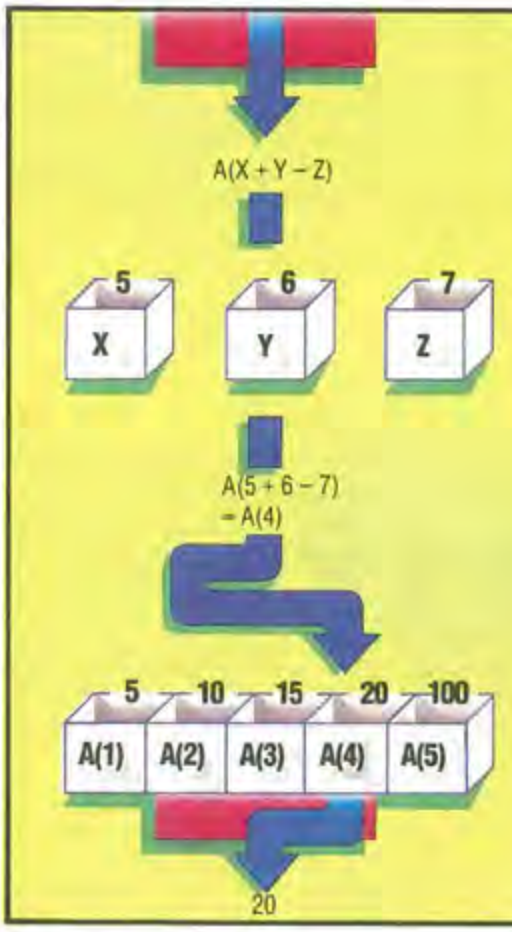
```
4 2 8 3 9
2 4 8 3 9
2 4 3 8 9
2 4 3 8 9
```

Še vedno so bile potrebne zamenjave, zato bomo postopek ponovili še enkrat:

```
2 4 3 8 9
2 3 4 8 9
2 3 4 8 9
```

Tokrat ni bilo nobene zamenjave, se pravi, da je vsaka številka manjša od svoje desne sosede. Številke so urejene po velikosti in postopek je končan.

Z naslovljivimi spremenljivkami zlahka prevedemo to rutino v basic, ker je naslov sam lahko spremenljivka. Če so bile naše številke vrednosti v polju, sledi: A(1) = 4, A(2) = 9, A(3) = 2, A(4) = 8 in A(5) = 3. Če ima X vrednost 1, bo A(X) vsebina od A(1), ker je enako 4. A(X + 1) bo vsebina od A(2), kar je enako 9, itd.



Naslovljive spremenljivke

Naslovljive spremenljivke (spremenljivke z več oddelki v škatli) povečujejo zmogljivost basica. Tu ima spremenljivka A naslov X + Y − Z. Vsaka od teh je spremenljivka, katere vrednost je prikazana znotraj male škatle. X ima vrednost 5, Y je 6 in Z je 7. X + Y − Z je istovredno s 5 + 6 − 7, kar znese 4. A(4) je četrti element v polju. Njegova vrednost je enaka 20. Stavek PRINT A(X + + Y − Z) bo na zaslonu izpisal rezultat 20.

Poglejmo sedaj program in poskusimo ugotoviti, kaj se v njem dogaja. Vrstica 20 pripiše vrednost spremenljivki N, ki je enaka številu številk, ki jih želimo urediti. Recimo, da želimo urediti 5 številk. Ko bomo program pognali, bomo vtipkali številko 5 in pritisnili na tipko RETURN.

Vrstica 30 je stavek DIM. Če je N enak 5, bo pripisal velikost polja na 5. Ta vrstica je istovredna z DIM A(5).

Vrstice 40 do 60 vsebujejo zanko FOR... NEXT, ki nam dovoljuje vtipkati 5 številk. Večina zvrsti basica obvesti uporabnika o tem z vprašanjem na zaslonu. Po vsaki vtipkani številki je potrebno pritisniti tipko RETURN. Številke so lahko tudi večmestne ali decimalne. Vrstica 90 pripiše vrednost spremenljivke S. Ta spremenljivka se uporablja kot kontrolni znak. Kasneje v programu se pogleda, ali je S enak 1 ali ne. Spremenljivki S se pripiše vrednost 1 le, če se dve številki zamenjata, kakor bomo videli v vrstici 240. Vlogo kontrolnega znaka si bomo pogledali še v nadaljevanju tečaja.

Vrstica 100 postavlja meje za zanko; v tem primeru od 1 do 4 (ker je N enak 5, je N – 1 enako 4). V prvem krogu zanke je L enak 1, zato bo tudi A(L) v vrstici 110 enak A(1) ali prvemu elementu v polju, in A(L + 1) bo enak A(2) ali drugemu elementu polja. V naslednjem krogu bo L povečan na 2, zatorej bo A(L) enak A(2) in A(L + 1) enak A(3). Vrstica 110 preveri, ali je A(L) večji od številke takoj za njo na desno v polju. Znak za »večji od« je >.

Če je prva številka večja od naslednje, skoči program v podprogram, tam se številki zamenjata. Če prva številka ni večja od druge, program ne skoči podprogram, temveč se nadaljuje v naslednjo vrstico, kjer stoji stavek NEXT L. Ko se zanka izvrši štirikrat, se program pomakne v vrstico 130, tam preskusi vrednosti kontrolnega znaka. Če se mu je v podprogramu pripisala vrednost 1, se program vrne v vrstico 90, da še enkrat ponovi postopek primerjav. Če S ni enak 1, pomeni to, da ni bilo nobene zamenjave, iz tega sledi, da so številke že urejene. Ostanek programa samo izpiše rezultat.

```
 10 PRINT "KOLIKO ŠTEVILK HOČEŠ
    UREDITI?"
 20 INPUT N
 30 DIM A(N)
 40 FOR X = 1 TO N
 45 PRINT "NASLEDNJA ŠTEVILKA"
 50 INPUT A(X)
 60 NEXT X
 70 REM
 80 REM PODPROGRAM "UREDI"
 90 LET S = 0
100 FOR L = 1 TO N – 1
110 IF A(L)>A(L + 1) THEN GOSUB 200
120 NEXT L
130 IF S = 1 THEN 90
140 FOR X = 1 TO N
150 PRINT "A(";X;") =";A(X)
160 NEXT X
170 END
180 REM
190 REM
200 REM PODPROGRAM "ZAMENJAJ"
210 LET T = A(L)
220 LET A(L) = A(L + 1)
230 LET A(L + 1) = T
240 LET S = 1
250 RETURN
```

Podprogram za zamenjavo številk uporablja začasno spremenljivko, kateri pripiše eno od številk, ki jih bo zamenjal. Ko zamenja številki (vrstice 210, 220, 230), pripiše kontrolnemu znaku S vrednost 1, nato se program s stavkom RETURN vrne v glavni program.

Naloge:

■ Razširite program tako, da bo računal povprečno vrednost vstavljenih številk. Povprečje je enako vsoti vseh številk, deljeni s številom vseh številk. To najlepše naredimo tako, da vstavimo stavek GOSUB tik pred stavkom END v vrstici 170. Podprogram mora prebrati vse elemente polja ter jih prišteti neki spremenljivki. Ko so vsi elementi sešteti, se vsota deli s številom elementov v polju. Do vsote najlaže pridemo tako, da uporabimo število elementov kot gornjo mejo zanke FOR...NEXT.

■ Spremenite eno vrstico v programu tako, da bodo številke urejene od največje proti najmanjši.

■ Ta naloga je namenjena predvsem lastnikom računalnika TI99/4A, ki nima spremenljivk za naslove v naslovljivih spremenljivkah. Vendar basic sprejema stavke kot npr. DIM A(12). Spremenite program tako, da bo stavek INPUT pričakoval natančno število številk, ki jih je treba vpisati, recimo 12. S tem se boste ognili problemu spremenljivk v vlogi naslovov. Prav tako je treba spremeniti vrstici 100 in 110. Iz istega razloga je treba spremeniti podprogram za zamenjavo številk, ker sicer ne bo delal.

■ In sedaj miselna naloga. Naš način urejanja številk prav gotovo ni edini. Premislite in poiščite alternativno rešitev.

Posebnosti računalnikov

IF... THEN — Če želite ta program izvajati na ZX 81 ali Spectrumu, se mora vrstica 130 glasiti: 130 IF S = 1 THEN GOTO 90

END — Tega stavka ni pri računalniku ZX 81 in Spectrumu, zato je treba uporabiti stavek STOP.

DIM A(N) — Pri večini računalnikov beseda LET v stavkih za pripisovanje, npr. 90 LET S = 0, ni obvezna, pač pa pri ZX 81 in Spectrumu. Računalnik Lynx jo doda programu avtomatsko.

Uredimo si programe

Sestavili bomo program za preureditev informacij tako, da bomo uporabili vdelane funkcije

Naš novi program nam bo razjasnil, kako lahko razmeroma kompleksne programe razkosamo v preproste podprograme, ki jih lahko pišemo in pregledujemo ločeno.

Uporaba podprogramov nam poleg prednosti, da jih lahko ločeno pregledujemo, prinaša tudi ugodnost, da razvijemo program po ločeni poti. Obstaja veliko prijemov za pisanje programov v basicu. Najpogostejši je »poskušanje z napakami«: pri tem načinu pisanja programov se kar usedemo k računalniku in začnemo vpisovati vrstice brez premisleka, kaj je treba narediti, da bo program deloval. Ta metoda vodi do slabo zgrajenih programov, ki pogosto ne bodo delovali ob prvem poskusu. Če zgradba programa ni jasna, je tudi težko poiskati napake.

Veliko ugodnejši način pisanja programov je ta, da se usedemo in najprej na papirju izdelamo zgradbo programa, pri tem pa vsako stopnjo dopolnjujemo in izboljšujemo, dokler ne dobimo natančnega programa, ki je pripravljen za vpis v računalnik. Diagram poteka nam je pri tem tudi v pomoč. Pa poglejmo, kako se to dela.

Problem: napisati moramo program, v katerega bomo vpisali neko število imen, ki jim bodo sledili priimki. Sedaj je treba obrniti vrstni red tako, da bo na prvem mestu priimek, nato vejica in prazen prostor ter nazadnje ime. Računalnik bo potem uredil imena po abecedi ter jih izpisal na zaslon.

Npr. če vpišemo imeni TOMAŽ NOVAK in ANTON STANOVNIK (v tem vrstnem redu), bo program izpisal:

NOVAK, TOMAŽ
STANOVNIK, ANTON

Preden sploh poskusimo napisati tak program, si bomo čisto na splošno zapisali želene vhode in izhode:

korak 1
Za vpis imena v poljubnem zaporedju, najprej ime, nato priimek. Za izpis imena po abecednem redu, najprej priimek, nato ime.

To zelo jasno pove, kaj želimo narediti. To je zelo pomemben korak k pravilno zgrajenemu programu. Za naslednji korak bomo razširili dele iz prvega koraka, vendar tako, da bo program še vedno deloval. Na tej ravni še ne smemo preveč razčlenjevati, potrebna je le malo širša razlaga:

korak 2
Ugotovi število imen, ki jih je treba vpisati.
Vpiši imena.
Obrni imena.
Uredi imena.
Izpiši imena.

Poglejmo ta korak, da se prepričamo, ali bo pravilno deloval. Ali je kaj narobe z njim? So kje kaki miselni preskoki? Če je vse v redu, pojdimo na naslednjo raven členitve.

Stopnje, do katerih smo prišli v drugem koraku, so dovolj majhne in preproste, da jih lahko zapišemo ločeno, kot majhne podprograme. Vsakemu podprogramu bomo dali ime, da jih bomo laže ločili. Podprogram 1, v katerem moramo poiskati število vseh imen, katera bomo vpisali, lahko označimo kot UGOŠT (ugotovi število). Podprogram 2, v katerem bomo vpisovali imena, lahko poimenujemo VSTAVI. Podprogram 3, za obračanje imen, OBRNI. Podprogram 4, za urejanje imen, UREDI. In nazadnje podprogram 5, za izpis imen, IZPIS.

korak 3.1. UGOŠT
Zahtevaj od uporabnika število vseh imen.
Dobi ustrezno številko N.
Uporabi N pri definiciji polja nizov.
korak 3.2. VSTAVI
Če je število vstavljenih imen manjše od N, zahtevaj od uporabnika dodatna imena.
Pripiši imena polju nizov.
korak 3.3. OBRNI
Ugotovi dolžino niza (imena).
Poišči presledek v nizu.
Pripiši znake v nizu do presledka začasni spremenljivki niza.
Pripiši originalnemu nizu drugo in potem še prvo začasno spremenljivko niza.
Dodaj vejico in presledek drugi začasni spremenljivki niza.
Pripiši originalnemu nizu drugo, kateri bo sledila prva začasna spremenljivka niza.
korak 3.4. UREDI
Primerjaj prvi podatek v polju z drugim.
Če je prvi podatek v abecedi zadaj, ju zamenjaj.
Primerjaj drugi podatek v polju s tretjim.
Zamenjaj ju, če je potrebno.
Ponovi, dokler ne primerjaš še zadnjega para.
Pojdi nazaj na začetek polja in znova začni pregledovati, dokler ne zmanjka zamenjav.

Opomba: Ta rutina je natančno ista, kot smo jo uporabili v prejšnjem poglavju tečaja. Podprogram ZAMENJUJ lahko s popravki uporabimo v podprogramu UREDI.

korak 3.5. IZPIS
Izpisuj podatke v polje, dokler vseh ne izpišeš.

Vsak korak, potreben za delovanje programa, je bil zdaj obdelan dovolj nadrobno. Podprogram UREDI je bil opisan skopo, ker smo ga že dodobra spoznali v prejšnjem poglavju. Opis podprograma ZAMENJAJ, ki je del tega podprograma, smo popolnoma izpustili. Sedaj pa poglejmo, kako lahko je te korake prevesti v basic.

korak 4
1. UGOŠT

Tri vrstice iz koraka 3.1. se dajo prevesti naravnost v basic. Uporabnik je seznanjen s stavkom PRINT, število se dobi s stavkom INPUT in polje se definira s stavkom DIM:

```
PRINT "KOLIKO IMEN ŽELIŠ VSTAVITI"
INPUT N
DIM A$(N)
RETURN
```

Spremenljivka N vsebuje maksimalno število vnesenih imen. Stavek DIM dimenzionira polje nizov. Spremenljivke niza vsebujejo nize, sestavljene iz alfanumeričnih znakov namesto samo iz numeričnih. Spremenljivka niza se vedno konča z znakom $. Samo A$ bi lahko vseboval le en niz. DIM A$(N) pa ustvari polje, ki lahko vsebuje »N« nizov. Naslovljive spremenljivke so bile opisane že prej.

Stavek RETURN premesti program nazaj v glavni program, in sicer v vrstico, ki sledi pozivu podprograma. Vrednosti, pripisane spremenljivkam v podprogramu, se tudi prenesejo nazaj ter jih lahko uporabimo kjerkoli v programu, tudi v podprogramu.

2. VSTAVI

Dokler je število vnesenih imen manjše od N, mora biti uporabnik obveščen, da mora ponovno vstaviti ime, ki bo prav tako del polja nizov. Najlaže bomo to naredili z zanko FOR...NEXT; vemo, da bo prvo ime prvi podatek v polju, in da bo N-to ime N-ti podatek v polju, torej:

```
FOR X = 1 TO N
PRINT "VSTAVI IME"
INPUT A$(X)
NEXT X
RETURN
```

To bo zadostovalo za vpis vseh imen v polje. Verjetno se že sprašujete, kaj se bo zgodilo, ko bo potrebno obrniti vrstni red imen in priimkov v podprogramu OBRNI. Vsak element (ime) v polju je treba poklicati iz polja, ga obrniti ter ga vstaviti nazaj v polje. Namesto da bi zapletali in daljšali program s tem, bomo rajši poklicali podprogram OBRNI, kar naravnost iz podprograma VNESI, po vpisu vsakega imena. Tako lahko ime obrnemo, preden ga sploh vstavimo v polje. To dosežemo samo z eno vrstico, ko jo vnesemo v podprogram VNESI:

```
FOR X = 1 TO N
PRINT "VSTAVI IME"
INPUT A$(X)
GOSUB [OBRNI]
NEXT X
RETURN
```

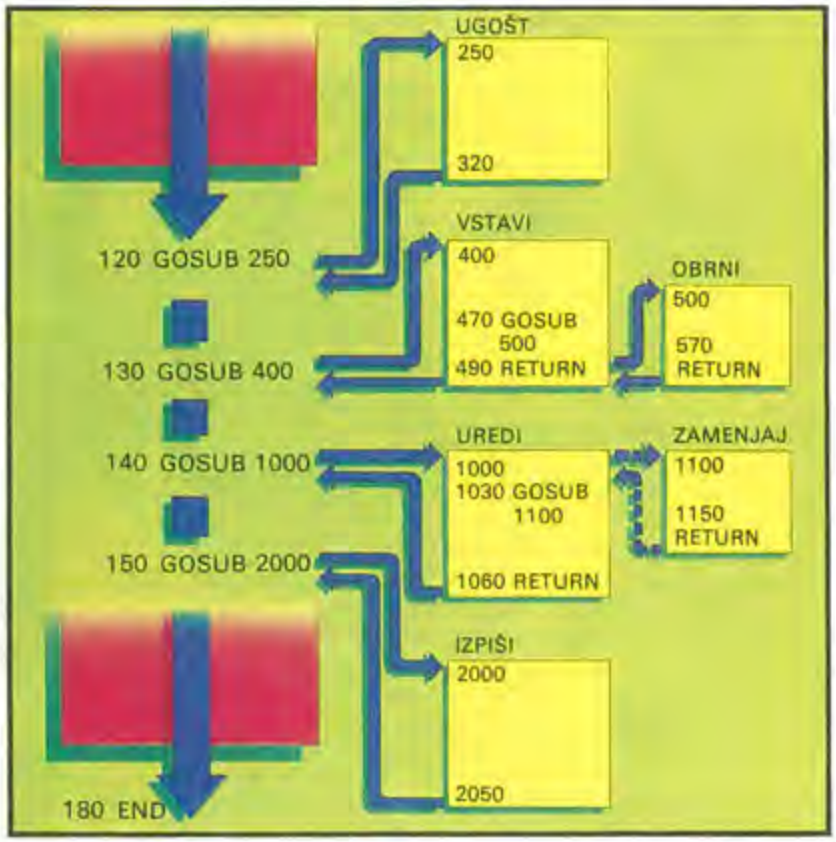
Vsa imena v polju bodo sedaj že obrnjena (na pravem mestu priimek, nato ime), zato jih lahko začnemo urejati po abecedi.

3. OBRNI

Da lahko obrnemo ime, moramo vedeti, kje stoji presledek, ki loči ime od priimka. Ko vemo, kje je presledek, lahko uporabimo različne funkcije, ki bodo odvzele del niza ter ga pripisale drugemu nizu. Funkcije v basicu so ukazi, ki opravljajo že opredeljene operacije z vrednostjo, ki sledi imenu funkcije. Ta del je vedno v oklepajih. Veliko funkcij je že vdelanih, vendar lahko opredelimo tudi svoje. Tipična vdelana funkcija je SQR ( ). Ta funkcija poišče kvadratni koren števila znotraj oklepajev. Npr: LET A = SQR(9): PRINT A bo izpisal 3.

Podprogram OBRNI uporablja funkcije LEN (za ugotovitev dolžine niza), INSTR (za iskanje pozicije praznega prostora), LEFT$ (za prenos določenega števila znakov z leve strani niza) in RIGHT$ (za prenos določenega števila znakov z desne strani niza). Na tem mestu ne bomo nadrobno opisovali, kako te funkcije delujejo. To bo sledilo v naslednjem poglavju.

Programi znotraj programa
Glavni program je tokrat zelo kratek. Pravo delo se opravlja v podprogramih (v basicu se imenujejo tudi podrutine). Vsak korak, potreben za delovanje programa, se loči in napiše kot kratek podprogram. Ti podprogrami so pretežno združeni z glavnim programom. Vsakič, ko program naleti na stavek GOSUB, skoči na ustrezno podprogramsko vrstico, kjer se ta del programa v celoti izvrši.
Konec podprograma je označen s stavkom RETURN. Ko ga program doseže, se vrne v vrstico, ki neposredno sledi stavku GOSUB.
Podprogrami se lahko med seboj tudi mešajo. Podprogram VPIŠI pokliče včasih drugi podprogram, imenovan OBRNI, podprogram UREDI včasih pokliče podprogram, imenovan ZAMENJAJ.
Z razkosanjem programa na ločene podprograme, ki jih povezuje glavni program, veliko laže razvijemo in pregledamo programe.



4. UREDI

Podprogram UREDI in podprogram ZAMENJAJ, katerega kličemo iz njega, sta zelo podobna rutinam, ki smo jih uporabili v prejšnjem programu.

5. IZPIS

Vsebina tega podprograma je naslednja:

```
FOR Q = 1 TO N
PRINT A$(Q)
NEXT Q
RETURN
```

Sedaj nam preostane le to, da napišemo glavni program. Ta je zelo preprost:

```
REM GLAVNI PROGRAM
GOSUB [UGOŠT]
GOSUB [VPIŠI]
GOSUB [UREDI]
GOSUB [IZPIŠI]
END
```

Imena podprogramov smo obdali z oglatimi oklepaji. Nekatere verzije basica lahko kličejo podprograme z imeni, nekatere pa zahtevajo številko vrstice, kjer se podprogram začne. Ko se program dejansko vpisuje v računalnik, vlagamo namesto imen določene številke vrstic, pri tem dodamo tudi še potrebne stavke REM in PRINT s sporočili.

Naloge:

Sedaj, ko smo pregledali skoraj vse pomembnejše lastnosti basica, je čas, da preskusimo svoje znanje s temi nalogami. Gibljejo se po zahtevnosti od zelo lahkih do srednje težkih.

■ **Spremenljivke:** Obkrožite v spodnjem izrazu vsa pravilna imena numeričnih spremenljivk in naredite križec čez nepravilna imena numeričnih spremenljivk. Pravilna imena spremenljivk niza pustite neoznačena.

A B6 SZ D$ 15 X$ A12 D9 Q81 Q5 6F H$

■ **Aritmetična 1:** Napišite kratek program, v katerem boste pripisali spremenljivki B vrednost 6 in jo nato natisnite s stavkom PRINT.

■ **Aritmetična 2:** Napišite kratek program, ki bo pripisal vrednost 5 spremenljivki A, 7 spremenljivki B in 9 spremenljivki C. Seštejte vrednosti teh treh spremenljivk in pripišite vsoto spremenljivki D. Izpišite s stavkom PRINT vrednost spremenljivke D.

■ **Aritmetična 3:** Preglejte naslednje vrstice ter poskusite ugotoviti vrednost spremenljivke C.

```
LET C = 5 + 4*3
PRINT C
```

■ **Aritmetična 4:** Kakšen bo rezultat, ki se bo izpisal v tem programu?

```
 10 REM TA PROGRAM UREDI
 20 REM IMENA PO ABECEDI
 30 PRINT "ODLOČI SE, KOLIKO"
 40 PRINT "IMEN ŽELIŠ VPISATI"
 50 PRINT "NATO VSTAVI IMENA"
 60 PRINT "V OBLIKI:"
 70 PRINT "IME (PRESLEDEK) PRIIMEK"
 80 REM
 90 REM  TO JE GLAVNI PROGRAM
100 PRINT
110 PRINT
120 GO SUB 250
130 GO SUB 400
140 GO SUB 1000
150 GO SUB 2000
160 REM
170 REM KONEC GLAVNEGA PROGRAMA
180 STOP
250 REM PODPROGRAM ZA ISKANJE ŠTEVILA
260 REM IMEN, KI SE BODO VPISALA
270 PRINT "KOLIKO IMEN"
280 PRINT "ŽELIŠ VSTAVITI"
290 PRINT
300 INPUT N
310 DIM A$(N)
320 RETURN
400 REM PODPROGRAM ZA VPIS IMEN
410 PRINT "VPIŠI IME V TEJ OBLIKI:"
420 PRINT "IME (PRESLEDEK) PRIIMEK <CR>"
430 PRINT "NPR. MATJAŽ NOVAK"
440 FOR X = 1  TO N
450 PRINT "VPIŠI IME"
460 INPUT A$(X)
470 GO SUB 500
480 NEXT X
490 RETURN
500 REM PODPROGRAM ZA ZAMENJAVO VRSTNEGA
    REDA IMEN
510 LET L =  LEN (A$(X))
520 LET S = INSTR(A$(X)," ")
530 LET C$ = LEFT$(A$(X),S-1)
540 LET F$ = RIGHT$(A$(X),L-S)
550 LET F$=F$+","
560 LET A$(X)=F$+C$
570 RETURN
1000 REM PODPROGRAM "UREDI"
1010 LET S=0
1020 FOR P=1  TO N-1
1030 IF A$(P)>A$(P+1) THEN  GO SUB 1100
1040 NEXT P
1050 IF S=1 THEN  GO TO 1000
1060 RETURN
1100 REM PODPROGRAM "ZAMENJAJ"
1110 LET T$=A$(P)
1120 LET A$(P)=A$(P+1)
1130 LET A$(P+1)=T$
1140 LET S=1
1150 RETURN
2000 REM PODPROGRAM "IZPIŠI"
2010 PRINT
2020 FOR Q=L TO N
2030 PRINT A$(Q)
2040 NEXT Q
2050 RETURN
```

```
LET A = 3
LET B = 2
LET C = 9
LET D = 4
LET E = (A + B)*(C — D)
PRINT E

LET E = S
LET E = E*E
PRINT E
```

■ **Primerjava 1:** Kakšna mora biti vrednost spremenljivke X, da bo program izpisal sporočili?

```
 70 LET A = 5
 80 LET B = X
 90 LET R = B — A
100 IF R = 0 THEN GOTO 120
110 GOTO 10
120 PRINT "ČESTITAM! ZMAGAL SI"
999 END
```

■ **Primerjava 2:** Pri kateri najmanjši vrednosti spremenljivke X bo program še skočil v vrstico 300?

```
250 IF X>6*100 THEN GOTO 300
```

■ **Primerjava 3:** Pri kateri najmanjši vrednosti spremenljivke Z bo program skočil na sporočilo »Čestitam .....«?

```
340 IF Z<10000 THEN GOTO 500
350 IF Z>=10000 THEN GOTO 520
```

■ **Izpis 1:** Vzemite, da je vrednost spremenljivke T enaka 50. Napišite stavek PRINT, ki bo izpisal: VREDNOST SPREMENLJIVKE T JE 50. Opozorilo: Napišite sporočilo med narekovaji, uporabite podpičje ter ime spremenljivke.

■ **Izpis 2:** Poglejte naslednji kratki program ter dopolnite stavek PRINT tako, da bo izpisal naslednje sporočilo:

```
ŽAL MI JE, VENDAR JE TVOJ REZULTAT 175
PRENIZEK
```

Dopolnite program tako, da bo vsakič mogoče vpisati nov rezultat:

```
620 REM SPREMENLJIVKA S JE REZULTAT
    DO SEDAJ
625 IF S <= 500 THEN GOTO 640
630 GOTO 700
640 PRINT "ŽAL MI JE"
```

■ **Izpis 3:** Katero sporočilo se bo izpisalo med delovanjem programa?

```
200 LET A$ = "TEČAJ PROGRAMIRANJA"
210 LET B$ = "KAKO BI ŽELEL";
220 PRINT B$
230 PRINT A$
```

■ **Vpis 1:** Stavek INPUT je le eden izmed mogočih načinov pripisovanja vrednosti spremenljivkam. Katero tipko moramo pritisniti med delovanjem naslednjega programa, da se bo izpisal odgovor 12?

```
60 INPUT N
70 LET N = N*2
80 PRINT N
```

■ **Vpis 2:** Kaj se bo izpisalo tukaj?

```
100 PRINT "PROSIM VPIŠI IME"
110 INPUT N$
120 PRINT "ZDRAVO";N$;"JAZ SEM TVOJ
    RAČUNALNIK"
```

Posebnosti računalnikov

Ta program ne bo deloval na računalnikih Atari 400 in 800, ker ta dva popolnoma drugače ravnata z nizi.

DIM

Na ZX 81 in Spectrumu zamenjajte vrstico 310 z:
310 SIM A$ (N, 30)

GOTO

V vrstici 1050 pride ukaz GOTO 1000 takoj za besedo THEN. V tem primeru dovoli večina računalnikov napisati stavek IF...THEN brez besede GOTO. Vrstica 1050 se lahko tudi napiše:
1050 IF S = 1 THEN 1000

INSTR

Tega ukaza ni pri računalnikih ZX 81, Spectrum, Commodore 64, Vic-20 in Oric-1. Pri računalnikih Commodore in Oric-1 zbrišite vrstico 520 in jo zamenjajte s:

```
515 LET S = 0
520 FOR P = 1 TO L
523 IF MID$(A$(X),P,1) = "," THEN LET
    S = P
524 IF S = P THEN LET P = L
525 NEXT L
```

INSTR

Pri Spectrumu in ZX 81 izbrišite vrstice 510 do 560 in jih zamenjajte z:

```
510 LET D$ = A$(X)
520 LET L = LEN D$
530 LET S = 0
540 FOR P = 1 TO L
550 IF D$(P) = "," THEN LET S = P
560 IF S = P THEN LET P = L
570 NEXT P
```

LEFT$

```
580 LET C$ = D$(TO S—1)
590 LET F$ = DS (S + 1 TO)
600 LET A$(X) = F$ + "," + C$
610 RETURN
```

RIGHT$

Nobenega teh ukazov ni pri Spectrumu in ZX 81. Njim ustrezni ukazi v Sinclair basicu so:

LEFT$(Z$,N)	zamenjajte z Z$ (TO N)
RIGHT$(Z$,N)	zamenjajte z Z$ (LEN (Z$) — N + 1 TO)
MID$(Z$,P,N)	zamenjajte z Z$(P TO P + N - 1)

MID$

Funkcije v basicu

Obstajajo funkcije, ki so že vdelane v basic, kar pomeni, da je bilo veliko programiranja že opravljeno za nas. Če znamo te funkcije uporabljati, smo tudi zmogljivejši pri programiranju

Vzemimo za zgled, da želimo izračunati v enem izmed naših programov kvadratni koren nekega števila. To lahko storimo na veliko različnih načinov. Najokornejši in najmanj zadovoljiv bi bil pač ta, da bi sestavili tabelo kvadratnih korenov, ki bi jo nato uporabili pri iskanju kvadratnega korena določenega števila. To ste se verjetno učili tudi v šoli. Boljša metoda je ta, da uporabimo funkcijo kvadratnega korena, ki je že vdelana v večini verzij basica. V tem primeru bo aritmetično operacijo prevzel basic, tako da uporabniku ni treba skrbeti za to. Poglejmo, kako deluje:

```
10 REM TA PROGRAM IZRAČUNA
20 REM KVADRATNI KOREN ŠTEVILA
30 PRINT "VPIŠI ŠTEVILO, ZA KATERO
   HOČEŠ"
40 PRINT "IZRAČUNATI KVADRATNI
   KOREN"
50 INPUT N
60 LET A = SQR(N)
70 PRINT "KVADRATNI KOREN OD" ;N;"JE";
   A
80 END
```

Vtipkajmo ta kratki program in se prepričajmo, da nam da res kvadratni koren vsakega vpisanega števila. Seveda pa si poglejmo pravila, ki jih moramo upoštevati pri uporabi te funkcije.

Funkcijo v basicu večinoma sestavlja ukaz (v našem primeru SQR), ki mu v oklepajih sledi izraz, ki ga je treba obdelati. V našem primeru je število N, vpisali smo ga s tipkovnico in želimo izračunati njegov kvadratni koren. Vrstica 60 pravi: »Naj se kvadratni koren števila N pripiše spremenljivki A.« Vrstica 70 izpiše vrednost spremenljivke A.

Izraz znotraj oklepajev se imenuje tudi »argument« funkcije in ni nujno, da je vedno spremenljivka: prav tako lahko uporabimo tudi dejanske številke. Vtipkajmo sedaj naslednje vrstice in poglejmo, kaj se zgodi, ko program poženemo:

```
10 PRINT SQR(25)
20 END
```

Vidimo, da to prav tako dobro deluje. Podobno lahko uporabimo bolj kompleksne argumente znotraj oklepajev. Poskusimo tole:

```
10 LET A = 10
20 LET B = 90
30 LET C = SQR(A+B)
40 PRINT C
50 END
```

Ta kratki program se lahko še skrajša, če spremenimo vrstice 30 in 40:

```
10 LET A = 10
20 LET B = 90
30 PRINT SQR(A+B)
40 END
```

Na funkcije moramo misliti kot na kratke programe, vdelane v basic, ki jih programer lahko uporabi ob vsakem času. Večina zvrsti basica ima kar lepo število vdelanih funkcij, kakor tudi možnost, da si uporabnik opredeli nove za uporabo znotraj programa. Kasneje si bomo ogledali, kako se to naredi. Sedaj bomo spoznali še nekaj pogosto rabljenih funkcij. Le-te obstajajo v dveh verzijah: bodisi številčne funkcije, v katerih je argument (del znotraj oklepajev) lahko številka, številčna spremenljivka ali številčni izraz, ali funkcije niza, v katerih je argument znakovni niz ali izraz, sestavljen iz znakovnih nizov. Najprej si oglejmo nekaj številčnih funkcij.

Prej smo napisali program, ki je računal število ploščic za polaganje. Majhna napaka v tem programu je bila ta, da smo lahko dobili razultat, ki ni bil celo število; recimo 988,24 ploščice. Sedaj si želimo, da se rezultat zaokroži na najbližje celo število. Cela številka v matematiki se v angleščini imenujejo »integer«. Ena od funkcij v basicu bo poiskala celi del vsakega števila. Tako pa deluje:

```
10 PRINT "VPIŠI ŠTEVILO Z DECIMALNIM
   DELOM"
20 INPUT N
30 PRINT "CELI DEL ŠTEVILA JE";
40 PRINT INT(N)
50 END
```

Če poženemo program in vstavimo številko 3.14, bo program na zaslon napisal:

```
CELI DEL ŠTEVILA JE 3
```

Če imamo opraviti s ploščicami, bomo vsekakor rezultatu prišteli 1, da si zagotovimo, da bomo kupili več, kakor je označeni rezultat, ne manj.

Ob drugi priliki bomo želeli izvedeti predznak števila, da se prepričamo, ali je negativen, pozitiven ali enak nič. Za to opravilo uporablja večina računalnikov funkcijo SGN. Preskusimo sedaj naslednje:

```
30 LETA(1,2) = 251
40 LETA(1,3) = 1220
50 LETA(1,4) = 20000

610 LETA(12,5) = 12122
```

Očitno je, da je takšen način vpisovanja podatkov težavno delo. Dosti preprosteje opravite to z uporabo stavkov READ in DATA ali pa s stavkom INPUT skupaj z gnezdenimi FOR... NEXT zankami. Oglejmo si, kako to lahko opravimo s stavkom READ:

```
10 DIM A(12,5)
20 FOR R = 1 TO 12
30 FOR C = 1 TO 5
40 READ A(R,C)
50 NEXT C
60 NEXT R
70 DATA 2300, 251, 1220, 20000, 12000, 2300
   354, 1325
80 DATA 13435, 13800, 2300, 290.5, 1405,
   12875, 12010, 2300
90 DATA 262, 1335, 12935, 15060, 2300, 250,
   1200, 12700
100 DATA 25000, 2300, 250, 1155, 12972,
    11000, 2300, 305
110 DATA 1050, 12812, 10145, 2300, 295, 1100,
    25317, 25245
120 DATA 2300, 282.5, 1250, 14771, 11215,
    2300, 311.5
130 DATA 1835, 12814, 12645, 2300, 311.5,
    2025, 18645
140 DATA 14712, 2300, 311.5, 2435, 28341,
    12122
150 END
```

V tem programčku moramo biti pozorni na več dejstev. Prvo je to, da je ukaz DIM prav na začetku programa. Ukaz DIM mora biti izvršen le enkrat, zato se po navadi postavlja na začetek programa ali preden je izvršena katerakoli zanka. Drugo pomembno dejstvo je, da sta v programu dve zanki FOR... NEXT, ena, ki postavlja »vrstični« del indeksa, in druga, ki postavlja »stolpič«. Ti dve zanki ne sledita druga drugi; »vgnezdeni« sta druga v drugo. Omenimo še izbrane meje indeksov. FOR R = 1 TO 12 bo povečevalo število vrstice od 1 do 12; FOR C = 1 TO 5 pa bo povečevalo število stolpiča od 1 do 5.

Prav v sredini gnezdenih zank je ukaz READ. Najpomembnejši del programa je:

```
20 FOR R = 1 TO 12
30 FOR C = 1 TO 5
40 READ A(R,C)
50 NEXT C
60 NEXT R
```

V prvem obhodu, ko sta izvršeni vrstici 20 in 30, sta vrednosti R in C obe enaki ena ter bo vrstica 40 ekvivalentna READ A(1,1). Prva vrednost med podatki je 2300. Pripisana je elementu polja, ki je v prvi vrstici in prvem stolpiču. Izbira osmih številk v vsakem ukazu DATA je čisto poljubna.

Potem ko se je to zgodilo, ukaz NEXT C vrača program na vrstico 30 in vrednost C je povečana na 2. Vrstica 40 je sedaj ekvivalentna READ A(1,2) in naslednja vrednost, 251, bo pripisana elementu v prvi vrstici in drugem stolpcu polja. Ta postopek se ponavlja, dokler ni C povečan na 5. Zatem ukaz NEXT R vrne program v vrstico 20 in R je povečan na 2. Vrstica 30 bo ponovno postavila C = 1, tako bo vrstica 40 ekvivalentna READ A(2,1).

Gnezdenje zank na takšen način je zelo uporabno, vsekakor pa je treba tudi nekoliko paziti. Vsaka zanka mora biti v celoti gnezdena znotraj druge zanke, vrstni red ukazov NEXT pa mora biti skrbno izbran. Prvi zanki, FOR R, pripada drugi ukaz NEXT. Kadar sta dve zanki gnezdeni, imenujemo prvo zanko zunanja in drugo notranja. Celotna notranja zanka bo vedno sklenjena, preden bo indeks zunanje zanke povečan. Zanke tako gnezdimo v tako »globino«, kakor to zahteva program, vendar so takšni programi kompleksni in jim težko sledimo in jih popravljamo. Zelo slaba je navada postavljati vejitvene instrukcije znotraj zank, in ukazov GOTO se je treba ogibati.

Oglejmo si še ukaze DATA. Opazno je, da vejice ločujejo podatkovne postavke, vendar pred prvo ali po zadnji vrednosti v vrstici ne sme biti vejice. Med posamezne podatkovne vrednosti so vstavljeni presledki, vendar to ni običajno: pri vlaganju zlahka naredimo kake napake in kasneje jih težko odkrijemo. Število ukazov DATA ni omejeno. Vsaka nova vrstica se mora začeti z ukazom DATA. Podatki so brani drug za drugim od prvega ukaza DATA do konca. Prepričajte se, da je število podatkovnih vrednosti pravilno, v nasprotnem primeru bo program med izvajanjem programa sporočil napako.

Prikazani program za sedaj pravzaprav ne dela nič, razen da spreminja ustrezne podatke v dvodimenzijsko polje. Potem ko je bil program naložen in pognan z RUN, se ne bo zgodilo nič posebnega in na zaslonu boste videli znak v basicu. Za kontrolo, ali so podatki pravilno razporejeni, lahko poskusimo nekaj ukazov PRINT (angl. »izpiši«). (V basicu je ukaz ključna beseda, ki je lahko takoj izvršena, ne da bi bila del programa, zato ne potrebuje številke vrstice. Takšni so npr. ukazi LIST, RUN, SAVE, AUTO, EDIT in PRINT.) PRINT A(1,1) < CR > bo povzročilo, da se bo na zaslonu pokazala številka 2300. Kaj bo izpisano z naslednjimi ukazi?

```
PRINT A(12,1)
PRINT A(1,5)
PRINT A(5,1)
PRINT A(5,5)
```

Da bi program delal kaj uporabnega, ga je treba razširiti. Takšen, kakršen je sedaj, je ustrezna zasnova za »glavni program«. Da bi ga lahko uporabili kot del večjega, bolj uporabnega programa, napišemo module v obliki podprogramov, ki bodo klicani z uporabo ukazov GOSUB, vrinjenih na ustrezna mesta pred ukazom END.

Na začetnih stopnjah načrtovanja programa za obdelavo gospodinjskega proračuna je najboljše začeti s kratkim opisom splošnih zahtev. Lahko se odločimo, da želimo imeti izračunane celotne in povprečne mesečne stroške, ali pa stroške po posameznih kategorijah (npr. za električni tok). Če želimo imeti znotraj programa izbiro, katere podprograme hočemo izvršiti, bomo verjetno program opremili s pripravljenim »me-

nujem«, ki bo direktno kontroliral potek programa in ga kot rezultat našega odziva usmerjal proti ustreznim podprogramom. Začetni osnutek programa na tej stopnji bi lahko bil takle:

```
GLAVNI PROGRAM
  (VPIS PODATKOV)
MENU
  (IZBIRA PODPROGRAMA)
KONEC
```

Majhna dodatna izdelava bi lahko pokazala, da bomo potrebovali podprograme, ki bodo računali celotne zneske po mesecih ali po posameznih kategorijah (MESVSOTA in KATVSOTA), povprečne mesečne stroške (MESPOVP) in povprečne letne stroške po kategorijah (LETPOVP). Razlog za uporabo enobesednih imen za te podprograme je ta, da nam omogoča načrtovanje programa na tej stopnji brez skrbi za detajle, kakor so npr. številke vrstic. Lahko pa se odločimo, da bo tudi v menuju obravnavan kot podprogram, s tem bo glavni del programa ostal kot ločen modul. Naslednja stopnja izdelave programa bi lahko bila takšna:

```
GLAVNI PROGRAM (VPIS PODATKOV)
MENU (KLICANJE PODPROGRAMOV)
KONEC

**PODPROGRAMI**
1 MENU
2 VSOTE
3 POVPREČJA

(2) VSOTA
4 MESVSOTA
5 KATVSOTA

(3) POVPREČJE
6 MESPOVP
7 KATPOVP
```

Osnutek programa kaže, da nam bo MENU podprogram dal na voljo izbiro med VSOTAmi in POVPREČJEm. Obe bosta podprograma. Podprogram VSOTA bo omogočal nadaljnjo izbiro med MESVSOTO in KATVSOTO. Šele to bodo podprogrami, ki bodo pravzaprav izvajali račune.

Podprogram za POVPREČJE nam bo dal na voljo izbiro med MESPOVP in KATPOVP, in vnovič bosta ta dva podprograma izvajala ustrezni račun. Na tej stopnji bi moralo biti mogoče prekontrolirati, ali bo naš »program« res delal to, kar si želimo, in sicer brez dejanskega kodiranja (detajlnega pisanja programa v basicu). Če smo s tem zadovoljni in je »za zdaj še vse v redu«, se lahko spoprimemo s pisanjem samih modulov (podprogramov). Edina sprememba v glavnem programu je klicanje podprograma pred ukazom END, ki pomeni sklep izvajanja programa. Tako lahko dodamo:

```
145 GOSUB * * MENU * *
```

Upoštevajmo, da za podprograme še uporabljamo »imena« — raje kakor številke vrstic. Mnogi programski jeziki, npr. pascal, dopuščajo klicanje podprograma po imenu, večina zvrsti basica pa tega ne dopušča, namesto imena se uporabljajo dejanske številke vrstic. Vsekakor se takšni »detalji« lahko vstavijo pozneje.

Oglejmo si, kako bi lahko bil napisan podprogram MENU (številke vrstic so izpuščene, če želite dopolniti program, lahko dodate ustrezne številke).

```
REM PODPROGRAM * * MENU * *
PRINT "ŽELIŠ V(SOTE) ALI P(OVPREČJA)?"
PRINT "NATIPKAJ ALI V ALI P"
INPUT L$
IF L$="V"THEN GOSUB * VSOTA *
IF L$="P"THEN GOSUB * POVPREČJE *
RETURN
```

Opomba: klicane podprograme smo označili z znaki *— — — *. Vi boste morali namesto tega uporabiti številke vrstic. Te lahko vstavite takrat, ko jih že poznate.

Vzemimo, da ste natipkali V za VSOTA. Program bo poklical podprogram VSOTA. Ta bo predstavil novi menu, ki bi lahko bil takle:

```
REM PODPROGRAM * * VSOTA * *
PRINT "ŽELIŠ VSOTE PO M(ESECIH) ALI
K(ATEGORIJAH)?"
PRINT "NATIPKAJ ALI M ALI K"
INPUT L$
IF L$="M"THEN GOSUB * MESVSOTA *
IF L$="K"THEN GOSUB * KATVSOTA *
RETURN
```

Recimo, da ste izbrali M za MESVSOTA. Oglejmo si, kako bi lahko napisali programski modul, ki bo izračunal celotne stroške za vsak mesec v letu.

```
REM PODPROGRAM * * MESVSOTA * *
REM IZRAČUN CELOTNIH STROŠKOV ZA
KATERIKOLI MESEC
PRINT "IZBERI MESEC:"
PRINT "1—JAN, 2—FEB, 3—MAR, 4—APR,
5—MAJ, 6—JUN"
PRINT "7—JUL, 8—AVG, 9—SEP, 10—OKT,
11—NOV, 12—DEC"
PRINT "PODAJ MESEC S ŠTEVILKO!"
LET T = 0
INPUT M
FOR C = 1 TO 5
LET T = T + A(M,C)
NEXT C
PRINT "CELOTNI STROŠKI ZA MESEC ST.";M;
"SO:";T
RETURN
```

Številka, ki predstavlja mesec, je vtipkana in ukaz INPUT pripiše to številko spremenljivki M (MESEC). M je uporabljen, da specificira vrstico dvodimenzijskega polja A. Zanka FOR...NEXT povečuje vrednost C (stolpič) od ena do pet tako, da bo pri prvem prehodu skozi zanko, če smo izbrali za mesec številko 3, ukaz LET ekvivalenten LET T=T+A(3,1). Pri naslednjem prehodu bo le-ta ekvivalenten LET T=T+A (3,2) itd.

Dvodimenzijska polja so idealna za vsak program, ki vsebuje tabele podatkov, bodisi da so le-ti statistične, finančne ali kakršnekoli druge narave.

Rešitev nalog na strani 204.

RND funkcija:

```
 40 IF R>6 THEN LET R=1
```

Zanka in povprečje:

```
  5 FOR L=1 TO 100
  :
 80 LET T=T+R
 90 NEXT L
100 LET A=T/100
110 PRINT A
120 END
```

Zamenjaj s podprogramom:
Izbriši vrstice 5, 80, 90, 100 in 110 v gornji rešitvi. Spremeni vrstice od 10 do 70 v (npr.) 1010 do 1070. Prepričaj se, da je vrstica 40 gornja rešitev funkcije RND. Potem dodaj vrstico 1080 RETURN. Včleni rezultat v glavni program. Spremeni vrstici 50 in 130 v glavnem programu v 50 GOSUB 1000 in 130 GOSUB 1000.

INKEY$:

```
 10 PRINT "PRITISNI KATEROKOLI TIPKO!"
 20 LET A$=INKEY$
 30 IF A$=" " THEN GOTO 20
 40 PRINT "PRITISNIL SI TIPKO";A$
 50 END
```

Za Spectrum dodaj: 15 IF INKEY$<>"" THEN GOTO 15

Časovna zanka:

```
  5 PRINT "PRITISNI TIPKO ZA PRESLEDEK PO
    10 SEKUNDAH"
 10 FOR L=0 TO 1
 20 LET R=R+1
 30 IF INKEY$=" " THEN GOTO 60
 40 LET L=0
 50 NEXT L
 60 PRINT "VREDNOST R PO 10 SEKUNDAH JE";R
 70 END
```

IF... THEN

```
 10 GOSUB 1000
 20 PRINT "UGIBANJE ŠTEVILA"
 30 FOR G=1 TO 5
 40 INPUT N
 50 IF N>R THEN GOTO 110
 60 IF N<R THEN GOTO 130
 70 IF N=R THEN GOTO 150
 80 NEXT G
 90 PRINT "NE GRE VEČ, IZGUBIL SI!"
100 GOTO 500
110 PRINT "PREDLOŽIL SI PREVEČ"
120 GOTO 80
130 PRINT "PREDLOŽIL SI PREMALO"
140 GOTO 80
150 PRINT "PRAV IMAŠ, ČESTITAM!"
500 END
1000 REM** PODPROGRAM ZA GENERACIJO
     NAKLJUČNEGA ŠTEVILA** (tu vstavi svoj
     podprogram)
1020 RETURN
```

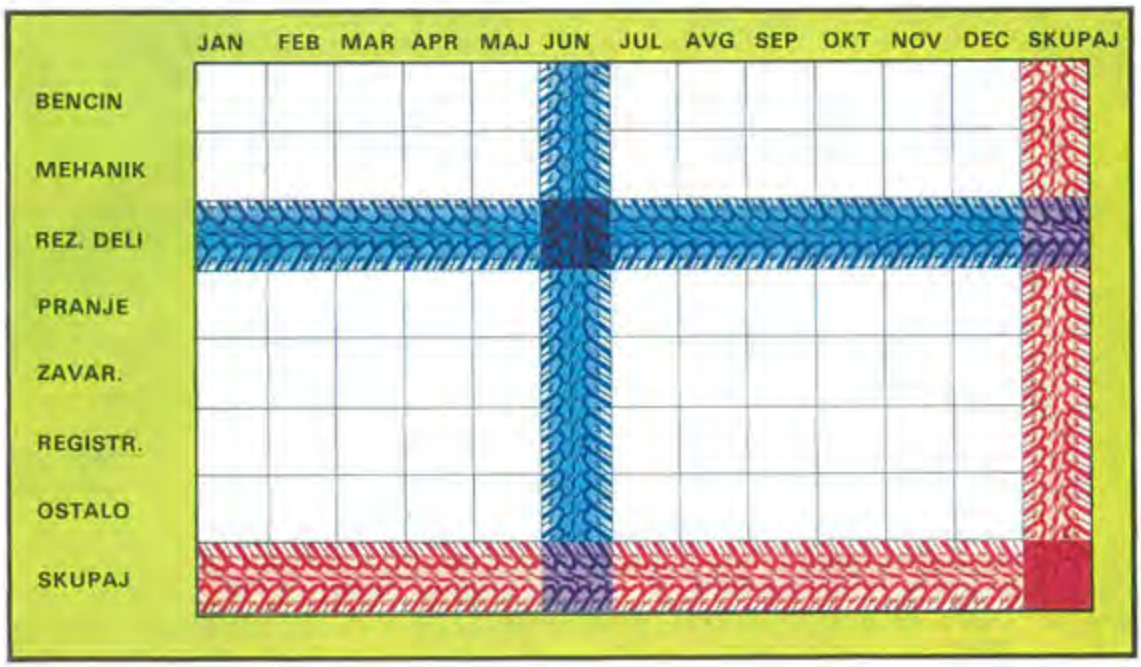
Nalogi

Prirejanje vrednosti: Napiši program, ki bo priredil vrednosti elementom (»bencin«, »vzdrževanje« itd.) matrike (glej spodnjo ilustracijo). Nato napiši podprogram, ki bo spraševal po mesecu, vrsti stroška in izpisal vsebino tako definiranega kvadratka. Končno napiši podprogram, ki bo poiskal vsoto vsakega izmed stolpičev, in jo postavi v spodnji kvadratek. Enako naredi še za vrstice in na koncu izračunaj celotno vsoto, ki naj bo shranjena v spodnjem desnem kvadratku.

Napaki: Naslednji program ne bo deloval pravilno in bo sporočil napako. Program ima dve napaki. Poišči ju in naredi potrebne popravke.

```
 10 DIM A(3,4)
 20 FOR R = 1 TO 3
 30 FOR C = 1 TO 4
 40 READ A(R,C)
 50 NEXT C
 60 NEXT R
 70 FOR X=1 TO 3
 90 FOR Y=1 TO 4
100 PRINT A(Y,X)
110 NEXT Y
120 NEXT X
130 DATA 2,4,6,8,10,12,14,16,18,20,22
140 END
```

Rešitvi sta v naslednjem poglavju.



Stroški za avto
Na sliki je prikazana mreža kvadratkov 8 × 13. Vrstice predstavljajo različne sestavine stroškov za vzdrževanje avtomobila, stolpiči pa mesece v letu. Sledite vaji »Pripisovanje vrednosti« in tako izračunajte celotne stroške vzdrževanja avtomobila

Kontrola

Vse zvrsti basica imajo »kontrolne strukture«, ki vodijo potek izvajanja programa. Nekateri računalniki pa ponujajo široko področje drugih možnosti, ki se razlikujejo v odtenkih

Prvih 10 delov tečaja o temeljih programiranja je skoraj v celoti obdelalo vse pomembnejše lastnosti jezika basic. V tem poglavju bomo podali pregled do sedaj obdelane snovi, obravnavali nekaj zanimivih stranskih učinkov in podali nekaj smernic za nadaljnje delo.

Najprej pregled: višji jeziki, kakor npr. basic, so uporabnika preskrbeli z množico ukazov, ki so znotraj računalnika prevedeni v takšno obliko, da jo ta razume. Vsak računalniški program se lahko napiše z uporabo dveh vrst konstrukcij. To so »sekvenčne« konstrukcije in »kontrolne strukture«, od katerih sta v basicu bistveni samo dve: to sta IF...THEN...ELSE in WHILE...DO. Večina drugih računalniških jezikov jih daje precej več.

Sekvenčne konstrukcije dopuščajo, da se naloga razdeli v vrsto podnalog, ki, ko so izvršene v določenem zaporedju, opravijo glavno nalogo. Velikost podnalog je odvisna od jezika; v basicu so predstavljene z ukazi, napisanimi v vsaki vrstici, vrstni red izvajanja je predstavljen s številkami vrstic. Če je treba pomnožiti spremenljivki pripisano vrednost z 10, bi lahko uporabili naslednjo sekvenco:

```
110 INPUT N
120 LET N = N*10
130 PRINT N
```

Poleg sekvenčnih konstrukcij potrebujemo še kontrolne strukture. To so konstrukti, ki spremenijo vrstni red izvrševanja ukazov v programu.

Najpreprostejša kontrolna struktura, ki jo ima basic, je GOTO. To je brezpogojni skok (ali vejitev), ki prenese delovanje programa na določeno vrstico, ne da bi bilo treba kaj preskusiti ali izpolniti kak pogoj. GOSUB je tudi brezpogojna vejitev, vendar se program zmeraj vrača (ang. RETURN) na mesto, ki je neposredno za ukazom GOSUB. Uporaba ukaza GOSUB se odlično ujema s strukturnim programiranjem.

Tudi kontrolna struktura IF...THEN...ELSE je dostopna v basicu. Ima obliko ukaza IF...THEN in naslednjo sintakso (»sintaksa« v računalniškem žargonu pomeni »obliko«):

IF (določen pogoj) je res THEN izvrši določeni ukaz (ELSE) izvrši naslednji ukaz

V standardnem basicu je ELSE iz stavka IF...THEN...ELSE privzeto vsebovan. V nekaterih narečjih basica ter nekaterih drugih jezikih, npr. v pascalu, je ELSE del ukaza.

IF...THEN...ELSE (IF...THEN v basicu) opravlja eno izmed dveh podnalog, odvisno od tega, ali je določen pogoj izpolnjen ali ne. Poglejmo naslednji program, ki naj izračuna kvadratne korene števil, podanih s tipkovnico, razen če je podana »signalna« vrednost —9999, ki končuje izvajanje programa:

```
10 PRINT "PODAJ ŠTEVILO"
20 INPUT N
30 IF N = —9999 THEN GOTO 70
40 LET S = SQR(N)
50 PRINT "KOREN ŠTEVILA";N;"JE";S
60 GOTO 10
70 END
```

Kontrolna struktura IF... THEN... ELSE
Če je pogoj izpolnjen, bodo izvršeni ukazi RES, če pa pokoj ni izpolnjen, bodo izvršeni ukazi NI RES.

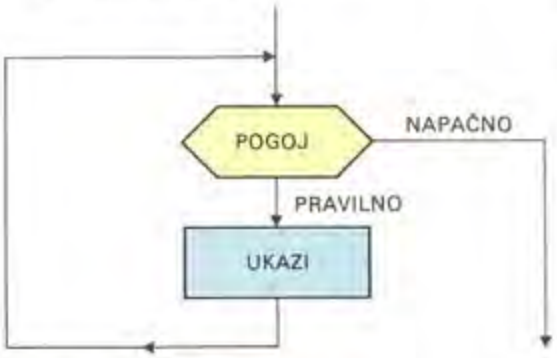
To, kar nam vrstica 30 pravzaprav pove, je »če (IF) je res, da je N = —9999, tedaj (THEN) pojdi na konec programa, drugače (ELSE) pa (če ni res, da je N = —9999) izvrši naslednjo vrstico programa in izračunaj koren števila N«. Druga pomembna kontrolna struktura (WHILE...DO, dokler...delaj) ni direktno dostopna v basicu, jo pa zlahka simuliramo. WHILE... DO je tipa »izvršuj zanko«, in to pomeni »ponavljaj (DO) ukaz ali vrsto ukazov, dokler (WHILE) je podani pogoj resničen« oziroma »WHILE je pogoj izpolnjen, DO nekaj«.

WHILE...DO vedno preskuša resničnost pogoja, preden se ukazi izvršijo. Če pogoj že ob prvem obhodu ni izpolnjen, ukazi (imenovani tudi telo zanke) niso izvršeni. Kot zgled obravnavajmo program igre, ki pove igralcu »KO BOŠ PRIPRAVLJEN, PRITISNI PRESLEDEK«. Ta del programa bi lahko bil zapisan takole:

```
WHILE PRESLEDEK NI PRITISNJEN
DO PREGLEJ TIPKOVNICO
ZAČNI IGRO
```

V basicu bi to lahko bilo napisano takole:

```
250 PRINT "PRITISNI PRESLEDEK KO BOŠ
    PRIPRAVLJEN"
260 IF INKEY$( )" " THEN GOTO 260
270 GOSUB *START*
```



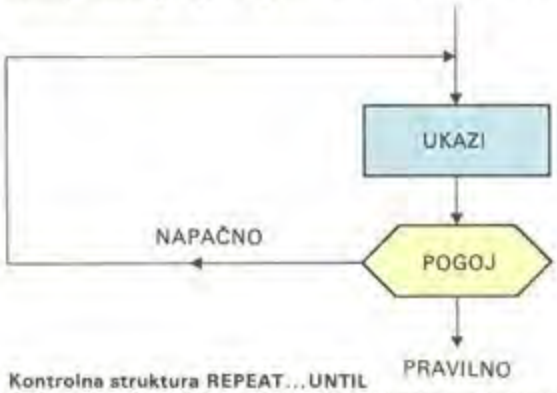
Kontrolna struktura DO... WHILE
Zanka se ponavlja, dokler je pogoj izpolnjen. Lahko se zgodi, da ukazi niso izvršeni (če začetni pogoj ni izpolnjen).

Vrstica 260 nam pove: če (IF) INKEY$ ni enak (<>) presledku (» «), tedaj (THEN) pojdi nazaj in prekontroliraj tipkovnico še enkrat. Nekoliko bolj elegantna oblika napisanega programa bi lahko bila:

```
250 PRINT "PRITISNI PRESLEDEK KO BOŠ
    PRIPRAVLJEN"
260 FOR X = 0 TO 1 STEP 0
270 IF INKEY$ = " " THEN LET X = 2
280 NEXT X
290 GOSUB *START*
```

V tem programu je del zanke (ki pregleduje tipkovnico) izvršen samo, če presledek ni pritisnjen. Če je ta pritisnjen (npr. INKEY$=» «), program stopi iz zanke FOR...NEXT v vrstico 290, ki kliče podprogram »START«. (Opomba: tukaj uporabljamo »označbo« oziroma ime podprograma. Mnoge zvrsti basica ne morejo klicati podprogramov po imenu, zato boste morali namesto imena uporabljati številke vrstic.)

Doslej se še nismo srečali z izrazom STEP (angl. korak). Kadar ga uporabljamo v zanki FOR...NEXT, omogoča spremembo »tekočega indeksa« v zanki za vrednost, različno od ena. V zanki FOR I = 1 TO 10 STEP 2 bo I imel ob prvem prehodu skozi zanko vrednost 1, tej vrednosti pa bodo sledile vrednosti 3, 5, 7 in 9. Ob naslednjem prirastku (na I = 11) bo presežena meja 10 in zanka bo končana. Možno je celo štetje indeksa za nazaj. FOR I = 10 to 1 STEP −1 bo povzročilo, da bo I štel od 10 do 1. Uporabljati STEP 0 je pravzaprav pametna zvijača, ki zagotavlja, da se zanka nikoli ne bo končala, razen če bo X »umetno povečan«, kot je to v našem ukazu IF...THEN.



Kontrolna struktura REPEAT...UNTIL
Zanka se ponavlja, dokler pogoj ni izpolnjen. Ukaz bo izpolnjen vsaj enkrat.

Še ena kontrolna struktura, ki tudi ni direktno dostopna v basicu, je REPEAT...UNTIL (angl. ponavljaj ... dokler). Tukaj preverjanje pogoja pride po glavnem delu zanke tako, da bo ukaz oziroma bodo ukazi v glavnem delu zanke ponovljeni vsaj enkrat. Oglejmo si »generator naključnih števil«:

```
10 PRINT "PRITISNI PRESLEDEK"
20 FOR X = 0 TO 1 STEP 0
30 LET R = 1
40 IF R > 9 THEN LET R = 1
50 IF INKEY$ = " " THEN LET X = 2
60 NEXT X
70 PRINT "VREDNOST R JE";R
```

Tukaj je glavno telo (povečevanje vrednosti R) izvršeno vsaj enkrat, ker je preskušanje pogoja za izstop iz zanke (IF INKEY$ = « «) postavljeno za stavkom, ki poveča vrednost R za 1, LET R = R + 1.

Še ena nebistvena, vendar koristna kontrolna struktura je CASE (angl. primer). V basicu je struktura CASE vključena v uporabo bodisi z ON...GOTO bodisi z ON...GOSUB. Oglejmo si, kako deluje. ON...GOTO je večkratna vejitev, ki združuje več pogojnih preskusov IF...THEN v en sam ukaz. Oglejmo si del programa, ki spreminja števila od 1 do 7 v sedem dni v tednu:

```
1050 IF D = 1 THEN GOTO 2020
1060 IF D = 2 THEN GOTO 2040
1070 IF D = 3 THEN GOTO 2060
1080 IF D = 4 THEN GOTO 2080
1090 IF D = 5 THEN GOTO 3000
2000 IF D = 6 THEN GOTO 3020
2010 IF D = 7 THEN GOTO 3040
2020 PRINT "PONEDELJEK"
2030 GOTO * END *
2040 PRINT "TOREK"
2050 GOTO * END *
2060 PRINT "SREDA"
2070 GOTO * END *
2080 PRINT "ČETRTEK"
2090 GOTO * END *
3000 PRINT "PETEK"
3010 GOTO * END *
3020 PRINT "SOBOTA"
3030 GOTO * END *
3040 PRINT "NEDELJA"
3050 GOTO * END *
```

Bolj strnjen način za dosego istega cilja v basicu je uporaba ON...GOTO takole:

```
1050 ON D GOTO 2020, 2040, 2060, 2080,
     3000, 3020, 3040
```

Desetiško	Dvojiško	Znak
32	00100000	␣
33	00100001	!
34	00100010	"
35	00100011	#
36	00100100	$
37	00100101	%
38	00100110	&
39	00100111	'
40	00101000	(
41	00101001	)
42	00101010	*
43	00101011	+
44	00101100	,
45	00101101	-
46	00101110	.
47	00101111	/
48	00110000	0
49	00110001	1
50	00110010	2
51	00110011	3
52	00110100	4
53	00110101	5
54	00110110	6
55	00110111	7
56	00111000	8
57	00111001	9
58	00111010	:
59	00111011	;
60	00111100	<
61	00111101	=
62	00111110	>
63	00111111	?
64	01000000	@
65	01000001	A
66	01000010	B
67	01000011	C
68	01000100	D
69	01000101	E
70	01000110	F
71	01000111	G
72	01001000	H
73	01001001	I
74	01001010	J
75	01001011	K
76	01001100	L
77	01001101	M
78	01001110	N
79	01001111	O
80	01010000	P
81	01010001	Q
82	01010010	R
83	01010011	S
84	01010100	T
85	01010101	U
86	01010110	V
87	01010111	W
88	01011000	X
89	01011001	Y
90	01011010	Z
91	01011011	[
92	01011100	\
93	01011101	]
94	01011110	^
95	01011111	_
96	01100000	`
97	01100001	a
98	01100010	b
99	01100011	c
100	01100100	d
101	01100101	e
102	01100110	f
103	01100111	g
104	01101000	h
105	01101001	i
106	01101010	j
107	01101011	k
108	01101100	l
109	01101101	m
110	01101110	n
111	01101111	o
112	01110000	p
113	01110001	q
114	01110010	r
115	01110011	s
116	01110100	t
117	01110101	u
118	01110110	v
119	01110111	w
120	01111000	x
121	01111001	y
122	01111010	z
123	01111011	{
124	01111100	|
125	01111101	}
126	01111110	~

ASCII
Tu imamo prikazane vse vrednosti ASCII med 32 in 126, njihove binarne ekvivalente ter znake, ki jih predstavljajo. Pomeni znakov, ki pripadajo vrednostim zunaj tega področja, se pri raznih računalnikih močno razlikujejo.

ON...GOSUB dela enako, le da v tem primeru vrednost spremenljivke določa, kateri podprogram bo klican. Tukaj imamo rahlo spremenjen program za izbiro grafike za kocko, katere številka je izbrana s funkcijo RND:

```
390 REM PODPROGRAM ZA IZBIRANJE, KI
400 REM UPORABLJA ON ... GOSUB
410 ON D GOSUB 530, 600, 670, 740, 810,
    880
470 RETURN
```

Čeprav ima vaša zvrst basica verjetno veliko ukazov in funkcij, ki jih tukaj nismo navedli, je večina teh ukazov razširitev »temeljnega« basica, in so namenjeni temu, da čimbolj izkoriščajo posebne lastnosti vašega računalnika. Mnogi od teh se bodo nanašali na grafične značilnosti, vdelane v opremo, kakor so to npr. ukazi PAINT, PAPER, INK, BEEP in CIRCLE. Takšni ukazi so »specifični za ta računalnik« in jih zato nismo zajeli v naš tečaj, čeprav vam bomo dali še več detajlov v drugih člankih.

Preden končamo temeljni del našega tečaja basica, je treba razjasniti še nekaj podrobnosti: obravnava množice znakov ASCII skupaj z nekaj funkcijami, ki nam olajšujejo delo z znaki, ter način opredeljevanja novih funkcij (ali funkcij, ki jih vaša zvrst basica ne pozna).

V zadnjih letih je bilo izpeljanih več načinov za digitalno predstavitev črk abecede in drugih znakov, kakor so števila in ločila. Eden izmed prvih načinov je bila Morsova koda, ki je za predstavljanje posameznih znakov uporabljala kombinacije pik in črt. S stališča računalnika ima Morsova koda pomanjkljivost, da ima za različne znake različno število bitov (od enega do šest). Druga prizadevanja, da bi se naredila bolj regularna in sistematična koda (npr. Baudotova koda, ki uporablja pet bitov, s katerimi se da prikazati 32 znakov), so zdaj zasenčena s skoraj univerzalnim sistemom, ki se sedaj uporablja pri kodi ASCII (American Standard Code for Information Interchange — Ameriška standardna koda za izmenjavo informacij).

Koda ASCII uporablja en zlog, s katerim lahko ponazori 94 znakov, ki se dajo natisniti, »presledek« in celo množico »kontrolnih« znakov. Osem bitov bi lahko dalo 256 različnih kombinacij (2^8), to je bistveno več, kakor je potrebno za predstavitev vseh znakov s tipkovnice standardnega pisalnega stroja ali računalnika, zato uporabljajo samo sedem bitov, kar da 128 različnih kombinacij. (Osmi bit je po navadi odveč, vendar se včasih uporabi za določitev nadomestne množice znakov v kakem tujem jeziku ali grafičnih znakov.) Binarne in decimalne kode ASCII so prikazane v priloženi tabeli.

Kot je razvidno iz tabele, je koda ASCII za črko A65 in za B66. Kodi za mali črki a in b sta 97 in 98. Vsaka mala črka ima v tabeli ASCII vrednost, ki je za 32 večja od njenega velikega ekvivalenta. Ta stalna »razlika« omogoča lahko spremembo malih črk v velike in narobe. Da to delamo, potrebujemo še dve funkciji, ki ju do sedaj v našem tečaju programiranja v basicu nismo potrebovali — to sta ASC in CHR$.

Funkcija ASC vzame znak, ki se da izpisati in vrne njegovo ekvivalentno kodo ASCII, tako da bo stavek PRINT ASC(»A«) izpisal na zaslon številko 65, PRINT ASC(»b«) pa bo izpisal 98.

Funkcija CHR$ dela ravno nasprotno: vzame številko, jo šteje kot kodo ASCII in vrne znak, ki ga ta koda predstavlja. Tako bo stavek PRINT CHR$(98) izpisal na zaslon b. Funkciji ASC in CHR$ se veliko rabita skupaj z funkcijami LEFT$, RIGHT$ in MID$ v programih, ki veliko uporabljajo nize znakov. Kot zgled imamo tukaj kratek programček, ki sprejme znak s tipkovnice in preveri, ali je to velika ali mala črka, in če ni velika, jo spremeni v veliko:

```
10 REM PRETVORBA MALIH ČRK V VELIKE
20 PRINT "PODAJ ČRKO"
30 INPUT C$
40 LET C = ASC(C$)
50 IF C > 90 THEN LET C = C - 32
60 PRINT CHR$(C)
```

V naslednjih poglavjih našega tečaja se bomo še srečali s podobnimi postopki z nizi znakov.

Na koncu si oglejmo še funkcije, ki jih vaš basic morda nima. Skoraj vse zvrsti tega jezika omogočajo programerju, da sam ustvarja nove funkcije, ki se uporabljajo skoraj enako lahko kakor že vdelane funkcije. Ukaz DEF pove basicu, da bo opredeljena nova funkcija. Spodaj imamo prikazano, kako opredeliti funkcijo, ki bo računala prostornino krogle (po obrazcu $V = \frac{4}{3}r^3\pi$, kjer je r polmer krogle), π (pi) konstanta, ki ima približno vrednost 3,14159:

```
10 FUNKCIJA REM, KI IZRAČUNA PROSTOR-
   NINO KROGLE
20 DEF FNV(X) = 4*3.14159*X*X*X/3
30 PRINT "PODAJ POLMER KROGLE"
40 INPUT R
50 PRINT "PROSTORNINA KROGLE POLME-
   RA"; R; "JE
60 PRINT FNV(R)
70 END
```

Takšen način opredelitve funkcije je zelo preprost, vseeno pa si pobliže oglejmo to vrstico:

```
DEFinira identifikacija funkcije
    ↓      ↓
20 DEF FNV(X) = 4*3.14159*X*X*X/3
        ↑      ↑
FuNkcija nadomestna spremenljivka
```

Kadar je funkcija opredeljena, črkam FN sledi identifikacijska črka — v našem primeru je to V — temu pa mora slediti »nadomestna spremenljivka«. Ista nadomestna spremenljivka mora biti uporabljena tudi v definiciji funkcije na desni strani enačaja. Kadar je funkcija uporabljena v programu, je lahko namesto nadomestne spremenljivke v definiciji uporabljena katerakoli številčna spremenljivka.

Kot naslednjo točko v gornjem programu bi lahko enakovredno uporabili funkcijo »volumen krogle« takole:

```
 999 LET A = 66
1000 LET B = FNV(A)
1010 PRINT B
1020 LET C = 5
1030 LET D = B + FNV(C)
1040 PRINT D
```

```
1050 LET G = FNV(16)
1060 PRINT G
```

Nekatere zvrsti basica dopuščajo uporabo več spremenljivk v definirani funkciji. V tem primeru lahko napišemo funkcijo, ki bo izračunala povprečje dveh števil, takole:

```
100 DEF FNA(B,C) = (B + C)/2
110 INPUT "PODAJ DVE ŠTEVILI"; B, C
120 LET    A = FNA(B,C):REM    FUNKCIJA
    "POVPREČJA"
130 PRINT "POVPREČJE "; B; "IN"; C; "JE"; A
```

Očitno je, da vrstica 110 vsebuje ekvivalent dveh ločenih stavkov v enem. Večina basicov avtomatično izpiše besede, ki so znotraj narekovaja, ki sledi ukazu INPUT. Tako je ta vrstica ekvivalentna vrsticama:

```
110 PRINT "PODAJ DVE ŠTEVILKI"
115 INPUT B, C
```

Ukazi, ki sicer pripadajo ločenim vrsticam, se lahko napišejo v eni, če zagotovimo, da je vsak od »samostojnih« ukazov ločen od prejšnjega z dvopičjem. Tako lahko varčujemo s prostorom pri dolgih programih, vendar pa so tako pisani programi manj pregledni in napake verjetnejše.

Spoznali smo najpomembnejše lastnosti jezika basic. Sedaj pa pogum in poskusite sami napisati nekaj programov!

Rešitev vaj s strani 208:

Napaki:
Pokazala se bo napaka »OUT OF DATA ERROR« (premalo podatkov), saj bi v ukazu DATA v vrstici 130 moralo biti 12 vrednosti. Pojavila se bo tudi napaka v vrstici 100, ko bo le-ta poskušala nasloviti element A(4,1). Ta vrstica bi morala biti takšnale:

```
100 PRINT A(X,Y)
```

Pripisovanje vrednosti:
Spodaj je prikazana različica programa, ki bo opravljal vse potrebne funkcije. Vaš program bi lahko bil drugačen.

```
 10 DIM A(8,13)
 20 FOR R = 1 TO 7
 30 FOR C = 1 TO 12
 40 READ A(R,C)
 50 NEXT C
 60 NEXT R
 70 REM DODAJ VSOTE
 80 GOSUB 300
 90 REM IZPIŠI POTREBNE PODATKE
100 GOSUB 200
110 PRINT "ŠE KAKŠEN PODATEK?"
120 PRINT "DY ALI N"
130 INPUT A$
140 IF A$ = "N" THEN GOTO 160
150 GOTO 100
160 END
200 PRINT "KATERI MESEC?"
210 PRINT "1 ZA JANUAR."
220 PRINT "13 ZA VSOTO, ITD."
230 INPUT M
240 PRINT "KATERI STROŠEK?"
250 PRINT "1 ZA BENCIN"
260 PRINT "8 ZA VSOTO, ITD."
270 INPUT X
290 RETURN
300 FOR I = 1 TO 7
310 LET T = 0
320 FOR C = 1 TO 12
330 LET T = T+A(R,C)
340 NEXT C
350 LET A(R,13) = T
360 NEXT R
370 FOR C = 1 TO 13
380 LET T = 0
390 FOR R = 1 TO 7
400 LET T = T+A(R,C)
410 NEXT R
420 LET A(8,C) = T
430 NEXT C
440 RETURN
500 REM TVOJI PODATKI NAJ SLEDIJO ZA TO VRSTICO
510 REM ŠTIRIINOSEMDESET VREDNOSTI
520 REM "DATA 11.35, 9.87" ITD.
```

Posebnosti računalnikov

ASCII

Računalnik Dragon-32 ima nestandardno zvrst kode ASCII, ki ne dopušča uporabe malih črk, tako da program, ki spreminja male črke v velike, tega tukaj ne bo opravil; vseeno ga preizkusite. Da bi več zvedeli o njegovih znakih, si preberite njegov priročnik.

DEF FNA(B)

Na računalnikih Oric-1, Vic-20, Dragon-32 in Commodore-64 znotraj oklepajev ne morete napisati več ko eno spremenljivko.

ASC( )

Na računalnikih Spectrum in ZX81 zamenjaj:

ASC(A$) s CODE A$
in zamenjaj
CHR$(65) s CHR$65.

CHR$( )

Če je argument izraz, mora biti v oklepajih. Preprosti argumenti, kakor sta A in 65, pa ne potrebujejo oklepajev.

ON... GOSUB

ON... GOTO

Ta dva ukaza nista na voljo na računalnikih Spectrum, ZX81 in Lynx.

INPUT" "

Na računalniku BBC Micro zamenjaj

INPUT "KATERIKOLI TEKST";M$$
z
INPUT "KATERIKOLI TEKST",M$$

DEF FN

Na računalniku BBC Micro morate definirati funkcijo na koncu programa po stavku END ali STOP, ne na začetku, kakor je bilo prikazano v danih zgledih. V tem primeru, kot tudi v še mnogih drugih, se je basic BBC pokazal kot močnejši od standardnega basica, zato dodatne informacije poiščite v svojem priročniku.

STEP 0

Na računalniku Oric-1, v dveh programih, ki prikazujeta lepo strukturo zanke, zamenjaj:

IF INKEY$ = " " THEN LET X = 2
z
IF KEY$ = " " THEN LET X = 1

Na računalniku Dragon-32 zamenjaj z:

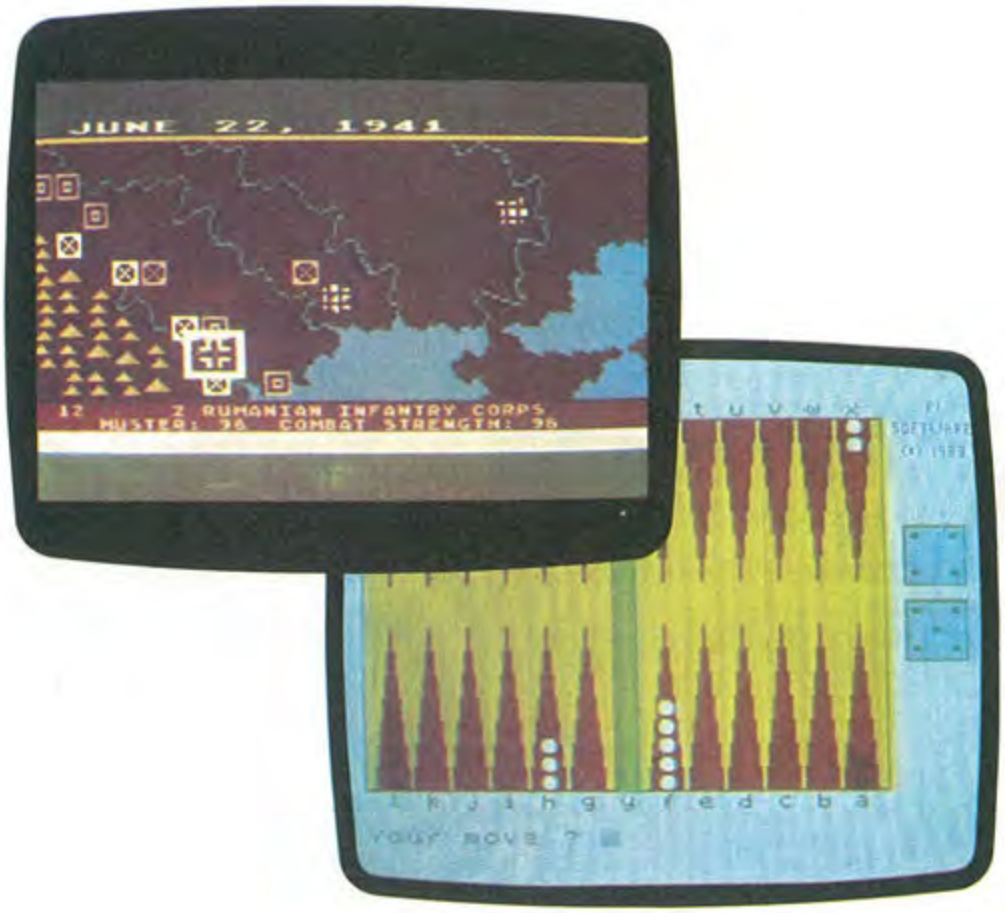
IF INKEY$ = " " THEN LET X = 1

Na računalniku Vic-20 in Commodore 64 zamenjaj:

IF INKEY$ = " " THEN LET X = 2
z
GET A$: IF A$ = " " THEN LET X = 1

8. poglavje

Grafika in zvok



Sodobni mikroračunalniki se močno razlikujejo po svojih grafičnih in zvočnih sposobnostih. S pravim znanjem boste iz svojega računalnika izvabili marsikaj.

Še zvok...

Grafika in zvok je novo poglavje, ki vas bo poučilo o tem, kako najbolje izkoristiti grafične in zvočne zmogljivosti računalnika

V zadnjih letih se je zmogljivost hišnih računalnikov zelo povečala. Igre, ki jih je mogoče igrati na novih računalnikih, so bistveno pripomogle k njihovi priljubljenosti. Zato proizvajalci vlagajo veliko časa in naporov v razvoj grafike hišnih računalnikov. Podobne pozornosti so deležne tudi zvočne lastnosti računalnika. Če bi vprašali uspešnega programerja računalniških igric, kako pomembni so programi, ki dajejo zvočne učinke, bi jih verjetno postavil na tretje mesto, takoj za konceptom igre in grafiko. Pametna uporaba zvočnih učinkov in glasbe lahko precej pripomore k vznemirljivosti in zabavnosti igre.

Poleg uporabe pri igrah pa nam lahko računalnik pomaga pri poglabljanju našega znanja o glasbi. Pri večini računalnikov najdemo ukaze v basicu, ki omogočajo igranje precej kompleksnih napevov, ki vsebujejo celo akorde. Nekateri računalniki omogočajo spreminjanje barve glasu, in to omogoča posnemanje glasbil. Z ustreznim programom pa se da tipkovnica računalnika uporabiti kot klaviatura.

...in svetloba

Visoka in nizka ločljivost

Grafika mikroračunalnikov se da razdeliti v dve kategoriji: grafiko z nizko in visoko ločljivostjo. Razlika med njima se najbolje vidi v tem, kako je sestavljen posamezen znak (črka, številka ali poljubna oblika).

Če si natančneje ogledate standarden znak na televizijskem zaslonu, vidite, da je sestavljen iz polja majhnih kvadratkov. Te kvadratke imenujemo grafične točke. Vsak znak, ki se pokaže na zaslonu, je sestavljen iz teh točk. Večina hišnih računalnikov ima znake, sestavljene iz 64 grafičnih točk, ki so urejene v osmih vrsticah, v vsaki vrstici pa je po osem točk. Črka A se da sestaviti tako, kakor kaže slika:

Vsaka osvetljena grafična točka v mreži je lahko zapisana v računalnikovem pomnilniku kot »1«, vsaka zatemnjena grafična točka pa kot »0«. Osem bitov sestavlja zlog, tako da je vsaka vrstica vzorca shranjena v eni lokaciji pomnilnika. Zato potrebujemo osem spominskih lokacij za opis enega znaka.

Grafični prikazovalniki so včasih sestavljeni iz blokov, ki so lahko veliki kot cel znak ali polovica ali četrtina znaka. Grafika, sestavljena iz teh velikih, preprostih blokov ima nizko ločljivost. Veliko hišnih računalnikov pa ima grafiko, kjer so bloki kar grafične točke. Taka grafika ima visoko ločljivost. Dober zgled za prikaz razlike med vrstama grafike je risba sinusne krivulje.

VZOREC BITOV

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

VZOREC GRAFIČNIH TOČK

Tudi če niste izkušen programer, lahko napišete kratke in preproste programe, ki dajejo precej kakovostne glasove. Če želite kar najbolje izrabiti zvočne lastnosti računalnika, lahko dobite na trgu programe, ki omogočajo neposredno pisanje in izvajanje tonov. V vsakem primeru je koristno, če veste, kako vaš računalnik ustvarja, oblikuje in nadzira zvoke, ki jih daje.

Nihajni krog

Nihajni krog je elektronsko vezje, ki daje ponavljajoče se signale. Če te signale ojačimo in pošljemo v zvočnik, dobimo ton določene višine. Hišni računalniki imajo največ po štiri nihajne kroge. Več ko je nihajnih krogov, več not lahko zaigramo.

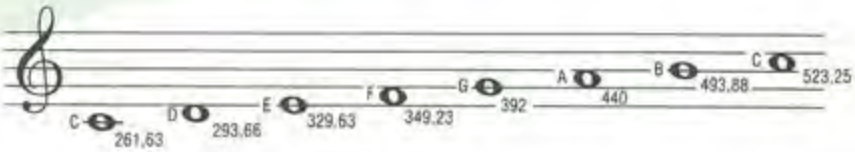
Vsak ton ima tri značilnosti: frekvenco, ovojnico (vsebuje informacijo o jakosti) in obliko signala.

Frekvenca

Frekvenca je najpomembnejša značilnost signala, ki jo moramo kontrolirati, saj določa višino tona. Frekvenca je število ponovitev signala v sekundi in se meri v obhodih na sekundo (Hz). Človekovo uho sliši frekvence, višje od 20 Hz in nižje od 20000 Hz. Čeprav ne moremo slišati frekvenc pod 20 Hz, se te lahko uporabljajo za spremembe lastnosti tonov. Ta tehnika se imenuje modulacija in jo med hišnimi računalniki za zdaj uporablja le Commodore 64.

Načini igranja not na računalniku so zelo različni. Nekateri hišni računalniki imajo ukaze v basicu, ki izračunajo frekvenco, tako da morate določiti le višino ali celo samo glasbeni črkovni simbol — A, A #, B in tako naprej. Pri drugih hišnih računalnikih je igranje težje, saj so na voljo le tabele, kjer so zapisane frekvence določenih not. Te frekvence morate z ukazom POKE shraniti v pomnilnih računalnika. Tabele vsebujejo natančne pretvorbe za lestvico srednjega C. Tabele so uporabne tudi za tiste, ki želijo programirati melodije v strojnem jeziku pri tistih hišnih računalnikih, kjer basic ne more pomagati pri izračunavanju frekvenc.

Note in frekvence
Za vsako noto v lestvici lahko izračunate frekvenco, če množite frekvenco note, ki je en polton pod njo, z 1,0594631. Tak postopek se mogoče zdi zapleten, toda če množenje ponovite 12-krat, se temeljna frekvenca podvoji. V eni oktavi je 12 poltonov, zato podvojitev frekvence premakne ton za eno oktavo navzgor. Ta tabela omogoča natančno pretvorbo notnih simbolov (za lestvico srednjega C) v frekvence.



Posebni znaki

Za ustvarjanje nenavadnih in privlačnih prikazov na zaslonu je pogosto koristno, če imate na voljo znake, ki se razlikujejo od navadnih alfanumeričnih znakov. VIC-20 in Commodore 64 imata poseben nabor grafičnih znakov, ki se dajo uporabljati direktno s tipkovnice, toda celo ti znaki ne vsebujejo vseh mogočih oblik. Večina hišnih raunalnikov omogoča ustvarjanje novih znakov. Običajno se da to doseči s spreminjanjem binarnih vzorcev v osmih lokacijah pomnilnika, kjer je znak shranjen. Pogosto se zgodi, da se stari vzorec izgubi ali zamenja z novim in novi znak prevzame nekatere lastnosti starega. Tako se lahko novi znak uporablja pri ukazu PRINT kar s pritiskom na tipko znaka, ki ga je zamenjal. Tukaj je zgled znaka, ki ga je določil uporabnik, in ustrezna binarna koda:

VZOREC BITOV

128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	1
0	1	0	1	1	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	1	1	0	0	1	1	0

VZOREC GRAFIČNIH TOČK

Način določanja novih znakov je odvisen od računalnika. Za Spectrum uporabnik izvrši ukaz USR in določi binarni vzorec. Na računalniku Commodore 64 mora uporabnik najprej prepisati celotni nabor znakov iz bralnega v delovni pomnilnik in potem z ukazom POKE zapisti osem ustreznih desetiških števil, ki dajo želeni vzorec. Napisanih pa je tudi nekaj podpornih programov, ki omogočajo preprostejše določanje novih znakov.

Za ustvarjanje večjih likov lahko združimo dva ali več znakov. Figure na desni strani so sestavljene iz štirih posebno oblikovanih znakov. Program, ki teče na računalniku Commodore 64, lahko piše skupine znakov v treh barvah. Novi znaki so bili izdelani s kratkim podprogramom, ki premakne nabor normalnih znakov iz bralnega v delovni pomnilnik in zamenja grafične znake ▗▝, ◨, ◪ in ' ' z branjem števil iz podatkovnega stavka in uporabo ukaza POKE za njihov zapis v ustrezne lokacije pomnilnika.

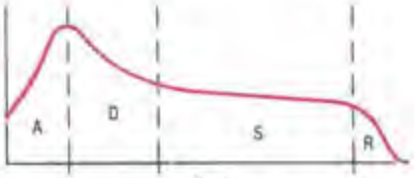
Število znakov, ki so lahko hkrati izpisani na zaslonu, je običajno omejeno tudi, če lahko delamo s figurami (sprites) (glej stran 136), zato so posebni znaki še posebej uporabni takrat, ko je potrebno na zaslonu pokazati več podobnih oblik.

E T
Te raznovrstne figure so sestavljene iz štirih znakov, ki jih je določil programer. Tak način združevanja znakov se da uporabiti na mnogih računalnikih, ki nimajo figur.

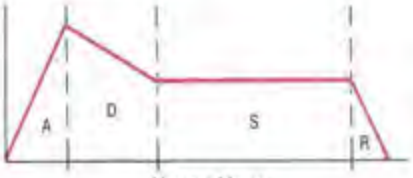
Ovojnica

Kako damo zvoku barvo

Ovojnica zvoka predstavlja potek jakosti zvoka od trenutka, ko se ta oglasi, do konca. Diagram kaže ovojnice not, zaigranih na klavirju in na drugih glasbilih. Ovojnice so ponavadi razdeljene na štiri območja: vzpon (A), umirjanje (D), stalni del (S) in padec (R). Pri klavirju se jakost hitro vzpenja do najvišje točke takoj potem, ko pritisnemo tipko, potem se nekoliko počasneje umiri, ima približno stalno jakost, dokler držimo tipko, in potem hitro pade na ničlo, ko tipko izpustimo. Vzpon, umirjanje in padec so časovni intervali, stalni del pa predstavlja jakost tona. Naprave, ki nadzorujejo vse lastnosti ovojnice, se imenujejo generatorji ADSR (po angleških besedah: attack, decay, sustain in release).



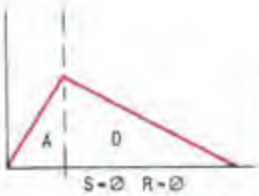
Klavir



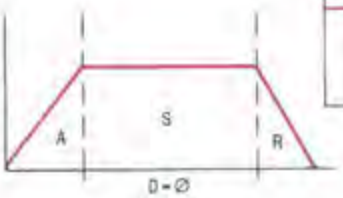
Umetni klavir

Vsaka naprava, ki lahko vključi in izključi ton, je generator ovojnice. Večina računalnikov ima samo generator »bip«, to je stikalo, ki vključi ton stalne jakosti za določen čas. V tem primeru so območja A, D in R enaka ničli.

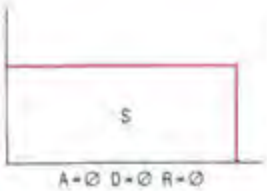
Oric-1 omogoča delno nadzorstvo nad ovojnico, najboljša na tem področju pa sta BBC Micro in Commodore 64, ki imata generator ADSR, ki lahko posnema ovojnice večine običajnih glasbil, in tudi nekaj »nenaravnih« zvokov.



Ksilofon



Violina



Orgle

Igra s svetlobo

Uporaba ukazov POKE in PRINT pri preprosti animaciji

Zdaj, ko že poznamo temeljna načela računalniške grafike, bomo videli, kako se da doseči preprosta animacija. Najprej je potrebno določiti način, kako se znaki pišejo na zaslon in kako se da njihova lega na zaslonu kontrolirati iz programa, pisanega v basicu. Programer ima na voljo dve metodi: ukaza POKE ali PRINT.

Ukaz POKE zapiše število v poljubno lokacijo pomnilnika. Vsak računalnik ima poseben del pomnilnika, kjer je vsaka lokacija povezana z določenim delom zaslona. Pri običajnem zaslonu, ki ima 24 vrstic in 40 kolon, ima v ta namen rezerviranih 1000 lokacij pomnilnika. Vsaka lokacija vsebuje število, ki predstavlja določen znak iz niza, ki jih računalnik pozna. To je lahko standardna koda ASCII, ali pa koda, ki jo določi proizvajalec računalnika. Dodatno k tem lokacijam, kjer so shranjene kode znakov, po navadi obstaja še polje, ki vsebuje informacijo o barvi posameznih znakov.

Commodore 64 ima zelo malo ukazov v basicu, ki lahko pomagajo programerju pri delu z grafiko. Za pisanje po zaslonu se često uporablja ukaz POKE. Naslovi lokacij, ki vsebujejo kodo znakov, so od 1024 do 2023, lokacije, v katerih so informacije o barvah znakov, pa so od 55296 do 56295. Commodore 64 ima črko A opredeljeno s kodo 1, črna barva pa ima kodo 0. Če želimo zapisati črno črko A v zgornji levi kot zaslona, moramo napisati naslednji program:

```
10 POKE 1024,1
20 POKE 55296,0
30 END
```

Če želimo napisati celo vrstico črnih A-jev na zgornjem robu zaslona, moramo program malo spremeniti.

```
10 FOR X = 0 TO 39
20 POKE 1024 + X,1
30 POKE 55296 + X,0
40 NEXT X
50 END
```

A-ji so napisani z zanko FOR...NEXT, ki povečuje lokaciji kode znaka in barve za eno, ob vsakem odhodu. Če vpeljemo ukaz, ki izbriše stari A vedno, ko napiše novega, se bo črka A pomikala po zaslonu. To je preprosta oblika animacije.

Koda za presledek na računalniku Commodore 64 je 32. Zdaj moramo samo postaviti ta znak na pravo lokacijo, eno mesto pred novim A-jem. Vrinimo to vrstico v gornji program

```
35 POKE 1024 + X,32
```

ali

```
15 POKE 1024 + (X—1),32
```

Uporaba ukazov POKE pri delu z grafiko je težaven proces. Ta metoda je verjetno najbolj uporabna za ustvarjanje statičnih ozadij z uporabo ukazov READ in DATA za branje kodnih znakov, preden jih z ukazom POKE zapišemo na prave lokacije..

Večina računalnikov ima veliko grafičnih ukazov, ki so del standardnega basica in omogočajo uporabniku preprosto ustvarjanje raznobarvnih slik. Večinoma so vključeni tudi ukazi za konstruiranje geometrijskih vzorcev. S temi ukazi uporabnik lahko riše na zaslon točke, jih povezuje z ravnimi črtami, riše kvadrate, loke, kroge in lahko pobarva notranjost narisanih likov.

Običajno brez težave zbrišemo stare like tako, da jih pobarvamo enako kakor podlago. Hitro risanje, brisanje in prepisovanje so dejansko temelji preproste animacije. Kakovost animacije je v veliki meri odvisna od hitrosti, s katero lahko izvajamo posamezne faze. Grafični liki so veliko bolj učinkoviti od navadnega risanja, saj ne zahtevajo brisanja starih lokacij, potem ko lik premaknemo. Tako zelo povečamo hitrost njegovega navideznega premikanja. Razvoj modularne grafike omogoča, da pišemo arkadne igre, ki so bile prej lahko pisane le v strojnem jeziku, tudi v basicu.

Načela animacije se lahko uporabijo v povezavi s preprostimi ukazi v basicu. Veliko hišnih računalnikov ima ukaze, ki omogočajo uporabniku pisanje na poljubno lokacijo na zaslonu. Spectrumov ukaz se glasi PRINT AT, na računalniku Dragon pa PRINT@.

Risanje zvezd
To je kratek program za računalnik Dragon. Spremenljivka X je lokacija na zaslonu, kamor želimo narisati zvezdico. Vrstica 40 izbriše staro zvezdico, ko se na zaslonu pokaže nova.

```
10 CLS: REM BRISANJE ZASLONA
20 FOR X = 160 TO 191
30 PRINT@ X,"*"
40 PRINT@X—1, " ";
50 NEXT X
60 END
```

Oblika signala

Oblika signala je odvisna od nihajnega kroga in daje glasu njegovo značilnost. Dva različna instrumenta, ki igrata note iste višine, ne zvenita enako, delno tudi zato ne, ker sta obliki signalov različni. Najpogostejši so oglati signali (impulzi), trikotni signali in žage. Ti signali so narisani na sliki.

Commodore 64 je za zdaj najbolj glasbeni hišni računalnik predvsem zato, ker lahko izbiramo tri temeljne oblike signala na vsakem od treh nihajnih krogov. Oblika signala se da spremeniti z uporabo filtrov, ki plemenitijo barvo glasu. Pomembno je dejstvo, da se dajo filtri nastavljati med trajanjem note. Tako je mogoče doseči natančno posnemanje naravnih glasov in večjo pestrost pri nenaravnih glasovih.

Šum

Šum je kompleksen tip glasu, ki ga povzroči naključno valovanje. Človekovo uho ne more razbrati ponavljajočega se vzorca in ne sliši čistega tona (dež, veter, nevihta). Ti šumi ne zvenijo enako zato, ker so pomešani z nekaterimi nosilnimi toni. Večina mikroprocesorjev, ki imajo možnost generirati šum, omogoča modulacijo šuma s posameznimi nosilnimi toni. Tako se dajo posnemati glasovi od šepetajočega vetra do bučnih eksplozij.

Izhodi

Izhod je po navadi narejen prek televizijskega zvočnika. V tem primeru lahko povežemo televizijo na hi-fi stolp prek videorekorderja. Nekateri računalniki imajo vdelane manjše zvočnike. Pri teh je nemogoče doseči dobro kvaliteto zvoka brez spreminjanja zgradbe računalnika ali nakupa zunanjih dodatkov. Računalnik ima lahko izhod, primeren za direktno povezavo s hi-fi opremo. V tem primeru je trud, vložen v generacijo kompleksnih oblik signalov, bogato poplačan.

Zvok in Vic 20

Podrobnejši pregled proizvajanja zvoka računalnika Vic 20

Vic 20 je eden prvih hišnih računalnikov, zato njegove zmožnosti nekoliko zaostajajo za novejšimi modeli. Tako programska jezika Vic 20 basic kakor tudi Commodore 64 basic nimata posebnih ukazov za zvok. Zvok krmilimo prek zaporedja ukazov POKE. Enako načelo uporablja tudi Commodore 64, tako da so ti napotki uporabni tudi zanj. Krmiljenje zvoka je omejeno na več stopenj glasnosti in frekvence treh oscilatorjev ter glasnosti generatorja šuma. Izhod gre prek zvočnika na televizorju. Zaradi nepraktičnega načina izbiranja frekvenc jih je težko uglasiti na note glasbene lestvice.

Zaradi vsega tega računalnik Vic 20 ni primeren za programiranje glasbe, dasi se dajo te omejitve preseči s premislekom, potrpežljivostjo in poznavanjem programskega jezika basic.

Krmiljenje zvoka

Računalnik Vic 20 je opremljen s tremi impulznimi oscilatorji in z generatorjem šuma. Vsak oscilator pokriva približno tri oktave, razpredelnica kaže frekvence posameznih oktav.

Ta razdelitev nam omogoča, da pokrijemo pet oktav z vsaj enim oscilatorjem v vsaki od njih. Oktava 3 je dostopna s katerimkoli od treh oscilatorjev.

Osc. 1	Osc. 2	Osc. 3	Frekv. obm. (Hz)	Oktava
●			(65,41 – 123,47)	1
●	●		(130,81 – 246,94)	2
●	●	●	(261,63 – 493,88)	3
	●	●	(523,25 – 987,77)	4
		●	(1064,5 – 1975,53)	5

Oscilatorje krmilimo s spreminjanjem vsebine petih pomnilniških lokacij takole:

Pomnilniška lokacija	Oscilator
POKE 36874,X	1
POKE 36875,X	2
POKE 36876,X	3
POKE 36877,X	šum

X je celo število med 135 in 241 (0 izključi oscilator), ki se nanaša na tabelo tonskih vrednosti na 73. strani knjižice z navodili za uporabo, ki je priložena računalniku Vic 20. Preden izberemo frekvenco tona, ki ga želimo slišati, moramo izbrati njegovo glasnost z ukazom:

```
POKE 36878,V
```

kjer je V celo število med 0 (izključeno) in 15 (glasno), ki se nanaša na vse tri oscilatorje in šum. Na primer:

```
POKE 36874,219:POKE 36875,219:POKE
36876,219:POKE 36878,7
```

Zazvenel bo ton, ki pripada noti A pri 440 Hz na oscilatorju 1, dalje ton, ki pripada noti A za eno oktavo više na oscilatorju 2, in še za eno oktavo više na oscilatorju 3, vsi srednje glasno 7. Na koncu ne smemo pozabiti postaviti vse te pomnilniške lokacije na nič z ukazom POKE, da izključimo oscilatorje.

Note in premori

V tem razdelku si oglejmo, kako določimo trajanje tonov in premorov. Na voljo imamo dve metodi. Pri prvi uporabimo zanko FOR...NEXT, kjer je dolžina tona določena z dolgo prazno zanko, kakor kaže naslednji zgled:

```
10 POKE 36878,7
20 POKE 36876,203
30 FOR P=1 TO 200
40 NEXT P
50 POKE 36878,0
60 POKE 36876,0
```

Po ukazu RUN zaslišimo ton, ki pripada noti D#(dis). Toda ta metoda zahteva skrbno merjenje notranjega časa zanke. Preprostejša in bolj elegantna metoda je uporaba že vdelane ure računalnika Vic 20, ki ima 60 obhodov na sekundo. Takt ure prenesemo v program s spremenljivko TI. Tako lahko natančno določimo dolžino časovnih intervalov, kot na primer:

```
10 POKE 36878,7
20 POKE 36876,203:D=TI
30 IF TI-D<15 THEN 30
40 POKE 36878,0
50 POKE 36876,0
```

V tem primeru zaslišimo po ukazu RUN enak ton kakor prej, le da tu vemo, da traja 15 taktov ure, to je četrt sekunde.
Spremenljivka D dobi vrednost TI v trenutku, ko se vključi zvok. Dalje vrstica 30 odšteje 15 obhodov, preden se izvrši vrstica 40, ki izključi zvok.

Računalnik Dragon 32

...in njegove grafične zmožnosti

Računalnik Dragon 32 razume posebno zvrst programskega jezika basic, znano pod imenom »Microsoft Extended Colour Basic«. Še več drugih računalnikov je osnovanih na tej zvrsti basica. Microsoft basic se uporablja preprosto in ima bogato zbirko ukazov za grafiko, na primer ukaze za risanje črt, krogov in drugih geometrijskih likov. Ko so liki narisani, jih lahko obarvamo. Tako dosežemo bogate grafične učinke z majhnim programskim naporom.

Računalnik Dragon 32 ima sedem načinov grafične ločljivosti. Delamo lahko z zaslonom, razdeljenim na 512 točk (v tem primeru so to znakovne celice), pri najnižji ločljivosti, in na 49125 točk pri najvišji ločljivosti. Na voljo je osem barv, vendar je pri visokih ločljivostih izbira omejena le na štiri ali celo le dve barvi.

Oblike grafične ločljivosti

Zanke lahko pišemo na zaslon v 16 vrsticah in 32 stolpcih. Ta razdelitev določi nizko ločljivost. Ukaz PRINT nam omogoča, da katerikoli znak iz nabora postavimo na katerokoli od 512 znakovnih celic na zaslonu. Poleg običajnega nabora črk in standardnih znakov imamo še 16 grafičnih znakov nizke ločljivosti, ki jih lahko obarvamo v osmih barvah. Naslednja oblika grafične ločljivosti nam razdeli zaslon na 32 vrstic in 64 stolpcev. Z ukazom SET izrišemo kvadratek, katerega stranica meri četrtino znakovne celice, na poljubno mesto na zaslonu. Kvadratki se brišejo z ukazom RESET.

Obe zgoraj opisani ločljivosti lahko uporabljamo hkrati. Dalje imamo še pet oblik visoke ločljivosti, vendar jih ne moremo uporabljati skupaj z opisanima oblikama nizke ločljivosti. Teh pet oblik visoke ločljivosti nam ponuja izbiro števila vrstic in stolpcev ter števila barv z ukazom PMODE na način, kakor kaže naslednja tabela:

PMODE	Ločljivost	Število barv
0	128*96	2
1	128*96	4
2	128*192	2
3	128*192	4
4	256*192	2

Seveda visoka ločljivost in večje število barv porabi več prostora v pomnilniku, in če pišemo daljše programe ter uporabljamo visoko ločljivost, moramo paziti, da nam v pomnilniku ne bi zmanjkalo prostora.

Četudi imamo pri visoki ločljivosti le omejeno število barv, to ni nič hudega, ker lahko izbiramo med dvema naboroma barv. To dosežemo z ukazom SCREEN. Na primer ukaz SCREEN 1,0 izbere visoko ločljivost in določi prvi nabor barv, medtem ko ukaz SCREEN 1,1 izbere alternativni nabor barv.

Računalnik Dragon 32 ni predrag in ima precej ukazov za grafiko, vendar je bolj primeren za statične kakor za hitro gibajoče se slike. Njegova največja pomanjkljivost je, da ni mogoče uporabljati hkrati grafike visoke in nizke ločljivosti. Tako v visoki ločljivosti ni mogoče risati diagramov z označenimi enotami in besedilom.

PAINT
Ta ukaz je zelo uporaben in nam omogoča risanje zanimivih slik na zaslonu. Ta ukaz povzroči, da računalnik pobarva zaslon od dane točke do prve mejne črte, ki jo doseže. To pomeni, da lahko pobarvamo notranjost poljubne zaprte krivulje, narisane na zaslonu, na primer kroga, trikotnika itd.

DRAW
Ukaz DRAW posnema gibanje peresa po zaslonu, torej sam omogoča, da narišemo iz dane točke črto v poljubni smeri. Sliko, narisano s tem ukazom, je mogoče zavrteti ali povečati.

GET in PUT
Ukaz GET povzroči, da računalnik shrani vsebino zaslona v pomnilnik, medtem ko jo ukaz PUT prikliče nazaj.

PSET in PRESET
Ta dva ukaza za visoko ločljivost ustrezata ukazoma SET in RESET za nizko ločljivost, ki smo ju omenili zgoraj. Torej z njima lahko prižgemo ali ugasimo katerokoli od slikovnih točk in ji določimo barvo.

LINE
Ta ukaz poveže dve točki z ravno črto visoke ločljivosti.

CIRCLE
Ta ukaz nariše na zaslon krožnico visoke ločljivosti z danim radijem in središčem. Mogoče je risati tudi krožne loke in napraviti razteg v dani smeri to omogoča risanje elipse.

Ukazi za barve
Takšno sliko je na računalniku Dragon 32 mogoče doseči z uporabo le nekaj ukazov visoke ločljivosti.

Visoka ločljivost
Tu je zgled kratkega programa za računalnik Dragon 32, ki prikazuje nekaj njegovih grafičnih zmožnosti. Program uporablja ukaz PMODE 3, ki ne postavlja najvišje ločljivosti, vendar dopušča pestrejše barve.

```
10 PCLS:PMODE3,1
20 SCREEN1,0
30 COLOUR0,1
40 FOR X=0 TO 127 STEP 10
50 LINE(X,85)-(127,
   85-X/3), PSET
60 LINE (X,85)-(127,
   85+X/3), PSET
70 LINE(255-X,85)-(127,
   85-X/3), PSET
80 LINE(255-X,85)-(127,
   85+X/3), PSET
90 NEXT X
100 CIRCLE(127,85),128,4,0,3
110 CIRCLE(127,85)30,4,3
120 PAINT(130,30),3,4
130 PAINT(130,130),3,4
140 GOTO 140
150 END
```

Preprosti zvoki

Proizvajanje zvoka z računalnikom Sinclair Spectrum

Spectrum je hitro postal najpopularnejši hišni računalnik. Njegova odlika je odlična barvna grafika ter primerna izbira dodatnih pomnilniških enot, in to po nizki ceni. Na žalost je bilo zaradi nizke cene marsikaj poenostavljeno. Večina pripomb gre na račun tipkovnice, nestandardnega basica, njegova najšibkejša točka pa je zvok, ker je opremljen z enim samim impulznim oscilatorjem. Tako moremo krmiliti dolžino in višino tona, vendar ni mogoče spremljati njegovega zvena (barve), to je ovojnice glasnosti. Naslednja pomanjkljivost je izhod, zelo majhen notranji piezoelektrični zvočnik, ki daje precej hreščeče tone. Sinclair sicer omogoča drug izhod signala prek vtičnic »mic« (mikrofon) ali »ear« (slušalka) kasetofona, ki se da ojačati na hi-fi napravi, toda zaradi slabe kvalitete osnovnega tona je ta prednost dvomljiva.

Pač pa so Spectrumova prednost pri proizvajanju zvoka preprosti basicovi ukazi zanj. To sta ukaza BEEP in PAUSE, ki omogočata lažje razumevanje temeljnih načel računalniškega proizvajanja zvoka. Spectrum ima zelo velik razpon frekvenc, ki obsega kar 10 oktav.

Če hočemo, da se bo Spectrum sploh oglasil, ga moramo najprej priključiti na hi-fi napravo ali vključiti notranji zvočnik z naslednjim ukazom:

```
POKE 23609,100
```

Po tem ukazu postane »klik« (zvok, ki ga sproži računalnik ob pritisku na tipko) bolj slišen.

Krmiljenje zvoka

Računalniku ukažemo, da proizvede določen zvok, takole:

```
BEEP d,n
```

Tu parameter d predstavlja dolžino tona in parameter n njegovo višino. Ton lahko traja od 0,00125 do 10 sekund. Njegova višina je izražena v poltonih od srednjega (enočrtnega) C na sredini klavirske klaviature (kateremu pripada vrednost 0) in daje v razponu od −60 do +69. Na primer, na naslednji ukaz se bo računalnik oglasil s tonom, ki pripada noti A pri 440 Hz, ki je za 9 poltonov višji od enočrtnega C in bo trajal pol sekunde:

```
BEEP 5,9
```

Če želimo napraviti premor med tonoma, napravimo to z ukazom PAUSE. Na primer:

```
PAUSE ms
```

Prve slike

Grafične zmožnosti računalnika Commodore Vic-20

Kakor drugi Commodorjevi hišni računalniki je tudi Vic-20 dobro izdelan, vendar nima dosti več od običajnega nabora basicovih ukazov. Nima posebnih ukazov za grafiko visoke ločljivosti. Potrebno je dobro poznati zgradbo in delovanje računalnika, ali pa ga razširiti z dodatno opremo v obliki vtičnega modula, ki poenostavi grafične programe. V nizki ločljivosti nam je na voljo bogata zbirka posebnih grafičnih znakov, ki jih z malo iznajdljivosti sestavimo in dosežemo zanimive grafične učinke. Uporabljamo lahko šestnajst barv, pri tem ena celica lahko vsebuje največ štiri različne barve. Zaslon je razdeljen na 23 vrstic in 22 stolpcev. Vsaka celica pa je sestavljena iz 18 krat 8 točk. Znaki se lahko izpišejo tudi v pravokotnem formatu 16 krat 8 točk. Na računalniku Vic-20 je sicer mogoča grafika visoke ločljivosti, vendar je povprečnemu uporabniku težko dostopna brez vtičnega modula, ki smo ga že omenili.

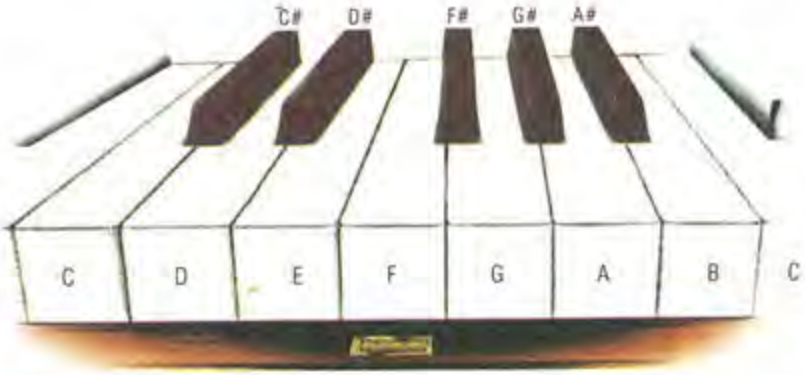
Zmožnosti nizke ločljivosti

Na voljo imamo velike in male črke, druge standardne znake ter več ko 60 posebnih grafičnih znakov PET. Na prednji strani mnogih tipk sta narisana dva majhna kvadratka z vzorcem. Z uporabo teh tipk se da dani

kjer parameter ms predstavlja dolžino trajanja premora v enotah po 0,001 sekunde (v milisekundah). Če želimo zaigrati C-durovo lestvico (C, D, E, F, G, A, B, C)* od enočrtnega C, tako da vsak ton traja po pol sekunde in s premorom četrt sekunde med dvema tonoma, napravimo to z naslednjim programom:

```
10 FOR I=1 TO 8
20 READ N
30 PAUSE 250
50 NEXT I
60 DATA 0,2,4,5
70 DATA 7,9,11,12
```

Ta oktava se razteza od srednjega (enočrtnega) C v razponu 12 poltonov. Imenuje se oktava, ker jo sestavlja osem tonov v durovi lestvici.



Lestvice in klaviature

Z uporabo ukaza INKEY$ lahko Spectrumovo tipkovnico uporabimo kot klaviaturo:

```
 10 REM **************
 20 REM * OKTAVA KLAVIRJA*
 30 REM **************
 40 IF INKEY$ = "Q"THEN BEEP1,0
 50 IF INKEY$ = "2"THEN BEEP1,1
 60 IF INKEY$ = "W"THEN BEEP1,2
 70 IF INKEY$ = "3"THEN BEEP1,3
 80 IF INKEY$ = "E"THEN BEEP1,4
 90 IF INKEY$ = "R"THEN BEEP1,5
100 IF INKEY$ = "5"THEN BEEP1,6
110 IF INKEY$ = "T"THEN BEEP1,7
120 IF INKEY$ = "6"THEN BEEP1,8
130 IF INKEY$ = "Y"THEN BEEP1,9
140 IF INKEY$ = "7"THEN BEEP1,10
150 IF INKEY$ = "U"THEN BEEP1,11
160 IF INKEY$ = "I"THEN BEEP1,12
170 GOTO 40
```

Opomba: Nota H je v tej knjigi pisana s črko B. Pri prevajanju je nismo spremenili, ker so nekateri ukazi vezani na to označbo.

Če bi želeli boljši »klavir«, bi lahko program še precej izboljšali. Kljub temu pa je Spectrum primeren pripomoček pri pouku glasbe. Edini način za izboljšanje zvoka je nakup dodatne opreme.

vzorec izrisati na zaslon. Poleg kvadratkov, ki merijo pol ali četrt celice, je tu več drugih simbolov, kot so simboli igralnih kart, kockasti vzorci, krožci in številni drugi znaki, ki, izbrani v primernem zaporedju, omogočajo risanje pestrih oblik na zaslon, na primer grafov, tabel, krepkih velikih črk in drugega.

Vsak od teh znakov se lahko izriše v komplementarni obliki (črno na belem namesto belo na črnem), kar seveda še popestri zbirko. S potrpežljivostjo in domišljijo se dajo na zaslon narisati dobre slike.

Znake izpišemo na zaslon z ukazom PRINT ali pa z ukazom POKE zapišemo ustrezne kode na pomnilniške lokacije, ki določajo lego na zaslonu in barve znaka. Commodorjeva izvedba ukaza PRINT je izredno močna, saj z njim lahko določimo barvo vsakega znaka posebej. S tem ukazom lahko krmilimo tudi pomik kurzorja, tako da brez težav izrišemo premikajoče se slike. Uporaba ukaza POKE je bolj zapletena in počasnejša, vendar je v nekaterih okoliščinah koristna.

Visoka ločljivost

Na nerazširjenem računalniku Vic-20 je mogoča grafika visoke ločljivosti, vendar pomnilnik zadošča le za polovico zaslona. To grafiko dosežemo s preslikavo bitov (bit mapping). S to tehniko posebej krmilimo vsako točko znotraj danega območja na zaslonu. Vsaka celica v mreži 23 vrstic in 22 stolpcev vsebuje 64 slikovnih točk, razvrščenih v osem vrstic in osem stolpcev. Na drugi strani pa v posebnem območju pomnilnika, ki je namenjen zaslonu, vsak bit predstavlja eno slikovnot točko. Če je bit postavljen na 1, je ustrezna točka na zaslonu prižgana, če je postavljen na 0, je točka ugasnjena. Z ukazom POKE spreminjamo vsebino tega dela pomnilnika in s tem prižigamo in ugašamo točke na zaslonu.

Vtični modul (Super Expander Cartridge)

Programiranje grafike visoke ločljivosti s preslikavo bitov je dolg in zamuden postopek. Druga možnost je nakup Commodorjevega vtičnega modula (Super Expander Cartridge), ki omogoča dostop grafike visoke ločljivosti in zvoka prek ukazov v basicu. Ukazi grafike visoke ločljivosti so: GRAPHIC, ki postavlja grafični način, POINT, ki nariše točko na zaslonu, in PAINT, ki obarva območje na zaslonu. Vendar nimamo ukaza »UNPOINT«, ki bi nam izbrisal že narisano točko. Z ukazom PAINT ni mogoče barvati likov na obeh straneh diagonale, ne da bi pokvarili diagonalo samo.

Grafika nizke ločljivosti računalnika Vic-20 je dobro napravljena in mnogostranska, medtem ko je grafika visoke ločljivosti težko uporabna brez vtičnega modula.

Zajček
Ta program nam predstavlja raznovrstnost in uporabnost posebnih Commodorejevih znakov. Program je napisan za računalnik Vic-20. Risba zajčkove glave je bila napravljena iz že obstoječih znakov (brez definiranja novih).

Zvočni sistemi

Drugi pregled zvočnih zmožnosti računalnika Vic 20

Doslej smo si v poglavju zvok in svetloba ogledali, kako krmilimo tri oscilatorje računalnika Vic 20 s spreminjanjem vsebine določenih pomnilniških lokacij z ukazom POKE in kako nastavimo glasnost, frekvenco in trajanje tona. Videli smo, da lahko trajanje tona ali premora med dvema tonoma določimo z zanko FOR...NEXT, ali pa, učinkoviteje, s štetjem »obhodov« njegove notranje ure, ki ima 60 »obhodov« na sekundo. S sestavljanjem teh glasbenih elementov — frekvence, glasnosti in takta — je mogoče oblikovati preproste melodije in druge zvočne učinke.

Igranje melodije

Da bi sestavili program za melodijo, moramo najprej določiti ustrezne note. Za zgled si oglejmo prvo vrstico pesmice »Oh, I do like to be beside the seaside«. Pravilno zaporedje not je naslednje:

D# E F D# C A# G# G G# D# D#

Dolžino tonov in premorov bomo določili s spremenljivko TI. Ko bomo pognali naslednji program, se bo do zaslišali toni te melodije. V programu smo uporabljali spremenljivke, da so bili izbrani ukazi POKE preprostejši.

```
10 V=36878
20 FOR I=1 TO 11
30 READ N: REM *NOTA*
40 POKE V,7: P=TI: REM *GLASNOST*
50 IF TI-P <15 THEN 50: REM *PREMOR*
60 POKE V-3,N: D=TI: REM *NOTA*
70 IF TI-D<20 THEN 70: REM *TRAJANJE*
80 POKE V-3,0: REM *KONEC*
```

Svetlobni program

Prvi koraki z grafiko računalnika BBC micro

Računalnik BBC micro je eden najpopularnejših hišnih računalnikov. Z nekaj programskimi vrsticami v basicu lahko dosežemo osupljive grafične učinke in tudi hitrost risanja je presenetljiva.

V BBC basicu imamo več ukazov za grafiko visoke ločljivosti, kot na primer: za risanje ravne črte, točke ter risanje in barvanje trikotnikov. Zadnji ukaz se uporablja za barvanje likov, sestavljenih iz trikotnikov, ker nimamo na voljo ukaza, kakor je PAINT. Basic računalnika BBC micro nima na voljo ukazov za risanje krogov in elips in tudi nima grafičnih likov (sprite). Po drugi strani pa je zmožen marsičesa, česar njegovi tekmeci ne znajo. Na primer: na zaslonu lahko prikazuje hkrati besedilo in grafiko visoke ločljivosti z ločenim krmiljenjem kurzorja za grafiko in besedilo. Iz basica pa nam je dostopen tudi operacijski sistem, ki krmili sliko na zaslonu. To je doseženo z naborom ukazov VDU ali ukazov za zaslon. Na zaslonu lahko določimo okna za besedilo in grafiko tako, da imamo ločeni področji za besedilo in za grafiko. Vsakemu od teh področij lahko definiramo barvo in ju lahko ločeno tudi brišemo.

Načini izpisa na zaslon

Računalnik BBC micro ima osem načinov izpisa na zaslon. V treh izmed njih lahko izpisujemo na zaslon samo besedilo. Lahko izbiramo med 20, 40 in 80 znaki v vrstici, odvisno od tega, kateri način izberemo. Glede na izbrani grafični način lahko uporabljamo po dve, štiri ali šestnajst barv. Ugodno pa je, da v načinih, kjer imamo omejeno število barv, to je štiri ali samo dve barvi, le-te niso določene, ampak jih izberemo izmed šestnajstih mogočih.

Način sedem (MODE 7) se razlikuje od drugih, ker ne uporablja standardnih ukazov ASCII. Namesto njih izpis uporablja znake videoteksa. Običajni grafični ukazi, kakor sta PLOT in DRAW, v tem načinu (MODE 7) ne delujejo.

Naslednja tabela prikazuje ločljivost in izbiro barv v vsakem od grafičnih načinov.

Način	Besedilo	Grafika	Barve
0	80 × 32	640 × 256	2
1	40 × 32	320 × 256	4
2	20 × 32	160 × 256	16
3	80 × 25		2 (črna in bela)
4	40 × 32	320 × 256	2
5	20 × 32	160 × 256	4
6	40 × 25		2 (črna in bela)
7	40 × 25		videoteks

```
90 NEXT I
100 DATA 203,207,209,203: REM *ŠTEVILČNE
    VREDNOSTI NOT*
110 DATA 195,187,179,175,179,203,203
120 POKE V,0: REM *UGASNE*
130 END
```

Ta program zaigra note v danem vrstnem redu z enakim trajanjem tonov in premorov med njimi. Zato melodija zveni nekoliko togo. Seveda pa je mogoče izbirati različne dolžine tonov in premorov — s tem dosežemo, da melodija zveni bolj uglašeno.

Zvočni učinki

Z uporabo dveh ali treh oscilatorjev je mogoče zaigrati preproste akorde. Naslednji program bo zaigral akord v C-duru (F#, A in D), začenši s tonom, ki pripada noti F#(fis), temu se bo čez nekaj trenutkov pridružil ton A in nato D; vsak bo trajal po eno sekundo. Akord bo zvenel dve sekundi.

```
10 POKE 36878,7
20 POKE 36874,233:D = TI
30 IF TI—D < 60 THEN 30
40 POKE 36875,219:D = TI
50 IF TI—D < 60 THEN 50
60 POKE 36875,147:D = TI
70 IF TI—D < 120 THEN 70
80 POKE 36878,0: POKE 36874,0
90 POKE 36875,0: POKE 36876,0
100 END
```

Veliko je še mogoče napraviti, da bo zven teh zvokov postal zanimivejši. Njegovo glasnost lahko med zvenenjem spreminjamo, tako da raste ali pada v odvisnost od spremenljivke. Na primer:

```
100 V = 36878
110 FOR I = 1 TO 12
120 POKE V,I
130 NEXT I
140 POKE V,0
```

Glasnost tona bo po tem programu rastla od 1. do 12. stopnje (na voljo imamo stopnje od 0 do 15). Dosežemo lahko, da se glasnost lahko spreminja, dalje lahko med zvenenjem tona spreminjamo njegovo frekvenco, kar se zgodi, če vrstico 120 gornjega programa zamenjamo s:

```
POKE V—3,203 + I
```

Z različnimi kombinacijami frekvenc, glasnosti in šuma lahko dosežemo prijetnejše zvoke.

Podali smo nekaj najpreprostejših napotkov, kako na računalniku Vic 20 krmilimo zvok. Glavni problem pri tem računalniku je pomanjkanje ukazov za zvok, kar pomeni, da moramo uporabljati komplicirane ukaze v basicu že za najnavadnejše zvočne učinke. Posledica tega so dolge programske enote, ki jih interpreter ne more prevajati dovolj hitro, zato so včasih premori med notami daljši od programiranih vrednosti. Temu se lahko ognemo z že omenjenim vtičnim modulom (Super Expander Cartridge).

Pri grafiki z visoko ločljivostjo ima zaslon izhodišče v gornjem levem kotu ne glede na to, za kateri način gre. Navpične vrednosti naraščajo od zgoraj navzdol (0 do 1023), medtem ko vodoravne od leve proti desni (0 do 1279). Če se med izvajanjem programa spremeni način, se prejšnja slika izbriše.

Barve ozadja in besedila se nastavijo z ukazom COLOUR in GCOL. Računalnik BBC mirco dopušča, da izmed 16 mogočih barv izberemo določeno število barv, v ta namen uporablja zvezo med logičnimi dejanskimi barvami. Recimo v načinu MODE 0, ki dopušča samo dve barvi, logično barvo 1, ki je barva ospredja, in logično barvo 0, ki je barva ozadja. Barva 0 je črna in 1 je bela, če prej ne izberemo drugače. Z ukazom COLOUR izbiramo med njima. Ukaz VDU 19 določi logične barve. Na primer, če želimo, da bi bila logična barva 1 zelena (ta nosi številko 2), potem to napravimo takole:

```
VDU 19,1,2,0,0,0
```

Tri ničle, ki sledijo, nimajo pomena, rezervirane so za poznejšo razširitev sistema. Tako bo po gornjem ukaz COLOUR 1 izbral zeleno barvo.

Ukaz GCOL ima pridružena dva parametra. Drugi se nanaša na številko logične barve, prvi pa na način, kako to barvo uporabimo. V ukazu GCOL a, b ima parameter a neko vrednost med 0 in 4, ki določi, kako bomo dano točko ali črto izbrisali glede na barvo podlage; z drugimi besedami, izbere logično operacijo med barvo podlage in barvo točke oziroma črte, s katero dobimo barvo.

Na voljo so operacije AND, OR, ekskluzivni OR in komplementiranje barvne podlage.

Kvalitete zvoka

Proizvajanje zvoka na računalniku BBC Model B

Zaradi izrednih zvočnih zmožnosti in širokega izbora basicovih ukazov za zvok je računalnik BBC micro med najprimernejšimi računalniki za zvočno programiranje. Opremljen je s tremi neodvisnimi impulznimi oscilatorji in z osmimi vrstami šuma, štirimi neodvisnimi ADSR ovojnicami. Lahko skladamo glasbo z do tremi toni v harmoniji, in posebni ukazi nam zagotavljajo, da bodo toni v akordih res zveneli v natanko istem času.

Proizvajanje zvoka

V najpreprostejši obliki zvok proizvajamo z ukazom SOUND.

SOUND C,V,P,D

Tu parametri pomenijo:

C = številka oscilatorja
V = jakost tona
P = višina tona
D= trajanje tona

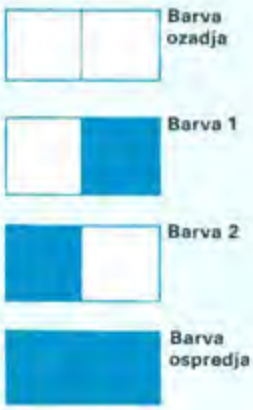
Oscilatorji nosijo številke od 1 do 3 in šum ima številko 0. Glasnost nastavljamo v razponu od 0 do —15. Višina je podana v četrtinah note v korakih med 0 (ton, ki pripada noti A# pri 116,5Hz) in 255 (nota D pri 4698,64 Hz), kar je razpon petih oktav in pol. Srednji (enočrtni) C nosi številko 28. Šum se lahko oglaša na osem načinov. Nastavljamo ga s parametrom P v ukazu SOUND, ki lahko dobi vrednosti od 0 do 7.

Število	Tip šuma
0	Visok tresoč se ton
1	Srednji tresoč se ton
2	Nizek tresoč se ton
3	Tresoč se ton, katerega višina se spreminja s 1. oscilatorjem.
4	Visok ton
5	Srednji ton
6	Nizek ton
7	Višina se spreminja s 1. oscilatorjem.

Trajanje tona krmilimo v stopnjah po 1/20 sekunde od 1 do 255, kar nam da največjo dolžino tona 12,75

Svetlobna zgradba

Grafične zmožnosti računalnika Commodore 64



Računalnik Commodore 64 ima veliko možnosti za grafično programiranje. Njegova glavna pomanjkljivost — enaka kot pri računalniku Vic-20 — je ta, da ima le zelo omejeno število basicovih ukazov za grafiko. To sta PEEK in POKE. Ta dva nam sicer omogočata dostop do pomnilniških lokacij računalnika, vendar taki programi niso pregledni. Tudi temu računalniku, kakor pri Vic-20, lahko dokupimo vtični modul, ki nam omogoča preprostejše programiranje grafike. Sama družba Commodore pa nam ponuja vtični modul, ki omogoča programiranje v Simonovem basicu, ki je obogaten z velikim številom ukazov za grafiko in zvok.

Računalnik Commodore 64 ima standardne velike in male črke ter druge standardne znake, ki se lahko izpišejo na zaslon na navaden ali na komplementaren način. Poleg tega ima še druge grafične znake, ki smo jih že omenili, ko smo govorili o računalniki Vic-20.

Zaslon je razdeljen na 40 stolpcev in 25 vrstic. Znake izpišemo na zaslon z ukazom PRINT, medtem ko

sekunde. Če želimo, da se nam računalnik oglasi s tonom, ki pripada noti A nad enočrtnim C s prvim oscilatorjem, s srednjo glasnostjo −7, za pol sekunde, mu moramo ukazati takole:

```
SOUND 1,−7,64,10
```

Če računalnik dobi naslednji ukaz SOUND, še preden je izvršil prejšnjega, ga postavi v vrsto, da tam počaka, da se izvrši prejšnji ukaz.

Ukaz SOUND lahko razširimo, tako da spremenimo njegov format:

```
SOUND & HSFC,V,P,D
```

V tem primeru znak & pomeni, da je število, ki mu sledi, zapisano v šestnajstiškem številskem sestavu. H je lahko 0 ali 1. Če je H = 1, potem se bo ukaz izvršil šele, ko bo izzvenel prejšnji ton. Če je H = 0 IN, prejšnji ton počasi izzveneva, računalnik pa bo napačno razumel konec prejšnjega tona in bo v kanal poslal novega, ko bo prejšnji še zvenel. Če je H = 1, se to ne zgodi. Če je H = 1, računalnik ne upošteva nobenega drugega parametra v ukazu SOUND& razen številke kanala (oscilatorja) C. S določi število not, ki bodo zazvenele hkrati na različnih oscilatorjih v akordu. Če je S = 0, se bo ukaz izvršil normalno. Če je S = 1, bo zazvenel še eden od ostalih dveh oscilatorjev, če pa je S = 2, bodo zazveneli vsi trije oscilatorji. Na posameznih oscilatorjih zazvenijo toni, ki čakajo v vrsti v posameznih kanalih. Konstrukcija tega ukaza je taka, da nam zagotavlja, da vsi toni v akordu res zazvenijo istočasno.

F je lahko 0 ali 1. Če je F = 0, nima nobenega učinka, če pa je F = 1, potem ta ukaz prekine izvajanje note, izbriše vse note, ki so v vrsti, in prične izvajanje note, ki je določena z novim ukazom. Ostali parametri imajo isti pomen kakor pri ukazu SOUND.

Delovanje ukaza SOUND najbolje ponazarja naslednji zgled. Ta program bo zaigral prvo vrstico pesmice »Happy birthday to you« v F# (nad srednjim C) ključu. Note so naslednje

```
F# F# G# F# B A#
```

V programu je vključen tudi trojni veliki akord zadnjega A#, ki je sestavljen iz D, F in A#.

```
10 SOUND 1,−7,40,20:REM*1ST F#*
20 SOUND 1,−7,40,10:REM*2ND SHORT F#*
30 FOR I=1 TO 3: READ N
40 SOUND 1,−7,N,20: NEXT I: REM*G#F#B*
50 SOUND &201,−7,32,30:REM*D*
60 SOUND &202,−7,38,30:REM*F*
70 SOUND &203,−7,48,30:REM*A#
80 DATA 44,40,50
90 END
```

Zvočne zmožnosti tega računalnika, dane z ukazom ENVELOPE, bomo obravnavali pozneje.

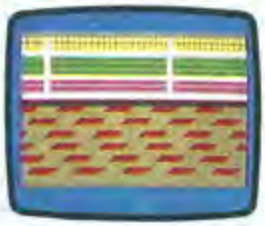
barvo izbiramo z zapisom kod v določene pomnilniške lokacije z uporabo ukaza POKE. Podrobnejši opis je podan v priročniku za računalnik. Posebni grafični znaki nam omogočajo, da lahko izrišemo na zaslon dovolj pestre oblike z uporabo grafike nizke ločljivosti. Na voljo imamo več ko 60 posebnih PET grafičnih znakov; na zaslon jih lahko zapišemo tudi v komplementarni obliki, to nam podvoji število znakov. Tako imamo precej čez 120 različnih znakov za risanje slik. Če ne najdemo primernega znaka, lahko definiramo novega. To pa ni tako lahko. Najprej moramo preslikati vse obstoječe znake, zakodirane v pomnilniku ROM, v pomnilnik RAM. Nato nadomestimo znak, ki ga ne potrebujemo, z novim, ali pa definiramo nov znak. To napravimo tako, da z ukazom POKE zapišemo na določene pomnilniške lokacije kode novega znaka.

Znak je določen z matriko z 8 krat 8 biti, razporejenimi v osem vrstic in osem stolpcev. Enka predstavlja točko, ki ima barve znaka, ničla pa točko, ki ima barvo ozadja. Računalnik Commodore 64 nam dopušča, da znak izpišemo v več barvah (do štiri). V tem primeru pa je vsaka točka znaka določena z dvema bitoma. Mogoče so štiri kombinacije ničel in enic v dveh bitih, in to se uporablja za predstavitev štirih različnih barv znotraj znaka. V tem večbarvnem načinu pa se število vseh mogočih barv zmanjša na polovico, to je od 16 na 8. Naslednja pomanjkljivost tega načina je, da morata biti dve barvi skupni vsem znakom na zaslonu, in tudi vodoravna ločljivost se zmanjša za faktor 2.

V Commodorjevem basicu ni ukazov za grafiko visoke ločljivosti. Za takšno grafiko lahko uporabljamo metodo preslikave bitov (bit mapping), kakor smo že omenili pri pregledu grafičnih zmožnosti računalnika Vic-20. Seveda pa je preprostejši nakup vtičnega modula s Simonovim basicom. V grafiki visoke ločljivosti je zaslon razdeljen na 200 vrstic s po 320 točkami. Večbarvni način je mogoč tudi v tej grafiki, vendar z omejitvijo, da ne moremo uporabljati več ko štiri barve v eni znakovni celici.

Pred nami je kratek program, ki naj nam pričara na zaslon prizor iz veleblagovnice. Pozneje bomo govorili še o grafičnih likih, ki jih je mogoče premikati (sprites), in podrobneje o Simonovem basicu.

Prvi korak
Ta statični prizor iz veleblagovnice je bil napravljen z grafiko nizke ločljivosti. Pozneje bomo temu prizoru dodali še premična grafična lika, voziček in žensko, ki ga potiska.



```
3000 REM ** SUPERMARKET **
3010 PRINT"□"
3020 POKE53280,6:POKE53281,12
3030 PRINT"■■■
```

Preizkus zvoka

Sinteza zvoka na računalniku Dragon 32

Računalnik Dragon 32 je opremljen z enim samim oscilatorjem za programiranje zvoka, vendar nam čudovito preprosti ukazi za zvok v njegovem Mircrosoft Extended Colour basicu omogočajo, da nam že preprosti programi igrajo dovolj zapletene melodije, ki še kar dobro zvenijo. Žal nima generatorja šuma. To je zelo nenavadno, saj si je težko predstavljati marsikatero video igrico brez primernih zvočnih učinkov, ki jih lahko dosežemo samo z generatorjem šuma.

Ukaz SOUND za zvok ima pri tem računalniku naslednji format:

```
SOUND P,D
```

kjer je P višina tona (1 – 255) in D trajanje (1 – 255). Višina tona ni natančna in ima malo skupnega s tonsko lestvico. Enočrtnemu C približno ustreza vrednost 89 in noti A pri 440 Hz pa vrednost približno 159. Trajanje ravno tako ni natančno. Eni sekundi približno ustreza vrednost 16, dvema 32 itd.

Program, ki sledi, nam bo pokazal, kako na tem računalniku dosežemo posebne zvočne učinke. V tem primeru bi z malo domišljije zaslišali vzlet kakega »letečega krožnika«.

```
10 FOR P = 10 TO 170 STEP 10
20 FOR D = 16 TO 1 STEP –1
30 SOUND P,D
40 NEXT D
50 NEXT P
```

Z ukazom PLAY lahko določimo natančno višino, trajanje in glasnost note. Določimo lahko niz takih not z izbranimi premori različnih dolžin. Tako je mogoče preprosto sestaviti melodije z različnimi dolžinami tonov in premorov. Ukaz PLAY ima naslednji format:

```
PLAY "T;O;V;L;N;P"
```

kjer je T takt (T1 – T255); O je oktava (01 – 05); V je glasnost (V0 – V15); L je dolžina tona (L1 – L255); N je notna vrednost (1 – 12 ali kar črka za ustrezno noto) in P je premor pred naslednjim tonom (P1 – P255).

Med parametri v tem ukazu ni nujno pisati podpičja, vendar ga je zaradi jasnosti priporočljivo pisati. Tudi vrstni red parametrov je lahko poljuben, ker črka, ki stoji pred številko, nedvoumno določa, na kateri parameter se nanaša vrednost. Parametri T, O, V in L ohranjajo svoje vrednosti, zato jih ni treba vedno ponovno vključiti v ukaz PLAY, če smo zadovoljni z njihovo trenutno vrednostjo. Parameter T določa takt. T1 pomeni zelo počasi in T255 hitro. Poudarimo naj še, da, če se spremeni parameter T ob nespremenjenih vrednostih parametrov L in P, se spremeni dolžina trajanja tona in premora, le razmerje med njima se ohrani.

Dalje lahko dolžine tonov zvezno spreminjamo z dodajanjem pik, kot na primer L1... ali L5., kjer vsaka pika podaljša dolžino tona za polovico osnovne vrednosti. Torej L1... = 1 + 1/2 + 1/2 + 1/2 = 2,5 in

Igra s svetlobo

Drugi pregled grafičnih zmožnosti računalnika BBC model B

Programski jezik BBC basic nima toliko ukazov za grafiko visoke ločljivosti kakor basic nekaterih drugih računalnikov. Nima na primer ukazov CIRCLE in PAINT. Vendar teh dveh ukazov ni težko nadomestiti z nekaj vrsticami programa v BBC basicu.

Grafični zaslon ima iste koordinate ne glede na način izbrane grafike visoke ločljivosti, in koordinatno izhodišče je vedno v gornjem levem kotu. Poglejmo si ukaze za grafiko visoke ločljivosti:

```
MOVEx,y
```

Ta ukaz premakne kurzor na točko s koordinatama x in y, ne da bi narisal črto. Omenimo naj, da lahko grafični kurzor premikamo neodvisno od kurzorja za besedilo.

```
DRAWx,y
```

Kakor pove že samo ime, ukaz DRAW nariše črto od točke, kjer je grafični kurzor, do točke s koordinatama x in y.

```
PLOTk,x,y
```

PLOT je večnamenski ukaz. Njegova funkcija se spreminja z vrednostjo parametra takole:

Vrednost parametra k	Funkcija
0	premik glede na zadnjo točko
1	narisati črto od izhodišča v barvi ospredja
2	narisati črto od izhodišča v komplementarni barvi ozadja
3	narisati črto od izhodišča v barvi ozadja
4	isto kot MOVE
5	isto kot DRAW

L5. = 1/5 + 1/10 = 0,3. V prvem primeru bo ton trajal dvainpolkrat dlje kakor ton, določen z L1, medtem ko bo v drugem primeru trajal 3/10 dolžine tega tona.

Ni preproste metode, s katero bi določili absolutno zvezo med dolžino tona in taktom. Najpreprosteje jo določimo z metodo poskušanja.

Parameter 0 določi oktavo, v kateri bo zazvenela nota, ki sledi. 01 se začne s tonom, ki pripada noti C pri 131 Hz, 05 pa sega do note B pri 2093 Hz, enočrtni C začenja oktavo 02. Znotraj ene oktave lahko predstavimo note na dva načina, s številko ali kar z notno črko. Prvi način je uporaben, če hočemo noto uporabiti kot spremenljivko, z drugim načinom pa postane program razumljivejši. Kako si ustrezajo številke in notne črke, nam kaže naslednja tabela:

1	2	3	4	5	6
C	C#	D	D#	E	F
7	8	9	10	11	12
F#	G	G#	A	A#	B

Gornjo razlago najlepše ilustrira tale zgled. Naslednji stavek bo zaigral noto F (6) v oktavi 02, ki je polovinka (L2), z jakostjo V15. Če oktave ne navedemo, potem je mišljena oktava 02. Podobno je pri jakosti, privzeta jakost je V15. Dalje sledi premor ene četrtinke dolžine L1, nato še nota A# iz tretje oktave, pri glasnosti V20. Takt je nastavljen na T3.

```
PLAY "T3;L2;6;P4;03;V20;L1;A#"
      < F >      < A# >
                premor
```

Dalje lahko spreminjamo vrednosti parametrov T, O, V in L z dodajanjem obrazil:

Obrazilo	Učinek
+	prišteje ena k trenutni vrednosti
−	odšteje ena od trenutne vrednosti
>	pomnoži trenutno vrednost z dve
<	deli trenutno vrednost z dve

Format je naslednji: T+, T−, T< ali T> za vsak parameter.

Ena najbolj koristnih lastnosti računalnika Dragon je uporaba podnizov. Melodijo lahko opredelimo kot niz, ki ga priredimo neki spremenljivki, in to lahko potem uporabimo na kateremkoli mestu v programu. Zgled:

```
10 A$ = "F;A#;G"
20 B$ = "C;D#;F;P4;XA$;"
30 PLAY B$
```

Tu opredelimo spremenljivko A$ in jo potem vključimo v spremenljivko B$. Dobljena melodija je sestavljena iz naslednjega zaporedja not:

C — D# — F — P4 — F — A# — G.

S to tehniko lahko dano melodijo ponavljamo, kolikor hočemo, ali pa jo vključimo v drugo, kakor smo napravili v gornjem primeru z XA$.

6	isto kot DRAW, samo v komplementarni barvi
7	isto kot DRAW, samo da v barvi ozadja

Za večje vrednosti parametra k se te funkcije ponavljajo, vendar z dodatnimi učinki, kakor na primer črtkane črte namesto polnih. Za vrednosti parametra k med 80 in 87 ta ukaz opravlja posebno zanimive funkcije. PLOT80,x,y poveže točko s koordinatama x in y z dvema prej definiranima točkama v trikotnik. Notranjost tega trikotnika se potem obarva v barvi ospredja. To nekako nadomešča manjkajoči ukaz PAINT, vendar le za preprostejše like.

Ukaz VDU x ustreza bolj običajnemu basicovemu ukazu PRINT CHR$(x). Videli smo, da ukazu VDU lahko sledi zaporedje števil. Na primer: VDU x,y,z pomeni isto kot PRINT CHR$(x);CHR$(y);CHR$(z). Ukaz VDU nam omogoča dostop do dela operacijskega sistema računalnika BBC, ki krmili grafiko in izpis na zaslon. Četudi lahko ukaz VDU uporabljamo znotraj basicovega programa, deluje v resnici neodvisno od jezika, ki ga uporabljamo. Tako se ukaz VDU lahko uporablja tudi v grafiki jezika pascal ali kateregakoli drugega jezika, ki ga ponuja BBC.

Na računalniku BBC Micro je zelo preprosto definirati nove znake. VDU 23 krmili to funkcijo. Znaki so sestavljeni iz 8 krat 8 točk. V pomnilniku je tak znak predstavljen z osmimi zlogi. Vsaka vrstica točk v znaku je predstavljena s po enim zlogom. Točke, ki so prižgane, so predstavljene z biti, postavljenimi na 1, medtem ko ugasnjene točke predstavljajo biti, postavljeni na 0. Vsak tak zlog nam predstavlja decimalno število, tako da je znak predstavljen z osmimi decimalnimi števili. Z naslednjim programom smo z ukazom VDU 23 definirali nov znak v obliki puščice, ki smo mu dali kodo ASCII 240. Za nove znake smemo uporabljati kode ASCII med 224 in 255.

```
10 REM DEFINIRANJE ZNAKA
20 MODE 2
30 VDU 23,240,16,56,124,146,16,16,16,0
40 PRINT CHR$(240)
50 END
```

Kakor smo povedali, ta kratki program definira znak v obliki puščice. Zadnjih osem števil v ukazu VDU določa obliko znaka, medtem ko ga stavek 40 izpiše na zaslon.

VDU 24 in VDU 28 krmilita grafiko oziroma okna za besedilo na zaslonu. Z uporabo teh ukazov lahko na zaslonu določimo okna za besedilo in grafiko. To je uporabno predvsem za pisanje interaktivnih programov, kjer je zaželen tako razdeljen zaslon. Da določimo okna na zaslonu, je dovolj podati koordinate spodnjih levih in gornjih desnih vogalov.

MODE 1

Ta kratki program izriše na zaslon barvito spiralno rožico v grafičnem načinu MODE 1. Omenimo samo, da smo za cvetne listke uporabili ukaz za polnjenje trikotnikov.

```
 10 REM FLOWER
 20 CLS
 30 MODE 1
 40 FOR D=1 TO 3
 50 A=600 : B=500
 60 MOVEA,B
 70 FOR C=1 TO 550
    STEP3
 80 GCOL0,RND(3)
 90 S=(C/(RND(5)+10))
100 X=S*5*SIN(C/16)+A
110 Y=S*5*COS(C/16)+B
120 PLOT85,X,Y
130 NEXT C
140 NEXT D
150 END
```

Spiralni vzorec smo dosegli s kombinacijo sinusa in cosinusa v vrsticah 100 in 110. Znaka FOR... NEXT povečuje radij kronice, določene v vrsticah 100 in 110. Koordinate središča spirale A in B se lahko spreminjajo in s tem se spreminja položaj rožice.

Spektakularni zvok

Zvok računalnika Oric 1

Značilnosti računalnika Oric 1 so velike in med njimi so posebej očitne zmožnosti programiranja zvoka. Toni, ki jih proizvaja, se raztezajo čez sedem oktav, računalnik ima tri oscilatorje, generator šuma in sedem ovojnic, s katerimi določamo zven (barvo) signala, ki ga proizvaja računalnik. Izhod za zvok je prek zvočnika na televizorju.

Njegov basic ima več ukazov za krmiljenje zvoka, kakor: ZAP, PING in EXPLODE. Imena teh ukazov že sama po sebi dovolj povedo. Naslednji program nam ponazarja, kako jih uporabljamo, in nam predstavlja ukaz WAIT, ki povzroči premor, ki traja toliko stotink sekunde, kolikor je vrednost parametra, ki mu sledi.

```
10 ZAP: WAIT 200
20 EXPLODE
30 GOTO 10
```

Najbolj pa je uporaben ukaz SOUND. Njegov format je takšen:

```
SOUND, C,P,V
```

kjer je C kanal oziroma številka oscilatorja ali generatorja šuma (1–6), P je višina tona (10–5000) in V je njegova glasnost (0–15). Kanali od 1 do 3 vodijo k oscilatorjem, drugi trije pa h generatorju šuma. Višina tona nastavlja na približno vrednost. Če je P = 10, je ton najvišji (približno 10 kHz) in za P = 500 je ton najnižji (približno 100 Hz). Glasnost je najvišja pri V = 15, V = 6 je normalna glasnost.

Če je V = 0, potem je glasnost dana z ovojnico, določeno z ukazom PLAY. Največja pomanjkljivost ukaza SOUND je, da ne moremo določiti trajanja tona, z drugimi besedami: ukaz SOUND se ne izključi sam. Izključimo ga z ukazom PLAY, tako da postavimo parametre na nič.

Ukaz MUSIC nam omogoča natančno nastavljanje not. Njegova preprosta zgradba omogoča razumevanje tudi zapletenih glasbenih programov. Njegov format je naslednji:

```
MUSIC C,O,N,V
```

kjer je C kanal 1, 2 ali 3); O je oktava (0–6); N je nota (1–12) in V je glasnost (0–15). Šuma s tem ukazom ne moremo vključiti. Ukaz lahko izbira le med kanali od 1 do 3, ki vodijo k oscilatorjem. Glasnost ima lahko vrednosti 0, kontrolo glasnosti prevzame ukaz PLAY, do 15. N predstavlja zaporedno številko note znotraj oktave, določene s parametrom 0. Oktava, ki nosi številko 0, se začne pri tonu s frekvenco 32 Hz, medtem ko se oktava 6 razteza tja do tona frekvence 3951, 07 kHz. Naslednja razpredelnica kaže, kako so števila od 1 do 12 prirejena ustreznim notam.

1	2	3	4	5	6
C	C#	D	D#	E	F
7	**8**	**9**	**10**	**11**	**12**
F#	G	G#	A	A#	B

MUSIC in PLAY
Naslednji program nam predstavlja uporabo ukazov MUSIC in PLAY. Zaigral nam bo akord v C duru (C,G in E), vsakič z drugo ovojnico.

```
 10 REM**********
 20 REM *AKORD*
 30 REM**********
 40 MUSIC 1,4,1,0:REM
    *C*
 50 MUSIC 2,3,8,0:REM
    *G*
 60 MUSIC 3,3,5,0:REM
    *E*
 70 FOR E=1 TO 7 : REM
    *IZBIRANJE OVOJ-
    NICE*
 80 PLAY 7,0,E,750:REM
    *IGRANJE AKORDA*
 90 PLAY 0,0,0,0:REM
    *KONEC AKORDA*
100 WAIT 50:REM
    *PREMOR*
110 NEXT E: REM
    *NASLEDNJA
    OVOJNICA*
```

Light show

Grafika računalnika Spectrum je sicer omejena

Računalnik Spectrum je primeren za uvajanje v grafiko visoke ločljivosti. Preprosta uporaba približa grafiko tudi tistim, ki imajo malo programerskih izkušenj.

V nizki ločljivosti imamo na voljo male in velike črke skupaj z naborom posebnih grafičnih znakov. Te lahko izpišemo na zaslon z ukazom PRINT v katerikoli od osmih barv. Barva znakov (ospredja), zaslona (ozadja) in roba se določa z ukazi INK, PAPER oziroma BORDER. Sami lahko določimo še 21 novih znakov.

Zaslon je razdeljen na 24 vrstic s po 32 celicami. Zadnji dve vrstici sta namenjeni za sporočila računalnika in za vhod iz tipkovnice. To pomeni, da je za grafiko na voljo 22 vrstic. Zaslon je torej pri nizki ločljivosti razdeljen na 22 krat 32 celic. Pri visoki ločljivosti je zaslon razdeljen na 176 krat 265 točk. Velika prednost te grafike je, da jo lahko uporabljamo skupaj z grafiko nizke ločljivosti, kar nam omogoča, da rišemo diagrame z označbo enot na koordinatnih oseh, s komentarjem idr. Vsebino zaslona lahko shranimo na trak z

Če želimo zaigrati noto A pri 440 Hz na prvem oscilatorju s srednjo glasnostjo 6, moramo računalniku podati naslednji ukaz:

MUSIC 1,3,10,6

Da bi izkoristili vse zvočne zmožnosti računalnika Oric 1, je najprimernejše uporabiti ukaz MUSIC skupaj z ukazom PLAY. Format ukaza PLAY je naslednji:

PLAY C,N,E,P

kjer je C kanal (0—7); N je šum (0—7); E je ovojnica (1—7) in P je dolžina ovojnice (0—32767). Parametra C in N nam dajeta naslednje možnosti izbire:

Število	Kanal	Šum
0	Vsi osc. so izklj	Izključen
1	Osc. 1	+ Osc. 1
2	Osc. 2	+ Osc. 2
3	Osc. 1 & 2	+ Osc. 1 ali 2
4	Osc. 3	+ Osc. 3
5	Osc. 1 & 3	+ Osc. 1 ali 3
6	Osc. 2 & 3	+ Osc. 2 ali 3
7	Osc. 1, 2 & 3	+ Osc. 1, 2 ali 3

Prej definirane ukaze MUSIC (ali SOUND), ki imajo parameter V postavljen na nič, lahko zaigramo istočasno z ukazom PLAY — kanal določa kombinacija oscilatorjev za ta ukaz. Parameter N določi, kateri oscilator bo spremljal šum. S parametrom E izberemo eno od sedmih ovojnic (zvenov ali barv tona). To je natančneje opisano v priročniku za računalnik Oric 1.

Periodo ovojnice spreminjamo s parametrom P. S tem parametrom določimo celotno trajanje ovojnice (od 0 do 32767). Ta čas se določa za vsako ovojnico. Za orientacijo lahko rečemo, da je ovojnica pri vrednosti parametra P = 5000 dolga približno dve sekundi.

Ukazi za zvok računalnika Oric 1 se uporabljajo preprosto in so izredno dognani. Edini računalnik, ki tudi že omogoča krmiljenje ovojnic, je BBC micro, ki pa gre še mnogo dlje v dognanosti zvoka in je seveda tudi dosti dražji.

SOUND

Ta kratki program uporablja ukaz SOUND za proizvajanje zvoka, ki posnema zvok pri pristajanju vesoljska ladje.

```
 10 REM**************
 20 REM *PRISTAJA-
    NJE*
 30 REM**************
 40 FOR P=10 TO 3000
    STEP 10
 50 SOUND 2,P,6
 60 PLAY 2,0,1,1
 70 NEXT P
 80 WAIT 75
 90 PLAY 0,0,0,0
100 END
```

ukazom SAVE in jo po želji lahko pokličemo nazaj. Za to skrbi ukaz SCREEN$, ki ga rabimo tudi za izpis vsebine zaslona na tiskalnik. V grafiki nizke ločljivosti postavimo znak na določeno mesto na zaslonu z ukazom PRINT AT.

Na voljo je več ukazov za posebne učinke. Znake lahko izpišemo v komplementarnih barvah, dalje imamo še ukaza FLASH in BRIGHT. Naslednji zanimivi ukaz za grafiko nizke ločljivosti je OVER, ki omogoča združevanje znakov na kateremkoli mestu na zaslonu. To je posebno primerno, ko združujemo besedilo s sliko v visoki ločljivosti. Tako lahko pišemo po diagramih, ne da bi izbrisali osnovno sliko.

Izpis na zaslon krmilita dve območji v pomnilniku: eno je namenjeno znakom, drugo vsebuje informacijo o parametrih, ki spremljajo posamezno celico. Gre za informacijo o barvi INK in PAPER, o utripanju, določenem z ukazom FLASH itd. Ti parametri so zakodirani v zlogu, ki pripada vsaki znakovni celici. To informacijo lahko za vsako znakovno celico posebej preberemo z ukazom ATTR iz basicovega programa.

Na računalniku Spectrum je preprosto uporabljati grafiko visoke ločljivosti, ker imamo zanjo dovolj basicovih ukazov in se da uporabljati hkrati z grafiko nizke ločljivosti.

Basicovi ukazi za visoko ločljivost so naslednji:

PLOTx,y

Ta prižge točko na zaslonu s koordinatama x in y, v barvi, določeni z ukazom INK.

DRAWx,y,p

Napravi črto, ki veže točko, kjer je trenutno kurzor, s točko s koordinatama x in y. Če dodamo še tretji parameter, potem poveže te dve točki s krožnim lokom. Če je vrednost parametra $p = \pi$, zariše polkrog, če je $p = \pi/2$, potem zariše četrtino kroga itd. Lok se lahko upogne na eno ali drugo stran zveznice med točkama. Na katero stran se bo upognil, določa predznak parametra p.

CIRCLEx,y,r

Ta ukaz izriše krožnico z radijem r in s središčem v točki (x, y). Pri nekaterih računalnikih ukaz CIRCLE dopušča, da krožnico stlačimo in dobimo elipso, vendar pri Spectrumu to žal ni mogoče.

Grafika visoke ločljivosti pa ima nekatere pomanjkljivosti. Posledica možnosti mešanja obeh grafik je, da moremo znotraj ene znakovne celice definirati samo eno barvo INK. Če se dve črti različnih barv srečata, bodo vse »prižgane« (aktivne) točke (točke barve INK) postale enake barve, barve, ki je določena s poslednjim ukazom INK. Torej bo odsek prve črte, ki prečka znakovno celico, v kateri se sreča z drugo, postal enake barve kakor druga premica.

```
 10 REM *OBRAZ
    SMEŠKA*
 30 BORDER 6
 40 PAPER 6
 50 INK 2
 60 CIRCLE 122,88,50
 70 CIRCLE 97,108,5
 80 CIRCLE 147,108,5
 90 PLOT 92,68
100 DRAW 152,68,PI/3
110 END
```

Sliši se neverjetno

Ukaz ENVELOPE računalnika BBC model B nam omogoča praktično neomejeno oblikovanje zvočnega signala

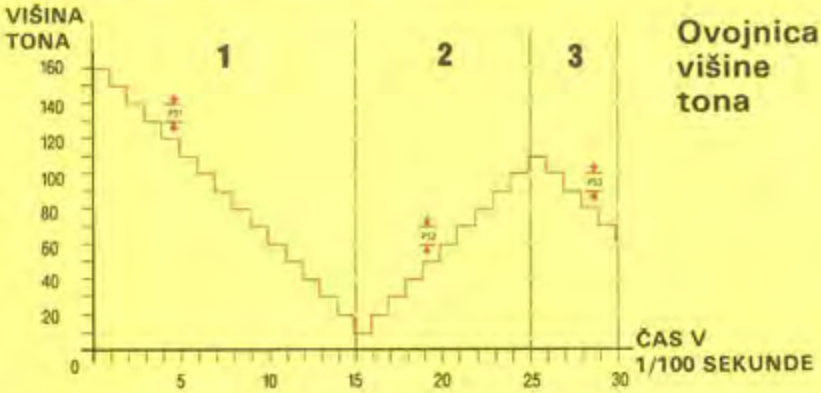
V prejšnjih poglavjih smo spoznali ukaz SOUND računalnika BBC micro. Zmožnosti proizvajanja zvoka pa pridejo do prave veljave samo, če uporabljamo ukaz SOUND skupaj z ukazom ENVELOPE. Ukaz ENVELOPE omogoča takšno oblikovanje zvočnega signala, da lahko dokaj dobro posnemamo običajna glasbila. Tudi zvoki, ki spremljajo igrice, se dajo programirati tako, da dovolj zvesto posnemajo zvoke v naravi, na primer strel, zvok, ki nastane ob eksploziji, itd. Sestava ukaza ENVELOPE je naslednja:

ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,AR,DR,SR,RR,FAL,FDL

S prvim parametrom N določimo številko ovojnice. Ta številka rabi za določitev ovojnice v ukazu SOUND ali SOUND&. Ena od štirih ovojnic lahko z ukazom SOUND nadomesti konstantno glasnost, določeno z vrednostjo parametra V 10 do −15. Parameter T lahko dobi naslednje vrednosti:

T (0 do 127) & (128 do 255)

Določa trajanje vsake stopničke ovojnice v stotinkah sekunde (glej diagram). Na primer, če je T = 5, to pomeni, da vsaka stopnička traja 0,05 sekunde. Če k željeni dolžini ovojnice dodamo število 128, se ovojnica ne ponavlja, temveč se v noti pojavi le enkrat. Če je T 133 = 5 + 128, vsaka stopnička traja 5 stotink sekunde.



Ovojnica višine tona

Iz diagrama se vidi, da gre tu za spreminjanje frekvence tona. Torej se pojem ovojnice tu nanaša ne samo na spreminjanje glasnosti (zvena), kakor smo videli v prejšnjih poglavjih, ampak tudi na spreminjanje frekvence. To nam ne koristi dosti, razen če je zaželen vibrato, pač pa dobimo nekakšno žvrgolenje, če je frekvenca dovolj visoka. Kakor kaže diagram, je perioda ovojnice razdeljena na tri časovne intervale. Za te tri intervale odgovarjata parametra PS in NS, kakor sledi:

Svetlobni relief

Uporaba premičnih likov na računalniku Commodore 64

Ena najbolj mikavnih prednosti računalnika Commodore 64 je možnost uporabe premičnih grafičnih likov, ki jih v angleščini imenujejo sprite, kar po naše pomeni duh, škrat. Ti grafični liki so narejeni tako kakor znaki, ki jih lahko definiramo sami, vendar so veliko večji, sestavljeni so iz 21 krat 24 točk. Tudi niso izrisani na zaslon kot običajni znaki, zato jih lahko premikamo po eno točko, medtem ko pri običajnih znakih potrebujemo po osem točk, da jih postavimo iz ene znakovne celice v drugo. Na zaslon lahko izrišemo do osem takšnih grafičnih likov hkrati, vsak izmed njih pa ima naslednje lastnosti, ki jih lahko programiramo:

Oblika in barva

Grafični lik je določen precej podobno kakor navadni znaki, vendar je za zakodiranje takšnega lika v binarni obliki potrebnih 63 zlogov računalniškega pomnilnika. Njegova oblika je torej zakodirana v 63 zaporednih pomnilniških lokacijah. Vsak grafični lik ima kazalec, ki kaže na pomnilniške lokacije, kjer je zakodirana njegova oblika. To pomeni, da lahko na te lokacije »gleda« več takih grafičnih likov; seveda so potem njihove oblike enake. Ravno tako lahko en tak lik spremeni svojo obliko, če se njegov kazalec premakne na druge pomnilniške lokacije, kjer je zakodirana drugačna oblika.

Vsakega od teh likov lahko obarvamo s katerokoli od 16 barv, ki so nam na voljo. Lahko je večbarven, vendar je to treba plačati z razpolovitvijo ločljivosti v vodoravni smeri, iz istega razloga, kakor smo to videli pri grafičnih znakih.

Velikost in gibanje

Te grafične like lahko raztegnemo v vodoravni, navpični ali v obeh smereh hkrati. Tako lahko podvojimo

PS1, PS2 & PS3 (−128 do 128)

PS se nanaša na spremembo višine tona. V začetku je višina tona določena z ukazom SOUND. PS1 določa stopnjo pozitivne ali negativne spremembe višine tona v prvem časovnem intervalu, kakor kaže diagram, medtem ko PS2 to določa v drugem intervalu in PS3 v tretjem. Podobno kakor pri ukazu SOUND je PS podan v četrtinkah poltona.

NS1, NS2 & NS3 (0 do 128)

Parameter NS se nanaša na dolžino posameznega časovnega intervala. Podan je v stotinkah sekunde. Kakor pri parametru PS tudi tu velja, da se NS1 nanaša na prvi, NS2 na drugi in NS3 na tretji časovni interval. Torej NS določa, kdaj bo ovojnica prišla iz kontrole enega parametra PS pod kontrolo drugega in tudi trajanje celotne periode. Vrednosti parametrov PS in NS na diagramu so naslednje:

T = 1	PS1 = −10	NS1 = 15
	PS2 = +10	NS2 = 10
	PS3 = −10	NS3 = 5

Višina tona pa je bila podana z ukazom SOUND s P = 160. Torej ima ukaz ENVELOPE v tem primeru naslednjo obliko:

ENVELOPE 1,1,−10,10,−10,15,10,5,0,0,0,0,0,0

Trajanje periode je določeno s formulo (NS1 + + NS2 + NS3) × T, kar v našem primeru znaša 0,3 sekunde. Periodično ponavljanje ovojnice izključimo s tem, da določimo parametru T vrednost, ki je večja od 127 in manjša od 256, kakor smo že omenili, ko smo razložili pomen parametra T.

njihovo prvotno velikost. Popolnoma razširjen lik ima lahko največ 48 krat 42 točk. Spet je treba to plačati; ločljivost se zmanjša na polovico v smeri razširitve.

Tak grafični lik se na zaslonu lahko pomika naprej po eno točko, prejšnji položaj lika se samodejno izbriše. Lahko jih pomaknemo tudi iz območja, ki je vidno skozi okno, določeno z zaslonom.

Prednost in trk

Ko se dva taka lika srečata, je videti, kakor da bi eden šel pred drugim. Če ima prednji lik kakšne luknje, je skoznje videti lik za njim. Kateri lik bo šel zadaj in kateri spredaj, določimo sami z nastavitvijo prednosti. Vsakemu liku določimo število od 0 do 7. Liki z nižjo vrednostjo se gibljejo pred liki z višjo vrednostjo. Po navadi se liki gibljejo pred običajnimi črkami in grafičnimi znaki, vendar lahko programiramo tako, da se gibljejo za njimi. S tem dobimo na zaslonu občutek globine.

Ko se dva lika srečata, se primerno nastavi določena pomnilniška lokacija, ki jo imenujemo »kolizijski register«. Ko z uporabo ukaza PEEK preberemo vrednost, ki je shranjena v tej pomnilniški lokaciji, lahko izvemo, ali sta lika trčila ali sta se samo srečala.

Obstaja še ena pomnilniška lokacija, katere naloga je zaznavanje trkov likov s črkami ali grafičnimi znaki v ozadju. To nam omogoča programiranje iger s hitrim gibanjem v basicu. Vendar basic nima posebnih ukazov za ustvarjanje teh grafičnih likov. Like moramo ustvarjati z zapisom določenih vrednosti na določene pomnilniške lokacije z ukazom POKE. Druga možnost je nakup vtičnega modula s Simonovim basicom, ki je opremljen s potrebnimi ukazi za oblikovanje grafičnih likov.

Simonov basic
Za približno 50 funtov je mogoče kupiti vtični modul, ki se priključi na računalnik. S tem modulom je mogoče programirati v Simonovem basicu, ki je obogaten z velikim naborom ukazov, ki se nanašajo na zvok, grafiko visoke ločljivosti, programiranja grafičnih likov itd. Vsega skupaj ima 114 dodatnih ukazov. Tu so vključeni tudi ukazi za postavitev grafike visoke ločljivosti, določitev barve ozadja in ospredja, ukazi za risanje krogov, elipse, pravokotnikov in ravnih črt. Med ukazi za krmiljenje grafičnih likov so ukazi za njihovo ustvarjanje, ugašanje in prižiganje ter za načine, kako jih postavimo na zaslon.

Drugi korak
Te vrstice lahko dodamo programu o veleblagovnici na strani 225. Ta del programa uporablja dva razširjena večbarvna grafična lika, žensko in nakupovalni voziček. Kazalec teh grafičnih likov lahko krmilimo, tako da ženska spreminja obliko. To nam pričara občutek, kakor da ta ženska prečka zaslon, plešoč in potiskajoč voziček. Če hočemo program o veleblagovnici uporabiti kot podprogram k temu programu, moramo vrstico 3270 nadomestiti s 3270 RETURN.

```
  90 REM ** FIGURICE 64 **
 100 PRINT"□"
 110 V=53248
 120 REM----BRANJE GRAFICNIH PODATKOV----
 130 FOR I=12288 TO 12350:READ A:POKE I,A:NEXT
 140 FOR I=12352 TO 12414:READ A:POKE I,A:NEXT
 150 FOR I=832 TO 894:READ A:POKE I,A:NEXT
 160 FOR I=896 TO 958:READ A:POKE I,A:NEXT
 170 FOR I=12416 TO 12478:READ A:POKE I,A:NEXT
 180 REM----POVECAVA FIGURIC----
 190 POKE V+23,7:POKE V+29,7
 200 REM----BARVA FIGURIC----
 210 POKE V+39,10:POKE V+40,10
 220 POKE V+41,1
 230 REM----VEC BARV----
 240 POKE V+28,3:POKE V+37,7:POKE V+38,9
 300 REM----KAZALCI POMNILNIKA----
 310 POKE 2040,192:POKE 2041,193:POKE 2042,194
 320 REM----POSTAVITEV KOORDINAT----
 330 Y0=150:Y1=Y0+42:Y2=Y0+34
 340 POKE V+1,Y0:POKE V+3,Y1:POKE V+5,Y2
 400 REM----VKLJUCITEV FIGURIC----
 410 POKE V+21,7
 500 GOSUB 3000:REM OPUSTIMO, CE NI PODPROGRAMA
1000 X0=20
1010 POKE 2040,13:POKE 2041,14
1020 POKE V,X0:POKE V+2,X0:POKE V+4,X0+48
1030 FOR  I=1 TO 500:NEXT
1040 POKE 2040,192:POKE 2041,193
1050 X0=X0+5
1060 POKE V,X0:POKE V+2,X0:POKE V+4,X0+48
1070 FOR I=1 TO 500:NEXT
1080 X0=X0+5
1090 IF X0>200 THEN 1110
1100 GOTO 1010
1110 FOR J=1 TO 10
1120 POKE 2040,13:POKE 2041,14
1130 FOR I=1 TO 50:NEXT
1140 POKE 2040,192:POKE 2041,193
1150 FOR I=1 TO 50:NEXT
1160 NEXT
1170 GOTO 1170
9000 REM----PODATKI ZENSKA GORNJI DEL----
9010 DATA,0,0,0,0,21,0,0,21,0,0,22,0,0,86,0
9020 DATA,0,86,0,0,86,0,0,40,0,0,252,0
9030 DATA,15,255,0,255,255,0,255,255,0
9040 DATA,195,255,0,195,255,0,195,243,254
9050 DATA,207,243,254
9060 DATA,143,240,0,143,252,0,15,252,0
9070 DATA,15,252,0,15,252,0
9100 REM----PODATKI ZENSKA SPODNJI DEL----
9110 DATA,15,252,0,15,252,0,15,252,0
9120 DATA,15,252,0,5,84,0,5,84,0,5,84,0
9130 DATA,5,84,0,10,40,0,234,40,0,234,40,0
9140 DATA,234,40,0,192,40,0,192,40,0,0,40,0
9150 DATA,0,40,0,0,63,0,0,63,0,0,0,0,0,0,0
9160 DATA,0,0,0
9200 REM----PODATKI ZENSKA ZGORNJI DEL ST.2----
9210 DATA,0,0,0,0,20,32,32,85,32,32,105,48,48,105,48
9220 DATA,48,105,48,48,105,48,48,40,48,48,252,48
9230 DATA,63,255,240,63,255,240,63,255,0
9240 DATA,3,255,0,3,255,0,3,240,0
9250 DATA,15,240,0
9260 DATA,15,240,0,15,252,0,15,252,0
9270 DATA,15,252,0,15,252,0
9300 REM----PODATKI ZENSKA SPODNJI DEL ST.2----
9310 DATA,15,252,0,15,252,0,15,252,0
9320 DATA,15,252,0,5,84,0,5,84,0,5,84,0
9330 DATA,5,84,0,10,40,0,58,168,0,58,168,0
9340 DATA,58,0,0,58,0,0,10,0,0,10,0,0
9350 DATA,10,0,0,15,192,0,15,192,0,0,0,0,0,0,0
9360 DATA,0,0,0
9400 REM----PODATKI VOZICEK----
9410 DATA,192,0,0,224,0,0,118,0
9420 DATA,0,55,192,0,32,60,0,53
9430 DATA,87,240,32,0,15,53,85,85
9440 DATA,32,0,3,53,85,85,0,0,3
9450 DATA,21,85,85,31,255,255,24,0
9460 DATA,0,12,0,0,12,0,0,31,255
9470 DATA,240,31,255,255,1,0,2,7
9480 DATA,0,14,7,0,14
```

Nekaj besed o pripravi knjige

Leta 1985 bo minilo natanko 25 let, odkar je Mladinska knjiga izdala prvo slovensko računalniško knjigo, dr. Franceta Križanića Elektronski aritmetični računalniki, v zbirki Sigma. Pričujočo izdajo lahko torej pojmujemo nekako kot jubilejno.

Knjiga Hišni računalnik ponuja izbor najboljših sestavkov iz zvežčičev, ki jih je londonska založba Orbis izdajala tedensko v zimi 1983/84. Izbor je izšel avgusta 1984 kot knjiga. Slovenska izdaja knjige torej prihaja le nekaj mesecev za angleško. Že bežen pregled bo pokazal, da ne gre za enostaven prevod. Ne le, da so poslovenjeni vsi programi, tudi v besedila so prevajalci včasih, po potrebi, posegli. Dodali pa smo še nekaj tem, med drugim tudi prispevek o ekspertnih sistemih, kjer so naši strokovnjaki v svetovnem vrhu.

Takšno delo so seveda lahko izpeljali samo naši najboljši strokovnjaki, ki so vsi bodisi predavatelji na Fakulteti za elektrotehniko bodisi njeni sodelavci, in sodelavci Instituta Jožef Stefan. Ne nazadnje je bilo pri prevodu in priredbi opravljeno pionirsko delo na terminologiji. Pridruženi slovarček je tudi prvi slovenski slovarček računalniških izrazov v javni rabi — opravičujemo se pa zaradi napak v tisku, do katerih je prišlo spričo prekratkih rokov.

Sledijo podatki o prevajalcih in avtorjih prispevkov:

1. poglavje
Kaj je računalnik — S. Starašinič
Prva pomoč kupcu računalnika — J. Lokovšek
Postavitev računalniškega sistema — S. Starašinič
Krmiljenje računalnikov — A. Jesenko

2. poglavje
Drobni tisk — J. Lokovšek
Sinclair Spectrum — N. Simić
Vprašanja in odgovori — S. Starašinič
Commodore 64 — J. Lokovšek
Dragon 32 — A. Jesenko
Vprašanja in odgovori — S. Starašinič
BBC model B — J. Lokovšek
Sinclair QL — N. Simić
Vprašanja in odgovori — S. Starašinič
Atari 400 in 800 — A. Jesenko
Acorn Electron — A. Jesenko
Hišni računalniki Made in Yugoslavia — B. Sovič
Epson HX 20 — A. Jesenko

3. poglavje
Biti in zlogi — M. Štiglic
Ko je 1 + 1 = 10 — M. Štiglic
Seštevanje in odštevanje — M. Štiglic
Množenje v dvojiškem sestavu — S. Starašinič
Pravilno ali napačno — M. Štiglic
Logična vsota in seštevalnik — S. Starašinič
Popolno pomnjenje — M. Štiglic
Shranjevanje podatkov — S. Starašinič
Razumem sporočilo — M. Štiglic
Živčni center — M. Štiglic
Pravilno naslovljeno — M. Štiglic
Digitalni dialog — M. Štiglic
Čakalnica — M. Štiglic
Dva načina prevajanja — B. Sobočan
PEEK in POKE — B. Sobočan
Parno pravilo — N. Simić
Detektivski posli — N. Simić
Vplivne zveze — N. Simić

4. poglavje
Tesen dotik — N. Simić
V arhivu — N. Simić
Hitri reaktorji — N. Simić
Ploščati pomnilnik — N. Simić
V žarišču — N. Simić
Opremljen za tiskanje — N. Simić
Na dveh kolesih — M. Špegel
Hitro risanje — N. Simić
Halo, računalnik — N. Simić
Branje grafičnih podatkov — B. Sobočan
Načrtovanje po naročilu — B. Sobočan

5. poglavje
Ljudje in igre — R. Osredkar
Elektronski vzgojitelj — B. Sobočan
Mikrosvetovi — M. Špegel
Lepa beseda — N. Simić
Preglednica — S. Starašinič
Doba prenosnih računalnikov — R. Osredkar
Komunikacija človek—računalnik — P. Tancig, zadnji del je avtorski
Miške — M. Špegel
Enoroki rokovnjači — M. Špegel
Gibajoče se slike — M. Štiglic
Vojna in mir — R. Osredkar
Šola na zaslonu — B. Sobočan
Ekspertni sistemi — N. Lavrač in I. Bratko

6. poglavje
Diagram poteka — S. Starašinič
Urejanje podatkov — S. Starašinič
Razvrščanje informacij — S. Starašinič
Verižna sporočila — T. Pleteršek
Urejanje kart — pravila igre — T. Pleteršek
Koda za urejanje — T. Pleteršek
Optimizacija hitrosti — T. Pleteršek
Inteligentne igre — S. Starašinič
Sam svoj avtor — S. Starašinič
Programski pripomočki — B. Sobočan
Strojna koda — T. Pleteršek
Zbirni jezik — T. Pleteršek, avtorski prispevek
Branje zemljevida — N. Simić

7. poglavje
Izvajamo ukaze — J. Vilfan
Pomembne zanke — J. Vilfan
Preidimo k stvari — D. Smerdelj
Pomen rutinskih programov — D. Smerdelj
Novo leto v basicu — D. Smerdelj
Izziv elementov — D. Smerdelj
Uredimo si programe — D. Smerdelj
Funkcije v basicu — D. Smerdelj
Prepustimo jih naključju — D. Smerdelj
Neka druga dimenzija — Z. Matič
Kontrola — Z. Matič

8. poglavje
Še zvok ... in svetloba — B. Sobočan
Ovojnica, Igra s svetlobo — B. Jurčič
Zvok in VIC 20 — B. Jurčič
Preprosti zvoki — B. Jurčič
Prve slike — R. Osredkar
Zvočni sistemi — B. Jurčič
Svetlobni program — B. Jurčič
Kvaliteta zvoka — B. Jurčič
Preizkus zvoka — B. Jurčič
Igra s svetlobo — B. Jurčič
Spektakularni zvok, Light show — B. Jurčič
Sliši se neverjetno — B. Jurčič

681.3

681.3

HIŠNI računalnik
Inv: 19108/M