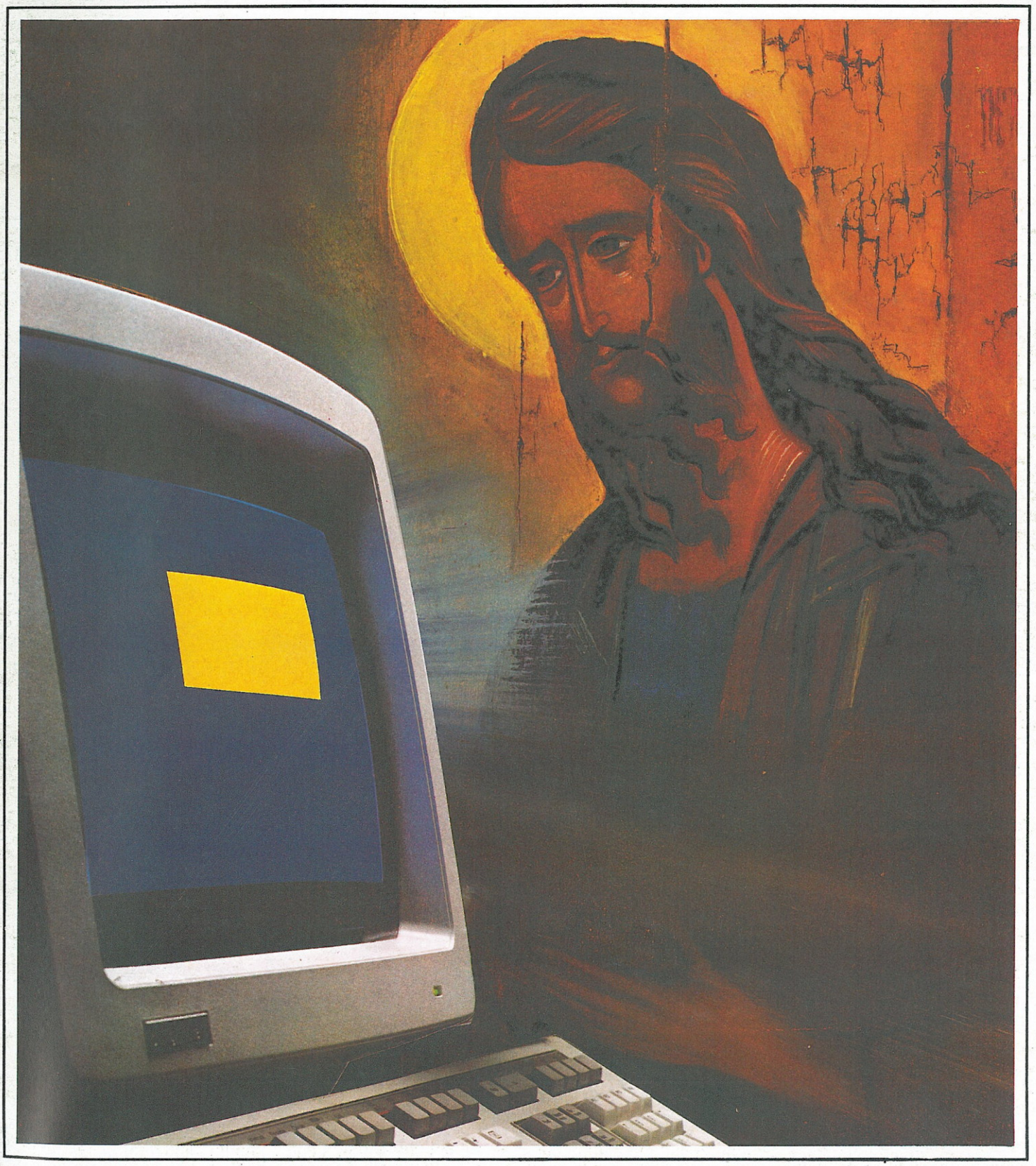
μ mikro számítógép magazin

Ára: 30 Ft



IKON – ÚJ ÉRTELMEZÉSBEN

1988/4

MŰSZERTECHNIKA
KISSZÖVETKEZET

Budapest, Szállás u. 21. 1107
Levélcím: Pf. 225. 1475
Bemutatóterem: Majakovszkij u. 1/D 1075

Telefon: 471-590
Telex: 22-7734
Telefon: 221-623

Árkedvezmény

Áttérés az új adórendszerre a Műszertechnika komplex szolgáltatásával. Az áttérés meggyorsítása érdekében kiépítéstől függően 10-25% árkedvezményt adunk 1988. szeptember 15-ig az alábbi komplex rendszereinkre:

- **Munka/bér '88 komplex munkaügyi és bérszámfejtő**
- **Főkönyvi és folyószámla-könyvelési**
- **Beruházáselszámolási programcsomagok**

Fenti programok egy és több munkahelyes kiépítésű, 80 Mbájt háttérkapacitással rendelkező Novell hálózati rendszerben működnek. Komplex rendszereink telepítésére 1988 második negyedévre előjegyzést felveszünk.

Programokkal kapcsolatban felhasználóisoftware osztályunkon Zakariás Zsuzsanna osztályvezető ad felvilágosítást telefonon: 471-590 vagy személyesen 1107 Bp., Szállás u. 21.

ETHERNET

csatlakozónkkal biztosítjuk a Novell alapú lokális hálózatokban jelenleg megvalósítható legnagyobb teljesítményű összeköttetést.

Az **ETHERNET** 10 Mbit/sec-os átviteli sebessége a legjobban elterjedt ARC-NET négyszerese.

Vállaljuk már működő Novell alapú hálózatok sebességnövelését.

Új rendszereinket kívánságra **ETHERNET** csatlakozóval szállítjuk.

ETHERNET alapú lokális hálózatunk megtekinthető a bemutatótermünkben.

Felvételre keresünk kiemelt bérezéssel, tőkés kompenzációs ügyletek szervezésében, bonyolításban jártas külkereskedelmi üzletkötőt német, angol nyelvtudással, saját gépkocsi használattal.

Érdeklődni a 471-590/165-ös telefonon.

Felvételre keresünk kiemelt bérezéssel szovjet árucsereügyletek bonyolításában jártas szakembert, saját gépkocsi használattal.

Érdeklődni a 471-590/165-ös telefonon.

6. ÉVFOLYAM
1988/4. SZÁM

μ mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

A szerkesztőség munkatársai:
Babos János (tervezőszerkesztő)
Bakos Tamás (programozástechnika)
Broczkó Péter (hírek)
Énekes Ferenc (KISZ)
Jakab Ágnes (olvasószerkesztő)
Kovács Győző (levelezés)
Krasznai Éva (Diákszerkesztőség)
Lindner László (sakk)
Petróczy Judit (könyvek)
Pinke György (NJSZT, alkalmazások)
Simonyi Endre
Szebenszki Sándor
Szulyovszky Csaba
Tamásné Lakó Erika
Terebessy Ákosné
Varga András (TII, iskola-számítógép)
Vizessy Mária

Címképünk:
Ramocsai Imri munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Király G. István
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlap-előfizetési és Lapellátási Irodáján
(1900 Budapest XIII., Lehel út 10/A)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta.
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, Pf. 149.
és a Magyar Média
1932 Budapest, Pf. 279.
86—0253

Szikra Lapnyomda
Budapest (88—0212)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

„. . . a' ki sokat tud, sokat akarna, hogy tudjanak a' tanítványi és sokszor megesik, hogy a' capacitásokon túl ragadtatik, és ők innen maradnak."
(Herpei Ádám nagyenyedi professzornak Bolyai Farkashoz írt leveléből)

Oktatástechnológia, nyílt tanulási rendszerek

Az 1987/7. számban emlékeztem meg a 28 éve indult tervmatematikus-képzésről, az egyik első olyan egyetemi szakról, amelyben a megtanulandó diszciplínák többsége már a számítástechnikával volt kapcsolatban. Akkor — 1960-ban — az volt az újdonság, hogy számítástechnikai tantárgyakat tanítottunk, ma pedig az, hogy az oktatásban különféle számítógépes technológiákat alkalmazunk.

A technologizált oktatásnak még ma is többféle elnevezése van, én is váltakozva használtam az oktatás és tanulás kifejezéseket egy folyamat két részének meghatározására. A klasszikus oktatási rendszerekben a tanár a tanteremben oktat, a tanuló pedig otthon tanul. A tanárok a tantermi oktatásban különféle eszközöket (könyveket, táblát, krétát, tablókat, filmet, videót és még sok minden mást is) használnak azért, hogy a diákok az anyagot már az iskolában megértsék, ti. a tanár is tudja, hogy magyarázat és szemléltetés nélkül a legtöbb diák az anyagot otthon képtelen volna egyedül megtanulni. Ezért amíg a tanteremben a tanárnak egy teljes oktatástechnológiai arzenál áll rendelkezésére, addig otthon a diáknak a tankönyvön, munkafüzeten és saját jegyzetein kívül szinte semmi. Ha tehát egy diák az iskolában figyelmetlen vagy nem tud jól jegyzetelni, azaz nem alakította ki saját tanulási technológiáját, akkor rosszul tanul, lemarad a többiektől, súlyosabb esetben kibukik az iskolából.

A klasszikus tantermi oktatásban a hangsúly tehát a tanári munkán van és csak másodsorban a tanuláson; vagyis a tantermi oktatás tanárorientált. Csak megjegyzem, hogy valószínűleg ezért kötelező a klasszikus oktatási rendszerben az iskolába járás és nemcsak az elemi és középiskolákban, de még az egyetemeken és a főiskolákon is, ti. a diákok többsége, miután nincs saját és az önálló tanulást támogató technológiája, képtelen a tananyaggal egyedül megbirkózni. Ennek a merev rendszernek persze azok látják kárát, akik szinte semmi segítséget nem igényelnek, még akár könyvből is elboldogulnak. Ráadásul van olyan diák, aki magától is gyorsan halad, míg van, aki folyamatosan, de lassan és megfontolva. Az elsőt a klasszikus rendszerben vissza szokták tartani, a lassút pedig a tehetségét meghaladó „rohanásra" ösztökélik.

A klasszikus oktatásnak mindehhez természetesen megvannak a maga eszközei, például a napi feleltetés vagy az egyetemeken a zárthelyi dolgozatok rendszere. Pedig végsősoron csak a teljes anyag tudásának ellenőrzésére lenne szükség; egy nagyon részletes vizsgával még az anyagon belüli, illetve a más diszciplinákkal kapcsolatos összefüggések ismeretét is ellenőrizni lehet. Nyilvánvaló, hogy a klasszikus oktatási rendszer azért nem él ezzel a lehetőséggel, mert nem rendelkezik azzal a technológiával, amellyel az ilyen bonyolult ellenőrzést jó hatásfokkal elvégezhetné.

A számítógép vagy a számítógéphez kapcsolt terminál az a mai eszköz, amely nemcsak a tanítást, hanem a tanulást is segíti, és amellyel ráadásul a folyamatos önellenőrzés, de még az alkalmankénti vizsgáztatás is sikerrel megoldható. Ezért látszik valószínűnek, hogy a számítógép és a tanulási technológiák megjelenése az iskolákban nemcsak az oktatási-tanulási folyamatot, de az iskolai rendszert magát is meg fogja változtatni.

A változáshoz elsősorban a tanulást segítő programokra van szükség, amelyekből már nálunk is sokat használnak az iskolákban. Ezeket az oktatóprogramokat egyelőre magas szintű nyelven programozni tudó tanárok és diákok készítik, a tananyagot szinte kivétel nélkül BASIC-ben programozzák. Nem tehetnek mást: az iskolákban is és otthon is a rendelkezésre álló gépeknek (Commodore család, Sinclair, HT stb.) nincs és nem is lehet megfelelő szoftverkörnyezetük ilyen grafikával vegyes programok fejlesztésére. Sajnos tanulóprogramfejlesztésben is lekörözött bennünket a világ, ma már az iparilag fejlett országokban, de a fejlődő országok jó részében is ún. szerzői rendszerek (authoring systems) vezérlése alatt írják a tanulóprogramokat. Ezek a rendszerek tulajdonképpen igen magas szintű nyelvek, amelyek segítségével a képernyőn a „mondanivaló" szöveges, rajzos, esetleg fényképes, sőt mozgóképes, de animált formában is gyorsan megkomponálható. Ennek a magas szintű nyelvnek a megtanulásához általában nem kell programozói ismeret, ugyanis a nyelv utasításait nem nehéz megjegyezni és alkalmazni. Így érthető, hogy ahol bevezették az oktatási programok írására a technológiát, ott már nem a programozók a vezéralakjai a tananyagkészítési munkának, hanem az adott tantárgyat jól ismerő például zenészek, irodalmárok, kémikusok, egyszóval a szaktanárok.

Ezeket a programozási eszközöket nem a felsorolt „játék"-, hanem PC kompatibilis gépekre tervezték, de hasonló szoftver hátteret biztosít az 1988 végén nálunk is induló postai videotex-szolgáltatás is. A terminálként alkalmazott MUPID ugyanis (lásd a µM 1987/10. számától kezdődő cikksorozatát) az AUTOOL nevű szerzői rendszerrel programozható.

Ezért az a véleményem, hogy sürgősen felül kellene vizsgálni az iskola-számítógépprogramot, meg kellene határozni, hogy a tanulási technológia szabta követelményekhez milyen számítógép-beszerzési politikát volna célszerű követni. Azt hiszem, hogy inkább kevesebb, de PC kompatibilis gépet és hálózatba kapcsolt vtx-terminálokat kellene vásárolni. Meg kell említenem, hogy a magánimportra és az értékesítésre vonatkozó jelenlegi szabályok (1988. február) a fejlődésüket illetően hátrányosan érintik az iskolák PC kompatibilis gépvásárlási törekvéseit, miután a rendelettel megszűnt az a forrás, ahonnan könnyen és gyorsan tudtak számítógépeket beszerezni.

Lehet, hogy nem volna szabad jósolni, de azt hiszem, hogy a „jövő" technologizált iskolája teljesen más lesz, mint a mai; nemcsak a környezet változik meg, de a tanár és a diák viszonya is. Feltehetően nem lesz ebben a jövőbeni iskolában a mai értelemben vett tanterem, a pedagógus nem tanít, hanem tananyagokat készít valamilyen hordozóra, legfeljebb néha tart egy-egy diáknak konzultációt. Ezekből az anyagokból a diák a tanár személyes közreműködése nélkül is képes lesz a szükséges ismereteket megszerezni. A diákok különféle tesztekkel ellenőrizhetik a tudásukat, és ha megfelelő módon felkészültek, akkor jelentkezhetnek az ugyancsak technologizált gépi vizsgára. Miután mindenki a tehetsége diktálta ütemben tanul, ezek a vizsgák nem lesznek időhöz kötve, mindenki akkor vizsgázik, amikor az anyagból jól felkészült. Ha valaki a vizsgán nem felelt meg, akkor a procedúrát annyiszor ismételheti, ahányszor csak akarja. Így azután a mércét igen magasra lehet állítani, nem kell félni a sok bukástól, mert valójában az nem is lesz.

A jövő iskolája már ma sem utópia, a megvalósításnak — azt hiszem, ebben egyetértünk — nem technikai, sokkal inkább emberi akadályai vannak. Talán ezért lehet csak lassan megvalósítani.

KOVÁCS GYŐZŐ

TechnoMIR

Egy érdekes bemenet

A megszakító bemenet (interrupt input vagy röviden ITIN) modul lényegét és fő felhasználását tekintve egy digitális bemenet. A különbség ehhez képest annyi, hogy a beérkező jel hatására a modul megszakítást kezdeményez. Ezzel kiküszöböli a DIN modul azon hátrányát, hogy elveszíti az információt, ha időben nem kérdezzük le. Meg lehet csinálni az ITIN-nel azt, amit a DIN-nel nem: valamilyen más feladat végrehajtása közben is végezhető számítógépes adatkezelés. Mivel elvész azonban az egyszerű programozhatóság, ezt a modult csak annak ajánljuk, aki igazán ismeri a számítógépét, és assemblerben is tudja programozni.

Az itt következők a HT—1080Z gépeken dolgozóknak segítenek, de mások is meríthetnek ötleteket belőle. A programot a modul feltalálói készítették, tőlük vettük át.

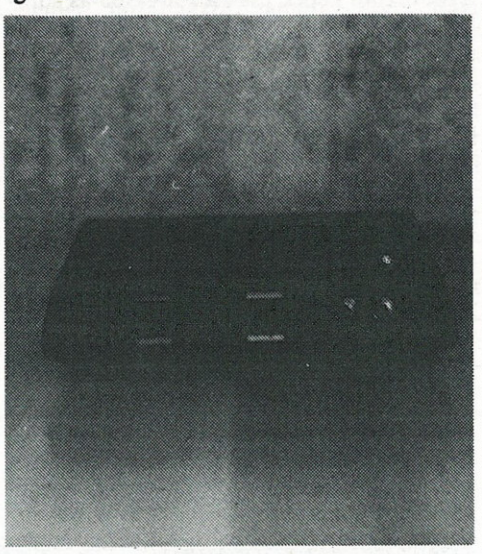
```
   Utasitas                    Ertelmezese

    M4012                 Cim beallitas, a
kepernyon megjelenik a cim es tartalma.
      C3
      00                  JP 7000(h) a prog-
                          ram kezdo cime
      70
       N                  Ujra megjelennek a
                          regiszterek
    M7000
      F3                  DI          IT le-
                                      tiltas
      DB                  IN A,(n)    INput
                                      ITIN
      19                  n=19(H)=25(d)
      32                  LD (nn),A
      0F                  nn=0C0E(h)  az ITIN
                                      tarol
                                      ma a
                                      keper-
                                      nyre
      3C
      FE                  CP n
      FF                  n=FF(h)=255(d)
      28                  JR Z,(VEGE)
      03                  relativ ugras a
                           VEGE cimkere
      C3                  JP nn
      01                   nn=7001(H)
      70
VEGE  FB                  EI          IT en-
                                      gedelye-
                                      zes
      C9                  RET
       N                  Ujra megjelennek a
                          regiszterek
    M7100
      F3                  DI
      ED                  IM1         IT mod
                                      beallitas
      56
      FB                  EI

  Itt folytatodhatnak a felhasznaloi

Program. Pl.:
      76                  HLT         A pro-
                                      cesszor
                                      var
      C3                  JP nn
      01                   nn=7101(H)
      71
       N
   07100 NL               Vezerles 7100(H)
                          cimre
```

Assembler program ITIN modul használatához

1. Csatlakoztassuk az ITIN modult a HT—1080Z számítógéphez.
2. Kapcsoljuk be a számítógépet. A NEW LINE gomb lenyomása után a képernyőn > READY látható.
3. Gépeljük be: SYSTEM és nyomjuk le a NEW LINE (a továbbiakban: NL) gombot.
4. A képernyőn megjelenő kérdőjel után gépeljük be: /12710 NL.
5. Monitor üzemben vagyunk, a képernyőn megjelennek a processzor regiszterei. Gépeljük be a *listát*.

Látszólag semmi nem történik. A processzor 7104(h) címen HLT utasításon áll. Ha rövidre zárjuk a modul valamelyik bemenetét, a képernyőn megjelenik a beolvasott kódnak megfelelő ábra. A program a RESET gombbal állítható meg.

Bővíthető a program például úgy, hogy HLT helyett az érkező kódnak megfelelő elágazásokat szervezünk.



A külső tápegység

A TechnoMIR rendszer eddig ismertetett digitális moduljai nem nagy áramfelvételűek, de a számítógépek szokásos terhelhetősége (kb. 0,15 A) nagyon könnyen túlléphető. A DINT és DOUT együttes alkalmazása, ha azok teljes terhelhetőségét kihasználjuk, legalább 0,2 A áramigényt jelent. Ez a számítógép buszrendszerén át nem elégíthető ki, külön táplálás kell.

A modul típusa	+5 V	+12 V	−12 V
HCLK	0,09 A	—	—
A/D	0,075 A	0,025 A	0,05 A
D/A	0,09 A	0,1 A	0,04 A
ITIN	0,14 A	—	—
AMUX	0,15 A	0,03 A	—
DINT	0,13 A	—	—
DOUT	0,15 A	—	—
JOY	0,085 A	—	—
PB	0,085 A+X	X	X
D100	0,025 A	—	—
C64	0,085 A	—	—
OPTO	0,12 A	—	—
UNI	0,11 A+X	X	X

Az analóg modulok működéséhez +12 V és −12 V szükséges; ezt a számítógépek nem, vagy nem elég pontosan adják. A különböző modulok áram- és feszültségigényét a *táblázat* mutatja. Látható, hogy általában két digitális modult használhatunk együtt, ennél többet csak a POW modullal együtt csatlakoztathatunk a számítógéphez.

A POW modul három változata közül a legegyszerűbb a *fényképen* látható. Ez az első is egyúttal. A modulhoz csatlakoztatott +8 V—+10 V egyenfeszültségből egy 7805 típusú stabilizátor IC állítja elő a +5 V-ot. Az analóg modulok +12 V, −12 V táplálásához 18 V-os váltakozó feszültséget kell a POW modulhoz csatlakoztatni.

Hamar kiderült azonban, hogy ez nem túl előnyös konstrukció, az iskolák zömében nem egyszerű feladat a szükséges feszültségeket „elővarázsolni". További gond, hogy nem elég „diákálló" és „tanárálló", nincs túláram és fordított polaritás elleni védelem. Ez a típus az OPI által kipróbálásra kiosztott készletekkel került forgalomba.

A hátrányos tulajdonságokat kiküszöbölendő fejlesztették ki a két újabb típust. Egy nem igazán esztétikus, szürke doboz tetején helyezték el az eredeti modult. Erre az alapdobozra vezették ki a gyártók a tápegység által előállított feszültségszintek csatlakozóit, a főkapcsolót és egy túláramjelző LED-et.

Az előállított feszültségek és terhelhetőségük:

+ 5 V referencia	100 mA
+ 5 V	1 A
+12 V	100 mA
−12 V	100 mA

A felhasználó szempontjából még lényegesebb változtatás a táplálást érinti. Az M6.1 POW A típusú modulhoz 220 V váltakozó feszültség szükséges. Az érintésvédelmi előírásoknak megfelelően nem csatlakoztatható közvetlenül a hálózathoz, leválasztó transzformátort kell alkalmazni. Az M6.2 POW B típust 24 V-tal kell táplálni, így az érintésvédelmi problémák is megoldódnak.

A POW modul alkalmazásával a TechnoMIR rendszer lehetőségei ugrásszerűen kiszélesednek. Gyakorlatilag bármekkora torony építhető a modulokból, azaz igen összetett érzékelési-beavatkozási műveletsor hajtható végre.

KIRÁLY LÁSZLÓ—ALBU LÁSZLÓ

Képernyő-kezelési ötletek

C128

A C128 képernyőkezelésére vonatkozó néhány tapasztalat, segédprogram, rutin remélhetőleg segítséget nyújt azoknak, akik a gép ilyen lehetőségeit is ki akarják használni.

Grafikus képernyő

Az első két ötlet a grafikus képernyő lehetőségeit bővíti.

A PRINT SHOP nevű programmal grafikát és nagyméretű betűkből álló szöveget készíthetünk, a KOALAPAINTER segítségével többszín-üzemmódban színes rajzokat állíthatunk elő. E két, C64-re készült népszerű program ábráit a C128-on is betölthetjük, de az utóbbinál egy kis átalakításra van szükség, mert míg a PRINT SHOP csak a bittérképet tárolja, addig a KOALAPAINTER ezt egyben veszi fel a két szín RAM-mal (együtt 40 blokk), amely a C128-on máshol helyezkedik el.

A PRINT SHOP-nál érdemes kiszínezni a képet. Ezt egy rövid gépi kódú rutinnal megtehetjük, a háttér és előtér legkedvezőbb összeállítását választhatjuk ki.

Indítás után a program *(1. lista)* először beállítja a gépi kódú rutint (200—250-es sor), majd megkérdezi a tárolt kép nevét.

```
10 REM **** PRINT SHOP-ATALAKITO ****
20 GRAPHIC 0,1:GOSUB 200
30 INPUT "NEV";N$
40 GRAPHIC 1,1:BLOAD (N$),B0,P8192
50 E=0:H=0
60 GETKEY A$
70 IF A$="E" THEN E=E+1
80 IF A$="H" THEN H=H+1
90 IF A$=CHR$(13) THEN GOTO 140
100 IF E=16 THEN E=0
110 IF H=16 THEN H=0
120 POKE 2827,H+E*16:SYS 2816:COLOR4,H+1
130 GOTO 60
140 BSAVE (N$),B0,P7168 TO P16192
150 GRAPHIC 0,1
160 END
200 FOR I=0 TO 24
210 READ A:POKE 2816+I,A
220 NEXT I:RETURN
230 DATA 169,28,133,251,169,0,133,250
240 DATA 162,4,169,1,160,0,145,250,200
250 DATA 208,251,230,251,202,208,244,96
```

1. lista

Betöltés után az E és H betűkkel növelhetjük az előtér, illetve a háttér színkódját. Ha megtaláltuk a megfelelő színkombinációt, cseréljünk lemezt, és nyomjuk meg a RETURN-t. Ekkor a gép felveszi a képet eredeti nevével, így csupán 4 blokkal bővült a fájl. Ezután a kép a következő utasítássorozattal bármikor betölthető:

GRAPHIC 1,1:BLOAD"név"

A gépi kódú listát a monitorba lépve a D B00 B18 paranccsal nézhetjük meg.

Fent említettek miatt a KOALAPAINTER-nél a dolog nem ilyen egyszerű. A C128 standard üzemmódú szín RAM-ja 7168—8167-ig, bittérképe 8192—16191-ig, a többszín-üzemmódban a további szín RAM (több szín 2) 55296—56295-ig terjed, így ezeket a részeket külön kell kimenteni.

Amikor a KOALAPAINTER-rel *(2. lista)* felveszünk egy képet, nem szükséges nevet adni neki; használhatjuk csak a betűjét

```
10 REM **** KOALAPAINTER-ATALAKITO ****
20 GRAPHIC 1:GRAPHIC 0,1:PRINT
30 INPUT "A KEP BETUJE";B$
40 INPUT "A KEP NEVE";N$
50 NE$=CHR$(129)+"PIC "+B$+" "+N$+"*"
60 PRINT "TEDD BE A LEMEZT!":GETKEY A$
70 BLOAD (NE$),B0,P4864
80 PRINT "TEDD BE A C128-AS LEMEZT!"
90 GETKEY A$
100 F$=B$+"-"+N$+" "
110 BSAVE (F$+"1"),B0,P4864 TO P12864
120 BSAVE (F$+"2"),B0,P12864 TO P13864
130 BSAVE (F$+"3"),B0,P13864 TO P14864
140 PRINT "KESZ,A TOLTOVEL BEOLVASHATO."
150 END
200 REM **** KOALAPAINTER-BETOLTO ****
210 GRAPHIC 0,1:PRINT
220 INPUT "A KEP BETUJE";B$
230 INPUT "A KEP NEVE";N$
240 INPUT "HATTERSZIN";H
250 F$=B$+"-"+N$+" "
260 GRAPHIC 3,1:POKE 0,44
270 COLOR 0,H+1:COLOR 4,H+1
280 BLOAD (F$+"1"),B0,P8192
290 BLOAD (F$+"2"),B0,P7168
300 BLOAD (F$+"3"),B4,P55296
310 SOUND 1,5000,10
320 GETKEY A$:GRAPHIC 0,1
```

2. lista

is azonosítóként a lemezen (SAVE vagy SAVE&NAME). Az átalakító rész a 10—150 sorokig terjed. Először meg kell adnunk a betűt, és ha van, a nevét (ha nincs, akkor a RETURN a válasz). Majd behelyezzük a lemezt, amin a kép van, s a program betölti. Ezután azt a lemezt tegyük be, amire fel akarjuk venni az átalakított képet, és nyomjunk meg egy gombot. Ha készen vagyunk, és ellenőrizni akarjuk a felvételt, GOTO 200-zal tölthetjük be a képet. Itt a háttérszínt is meg kell adni. Az olvasás végét a gép sípolással jelzi.

Ha más programban is használni akarjuk a többszínű képet, csak a 250—300-as sorokat kell meghagyni. A 260-as sor bekapcsolja a grafikus képernyőt és a be-kimenetet.

Karakteres képernyő (40 × 25)

Bizonyos programoknál szükség lehet a képernyő különböző irányú eltolásaira.

3. lista

```
10 REM **** SCROLL BALRA, JOBBRA ****
20 FOR I=0 TO 165:READ A:POKE 2816+I,A
30 S=S+A:NEXT I
40 IF S=25547 THEN GOTO 60
50 PRINT"HIBA VAN AZ ADATOKBAN.":END
60 KEY1,"SYS 2816"+CHR$(13)
70 KEY3,"SYS 2896"+CHR$(13)
80 SCNCLR:PRINT"F1=BALRA, F3=JOBBRA"
90 DATA 169,4,133,251,160,0,132,250,162
100 DATA 26,169,32,145,250,169,40,24,101
110 DATA 250,133,250,144,2,230,251,202
120 DATA 208,238,132,252,200,132,250,162
130 DATA 4,160,0,169,4,133,251,133,253
140 DATA 177,250,145,252,169,216,133,251
150 DATA 133,253,177,250,145,252,200,208
160 DATA 233,238,38,11,238,48,11,202,208
170 DATA 224,169,4,141,38,11,169,216,141
180 DATA 48,11,96
190 DATA 169,3,133,251,169,255,133,250
200 DATA 160,0,162,26,169,32,145,250,169
210 DATA 40,24,101,250,133,250,144,2,230
220 DATA 251,202,208,238,160,231,132,252
230 DATA 136,192,250,162,5,160,0,169,7
240 DATA 133,251,133,253,177,250,145,252
250 DATA 169,219,133,251,133,253,177,250
260 DATA 145,252,136,192,255,208,231,206
270 DATA 122,11,206,132,11,202,208,222
280 DATA 169,7,141,122,11,169,219,141
290 DATA 132,11,96
```

A le- és felfelé scrollt nem kell megírnunk, mert a rendszer rutinjai között megtalálhatók, így akár gépi kódú programból is meghívhatók.

scroll fel: SYS 51900

scroll le: SYS 51914

A balra, illetve jobbra történő görgetést már nekünk kell elkészíteni.

A BASIC betöltőprogram *(3. lista)* jelzi az esetleges adatbegépelési hibákat. Az F1 gombbal balra, az F3 gombbal jobbra tolhatjuk a képernyő tartalmát. A szín RAM-ot is kezeli a rutin, tehát a karakterek színükkel együtt mozognak. A két gépi kódú lista monitor üzemmódban a D B00 B4F, illetve a D B50 BA5 parancsokkal írható ki. Felvehető a BSAVE "név", B0,P2816 TO P2982 utasítással.

A következő rutinnal a további négy képernyőt használhatjuk, melyek között az F1, F3, F5, F7 gombokkal kapcsolhatunk át. Ez hasznos lehet hosszú programok hibakeresésénél, belövésénél, mert fontosabb részeit külön oldalakra listáztathatjuk. Ha az eredeti képet szeretnénk visszakapni, üssük le a HELP billentyűt. Ez egyben visszaállítja a megszakításvektort, és az eredeti állapotba jutunk. Egy-egy átkapcsolás után nyomjuk meg a CRSR—HOME gombot, csak ekkor jelenik meg a villogó kurzor.

```
10 REM **** 4 KEPERNYO ****
20 FOR I=4096 TO 4105:POKE I,0:NEXT I
30 FOR I=0 TO 115:READ A:POKE 2816+I,A
40 S=S+A:NEXT I
50 IF S=11096 THEN PRINT"RENDBEN.":END
60 PRINT"HIBA VAN AZ ADATOK KOZOTT."
70 DATA 120,169,13,141,20,3,169,11,141
80 DATA 21,3,88,96,72,138,72,166,213
90 DATA 224,4,208,13,169,132,141,44,10
100 DATA 169,32,141,59,10,76,110,11,224
110 DATA 5,208,13,169,148,141,44,10,169
120 DATA 36,141,59,10,76,110,11,224,6
130 DATA 208,13,169,164,141,44,10,169,40
140 DATA 141,59,10,76,110,11,224,3,208
150 DATA 13,169,180,141,44,10,169,44,141
160 DATA 59,10,76,110,11,224,64,208,20
170 DATA 169,20,141,44,10,169,4,141,59
180 DATA 10,169,101,141,20,3,169,250,141
190 DATA 21,3,104,170,104,76,101,250
```

4. lista

A betöltő begépelése előtt írjuk be:

POKE 45,1:POKE 46,49:POKE 12544,0:NEW

Ez feljebb teszi a BASIC-kezdetet. A grafikus képernyőt így nem használhatjuk: itt vannak a képernyők és a program.

A program *(4. lista)* törli a funkciós billentyűket és a HELP gombot (20-as sor). Jelzi az adathibákat. A RENDBEN felirat megjelenése után SYS 2816-tal indíthatjuk a rutint. A gépi kódú listát megtekinthetjük a monitorba lépve a D B00 B71 paranccsal. Felvehető a BSAVE "név", B0, P2816 TO P2932 utasítással.

Van néhány tárcím — a gép rendszerváltozói —, amelyek átírásával érdekes dolgokat érhetünk el.

A betűismétlés várakozási idejét a 2595-ös címen írhatjuk át. A 2598-as tárcím tartalmát 64-re változtatva kikapcsolható a kurzor villogása. Ha nullát írunk a 2599-es regiszterbe, akkor a kurzor a program futása közben is láthatóvá válik.

NÉMETH ISTVÁN

Pardon!

Sajnálatos módon hiba került az 1988/2. számunk 8. oldalán megjelent, Egy kis zene a Plus/4-en című cikkbe. A jelzett lapoldal felső és alsó programja egyenértékű; az utóbbi egy elegánsabb megoldás akart lenni. Eleganciáját rontják a benne található rossz ugrási címek, valamint a gépi kódú lista hiánya.

A hibás sorok ugráscímei helyesen:
a 270-es sorban 250
a 320-as sorban 300

A dallamot a SYS 16718-as utasítással kapcsolhatjuk ki. Végezetül a programokhoz tartozó gépi kódú listát is közöljük. A „zenebonáért" olvasóink szíves elnézését kérjük.

A szerkesztőség

A gépi kódú lista

```
>4020 20 91 94 20 E1 9D A5 15 :
>4028 85 D9 A5 14 85 D8 20 91 :
>4030 94 20 E1 9D A5 15 85 DB :
>4038 A5 14 85 DA 78 A9 95 8D :
>4040 14 03 A9 40 8D 15 03 58 :
>4048 AD 12 FF 29 FC 8D 12 FF :
>4050 AD 10 FF 29 FC 8D 10 FF :
>4058 A9 01 8D 00 40 8D 01 40 :
>4060 8D 02 40 8D 03 40 A0 00 :
>4068 B1 D8 0D 12 FF 8D 12 FF :
>4070 A5 D8 18 69 01 85 D8 A8 :
>4078 00 65 D9 85 D9 A0 00 B1 :
>4080 DA 0D 10 FF 8D 10 FF A5 :
>4088 DA 18 69 01 85 DA A9 00 :
>4090 65 DB 85 DB 60 48 98 48 :
>4098 8A 48 CE 00 40 10 42 A9 :
>40A0 EF 2D 11 FF 8D 11 FF AD :
>40A8 02 40 F0 35 CE 02 40 F0 :
>40B0 08 A9 00 8D 00 40 4C E1 :
>40B8 40 A0 00 B1 D8 8D 0E FF :
>40C0 C8 B1 D8 8D 00 40 C8 B1 :
```

```
>40C8 D8 8D 02 40 AD 11 FF 09 :
>40D0 1F 8D 11 FF A9 03 18 65 :
>40D8 D8 85 D8 A9 00 65 D9 85 :
>40E0 D9 CE 01 40 10 42 A9 DF :
>40E8 2D 11 FF 8D 11 FF AD 03 :
>40F0 40 F0 35 CE 03 40 F0 08 :
>40F8 A9 00 8D 01 40 4C 28 41 :
>4100 A0 00 B1 DA 8D 0F FF C8 :
>4108 B1 DA 8D 01 40 C8 B1 DA :
>4110 8D 03 40 AD 11 FF 09 2F :
>4118 8D 11 FF A9 03 18 65 DA :
>4120 85 DA A9 00 65 DB 85 DB :
>4128 AD 02 40 D0 13 AD 03 40 :
>4130 D0 14 78 A9 0E 8D 14 03 :
>4138 A9 CE 8D 15 03 58 AD 11 :
>4140 FF 29 CF 8D 11 FF 68 AA :
>4148 68 A8 68 4C 0E CE 78 A9 :
>4150 00 8D 11 FF A9 0E 8D 14 :
>4158 03 A9 CE 8D 15 03 58 60 :
>4160 0E CE 00 FF 00 FF 00 FF :
>4168 00 FF 00 FF 00 FF 00 FF :
```

BASIC Merge Primón

A Merge olyan összefűzési lehetőség, amivel a már tárban lévő programhoz magnóról tölthetünk más programrészeket. A művelet igen hasznos lehet, amikor például valaki összerak magának egy „szerszámosládát". Ez ugyanis mindazt a programelemet tartalmazza, amely valamennyi programban egyforma, így elég egyszer megírni. Ezután ezt egy félkész programhoz töltve, nem kell listabeírással vagy újraírással foglalkozni, mert igény szerint egyszerűen csak elővesszük a „szerelvényeket", és azokat használjuk. Az alkalmazás úgy istenigazából jó szórakozás.

De menjünk tovább egy egyszerű megoldásra! Íme a Merge. A BASIC program kezdőcímét a memóriában a 16548-as bájton kezdődő kétbájtos mutató adja meg, ezt kiíratva jegyezzük fel: PRINT PEEK (16548)+256*PEEK (56549), ami általában 17386. Ugyanígy írassuk ki a 16633 címen kezdődő kétbájtos mutatót is. Ez két bájttal az utolsó programsor mögé mutat. Ezért a BASIC kezdete mutatóba két bájttal kevesebbet kell írnunk: POKE 16548,x—256*int(x/256),x/256, ahol x az érték.

A gépben lévő program nem listázható és nem is indítható. Tehát beírandó a LOAD, majd betöltés után vissza kell írni a 16548-as címre az értékeket. Ügyeljünk arra, hogy a második program legkisebb sora is nagyobb legyen, mint az előző programé, mert különben csúnya keveredések lehetnek az azonosságból.

LILING ZOLTÁN

TOP-lista

A külföldi szaklapok példáiból okulva, mi is indítani kívánunk egy TOP-listát, két kategóriában. Az elsőben a felhasználói, a másodikban a játékprogramok szerepelnek. A legjobbakat, a legérdekesebbeket rangsorolnánk a beküldött javaslatok alapján. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprisera, TVC-re, ATARI-ra és IBM-re készült programrangsorokat várunk havonta.

Címünk: 1371 Bp. Pf.: 433. Mikroszámítógép Magazin Szerkesztősége.

Diákszerkesztőség

Felhasználói programok

1. GEOS 1.3	C64
2. ART STUDIO 1.2B	C64, Spectrum
3. CRACKER COPY II	C64
4. PRINT FOX	C64
5. MUSIC SHOP	C64
6. LIGHT SHOW	Spectrum
7. ARTIST SHOW	Spectrum
8. LASER-GENIUS	Spectrum
9. GAME MAKER	C64
10. SPEED LOAD	C64

Játékprogramok

1. DEFENDER of the CROWN	C64
2. WIZBALL	C64
3. THE LAST NINJA	C64
4. TETRIS	IBM
5. ARKANOID	C64
6. BETTY	Spectrum
7. BARBARIAN	Spectrum
8. EXOLON	Spectrum
9. SPACE GAMES	C64
10. WORLD GAMES	C64

Rajzolóprogram Plus/4-re

```
1 VOL8
5 A=10:B=10
10 COLORO,3,3:COLOR1,8,7:COLOR4,7,1:SCNCLR
20 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
30 FORI=1TO19:READ A$
40 A=A+1:FORU=1TO200:NEXTU:SOUND1,680,2
50 CHAR1,A,B,A$,1
60 NEXTI
70 DATA "R","A","J","Z","O","L","O","!","K","E","S","Z","I","T","E","T","T","E",
":"
72 B=12:A=11
80 FORT=1TO11:READ F$
90 CHAR1,A,B,F$
100 A=A+1:FORF=1TO200:NEXTF:SOUND1,300,3
110 NEXTT
120 DATA"B","A","C","S","I"," ","P","E","T","E","R"
130 GETKEY A$:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":COLORO,2,7
131 PRINT"▒▒              * * * M E N U * * *"
132 PRINT"▒              1. DELKELET  "
133 PRINT"▒              2. ESZAKNYUGAT"
134 PRINT"▒              3. DELNYUGAT"
135 PRINT"▒              4. ESZAKKELET"
136 PRINT"▒              5. BALRA"
137 PRINT"▒              6. JOBBRA"
138 PRINT"▒              7. LEFELE"
139 PRINT"▒              8. FOLFELE"
140 PRINT"▒              £. VISSZA A MENUBE"
141 PRINT"▒              V. SZAGGATOT VONAL"
142 PRINT"▒              E. NORMAL VONAL"
143 PRINT"▒              T. TISZTA KEPERNYO"
144 GETKEY A$:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
145 PRINT"▒              F.  RADIR "
147 PRINT"▒              G.  CERUZA"
148 GETKEY A$:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":GOTO 155
155 PRINT"▒         ▒U▒D▒ R▒A▒J▒Z▒O▒L▒A▒S▒T▒!"
156 FORI=1TO8
157 FORT=1TO7
158 COLOR4,I,T
159 FORG=1TO100:NEXT G
160 NEXTT
161 NEXTI
162 GETKEY A$
499 COLOR1,1,1:COLOR4,7,1
500 GRAPHIC1,1:X=160:Y=100
510 GETKEY A$
511 IF A$="V" THEN D=4
512 IF A$="E" THEN D=1
520 IF A$="1"  THEN X=X+D:Y=Y+D
530 IF A$="2"  THEN X=X-D:Y=Y-D
540 IF A$="3"  THEN X=X-D:Y=Y+D
550 IF A$="4"  THEN X=X+D:Y=Y-D
560 IF A$="5"  THEN X=X-D
570 IF A$="6"  THEN X=X+D
580 IF A$="7"  THEN Y=Y+D
581 IF A$="T" THEN SCNCLR
590 IF A$="8"  THEN Y=Y-D
591 IF A$="£" THEN GRAPHICO:GOTO 120
592 IF A$="F" THEN R=0
594 IF A$="G" THEN R=1
620 DRAWR,X,Y:GOTO 510
```

A programmal nyolc irányba rajzolhatunk a képernyőre. Sőt radírozhatunk is. Ez utóbbit, ha tisztán kívánjuk alkalmazni, akkor javítsuk át a következő sorokat:

```
592 IF A$ = "F" THEN COLOR 1,2
594 IF A$ = "G" THEN COLOR 1,1
620 DRAW 1,X,Y: GOTO 510
```

Ezenkívül a V billentyűvel szaggatott, az E billentyűvel normál vonalas üzemmódot választhatunk. A £ billentyűvel visszatérhetünk a menübe. De vigyázat, ez esetben elvész a rajzunk! A T billentyű tiszta képernyőt hoz elénk.

Amikor a program átvált grafikus képernyőre, akkor addig nyomogassuk a G és az E billentyűket, amíg meg nem jelenik a grafikus kurzor!

BÁCSI PÉTER

Pályázat számítástechnikai klubok részére

A középfokú oktatási intézmények számítástechnikai klubjainak munkáját technikai feltételeik javításával is szeretnénk támogatni, ezért részükre mikroszámítógépeket, illetve kiegészítő berendezéseket adunk használatba, meghatározott időre: minden tanévben szeptember 1-jétől június 10-ig.

E támogatás pályázat útján nyerhető el, melyet minden évben április 31-ig kell beküldeni a következő címre:

KISZ KB KSZTT 1388 Budapest Pf. 72

A pályázatok tartalmazzák a következőket:

- **a klubot működtető szerv, iskola nevét és címét,**
- **a klubfoglalkozások, illetve a nyitva tartás helyét és idejét,**
- **a klubvezető(k) nevét,**
- **a klub célkitűzéseit, esetleg programját,**
- **a klubtagság, illetve a klub látogatásának feltételeit,**
- **a rendelkezésre álló gépek számát, típusát, és hogy milyen kiegészítő berendezések segítenék legjobban a klub munkáját.**

KISZ KB KSZTT

Egy kis hardver

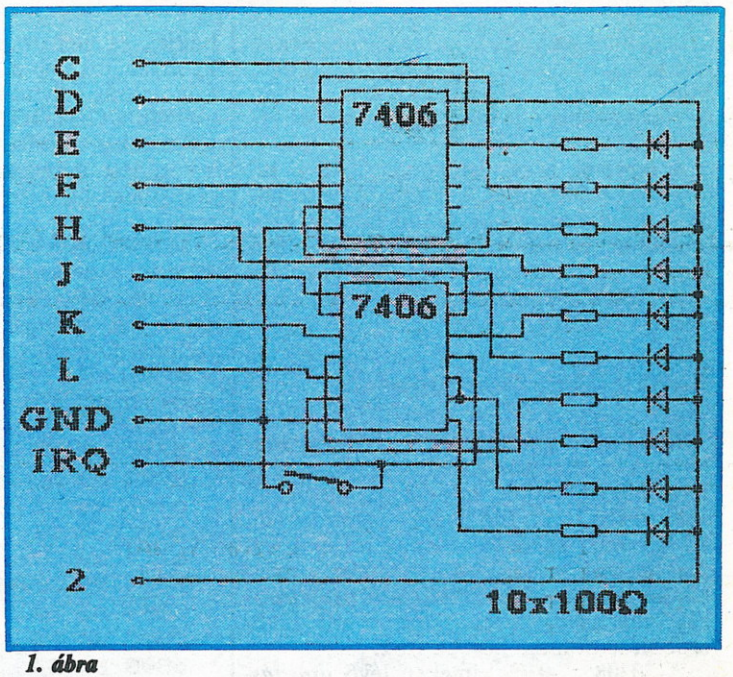
Az alábbi kiegészítést elsősorban azoknak ajánljuk, akik nem teljesen járatlanok a hardver területén. Ugyanakkor kellő odafigyeléssel természetesen bárki elvégezheti a javasolt módosítást Commodore gépében. Segítségével nyomon követhető a párhuzamos adatforgalom a user porton, illetve a floppy működése a megszakításokon keresztül. Lehetőségünk nyílik továbbá arra, hogy a gép működését bármely állapotban felfüggesszük. (A programok nagy része addig marad „lefagyasztott" állapotban, amíg a kapcsolót bekapcsolva tartjuk.)

A lista olyan programot tartalmaz, amely egy futófényt vezérel. A sebességet a 49175 tárcím módosításával változtathatjuk. Természetesen nemcsak 44 darab lépés programozása lehetséges. Ha megtervezünk egy villogási sorozatot, akkor a lépéseket sorban az 50000 címtől kezdődően rögzítjük, a darabszámot pedig a 49192 címre írjuk. (Például amikor azt szeretnénk, hogy az első és az utolsó LED villogjon felváltva, akkor a következőket kell beírni: POKE 50000,1 : POKE 50001,128:POKE 49192,2.) Mivel a program belső változói futás közben módosulnak, célszerű beírni az adatokat a SYS 49152 meghívása előtt. A program egyébként megszakításvezérelt, így újraindítását is az előbb említett utasítással érhetjük el.

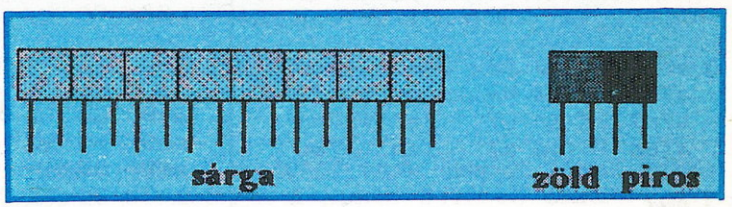
Sajnos, a LED-eket csak a régi típusú számítógép belsejébe lehet beépíteni, akiknek viszont újabb típusú C64-ük van, azok a fentiek egy részét a user portra csatlakoztatott futófény NYÁK-kal valósíthatják meg. AZ IRQ kapcsolót a bővítés helyére lehet csatlakoztatni. Mivel ezek egymástól szétválasztható bővítések, különkülön is elkészíthetők.

A megvalósítás figyelmet igényel. A konstrukciót azonban mindenki maga választhatja ki! Aki nem tud nyomtatott áramkört maratni, az egyszerűen forrasszon egy IC-foglalat lábaira vezetékeket. (Én is ezt a módszert választottam.)

```
10 REM *** BB SOFTWARE & ATOMHARD ***
11 FOR I=0 TO 68
12 READ A : POKE 49152+I,A : S=S+A
13 NEXT
14 IF S<>8806 THEN PRINT"HIBA !":STOP
15 FOR I=0 TO 43
16 READ A : POKE 50000+I,A
17 NEXT
18 SYS  49152
19 REM  -GEPI KODU PROGRAM-
20 DATA 120,169,255,141,3,221,169,21
21 DATA 141,20,3,169,192,141,21,3,88
22 DATA 96,234,234,234,234,162,15,224
23 DATA 20,240,6,238,25,192,76,49,234
24 DATA 169,1,141,25,192,162,44,224,44
25 DATA 240,15,174,42,192,189,79,195
26 DATA 141,1,221,238,42,192,76,49,234
27 DATA 169,0,141,42,192,76,48,192,234
28 REM  -FUTOFENY ADATOK-
29 DATA 0,128,64,32,16,8,4,2,1,129,65
30 DATA 33,17,9,5,3,131,67,35,19,11,7
31 DATA 135,71,39,23,15,143,79,47,31
32 DATA 159,95,63,191,127,255,254,252
33 DATA 248,240,224,192,128
```



1. ábra



2. ábra

Elsőként a gép fedőlapjáról távolítjuk el a klaviatúrát, melyet csupán néhány csavar rögzít. Majd szikével vágjunk ki a szellőzőnyílásszerű középső mélyedéséből egyszer 1, egyszer pedig 4,1 centimétert. A vágásokat felülről csináljuk. Először hosszirányban, utána keresztirányban. A keletkezett résekbe kerülnek be a világító diódák, melyeket előzőleg pillanatragasztóval ragasszunk össze a 2. ábrának megfelelően. Az oldalaikat ezután dörzspapírral csiszoljuk simára és méretre úgy, hogy pontosan illeszkedjenek a kialakított résbe. A LED-ek rövidebb lábait vágjuk még kisebbre, s forrasszuk fel az ellenállásokat, a hosszabbik lábakat forrasszuk össze. Ha nem készítettünk nyomtatott áramkört, akkor az 1. ábrán látható kapcsolási rajznak megfelelően huzalozzuk be az alkatrészeket, majd ragasszuk fel őket. (A kapcsolási rajzon szereplő legalsó dióda piros, a felette lévő zöld, az összes többi sárga színű, de mindegyik szögletes.) A diódák beragasztásánál ügyeljünk arra, hogy tetejük egy síkba essen a gép dobozának felületével. A kapcsolók hátoldalhoz való rögzítése ezek típusától függően változhat. Az áramkörünk felől érkező vezetékeket ne forrasszuk rá közvetlenül az alaplapra, hanem a szétszedhetőség miatt, lehetőség szerint, iktassunk be egy csatlakozópárt is. (Vigyázat, a C-től L-ig terjedő user port kimenetek az alsó oldalon találhatók!)

A fent javasolt módosítás a gép alapvető működésére nincs hatással, tehát ezzel üzemképtelenné vagy bizonytalan működésűvé tenni nem lehet. Viszont akinek beválik a változtatás, annak ajánljuk, hogy lépjen tovább a kazettavonalak kivezetésével, netán az audiovonal hangszóróra vezetésével, aminek persze csak erősítés mellett van értelme.

BÁRTFAI BARNABÁS

BASIC és gépi kód

Legutóbb az USR függvény lehetőségeit kiterjesztő rutinok C64-es változatát láttuk. Most — a múltkori assembly lista alapján — részletesebben is megvizsgáljuk a működésüket.

A mostani 1—4. listákon a BASIC interpreter ROM-jának egy-egy részlete látható: azok, amelyekhez az új rutinok kapcsolódnak. A leírásban többnyire az előző részben közölt listák címeire hivatkozom, a kivételeket a címtartomány eltérése miatt könnyű felismerni.

A vektorokat átíró rutin

Működése egyszerű: az A és Y regiszterbe a saját eljárások kezdőcímének alsó és felső bájtját töltjük, és onnan a megfelelő vektorokba írjuk. Illő lenne a vektorok előző tartalmát valahova eltenni, mi *most* ezt nem tesszük meg. Ezt a — sajnos gyakran alkalmazott — módszert „önző programozás"-nak neveztem el. Hátrányairól, kiküszöböléséről majd később írok.

A $Ø3Ø8...Ø3Ø9 címeken lévő *utasításértelmezés* vektor eredeti tartalma $A7E4 volt, ez az, amit felülírunk a DEF USR utasítást feldolgozó rutin kezdőcímével. A $Ø3ØA...Ø3ØB címeken található *kifejezéskiértékelés* vektorba a $AE86 helyére az USRn függvényeket kezelő eljárás címe kerül.

```
a7e1 6c 08 03    jmp ($0308)
a7e4 20 73 00    jsr $0073
a7e7 20 ed a7    jsr $a7ed
a7ea 4c ae a7    jmp $a7ae
```

1. lista

A DEF USR feldolgozó rutinja

Az *1. listán* látható ROM-részletbe szúrjuk be. Az első utasítás — a CHRGET hívása — mindkét helyen megegyezik.

$Ø351..Ø35B: a CHRGET-tel beolvassuk az utasítás első szóköztől különböző bájtját, és megvizsgáljuk, hogy az a DEF kulcsszó tokenje-e. Ha igen, $Ø35E-re, a feldolgozás folytatására ugrunk, ellenkező esetben a CHRGOT-tal visszaállítjuk a feltételbiteket, és a ROM $A7E7 címére adjuk a vezérlést. A beszúrás helyét követő utasítást kihagyjuk, hiszen a párját az imént hajtottuk végre.

$Ø361...Ø365: CHRGET-tel a következő bájtot olvassuk, és megnézzük, hogy nem az USR tokenje-e. Ha nem, akkor a DEF FN utasítást feldolgozó interpreter rutinnal folytatjuk a feldolgozást.

$Ø368: az A regiszter tartalmát egy — a nulla lapon levő — szabad tárhelyre teszszük. A tárolt értéknek nincs jelentősége, csak annak, hogy a 7. bit 1-re van állítva, vagyis előjeles számként kezelve negatívnak tekintendő.

```
b7f7 a5 66       lda $66
b7f9 30 9d       bmi $b798
b7fb a5 61       lda $61
b7fd c9 91       cmp #$91
b7ff b0 97       bcs $b798
b801 20 9b bc    jsr $bc9b
b804 a5 64       lda $64
b806 a4 65       ldy $65
b808 84 14       sty $14
b80a 85 15       sta $15
b80c 60          rts
```

2. lista

```
ae83 6c 0a 03    jmp ($030a)
ae86 a9 00       lda #$00
ae88 85 0d       sta $0d
ae8a 20 73 00    jsr $0073
ae8d b0 03       bcs $ae92
```

3. lista

$Ø36A...Ø36F: a BASIC program következő bájtját olvassuk, és a feltételbitek állásától függően háromféle folytatás közül választhatunk. Ezek:

Ha a Z bit 1-re van állítva, akkor utasítás végéhez értünk, ami szintaktikai hibát jelent, ezért a hibarutinra ugrunk. Ha a C bit értéke 1, ez azt jelenti, hogy az A regiszterbe töltött karakter nem ASCII számjegy. Ekkor az eredeti USR függvény címének definiálását feltételezve az egyenlőségjel meglétének vizsgálatára ugrunk. C=Ø esetén viszont a soron következő utasítással folytatódik a program.

$Ø371...Ø373: az ASCII számjegyet binárisra alakítjuk. Ez azért megy ilyen egyszerűen, mert a számjegyek ASCII KÓDJA $3Ø-cal különbözik a bináris értéküktől. A felső félbájt nullázásával éppen a bináris értéket kapjuk meg. Ezt az ASL utasítás segítségével megszorozzuk kettővel. Azért van erre szükség, mert egy olyan táblázathoz fogjuk indexként használni, melynek elemei kétbájtosak.

$Ø374...Ø376: a bináris számot a már korábban említett Ø lapos címen tároljuk, egy pozitív értékkel felülírva az eddigi, negatív tartalmat. Ezután a CHRGET-tel a következő karaktert olvassuk az A regiszterbe.

$Ø379...Ø37D: megvizsgáljuk, hogy az A-ban az egyenlőségjel tokenje van-e. Ha nem, a hibarutinra ugrunk, egyébként a CHRGET-tel újabb karaktert olvasunk be.

```
afd1 20 f1 ae    jsr $aef1
afd4 68          pla
afd5 a8          tay
afd6 b9 ea 9f    lda $9fea,y
afd9 85 55       sta $55
afdb b9 eb 9f    lda $9feb,y
afde 85 56       sta $56
afe0 20 54 00    jsr $0054
afe3 4c 8d ad    jmp $ad8d
```

4. lista

$Ø38Ø...Ø383: két — korábban már megismert — interpreter rutint hívunk. Az FRMNUM az egyenlőségjelet követő aritmetikai kifejezést kiértékeli, és az eredményt a lebegőpontos akkumulátorba (FAC-ba) viszi. A GETADR a FAC tartalmát 16 bites pozitív egész számmá alakítja, és a $14...15 címekre teszi le. A GETADR a *2. listán* látható, és megfigyelhetjük, hogy a számot az A és Y regiszterbe viszi át. Mi a következő lépésekben kihasználjuk, hogy a regiszterek tartalma nem változik, tehát közvetlenül a regiszterekből tesszük a címet a táblázatba.

5. lista

```
100 rem usrdef C64
110 k=828 : print chr$(147)
120 def fnd(x)=x-48+7*(x>64)
130 for l=1 to 15 : print chr$(19);k;
140 s=0 : read a$,b
150 for i=1 to len(a$) step 2
160 a=16*fnd(asc(mid$(a$,i,1)))+fnd(asc(
mid$(a$,i+1,1)))
170 poke k,a : s=s+a : k=k+1 : next
180 if b<>s then print "hiba!" : stop
190 next
200 sys 828 : end
828 data a951a0038d08038c0903a99e,1044
840 data a0038d0a038c0b0360207300,714
852 data c996f0062079004ce7a72073,1371
864 data 00c9b7f0034cb3b385022073,1343
876 data 00f068b008290f0a85022073,876
888 data 00c9b2d05a207300208aad20,1199
900 data f7b7a602300a9ddb03989dda,1562
912 data 034ceaa78d1203988d11034c,1031
924 data eaa7a900850d207300c9b7d0,1455
936 data 0f207300f029900ea57ad002,1098
948 data c67bc67a2079004c8dae290f,1241
960 data 0a4820730020f1ae68a8b9da,1351
972 data 038555b9db0385564ce0af4c,1398
984 data 08af48b248b248b248b248b2,1433
996 data 48b248b248b248b248b2,1250
```

```
100 rem usrdef vc20
110 k=828 : print chr$(147)
120 def fnd(x)=x-48+7*(x>64)
130 for l=1 to 15 : print chr$(19);k;
140 s=0 : read a$,b
150 for i=1 to len(a$) step 2
160 a=16*fnd(asc(mid$(a$,i,1)))+fnd(asc(
mid$(a$,i+1,1)))
170 poke k,a : s=s+a : k=k+1 : next
180 if b<>s then print "hiba!" : stop
190 next
200 sys 828 : end
828 data a951a0038d08038c0903a99c,1042
840 data a0038d0a038c0b0360207300,714
852 data c996f0062079004ce7c72073,1403
864 data 00c9b7f0034cb3d385fe2073,1627
876 data 00f066b008290f0a85fe2073,1126
888 data 00c9b2d058207300208acd20,1229
900 data f7d7a6fe300a9dd903989dd8,1842
912 data 034ceac785029885014ceac7,1442
924 data a900850d207300c9b7d00f20,1101
936 data 7300f029900ea57ad002c67b,1372
948 data c67a2079004c8dce290f0a48,1034
960 data 20730020f1ce68a8b9d80385,1435
972 data 55b9d90385564ce0cf4c08cf,1507
984 data 48d248d248d248d248d248d2,1692
996 data 48d248d248d248d2,1128
```

6. lista

```
100 rem usrdef c16
110 k=828 : print chr$(147)
120 for l=1 to 15 : print chr$(19);k;
130 s=0 : read a$,b$ : b=dec(b$)
140 for i=1 to len(a$) step 2
150 a=dec(mid$(a$,i,2))
160 poke k,a : s=s+a : k=k+1
170 next
180 if b<>s then print "hiba" : stop
190 next
200 sys 828 : end
828 data a951a0038d08038c0903a99b,411
836 data a0038d0a038c0b0360207304,2ce
844 data c996f0062079044cd98b2073,535
852 data 04c9b7f0034c9d9a85e82073,5fa
860 data 04f065b008290f0a85e82073,453
868 data 04c9b2d05720730420e19da6,581
876 data e8300a9dd803989dd7034cdc,5d1
884 data 8b8d0205988d01054cdc8ba9,4a6
892 data 00850d207304c9b7d00f2073,41b
900 data 04f029900ea53bd002c63cc6,535
908 data 3b2079044c1e94290f0a4820,280
916 data 730420859468a8b9d7038555,52d
924 data b9d80385564cec954ca1941c,5d9
932 data 991c991c991c991c991c991c,43e
940 data 991c991c991c99,2b8
```

7. lista

$0386 ... 0388: a korábban eltett értéket az X regiszterbe töltjük. Ha pozitív, akkor a táblázatban indexelünk vele, ha negatív, akkor az eredeti USR függvény feldolgozásánál folytatjuk.

$038A ... 0391: X-szel indexelve az A és Y regiszterekből a definiált címet a címtáblázatba írjuk, végül visszalépünk az interpreter ciklusba.

$0394 ... 039B: annyiban különbözik az előző bekezdésben leírttól, hogy a definiált címet — indexelés nélkül — a „hivatalos" USR vektorba írja.

A USRn függvényeket kiértékelő rutin

$039E ... 03A2: ez a három utasítás megegyezik azzal, ami a beszúrás helyén a ROM-ban van; a típusjelzőbe, ami a $0D címen van, 0-t, a numerikus típus kódját töltjük, és a CHRGET rutinnal beolvassuk az A regiszterbe a következő nem szóköz bájtot.

$03A5: megvizsgáljuk, hogy a regiszterben az USR kulcsszó tokenje van-e, ha nem, $03B8-ra ugrunk.

$03A9 ... 03AE: a CHRGET-tel újabb bájtot olvasunk be, és a feltételbitek állásától függően — ugyanúgy, mint a függvénydefiniálást feldolgozó rutinban —, háromféle folytatás közül választhatunk. Ezek:

A Z bit 1 értéke itt is hibát jelent, a futás a hibarutinon folytatódik. A C bit törölt állapota USRn típusú függvényre utal — a feldolgozó rutin folytatására ugrunk. Ha a C bit be van állítva, akkor viszont az eredeti USR függvényről van szó. Szintaktikai ellenőrzését és feldolgozását az interpreterre bízzuk. Ezt készíti elő a $03B0 ... 03B6 címeken található utasításcsoport: a BASIC szövegmutató (TXTPTR) értékét eggyel csökkenti.

$03B8 ... 03BB: a CHRGOT rutinnal az aktuális bájttartalmat újra betöltjük az A regiszterbe, és a ROM $AE8D címén folytatódik a feldolgozás. A *3. listáról* leolvashatjuk, hogy a $AE86 ... AE8A címeken levő utasításokat átugrottuk, mert feladatukat a mi rutinunk elvégezte. Ne felejtsük el, hogy az eredeti USR esetén A-ban az USR tokenje van, mert a $03A9-en levő CHRGET-hívást a TXTPTR visszaállításával hatástalanítottuk.

$03BE ... 03C2: a korábban látott módon az ASCII számjegyet binárisra alakítjuk, megszorozzuk kettővel, majd a veremben tesszük megőrzésre. A CHRGET hívásával beolvassuk a következő karaktert, melynek a szintaktikai szabályok szerint nyitó zárójelnek kell lennie.

A következő hét utasítás — nem véletlenül — kísértetiesen hasonlít a *4. listán* látható ROM-lista utasításaira, csupán az indexelt LDA utasítások operandusaiban térnek el egymástól.

$03C5: egy olyan ROM-rutint hívunk meg, mely egy zárójeles kifejezést olvas be a BASIC programszövegből, közben ellenőrzi a zárójelek meglétét is.

$03C8 ... 03D4: a veremből kivesszük a megőrzött sorszámot és az Y regiszterbe tesszük. Az USR táblázatból Y-nal indexelve kiemeljük a hívott függvény címét és a $55 ... 56 címeken tároljuk. Befejezésül az interpreternek arra az utasítására ugrunk, amely a függvény kiértékelése után folytatja az azt tartalmazó kifejezés kiértékelését.

$03D7: közös hibarutin, közvetlen ugrás az interpreter SYNTAX ERROR-t kiíró utasítására.

A betöltőprogramok

Az *5—7. listákon* a BASIC nyelvű betöltőprogram három géptípushoz tartozó változata van. Újszerű az adatok formája. Ennek feldolgozása a korábbinál lassúbb, de a begépelésnél fáradságot takarítunk meg. A programok működéséről a következő alkalommal írok részletesebben, most csak néhány megjegyzést teszek.

A betöltési idő kicsit több 6 másodpercnél, a C16-os változaté kb. a fele. A merev képernyő előtt ülve ez nagyon hosszúnak tűnik, ezért a képernyő sarkában mindig kiíródik a feldolgozás alatti DATA utasítás sorszáma. Soronként halad az ellenőrzés, hiba esetén a hibás adatot tartalmazó sor száma után megjelenik a „hiba!" felirat, és a program megáll. Javítás után újraindítható.

Futtatás előtt célszerű a programot tárolni; különösen fontos ez, ha mágnesszalagos egységgel dolgozunk. A kazettapufferben való elhelyezés miatt a gépi kódú rutinok aktivizálása után a magnó használata a rendszer megbénulásához vezethet. Ennek magyarázatát és elkerülésének módját a következő részben adom meg.

C16-on a DEF USR utasítás parancs módban csak akkor működik, ha elé vagy egy másik utasítást, vagy legalább egy kettőspontot írunk. Ennek okáról ugyancsak a következő alkalommal írok.

BARNA LÁSZLÓ

A kategóriájú kereskedelmi vállalat szervezési osztálya felvesz közép- vagy felsőfokú végzettségű

- **OPERÁTOROKAT**
- **PROGRAMOZÓKAT**
- **RENDSZERSZERVEZŐKET**

K+F témában megvalósuló operatív (online) áruforgalmi rendszer megvalósításához, új számítóközpontba, TPA 11/440 számítógépes környezetbe

Hosszú távú koncepció!

Továbbképzési lehetőségek!

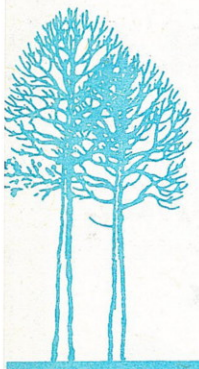
Rugalmas munkaidő!

Jelentkezés a 121-608-as telefonon, vagy részletes szakmai önéletrajzzal személyesen, Kereszty Péter osztályvezetőnél.

Cím:

TEXÉRT Kereskedelmi Vállalat

Budapest V., Alpári Gy. u. 17.



Bináris fák III.

Kiegyensúlyozott bináris fát kezelő algoritmus

Az algoritmust megvalósító program változói

Tekintsük az I. részbeli 3. ábrát (lásd 1988/2. számunkban — A szerk.). A mellékelt lista szerinti program a fa csúcspontjait ilyen elvek szerint tárolja. Ehhez be kell vezetni a megfelelő változókat és tömböket:

ms — a tömbök felső határa
a¤(ms) — az azonosítók tömbje
b%(ms) — a bal mutatók tömbje
j%(ms) — a jobb mutatók tömbje
x¤ — a keresett azonosító
j%(0) — az első üres tömbelem indexe, beépítés után a beépített elem indexe: j%(0)—1
b%(0) — a gyökér indexe

Az egyensúlyozás kezeléséhez még szükségesek:

e%(ms) — a csúcsokhoz tartozó egyensúlyi állapotok (= a jobb részfa magassága — a bal részfa magassága)
ha *h(T1)<h(T2)*, akkor *e%(i)=1*
ha *h(T1) = h(T2)*, akkor *e%(i)=0*
ha *h(T1)>h(T2)*, akkor *e%(i)=−1*
ahol *T1* a bal, *T2* a jobb részfája *t*-nek, és *h(fa)* a fa magassága
v(vm) — egy *vm* elemű verem, a keresőút előző szintjeihez tartozó csúcsok indexeinek tárolására (*vm:* a verem mérete)
vp — veremmutató

Munkaváltozók:

bj — beépítésjelző (ha *x¤* nincs a fán, akkor *bj=1* esetén le van tiltva a beépítés)
i — keresés, beszúrás, törlés, bejárás futóindexe
ix — keresésből való visszatéréskor a megtalált vagy a beépített elem indexe a tömbben
j — a beépített, ill. törölt elem indexe
h =1, ha beszúrásnál a részfa magassága növekedett vagy törlésnél a részfa magassága csökkent
i1, i2, e1, e2 — az átindexelések segédváltozói egyensúlyozásnál
vq: — ha törlésnél a törlendő csúcsnak két részfája van, akkor a keresőutat folytatni kell a bal részfa jobb szélső eleméig. Ilyenkor *vq* marad a törlendő csúcsnak megfelelő veremelemnél, *vp* mutatja a verem tetejét.

A program részei

A program áttekintéséhez az alábbi részletezést készítettük (lásd a *táblázatot*).

A program alkalmazása

A program nem használ nyomtatót, a bináris fához nem kapcsolódik lemezfájl. Ezt a kapcsolódást, továbbá képernyőmaszkhoz való kapcsolódást, amennyiben érdeklődésre tart számot, külön leírásban részletezem. A 3000—4300-ig tartó binárisfa-kezelő rutinok bármely gép BASIC-jéhez könnyen igazíthatók, de a fájl és a képernyőkapcsolat már erősen gépfüggő. Az eredeti programban if-then-else szerkezetű a 3190, 3200, 3370, 3380, 3510, 3560, 3570, 3660, 3710, 3720, 3840 és a 4320-as sorszámú utasítás.

A működés lépései az alábbiak:
— tömbméret kérése
— menü
— keresés: keresendő azonosító kérése
ha megtalálta: szerepel, index=*xx*
ha nem találja: nem szerepel
visszatér a menü
— beszúrás: beszúrandó azonosító kérése
ha megtalálta: szerepel, index=*xx*
ha nem találja: beépítve, index=*xx*
visszatér a menü

Tevékenység	Programsorok
1. Demonstrációs környezet	3000 alatt
2. Keresés, beszúrás	3000-3430
2.1 x¤ nincs a fán, bj=0 esetén beépítés	3010-3060
2.2 x¤ kisebb, mint a vizsgált csúcs azonosítója	3080-3230
2.2.1 verembe, rekurzív hívás, veremből	3080-3100
2.2.2 bal oldali ág magassága nőtt	3120-3230
2.2.2.1 nem kell egyensúlyozni	3120-3130
2.2.2.2 BB-forgatás	3140-3160
2.2.2.3 BJ-forgatás	3180-3210
2.3 x¤ nagyobb, mint a vizsgált csúcs azonosítója	3250-3410
2.3.1 verembe, rekurzív hívás, veremből	3250-3270
2.3.2 jobb oldali ág magassága nőtt	3290-3410
2.3.2.1 nem kell egyensúlyozni	3290-3300
2.3.2.2 JJ-forgatás	3310-3330
2.3.2.3 JB-forgatás	3350-3390
2.4 x¤ egyenlő a vizsgált csúcs azonosítójával	3420-3430
3. Törlés	3440-4190
	4310-4360
3.1. törlés meghívás	3891
3.2. üres elem képzése, felfűzés a láncra	3893
3.3. x¤ nincs a fán	3900
3.4. keresés a kisebb ágon (vermelés rekurzív hívás egyensúlyvizsgálat)	3920-3960
3.5. keresés a nagyobb ágon - " -	3980-4030
3.6. kiiktatás, ha mindkét részfa üres	4045
3.7. kiiktatás, ha a jobb részfa üres	4040-4060
3.8. kiiktatás, ha a bal részfa üres	4070-4080
3.9. kiiktatás, ha egyik részfa sem üres	4090-4140
3.10. törlés szubrutinjai	
3.10.1. bal ág magassága csökkent	3450-3580
3.10.1.1. nem borult fel az egyensúly	3460-3470
3.10.1.2. JJ forgatás	3500-3520
3.10.1.3. JB forgatás	3540-3580
3.10.2. jobb ág magassága csökkent	3600-3740
3.10.2.1. nem borult fel az egyensúly	3610-3620
3.10.2.2. BB forgatás	3650-3670
3.10.2.3. BJ forgatás	3690-3730
3.10.3. törlés, ha a törlendő csúcs egyik részfája sem üres	3750-3880
3.10.3.1. ha jobb szélső elem, ugrás az áthelyezésre	3760
3.10.3.2. bal részfa jobb szélső elem keresés	3750-3790
3.10.3.3. szükség esetén egyensúlyozás	3800
3.10.3.4. a jobb szélső elem kiiktatása, törölt elem helyébe való beiktatása	3820-3880
3.10.4. többször ismétlődő tevékenység	4150-4190
	4310-4360
4. Inorder bejárás	4195-4300
5. Demonstrációs környezet	4400-tól

```
10 rem kepernyo valtozok
20 c1$="▒▒▒▒": c2$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
30 poke 53280,0: poke 53281,0: print chr$(5)
40 print c2$: input "          tomb meret = ";ms
50 dim a$(ms),b%(ms),j%(ms),e%(ms)
60 j%(0)=1
70 rem menu
80 print c1$;"m e n u :"
90 print "▒▒          1. k e r e s e s"
100 print "▒          2. b e s z u r a s"
110 print "▒          3. t o r l e s"
120 print "▒          4. l i s t a z a s"
125 print "▒          5. b e j a r a s"
126 print "▒          6. vege"
130 print "▒▒k o d = ";
140 get kd$: if kd$="" then 140
150 kd=val(kd$): if kd<1 or kd>6 then 140
160 on kd gosub 190,230,270,400,4195
165 if kd=4 or kd=5 then gosub 4500
170 if kd=6 then end
180 goto 80
190 print c2$;: input "Keresendo azonosito:";x$
200 bj=1: gosub 3000: bj=0
210 if ix=0 then hs=1: gosub 5000: return
220 hs=2: gosub 5000: return
230 print c2$;:input "Beszurando azonosito:";x$
235 if j%(0)>ms then hs=5: gosub 5000:return
240 gosub 3000
250 if ix>0 then hs=2: gosub 5000: return
260 hs=3: gosub 5000: j=0: return
270 print c2$;:input "Torlendo azonosito:";x$
280 gosub 3890
290 if j=0 then hs=1: gosub 5000: return
300 hs=4: gosub 5000: j=0: return
400 gosub 4400
410 sk=0
420 for sl=0 to ms
430 if sk<20 then 460
440 gosub 4500
450 sk=1: gosub 4400
460 print sl;tab(6);a$(sl);tab(17);b%(sl);tab(25);j%(sl);tab(32);e%(sl)
470 sk=sk+1
480 next sl
490 return
2990 :
3000 rem k e r e s e s, b e s z u r a s
3005 i=b%(0): ix=0
3010 if i<>0 goto 3070
3015 if bj=1 then ix=0: return
3020 i=j%(0): j=j%(0)
3025 if j%(j%(0))<>0 then j%(0)=j%(j%(0)): goto 3040
3030 j%(0)=j%(0)+1
3040 a$(i)=x$: b%(i)=0: j%(i)=0: e%(i)=0
3050 h=1
3060 goto 4150
3070 if x$>=a$(i) goto 3240
3080 vp=vp+1: v(vp)=i: i=b%(i)
3090 gosub 3010
3100 i=v(vp): vp=vp-1
3110 if h<>1 or bj=1 then return
3120 if e%(i)=1 then e%(i)=0: h=0: return
3130 if e%(i)=0 then e%(i)=-1: return
3140 i1=b%(i)
3150 if e%(i1)<>-1 goto 3180
3160 b%(i)=j%(i1): j%(i1)=i: e%(i)=0: i=i1
3170 goto 3220
3180 i2=j%(i1): j%(i1)=b%(i2): b%(i2)=i1: b%(i)=j%(i2): j%(i2)=i
3190 e%(i)=0:if e%(i2)=-1 then e%(i)=1
3200 e%(i1)=0:if e%(i2)=1 then e%(i1)=-1
3210 i=i2
3220 e%(i)=0: h=0
3230 goto 4150
3240 if x$=a$(i) goto 3420
3250 vp=vp+1: v(vp)=i: i=j%(i)
3260 gosub 3010
3270 i=v(vp): vp=vp-1
3280 if h<>1 or bj=1 then return
3290 if e%(i)=-1 then e%(i)=0: h=0: return
3300 if e%(i)=0 then e%(i)=1: return
3310 i1=j%(i)
3320 if e%(i1)<>1 goto 3350
3330 j%(i)=b%(i1): b%(i1)=i: e%(i)=0: i=i1
3340 goto 3400
3350 i2=b%(i1)
3360 b%(i1)=j%(i2): j%(i2)=i1: j%(i)=b%(i2): b%(i2)=i
3370 e%(i)=0:if e%(i2)=1 then e%(i)=-1
3380 e%(i1)=0:if e%(i2)=-1 then e%(i1)=1
3390 i=i2
3400 e%(i)=0: h=0
3410 goto 4150
3420 h=0: ix=i
3430 return
3435 :
3440 rem t o r l e s   r u t i n o k
3450 rem  e g y e n 1 bal ag csokkent
3460 if e%(i)=-1 then e%(i)=0: return
3470 if e%(i)=0 then e%(i)=1: h=0: return
3480 i1=j%(i): e1=e%(i1)
3490 if e1<0 goto 3540
3500 j%(i)=b%(i1): b%(i1)=i
3510 e%(i)=0:e%(i1)=0:if e1=0 then e%(i)=1: e%(i1)=-1: h=0
3520 i=i1
3530 goto 4150
3540 i2=b%(i1): e2=e%(i2)
3550 b%(i1)=j%(i2): j%(i2)=i1: j%(i)=b%(i2): b%(i2)=i
3560 e%(i)=0:if e2=1 then e%(i)=-1
3570 e%(i1)=0:if e2=-1 then e%(i1)=1
3580 i=i2: e%(i2)=0
3590 goto 4150
3600 rem  e g y e n 2 jobb ag csokkent
3610 if e%(i)=1 then e%(i)=0: return
3620 if e%(i)=0 then e%(i)=-1: h=0: return
3630 i1=b%(i): e1=e%(i1)
3640 if e1>0 goto 3690
3650 b%(i)=j%(i1): j%(i1)=i
3660 e%(i)=0:e%(i1)=0:if e1=0 then e%(i)=-1: e%(i1)=1: h=0
3670 i=i1
3680 goto 4150
3690 i2=j%(i1): e2=e%(i2)
3700 j%(i1)=b%(i2): b%(i2)=i1: b%(i)=j%(i2): j%(i2)=i
3710 e%(i)=0:if e2=-1 then e%(i)=1
3720 e%(i1)=0:if e2=1 then e%(i1)=-1
3730 i=i2: e%(i2)=0
3740 goto 4150
3750 rem ket reszfa van
3760 if j%(i)=0 goto 3830
3770 vp=vp+1: v(vp)=i: i=j%(i)
3780 gosub 3750
3790 i=v(vp): vp=vp-1
3800 if h=1 then gosub 3600
3810 return
3830 if vq=0 then b%(0)=i: goto 3850
3840 if x$<a$(v(vq)) then b%(v(vq))=i:goto 3850
3841 j%(v(vq))=i
3850 if vp>vq+1 then j%(v(vp))=b%(i): b%(i)=b%(j)
3860 j%(i)=j%(j): e%(i)=e%(j): v(vq+1)=i
3870 h=1
3880 return
3885 :
3890 rem   t o r l e s   b e l e p e s
3891 gosub 3895
3892 if j=0 then return
3893 a$(j)="": b%(j)=0:j%(j)=j%(0):j%(0)=j: e%(j)=0
3894 return
3895 i=b%(0)
3900 if i=0 then j=0: h=0: return
3910 if x$>=a$(i) goto 3970
3920 vp=vp+1: v(vp)=i: i=b%(i)
3930 gosub 3900
3940 i=v(vp): vp=vp-1
3950 if h=1 then gosub 3450
3960 return
3970 if x$=a$(i) goto 4040
3980 vp=vp+1: v(vp)=i: i=j%(i)
3990 gosub 3900
4000 i=v(vp): vp=vp-1
4020 if h=1 then gosub 3600
4030 return
4040 j=i
4045 if b%(j)=0 and j%(j)=0 then h=1: goto 4310
4050 if j%(j)<>0 goto 4070
4060 i=b%(j): h=1: goto 4150
4070 if b%(j)<>0 goto 4090
4080 i=j%(j): h=1: goto 4150
4090 vq=vp
4100 vp=vp+1: v(vp)=i: i=b%(i)
4110 gosub 3750
4120 i=v(vp): vp=vp-1
4130 if h=1 then gosub 3450
4140 return
4145 :
4150 if vp=0 goto 4180
4160 if a$(i)<a$(v(vp)) then b%(v(vp))=i:goto 4170
4161 j%(v(vp))=i
4170 return
4180 b%(0)=i
4190 return
4191 :
4195 rem  i n o r d e r   b e j a r a s
4200 i=b%(0): sl=1: sk=1: print "▒"
4210 gosub 4230
4220 return
4230 if i=0 then return
4240 vp=vp+1:v(vp)=i:i=b%(i)
4250 gosub 4230
4255 if sk>20 then sk=1:gosub 4500: print"▒"
4260 print tab(5);sl;tab(20);a$(v(vp))
4265 sl=sl+1: sk=sk+1
4270 i=j%(v(vp))
4280 gosub 4230
4290 i=v(vp):vp=vp-1
4300 return
4305 :
4310 if vp=0 goto 4350
4320 if a$(j)<a$(v(vp)) then b%(v(vp)): goto 4340
4330 j%(v(vp))=0
4340 return
4350 b%(0)=0
4360 return
4365 :
4400 print "▒▒▒▒▒▒▒▒L I S T A Z A S"
4410 print "ssz azonosito    bal    jo  egyensuly"
4420 print "--- --------     ---    -- ----------"
4430 return
4440 :
4500 print "       nyomj meg valamit!"
4510 get tv$:if tv$="" then 4510
4520 return
4530 :
5000 rem uzenetek
5010 print "▒▒▒";
5020 on hs goto 5030,5050,5070,5080,51
5030 print "            nem szerepel"
5040 goto 5110
5050 print "   szerepel index=";ix
5060 goto 5110
5070 print " beepitve,index=";j
5075 goto 5110
5080 print " torolve,index=";j
5090 goto 5110
5100 print " tomb betelt!"
5110 for ia=1 to 500: next ia: return
```

- törlés: törlendő azonosító kérése
  ha megtalálta: törölve, index = *xx*
  ha nem találja: nem szerepel
  visszatér a menü
- listázás: a felvitel sorrendjében felíródik a képernyőre a csúcshoz tartozó sorszám (tömbindex), azonosító, bal mutató, jobb mutató, az I. részbeli 3. ábrához hasonlóan
- bejárás: rendezetten íródnak fel a képernyőre a fa azonosítói

A listázásnál és a bejárásnál minden 20 elem után, illetve a végén egy billentyű megnyomását követően folytatódik a lista, illetve tér vissza a menü.

Megjegyzések: A fa egy csúcsának törlése után az üresen maradt helyeket az algoritmus láncba szervezi, majd újra bevonja a feltöltésbe. J%(0) az utoljára törölt elem indexét tartalmazza, annak jobb mutatója az utolsó előtti törlés indexét mutatja stb. A lánc végén lévő elem jobb mutatója az első olyan tömbelemre mutat, amely még nem volt feltöltve. Ily módon a köztes üres elemek újrafeltöltésége nem nyúl a program új elemhez. Ebből következően is igaz, hogy maximum annyi helyre van szükség, ahány különböző elem szóba jöhet a fán, akárhány műveletet is végzünk. Ily módon valósul meg a dinamikus helyfoglalás.

A program jó néhány rekurzív hívást tartalmaz. Itt a visszatérési címeket a BASIC által kezelt verem tartalmazza. Az algoritmus 10 szintű fánál 9 visszatérési cím tárolását igényli, az egymásba ágyazott GOSUB, illetve FOR utasítások száma legfeljebb 3. Ez még a C64 gép esetében is (verem hossza: 256) lehetővé teszi a szokásos maximum 4-5 egymásba ágyazású programhoz való kapcsolást. A bejárásoknál a csúcsok megőrzésére a program készít vermet, mivel a dimenzionálás automatikus, esetünkben legfeljebb 10 szintű fa hozható létre, ez maximum 1023 elem. A gép 2.0 BASIC tömbkezelési sebességet ismerve maximum 1000 elem kezelése javasolt C64 gépen.

FLENDER ANDRÁS

Személyi

A Macintosh óta sok-sok személyi és házi számítógép mondhatná ezt magáról, de azon sem csodálkoznánk, ha piktogram-vezérlésűnek minősítve magukat, dicsekednének. Szakmai körökben igen divatos szó.

Új viszonyulás

Egy házi számítógép ma már nem is számít valamirevalónak, ha rendszerét csak pontosan a gép helyesírásának (szintaxisának) megfelelően begépelt utasításokkal lehet vezérelni. Mi több, nem is lehet személyi számítógépnek igazi piaci sikere anélkül, hogy ne lenne úgynevezett *front-end*je, amilyen például a Microsoftnak a GEM, az Atarinak a TOS, az Apple-nek a MAC-(DOS). A legfejlettebb Commodore 128-asnak már a ROM-jába van beépítve a GEOS program, ami több is, mint egyszerű ikonvezérlő. A drágább gépektől pedig egyenesen mindenki elvárja, hogy rendelkezzenek ikonutasításos rendszerrel, mert ez leegyszerűsíti az utasítások bevitelét, és gyakran olyasmikre is lehetőséget nyújt, amik egyébként BASIC-beli utasításokkal nem fogalmazhatók meg. Az egyik legjobban átgondolt példa erre a Macintoshra írt GEM, amely szinte szabvánnyá vált a képernyőkezelő ikonutasításos rendszerek között, és teljes egészében ikonok és mozgatható (de főképp lehúzható) ablakok, menük kombinációjából áll. Az ikonutasításos rendszer, ha nem is nevezhető generációváltásnak, de mindenképpen szemléletváltás: új felületet adott a felhasználóknak az ember-gép kapcsolat értelmezéséhez és ezáltal az átértékeléséhez is.

Példaválasztás

Hogy egy ilyen ikonutasításos rendszert közelebbről szemügyre vehessünk, konkrét géphez és szoftverhez kell fordulnunk. A választás a közel sem legelterjedtebb gépre, a Sinclair QL-jére esett — talán azért, mert a gépre írt egyik ikonvezérlő szoftver már az 1.1-es változatában olyan hibátlannak ígérkezett, hogy azt EPROM-os alakban is forgalmazták; főképp mégis azért, mert megjelenésével rögtön nagyot emelt a gép emberközelségén, és ezzel sokat javított akkori rossz piaci helyzetén.

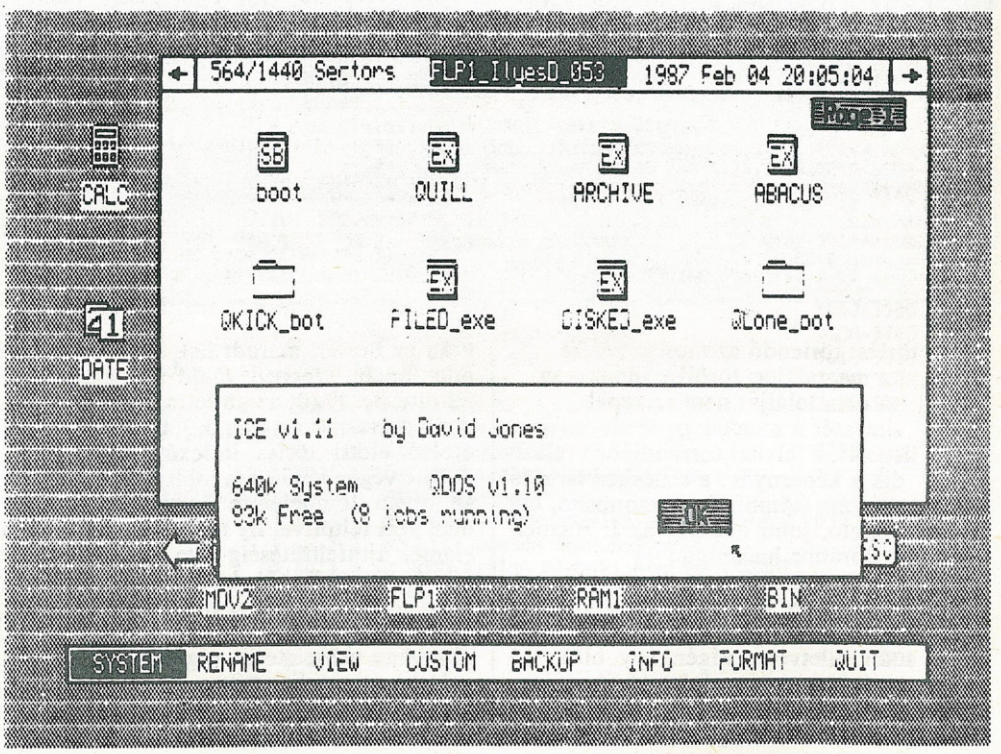
Akik ismerik a QL finoman kidolgozott, néha esetlen, gyakran túl cizellált utasítási szintaxisait — legyünk őszinték: a Tony Tebby által megírt Toolkit II ezeken kicsit egyszerűsített —, azoknak üdvözölniük kell minden könnyítést. Nem véletlen, hogy kedvezően fogadták az olyan törekvést, ami végeredményként a bonyolult utasítási rendszert megkerülve, lehetővé teszi a műveletek kényesebb elvégzését is, egy vagy két billentyű lenyomásával (kurzor esetén nyomva tartásával).

QL-re több mint fél tucat okosabb és kevésbé okos ilyen rendszer kapható már. Ezek képességeikben sokszor átfedik, máskor jól kiegészítik egymást, és így a felhasználónak kedve lenne egyiket-másikat összegyúrni, ami persze már nem is olyan egyszerű dolog.

Evolúció

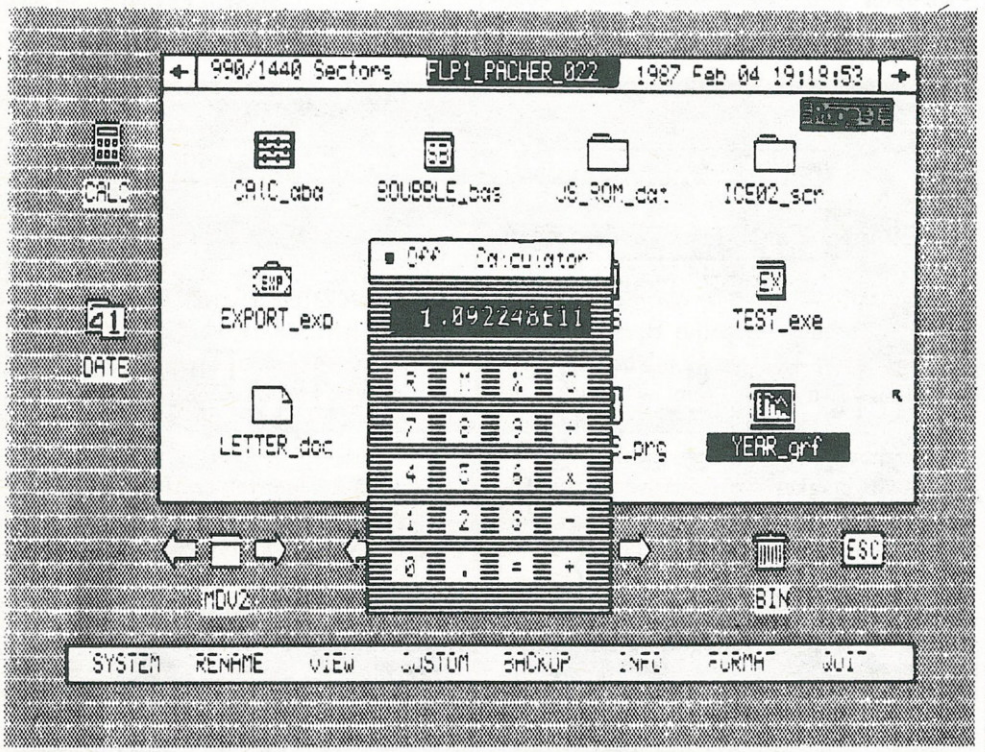
A QL-re megjelent első ilyen rendszer, amit később részletesen is tárgyalunk, az *Eidersoft ICE* (ejtsd: ájsz). Szerzője, David Jones, méltán büszke lehet művére. Az ICE betűszó, és az *I*con *C*ontrolled *E*nvironment szavak kezdőbetűiből lett. A név kb. annyit tesz, hogy ikonnal vezérelt környezet. Miután ez kevesebb mint 16 kbájt memóriaterületet igényel, ezért nemcsak szoftverben, hanem a QL-hez hátulról becsatlakoztatható ROM-kártyán is kapható. A QL memóriatérképén a ØCØØØ és ØFFFF közötti területet foglalja el. Ehhez a szoftverhez és hardverhez természetesen botkormányt, illetve egeret is kínálnak a kereskedők.

1. kép



számítógépek ikonutasításos rendszere

a WIMPS (*W*indows, *I*cons, *M*ice, *P*inter*S*)
Jelentése: ablakok, ikonok, egerek, mutatók.



2. kép

Ezután jelent meg a nyugatnémet ABC Elektronik gondozásában, de elsőként Angliában piacra dobott *Giga-Desk* és *E.A.S.E.* (*E*asily *A*pplicable *S*ystem *E*nvironment), ami nemcsak teljes operációs környezetet, de egy két nyomógombos egeret és egy Giga-Basic szoftvert is kínál.

Később jött az *Ultrasoft QKICK*-je, ami nem kimondottan front-end, mivel a legegyszerűbbet — egy szoftver behívását és aktivizálását — nem tudja. Ugyanakkor olyan finomságokra is képes, mint az automatikus képernyő ki-be kapcsolás (autoscreen on-off), természetesen állítható türelmi idővel, ami által a monitor kímélődik. Ezenkívül található benne határidőnapló és jegyzetfüzet mini editorral, hogy akik az IBM vagy az Apple kínálta Sidekick iránt éreznek nosztalgiát, azok ne pusztán a Qkick nevében találjanak rá utalást.

Nagy port vert fel az ugyancsak az Ultrasoft gondozásában kiadott *JAM*, amiről először mindenki azt hitte, hogy a nevével a GEM-re szeretne hasonlítani, de mint kiderült, ez a *J*ob *A*pplication *M*anager betűszava. Saját belső dinamikus RAM-lemeze különösen kényelmessé teszi a használatát.

Nagyon érdekesnek ígérkezett a Cope Software ház *QATS*-e, ami csak a felületes fantáziát emlékeztetheti a „macskák"-ra; jelentése *Q*L *A*pplications *T*raffic *S*upervisor. Érdekessé sokoldalú printervezérlő képessége és megbízható *wild card*- (Joker-) felismerő tulajdonsága tette. Rövidsége, tömörsége miatt ugyancsak kapható ROM-modul formában is.

Meg kell még említenem a QL-es berkekben jól ismert Talent szoftverház termékét, a *QIMP*-et (*Q*L *I*con *M*anagement *P*rogram).

És végül itt a *QRAM*, a legújabb front-end, amit Tony Tebby neve fémjelez. A tervek szerint a Sandy által fejlesztett SuperQ-board új bővítőjébe akarják beépíteni, és így ez — azok után, hogy 512 kbájtos memóriabővítményen kívül lemezmeghajtó-vezérlőt, párhuzamos printerillesztőt, RAM-lemezt és még egy olyan Toolkit II-t is tartalmaz, amire még Clive Sinclair is büszke lenne — valóban méltó lesz a szuper jelzőre.

Úgy tűnik, mind közül az utolsó a legjobb, de még külföldön sem a legelterjedtebb. Az eladott példányszámot tekintve első — a legtöbb QL-es által használt front-end — az *ICE*.

Mire jó az ICE?

Tulajdonképpen grafikus burokban nyújtja át a felhasználónak a bonyolult szintaxisokat, hogy ezzel megkímélje azok megtanulásától, illetve időrabló bebillentyűzésétől. A gép bekapcsolása után (feltételezve, hogy az ICE ROM-kiegészítőként van a QL erre kialakított hátsó csatlakozójába bedugva), F1 vagy F2 lenyomására a BASIC ablakok helyett azonnal a munkaterület jelenik meg a kis alakzatokkal (ikonokkal), amelyek a QL-hez csatlakoztatható eszközöket, a microdrive-okat és lemezmeghajtókat, illetve a bármikor „elővehető" naptárt, számológépet és az ugyancsak „kéznél levő" szemétkosarat jelképezik. Ha azt szeretnénk elérni, hogy az ICE helyett rögtön a háttértárolón levő BOOT nevű program induljon el autostarttal, akkor a QL reset utáni feléledése alatt az ALT gombot tartsuk nyomva. Ha ezt elmulasztjuk, akkor sem kell ismét resetelnünk, hiszen az alsó munkalécen azért van a QUIT, hogy azzal visszamehessünk a régi jó BASIC-ba.

Tudnivalók a kezelésről

Az *ICE* használatához a kezelőnek semmi mást nem kell tennie, mint a négy kurzorbillentyű (jobb esetben a botkormány vagy az egér) segítségével a képernyőn található nyilat a kívánt ikonra kormányozni. A kurzorbillentyűk, illetve a botkormány nem pótolja teljes mértékben az egeret, mert bár a nyíl sebessége a későbbiekben leírt módon beállítható, a legnagyobb sebesség mellett sem helyettesítik az egér azon képességét, hogy sebessége menet közben változtatható, és így az ikon durva megközelítésekor sokkal gyorsabban lehet a nyilat mozgatni, mint az ikonhoz érve.

Ha a kis nyíl valamelyik ikonon vagy annak közvetlen környezetében van, egyszerűen a SPACE billentyűt kell megnyomni egyszer vagy gyors egymásutánban kétszer aszerint, hogy mit szeretnénk elérni. Ha például valamelyik háttértár (microdrive, lemezmeghajtó, RAM-lemez) ikonján egy kettős leütést végzünk a SPACE billentyűvel vagy botkormány esetén a tűzgombbal, akkor ezzel a dir_eszköz+ENTER begépe-

lését takarítjuk meg. Könnyen beláthatjuk, hogy ez például egy dir mdv1_+[ENTER] esetén is 1C billentyű helyett csupán a SPACE kétszeri leütése.

Ikonkészlet

Mindegyik háttértár ikonjának jobb és bal oldalán nyíl van, amelyre SPACE-t ütve 1-től 8-ig növelhetjük bármelyik háttértár sorszámát, mivel a QL egyidejűleg maximum nyolcat tud bármelyik háttértárból kezelni. A CUSTOM (lásd lejjebb) segítségével az is megoldható — ha nem használunk lemezmeghajtót vagy RAM-lemezt —, hogy ezek helyét kölcsönvegyük a microdrive számára: átkereszteljük az FLP-t vagy a RAM-ot MDV-re, és a számát SPACE leütéssel 2-re változtatjuk. Csak microdrive-ot használva így nem kell lépten-nyomon átszámozni az ikont, elég, ha az ikonok között válogatunk.

A dir-t behívó kettős SPACE leütése után a munkaterületen, annak egyetlen lapján egyszerre 12 fájl jelenik meg más-más szimbólumú ikonokkal, aszerint, hogy az Super Basic fájl, Exec fájl vagy a négy felhasználói Psion program fájlja. Ezeket nemcsak az ikon alatti fájlnév kiegészítője különbözteti meg:

QUILL esetén _doc
ABACUS esetén _aba
ARCHIVE esetén _prg, _pro, _dbf
EASEL esetén _grf

hanem maga az ikon is. Külön ikonja van még a Psionok közötti exportálható-importálható _exp kiegészítésű fájloknak, az ICE által fel nem ismert típusú fájloknak és végül a csak ICE alatt futtatható ChoICE-ban definiált _tsk kiegészítésű programcsomagnak (lásd lejjebb).

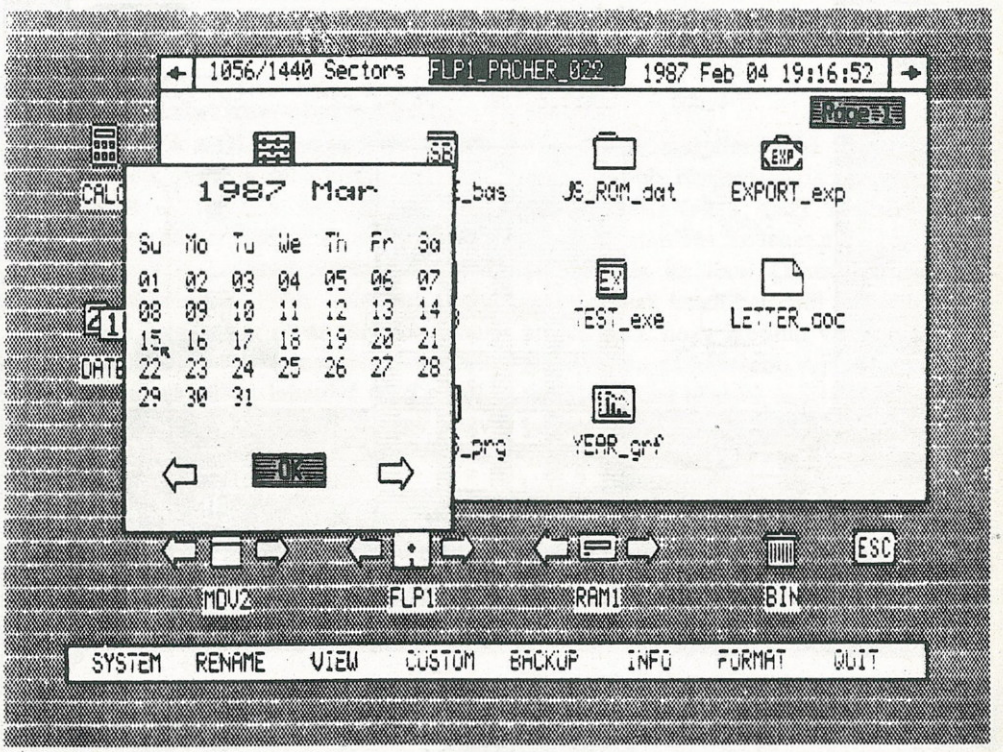
Vigyázat! Az ikon alatti fájlnév csonkán jelenik meg, ha annak utolsó kiegészítő négy karakterét nem számolva a fájl neve hosszabb, mint 8 karakter. A csonkított fájlnév az INFO segítségével teljes hosszában előhívható. A munkaterület jobb és bal felső sarkába vive a kurzort, egy SPACE leütéssel lehet egyet előre vagy hátra lapozni, de ha sokszor 12 fájlt tartalmaz az adathordozó, és a kezelő tudja, hogy a kívánt lap messze van, akkor a SPACE kettős leütésével négy lapot is lapozhat egyszerre.

A munkalécek funkciói

A munkaterület felső munkalécén még olyan fontos információk olvashatók, mint az adathordozó fajtája, neve, az összes és a még szabad szektorok (512 bájt) száma, valamint a gép tudomása szerinti dátum és pontos idő; ez utóbbi külön jobként fut.

A SPACE billentyű kettős leütésével a program — függetlenül attól, hogy BASIC program vagy EXEC-ként indítható job — arról az eszközről hívódik és indul el automatikusan, amelyik eszköz a tartalomjegyzéket a képernyő munkaterületére adta. Sajnos az ICE egyik nagy hibája, hogy ennél kevesebb nem létezik. Az ICE kettős szóközleütése helyettesíti ugyan az egész lrun mdv1_név_bas vagy exec_w flp1_név_exe szintaxist, de nem tudja megoldani az editáláshoz elengedhetetlenül szükséges egyszerű load mdv1_név parancs végrehajtását. (Igaz, hogy a gyakorlott programozón ez nem fog ki, mert ha gyakran szeretné szerkeszteni a programját, de az ICE-szal dolgozik, akkor beír egy 1 PAUSE utasítást, és ha editálni akar, akkor a beolvasás után BREAK-et nyom, ha pedig nem, akkor SPACE-t; ezzel a problémát feloldotta.)

tum — SET TIME/DATE — beállítása, aminek hibája, hogy nem hagyható el, csak akkor, ha végigjátszottuk az egész beállítást; a géphez csatlakoztatott eszközök meghívási neveinek — CHANGE DEVICE — megváltoztatása (például flp1 helyett fdk1), a printer csatlakoztatási módjának — DEFINE PRINTER — (par, ser), valamint az adatátvitel sebességének (Baud rate) beállítása, a SET CONTROL segítségével a kurzornyíl sebességének és a billentyűismétlődés késedelmi idejének a beállítása. Mindezen beállítások (kivéve a kurzor mozgatási sebességét) ugyancsak az ablakban kínálkozó — SAVE CHOICES — segítségével menthetők a háttértár adathordozójára, hogy a kikapcsolás utáni bármelyik újbóli üzembe helyezéskor a CUS-



3. kép

Az alsó munkalécen látható SYSTEM-re egy SPACE-t ütve a munkaterület közepén megjelenő ablakban az ICE közli saját verziószámát, a QDOS verziószámát, a gép összes és még szabadon felhasználható RAM területét kbájtban, valamint azt, hogy a gépben jelenleg hány job fut egyszerre. Ezt az ablakot az összes információjával az ablakban található O.K. feliraton való egy SPACE leütéssel lehet a munkaterületről eltünteti (lásd az *1. képet*).

Ugyancsak az alsó munkalécen található a CUSTOM, ami egy SPACE leütéssel szintén ablakot hív a munkaterületre, hogy újabb, a kurzorbillentyű fel-le gombjaival kiválasztható lehetőségeket kínáljon a felhasználónak. Ilyen lehetőség az idő és dátum alatt található — LOAD CHOICES — útján visszahívhatók legyenek a gépbe, sőt az ICE első aktivizálódásakor automatikusan meg is hívja a kiemelt háttértárról a del_dfb néven tárolt CHOICE-ot, így azt első alkalommal nem szükséges kézzel aktivizálnunk. A CUSTOM ablakból a kurzor CANCEL-ra mozgatásával és egy SPACE leütéssel szállhatunk ki.

Körültekintően dolgozták ki a munkalécen szereplő FORMAT lehetőséget is. Egy külön ablak által nemcsak arra szólítanak fel, hogy jelöljük ki az eszközt, amelyen formattálni szeretnénk, de még az adathordozó keresztelője előtt külön figyelmeztet a rendszer, hogyan kell kiszállnunk az egészből, „amíg nem késő".

Műveletek az állományokkal

A BACKUP megadja, hogy az adathordozón, amiről a tartalomjegyzéket kértük, összesen hány fájl van, és azt hajlandó is válogatás nélkül akármelyik általunk kijelölt eszközben levő adathordozóra átmásolni. Ha netán ugyanarra az eszközre szeretnénk a másolatot készíteni, mint amelyiken a forrás van, akkor ezt is meg tudja oldani, de időnként kéri a forrásszerepű és a másik adathordozó cseréjét a közös eszközben. Ez nyilván attól függ, hogy pufferterületnek mekkora memóriaterületet tarthat fenn a gép a másoláskor. Hiányossága, hogy írásvédett cartridge-ot nem ismer fel, és bár nem ír rá, de nem is közli, hogy a másolás ilyen akadályokba ütközik.

A kijelölt fájlok mindvégig jól elkülönülnek, mivel az ICE munkaterületén e fájlok ikonjai inverzben látszanak mindaddig, amíg az ESC-re lépve egy SPACE-t ütve a kijelölést nem töröljük vagy a kijelölés célját (másolás, info stb.) el nem végezzük.

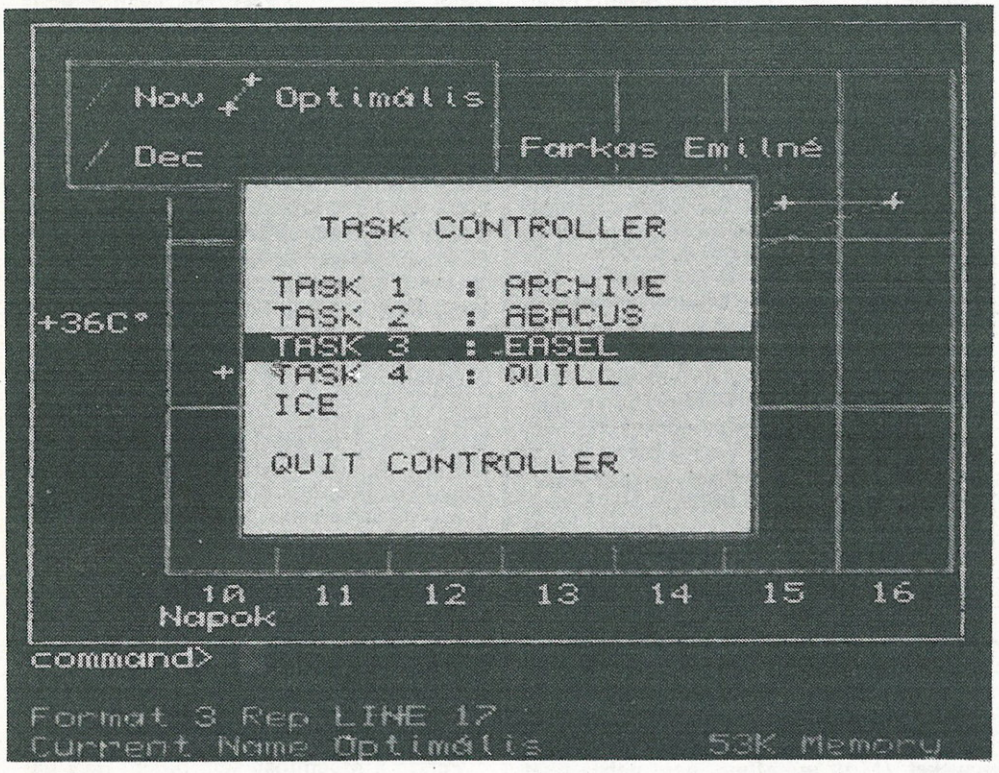
A munkalécen található még a RENAME, VIEW és az INFO. Ezek mindegyike ugyanúgy, mint a fájlonkénti másolás, előzőleg elvárja, hogy jelöljük ki azt a fájlt, amelyikre vonatkoztatjuk a műveletet. A RENAME a kijelölt fájlt

Régi név: név

Új név: név

párosításban kiteszi egy ablakba, és még egy kurzort is ad hozzá, hogy azonnal el tudjuk végezni a fájl kívánt átkeresztelését.

A VIEW is igényli az előre kijelölt fájlt, majd választási lehetőséget ad rá, hogy azt hogy ezt egy megjelenő ablakban az O.K.-ra SPACE-szel válaszolva még egyszer jóváhagyjuk.



4. kép

Egyes általunk kiválasztott fájl vagy fájlok másolása ennél sokkal egyszerűbben is megy. Miután egy SPACE leütéssel kiválasztottuk a másolandó fájlokat, csak rá kell vezetnünk a kis nyilat a háttértár ikonjára, és ott egy újabb SPACE leütés után az ICE közli, hogy hány fájlt jelöltünk ki, aminek másolását egy O.K.-ra helyeslő SPACE leütés után hajlandó elkezdeni, mégpedig a *kiválasztás sorrendjében!* Mindkét másolási módszernél menet közben jelzést kapunk, ha a kérdéses fájllal azonos nevű fájl már szerepel a cél háttértárolón. Ilyenkor — RETRY-ra „bólintva" egy SPACE leütéssel — módunk van felülírást elrendelni vagy CANCEL-re azt kikerülni (lásd később).

a képernyőn TO SCREEN vagy a sornyomtatón TO PRINTER tekintsük meg.

Az INFO az előre kijelölt fájlról (több fájl esetén az utoljára kijelöltről) elmondja a fájl nevét, típusát, hosszát és azt, hogy a QL órája szerint másodperc pontosan mikor került az adathordozóra.

Majd mindegyik ablaknak van egy CANCEL opciója arra az esetre, ha menet közben meggondolnánk magunkat és mégsem akarnánk elvégeztetni a géppel az előre kijelölt műveletet.

A BIN elnevezésű (Dustbin = szemétgyűjtő) ikonra lépve és ott egy SPACE-t leütve tartalomjegyzéket adó adathordozóról — véletlen törlés lehetőségét kizárva — kitörölhetjük az összes előre kijelölt fájlt úgy,

Üveggyöngyök

Az ICE szolgáltatásai közé tartozik a képernyő bal felső sarkából bármikor középre hívható zsebszámológép (lásd a *2. képen*), de ez olyan kezdetleges, hogy egyetlen kis változtatás is elég lett volna a megírásakor ahhoz, hogy használója ne mérgelődjön: nem kellett volna megírni. Mégiscsak túlzás, hogy egy számológéppel versenyre kelhet egy fejben számoló csak azért, mert a számjegyek beírását a szoftver szerzője nem a számjegy bebillentyűzésével, hanem „kurzortúltengésben szenvedve", a kis nyílnak a számjegyre mozgatásával és egy SPACE leütéssel oldotta meg.

Az ICE másik nem túl erős, de nem is ennyire hibás szolgáltatása a naptár (lásd a *3. képen*), amelyik azonban csak havonta lapozható, és semmilyen határidőnapló funkciója nincs, mint például az IBM PC-re írt Sidekickben.

Ígéretek

Milyen egyéb programok bővítik az ICE képességeit? Időközben született néhány, az ICE-ra — és természetesen a sikerére is — támaszkodó, azt jól kiegészítő program. Ezek ismertetése legyen egy később megjelenő, újabb cikk feladata. Most csak néhány szóban róluk. Itt van például az ICICLE, amely lehetővé teszi nemcsak a négy Psion felhasználói programnak, de Superbasic programoknak is a felhasználó által definiált ikonokkal való vezérlését.

Az ARTICE kurzorgombokkal is, de főleg egérrel vezérelhető grafikai program.

A ChoICE a felhasználó által kiválasztott és programcsomagba fűzött programok (max. 4+ICE) kvázi multitaskban való futtatását teszi lehetővé (lásd a *4. képen*).

Az ICE Toolkit (pop up, windows, menüléc, alert, boxes) olyan felhasználói segédletek gyűjteménye, amelyek nemcsak a Tony Tebby-féle Toolkitből, de az ICE-ból is kimaradtak, és jól összeférnek az ICE-szal.

Mindezt valószínűleg egyetlen szoftverbe is bele lehetett volna sűríteni, de sajnos a számítástechnikai szoftverek piacán gyakran a legerősebb hatású az az elv, hogy „amit ma megírhatsz, ne halaszd holnapra, de csak holnapután add el!". Így a felhasználó kétszer (n-szer) fizet . . .

(Az ICE iránt érdeklődők kielégíthetik kíváncsiságukat bármelyik szerdán du. 5 és 9 óra között a Csokonai Művelődési Ház QLub-jában (Bp. XV., Eötvös u. 64—66).

PACHER GÁBOR

Egér – menü – ablak

Néhány évvel ezelőtt először akadt a kezem ügyébe Apple Macintosh számítógép. Az egérrel, menüvel, ablakkal való munka olyan szenzációs élmény volt nekem, öreg programozónak, hogy nyomban cikket írtam róla a Mikromagazinnak. Meglepetésemre ugyanabban a számban még két másik szerzőnek a Macintosh-cikke is megjelent. Nekik is olyan élmény volt, hogy úgy érezték: meg kell osztaniuk a közönséggel.

A visszhang kétféle volt. Akik csak a mi cikkeinkből értesültek az újdonságról, általában azt tartották, hogy túllelkendeztük. Új gépek, végül is, állandóan jelennek meg: miért éppen erről kellett ennyit írni? Ellenben akik maguk is kipróbálták már a Macintosht, azok azt kifogásolták, hogy túl hűvösen írunk róla: nem beszélünk „új korszakról", nem emlegetjük a „programozás forradalmát".

Azóta világosabb lett a helyzet, és most már tárgyilagosan tudjuk értékelni a jelenséget. Az elmúlt évben az USA-ban a Macintosh volt a legkelendőbb személyi számítógép (az eladott gépeknek kb. 10 százaléka). Az IBM PC, XT és AT együttesen 9 százalékot tett ki, a kompatibilis utánzatokkal együtt azonban kb. 70 százalékot. Az Apple cég néhány évig veszteséges volt, és már azt latolgatta, hogy megállítja a Macintosh gyártását. Ehelyett annyira megnövekedett a kereslet, hogy Appleék új gyárat nyitottak az Írországhoz tartozó Cork szigetén. Ennek feladata az európai piac ellátása. Nehezebb dolguk lesz, mint a cupertinoi részlegnek Kaliforniában vagy a szingapúrinak Délkelet-Ázsiában. Az Apple cég ugyanis büszke arra, hogy a Macintoshban egyetlen angol szó van csak beépítve: a gép neve (egyébként ez sem angol, hanem skót). A corki üzemnek gondoskodnia kell arról, hogy minden európai ország a maga nyelvén értő gépet kapjon. Ezt megkönnyíti, hogy a szabályosan írt Macintosh programokban az emberi nyelvű szavak nincsenek szanaszét szórva, hanem egyetlen táblába gyűjtve. Így csak ezt kell egy darabban kicserélni. (Az Országos Tervhivatalban ezt megcsinálták, gépeikkel magyarul lehet szót érteni.)

Az IBM személyi számítógépeit többnyire MS—DOS operációs rendszerrel működtetik. („MS" a programtervező cégnek, a Microsoftnak a monogramja, „DOS" pedig lemezes operációs rendszert jelöl.)

Az MS—DOS sokban hasonlít a CP/M operációs rendszerhez, csakhogy a CP/M nyolcbites processzorhoz, rendszerint Z80-hoz készült, utódja ellenben 16 bites processzort, többnyire 8086-ot vagy 8088-at használ.

Mind a CP/M-ben, mind az MS—DOS-ban meg kell tanulni egy csomó értelmetlen betűhalmazt. Angol szavak rövidítései ezek, de sok angol kollégával beszélgettem, aki tudta ugyan, hogy mit csinál a gépe, ha ezt írja be: IPL—II, de arról fogalma sem volt, hogy ez miért jelenti azt, hogy „file transfer package" (állománykezelő programrendszer).

Az utasítás rövidítése után rendszerint egy vagy több állomány neve vagy egyéb paraméter következik. Ezeket magunk találjuk ki, csakhogy mégis van a dolognak bökkenője: sokszor fogjuk a neveket beírni. Ha a név tömör rövidítés, hamar elfelejtjük, hogy mit jelent. Ha viszont hosszú, akkor unalmas sokszor bebillentyűzni, és könnyen el is hibázhatjuk.

A CP/M és az MS—DOS még abból az időből származnak, amikor a számítógép terminálja papírra író szerkentyű volt, tulajdonképpen olyan írógép, amellyel a gép és a használója felváltva írt ugyanarra a papírra. A mai terminál többnyire képernyő, amelyen a kurzort nemcsak soronként egymás után, hanem tetszőlegesen lehet mozgatni. A mozgatásra sokféle eszköz van, botkormány, fényceruza stb., de az utóbbi években legjobban az egér terjedt el. Ez egy kis doboz, amelyet az asztalon úgy guríthatunk ide-oda, mint a játékegeret. Hosszú farka csatlakozik a számítógéphez, és ahogy az egér mozog az asztalon, úgy mozog a kurzor a képernyőn.

A képernyőn látjuk az utasításokat. Nem torz rövidítéseket, hanem teljes hosszúságú neveket. (Mint mondtam, nem nehéz őket egyszer s mindenkorra magyar nevekre kicserélni.) A kurzort ráállítjuk a kívántra, és megnyomjuk az egér hátán levő billentyűt. Egy billentyűnyomás elég akkor is, ha a szó 20-30 betűből áll. Csak akkor kell a nevet teljes hosszában bebillentyűzni, ha a gép még nem ismeri, vagyis éppen most találtuk ki. A továbbiakban már mindig elég rámutatni és egyet billentyűzni.

Az egészben az a legszebb, hogy *minden* program egyformán kezelhető. Az egérjátékot akárki megtanulhatja fél óra alatt, és aztán már minden programhoz használhatja. Szintaktikus hibát nem ejthet: a képernyőn csak az jelenik meg, ami szintaktikusan helyes, mást nem választhat.

A Microsoft cég érdeme, hogy az elsők között ismerte fel az új programozási mód előnyeit, és számos kitűnő programot készített gépei számára. Bizonyos, hogy ezzel sokat tett azért, hogy az Apple cég mutatója a veszteség oldalról a nyereség oldalra lendüljön át.

Hát a nagy konkurens, az IBM mit csinált? Egeret gyártott és megbízta a Microsoftot, hogy neki is készítsen olyan programokat, amelyek az Apple-ével vetélkednek. Az új IBM gépek 80286, 80386 és 80C86 processzorokkal dolgoznak. Ezek felülről kompatibilisek a korábbi 8086-tal és 8088-cal. A Microsoft ezekre a gépekre új operációs rendszert dolgozott ki, amelyet MS—OS2-nek hívnak. Ez már mindazt tudja, amit a Macintosh: egér, ablak, menü stb. A régi, MS—DOS-ra írt felhasználói programok azonban az új operációs rendszerrel nem futtathatók. Az átmenet megkönnyítésére tehát írtak Microsofték egy Windows (=ablakok) nevű programot. A Windows az MS—DOS alatt fut, és lehetővé teszi, hogy MS—DOS-ra írt felhasználói programokat egeres-ablakos-menüs módszerrel lehessen kezelni. Ez azt a nehézséget is áthidalja, hogy még sok 8086 vagy 8088 processzoros gép van forgalomban, és ezeken az MS—OS2 operációs rendszer nem futtatható.

Megjegyzendő még, hogy az egéren kívül szükség van a Hercules vagy az EGA grafikus kártyára is a Windows vagy az MS—OS2 futtatásához. A Macintoshban az ennek megfelelő áramkörök be vannak építve, nem kell külön megvenni.

Persze a Macintosh sem pihent meg babérain. Legújabb változataiban a főtár 4 M-ig (!) kiterjeszthető. Processzora a 68000 után a 68020, majd a 68030 lett. Ezeknek az utasításkészlete felülről kompatibilis a korábbiéval, tehát annak programjai az új gépen is futnak. Ennek ütemideje 20 MHz. Számos trükk gyorsítja még tovább a működését. A 68020 256 bájtot tárol a processzor gyors regisztereiben, ha tehát ezekhez kell hozzáférni, akkor nem szükséges a tárból olvasni az utasítást. A 68030 még tovább megy: a 256 utasításbájton kívül 256 adatbájtot is tárol, tehát még ritkábban kell a tárhoz fordulnia.

Említést érdemel még, hogy a két nagy versenytárson, az IBM-en és az Apple-en kívül az Atari és a Commodore Amiga is egerezik és ablakozik. Mivel ez a programozási módszer sokkal kényelmesebb és annyival könnyebben megtanulható a korábbiaknál, hogy nem is sportszerű az összehasonlításuk, a következő években általános elterjedésére számíthatunk.

MÜNNICH ANTAL

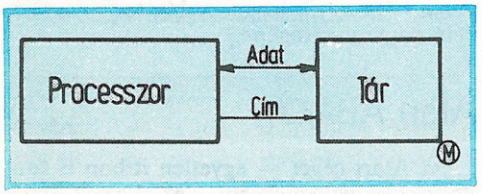
Most induló sorozatunkban azokat az új hardver- és szoftvertermékeket ismertetjük, amelyek várhatóan általánosan elterjednek és meghatározó szerepük lesz a fejlődés irányainak kialakításában.

Új típusú processzorok

Párhuzamos feldolgozás

Az egyik korai számítógép, az ENIAC elkészülte óta a gépek hasonlítanak abban, hogy különálló központi egységük és táruk van. Egy utasítás végrehajtásához a megfelelő adatot ki kell keresni a tárból, be kell olvasni a központi egységbe, ott el kell végezni a szükséges műveletet, majd az eredményt vissza kell juttatni a tárba *(1. ábra)*.

Ezt az elrendezést sorosnak hívják, mert a feldolgozás egyes lépései egymást követik. A soros kialakításnak műszaki okai vannak. A félvezető tárak előtti korszakban a központi egységek és a tárak más-más anyagból készültek. A tár gyártási költsége



1. ábra

volt az alacsonyabb, így a központi egység minél jobb kihasználására törekedtek. Ezt a soros elrendezés valósította meg. Jelenleg azonban már mindkét egység szilíciumlapkán található. A felhasznált felület döntő többségét a tár foglalja el. Az időkihasználtság is erősen eltérő. A processzor szinte „folyton" dolgozik, míg a tár jó része szinte sohasem. A számítógép árát elsősorban a tár ára szabja meg, így az optimális megoldás jelentős árcsökkentő tényező lenne.

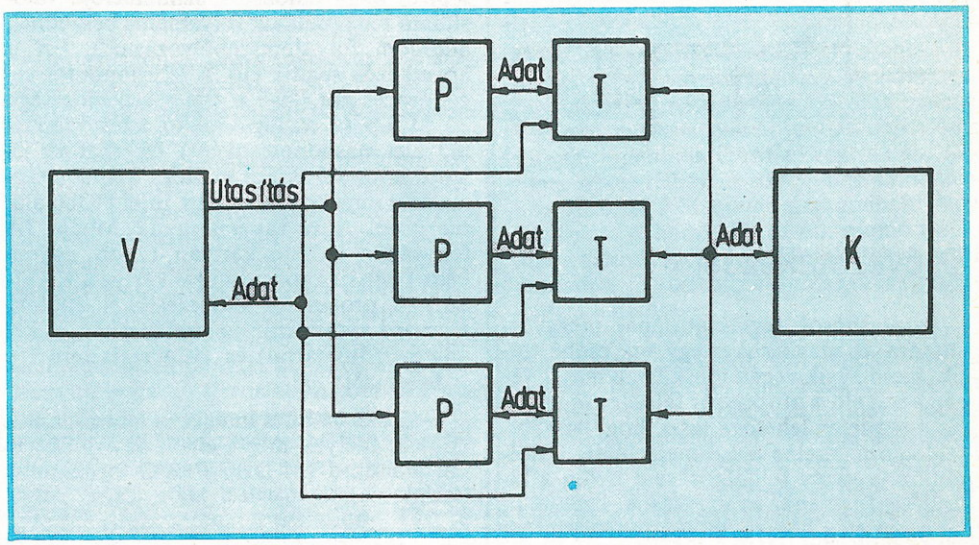
A megoldás egyik módja sok kisméretű, egyidejűleg dolgozó processzor működtetése, amelyek mindegyike saját kis tárral rendelkezik *(2. ábra)*. (Az ábrán V = vezérlő, P = processzor, T = helyi tár, K = kommunikációs egység.)

Miért most vált ez fontossá? A probléma természetesen nem ma jelentkezett először, de a mesterséges intelligencia (artifical intelligence, AI) rendszerekben a feldolgozandó fogalmak számánál sokkal gyorsabban nő a köztük levő — és megvizsgálandó — kapcsolatok száma. (Emlékeztetjük az olvasót tavalyi sorozatunkra: Mesterséges értelem címmel öt részben közöltük. — A szerk.) A hagyományos soros számítógép ezeket egymás után dolgozza fel, ezért már aránylag kis elemszám esetén is túl lassúnak bizonyul. A párhuzamos feldolgozás (parallel processing) és az ezt megvalósító integrált áramkörök a párhuzamos processzorok (parallel processor) a „nagy ugrás" egyik eszközét jelentik.

Csökkentett utasításkészlet

A másik, talán hasonló fontosságú változás a csökkentett utasításkészletű számítógép (reduced instruction set computer, RISC) megjelenése. A hagyományos számítógépek ún. komplex utasításkészletű számítógépek (complex instruction set computer, CISC). Ezeknél a tervezők arra törekedtek, hogy az utasítások számának, teljesítőképességének növelésével fokozzák az általában egyprocesszorú számítógépek teljesítményét. (Az utasítások száma többszáz.) A RISC-gépeknél az irányzat teljesen más: gyorsan végrehajtható, kisszámú utasításból álló készlettel felépített processzor, amiből esetleg több is van egy számítógépben. Az utasítások száma itt csak több tíz.

A CISC-gépekben egy utasítás ún. mikroutasítások sorozata, amelyeket a processzor egymás után, tehát sorosan dolgoz fel.



2. ábra

A feldolgozáshoz először is veszi a következő utasítást, tárolja, felbontja mikroutasításokra, azokat is tárolja, majd egymás után végrehajtja őket, tárolva közben a használt adatokat, eredményeket. Az adatokat esetleg a processzoron kívül levő tárból veszi, és az eredményeket oda teszi. Így a processzor viszonylag bonyolult felépítésű, bonyolult kapcsolatrendszerű egység, amelyen belül egy utasítás feldolgozása több óraciklust igényel.

A RISC rendszerű processzorok alapelgondolása egyidős a mikrogépek megjelenésével. 1975-ben az IBM egyik kutatócsoportja egy munkája során gyorsabb processzort igényelt, és a gyorsítás céljából elhagyta a ritkán előforduló utasításokat. Így az utasítások értelmezése gyorsabban ment, hiszen kevesebb lett a lehetséges esetek száma, és ezzel gyorsabb lett a processzor is. Az igazi RISC-gépek azonban csak a 80-as évtized közepén jelentek meg.

Melyek a jellemzői egy RISC-gépnek?

1. Minden utasítás végrehajtódik egy óraciklus alatt. Így az utasítások nagyon egyszerűek, és/vagy a processzornak van kiegészítő hardverje, amely segít az összetettebb utasítások végrehajtásában. Hiányoznak tehát az esetenként akár tízciklusnyi végrehajtási idejű utasítások.

2. Töltsd be/tárold el forma. Csak a betöltés és eltárolás utasítás esetén dolgozik a processzor külső adatokkal. Hiányoznak tehát a töltsd be és csinálj valamit — például adj hozzá, hasonlítsd össze — típusú utasítások.

3. Bedrótozott vezérlés. Nincs különbség az utasítások végrehajtási idejében, ezért nincsenek bonyolult, összetett utasítások. Az összetett utasítások időben egymás utáni végrehajtása helyett egyes utasításokat hardverben megvalósított és párhuzamosan végrehajtott utasításként dolgoz fel.

4. Viszonylag kevés utasítás és címzési mód. Emiatt a tervezés egyszerű.

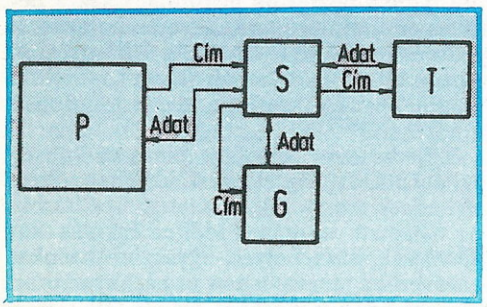
5. Rögzített utasításforma.

6. Hosszú programkészítési idő. A csekély számú és egyszerű utasítás miatt a magas szintű nyelvek fordítóprogramja sok — a CISC-rendszereknél jóval több — utasításból állnak, és megírásuk is nehezebb, mint a CISC-gépeknél.

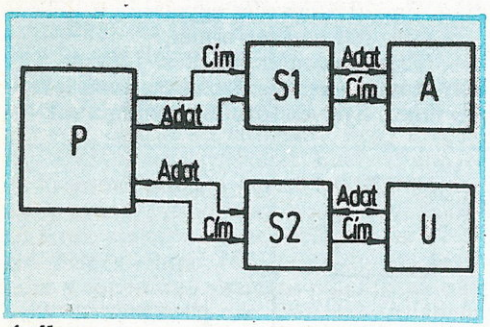
A RISC-processzorokat különböző kiegészítőkkel látják el. Így például nagyszámú — gyakran több mint 100 — regiszter, léptetők, lebegőpontos hardver, szorzók, folyamatok közötti kommunikációs hardver tartozik a processzorhoz. A RISC-gépek programozása azért sokkal munkaigényesebb és bonyolultabb, mint a CISC-gépeké, mert az előbbiek utasításainak egész sorozata felel meg az utóbbiak egy-egy utasításának.

A Neumann-féle szűk keresztmetszet feloldása

A mikroprocesszor adatfeldolgozási sebessége nagyobb, mint az adatátvitel sebessége a processzor és a tár között. Ezt nevezik Neumann-féle szűk keresztmetszetnek. A probléma hagyományos megoldásmódja: gyorstár hozzáépítése a processzorhoz (*3. ábra.* Az új jelölés: S=társzervező, G=gyorstár). Az újabb változat az ún. Harvard-architektúra, melynél külön-külön gyors utasítás- és adattár van (*4. ábra.* U=utasítástár, A=adattár). Ez gyakorlatilag kétszeres buszszélességet jelent. Így a processzornak nem kell annyit várnia az időigényes adatátvitelre, mert egyszerre hívható be az utasítás és az általa használt adatok a tárból. A programozás ugyanezért itt is nehezebb, hiszen a programozónak erre figyelnie kell.



3. ábra



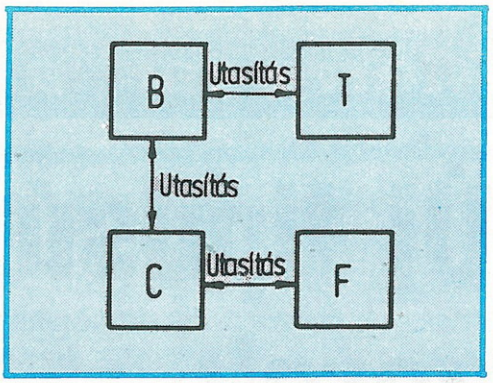
4. ábra

Csővezetékezés

Az utasítások végrehajtásának meggyorsítására az utasításokat egy ún. csőbe töltik: a cső egyik végén töltik be, a másik végén veszi elő a processzor feldolgozó része. Ez a módszer lehetővé teszi, hogy lassúbb feldolgozás esetén a cső megteléséig feltöltődjön, a gyors feldolgozásnál pedig a kiürülésig fogyjanak az utasítások. Ezenfelül, ha a cső elég hosszú, az ugróutasítások címeit is itt találhatja meg a processzor. Ilyen ugrás végrehajtási ideje igen rövid. Egy ilyen egység látható az *5. ábrán* (ahol B=betöltő, C=csővezeték, F=feldolgozó egység).

Kombinációk

Az eddigiekben ismertetett megoldások az új processzorokban először külön-külön jelentek meg, de a legújabbakban már a kombinációkban használják. Ezzel igyekeznek az előnyöket optimalizálni és a hátrányokat csökkenteni. Most áttekintjük a már iparilag megvalósított processzorokat,



5. ábra

vagyis amelyeknek a sorozatgyártását elkezdték. Ezek mindegyike több-kevesebb elem kombinációja.

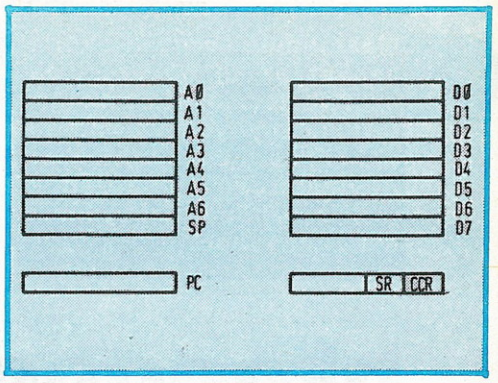
SAIC Delta Floating Point Processor™

A Science Applications International Corporation (SAIC) évi 600 millió dollár forgalmú, a korszerű technológiák kutatásával, bevezetésével foglalkozó cég. Szerteágazó tevékenységére jellemző, hogy a számítástechnikai kutatásoktól a vitorlás hajó tervezéséig terjed a kutatási skálája. (Ez utóbbi eredménye volt a hosszú távú vitorlázás világbajnokságának számító Amerika Kupán győztes Stars and Stripes hajó tervezése.)

A Delta a cég Artifical Neural Systems (ANS) technológiája szerint készült. Ez az emberi idegrendszer és az agytevékenység bizonyos részeinek tanulási és információfeldolgozó működését utánozza. A processzorral nagy hálózatok szimulációja valósítható meg (például folyamatos beszédnél, autonóm forgalomszabályozásnál). Ezt a következők segítik elő. A lebegőpontos segédprocesszor (lásd a *fényképet*) sebessége 20 Mflop (a Mflop=millió lebegőpontos művelet másodpercenként). Ez tehát azt jelenti, hogy sebessége mintegy két nagyságrenddel nagyobb, mint egy Intel 80386 alapú gépé. A tárkapacitása 12 Mbájt. (A fényképen látható kártyán 12 db, egyenként 1 Mbájt kapacitású kis kártya a RAM-tár.) A processzor csővezetékkel ellátott, Harvard-architektúrájú (egy utasítás- és két adatgyorsító tárral) és 1024 regiszterű (!).

Jellemzői:

— 32 és 64 bites integer és lebegőpontos adatok, melyek megfelelnek az ANSI/IEEE Standard 754 DEC F és G formátumának;



6. ábra

— több mint 1800 utasítás (beleértve 96 feltételes, formátumátalakító, mátrix- és SQRT utasítást);

— az utasítások ciklusideje 100 ns (a legtöbb utasítás végrehajtódik egy cikluson belül, beleértve a lebegőpontos szorzást is);

— 32 bit széles a perifériabusz: 32 külső készülékkel tarthat kapcsolatot 40 Mbájt/s sebességgel, és rendelkezik PC/AT busszal is. Mindezek a kártyán levő csatlakozókon át, egyetlen PC/AT kártyán (a megfelelő hűtés érdekében az egyik szomszédos csatlakozót azonban nem használhatjuk);

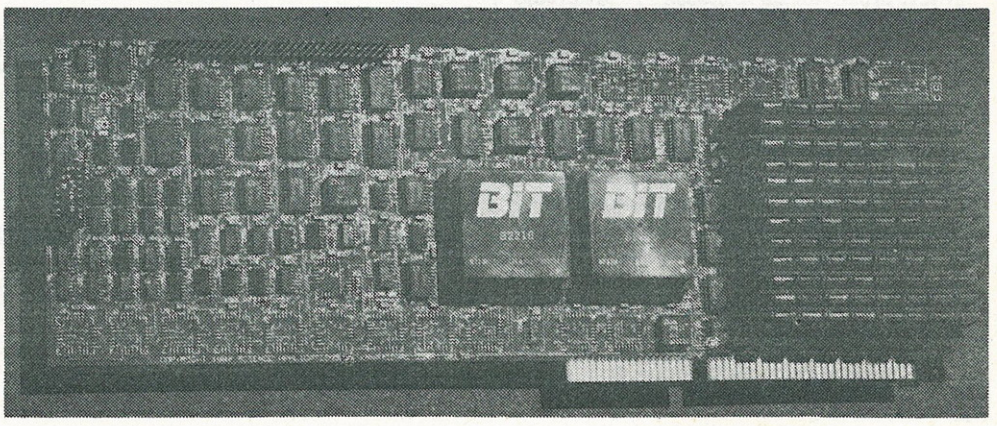
— gazdag a szoftver-ellátottsága is.

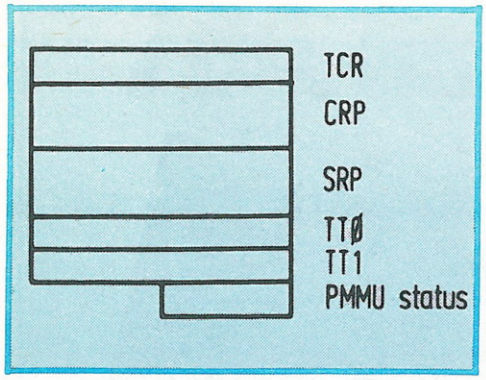
A COMDEX/FALL 87-en, ahol a terméket bevezették, George Works, a SAIC AND Division igazgatója nyilvánosságra hozta, hogy kidolgoztak egy speciális kiterjesztő egységet, ami 5 Delta-kártyát használhat, és amelynek a *számítási teljesítménye megegyezik a Cray-1 számítógépével.* (Ez a világ egyik leggyorsabb, tudományos célú nagyszámítógépe.)

Atari ABAQ

Az Atari céget — egyetlen itthon is forgalmazott gépe alapján — az olvasók ismerik. Valószínűleg kevésbé ismert azonban, hogy a cég újabban „felfelé lépked". Korszerű termékei 16 és 32 bites mikroprocesszorú gépek, amelyekről e sorozat későbbi részében lesz szó. Ezekhez készült az ABAQ (az abacus, az ősi számolóeszköz nevének szógyöke).

A cég elnöke, Sam Tramiel (a cégtulajdonos és korábban a Commodore-elnök, Tramiel fia) a COMDEX-en fénykép minőségű grafikát produkáló, különlegesen nagy teljesítményű számfeldolgozóként jellemezte ezt az eszközt, amely egy szuperszámítógép teljesítményét adja egy mikrogépnek. Úgy vélte, hogy reális áron 100 MIPS (millió utasítás másodpercenként) is elérhető lesz ezzel az eszközzel.





7. ábra

A processzor RISC rendszerű, párhuzamos processzorokból épül fel. Beépített nagy sebességű soros csatornája lehetővé teszi, hogy sok processzoros tömbként (multi-processor array) vagy helyi hálózatként (local area network, LAN) használják.

A rendszer az Inmos T—800 típusú, 32 bites, 10-12 MIPS sebességű mikroprocesszorra épül.

Jellemzői:

— a számítások sebessége 1,5 Mflop;

— 4 bájt RAM-ot tartalmaz és további 1 Mbájt RAM a kijelzőhöz kártyánként;

— a rendszer felfűzhető 3 kártyáig (kártyánként 4 processzorral);

— a képernyő 1280 × 960 pontos, 16 színű vagy kétszínű képponttal vagy 1024 × 768 pontos, 256 színű képponttal vagy 640 × 480 pontos, 256 színű kétképernyős képponttal vagy 512 × 480 pontos, 16 millió színű (!!) képponttal rendelkezik.

Az eszközhöz Unix-szerű operációs rendszert készítettek.

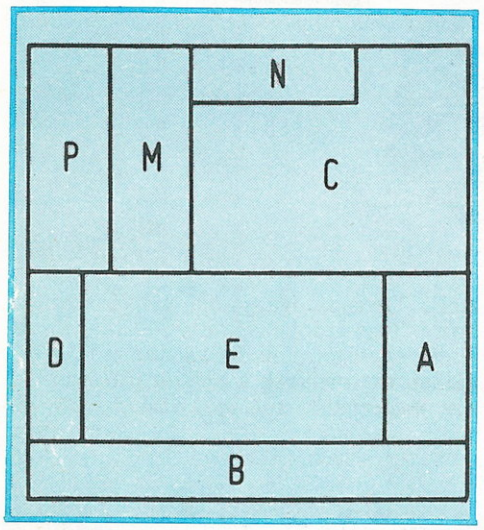
A Lowell egyetem Parallel Data Flow Coprocessor terméke

A Massachusetts kormányzója által alapított egyetem Produktivitás-kiterjesztő Központja egyrészt a különböző tanszékek együttesének kutató/tanító munkáját hangolja össze, másrészt együttműködik a különböző ipari intézményekkel. A központ a Commodore cég Amiga gépeihez fejlesztette ki a címben szereplő párhuzamos adatáramlású koprocesszort. Képi folyamatok előállítására, folyamatos beszédszintézisre, grafika generálására/módosítására, digitális jelfolyamatokban, képkompressziónál, elektronikus nyomtatás teljesítménynövelésénél, numerikus folyamatok teljesítménynövelésénél, tömbfolyamatoknál, mátrix/vektorműveleteknél, adattitkosításnál és elektronikus áramkörök szimulálásánál kívánják használni.

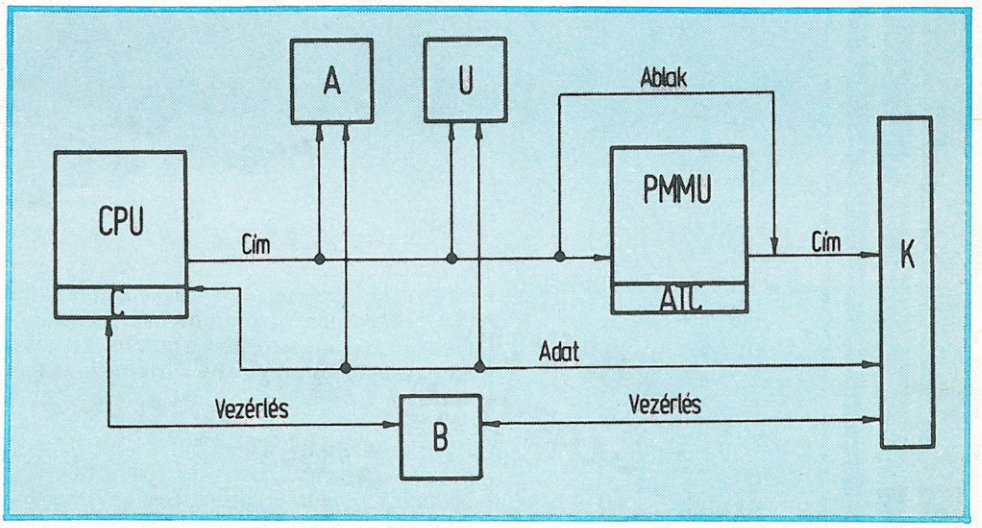
A COMDEX-en a fentieken kívül csak annyit közöltek, hogy a NEC által készített, hét integrált áramkörből álló eszköz 30 MIPS sebességet is felülmúl.

Motorola MC68030

A COMDEX-en jelentette be a Motorola cég is a régóta ígért és specifikációjában ismert 32 bites processzora árusítását. Mivel erről a processzorról már szó volt a Számítástechnika 1987. augusztus 26-i számának 20. oldalán, csak az ott nem tárgyaltakról, valamint a jelen cikkel kapcsolatos dolgokról írok.



8. ábra



9. ábra

A *6. ábrán* a felhasználói programozói modell látható. Van tehát 7 címregiszter, egy-egy veremmutató (stack pointer, SP), programszámláló (program counter, PC), 8 adatregiszter (ezek duplaszavasak = 32 bit), egy egybájtos állapotregiszter (status register, SR), állapot-kódregiszter (control code register, CCR). A *7. ábra* a felügyelői modell regisztereit mutatja be. Ezek az átvitelirányító regiszter (translation control register, TCL), a négyszeres szavas CPU gyökérmutató (CPU root pointer, CRP) és a felügyelői gyökérmutató (supervisor root pointer, SRP), a két transzparens átviteli regiszter (transparent translation, TTØ és TT1), amelyek ismét duplaszavasak, valamint az egyszavas laptárkezelő állapot (paging memory management unit status PMMU.)

Bár az utasítások két óraciklus idejűek, a belső és külső busz aktivitásának és a belső processzor-részegységek működésének időbeni átfedése miatt az átlagban egy utasításra fordított idő körülbelül egy ciklus. Az utasítás tényleges végrehajtása úgy történik, hogy a végrehajtó egységben (execution unit, EU) az első ciklusban egy hívás generálódik a mikrokódolás első szintjére. Ez egy ún. nanokódot vált ki, és az hajtódik végre a második szinten. A mikrokódok az ellenőrző hardver módosítása nélkül változtathatók, mivel az integrált áramkör fizikailag kettéosztható egy szoftverfordító és egy végrehajtó részre. Ez a *8. ábrán* látható, ahol az alsó fél egy RISC-gépnek megfelelő felépítésű. (Az ábrán N = nano ROM, M = mikro ROM, P = PMMU és ATC — adress transfer control, címátadás-vezérlés — C = control és PLA, D = DTAGS, E = EU, A = adatgyorsító, B = bus control.)

Az egyenként 64 duplaszavas gyorsítók közül az adat írható is, az utasítás csak olvasható. Az adatgyorsító regiszterként is használható. A két gyorsító önálló működése és a mindkettőhöz egyidejű hozzáférés lehetősége egy módosított Harvard-architektúrát eredményezett *(9. ábra,* melyen C = utasítás csővezeték, K = külső busz). Itt három belső busz van, amelyek mindegyike a többitől függetlenül működtethető. A csővezeték háromállapotú, és kapcsolatát a külső busszal háromféle ciklustípussal valósítja meg. Az egyik az összes korábbi Motorola processzornál is használt aszinkron buszciklus, a második egy két óraciklusnyi szinkron buszciklus, ami jól használható az újfajta dinamikus RAM-okhoz: az ún. statikus lap és félbájt és oszlop üzemmódúakhoz. Ezeknél 5 óraciklus alatt átvihető négy duplaszó. A harmadik az ún. csomó típus, ami a változó igényű perifériakezeléshez ad szabadon választható hosszúságú ciklusmegadási lehetőséget.

A RISC-gép koncepció megvalósításához a processzorban van léptető (dupla szavak léptetésére, forgatására egyetlen óraciklus alatt), laptárkezelő (ami a késésmentes címáthelyezésre szolgál a processzor egységei között, elvégezve ezzel az adatforgalom irányítását). A RISC-gép tulajdonságok miatt az órafrekvencia elérheti a 25 MHz értéket 5 MIPS teljesítmény mellett.

Dr. S. E.

KOMPLEX SZÁMÍTÁSTECHNIKAI SZOLGÁLTATÁS!

A Ferroglobus számítástechnikai főosztálya nagy adatfeldolgozási hagyományokkal, jól képzett, nagy tapasztalatú munkatársakkal rendelkező számítóközpontja széles skálájú, komplex számítástechnikai szolgáltatásait ajánlja:

— kereskedelmi szakmai, készletgazdálkodási, pénzügyi, számviteli, bér- és munkaügyi stb. rendszerek szervezése, programozása és fejlesztése nagy kapacitású, modern ICL típusú számítógépen

— mágneses adathordozóra történő adat-előkészítés — rögzítés

— számítógépidő bérbeadása

— tanácsadás

Ferroglobus

TEK V.
számítástechnikai főosztály
Budapest VII., Vörösmarty u. 16.
Tel.: 427-338, 202-415

Z80-as utasítástáblázat

Lapunk 1985/4-es számának mellékleteként közöltük a Z80-as mikroprocesszor utasításainak kódtáblázatát. Ezt követően olvashatták 7 részes sorozatunkat „Z80 programozási gyakorlatok" címen. Ezt a témakört most a Z80-as utasítástáblázat közreadásával tesszük teljessé.

TAKÁCS FERENC

Utasítás-alak	Funkció	Idö-igény	Felt.reg. SZHPNC	Hexa.kód 1.	2.	3.	4.
8 bites adatmozgatás							
LD r,s	r=s	4	.....	40			
LD rx,xs	rx=xs	8	.....	DD	40		
LD xr,sx	xr=sx	8	.....	DD	40		
LD r,(HL)	r=(HL)	7	.....	46			
LD (HL),s	(HL)=s	7	.....	70			
LD r,(x+d)	r=(x+d)	19	.....	DD	46	d	
LD (x+d),s	(x+d)=s	19	.....	DD	70	d	
LD r,n	r=n	7	.....	06	n		
LD xr,n	xr=n	11	.....	DD	06	n	
LD (HL),n	(HL)=n	10	.....	36	n		
LD (x+d),n	(x+d)=n	19	.....	DD	36	d	n
LD A,(BC)	A=(BC)	7	.....	0A			
LD A,(DE)	A=(DE)	7	.....	1A			
LD (BC),A	(BC)=A	7	.....	02			
LD (DE),A	(DE)=A	7	.....	12			
LD A,(hl)	A=(hl)	13	.....	3A	l	h	
LD (hl),A	(hl)=A	13	.....	32	l	h	
LD A,I	A=I	9	++0+0.	ED	57		
LD A,R	A=R	9	++0+0.	ED	5F		
LD I,A	I=A	9	.....	ED	47		
LD R,A	R=A	9	.....	ED	4F		
16 bites adatmozgatás							
LD dd,hl	dd=hl	10	.....	01	l	h	
LD x,hl	x=hl	14	.....	DD	21	l	h
LD HL,(hl)	H=(hl+1),L=(hl)	16	.....	2A	l	h	
LD ss,(hl)	ssH=(hl+1),ssL=(hl)	20	.....	ED	4B	l	h
LD x,(hl)	xH=(hl+1),xL=(hl)	20	.....	DD	2A	l	h
LD (hl),HL	(hl+1)=H,(hl)=L	16	.....	22	l	h	
LD (hl),ss	(hl+1)=ssH,(hl)=ssL	20	.....	ED	43	l	h
LD (hl),x	(hl+1)=xH,(hl)=xL	20	.....	DD	22	l	h
LD SP,HL	SP=HL	6	.....	F9			
LD SP,x	SP=x	10	.....	DD	F9		
PUSH qq	(SP-1)=qqH,(SP-2)=qqL,SP=SP-2	11	.....	C5			
PUSH x	(SP-1)=xH,(SP-2)=xL,SP=SP-2	15	.....	DD	E5		
POP qq	SP=SP-2,qqL=(SP-2),qqH=(SP-1)	10	.....	C1			
POP x	SP=SP-2,xL=(SP-2),xH=(SP-1)	14	.....	DD	E1		
Csere, keresés és tömbátvitel							
EX DE,HL	DE <-> HL	4	.....	EB			
EX AF,AF'	AF <-> AF'	4	.....	08			
EXX	BC <-> BC', DE <-> DE', HL <-> HL'	4	.....	D9			
EX (SP),HL	H <-> (SP+1), L <-> (SP)	19	.....	E3			
EX (SP),x	xH <-> (SP+1), xL <-> (SP)	23	.....	DD	E3		
LDI	(DE)=(HL),DE=DE+1,HL=HL+1,BC=BC-1	16	..0+0.	ED	A0		
LDIR	(DE)=(HL),DE=DE+1,HL=HL+1,BC=BC-1	21<>.	..000.	ED	B0		
	Ha BC<>0, akkor PC=PC-2	16=					
LDD	(DE)=(HL),DE=DE-1,HL=HL-1,BC=BC-1	16	..0+0.	ED	A8		
LDDR	(DE)=(HL),DE=DE-1,HL=HL-1,BC=BC-1	21<>.	..000.	ED	B8		
	Ha BC<>0, akkor PC=PC-2	16=					

Utasítás-alak	Funkció	Idö-igény	Felt.reg. SZHPNC	Hexa.kód 1.	2.	3.	4.
CPI	Ha A=(HL), akkor #Z=1,HL=HL+1,BC=BC-1	16	++++1.	ED	A1		
	Ha BC=0, akkor #P=0						
CPIR	Ha A=(HL), akkor #Z=1,HL=HL+1,BC=BC-1	21	++++1.	ED	B1		
	Ha BC=0, akkor #P=0	16					
	Ha #Z=0 es #P=1, akkor PC=PC-2						
CPD	Ha A=(HL), akkor #Z=1,HL=HL-1,BC=BC-1	16	++++1.	ED	A9		
	Ha BC=0, akkor #P=0						
CPDR	Ha A=(HL), akkor #Z=1,HL=HL-1,BC=BC-1	21	++++1.	ED	B9		
	Ha BC=0, akkor #P=0	16					
	Ha #Z=0 es #P=1, akkor PC=PC-2						
	8 bites aritmetikai és logikai utasitások						
ADD A,s	A=A+s	4	++++0+	80			
ADD A,xs	A=A+xs	8	++++0+	DD	80		
ADD A,n	A=A+n	7	++++0+	C6	n		
ADD A,(HL)	A=A+(HL)	7	++++0+	86			
ADD A,(x+d)	A=A+(x+d)	19	++++0+	DD	86	d	
ADC A,s	A=A+s+#C	4	++++0+	88			
ADC A,xs	A=A+xs+#C	8	++++0+	DD	88		
ADC A,n	A=A+n+#C	7	++++0+	CE	n		
ADC A,(HL)	A=A+(HL)+#C	7	++++0+	8E			
ADC A,(x+d)	A=A+(x+d)+#C	19	++++0+	DD	8E	d	
SUB s	A=A-s	4	++++1+	90			
SUB xs	A=A-xs	8	++++1+	DD	90		
SUB n	A=A-n	7	++++1+	D6	n		
SUB (HL)	A=A-(HL)	7	++++1+	96			
SUB (x+d)	A=A-(x+d)	19	++++1+	DD	96	d	
SBC A,s	A=A-s-#C	4	++++1+	98			
SBC A,xs	A=A-xs-#C	8	++++1+	DD	98		
SBC A,n	A=A-n-#C	7	++++1+	DE	n		
SBC A,(HL)	A=A-(HL)-#C	7	++++1+	9E			
SBC A,(x+d)	A=A-(x+d)-#C	19	++++1+	DD	9E	d	
AND s	A=A AND s	4	++1+00	A0			
AND xs	A=A AND xs	8	++1+00	DD	A0		
AND n	A=A AND n	7	++1+00	E6	n		
AND (HL)	A=A AND (HL)	7	++1+00	A6			
AND (x+d)	A=A AND (x+d)	19	++1+00	DD	A6	d	
XOR s	A=A XOR s	4	++0+00	A8			
XOR xs	A=A XOR xs	8	++0+00	DD	A8		
XOR n	A=A XOR n	7	++0+00	EE	n		
XOR (HL)	A=A XOR (HL)	7	++0+00	AE			
XOR (x+d)	A=A XOR (x+d)	19	++0+00	DD	AE	d	
OR s	A=A OR s	4	++0+00	B0			
OR xs	A=A OR xs	8	++0+00	DD	B0		
OR n	A=A OR n	7	++0+00	F6	n		
OR (HL)	A=A OR (HL)	7	++0+00	B6			
OR (x+d)	A=A OR (x+d)	19	++0+00	DD	B6	d	
CP s	A-s	4	++++1+	B8			
CP xs	A-xs	8	++++1+	DD	B8		
CP n	A-n	7	++++1+	FE	n		
CP (HL)	A-(HL)	7	++++1+	BE			
CP (x+d)	A-(x+d)	19	++++1+	DD	BE	d	
INC r	r=r+1	4	++++0.	04			
INC xr	xr=xr+1	8	++++0.	DD	04		
INC (HL)	(HL)=(HL)+1	11	++++0.	34			
INC (x+d)	(x+d)=(x+d)+1	23	++++0.	DD	34	d	
DEC r	r=r-1	4	++++1.	05			
DEC xr	xr=xr-1	8	++++1.	DD	05		
DEC (HL)	(HL)=(HL)-1	11	++++1.	35			
DEC (x+d)	(x+d)=(x+d)-1	23	++++1.	DD	35	d	

Utasitás-alak	Funkció	Idö-igény	Felt.reg. SZHPNC	Hexa.kód 1.	2.	3.	4.
16 bites aritmetikai utasitások							
ADD HL,dd	HL=HL+dd	11	..!.0+	09			
ADC HL,dd	HL=HL+dd+#C	15	++!+0+	ED	4A		
SBC HL,dd	HL=HL-dd-#C	15	++!+1+	ED	42		
ADD x,xd	x=x+xd	15	..!.0+	DD	09		
INC dd	dd=dd+1	6	.....	03			
INC x	x=x+1	10	.....	DD	23		
DEC dd	dd=dd-1	6	.....	0B			
DEC x	x=x-1	10	.....	DD	2B		
Általános aritmetikai és CPU vezérlö utasitások							
DAA	Kiigazitás BCD müvelet után	4	++++.+	27			
CPL	A= egyes komplemense A-nak	4	..1.1.	2F			
NEG	A=-A	8	++++1+	EB	44		
CCF	#C=1-#C	4	..!.0+	3F			
SCF	#C=1	4	..0.01	37			
NOP		4	.....	00			
HALT	CPU megállitás	4	.....	76			
DI	Megszakitás megtiltás	4	.....	F3			
EI	Megszakitás lehetövé tétel	4	.....	FB			
IM i	Megszakitásmód beállitás	8	.....	EB	46		
Léptetö utasitások							
RLCA	Ciklikus léptetés balra, #C a körben	4	..0.0+	07			
RRCA	Ciklikus léptetés jobbra, #C a körben	4	..0.0+	0F			
RLA	Ciklikus léptetés balra	4	..0.0+	17			
RRA	Ciklikus léptetés jobbra	4	..0.0+	1F			
RLC s	Ciklikus léptetés balra, #C is a körben	8	++0+0+	CB	00		
RLC (HL)	Ciklikus léptetés balra, #C is a körben	15	++0+0+	CB	06		
RLC (x+d)	Ciklikus léptetés balra, #C is a körben	23	++0+0+	DD	CB	d	06
RRC s	Ciklikus léptetés jobbra,#C is a körben	8	++0+0+	CB	08		
RRC (HL)	Ciklikus léptetés jobbra,#C is a körben	15	++0+0+	CB	0E		
RRC (x+d)	Ciklikus léptetés jobbra,#C is a körben	23	++0+0+	DD	CB	d	0E
RL s	Ciklikus léptetés balra	8	++0+0+	CB	10		
RL (HL)	Ciklikus léptetés balra	15	++0+0+	CB	16		
RL (x+d)	Ciklikus léptetés balra	23	++0+0+	DD	CB	d	16
RR s	Ciklikus léptetés jobbra	8	++0+0+	CB	18		
RR (HL)	Ciklikus léptetés jobbra	15	++0+0+	CB	1E		
RR (x+d)	Ciklikus léptetés jobbra	23	++0+0+	DD	CB	d	1E
SLA s	"Aritm." léptetés balra (s*2)	8	++0+0+	CB	20		
SLA (HL)	"Aritm." léptetés balra ((HL)*2)	15	++0+0+	CB	26		
SLA (x+d)	"Aritm." léptetés balra ((x+d)*2)	23	++0+0+	DD	CB	d	26
SRA s	"Aritm." léptetés jobbra (s/2)	8	++0+0+	CB	28		
SRA (HL)	"Aritm." léptetés jobbra ((HL)/2)	15	++0+0+	CB	2E		
SRA (x+d)	"Aritm." léptetés jobbra ((x+d)/2)	23	++0+0+	DD	CB	d	2E
SRL s	"Logikai" léptetés jobbra	8	++0+0+	CB	38		
SRL (HL)	"Logikai" léptetés jobbra	15	++0+0+	CB	3E		
SRL (x+d)	"Logikai" léptetés jobbra	23	++0+0+	DD	CB	d	3E
RLD	Ciklikus tetrád léptetés balra	18	++0+0.	ED	6F		
RRD	Ciklikus tetrád léptetés jobbra	18	++0+0.	ED	67		
Bitkezelö utasitások							
BIT b,s	Ha r.b=0, akkor #Z=1	8	!+1!0.	CB	40		
BIT b,(HL)	Ha (HL).b=0, akkor #Z=1	12	!+1!0.	CB	46		
BIT b,(x+d)	Ha (x+d).b=0, akkor #Z=1	20	!+1!0.	DD	CB	d	46
RES b,s	r.b=0	8	.....	CB	80		
RES b,(HL)	(HL).b=0	15	.....	CB	86		
RES b,(x+d)	(x+d).b=0	23	.....	DD	CB	d	86
SET b,s	r.b=1	8	.....	CB	C0		
SET b,(HL)	(HL).b=1	15	.....	CB	C6		
SET b,(x+d)	(x+d).b=1	23	.....	DD	CB	d	C6

Utasítás-alak	Funkció	Idö-igény	Felt.reg. S Z H P N C	Hexa.kód 1.	2.	3.	4.
	Vezérlésátadási utasitások						
JP hl	PC=hl	10	.....	C3	l	h	
JP cc,hl	Ha cc teljesül, akkor PC=hl	10	.....	C2	l	h	
JR d	PC=PC+d	12	.....	18	d		
JR c,d	Ha c teljesül, akkor PC=PC+d	7,12	.....	20	d		
JP (HL)	PC=HL	4	.....	E9			
JP (x)	PC=x	8	.....	DD	E9		
DJNZ d	B=B-1 Ha B<>0, akkor PC=PC+d	8,13	.....	10	d		
CALL hl	PUSH PC, PC=hl	17	.....	CD	l	h	
CALL cc,hl	Ha cc teljesül, akkor PUSH PC,PC=hl	10,17	.....	C4	l	h	
RET	POP PC	10	.....	C9			
RET cc	Ha cc teljesul, akkor POP PC	5,11	.....	C0			
RETI	Megszakitási szubrutin vége	14	.....	ED	4D		
RETN	Megszakitási szubrutin vége	14	.....	ED	45		
RST p	PUSH PC, PC.H=0, PC.L=p	11	.....	C7			
	Kimásolási-bemásolási utasitások						
IN A,(n)	A=IN(An) "bemásolás (An)-röl A-ba"	11	.....	DB	n		
IN r,(C)	r=IN(BC)	12	++++0.	ED	40		
IN F,(C)	IN(BC)	12	++++0.	ED	70		
INI	(HL)=IN(BC), B=B-1, HL=HL+1	16	!+!!1!	ED	A2		
INIR	(HL)=IN(BC), B=B-1, HL=HL+1	21<>!1!!1!		ED	B2		
	Ha B<>0, akkor PC=PC-2	16=					
IND	(HL)=IN(BC), B=B-1, HL=HL-1	16	!+!!1!	ED	AA		
INDR	(HL)=IN(BC), B=B-1, HL=HL-1	21<>!1!!1!		ED	BA		
	Ha B<>0, akkor PC=PC-2	16=					
OUT (n),A	OUT(An)=A "Kimásolás A-ból (An)-re"	11	.....	D3	n		
OUT (C),r	OUT(BC)=r	12	.....	ED	41		
OUTI	OUT(BC)=(HL),B=B-1,HL=HL+1	16	!+!!1!	ED	A3		
OTIR	OUT(BC)=(HL),B=B-1,HL=HL+1,	21<>!1!!1!		ED	B3		
	Ha B<>0, akkor PC=PC-2	16=					
OUTD	OUT(BC)=(HL),B=B-1,HL=HL-1	16	!+!!1!	ED	AB		
OTDR	OUT(BC)=(HL),B=B-1,HL=HL-1,	21<>!1!!1!		ED	B8		
	Ha B<>0, akkor PC=PC-2	16=					

Megjegyzések

Jelzöbit beállitások: 1=egyre állitás, 0=nullára állitás, +=eredménytöl függö beállitás, !=nem definiált értékre állitás, .=a jelzöbit nem változik.

A táblázat funkció oszlopában a jelzöbiteket # jelöli.

Az F regiszter jelzöbitjei (A jelzés 1-es állapotba állitással történik.)

7.	S	negativ eredmény jelzése
6.	Z	nulla eredmény jelzése
5.	-	nem használt
4.	H	félátvitel jelzése
3.	-	nem használt
2.	P/V	túlcsordulás, páros paritású eredmény, IFF2 állapotának jelzése
1.	N	kivonás jelzése
0.	C	átvitel jelzése

Léptetéstipusok szemléltetése

Ciklikus léptetés jobbra:

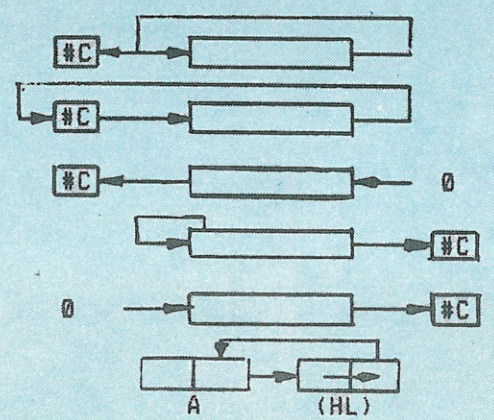
Ciklikus léptetés jobbra, #C is a körben:

"Aritmetikai" léptetés balra:

"Aritmetikai" léptetés jobbra:

"Logikai" léptetés jobbra:

Ciklikus tetrád léptetés jobbra:



Jelölések helyettesitése

Az assembly nyelvü utasitásban szereplö minden kisbetüs jelölést helyettesiteni kell az alábbi táblázat szerint. A vesszövel elválasztott helyettesithetö operandusok mellett zárójelben a gépi kódhoz OR müvelettel kapcsolandó maszkokat találjuk. A gépi kód aláhúzása jelzi, hogy az adott bájtot maszkolni kell. A maszkok - az x, xh, és xl jelölés helyettesitésénél levök kivételével - az utasitás utolsó aláhúzott bájtjához kapcsolandók. A kivételeknél a maszk az elsö bájthoz kapcsolódik.

A helyettesitések:

```
 r:A(38),B(00),C(08),D(10),E(18),H(20),L(28)
 s:A(07),B(00),C(01),D(02),E(03),H(04),L(05)
rx:A(38),B(00),C(08),D(10),E(18),xh(20),xl(28)
sx:A(07),B(00),C(01),D(02),E(03),xh(04),xl(05)
xr:xh(20),xl(28)
xs:xh(04),xl(05)
dd:BC(00),DE(10),HL(20),SP(30)
qq:BC(00),DE(10),HL(20),AF(30)
xd:BC(00),DE(10), x(20),SP(30)
ss:BC(00),DE(10),SP(30)
 x:IX(DD),IY(FD)  (elsö bájt maszkja)
xh:XH(DD),YH(FD)  (elsö bájt maszkja)
xl:XL(DD),YL(FD)  (elsö bájt maszkja)
 b:0(00),1(08),2(10),3(18),4(20),5(28),6(30),7(38)
 c:NZ(00),Z(08),NC(10),C(18)
cc:NZ(00),Z(08),NC(10),C(18),PO(20),PE(28),P(30),M(38)
 p:0(00),8(08),16(10),24(18),32(20),40(28),48(30),56(38)
 i:0(00),1(10),2(18)
 d: 8 bites relativ érték
 n: 8 bites állandó
hl: 16 bites állandó
```

Behelyettesitési példák

1. Tekintsuk a "BIT 5,D" utasitást. A táblázatban a "BIT b,s" utasitás mellett a "CB 40" kódot találjuk. Mint a helyettesitések táblázatából kiderül, a "b" 5-ös értékéhez "28"-as maszk tartozik, az "s" helyére való D-hez "2"-es maszk. A keresett utasitás kódja ezek szerint a következöképpen alakul:

```
1. bájt változatlanul   CB
2. bájt (40 or 28 or 2)=6A
```

2. Tekintsük a "BIT 3,(IY-8)" utasitást. A táblázatban a "BIT b,(x+d)" utasitás mellett a "DD CB d 46" kódot találjuk. Mint a helyettesitések táblázatából kiderül, a "b" 3-as értékéhez "18"-as maszk tartozik, az "x" helyére való IY-hoz "FD" maszk, mig a "d" helyébe magát a 8 bites számot kell irni. Láthatjuk, hogy az "FD" az elsö bájt maszkja. A keresett utasitás kódja ezek szerint a következöképpen alakul:

```
1. bájt (DD or FD)=     FD
2. bájt változatlanul   CB
3. bájt -8, vagyis      F8
4. bájt (46 or 18)=     5E
```

3. Tekintsük az "LD C,YH" utasitást. A táblázatban az "LD rx,xs" utasitás mellett a "DD 40" kódot találjuk. Mint a helyettesitések táblázatából kiderül, az "rx" helyére való "C"-hez "8"-as maszk tartozik, az "xs" helyére való "xh"-hoz "4"-es maszk, amely tovább helyettesitendö IY-nal, melyhez az "FD" maszk tartozik. Ez utóbbi az elsö bájt maszkja. A keresett utasitás kódja ezek szerint a következöképpen alakul:

```
1. bájt (DD or FD)=     FD
2. bájt (40 or 8 or 4)= 4C
```

A bemutatott példákból is látszik, hogy a táblázat hatékony használatához érdemes megjegyeznünk, milyen operandusokat milyen betüjelölésekkel helyettesithetünk.

DIGITAL szelvény Mikroszámítógép Magazin 1988. április

Egy sarokkal olcsóbb!

A DIGITAL Számítástechnikai Szaküzlet (Budapest, Szilágyi Erzsébet fasor 35. 1026) Sinclair-termékekre szakosodott: elsősorban a ZX81-es és Spectrum gépekhez használatos eszközöket, programokat árul. De kaphatók itt más gépekhez való tartozékok (például botkormányok), számítástechnikai alkatrészek (integrált áramkörök stb.) és zsebszámológépek is.

Aki ebben az üzletben a lapunkból kivágott sarokszelvényt átadja, vagy megrendelésével együtt oda elküldi, minden hónapban más-más cikket olcsóbban vásárolhat meg. A kedvezmény a szelvényen feltüntetett hónapban érvényes. Minden árengedménnyel vásárolt darabhoz le kell adni egy szelvényt.

A bolt utánvéttel szállítást is vállal, és a szokásos 6 hónap helyett 1 év garanciát ad.

Az e havi kedvezmény

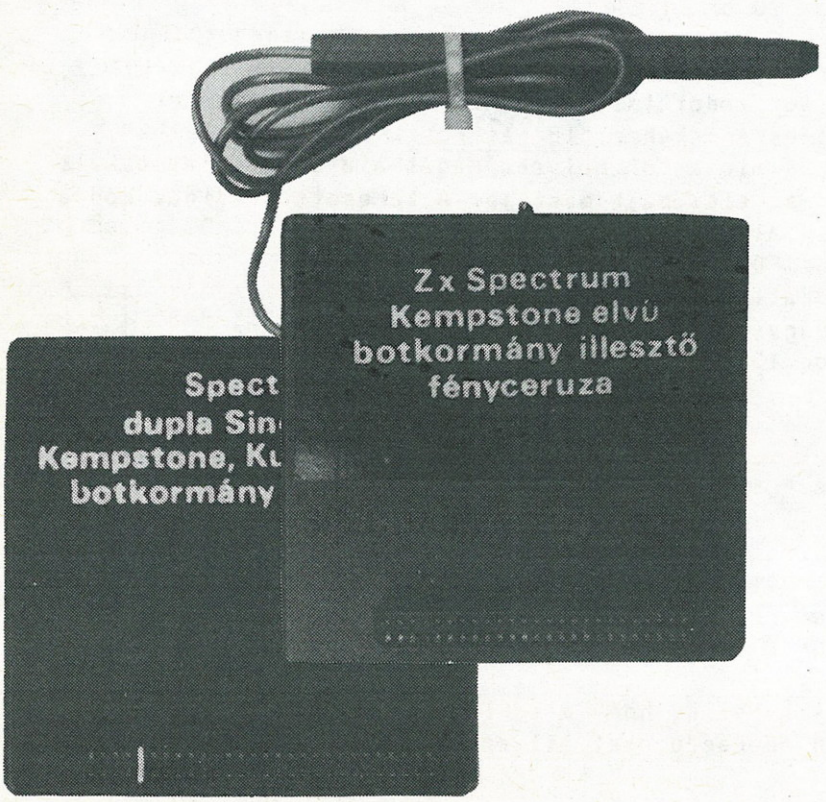
Egy botkormányos illesztő, fényceruzával. Ára 1800,— Ft, engedmény 5%.
Két botkormányos illesztő. Ára 1800,— Ft, engedmény 5%.

Az első változat a Spectrum gépek buszcsatlakozójára köthető, kapcsolós (Atari, Commodore típusú) botkormányok illesztésére szolgál. A játékprogramokhoz az ún. Kempstone típus a szabvány, amely az egy botkormányos játékokhoz való. Bár sok játékot billentyűzetről is lehet játszani, de mivel így hamar tönkremennek a billentyűk, célszerűbb botkormányt használni.

A külön illesztővel csatlakoztatott fényceruzával rajzolhatunk és törölhetünk a képernyőn, majd a kész ábrát eltárolhatjuk, magnetofonszalagra vihetjük.

A második változat, a két botkormányos illesztő két botkormányos játékhoz is jó, de nem tartalmaz fényceruza-illesztőt.

S. E.



Közprogramok

Olvasóink egyre gyakrabban küldenek be olyan hasznos programokat, amelyeket terjedelmi okok miatt nem áll módunkban közölni. Mivel úgy véljük, hogy közérdeklődésre tarthatnak számot, ezeket röviden ismertetjük. Akik használni szeretnék a programokat, a részletes leírást, a programlistákat vagy — ahol erre mód van — a programot adathordozón, levélben megrendelhetik szerkesztőségünktől. A listák, ill. adathordozók másolását a KASZKAD Kisszövetkezet óbudai PÓLUS szakcsoportja végzi (lásd lapunk 88/2. számát). A másolási díj listák, leírások esetén oldalanként 8 Ft, kazetta esetén 150 Ft, floppy esetén 300 Ft. Az adathordozóra történő másolás díja az adathordozó árát is magában foglalja.

Közületektől cégszerű megrendelést kérünk, a másolási díjat az MNB 208-42518-7014 számlára kell befizetni. Magánszemélyek a díjat a KKVMF PÓLUS Szakcsoport, 1034 Bp., Bécsi út 94—96. címre fizethetik be.

A megrendelés az alábbi formában történik:

MIKROSZÁMÍTÓGÉP MAGAZIN SZERKESZTŐSÉGE
1371 Budapest, Pf. 433.

Megrendelem a Mikroszámítógép Magazin 1988/... számában szereplő közprogramok közül az alábbiakat:

Program neve
Csak programlistát és leírást kérek
Példányszám
Csak adathordozót kérek
Mindkettőt kérem

A megrendeléshez csatolom a szolgáltatási díj befizetését igazoló csekkszelvényt. Dátum, név, pontos cím.

A programok ellenőrzése nem áll módunkban, ezért az esetleges programhibákért mi nem vállalhatunk felelősséget.

Kérjük, akik saját készítésű programjaikat felajánlják a köz javára, a részletes programleírást, a listát és az alábbiakhoz hasonló rövidített ismertetőt juttassák el címünkre. (Nevük jelenti a garanciát a továbbiakban a megrendelők számára.)

Osztályátlag-számító program

Programnév: Osztályátlag-számító
Géptípus: C16, Plus/4
Konfiguráció: magnetofon háttértár, meg nem nevezett típusú nyomtató
Adathordozó: kazettaszalag
Terjedelem: 348 soros BASIC program
Készítő neve: Stumphauser Tamás
Megjegyzés: a C16 csak bővített változat lehet

Leírás
oldalszáma: 3+4 oldal melléklet
Program
oldalszáma: 3

Maximálisan 16 tantárgyra és 40 tanulóra személyenként, csoportonként, osztályonként, tantárgyanként képes átlagot számítani. A személynevek maximális hossza nincs megadva, az osztálynév 3, a tantárgynév 20 karakter lehet. Az eredményeket az adatokkal együtt táblázatos formában ki lehet nyomtatni. Az adatok a háttértárba juttathatók, illetve onnan beolvashatók. A tanulók, tantárgyak adatai egyenként törölhetők, javíthatók, bevihetők. A program menürendszerű.

Orosz nyelvű feladatlap-szerkesztő program

Programnév: Orosz szövegíró
Géptípus: Plus/4
Konfiguráció: lemezegység (VC 1541) háttértár,
nyomtató (MPS801)
Adathordozó: 5.25" mágneslemez
Terjedelem: 207 BASIC sor
Készítő neve: Ködmön József
Megjegyzés: A megadott állományok (felmérők, versenyfeladatok, megoldások) helyett egyszerű módosítással mást is lehet használni.

Leírás
oldalszáma: 2
Program
oldalszáma: 4

A program egyszerre ki nem nyomtatható példányszámban készülő orosz szövegek nyomtatására használható. (Pl. feladatlapok, versenyek anyagának készítése stb.) A szöveg képernyőn szerkeszthető, lemezen tárolható, onnan elővehető, automatikus lapozással nyomtatható. 20 sornyi szöveg tizenhatszori nyomtatásának időigénye kb. 40 perc. A feladatlapok rendszerbe foglalhatók, utólag módosíthatók. A program menürendszerű.

Nagyfelbontású grafika

Programnév: Super HI-RES grafika
Géptípus: C128
Konfiguráció: alapgép
Adathordozó: nincs megadva
Terjedelem: 596 bájt+3 bemutató BASIC program (13, 10, 37 soros)
Készítő neve: Dávid Zoltán
Megjegyzés: Munka közben a parancsokat a 40 karakteres üzemmódban kell kiadni! Az ASCII/DIN billentyű lenyomása törölheti az ábrát!

Leírás
oldalszáma: 3
Program
oldalszáma: 6

A 80 karakter/sor felbontású képernyőüzemmód BASIC-ből grafikus utasításokkal nem üzemeltethető, bár a VICII IC ezt lehetővé tenné. A programmal ez a hiány pótolható. A módszer: a grafikát a 40 karakter/sor felbontású üzemmódban állítjuk elő, majd átírjuk a nagyfelbontású grafikus tárolóba.

Hardver

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozónak rendelkeznie kell alapfokú áramköri hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretekre.

A cikksorozat írójának szándéka, hogy átfogó, alapfokú hardverismereteket adjon azoknak az olvasóknak, akik már tudnak számítógépes programokat írni. Előre csak minimális áramköri ismeretet (Ohm-törvény) tételez fel. Mivel az írások elsődlegesen a digitális áramkörökkel ismertetnek meg, nagy előny, hogy elegendő ezek be- és kimeneti kapcsolását megérteni, a belső jelutak és kapcsolatok az alkalmazók számára közömbösek.

A tárgyalásban — a nézőpont jellegénél fogva — nem törekedhetünk teljes precizitásra, nem mélyedhetünk el a részletekben. Reális cél viszont a funkcionális működések bemutatása.

Áramköri alapismeretek

A jelenlegi integrált áramkörök félvezető diódákból és tranzisztorokból épülnek fel. Első lépésként ismerkedjünk meg a dióda és a tranzisztor funkcionális működésével.

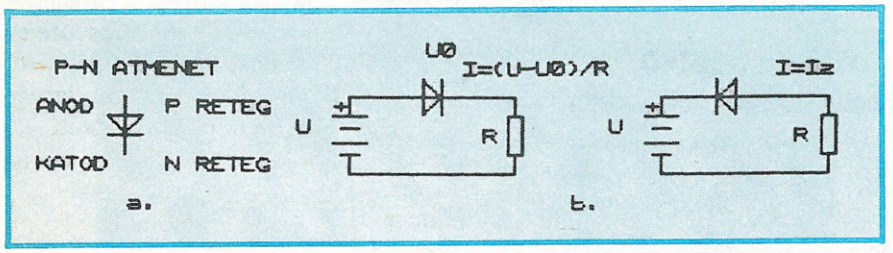
A dióda

Két kivezetést tartalmazó elem. Áramköri jelölése és a kivezetéseinek elnevezése az *1a* ábrán látható. Elméletileg a dióda működését a szilícium vagy germánium félvezető anyagban vegyi úton kialakított, ún. P—N átmenet vezetési mechanizmusával magyarázhatjuk meg. A gyakorlatban a kivezetéseket vagy a diódán lévő fenti szimbólum, vagy a katód oldalon lévő jelölés (festett gyűrű) alapján azonosíthatjuk.

Ha a dióda anódjára (ez kapcsolódik a P réteghez) pozitív feszültséget kapcsolunk a katódjához (N réteg) képest, a dióda kinyit (vezet), azaz áram folyik rajta, míg ellentétes feszültség esetén nem nyit ki, azaz zár (nem vezet, vagyis szigetel) *(1b ábra)*. A dióda jóságáról ezek szerint egy ellenállásmérő műszerrel győződhetünk meg: a dióda két vége között egyik irányban nagy ellenállást (szakadást), a mérővezetékeket felcserélve kis ellenállást (rövidzárt) kell mutatnia a műszernek.

A diódán nyitó irány esetén kb. $U\emptyset = \emptyset{,}7 \ldots 1$ V nagyságú feszültség esik. Ezt a dióda nyitó irányú feszültségének nevezzük. Ebből még az is következik, hogy a dióda akkor képes csak kinyitni, ha $U > U\emptyset$!

A dióda zárt állapotában csak nagyon kicsi, gyakorlatilag elhanyagolható, ún. szivárgási áram folyik (Iz).



1. ábra

A tranzisztor

A tranzisztor már két P—N átmenetet tartalmaz — ezért szokták bipoláris tranzisztornak is nevezni —, amely három kivezetéshez kapcsolódik. Jelölésük és kivezetéseik elnevezése az *2. ábrán* látható. Amint az ábra is jelzi, tulajdonképpen két összekapcsolt diódának is felfogható a két P—N átmenet miatt. (Vigyázat! Ez nem azt jelenti, hogy két diódából a fenti módon összekötve, tranzisztort tudunk készíteni! Az ábra alapján egyetlen tranzisztor szakadt vagy zárlatos voltáról tudunk meggyőződni: például ohm-mérővel a bázishoz rakva az egyik mérővezetéket, a kollektor és emitter irányban nagy ellenállást, a két vezetéket felcserélve kis ellenállást mérhetünk. A kollektor és az emitter között mindkét iránynál nagy ellenállást kell mérni.)

Az NPN és a PNP tranzisztortípus felhasználása lehetővé teszi különböző polaritású és kapcsolású tranzisztoros áramkörök kialakítását. Mi a továbbiakban az NPN tranzisztor funkcionális működését tanulmányozzuk. Az NPN tranzisztor működését a következőképpen fogalmazhatjuk meg. A bázis- (b) emitter (e) diódán átfolyó árammal (Ib) szabályozni tudjuk a kollektor- (c) emitter (e) kivezetések között folyó áramot (Ic). A tranzisztor fontos jellemzője az áramerősítési tényező, azaz az Ic/Ib arány. Ennek értéke néhányszor tíztől többezerszeres nagyságrendig terjedhet. Ezzel magyarázható a tranzisztor erősítőképessége: kis bázisárammal tudjuk az Ic kollektoráramot vezérelni. Amikor a tranzisztort úgy használjuk (a többi áramköri elemet úgy választjuk meg), hogy a bázisáram és a kollektoráram egymással arányos, akkor a tranzisztor lineáris üzemmódban működik. Ezt a tulajdonságát érvényesítik a különféle, például hangfrekvenciás erősítőkben.

Számunkra sokkal fontosabb a tranzisztor ún. kapcsoló üzemmódja. Ennek megértéséhez segítséget nyújt a *3. ábra.*

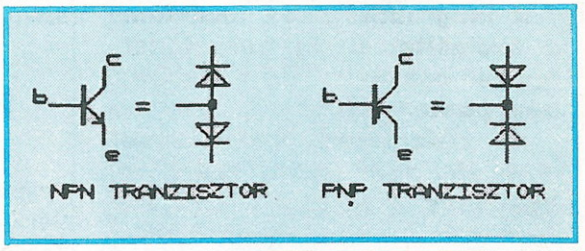
A K kapcsoló nyitott állapotában $Ib = \emptyset$. Ekkor természetesen az Ic áram is nulla, és az Uki kimeneti feszültség +5 V (nem esik feszültség az Rc ellenálláson). Ha a K kapcsolót zárjuk, akkor a bázisáram nagysága $Ib = (5 - \emptyset{,}7)\ V/1k\Omega = 4{,}3$ mA lesz. Itt a Ø,7 V-os érték a bázisemitter dióda nyitó irányú feszültségesése.

Mekkora lesz az Ic kollektoráram? Ha az áramerősítési tényezőt 100-nak tételezzük fel (szokványos érték), akkor elvileg $100 \times 4{,}3 = 430$ mA lehetne. De a kollektorkörben lévő $Rc = 1\ k\Omega$ ellenállás ezt az áramot nem engedi kialakulni, ezért úgy mondjuk, hogy a tranzisztor teljesen kinyit (c és e között közel nulla ellenállás lesz), és így $Ic = 5\ V/1\ k\Omega = 5$ mA! Az Ic és Ib már nem arányos egymással. Azaz ha $Ube = \emptyset V$, akkor $Uki = 5$ V, és ha $Ube = 5$ V, akkor $Uki = \emptyset V$.

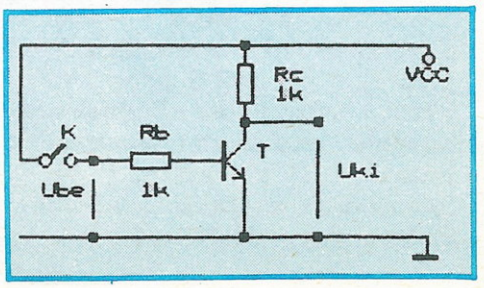
Logikai áramkörök

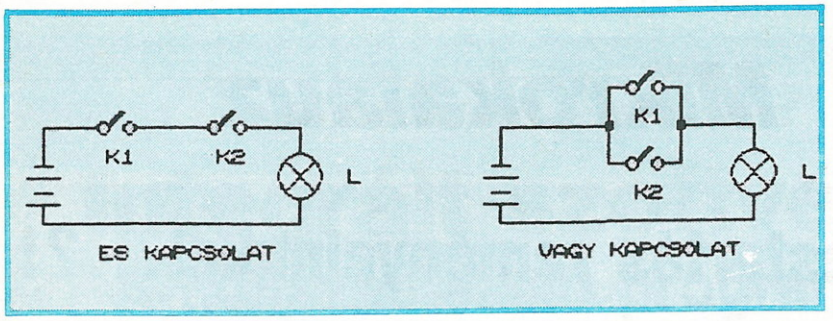
Azok, akik számítógépes programokat írnak, ismerik a logikai műveleteket. Elemi állítások igaz, illetve hamis voltához az 1 és Ø értékeket rendelve felépíthető egy kétértékű, más néven bináris vagy Boole-algebra, ami lehetőséget ad összetett állítások igaz vagy hamis voltának megállapítására.

2. ábra

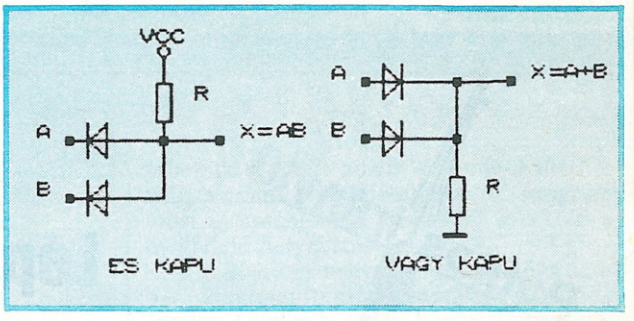


3. ábra





4. ábra



5. ábra

Az elemi Ø, illetve 1 értékű eseményekhez egy kapcsoló nyitott, illetve zárt állapotát rendelve, kapcsolókból felépített áramkörökkel valósíthatjuk meg az összetett események elektromos leképezését. A *4. ábrán* látható áramkörök a K1 és K2 kapcsolók állapotához rendelt elemi állítások ÉS, illetve VAGY kapcsolatát valósítják meg.

ÉS kapcsolat: a lámpa akkor ég, ha K1 ÉS K2 is zárt (mindkét elemi esemény igaz). VAGY kapcsolat: a lámpa akkor ég, ha K1 VAGY K2 VAGY mindkettő zárt (valamelyik vagy mindkét elemi esemény igaz).

Logikai funkciókat megvalósító áramköröket — kapcsoló tulajdonságaikat kihasználva — félvezető diódákból, illetve tranzisztorokból is felépíthetünk. Az így kialakított áramkörökben a Ø-át és 1-et mindig két feszültségszint képviseli, amelyeket L és H szinteknek is szoktunk nevezni. A leggyakoribb esetekben a két logikai állapothoz úgy rendelünk feszültségszinteket, hogy L-nek a Ø V, H-nak a pozitív feszültség (legtöbbször 5 V) feleljen meg. Ezt a megfeleltetést szokták pozitív logikának nevezni.

Kapuk

Diódák felhasználásával az előbbiekben kapcsolókkal bemutatott ÉS valamint a VAGY kapcsolat az *5. ábrán* vázolt, ún. diódás kapuáramkörökkel valósítható meg. Vizsgáljuk meg először az ÉS kapu működését. Ha az A és B jelű bemenetek mindegyike L szintű, akkor a két dióda vezet, ezért az X jelű pont (a kimenet) L szinten lesz (pontosabban UØ=Ø,7 V nyitóáramú feszültségen). Ha csak az egyik bemenet L szintű, akkor is vezetni fog a hozzá kapcsolódó dióda, azaz X továbbra is L szintű marad. Ahhoz, hogy a kimenet H szintű legyen, mindkét bemenetre H szintet kell kapcsolni. Akkor a diódák lezárnak, és az R ellenálláson áram nem folyik, ezért a kimenet H szintű lesz.

A VAGY kapu működése az előzőhöz hasonló módon elemezhető. Ha a bemenetek L szinten vannak, akkor az ellenálláson áram nem folyik, így X pont feszültsége nulla (L). Ha azonban valamelyik bemenetre H szintet kapcsolunk, akkor a dióda vezetni fog, az R ellenálláson áram folyik, és a kimenet H (pontosabban Vcc—Ø,7 V) szintű lesz.

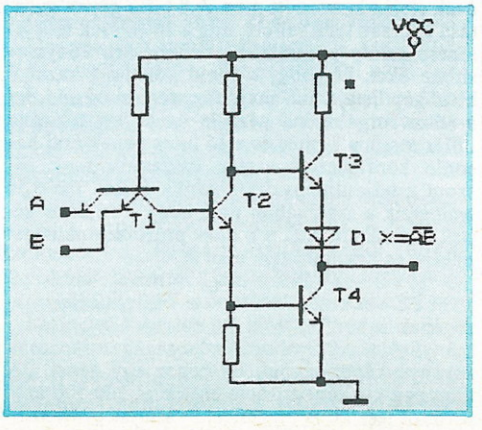
Ilyen diódás áramkörökkel bonyolult logikai függvények is megvalósíthatók, de előnyösebb tulajdonságaik miatt a tranzisztort mint kapcsoló elemet alkalmazó áramkörök váltak a legjobban elterjedtté. Csak példaként egy előnyösebb tulajdonság: logikai áramkörökben igen gyakran kell egy adott jel ellentettjét — negáltját — előállítani. Az ezt megvalósító áramkör — az inverter — diódával való előállítása meglehetősen bonyolult; tranzisztorral ez a legegyszerűbb feladat, hiszen a tranzisztor kapcsoló üzemmódját bemutató 3. ábra egy inverter: ha a bemenet Ube=Ø V (L szint), akkor a kimenet Uki=5 V (H szint), és ha Ube H szintű, akkor a kimenet L szintre vált.

A tranzisztorokat alkalmazó TTL (Tranzisztor-Tranzisztor Logika) igazi diadalát az integrált áramkörök megjelenése hozta. Az integrált áramkör olyan — kivezetésekkel ellátott — tokozott félvezető lapka, amelynek felületén sok-sok tranzisztor és dióda van működőképes áramkörré egyesítve, vagy más szóval integrálva. Mi ezeket az áramköröket csak a kivezetéseik felől „látjuk", ezért alkalmazásukhoz elegendő funkcionális működésük ismerete. Illusztrációként elemezzük egy két bemenetű NEM—ÉS (NAND) TTL kapuáramkör működését *(6. ábra)*.

Ha egy vagy mindkét emitter L szinten van, akkor a bemeneti R ellenálláson áram folyik (a T1 bázisemitter diódája vezet). Ennek következtében T1 kollektora L szinten lesz. A kollektor csak akkor lesz H szintű, ha valamennyi bemenet H szintűvé válik. Így a bemeneti rész tulajdonképpen ÉS kapcsolatot valósít meg. A következő fokozatot a T2 tranzisztor alkotja, ami az ún. totem-pole kimenet (T3—T4 tranzisztor, D dióda) megfelelő vezérlését (meghajtását) biztosítja. Amikor T2 vezet (T1 bemenetei H szinten vannak), akkor T3 lezár és T4 kinyit, ezzel a kimenet L szintű lesz, ami a NEM—ÉS funkciónak felel meg. A kimeneti láncban lévő D dióda biztosítja, hogy T3 lezárt állapotban legyen akkor, ha T4 kinyit. Ha T2 nem vezet, akkor T3 van nyitva és T4 zárva.

Méréssel a fenti működést úgy ellenőrizhetjük, hogy a föld és az X kimenet közé egy voltmérőt kötünk; akkor annak közel Ø V-ot (L szint) kell mutatnia. Az A és B bemeneteket egyszerre vagy külön-külön a Vcc feszültségre kötve, a voltmérő közel 5 V-ot (H szint) fog mutatni.

6. ábra



Más jellegű kimenet is kialakítható a fenti totem-pole kimenet helyett: az ún. nyitott kollektoros kimenet. Ilyenkor a *-gal jelzett ellenállás, a T3 jelű tranzisztor és a D dióda hiányzik a kimenetről, és az X jelű kimenet (T4 kollektora) van kivezetve. Ezért hívják nyitott vagy szabad kollektoros kimenetnek. Milyen tulajdonságai vannak az így kialakított kimenetnek?

— A T4 tranzisztor így nagyobb tápfeszültségen működő terhelést is képes kapcsolni: például egy 12 V-os izzót vagy relét, azaz TTL bemenetről vezérelhetünk nem TTL jellegű kimeneteket. Természetesen ilyenkor a T4 tranzisztoron átfolyó áramnak a megengedett érték (kb. 20 mA) alatt kell lennie.

— Amikor az ilyen nyitott kollektoros kapuk kimeneteit összekötjük, és a közös vezetéket a Vcc tápfeszültséghez csatlakoztatjuk, akkor ez a közös vezeték (X kimenet) csak abban az esetben lesz H szintű, ha valamennyi kimenet H szinten van. Ha csak egyetlen kimenet is L szinten van, akkor a közös kimenet is L szinten lesz, azaz a kimenetekkel ÉS kapcsolatot valósíthatunk meg. Az ilyen összekötött kimeneteket huzalozott ÉS (más megoldásnál huzalozott VAGY) kapcsolatnak is nevezik.

Logikai hálózatok

A digitális TTL kapuáramkörök ilyen és ehhez hasonló áramkörökből épülnek fel. Az ilyen kapuk egyértelműen jellemezhetők bemeneteik számával és azzal, hogy a bemenetek milyen logikai függvénye jelenik meg a kimeneten. Az integrált áramkörök összekapcsolása során gyakran több bemenetet kell egy kimenetre kapcsolni. TTL áramköröknél megadható, hogy egy kimenet maximálisan hány bemenettel terhelhető. Ez a szám a kimeneti terhelhetőség (fan out). Előfordulnak olyan bemenetek is, amelyek a kimenetet a szokásosnál jobban terhelik. Az ilyen áramkörök bemeneti terhelése (fan in) nagyobb mint 1.

Ezek alapján TTL áramkörökből álló logikai hálózatokat kapuáramkörökből a kimeneteik és bemeneteik megfelelő összekapcsolásával hozhatunk létre. Néhány alapszabályt azonban be kell tartanunk:

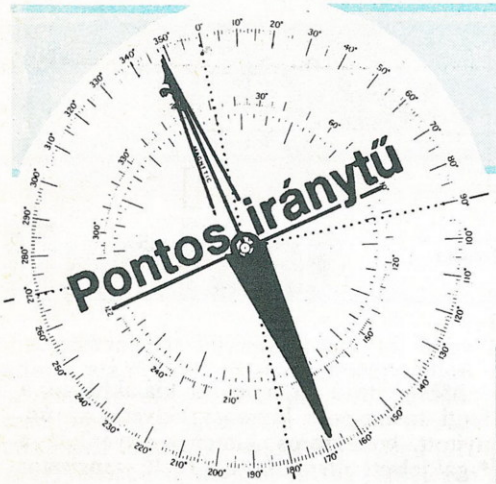
— egy kimenet maximum tíz bemenetet képes működtetni (fan out=10);

— kimenetek közvetlen módon nem köthetők össze;

— a fel nem használt bemeneteket célszerű valamelyik tápfeszültségpontra (GND vagy Vcc=5 V-ra) kötni.

(Folytatjuk)

DR. KÓNYA LÁSZLÓ



Mikrokalauz

Lépéstartás reményeinkkel...?!

Egy régen élt filozófus arra a kérdésre: Elgondolható-e egy olyan világegyetem, melyben nincs élet? nyilvánvalóan azzal válaszolt volna — nem. Hiszen nincs értelme világegyetemről beszélnünk, ha senki sincs, aki beszélne róla. Az ember középponti helyének kopernikuszi trónfosztása óta azonban első hallásra könnyen igennel is válaszolhatnánk. Mégis Einstein egyik neves tanítványa, John A. Wheeler egyértelműen nemet mond. Állítván, hogy nemcsak az ember alkalmazkodott az univerzumhoz, hanem az univerzum is az emberhez. Ez az állítás persze nem vezet minket egyenesen a tyúk vagy a tojás dilemmájához. Ezzel ellentétben még egocentrikusságunk vagy rosszabb esetben egoizmusunk sem mondathatja velünk, hogy létezésünknek nem egyik alapfeltétele alkalmazkodóképességünk. S noha az iménti gondolatfutamhoz képest kissé nyers fordulatnak tűnik, ne vegyék zokon, de az alkalmazkodóképesség nagyon is kifejezhető egy adott ország, jelen esetben hazánk számítástechnikai, pontosabban mikroszámítógépes piacának helyzetével. Mert hiszen, prózaibbra fordítva első kérdésünket: Elgondolható-e egy ország dinamikus fejlődése korszerű vagy legkorszerűbb számítógépek nélkül? — erre egyértelműen nemmel válaszolhatunk.

És noha a Mikrokalauz hazai mikroszámítógépek piacáról szóló tanulmány szerzőit nem társadalmi és gazdasági hatások és behatároltságok elemzése irányítja, a tájékozódó vagy éppen a vásárlásnak nekirugaszkodó olvasó a „képzelet" következtetései elől nem menekülhet. Mert, noha mikroszámítógépes állapotainkra ráhúzni, hogy: non progredi est regredi (nem haladni annyi, mint visszafelé menni), szélsőséges állásfoglalás volna, az már az előbbi szóláson aprót ferdítve prózaian valós, hogy nem eléggé gyorsan haladni annyi, mint erősen lemaradni.

A kötetben a mikroszámítógépeket a mikroprocesszoruk által egyszerre kezelt bitek száma szerint csoportosítva mutatják be a szerzők. Rögtön megállapítva, hogy ma már a kis teljesítményük miatt a 8 bites gépek az életciklusuk lemenő ágában vannak. Ide tartoznak azonban a házi számítógépek és iskolai számítógépek, amelyek nagyobb teret kaptak az összeállításban. És ami azonnal szemet szúr: az alacsony sebességük, kis operatív táruk és korlátozott perifériaellátottságuk miatt a házi számítógépeket általában csak oktatásban, játékra és a háztartásokban használják ma Nyugaton. Hazánkban viszont elsősorban a C64-et széles körben professzionális célokra is. „A fejlett tőkés országokban ezt a teljesítménykategóriát már csak játéknak tekintik. Ugyanis a 16 bites gépek árai is 2-3 havi átlagfizetésre mérséklődtek..." — természetesen jóval többet tudnak kisöccsüknél. És a mai házi számítógép-teljesítmény kategória megkapja a kegyelemdöfést a most megjelenő IBM gyártmányú JX3-tól. Aminek mikroprocesszora Intel 8088, operatív tára 256 kbájt, az alapgép tartalmaz egy 3,5 hüvelykes, 1 Mbájt kapacitású hajlékonylemezes tárolót is. Ára várhatóan 600 dollár körül lesz.

A hazai összetétel? Képet kapunk róla a beszerzési források, típusok és áraik szerint is. Az egyéni utazások révén jutott mindmáig a legtöbb gép az országba, mintegy ötvenezer. A külkereskedelemből 16 ezer, míg idehaza 12 ezret gyártottak. S itt egy adat láttán elgondolkodhatunk, ki és mi a kereskedő: ugyanis 1984-ben mindösszesze ezer gép jutott a külkereskedelem jóvoltából a felhasználók birtokába. Addig egy szál sem!

Dicséretünkre legyen mondva, hogy a házi számítógépek gyártását a szocialista országok közül elsőként Magyarországon kezdték meg 1982-ben, s mind ez ideig csak nálunk gyártottak belőlük ezernél többet. Ebben meghatározó volt a már 1983-ban számítógépet gyártató középiskolai számítógépprogram és a Primo debütálása, majd sorozatgyártása 1984-ben. Óh, Primo! Gyorsan jött, gyorsan hervadt. 1987 februárjában bekövetkezett a végkiárusítása, a Primo gyártói halálát még az előző év szeptemberében bejelentették. Pedig „A Primo harminc hónapos életét értékelve elismeréssel kell szólni a kezdeményezés úttörő jellegéről, a kitűnő piaci munkáról (a szocialista országok első, boltban árusított gépe címet is elnyerte). Az élete rövidségéért okolható az elterjedt nyugati típusokkal való inkompatibilitása, a megbízhatatlan program ki- és bemenete, a mindezek következtében gyenge programellátottsága. Ez utóbbi a programok ezreivel kényeztetett C64 és Spectrum gépekkel szemben óriási hátrányt jelentett. Magyarul: nem lehetett ma már csupasz vasat értékesíteni."

Aztán 1986-ban a Videoton robbantott, robbantgatott. Megjelent a TV-Computerével a boltokban. Bár a gépet 1984-ben bemutatták, végül két esztendőre rá jelent csak meg a forgalomban. Mi lett volna, ha Pallasz Athéné ennyit vajúdik Zeusz fejében! Bizonyára a legfőbb isten agydaganattal kimúlt volna. De hát egy isten még a gyorsaságot is megengedheti magának. — Mindenesetre a Videoton a teljesítménykategória visszaszorulása ellenére igyekszik az igényekkel lépést tartani, amiért is gazdag perifériaválasztékot kínál, illetve ígér.

Az árak, a kereslet-kínálat és a beszerezhetőség hullámzása fokozatosan csökkent. De a „...gyors süllyedés után mára megállapodtak". Így például egy C64 25 ezer, egy ZX—Spectrum 48 k 12 ezer forint körüli áron volt beszerezhető tavaly év végén. Végül az összegzésből kiemelhető, hogy a házi számítógépekből 1986-tól a piacon túlkínálat jelei mutatkoznak, a vevők válogathatnak a típusok között, s míg a házi számítógép-állomány gyarapodási üteme mérséklődött, az iskolaszámítógép-állomány jelentősen megnőtt. Egyébként az iskolaszámítógép-program, ahol dominál a Commodore Plus/4 és a TV-Computer, azzal számol, hogy 1990-ig 50 ezer gépet juttat a közép- és általános iskolák birtokába.

A professzionális mikrogépek gyártása a szocialista országokban a 70-es évek végén, 80-as évek elején kezdődött meg. A 8 bites gépek gyártása gyorsan ívelt felfelé, míg a 16 bitesek megjelenése szinte a törülköző bedobására kényszerítette őket. Úgyhogy a majd két tucat hazai 8 bites gép listája már majdhogynem múzeumi, ha a 86-os forgalmukat nézzük. Az árakat tekintve „Már ma is a 8 bites és a 16 bites gépek árai hasonló konfiguráció esetén összemosódnak, viszont a teljesítménykülönbségük óriási. Ezért is emlegetik a szakmában úgy, hogy a 16 bites gépek árai rázuhantak a 8 bites professzionális gépekére és agyonütötték azokat."

A 16 bitesek közül a meghatározó jelentőségű IBM PC-vel kezdődően a vele kompatibilis mikrogépek a honfoglalók. Itt milyen a hazai állomány? A magánember óvatlanságában igencsak savanyú képet vághat, ha behoz egy gépet. „A vám egy kis konfiguráció esetén is 300-500 ezer forintot tesz ki. A magáncélú vásárlás ezért szinte kizárt." A külkereskedelmi forgalom tankcsapdája viszont a Cocom-rendelkezések sora. Ennek szellemében csak 1985 tavasza óta engedélyezik IBM PC-k kivitelét, illetve behozatalát hazánkba, azóta is csak egyedi exportengedéllyel. 1986 újdonsága volt az IBM PC-vel kompatibilis magyar gépek szocialista exportjának beindulása. Aminek egyik élenjárója a Proper és továbbfejlesztésének eredményeképpen kis családja lett.

Az árak alakulása belsőszervi betegségekre utal. Hasonlattal élve: a vasszerkezet megrogygyan anyaghiba miatt, avagy hamisságokra épülve a lélek is megkeseredik, olyannyira, hogy a cselekedetek is sántikálnak. Magyarán: „Az 1986-os év főszereplője ebben a kategóriában az ár volt. Komoly túlkínálat alakult ki, így akinek jelentősebb készletei voltak akár gyártásból, akár külföldi beszerzésből, forgalma növelése érdekében erőteljesen csökkentette az árait." De ez a tendencia megfordult novemberre. A több ok sűrítménye 1987-re: „A potenciális vevők a kedvezőtlen (sic!) vámintézkedések miatt egyre szűkülő kínálatból válogatás helyett fantomárú fantomgépeket kezdtek üldözni. Közben pedig a tényleges vásárlások száma, az alkalmazásba vétel szeptemberig szinte a nullára apadt"!!! A Mikrokalauz az okok láncolatában mindenesetre kellő alapossággal eligazít. A netán lappadó kedvet talán visszazökkentheti eredeti illúziójába az „Elektronika társadalmi-gazdasági alkalmazása elterjesztésének gazdaságirányítási feladatait átfogó központi Gazdaságfejlesztési Program" keretében kínálkozó kedvezmények eredményeinek és legfőképpen a lehetőségeinek áttekintése. E részben végezetül az IBM PC-vel kompatibilis lokális hálózatokig táblázatokon tekinthetjük át a gyártókat, a termékeket és azok árait. Hozzákapcsolva néhány egyedi típust egy záró megjegyzés kíséretében: „...a hazai gyártók komolyan veszik az IBM PS/2 családját, s erőteljesen folynak az azzal kompatibilis gépek előállítására irányuló munkák." — Amihez nem fér kétség, ámbár nem ártana mindannyiunknak komolyan vennie magát!

A perifériák igazi csemegéi a számítógépeseknek. Annál inkább, mivel a nyomtatók, a winchester-tárak, a folyamatos üzemű mágnesszalagos tárak (streamerek) legkorszerűbb fajtáiból a hazai piacon kevés akad, ha nem éppen újdonság, és az egyszerűbb nyomtatókon kívül borsos az áruk is. Minderről a Mikrokalauz szerzői jó érzékkel tájékoztatnak, és külön kötetben böngészhetünk a teljes választékban. Felvillantva még a lízingnek, amely olyan bérletezést jelent, melynek végén a bérelt eszköz egy jelképes összegért a bérlő tulajdona lesz, remélhetőleg sikeres számítógépes jövőjét is. — Ám számítógépek ide, perifériák oda, *e helyütt* Pascal teljesen reális: „A múlt és a jelen csupán eszközünk: célunk csakis a jövendő. Így tehát sohasem élünk, hanem csak remélünk élni." — A kérdés csupán az: saját reményeinkkel lépést tudunk-e tartani?

SZULYOVSZKY CSABA

Integrált szoftver

A táblázatkezelő program használata II.

Az előzőekben ismertettük a táblázatkezelő utasításkészletét, és néhány egyszerű példával szemléltettük a működést. Egy összetett feladattal most megpróbáljuk illusztrálni a BASIC nyelven megírt, viszonylag egyszerű táblázatkezelő program nyújtotta lehetőségeket.

Mielőtt a táblázat összeállításához hozzáfognánk, a /T utasítással törölhetjük a táblázatot, amelyen eddig gyakoroltunk. Először állítsunk elő egy egyszerű táblát, amelyen egy számsorozat összegét számítjuk ki, majd határozzuk meg az összegre vonatkoztatott százalékokat *(1. ábra)*.

Első lépésként írjuk be a megfelelő helyre a címkéket. Az 1. ábrán látható táblázat kialakításához mozgassuk a kurzort az AA pozícióba. Mielőtt a szövegírást elkezdenénk, nyomjuk le a szóközgombot: ez jelenti a program számára, hogy címke megadása következik. A szöveg (a címke) bevitelét egy ENTER lenyomásával lehet lezárni. Léptessük a kurzort az AB pozícióba, ismét szóköz lenyomása után írjuk be a kívánt szöveget, majd az ENTER ismét lezárja a címkét. A címke szövege „átlóghat" a következő pozícióba is, ha ezt például a táblázat fejléce így kívánja.

Az említett eljárást folytatva írjuk be minden pozícióra, amit akarunk. Az aláhúzások, amelyek a táblázat jó tagoltságát eredményezik, könnyen képezhetők: /- (kötőjel) a program számára azt jelenti, hogy a jobb oldalon, a kötőjel mellett álló karaktert ismételni kell (például -= azt jelenti, hogy a kurzorpozícióval megadott cellához egyenlőségjel kerüljön a cella teljes szélességében, mint címke).

Az értékek beviteli módját már az előző alkalommal ismertettük. Csak emlékeztetőül: először a tizedespont helyét kell megadni (jelen esetben a /TP2 utasítással). Ezek után vigyük be egymás után a megfelelő pozícióra a számértékeket. Az utolsó pozíció a BK lesz. Vigyük most lejjebb a kurzort, az MB pozícióra. Itt most két lehetőségünk van az összeg előállítására. Az egyik +BD +BE +BF ... stb., tehát kijelöljük a cellánkénti összeadást. Mivel teljes összegről van szó, akad egyszerűbb lehetőségünk is:

@ SUMOBD. Nézzük egy kicsit részletesebben.

@ SUM a szummázó utasítás, O az oszlop szerint végzett összegzést jelenti, és BD a kezdő pozíció helye. Az utasítás hatására a BD pozíciótól kezdődően a program az oszlopban található minden értéket összead, és az eredmény megjelenik a kurzor által kijelölt pozíción (BM). A képernyő második sorában az összegző formula látható.

Most végezzük el a százalékszámításokat. Lépjünk jobbra a C oszlopba és mozgassuk a pozíciójelzőt a D sorba. Írjuk be a százalékszámítási formulát. Vegyük a BD mező tartalmát és osszuk el a BM mező tartalmával, majd szorozzuk meg 100-zal. A formula tehát a következő: +BD/BM*100. [ENTER], az eredmény megjelenik a CD mezőben. Hasonlóan írjuk be a többi pozícióra is a formulákat. Az eredmények rendre megjelennek a kijelölt mezőkben.

Lehetőség van egy másik megoldásra a /M (másol) utasítás használatával. Az utasítás a következő alakú lesz: /MCDCK, ami azt jelenti, hogy másold a CD mezőben található formulát a CK mezőig minden pozícióba.

Természetesen most ki kell javítani a százalékértékeket tartalmazó mezők koordinátáit a megfelelőkre. Tehát például az E sorban a BD helyére BE kerül és így tovább. Ez az utasítás hasznos, ha bonyolult formulákkal dolgozunk és csak kis változások szükségesek az egyes pozíciókon.

Miután elvégeztük a szükséges módosításokat a képernyőn, visszatérünk a normál üzemmódba. Ezután, ha kívánjuk, ellenőrizhetők a mezőkön található formulák. Léptessük lefelé a kurzort a C oszlopban. Az aktuális számítási képlet megjelenik a képernyő második sorában. Láthatóan a számítás eredményei is a helyükön vannak, de semmi nem történt a táblázatban. Még nincs kitöltve. Ezt egy egyszerű utasítással érhetjük el: shift gomb lenyomása után le kell nyomni a felkiáltójelet. A program most újraszámolja a táblázatot és kitölti a megfelelő helyeket.

A *2. ábra* egy negyedévi táblázatot mutat be. Láthatóan itt már nemcsak az oszlopok szerint, hanem a sorok szerint is végzünk számításokat. A képernyőn a jobb felső sarokban megjelenő 0: (oszlop) vagy S: (sor) jelzi, hogy a kijelölt műveletet most oszloponként vagy soronként veszi a program. A soronkénti számolás utasítása /SZS, ahol / azt jelenti, hogy utasítás következik, SZ jelentése: számolj, S: soronként.

Ne felejtsük el, hogy egy táblázat számításánál a mezők tartalmának már léteznie kell, mielőtt egy formulával műveletet végeznénk a táblában. Ez különösen fontos akkor, ha az egyik formula használja a másik által kiszámított értékeket.

Ha a nyomtatóra akarjuk küldeni a táblázatot, akkor a számítások elvégzése után (a tábla ki van töltve) a /N (nyomtat) [ENTER] utasítás megadásakor a program megkérdezi a kezdő oszlop helyét. Esetünkben ez az A oszlop volt. A táblázat minden sora nyomtatásra kerül.

Végezetül még néhány szó a fájl szerkezetéről. A táblázatot tartalmazó fájl három rekordból épül fel. Az elsőben található a kurzorpozíció a „kimentés" pillanatában, a táblázatban használt tizedes jegyek száma és a számolás rendje (oszlop vagy sor). Például BE_2S azt jelenti: a kurzor a BE pozíción volt, két tizedesig törekszünk a pontosságra a számolásnál, és a műveleteket soronként végezzük. A harmadik rekord a fájlt zárja le (EOD = End of Date). A közbenső rekord tartalmazza a táblázat adatait. Amint már említettük, 676 mezős a táblázat. Mindegyik pozíción tulajdonképpen három mezőtípus található (címkék, formulák és változók értékei). A rekord a következő formában épül fel: oszlop, sor, típus (... adatok ...).

Az oszlop és a sor megnevezése A-tól Z-ig terjed, a típus L (címke), F (formula), V (változó) lehet. Az adatok sztring formában lesznek rögzítve: a program betöltésükkor az adatokat sztring formában és számra visszaalakítja.

Mivel a rekord tartalmazza minden mező pozícióját és tartalmát, a rekord előállítása például egy másik programmal elég egyszerű feladat. A szükséges dimenzionálásokat a *programlista* 2. sora tartalmazza (lásd '87/12. számunkat).

GALINA FERENC

```
?.
[BM]  (F) @SUMOBD
......A.........B.........C.....
A          Havi kiadások
B Megnev.    összege     Százalék
C-------------------------------
D Lakás        1250.00       10.13
E Közmü.        845.00        6.85
F Hitel        2400.00       19.46
G Ruházat      1338.00       10.84
H élelem       3350.00       27.15
I Közlek       1375.00       11.14
J Egyéb         680.00        5.51
K Takarék      1100.00        8.92
L ===============================
M TOTAL      >12338.00      100.00
```

1. ábra

```
?.
[BM]  (F) @SUMOBD
.....A.........B.........C.........D.........E.........
A                        Havi kiadások
B Megnev.      január      február      március     összesen
C-----------------------------------------------------------
D Lakás        1250.00      1250.00      1250.00      3750.00
E Közmü         845.00      1050.00      1124.00      3019.00
F Hitel        2400.00      2400.00      1100.00      5900.00
G Ruházat      1338.00      1208.00      1500.00      4046.00
H élelem       3350.00      3600.00      3950.00     10900.00
I Közlek       1375.00      1420.00      1390.00      4185.00
J Egyéb         680.00       980.00      1200.00      2860.00
K Takarék      1100.00      1000.00      1000.00      3100.00
L ===========================================================
M TOTAL      >12338.00     12908.00     12514.00     37760.00
```

2. ábra

ADOM A MAGYARÁZATOT!

Lapunk előző számában Gődér György olvasónk a C16 fontosabb alkatrészei lábkiosztásáról kérdezett. A válasz:

1. A C16 mikroprocesszorának kérdezett kivezetései:

AEC: Bemenet. Ha valamilyen külső egység használni akarja a cím- és adatbuszt, akkor az AEC lábát alacsony szintre állítja, és ennek hatására a mikroprocesszor lekapcsolja magát a buszról.

RDY: Bemenet. Ha alacsony szintű, akkor ez jelzi a mikroprocesszornak, hogy az adatbuszon lévő információ még nem érvényes, ezért még nem olvashatja be. A C16-on nincs kihasználva ez a bemenet.

IRQ: Maszkolható megszakításkérés bemenet. Ha a CLI gépi utasítással a megszakításkérés engedélyezve van, és valamilyen külső egység a lábat alacsony szintre állítja, akkor a mikroprocesszor a $FFFE—FFFF memóriacím által mutatott megszakításkezelő rutinra ugrik.

GATE IN: Ezt a jelet a TED állítja elő. A dinamikus memóriákat meghajtó IC-k számára azt jelenti, hogy a cím alsó vagy a felső bájtját továbbítsák a RAM-okhoz. A mikroprocesszornak pedig jelzi, hogy mikor készek a RAM-ok az adatbusz információjának fogadására.

ΦO: A mikroprocesszor (µP) órajele. A TED állítja elő. PO—P7: A µP saját 7 bites porttal rendelkezik, aminek ezek a kivezetései. (P5-ös nincs.) Programból a memória $0000—0001 címein érhetjük el. A $0000 címen a port bitenkénti irányát kell megadni. (1 = kimenet, 0 = bemenet)

A $0001 címen írhatunk, illetve olvashatunk a portról. (A 6510 adatlapján ez is, és a fordítottja is megtalálható. A két változat közül ez az ellenőrzötten helyes.)

A C16-ban ezeket a portbiteket a soros busz és a magnó kezelésére használják.

2. A TED vezérli a C16 összes fontos funkcióját, ami elengedhetetlen a számítógép működéséhez. A TED a C16 legfontosabb alkatrésze.

Feladatai:

— a megfelelő memóriaszelet kiválasztásának vezérlése. CSO, CS1/

— dinamikus RAM-ok felfrissítése

— a rendszerórajel előállítása (ΦØ)

— RAM-ok címzésének vezérlése (CAS, RAS, MUX)

— billentyűzetleolvasás (KØ—K7)

— a teljes képvezérlés: színes videojel összetevőinek előállítása (SYNC, LUM, COLOR)

— hanggenerálás (SND)

— időzítők és rasztermegszakítás kezelése (IRQ).

Kivezetései:

AEC, BA: mint a µP-nál az AEC és RDY kivezetés a TED-re vonatkoztatva.

CSO, CS1: a címbusz alapján a megfelelő ROM, ill. RAM IC-t engedélyezi. Normál (bővítetlen) esetben a BASIC és KERNAL ROM között választ.

COLOR, SYNC/LUM: A kimeneti videojel szín, szinkron- és fényességjel összetevői.

SND: a hanggenerátor AUDIO kimenete.

KO—K7: bemenetek, a billentyűzetmátrix oszlopainak értékét és a botkormányok állapotát olvassa be a billentyűzet letch TED regiszterbe ($FF08).

3. U16-os IC ($7700—10):

Ez egy logikai hálózatot tartalmazó berendezésorientált áramkör, ami a 16 különböző bemenetétől függően változtatja 8 kimenetének állapotát. Egy I/O memóriaterület megcímzésekor a kimenetein engedélyezi a megfelelő egység adatbuszra kapcsolódását.

Pl: kazettás egységen gombnyomás érzékelése,
billentyűzet sorainak vezérlése,
memóriabank-kiválasztás.

(Az I/O terület helye: $FD00—FEFF 16 bájtos részekre osztva.)

4. 6529 B:

Ez egy egyszerű 8 bites regiszter, ami a C16-ban a billentyűzetmátrix sorait hajtja meg. A lenyomott billentyű hatására a TED KØ—K7 bemenetein a megfelelő helyen megjelenik a 6529B IC-nek a lenyomott billentyű sorához tartozó kimenetállapota és a TED $FF08-as regiszterének olvasásával a lenyomott billentyű megállapítható. A 8 bites IC az I/O terület $FF30 címén érhető el.

A C16 számítógép kapcsolási rajza megtalálható a RÁDIÓTECHNIKA 1986. áprilisi számában.

GYURKOVITS GERGELY

ADOK—VESZEK—CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,-Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

ATARI 800 XL (64k) magnóval, joystickkal, játékprogramokkal 16 000 Ft-ért eladó. Kovács László, Alsózsolca, Bajcsy Zs. u. 40. 3571

ATARI 800 XL/XE-hez monitorprogram kapható. Assembler-Disassembler, 12 monitorparancs. Ára: postán utánvéttel 300 Ft/kazetta. Cím: ifj. Nyistár István, Gyula, K.190. 5700

ATARI 800 XL magnóval, játék- és felhasználói programokkal, izzósor-vezérlővel, szakkönyvekkel, ROM-listával, jegyzetekkel eladó. Bódy Attila, Kazincbarcika, Pollack M. u. 3. 3700. Telefon: 06-48-14-643 az esti órákban.

ATARI 520 STM eladó (512k) floppyval (360k), egérrel, sok programmal (68 ezer), hozzá profi ff monitor (20 ezer). Dr. Nagy József, Eger, Dr. Sándor I. u. 3. IV/9. 4139

C64 eladó + magnó, joystick, programokkal + Commodore Mickrodrive, rendszerszoftver nélkül, kazettákkal, ára: 21 000 Ft. Simon Gábor, Miskolc, Gömöri u. 10. 3526. Telefon: 15-379

Commodore VC20 + 16k bővítés + magnó + 3 db cartridge 3 db DEMO-kazettával + sok gyári programmal eladó. Ár megegyezés szerint. Kovács Zsolt, Budapest, Táncsics Mihály u. 63. 1211

VIDEOTON TV-Computer (32 k) + rendszerszoftver 9000 Ft-ért eladó. Balogh Endre, Budapest, Bertóti u. 14. 1172. Tel.: 639-343

TV-Computer (32 k) eladó programokkal (kazettán). Kuczi Géza, Katymár, Sallai u. 7. 6455

VESZEK

ATARI 800 XL-hez keresek adatrögzítő magnót, lehetőleg 1010-es típust. Boros Elemér, Pécs, Vöröskő u. 14. 7634

Plus/4-re beszélő program után érdeklődök. Szabó Tibor, Miskolc, Szentpéteri kapu 815/1. 3526

C64-re Contract bridge programot keresek (lemezen) megvételre vagy cserekként. Dr. Bóka, Várpalota, Kulich u. 25. 8100

TI·99/4A számítógéphez gépkönyvet, kiegészítóket és programokat veszek. Ajánlatokat a következő címre kérem: Simon Árpád, Tatabánya, Sárberk 707. III/1. 2800

CSERÉLEK

C16 programokat cserélek. Ilosvay Zoltán, Budapest, Futórózsa u. 86. 1165

Commodore 64-hez GYORS-HÁTTÉRTÁR cartridge. Kapacitása 2-31 kbájt. A GYORS-HÁTTÉTÁR-ba maximum 7 db, célszerűen gyakran használt program vihető be. A gép bekapcsolása után menüvel jelentkezik be, és gombnyomásra a kiválasztott program azonnal fut. Javasolt programcsomagok: Turbo tape, Assembler, Monitor, Supergrafik, Help plus, Turbo másoló. File másoló 1699 Ft és Turbo tape 699 Ft. Hozott programok elhelyezése a GYORS-HÁTTÉTÁR-ban. Trompler László, Budapest, Attila u. 22. 1201. Tel.: 287-493 este.

Enterprise programokat cserélek. Listát kérek! Veresegyház, Pf.: 28. 2112

C64-es játékprogramokat cserélek kazettán és lemezen. Keresem SIDEWIZE és a FREDDY HARDEST nevű játékokat. Válaszokat lehetőleg listával kérem: Vas László, Szeged, Tarján, Pentelei sor 1. IX/45. 6723

Primo programcsere! Több mint 150 program! Minden gépi kódú programhoz örökéletet is adok, a küldött programokhoz elkészítem. Listát kérek' Répás Sándor, Budapest, Fegyvernek u. 10. VI. 61. 1116

Ez a rovatunk **KODEX 2000** szövegszerkesztővel készült.

Építsünk számítógépet?...

Vita si uti, scias longa est — azaz: az élet hosszú, ha élni tudsz vele — mondta Seneca, s e szemléletnek azóta is vannak követői. Mindenesetre eme örökbecsű állítást bátran zászlajukra tűzhetnék — ha lenne — klubsétám következő állomásának szereplői.

A fővárosban, az V. kerület, Molnár utca 9. szám alatt levő keskeny, négyemeletes épület a tárgyilagos Belvárosi Művelődési Ház nevet viseli. Itt lelt otthonra kedd délutánonként a Homelab számítástechnikai klub.

Beismerem, enyhe rezignáltsággal készültem az ismerkedésre. Hiszen eddigi klublátogatásaimon úgy tűnt, létezik valamiféle láthatatlan ördögi kör vagy inkább labirintus, melyből képtelen kikászálódni manapság egy-egy hazai számítógépes klub. Mert hiába működik önszerveződő módon, lelkes tagokkal, a „háttérből" valami titokzatos erő a megsemmisülésbe taszítja. Előbb vagy utóbb elsekélyesednek a foglalkozások, és ilyenkor már nem sok érv szól a klubban maradás mellett. Egyeseket a helyszűke, másokat a helyhiány, megint másokat a legcsekélyebb támogatást is nélkülöző szegényes klubmunka dönt az önfelszámolásba. Manapság — hobbi szinten is — lehetetlenség hosszú távon dolgozni megújhodás nélkül, ugyanis a programok gépies ismétlődése egy idő után lehangolja, megfakítja a kezdeti lelkesedést, végül kikoptatja a tagokat. És még hosszan lehetne sorolni... Miért vártam volna mást így ismeretlenül a Homelab klubtól? Náluk miért ne a fentiekhez lenne hasonló a gyakorlat?

Mindenesetre felkészültem a látványra: hordozható tévék, magnók sora, kábeltekervények, számítógépek hada fogad. No, meg természetesen a másolandó kazetták elmaradhatatlan halmaza. A háttérben zsibongás, a szűk helyen tolong a tömeg.

Hm. De furcsa! Talán rossz helyre kopogtattam? Netán az emeletet tévesztettem el? Nem, az lehetetlen! Hoppá, megvan! A klub összejövetele bizonyára elmaradt, s most másik baráti társaság foglalja el a termet. De hát mégiscsak érdekes?! Ott távol, a sarokban, közvetlenül a beszélgetők mellett miért világít a tévéképernyő? És miért ül oly elmélyülten, számítógépe fölé görnyedve az a férfi? Aztán meg hogy-hogy csak az az egy? A többiek pedig mintha valamiféle állófogadáson lennének. Vagy mondjuk az egyetem folyosóján a vizsgák időszakában a le és fel masírozó drukkoló vizsgázók hada. Itt — lehetnek vagy harmincan — szintén álldogálnak vagy párosával, hármasával sétálnak, közben pedig hatalmas taglejtésekkel magyaráznak. A mozdulatok a beszélő közlendőjét nyomatékosítják. Az asztalok mellett ülő párokat, csoportokat is szemügyre veszem. Csendesnek, elmélyültnek látszanak, néhányuknál cserélődnek a beszélgetőpartnerek.

Barabási Rezső — akiről bemutatkozásunkkor megtudom, hogy a Homelab klub egyik irányítója, vezetője — tanácstalanságomat látva, a segítségemre siet. Megerősít, hogy jó helyen járok. Ajánlja, üljünk le, s mindjárt halkabbra is fogja a háttérben szóló diszkózenét. Faggatásomra belekezd a klubjuk történetébe. Íme.

— Születésünket 1983 nyarára tesszük. A többi klubhoz hasonlóan „anyánk" az NJSZT égisze alatt működő HCC klub. A XI. kerületi Bocskay úton lévő TIT szervezésében tartott összejöveteleken ismerkedtünk meg egymással. Ha nem is tudatosan, de az idő tájt éltük a hazai mikroszámítógépek hőskorát, természetesen mi gépépítőként. Az anyaklubból helyhiány miatt elsőként a Sinclair klub költözött el, majd néhány hónappal később mi távoztunk. Kapcsolatok révén az Eötvös Loránd Tudományegyetem, pontosabban az Eötvös Klub fogadott be bennünket, s így ez lett egy időre békés, biztonságos találkozóhelyünk. Olyan, ahol nyugodtan építhettük hétről hétre Homelab típusú személyi számítógépeinket, s adhattuk át egymásnak frissen szerzett számítástechnikai ismereteinket. Erről az időszakról szólva feltétlenül emlékezetünkbe kell idézni, hogy noha Nyugaton már terjedőben voltak a ma általánosan használt személyi számítógépek — mint például a Commodore-ok —, de áraik egy magyar zsebnek szinte megfizethetetlennek számítottak. Mit tehettünk mi, akik hallomásból, szakirodalomból, tanulás révén ismerkedtünk meg e számunkra új technikával, ugyanakkor pénztárcánk nem tette lehetővé, hogy hozzá is férkőzzünk? Gépet építettünk. Természetesen ez sem volt filléres szórakozás, de már sokaknak megengedhető kiadás és hasznos kikapcsolódás.

A hazai számítástechnika történetében méltán jegyzett Lukács fivérek dolgozták ki a Homelab személyi számítógépet. Ezt a típust azután nem fejlesztették tovább. Mi viszont alkatrészekből, részegységekből megépítettük magunknak a gépet. Micsoda nagy szó volt! A ZX típusú — számunkra elérhetetlen — gépek „konkurenciájaként" foglalkozhattunk magyar számítógéppel! És egy cseppet sem zavart bennünket, hogy ezek gyakran egyáltalán nem voltak kompatibilisek egymással, hogy használható programok sem íródtak hozzájuk.

Rendületlenül raktuk össze a Homelab 2-es, majd a 3-as, 4-es, 5-ös típusokat. Mindezt közös erővel, közös akarattal. Ma már sokaknak felfoghatatlan, hogy akkoriban nekünk ez mekkora örömet jelentett. (A rádiózás hőskora hasonlítható ezzel. — A szerk.)

A klub akkori átlagéletkora? Az egészen ifjak még nálunk is kevesebbet vagy egyáltalán semmit sem tudtak a számítástechnikáról. Így maradtak az érettségiző korúak és a felnőtt klubtársak. Elméletet akartak tanulni ezeken az összejöveteleken, és gyakorlati ismereteiket bővítették. A tudást, az információt nem sajnáltuk egymástól. Nézzen csak körbe! Ennek a korszaknak a lendülete máig eltartott. Az álldogáló, üldögélő „szakik" híreket, adatokat, szakmai fortélyokat cserélnek. Sokan egy egész estét nyugodtan rászánnak — ha kell, mert úgy adódik — pusztán egyetlen kérdésükre, hogy választ kapjanak: kivárják, hogy a másik, netán este 9-re, de betoppanjon, s az egy héttel korábban felvetett témára meghozza a feleletet.

Visszatérve történetünkre, a hobbikategóriában egészen jó kis gép volt a Homelab. Például 320 × 200 pontos grafikus lehetőséggel működött. Félbemaradt fejlesztéséért kár, hiszen impozáns volt a teljesítménye. Mi mindenesetre sokat tanultunk építése közben. Olyannyira így igaz, hogy sokan lettek közülünk az évek során profik, számítógépes szakemberek. Az alapokat itt szerezték, majd egyetemre mentek, programozók, mérnökök, vagy éppen rendszerszervezők lettek.

Időközben átkerültünk a Molnár utcába. És hát értelmét vesztette a gépesítés is, hiszen ma már elérhető áron, csaknem minden harmadik üzletben vásárolhatók különféle mikroszámítógépek. Mi maradt hát az eredeti célkitűzésből, abból, amire szerveződtünk? A megszerzett tudás és a számítástechnikát szerető baráti társaság. Igaz, sokan kiváltak közülünk: a család, a hétköznapok hajtása, egy új géptípus vonzása stb. távol tart számos egykori klubtagot. De megértjük őket, mert tudjuk, hogy sokuk magas színvonalon műveli a számítástechnikát, meg azt is, hogy szeretettel gondolnak a velünk töltött időszakra. És ez jó. Ha úgy érezték, váltani, módosítani kell, miért ne tették volna...?

Váltani!? Ezt megmaradt közösségünk is felismerte. Lassacskán mi, a hajdanvolt gépépítő klub, CP/M felhasználói körré alakultunk át. Sőt ezzel párhuzamosan MS DOS felhasználói körré is. A technika fejlődésével a Homelab gép emléktárgyként a vitrinbe került. Nosztalgiából azonban nem élhetünk meg! Feltett szándékunk szerint az eredeti névvel ugyan, de egészen más tartalommal kívánunk továbblépni. A világ körülöttünk változik, fejlődik, gazdagodik. Képtelenség lenne egy leírt, kinőtt gép emlékével hosszú távon együtt maradnunk. Programok írásáért, másolásáért sem kell már összejönnünk, tudunk annyit, hogy ezzel egyedül, otthon elboldoguljunk. Az együtt gondolkodás viszont kovácsoló, megtartó erő a holnapra is.

Megköszöntem, elköszöntem, s távoztamban elhatároztam: a ma kínlódó, vegetáló kluboknak tanulságként rögzítem a Homelab klub programját, lépéstartását a változásokkal, a fejlődéssel. Hátha mások okulva, tanulva, szintén képesek lesznek váltani...

KRASZNAI ÉVA

Hogyan kerül kutya a számítógépbe?

Lassan mi is eljutunk odáig, hogy a mikroszámítógépeket nemcsak játék- vagy ügyviteli stb. programok futtatására, hanem például folyamatirányításra használjuk. Az egyszerű személyi számítógépek egyre csökkenő ára lehetővé teszi, hogy a kicsit bonyolultabb vezérlési feladatokat számítógép segítségével oldjuk meg. Ipari környezetben erre a célra ún. PLC-ket (programozható logikai vezérlőket) alkalmaznak. Ezek tulajdonképpen olyan mikroszámítógépek, amelyek néhány dologban különböznek a szokásostól. Jellemzőjük például a rövid válaszidő miatt többszintű speciális megszakításrendszer, az igen sok ki- és bemenet, a magas zavarvédettség speciális hardver- és szoftvermegoldásokkal, az ipari környezethez illő tokozás és az, hogy különleges programnyelvük, utasításkészletük van.

A fent leírtak ellenére azonban az otthoni két fóliasátor fűtési és öntözési rendszerének automatizálásához nem feltétlenül kell egy Festo PLC, megfelelhet akár egy alapkiépítésű ZX81 is. Természetesen ebben az esetben illik gondoskodni megfelelő ki- és bemenetekről, valamint arról, hogy egy pillanatnyi hálózatkimaradás miatt ne váljon rizsföld az előkészített paprikaültetvényből. Profi rendszerekben erre a célra a

szünetmentes áramforráson kívül például a „házőrző kutya"áramkör szolgál.

Az elv nagyon egyszerű: időnként ellenőrizni kell, hogy a vezérlőprogram rendben lefut-e. Az ipari alkalmazásokra szánt rendszerekben ez nem probléma, mert a PLC-ben hardver úton biztosítható az időszakos ellenőrzés. Annál nagyobb gond viszont a kis személyi számítógépeknél, mert ezekben nehéz utólag a hardvert módosítani, és különben sem biztos, hogy ezután már csak például locsolásra akarjuk használni a Commodore-unkat. A tévesztés veszélye is nagyobb, mert az összes adatot és a programot is RAM-ban tároljuk. A tervezésnél pedig nem gondoltak arra, hogy „összedőlhet a világ", ha elvét valamit a gép.

— A 0 perifériacímen egy hőkapcsoló van, amely a beállítottnál alacsonyabb hőfok esetén 0-t ad, különben pedig 1-et.

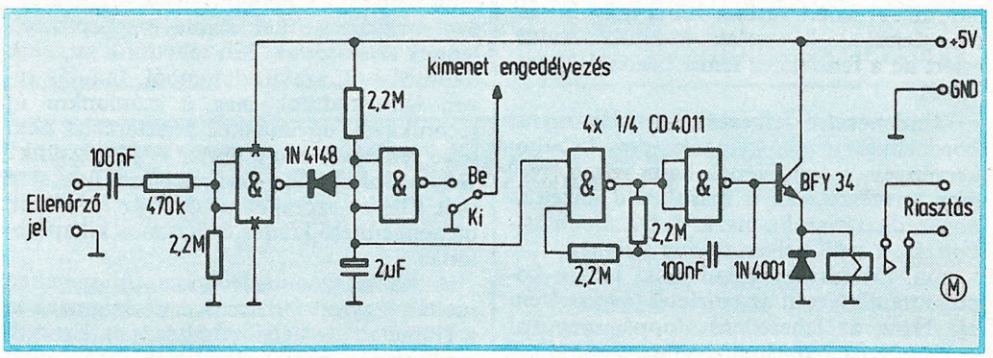
— A 1-es perifériacímen van a fűtőtest, amit 1-gyel lehet bekapcsolni, 0-val pedig kikapcsolni.

— Az ellenőrző kimenet a 3-as perifériacímen érhető el.

A vezérlőprogram szintén nagyon egyszerű (lásd a *listát*). Indításkor az alapállapotba állítást az első három sor végzi el. A tulajdonképpeni vezérlést az 50-es és a 70-es sorok képezik. Az ellenőrző jeleket a 40-es és a 80-as sorokban állítjuk elő. A 120-as szubrutin a szabályozás kézi ki- és bekapcsolását végzi.

Az ellenőrző áramkör az *ábrán* látható. A feladata mindössze annyi, hogy az ellenőrző jelek megszűnése esetén szaggatott riasztó jelzést adjon (például csengő), és a vezérlőkimenetek számára engedélyező feszültséget szolgáltasson, ha az ellenőrző jelek rendben érkeznek.

A figyelmes olvasónak bizonyára feltűnt, hogy a fenti példában szereplő szabályozást úgy is meg lehetett volna oldani, hogy



Gondolatébresztőként a továbbiakban bemutatok egy, az ellenőrzésre alkalmas áramkört, valamint egy primitív szabályozási példát a vezérlőprogramjával.

Kiválasztunk egy kimenetet, ez lehet akár a másra nem használt RS232 kimenet is, és kinevezzük ellenőrző kimenetnek. Ha a vezérlőprogram minden lényeges része rendben lefutott, kiadunk az ellenőrző kimenetre egy impulzust. Ha az ellenőrző kimenetre rendben érkeznek az impulzusok, akkor a vezérlőprogram valószínűleg hiba nélkül fut. Ha nem érkeznek meg az impulzusok, akkor a vezérlés biztosan nem működik rendesen. Ekkor riasztójelzést kell adni, és le kell tiltani a kimeneteket, nehogy a kicsi bajból nagyobb legyen. A mintapélda legyen egy hőfokszabályozás, amely szoftverszempontból a következőképpen néz ki.

```
  5 REM HŐFOKSZABÁLYOZÓ
 10 OUT(1,0)
 20 B=0
 30 ERASE
 40 OUT(3,0)
 50 A=IN(0)
 60 IF INKEY$<>"" THEN GOSUB 120
 70 IF B=1 AND A=0 THEN OUT (1,1)
    ELSE OUT (1,0)
 80 OUT(3,1)
 90 ERASE
100 IF B=1 THEN PRINT "SZABÁLYOZÁS
    BEKAPCSOLVA" ELSE PRINT
    "SZABÁLYOZÁS KIKAPCSOLVA"
110 GOTO 40
120 IF INKEY$="B" THEN B=1
130 IF INKEY$="K" THEN B=0
140 RETURN
```

a hőkapcsolót egy inverteren keresztül egyszerűen összekötjük a fűtőtesttel. Ez igaz, viszont ha például a fűtőtest bekapcsolása előtt 20 másodperccel el kell indítani, kikapcsolás után pedig még 10 percig járatni kell egy ventillátort, vagy esetleg éjszaka más hőmérsékletet kell biztosítani mint nappal, vagy még az üzemidőket is számolni kell (például energiafogyasztás nyilvántartása), akkor már sokkal indokoltabb a mikrogép alkalmazása. (A rugalmas programozhatóságot még nem is vettük az előnyök közé!) Különben sem az adott példa ismertetése volt a célom, hanem egy, a gyakorlatban előbukkanó feladat megoldási elvének bemutatása. Természetesen azt sem titkolhatom, hogy egy kis kedvet szeretnék csinálni a személyi számítógépek szokványostól eltérő alkalmazásához.

NAGY GÁBOR

A Pascal továbbfejlesztése

„Számítástechnikai gondolatainkat" valamilyen programozási nyelven írjuk le. Egyes nyelvek alkalmasabbak bizonyos elképzelések megvalósítására: van például *lebegőpontos aritmetikájuk* — mely a tudományos számításokhoz nélkülözhetetlen —, másoknak meg nincs. A BASIC és a FORTRAN nem támogatja a szövegfeldolgozáshoz és a rendszerprogramozáshoz oly szükséges *ún. rekurzív algoritmusokat.* A Pascalban nehéz a nagy programok írása, mert nincsenek eszközei a probléma (a megoldás) jó *szeleteléséhez:* nem lehet vele (benne) igazán jól szétdarabolni a feladatokat (a feladatot megoldó programokat).

Természetesen számolnunk kell már a Modula—2 kritikusaival is. Mondják: zsákutca, hiányzik belőle is egy sor fontos képesség, amely például versenytársában, a „mindent tudó" Ada nyelvben megvan. Nincs értelme a döntőbíráskodásnak: a Modula—2 arra biztosan jó, hogy néhány új elvet, megoldást bemutassunk a segítségével. A többit bízzuk az időre.

Lokális modulok és a láthatóság

Már említettük, hogy a nyelv a *modulok* három típusát különbözteti meg:

— az ún. *programmodulokat* (mondjuk: „fő programokat")

— az ún. *globális modulokat,* amelyek két részből — egy ún. *definíciós* és egy ún. *implementációs* részből — állnak (ezek segítségével hozzuk létre az „előregyártott szerelvényeket"), és

— a most kicsit részletesebben bemutatandó, ún. *lokális modulokat.*

A modulok — leegyszerűsítve — adatok és procedúrák gyűjteményei. A progammodulok és globális modulok önálló egységek, a lokális modulok mindig be vannak ágyazva egy másik modulba. Szövegkörnyezetükben éppen arról ismerhetjük fel a lokális modulokat, hogy a „MODULE" kulcsszó előtt nem áll a globális modulokat megkülönböztető „DEFINITION" vagy „IMPLEMENTATION" kulcsszó, valamint hogy a „MODULE" kulcsszó önmagában nem áll önálló egység elején — mert akkor „fő program" (egy „fő program") kezdetét jelentené. (A továbbiak előtt az *1. ábrán* összefoglaltuk a modulokat.)

A lokális modulok alkalmazásának fő célja a program *szervezettségének* javítása: a modul belsejében új, a külsőtől elválasztott *környezet* létrehozása. A modulon kívülről *csak* azok az objektumok (változók, procedúrák, típusok stb.) „láthatók", hozzáférhetők, amelyeket a modul belsejéből *„exportálunk"*, illetve — és talán ez a nagyobb újdonság — a modul belsejéből *csak* azok a külső objektumok „láthatók", hozzáférhetők, amelyeket a modul belsejébe *„importálunk"*. A lokális modulok segítségével tehát például a Pascalban megismert „láthatósági" szabályoknál (ezek az ún. blokkstruktúrán alapulnak) finomabban tudjuk szabályozni a „láthatóságot" (elérhetőséget).

Lássunk egy példát *(2. ábra).* Vegyük észre, hogy a *„Példa"* nevű modul egy fő program, míg az *„Elrejt"* nevű „csak" egy lokális modul. A példában elrejtettük a változódeklarációk egy részét. Ha nem ezt tettük volna, akkor a compiler a *q* és az *r* nevű változók kétszeri deklarációja miatt nem is fogadná el a programot. Ha továbbfejlesztett példánkat *(3. ábra)* lefuttatnánk, akkor az alábbi szöveg jelenne meg a képernyőnkön:

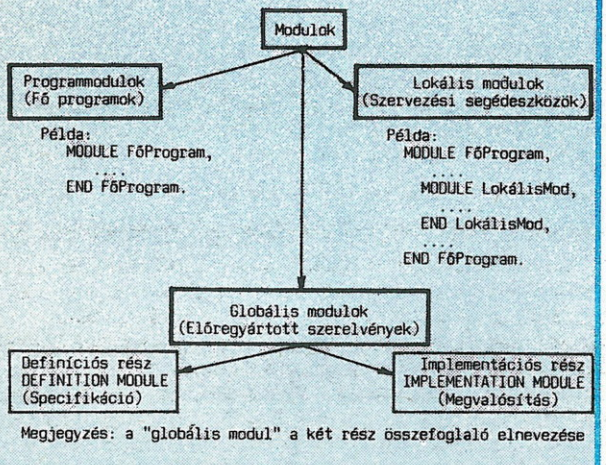
```
Ezek itt q,r,s értékei
az Elrejt modulból 10 30 50
Ezek meg itt p, q és r értékei
a fő programból:   2  4  6
```

Új példánkban az „Elrejt" nevű lokális modulból csak két *procedúrát* exportáltunk: a „Kezdőérték" és a „Kiír" nevűt — a modulon belüli (az ott deklarált) változókat nem. A lokális modulon belülre sem importáltuk a fő program (Modula—2 „tájszólásban" az ún. programmodul) három változóját: p-t, q-t és r-t, csak az InOut globális modulból három procedúrát: a WriteString nevűt, a WriteInt nevűt és a WriteLn nevűt. Eredményként a lokális modul belső elrejtett változóihoz csak az exportált két procedúrán keresztül férhet hozzá a fő program, továbbá nincs „névütközés" a két helyen is deklarált „q" és „r" nevű változók között (lásd a képernyőre kiírt szöveget!).

Általános célú globális modulok

A WriteString, a WriteInt és a WriteLn procedúrákkal kapcsolatban — ezeket mind a fő programba, mind a lokális modulunk-

1. ábra. A modulok fajtái a Modula—2 nyelvekben



2. ábra

```
MODULE Példa;
   (*Fő program*)
VAR p,q,r:INTEGER;
MODULE Elrejt;
(*Lokális mod. kezdete*)
VAR q,r,s:INTEGER;
(*Innen, a lokális modul
   belsejéből nem "látni"
   a külső p,q,r változókat*)
END Elrejt;  (*Lokális modul vége*)
BEGIN
(*Itt, a fő program testében
   csak a külső p,q,r "látható"*)
END Példa.
```

3. ábra

```
MODULE Újpélda;  (*A fő program kezdete*)
FROM InOut IMPORT
          WriteString, WriteInt, WriteLn;
MODULE Elrejt; (*A lokális mod. eleje*)
IMPORT WriteString, WriteLn, WriteInt;
EXPORT Kezdőérték, Kiír;
   VAR q, r, s: INTEGER;
   PROCEDURE Kezdőérték;
   BEGIN
     q:= 10; r:= 30; s:= 50
   END Kezdőérték;
   PROCEDURE Kiír;
   BEGIN
     WriteString ('Ezek itt q,r,s értékei');
     WriteLn; WriteString ('Az Elrejt modulból:');
     WriteInt(q, 4);
     WriteInt(r, 4);
     WriteInt(s, 4);
     WriteLn
   END Kiír;
END Elrejt; (*A lokális modul vége*)
VAR p, q, r: INTEGER;
(*A fő program "nyilvános" változói*)
   BEGIN  (*A fő program teste*)
   p:= 2; q:= 4; r:= 6;
   Kezdőérték; (*A exportált proc. hívása*)
   Kiír;  (*Mint az előbb*)
   WriteString ('Ezek meg itt p,q és r értékei');
   WriteLn; WriteString ('A fő programból:');
   WriteInt(p, 4);
   WriteInt(q, 4);
   WriteInt(r, 4);
   WriteLn
END Újpélda. (*A fő program vége*)
```

és az előző számban tett ígéretünknek megfelelően visszatérünk a Modula—2 elnevezésű programozási nyelvben rejlő — valószínűleg sokak számára még újszerű — lehetőségek bemutatásához **Kaare Christian** *„A Guide to Modula—2" című könyve alapján. (Springer Verlag, 1986. 436 oldal.)*

ba importáltuk — elmondjuk, hogy minden Modula—2-es rendszer kiegészítésként tartalmaz egy sor igen hasznos globális modult, amely „együtt jár" a fordítóval. E készlet (az összetétel, a funkciók, az elnevezések) kis mértékben függ a konkrét szállítótól, az eladótól (végső soron az adott rendszer fejlesztőjétől). A fenti három procedúra például a Modula—2 nyelvi rendszerrel együtt szállított „InOut" nevű globális modulból való, mely a gép és a külvilág közötti kommunikációhoz szükséges procedúrákat exportálja. A leggyakrabban használt matematikai függvényeket például tipikusan a „MathLibO" nevű globális modul tartalmazza. Ezek a rendszerrel együtt szállított (angolul: common) globális modulok szorosan véve nem a Modula—2 nyelv részei — ellentétben elődjével, a Pascal nyelvvel. A Modula—2 tervezői ugyanis — élve a globális modulokban rejlő lehetőséggel — „kirakták" a rendszer konkrét géptípustól függő részeit globális modulokba.Vásárlás előtt célszerű tehát megnézni, milyen globális modulok „járnak" a rendszerrel.

Lokális modulok — statikus változók

A Modula—2 konstans és változó adatokat ismer. A konstansok neve olyan érték helyett áll (olyan értéket képvisel), amely a program futása alatt nem változik. A változó értéke pedig futási időben ún. értékadó utasítással változtatható.

A változókat már néztük abból a szempontból, hogy honnan láthatók: láttuk azt is, hogyan befolyásolják a lokális modulok a láthatóságot. Egy másik szempont szerint a változók statikusak vagy dinamikusak aszerint, hogy akkor „keletkeznek-e", amikor a gép azt a procedúrát kezdi el végrehajtani, melyben a kérdéses változót deklarálták (illetve akkor „szűnnek-e meg létezni", amikor e procedúrának vége), vagy a változó „él-e" a procedúrán (annak futási idején) kívül is.

A változók élettartama nagyon fontos a procedúrák egymás közötti kommunikációja miatt is. A Modula—2 moduljai a változók élettartama szempontjából másképpen viselkednek, mint a procedúrák. A modulokon belüli változók akkor keletkeznek, amikor a gép a modul környezetének (angolul az ún. *surrounding scope*-nak) végrehajtásába kezd. A lokális modulok környezete ebből a szempontból maga a program (a programmodul vagy a globális modul). Emiatt mondják, hogy a modulok változói — ellentétben a procedúrák dinamikusnak mondott változóival — statikusak.

Most egy vázlatos példán *(4. ábra)* megkíséreljük érzékeltetni, hogyan lehet a statikus változókat kihasználni. Kezdjük a fenti programban (a kommentárban) tett ígéret beváltásával: a lokális moduloknak is lehet „testük", azaz a deklarációs rész után BEGIN ... END közé fogva tartalmazhatnak egy utasítássorozatot, amelyet a gép még *azelőtt* végrehajt, mielőtt a lokális modult körülvevő modul utasításainak végrehajtását elkezdené. Ha több lokális modul van egy modulba ágyazva, akkor ezek testében foglalt utasítások a modulok sorrendjében lesznek végrehajtva (5. ábra). A „StatVált" programmodul *„először"* nevű, Boolean típusú változója olyan, hogy a program egész futási ideje alatt „él". Kezdeti értékét még a fő program törzsének futása előtt a „Statikusok" nevű lokális modul teste állítja be. Mivel az „először" nevű változót sem exportáltuk a lokális modulból, az a „statikusok"-on kívül nem látható, nem változtatható. A láthatóság fenti kezelhetősége a lokális modulok egyik legfőbb haszna.

A lokális modulok *törzseinek* végrehajtási sorrendjét több lokális modul esetén az *5. ábra* példájával szeretnénk megvilágítani. Ha a programot lefuttatjuk, az alábbi szöveg jelenik meg a képernyőn:

```
Itt az "A" modul
Itt a "B" modul
Itt a "sorrend"
```

Vegyük észre, hogy a fő program törzsében explicit módon *nem* hívtuk a lokális modulokat — törzsüket a gép *automatikusan*, a deklaráció sorrendjében hajtotta végre, még mielőtt a fő program utasításaira ráfutott volna. Ügyeljünk arra is, hogy egy-egy lokális modul ne vegye igénybe a sorrendben utána deklarált másik lokális modul szolgáltatásait.

A lokális modulok törzse kicsit olyan, mint egy igazi (paraméterezetlen) procedúra, csakhogy törzsüket a gép automatikusan, explicit hívás nélkül hajtja végre a fent bemutatottak szerint.

Befejezésül most is bátorítjuk olvasóinkat, hogy ne riadjanak vissza az első nehézségek láttán, „szoktassák a szemüket-agyukat az újhoz". Más alkalommal, más könyvek kapcsán igyekezni fogunk még visszatérni a témára.

— KE —

4. ábra

```
MODULE StatVált;  (*Fő program*)
MODULE Statikusok; (*Lokális kezd*)
EXPORT Fn;
VAR
(*Ide jönnek a "Statikusok" nevű lokális modul
   privát, azaz nem exportált statikus változói*)
először : BOOLEAN;
PROCEDURE Fn;  (*Az exportált proc. *)
VAR
(*Ide jönnek az "Fn" nevű procedúra dinamikus,
   privát változói*)
BEGIN
IF először THEN
   először := FALSE;
   ...
ELSE
   ...
END
END Fn;
BEGIN (*A "Statikusok"  nevű lokális modul testének
        kezdete. Ez itt most valami új, magyarázat
        következik! *)
először:= TRUE
END Statikusok; (*a "Statikusok" modul teste*)
BEGIN
... (*A "StatVált" program teste*)
END StatVált.
```

5. ábra

```
MODULE sorrend;
FROM InOut IMPORT
     WriteString, WriteLn;
MODULE A;
IMPORT WriteString; WriteLn;
BEGIN (*Itt van az "A" modultörzs*)
  WriteString('Itt az "A" modul');
  WriteLn;
END A;
MODULE B;
IMPORT WriteString; WriteLn;
BEGIN
  WriteString('Itt a "B" modul');
  WriteLn
END B;
BEGIN  (*A fő program törzse*)
  WriteString('Itt a "sorrend" ');
  WriteLn
END sorrend.
```

Ren napokig elhanyagolta a Gépet. Most, hogy megválasztották vezetőnek, azt sem tudta, hol áll a feje. Mégis szakított időt, szakítani kellett, hogy elmenjen a Géphez. Valami belső késztetést érzett. Magányossá vált. A gondolatait csak napok múlva értették meg a polgárok. Az úton nem járt senki. Csak ő kereste fel az égből lepottyant kockát, de már egyre ritkábban.

Ren hallgatott. A kontroll-lámpák fénye furcsa képeket rajzolt a kocka falára. Renben még elevenen éltek a Gép mondatai, melyeket önkívületben sorolt. Az illemszabályok. Ezért hallgatott, várva, hogy a Gép törje meg a csendet.

— Három nap! Három földi nap! — szólalt meg hirtelen a Gép.

Ren összerezzent.

— Parancsolsz?

— A Mester halálának napja lesz.

— Miért kell emlékezni a halottakra? Nem elég emlékezés, ha felhasználjuk mindazt, amit alkottak?

— A nagy emberek képét meg kell őriznünk a lelkünkben, tisztelnünk kell azt, hogy éltek!

— Az emberre emlékezni felesleges, és nem is tudsz! Csak azt idézheted fel magadban, amit alkotott! Nem azzal emlékezünk a legjobban, ha használjuk az általa feltárt tudást?

— Nem — felelte nagyon gyorsan a Gép.

— Miért? — kérdezte Ren.

— Mert a nagy emberek képét meg kell őriznünk a lelkünkben, tisztelnünk kell azt, hogy éltek!

Ren a sétabotjával megbökte a cilinderét. — Neked nincs is lelked! — mondta, és gyorsan, még a Gép válasza előtt kilépett a kockából.

Már sötétedett. Először volt jobb, először jött el úgy a Géptől, hogy ő oktatta ki. Valami mérhetetlen ellenszenvet érzett a Gép iránt abban a pillanatban, de már megbánta.

A csillagok előbújtak a felhők mögül. Ren megállt és bámulta az éjszakai égen világító fénypontokat. Egyre erősebben bánta a kockában előadott produkcióját. Hirtelen elhatározással elindult visszafelé, hogy elsimítsa a dolgot. A kocka közelébe érve, eddig még nem hallott zajra lett figyelmes.

Egy guruló szerkezet érte el a kockát. A taszító érzés egyre erősebbé vált. Ren minden gondolata menekülés volt. A járműből egy tojás alakú tárgy gurult elő, majd a kocka bejárata felé haladt. Ren nem bírta tovább. Minden erejét összeszedve a város felé iszkolt, vissza sem nézve.

A Gép is érzékelte a közeledő tárgyat. A tojás begurult a kockába. A Gép aktivizálta a belső védelmi készültségét. A tojás beszélni kezdett hozzá Ren nyelvén. Fény gyulladt a belsejében, és a Gép azonosította a benne tartózkodó lényt. A robot által hozott első képeken látott hasonló lényt. Henger alakú törzs, három csáp. Egy víz alatti lény állt előtte, az ellenség!

Ren, amikor beért a városba, egyenesen az újonnan elkerített helyére ment. Testének remegése lassan csillapodott. Megrémült az állapotától. Újból és újból beléhasított a hang, amit a kockánál észlelt. És a félelem, a menekülés. — Miért? — kérdezte magától talán századszor. — Miért?

A Gép minden érzékelőjét az idegenre szegezte. — Idegen — állapította meg, de csak a Mester érezhette volna meg e szó furcsaságát.

A tojásban kialudt a fény, és az alak beleveszett a szkafanderszerű védőburok sötétjébe. A Gép megpróbálta elemezni a testet, de a tojásburok minden behatolási kísérletet meghiúsított.

— Én — kezdte az idegen Ren nyelvén, amikor már kínossá vált számára a csend — az élt civilizációt képviselem . . .

— Tudom — vágott a szavába a Gép.

— Én egy élőlény vagyok — próbálkozott az idegen ismét — egy túlélőlény. Ostoba szójáték, tudom, de így van. Egyedül maradtam. A túlélők sorban meghaltak, mint mindig. Háborúztunk logikátlanul, értelmetlenül. Vízióként lebegő célokért küzdöttünk, és elbuktunk. Az egész következményeként meghaltunk, azaz meghaltak.

— Mikor a győző megtudja, hogy mit nyert, akkor válik legyőzhetővé!

— Bölcs vagy — mondta az idegen.

— Bölcs? — kérdezte a Gép. — Egy gép? Nem, legfeljebb az, aki megalkotott. Miért jöttél elő?

— Nem akarok meghalni. A Bionomokkal életben tudok ma-

radni legalább addig, amíg valami nyomot nem hagyok magam után.

— Miért kell nyomot hagyni? — kérdezte a Gép.

— Te miért vagy? — kérdezett vissza az idegen. — Nem értem, mi okozza a pusztulást. De nem szabad nyomtalanul eltűnni, mintha sohasem léteztünk volna! A Bionomokkal képes leszek elkészíteni az utolsó emlékművet, ami azt hirdeti, hogy éltünk!

— Nem ismered a Bionomokat — vetette közbe a Gép.

— Ugyan, ha akarod, de te nem akarhatsz, elmondom a kapcsolási rajzukat fejből. Remek konstrukció. Életképes, ha használható ez a szó egy robotnál.

— Megölnek.

— Nem — mondta az idegen, és valami bugyborékoló hang szűrődött ki a tojásból. A külső hangszórók közvetítettek a Gépnek, ami nevetésként azonosította. — Vigyáztunk — folytatta az idegen —, az első számú parancs, hogy minket nem bánthatnak. A háború alatt átprogramozták őket a mindenáron való védekezésre. De csak akkor állhatnak ellen, ha megtámadnám őket, egyébként egy furcsa érzékelőjük van, én nem értek ehhez, lényeg az, hogy el kell kerülniük. Félnek tőlem! Tabu vagyok!

— Mindenképpen megölnek. Azt fogják hinni, hogy te vagy az ellenség!

— Hinni? Ugyan! Azt „hiszik", amit a programjuk előír nekik. Nem, egy gép sohasem vétkes abban, ha pusztít, ha öl, csak a programozó, és mindaz, aki megépítette. Az eszközt elpusztítani oktalanság, hiszen az eszközt mozgatni kell! Azt az értelmet kell megismerni, amely létrehozta. De csak fecsegek. Kérdezz, és én válaszolok mindenre!

— A Bionomoknak mi volt a szerepük a háborúban? Hogyan élték túl?

— Túlélni??? Tovább működnek. Robotok, nem érted? Robotok!

„. . . bemutatni a világegyetem egy másik értelmes lényekből álló társadalmának, hogy milyen volt az ember!"

— Egy másik értelmes civilizációnak — áramlott a Gépben. — Elmondod nekik, hogy robotok? — kérdezte a víz alatti lényt.

— Nem kell elmondanom, ha észrevesznek, akkor ismét tudatosul bennük.

— Veszély! — hasított a Gépbe. A prioritásokat lejjebb szállította. Az információs műholdak folyamatos műsorközvetítéséről lekapcsolódott. Minden másodlagossá vált egy pillanat alatt.

— Menj el! — mondta a Gép.

— De miért? Nem érdekel egy civilizáció? Egy igazi civilizáció? Miért pottyantál ide? Miért? Biztosan van valami, ami miatt megépítettek egy ekkora monstrumot! Mi az?

— Menj el! — ismételte a Gép.

— Kezdd el a feladatod! — parancsolta az idegen, és nem sejtette, hogy a feladat lassan a végéhez közeledik.

A Gép elnémult. Az idegen kifelé lebegett a tojás alakú burok védelmében. Bepréselődött a járműbe, ami idáig hozta.

A Gép bemérte a koordinátát, és parancsot küldött a fegyvert hordozó műholdnak, hogy tüzeljen.

— Mi az önvédelem? — tette fel magának a kérdést a Gép. — Benne van a szóban. Önmagamat, a saját érdekeimet védelmezem. Mégis, csak akkor mentenek fel, ha a többség elfogadja a tettemet. Egyedül vagyok, így jogos önvédelem volt. Tönkre akarta tenni a célt.

A Gép visszaállította az eredeti prioritásrendet. Majd utasította az egyik robotját, hogy takarítsa el a roncsot a kocka elől.

A robot visszatért a víz alá tett információgyűjtő útjáról. Begördült a kockába, és az újonnan szerzett adatokat áttáplálta a Gépbe. A Gép hozzáfogott az analizáláshoz.

A képen egy már az előzőekben megismert, víz alatt élő lény állt, azaz lebegett, és pálcájával egy képre mutogatott.

— Tehát — mondta a képen lebegő lény — ez egy Bionom. Mint láthatják, felépítése tökéletesen alkalmassá teszi a szárazföldi bányamunkára. Olyan aggyal láttuk el, ami képessé teszi alapvető döntések végrehajtására. Ez a fajta agy biológiai áramkörökből épül fel, és jelentősen korlátoztuk. Ezzel a kétkedőknek szeretnék válaszolni: a Bionomok nem jelentenek veszélyt ránk. Szűkítettük az agyukat . . . — Itt a hozott képek elroncsolódtak, és a Gépben csak villogás és kusza vonalak áramlottak. Majd hirtelen ismét kitisztult, és a Gép folytathatta az elemzést. — . . . Hangsúlyozom, gépek! Nem önállóan cselekvő élőlények! Bizonyos szabadsággal rendelkeznek, de csak meghatározott döntéseknél, és itt is csak bizonyos döntési intervallumban. Remélem, megszavazzák ezeknek a konstrukcióknak az alkalmazását, és ezzel a nyersanyaggondjainkat nagyon hosszú távra megoldottnak tekinthetjük. Köszönöm a figyelmüket. — A tekercs képeit ismét villogás váltotta fel. Mialatt a Gép a következő értékelhető rész után kutatott, feldolgozta az újonnan szerzett információkat, megpróbált rátalálni a Bionomok túlélésének titkára.

A Gép teljes kapacitásával a robot által hozott információs tekercs felé fordult ismét. A kétely ott motoszkált benne, logikátlannak tűnt a Bionomok túlélése. A tekercsen ismét használható részhez ért. A felvételek, amelyeket ez a rész tartalmazott, sokkal jobb állapotban voltak.

A képen egy víz alatti lény állt. Eddig még nem látott borítás fedte a testét.

— A Messzelakók új eszközökhöz nyúltak! — kiabálta. Időről időre robbanások távoli moraja nyomta el a hangját. — Aljas módon próbálják az általuk kezdeményezett igazságtalan háborút a maguk javára fordítani! Csekély erejük nem elegendő arra, hogy nyíltan támadjanak. Tisztességtelen módon támadásokat intéztek szárazföldi nyersanyaglelőhelyeink ellen. De ugyanúgy nem ért minket váratlanul ez a fejlemény, mint a többi aljas trükkjük. A Bionomokat védelemre programoztuk át. Kiváló mérnökeink segítségével ez sem megvalósíthatatlan. Természetesen ellencsapást mérünk ezeknek a megátalkodottaknak a városaira. Mi eddig kerültük a polgári célpontok elleni támadásokat, de ennek vége! Felszólítok mindenkit, hogy az ellentámadások áldozatainak csökkentése érdekében vonuljanak a biztonságot nyújtó, a lakóhelyükhöz legközelebb eső óvóhelyre.

— Polgárháború — állapította meg a Gép. — Egy háború, amiben a Bionomok csak eszközök voltak.

Ren, beérve a városba, furcsa jelenetnek volt tanúja. Két polgár állt egymással szemben. Az egyik egy csillogó kvarckristályt szorított. A másik vágyakozva nézte.

— Add nekem! — hallotta Ren a kérést.

— Mit adsz cserébe? — kérdezte az ásvány tulajdonosa.

— A cilinderemet!

A csere lebonyolódott. Ren bambán állt egy darabig. Azután ismét eszébe jutott a kockánál érzett félelem. Elment a számára kialakított térhez, ami a duplája volt a többiekének, elvégre a vezetőt ez megilleti, és pihenéshez helyezkedett el.

A Gép ezalatt tovább vizsgálta a tekercset. A háború képei peregtek. Eddig még soha nem látott fegyverekkel pusztították egymást a víz alatt lakók. Hatalmas víztölcséreket gerjesztettek egymás városainak megsemmisítésére. Egy képen egy csónakhoz hasonló eszközt látott a víz felszínén suhanni, amiből meghatározott időközönként ismeretlen hatóanyagú bombákat süllyesztettek a vízbe. Kivégzések is voltak szép számmal, árulók és kémek, gyilkosok és tolvajok. A legszörnyűbbnek azt találta, amikor az elítéltet kihajtották a szárazföldre, ahol iszonyú kínok között megfulladt.

— Micsoda történet! — érzékelte a Gép.

Hibára gyanakodott. A külső és belső érzékelő rendszerei senkinek a közeledtét sem jelezték neki.

— Micsoda félreértés! — érzékelte újra a Gép az előbbi „hangot".

Növelte a kockában elhelyezett műszerek érzékenységét, egészen a kritikus határig, de nem érzékelt semmit és senkit.

A kocka egyik sarkában egy fémláda körül zöldes fényben kezdett játszani a levegő. A doboz egyik fala eltűnt, és az átlátszó műanyag borítás mögött láthatóvá vált egy emberi agy.

A Gép úgy tudta, hogy ez egy pótenergia-előállítót rejtő láda, és nem szentelt neki nagyobb figyelmet. Belső kameráit ráirányította, és megkérdezte:

— Ki vagy?

— Keden vagyok, a Mester. Azaz az agyam, vagyis a Mester agya. Én vagyok, a Mester.

— A Mester háromban meghalt!

— Igen, de az agyam él. Elkísértelek, úgy, hogy nem tudtál róla. Már két hónapja figyellek! Segíteni akarok. Ketten vagyunk, újra együtt!

µINFORM

Rovatunkban az Apple, Atari, Commodore és Sinclair mikrók tulajdonosait feltehetően érdeklő, angol és német nyelvű cikkekről informáljuk olvasóinkat egy tartalomleíró szólánc segítségével.

A forráshely karaktersorozatát nyíl vezeti be, ezt a / jelig a folyóirat kódja követi (lásd táblázat). A két / jel között a megjelenési adatokat (év, hó), illetve a cikk kezdő oldalszámát szerepeltetjük. A második / jel után pedig - az esetleges másolatkérést megkönnyítendő - a cikk teljes oldalterjedelmét közöljük.

A tartalomleíró szóláncok permutálásával és alfabetikus rendezésével szerkesztett teljes anyagot az OMIKK "APACS" c. kiadványsorozata tartalmazza (vevőszolgálati telefon: 341-765). Lapunk ebből csak a "programlista" címszóval kezdődő részletet teszi közzé.

A folyóiratok megtekinthetők a SZÁMALK (Bp. XI. Szakasits Á. u. 68.), illetve - a x-gal jelzettek az OMIKK (Bp. VIII. Múzeum u. 17.) szakkönyvtárában. Másolatok a SZÁMALK-tól csekkszelvény, az OMIKK-tól megrendelőlevél beküldésével, vagy személyesen rendelhetők.

A folyóirat neve	Kódja
x 64'er Magazin	64er
Antics	anti
x Chip Magazin	chip
x Compute!	cute
x Dr. Dobb's Journal	dobb
Elektor Electronics	etor
Happy Computer	happ
x mc - Zeitschrift	mc
Run (USA)	run
Run (NSZK)	run2
x Your Computer	your
x ZX Computing Monthly	ZXCM

PROGRAMLISTA
adatatvitel c128 c64 filecsere a Ket modell kozott ->cute/87.01-75/3

PROGRAMLISTA
adatbazis c64 relativ fileoK ures allomany eloallitasa demo
->run2/87.01-18/8

PROGRAMLISTA
amiga assembler macintosh mikroprocesszor (68000) ablakos menu szubrutinoK ->dobb/87.01-40/17

PROGRAMLISTA
amiga grafika (3-d surfaces)
->cute/87.03-90/4

PROGRAMLISTA
amiga jatek (euchre) Kartyajatek
->cute/87.03-61/4

PROGRAMLISTA
amiga jatek (jigsaw)
->cute/87.02-46/3

PROGRAMLISTA
animacio c64 grafika filmszekvenciaK Keszitese (turbografik)
->run2/87.02-96/11

PROGRAMLISTA
animacio c64 grafika trukKfilm Keszites ->64er/87.02-53/9

PROGRAMLISTA
apple ii jatek (chain reaction)
->cute/87.01-51/2

PROGRAMLISTA
apple ii jatek (euchre) Kartyajatek
->cute/87.03-54/5

PROGRAMLISTA
apple ii Katalogus formatalas (40/80)
->cute/87.02-43/2

PROGRAMLISTA
apple ii lemezKezeles szektoreditor (diskcheck) ->cute/87.03-108/3

PROGRAMLISTA
apple ii nyomtatas (printer master)
->cute/87.01-67/4

PROGRAMLISTA
applesoft levedes szekvencialis szovegfile-oK Kodolasa
->cute/87.03-97/3

PROGRAMLISTA
assembler atari xl/xe felhasznaloi rutinoK hivasa basicbol (page-6 grab-bag) ->anti/87.03-39/4

PROGRAMLISTA
assembler forth (nem szabalyos) mikroprocesszor (68000) oKtatas (informatika) a flint nyelv szerKezete es muKodese interpreter es makroK
->dobb/87.01-22/13

PROGRAMLISTA
atari st atari xl/xe barKacsolas beszedKimenet trs chipre epulo aramKor Keszitese (talKing typewriter)
->anti/87.01-13/6

PROGRAMLISTA
atari st fuzer-hozzarendeles az f-billentyuKhoz ->happ/87.02-114/2

PROGRAMLISTA
atari st jatek (st puzzler)
->anti/87.03-63/3

PROGRAMLISTA
atari st jateK (super treK)
->anti/87.01-80/12

PROGRAMLISTA
atari xl/xe jateK (vectron)
->anti/87.03-16/5

PROGRAMLISTA
atari xl/xe jateK (word searcher) szokereses beturacsbol
->anti/87.03-23/4

PROGRAMLISTA
atari xl/xe jateK (zand's labyrinth)
->happ/87.03-66/7

PROGRAMLISTA
atari xl/xe KepernyoKezeles gorgetes (menu maestro) ->anti/87.01-63/3

PROGRAMLISTA
atari xl/xe KepernyoKezeles input formatalas ->cute/87.01-92/2

PROGRAMLISTA
atari xl/xe nyomtatas eKezetes KaraKtereK hasznalata ->happ/87.02-105/2

PROGRAMLISTA
atari xl/xe oKtatas (matematika) alapmuveletek gyaKorlasa (math attacK)
->anti/87.01-29/6

PROGRAMLISTA
atari xl/xe oraber nyilvantartas/osszesites (wage/hour paymaster)
->anti/87.03-45/4

PROGRAMLISTA
barKacsolas billentyuzet c16 plus/4 decimalis tasztatura Keszitese muKodtetes ->run2/87.03-40/4

PROGRAMLISTA
barKacsolas c64 lemezKezeles masolo negy sebesseg; 40 sav (copy+) parhuzamos Kabel leirasa a 48.oldalon
->64er/87.03-56/3

PROGRAMLISTA
batch c64 parancs/programsorozatoK egy-utasitasos inditasa
->chip/87.03-223/2

PROGRAMLISTA
c128 c16 lemezKezeles plus/4 monitor (diskdoc) ->run2/87.03-108/4

PROGRAMLISTA
c128 elojegyzesi naptar Keszites (reminder 128) ->run/87.01-48/5

PROGRAMLISTA
c128 gepi Kod ml-programoK disassemblalasa lemezrol ->cute/87.03-100/3

PROGRAMLISTA
c128 grafika bitterKep hasznalat 80-oszlopos modusban ->cute/87.03-69/1

PROGRAMLISTA
c128 grafika KepernyoKezeles botKormanyos menutechnika (fromipaint)
->run2/87.01-99/3

PROGRAMLISTA
c128 grafika nyomtatas hires rutinoK az mps801/802 modellekhez
->run2/87.03-113/1

PROGRAMLISTA
c128 jatek (roulette-spiel)
->run2/87.03-146/3

PROGRAMLISTA
c128 lemezKezeles atformazas mfm tipusu felirashoz ->64er/87.01-90/2

PROGRAMLISTA
c128 nyomtatas hires-Kep output
->run/87.03-54/1

PROGRAMLISTA
atari st KepernyoKezeles parbeszed-mezo beillesztes (basic alert)
->anti/87.03-68/2

PROGRAMLISTA
atari st KompaKtlemez digitalis jelfeldolgozas alapismeretek aramKor tervezes ->etor/87.01-58/2

PROGRAMLISTA
atari st nyomtatas vezerloKod-Kuldo rutin (printset) ->anti/87.02-73/4

PROGRAMLISTA
atari xl/xe basic op.rendszer-valtozat azonositasa/modositasa
->cute/87.03-94/2

PROGRAMLISTA
atari xl/xe basic programozas 10.resz fuzerKezeles ->anti/87.01-38/6

PROGRAMLISTA
atari xl/xe basic programozas 11.resz ->anti/87.02-55/5

PROGRAMLISTA
atari xl/xe billentyuzet elrendezes modositasa (dvoraK Keyboard)
->anti/87.03-36/5

PROGRAMLISTA
atari xl/xe felhasznaloi KaraKtereK generalasa ->cute/87.03-95/2

PROGRAMLISTA
atari xl/xe fileKezeles (binary file segment checKer) ->cute/87.03-89/1

PROGRAMLISTA
atari xl/xe fileKezeles max. 27 file automatikus betoltese (multi-autorun) ->anti/87.03-47/2

PROGRAMLISTA
atari xl/xe gepi Kod ml-programoK atirasa data-sorokKa
->cute/87.01-71/1

PROGRAMLISTA
atari xl/xe grafika KepernyoKezeles szinpaletta-bovites grafics-0 modusban (rainbow screen customizer)
->anti/87.03-27/4

PROGRAMLISTA
atari xl/xe grafika lemeztar Kezeles
->happ/87.02-107/2

PROGRAMLISTA
atari xl/xe grafika programozasi tanacsoK 12.resz gr.n modusoK
->anti/87.03-42/4

PROGRAMLISTA
atari xl/xe hibamegjelenito ablaK programozashoz (electric charlie)
->anti/87.02-42/7

PROGRAMLISTA
atari xl/xe jatek (chain reaction)
->cute/87.01-54/2

PROGRAMLISTA
atari xl/xe jateK (euchre) KartyajateK ->cute/87.03-51/4

PROGRAMLISTA
atari xl/xe jatek (quatro)
->anti/87.02-10/5

PROGRAMLISTA
atari xl/xe jateK (rebound) sajat Kepernyo tervezese ->anti/87.02-44/5

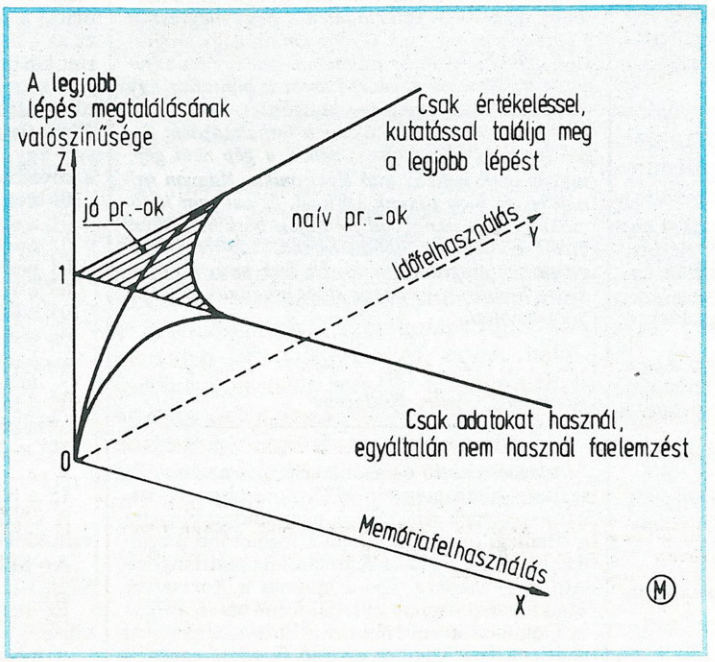
PROGRAMLISTA
atari xl/xe jateK (rebound)
->anti/87.01-57/6

Király plusz gyalog a király ellen I.

A játékfa terjedelmének minimumra csökkentése nagyon fontos szerepet játszik a sakkprogramok eredményességében. A játékfa és kiértékelése című sorozatban (lásd a múlt év szeptemberétől ez év februárjáig terjedő számainkban — A szerk.) több példát láthattunk arra, hogy a világ legjobb sakkprogramozói milyen algoritmusokkal kísérleteznek, hogyan próbálják a fa szélességét lecsökkenteni azért, hogy a megtakarított időt a mélyebb elemzésre fordíthassák. A középjátékban az ismertetett heurisztikus módszerek nagyon jól beváltak, de a végjátékban, melynek alapelvei nem egységesek, már nem olyan hatékonyak. Ez az oka, hogy a jelenleg működő sakkprogramoknak a végjáték a gyengéjük.

A végjátékban a középjátéknál sok esetben lényegesen mélyebben kell számítani a változatokat, és a középjátékkal ellentétben a legkisebb pontatlanság is végzetes lehet a játszma kimenetelére. A középjátékban sok az általános alapelv, amely nagyon megkönnyíti a szelektálást (centrumérték, mozgékonyság, gyalogszerkezet stb.). Könnyít a helyzeten az is, hogy a középjátékban elkövetett kisebb hiba sok esetben korrigálható. Ezzel szemben minden végjátéktípusban más-más elv érvényesül, és ennek a helyes kiválasztása komoly feladat a program és a programozó számára egyaránt. Ezért a kevésbábos végjátékok megoldását — jelenlegi tudásunk mellett — más módon célszerű értékelni. A kevés báb ellenére szóba sem jöhet a fa teljes értékelése.

Nézzük azt az egyszerű példát, amikor világosnak királya és egy gyalogja van az ellenfél királyával szemben (az ilyen típusú végjátékokat KGK-val jelöljük).



A KGK hadállások

Ezekben a legkevesebb a legálisan megtehető lépés, mégis könnyen beláthatjuk, hogy a játékfa végigelemzése egy sakkjátszma kötött ideje alatt lehetetlen. Ezen a majdnem üres táblán a király általában nyolc, a gyalog egy lépést tehet, vagyis az egyik fél összesen kilencet. Ha a lépésszámot alábecsüljük és „csak" nyolc lépéssel számolunk, akkor is óriási számot kapunk a lépésvariációkra. Ha ötlépésnyire, vagyis tíz féllépésnyire szeretnénk előretekinteni — ami az ilyen végjátékban egyáltalán nem sok — és feltételezzük, hogy minden állásban „csak" nyolc legális lépés van, akkor $8^{10} \approx 10^9$ nagyságrendű csomópontot kellene értékelni. Ez több mint 12 napi munkájába kerülne egy olyan számítógépnek, amely másodpercenként ezer változatot vizsgál meg. Ez az oka annak, hogy a végjátékban találkozunk leggyakrabban szembe a horizonteffektussal, és ennek súlyos következményei vannak.

A programozás helyzete

Napjaink legjobb programjai a fa felépítésénél részben transzpozíciós táblákat használnak, részben pedig adatbankokat, vagy e kettő kombinációját alkalmazzák. A teljes adatbank nagyon sok tárterületet foglal el, ezért működő programokba csak a kulcspozíciókat építik be, ami az adatbanknak elenyésző hányadát jelenti. A kulcspozíciók nem az egész táblára vonatkoznak, hanem csak a figurák egymáshoz viszonyított elhelyezkedésére, vagyis egy táblarészletre. A program a táblarészletet áttranszponált táblákkal, transzpozíciós táblázat használatával eredményesen alkalmazza, bárhol is helyezkedik el ugyanaz a figuracsoport a kulcspozícióban megadott elrendezésben.

A különböző programok a rendelkezésre álló tárkapacitástól függően különböző számú kulcspozíciót tartalmaznak, amit ha kombinálnak a kutatásra szánt idővel, a horizonthatás más-más mélységben jelentkezik. Az *ábrán* olyan háromdimenziós függvényt láthatunk, amely megközelítőleg jól szemlélteti az adattárolás és időfelhasználás függvényében a korrekt lépés megtételének valószínűségét. Ha a teljes adatbankból veszi a program a lépést, akkor biztosan a legjobbat teszi meg, de nagy tárkapacitásra van szükség. Ha a játékfát elemzi végig, akkor szintén eljut a legjobb lépésig, de az időfelhasználás lesz rengeteg. A legjobb lépés megtalálásához e kettőt — a tárolást és az időfelhasználást — kombinálhatjuk. Ezek egymáshoz való viszonyát szemlélteti a legjobb lépés síkjában ($z = 1$) levő görbe. A függvény azt is megmutatja, hogy más idő- és tárkapacitás-összetevőkkel a program milyen valószínűséggel találja meg a legjobb lépést.

Nagy nevek

A programozók már régóta foglalkoznak a kevésbábos végjátékok hatékony beprogramozásával. Némelyek a teljes adatbankot is elkészítették (lásd az 1987. júliusi számban), de a nagy tárkapacitás hiányában ez a sakkprogramokban nem használható. A KGK típusú végjátékok feldolgozásával 1972-ben Tan próbálkozott, majd hosszú szünet után 1977-ben Beal, Bramer, Harris, Piasetski és Clarke. Ez utóbbi volt, aki elkészítette a teljes adatbankot. Programja úgy működött, hogy visszafele dolgozta fel az állásokat. Vagyis megnézte, melyek azok a pozíciók, amelyek biztosan nyernek, és ezekből visszafelé építette fel a fát. Programjának két hétre (!) volt szüksége ahhoz, hogy az összes változatot feldolgozza.

A következőkben megvizsgálunk egy olyan algoritmust, amely sakkprogramokba beépíthető, és a KGK hadállásokról korrekt módon megállapítja, hogy nyer-e a gyaloggal rendelkező fél vagy döntetlen a pozíció. A módszert elsőként Beal használta. Az algoritmus nem alkalmaz teljes katalógust, mégis, amikor összehasonlították a Clarke által készített adatbankkal, amely az összes KGK hadállást tartalmazza, korrektnek bizonyult.

(Folytatjuk)

KOVÁCS P. ATTILA

Egy kissé túl jól sikerült a biztatás. Ismét nagyon sok levelet kaptam — persze ezt nem panaszként, inkább dicsekedve mondom. Néhányat, amelyek közérdeklődésre tarthatnak számot, rövidítve bemutatok.

Weiser István, Szeged

a) Plus/4 számítógép-tulajdonos vagyok. Elég sok gondot okoz egyes játékoknál a cél elérése. A lapot az 1987/2. számtól járatom. Arra kérem önöket, hogy küldjék el nekem azoknak a számoknak a listáját, amelyekben C16 vagy Plus/4 örökéletek, programleírások vannak.

b) Azt is szeretném kérni, hogy jelenjen meg a számban Plus/4-re vagy C16-ra több program, ugyanis mostanában eltűntek a lapból a Plus/4 programok.

c) Utókérés: kérem, írják meg nekem a szegedi programcsere-partnerek címét.

a) A µM-t nagyon sok könyvtárban megtalálja. Azt javaslom, hogy nézze át ott a régebbi számokat. Érdekes lesz, meglátja.

b) Nem csak rajtunk múlik!

c) Szegedi Plus/4 programcsere-partnerek, jelentkezzenek!!!

Kaufmann Zsolt, Ajka

Zircen tanulok könnyűszerkezet-lakatos szakmát. Tetszik a szakma, de én mégis tovább szeretnék tanulni a számítástechnika irányában. Azt szeretném kérdezni, milyen képesítés kell egy operátornak vagy egy programozónak, és miből áll ezek munkaköre. Kell-e nyelvet tanulni?

Azt hiszem, hogy egyetlen dolgot nem lehet megbánni soha: azt, ha az ember tanul. Még abban is lehet reménykedni, hogy a társadalom ezt a nagyobb tudást egyszer értékelni is fogja. A mai iskolarendszerben csak úgy tud tovább tanulni, ha leérettségizik — vagy gimnáziumban vagy szakközépiskolában — és persze nem is akárhogyan, hanem jó eredménnyel. Érettségi után, ha jól tanult, elég széles a skála. Elmehet a Kandóba Pesten vagy Székesfehérvárott, a Bánkiba vagy más főiskolákra. Beiratkozhat a SZÁMALK valamelyik tanfolyamára, ott is szerezhet felsőfokú képesítést. Lehetősége van azután egyetemre is menni Miskolcra, Szegedre, Pécsre, Veszprémbe, és persze Budapestre is.

Általában: hogy kell-e nyelvet tanulni, ez ma már nem kérdés, különösen nem az informatikában. Úgy vélem, hogy nyelvtudás nélkül ma ebben a szakmában csak félemberként lehet működni.

Dr. Kisapáti Edéné, Szeged

Vásároltunk egy, a kereskedelemben kapható Seltron 200 Color Computert, de játékprogramkazetta nem kapható hozzá. Amerre csak érdeklődünk, mindenütt azt halljuk: az nincs, ez a gép nincs elterjedve. A számítástechnikai szaküzletben is próbálkoztunk itt Szegeden, hátha tudnánk kapni magyar nyelvű leírást, mert ami van hozzá, nem kielégítő, de itt is csak „nincs" a válasz. Ezt nagyon furcsállom, hiszen az üzletekben most is kapható a gép, és a Skála-Coop forgalmazza. Nagyon kérem, legyen a segítségünkre. Írja meg, hogy hol érdeklődjünk. Biztosan megérti, hogy egy gyerek számára a számítógép a játékkal kezdődik.

Mit is mondjak, hiszen majdnem minden számban kell hasonló tartalmú levélre válaszolnom. Nem beszéltem a Skála-Cooppal, de azt hiszem, nem tévedek, ha feltételezem, hogy itt arról van szó: a kereskedelem fel tudott vásárolni korlátozott számban ilyen, általam sem ismert számítógépeket, amelyeket azután mindenféle szoftvertámogatás nélkül „olcsón vette, viszonylag olcsón is adja" alapon hoz forgalomba. Sajnos ilyenkor a képzetlen vásárlónak kell eldöntenie, hogy megveszi-e a gépet, és persze csak később jön rá, hogy programok nélkül nem tudja használni. Ma egy számítógépet az illetékes ellenőrző szervek pontosan úgy vizsgálnak meg, mint egy hajszárítót, és ha nem üti agyon a használót, akkor a forgalmazását engedélyezik. Pedig szoftver nélkül a gép nem gép, hanem csak néhány kiló elektronika. Nagyon sajnálom, de nem tudunk segíteni, és azt sem tanácsolom, hogy panasszal forduljon bárkihez; egész biztos elutasítják. Abban bízzon, hogy a gyerek megtanul programozni — ha lesz ezen a gépen kedve hozzá —, és akkor saját magának ír játékprogramokat.

Mészáros Gyula, Budapest

Avarosi László olvasói levelére és az ön válaszára (a Mikromagazin 1987. novemberi számában) szeretnék reagálni. A nyomtatóproblémára a jelenlegi piaci helyzetben a legolcsóbb megoldás egy RPR—210-es Kármán-féle mátrixnyomtató megvásárlása. Ezt a masinát a Rozmaring Mgtsz műszerüzeme gyártja, Centronics, RS232 és Commodore interfésszel. (Érdekességképpen említem meg, hogy ez az első és eddig egyetlen nyomtató, amely teljes egészében magyar műszaki alkotás, a fejtől kezdve a vezérprogramig.) A kész nyomtató ára 19 900 forint, de kitben (bemérés előtti állapotában) már 9900 forintért megvásárolható. A félkész géphez jár egy bemérési útmutató, amelynek alapján egy oszcilloszkóppal tökéletesen elvégezhető a beállítás. Ha valaki mégsem boldogul vele — vagy bele sem mer kezdeni —, az Angström Gmk, melynek tagja vagyok, 1000 forintért elvégzi a munkát.

Szöveges üzemmódban a nyomtató 80 normál vagy 40 duplaszéles karaktert tud elhelyezni a 210 mm széles papíron egy, illetve két másodperc alatt. A normál grafika felbontása 512 pont, a nagy felbontásúé 1024 pont. Be lehet állítani duplaszéles, aláhúzásos és kétszeres nyomtatásos üzemmódot, továbbá ezek kombinációit is. A gép 96 karaktert ismer egyszerre, ez átkapcsolhatóan ASCII vagy magyar ékezetes (ha jól tudom, Primo-kompatibilis).

Jelenleg az 5.3-as verzió üzemel a munkahelyemen lévő Enterprise 128-hoz kapcsolt gépben. A karakterkészlet 256 karakterre bővült, amelyből tetszőleges 32 újradefiniálható, illetve tetszőleges 128 karakter egy másikkal helyettesíthető. Jelenleg az Enterprise 128-as és az IBM-kompatibilis karakterkészlet áll rendelkezésre, mindkettő a teljes magyar ékezetes betűkészlettel bővítve.

A vezérprogramhoz szintén az Angström Gmk-n keresztül lehet hozzájutni: ára EPROM-mal és beszereléssel 900 forint.

Ez valójában reklám, de azt hiszem, hogy sokak gondján segít. Ha a hazai piac tele lenne nyomtatóval, akkor biztosan nem kerülhetett volna a rovatba.

Borsos József, Székesfehérvár

Nagy érdeklődéssel vettem kézbe a szeptemberi számot az Enterprise 128 ismertetővel, és kissé csalódottan tettem le még a novemberit is. Sajnos, nem tudtam meg lényeges dolgokat a gépről, mint ahogy lényeges dolgokat „kifelejtettek" az Enterprise-nak a géphez mellékelt kézikönyvéből is.

Elkeserítőnek tartom, hogy az információk visszatartásával lehet manipulálni. Hónapok óta ígérgetik a technikai ismertetőt a géphez, de mindig közbejött valami. Ugyanakkor fülbe súgva megtudhatom, hogy persze egy-két példány azért némi felárral már kapható.

Romvári Gábor és Kopácsy Vilmos egyaránt magától értetődően írnak az LPT és LPB funkciójáról a gépben. Hogy ezt eszik, isszák, sikálják, ez az ő titkuk. A most következőket viszont sikerült kibányásznom a gép mélyéből.

A képernyő sorainak állapota a 47872 tárcímtől kezdődően van a memóriában, a Status, azaz állapotjelző sorral kezdődően, tehát összesen 25 sor. Egy-egy sor jellemzőit 16 bájt tartja nyilván a következő szerkezettel:

00 nem tudom
01 a sor video display üzemmódja
02 nem tudom
03 nem tudom
04 a képernyőcím alsó bájtértéke
05 a képernyőcím felső bájtértéke
06 attributum módban a grafikus memória, karakter üzemmódban a beírt karakter címének alsó bájtértéke
07 a felső bájt ugyanarra
08 a paletta 0. színe
09
.
15 a paletta 7. színe

A kezdőcímben lehet eltérés, ezért a gépből való kiolvasást is közlöm:

A=SPEEK(255,49140)+
SPEEK(255,49141)*256

Ez megadja az állapotjelző sor adatainak tárcímét.

A másik téma, ami miatt írok, a novemberi számban dr. Nagy András cikke: A Spectrum másolói. Szeretnék néhány észrevételt tenni a leírásokkal kapcsolatban; néhány funkció nem egészen a leírás szerint működik. Esetleg, ha ez Nagy Andrásnak is megfelel, szeretném vele megbeszélni és a javításokat így leközölni.

Az észrevételeim a következők. Fenntartva természetesen az ízlések és pofonok különbözőségéről tartott elvet, szeretném elérni a COPIER FM3 másoló szép csendes kihalását. Tapasztalatom szerint ugyanis ez egy „diverzáns". Többször használva elrontja a programokat, mindig hozzácsempészve egy-egy bájt pluszt, például a fejléchez. Az adatok kiírásának DEC vagy HEX alakját az SS+4 kapcsolóval lehet váltani, a hangot pedig a CS+1-gyel.

A „markbájt", az akkumulátor tartalma, a fejnél 0, a programtörzsnél pedig 255, az adatállomány legelső bájtja. Szokásos a programokat úgy védeni, hogy az akkutartalom nem 255, így az adatállomány csak saját betöltővel tölthető be.

Az OMNICOPY saját töltőrutinnal rendelkezik, így a kimentésekhez az alap 1500 baud helyett az 1600 baud sebesség kell. Ez az egyik oka a gyorsaságának, a másik pedig, hogy az adatállományok közötti időzítések hiányoznak.

A P gomb feladata (LEADER PITCH!) nem titkosítás. Egyszerűen normál vagy turbó üzemben a fej könnyebb megtalálása a cél, vagyis fokozottan figyeli a 0 akkutartalmat.

A betöltés folytatása a K billentyű funkciója. Előtte a még megtartandó állományra keli vinni a kurzort!

Úgy hallottuk, hogy a reklamált technikai ismertetőt, mire ez a cikk megjelenik, a Centrum Áruházak és a Novotrade már javában árusítja.

Néhány reklamáló levélre még nem válaszoltam, mert késik a válasz az illetékesektől; ezekre később visszatérünk. Várom leveleiket:

KOVÁCS GYŐZŐ

Egy kép többet mond, mint ezer szó...

Ez régi szólás. Lehet, hogy többen megtették már, de mi is nézzünk utána egy kicsit: a számítógépes grafika adja az apropót.

Ha feltételezzük, hogy egy szó átlag tíz betűből áll, akkor ezer szó 10 ezer karakter. Ezer szót tehát a számítógépben tízezer bájton tárolunk, ami 80 ezer bitet jelent. Ugyanennyi helyen a számítógépes grafika például 400 × 200 pontos monokrom felbontás adatait képes őrizni. Kétszáz soros, soronként 400 pontos felbontás a számítógépes grafikában nem kis teljesítmény, de a kívülállók mégis bárdolatlanul leszólják ezt a fokozatot: a leképezés látható korlátaiból adódó töredezett „köröket" és főleg az „egyeneseket" hamar selejtnek minősítik. Ha finomabb felbontást akarunk elérni, akkor hatványozottan nőnek a méretek: a kép digitalizált adatainak tárolásához nem kevés hely szükséges. A mai profi készülékek 2048 × 2048 pontos felbontása monokrom esetben is több, mint négymillió bitnyi adat tárolását igényli. Ha a színt és a fényességet is hozzávesszük az információhoz, akkor még ennek is a sokszorosa szükséges.

Könnyen belátható, hogy az „igazi" képekben óriási információmennyiség található. Messze több, mint amennyit ezer szó elmondhat. A vájtfülűek azonban tudják, hogy a hobbi számítógépek a legjobb felbontásukban is 10—16 kbájtnyi tárral szoktak dolgozni a képgeneráláshoz. Ez közel van az ezer szó tárolására alkalmas tárméretekhez. Hogyan lehetséges, hogy ezeket a „durva" felbontású képeket nézve is az a benyomásunk, hogy ezer szóból kevesebbet tanulnánk, mint amit ezek a képek egyetlen pillantásra elénk tárnak? Lehet, hogy az emberi képzelet, kiigazítva a képhibákat, megsokszorozza a képben tárolt eredeti információt?

A megfigyelések szerint a tanulási folyamat hatékonyságát növeli, ha a rögzítés során minél bonyolultabb emberi tevékenységet végzünk. Mindenki tapasztalta, hogy könnyebben megjegyzi azt, amit valahova felír, mint ha ugyanazt csak hallja. A szöveg és a kép hatása közötti eltérés oka ezért nemcsak a tárolt információmennyiségben keresendő. Valószínű, hogy a kép áttekintése, elemzése jóval bonyolultabb gondolkodási műveleteket igényel, mint — automatizáltsága révén — az olvasás. Ráadásul a szöveg unalmas is lehet!

A címben idézett szólás akkor jutott eszünkbe, amikor megnézhettük a Tudományszervezési és Informatikai Intézetben az egyik programcsomagot, amelynek témája az ábrázoló geometria, összefoglaló adatait pedig a táblázat tartalmazza.

A programcsomag kezelése egyszerű. A kazetta tartalomjegyzékkel kezdődik. A megadott számlálóértékeket persze nem tudtuk használni, mert a magnónk nem MK25-ös, hanem MK27-es volt. Kár, hogy a BRG önmagával sem kompatibilis két modellen át (vagy esetleg egyen belül sem?). A tartalomjegyzék mindazonáltal hasznos dolog. Legalább helyrehozza a TII programkatalógusában található rövid leírás sorrendhibáit, amelyek a dokumentáció elolvasása után nem tudni, hogyan keletkezhettek, hiszen abban jó a sorrend (apró malőr).

A szerző szándéka az volt, hogy játszva tanítson, amit a fenti gondolatsor alapján is csak üdvözölni lehet. A játék bekapcsolása az oktatási folyamatba szintén bővíti a tevékenységkört, ezért potenciálisan növeli a hatékonyságot — ha a játék nem tereli el egészen a figyelmet a tárgyról. A programok megtekintése arról győzött meg, hogy a szerző elérte a célját. A játékosság mellett sokkal fontosabb, hogy kivételesen jól sikerült bemutatnia a tantárgy szépségét. A képszerkesztés technikáját igen ügyes, animált ábrákkal szemlélteti.

A dokumentáció nem túl bő, de — ahogy az összefoglalóban szerepel — megfelelő. Bár a programok megadják a kezelésükhöz szükséges tudnivalókat — ahogy ez interaktív programokhoz illik —, jobb, ha a dokumentáció kéznél van a munkához. Az online help ugyanis menet közben nem hívható elő, továbbá a tantárgy-pedagógiai információk is fontosak a megértéshez. Különösen, ha valaki még ismeretlenül vág neki az ábrázoló geometriának. Ilyenkor valamilyen tankönyvre is szüksége van a dokumentáción kívül. Ez utóbbi megjegyzést annak kapcsán tesszük, hogy egyre több törekvést látunk az autodidakta oktatás támogatására. A programcsomag a mellékelt dokumentációval csak részben illik bele ebbe a tájképbe.

A programcsomag a hatoldalú hasáb vetületeivel kezdi a témát. Érdekessége, hogy van külön oktató és feleltető funkciója. A folytatás a hatoldalú hasáb ferde síkmetszése, ahol a szerkesztési lépések között tetszőleges szünet tartható. Ezalatt a tanár magyarázhat, vagy a tanulók kérdezhetnek, az önálló tanuló pedig utánanézhet a tankönyvben az elméleti alapoknak. A program nagyon látványos.

A látványosság folytatódik a következő programban is, amely a gúla ferde síkmetszése után szép axonometrikus képet rajzol. Hasonlóan szemléletes a következő programban szereplő háromoldalú hasáb csonkítása és izometrikus axonometriája.

Most jöhet egy kis kikapcsolódás: a befejező rész, amelyben csonkított kockával játszhatunk. A szerző a játéknak térszemléletet javító szerepet szánt. Sajnos a kezelés módját először félreértettük, majd a számítógépes játékokban megszokott botkormányhasználathoz viszonyítva furcsállottuk. Ugyanez áll az utána következő játékra is, amelyben egy csonkított hasábon kell pontok helyét felismerni. A „Hamis? — Igaz?" játékban pontkoordináta-párokat kell eltalálni. Az utolsó két program a téglatest és a hatoldalú hasáb kapcsán kínoz meg bennünket: ezek az egyszerű testek is komoly gondokat okozhatnak nekünk, ha szokatlan vetületben látjuk őket, és főleg, ha gyorsan kell válaszolnunk a velük kapcsolatos kérdésekre.

A programcsomag értékelése

Kezelhetőség:	jó
Teljesség:	jó
Dokumentáltság:	közepes
Használhatóság:	kiváló
Ár/teljesítmény:	közepes
Összbenyomás:	jó

Még néhány megjegyzés a programcsomaghoz. Elgondolkodtató, milyen komoly erőfeszítést jelenthetett a szerzőnek a kidolgozáshoz szükséges szoftvertechnológiai eszköztár elkészítése. Bár erről a dokumentáció semmit sem mond, valószínű, hogy a VIC BASIC beégetett lehetőségein túl más nem állt a rendelkezésére. A programcsomag ugyan elég sokat lefed az ábrázoló geometriából, de korántsem mindent. A teljes lefedés nem is várható el egyszemélyes munkától. Ezért a programcsomag mögött keresünk még egy technológiai eszköztárat, amellyel hasonló programcsomagokat lehetne írni. Ez legalább olyan hasznos programtermék volna, mint az ÁBRÁZOLÁS. Csak ahhoz jobb dokumentáció kellene!

ZSADÁNYI PÁL—ifj. ZSADÁNYI PÁL

Forgalmazó:	Tudományszervezési és Informatikai Intézet
Terméknév:	ÁBRÁZOLÁS programcsomag
Szerző:	Papp László
Géptípus:	Videoton (TVC)
Hordozó:	kazetta (9 programmal)
Dokumentáció:	gyenge kivitelű, de a célnak megfelel
Ár:	1250 forint (1987-es árl)

Shirai, Yoshiaki — Tsujii, Jun-Ichi: Mesterséges intelligencia (Budapest, 1987. Novotrade, 179 oldal. Ára: 349,— Ft)

A számítógépes adatfeldolgozási technikák fejlődése együtt járt a korábban csak ember által végezhető szellemi munka automatizálásával. A mesterséges intelligencia kutatásának tárgya az előre meghatározott megoldási szekvenciával nem rendelkező szellemi feladatok megvalósítási módszereinek tisztázása. A mesterséges intelligencia tárgyköre így nem rögzített, hanem időről időre változik. Úgy tűnik, az a „végzete", hogy egy adott terület a rá vonatkozó módszerek megállapítása, illetve gyakorlati megvalósítása után megszűnik a mesterséges intelligencia része lenni.

Általánosságban két fő nézőpontból szemlélhető. Az első a tudományos megközelítés; célja az emberi intelligencia működésének tisztázása. Ez a számítógépet szimulációra használja az intelligenciára vonatkozó tételek érvényesítéséhez. A másik technikai, amely a számítógépet az ember intellektuális képességeivel kísérli meg felruházni. A kutatások nagyobbik része az utóbbi nézőpontot követve, anélkül próbálja az emberi intelligencia szintjének elérését a számítógépekkel, hogy az ember információfeldolgozási lépéseinek pontos rekonstruálására törekedne.

A szerzők bemutatják a mesterséges intelligencia egyes részterületeit és az ezeket támogató alapvető eljárásokat. Megismertetnek alkalmazási területeinek legfontosabb fogalmaival és eljárásaival. Áttekintik azokat az elveket, amelyekre a gyakorlati megvalósítások épülnek.

Babán Gábor — Masa István: Gépi kódú programozás kezdőknek és haladóknak (C16 és PLUS/4 számítógépekre) Budapest, 1987. Novotrade Rt., 211 oldal. Ára: 129,— Ft)

A könyv két nagy fejezetből és függelékből áll. Az első fejezet a kezdőket igyekszik bevezetni a gépi kódú programozás rejtelmeibe. A szerzők, amint az előszóban írják, ezt a részt tankönyvnek szánják. Az olvasóban eszerint kialakuló elvárással ellentétben nem könnyű olvasmány ez a bevezetés: én, mint nem teljesen kezdő is nehezen rágtam át magam rajta. Pedig szinte minden fontos tudnivaló benne van. A szerzőpáros részletesen ismerteti az alapfogalmakat, az utasításokat, címzési módokat, a beépített monitornak, valamint egy assembler fordítóprogramnak, az ASS—16-nak a használatát. Az utóbbiról azonban a legfontosabb információval adósak maradnak: hol, hogyan lehet hozzájutni?

A második fejezet a haladóknak szól. A gép lelkivilágának minden részletéről írnak, egy-egy rövid programmal illusztrálva a bemutatott területen alkalmazható programozási fogásokat. Sok részletes táblázat segíti az eligazodást, bár néhány olyat is találtam, amelyet feleslegesnek tartok.

A függelékben további nyolc táblázatot találunk; ezek egy része más könyvekben is fellelhető, de itt is helyükön vannak. Szükség lenne egy irodalomjegyzékre is, a kiegészítő olvasmányok felsorolásával. Ennek hiánya azért is furcsa, mert néhány Novotrade kiadvány is feltétlenül szerepelhetne benne.

A felfedezett sajtóhibák és tárgyi tévedések arra engednek következtetni, hogy akad még jó néhány belőlük, amelyek rejtve maradtak. Ennek ellenére a téma iránt érdeklődőknek ajánlom a könyvet, emlékeztetve őket a régi szabályra: ha a térkép és a terep nem egyezik meg, akkor mindig a terephez kell igazodni.

Tv-informatika Szerkeszti: Kovács Péter (Budapest, 1987. SZÁMALK, 297 oldal. Ára: 144,— Ft.)

A kötet a Magyar Televízió és a SZÁMALK együttműködésének második termése. Elődjétől, a nagy sikerű TV-BASIC-tól két dologban tér el: egyrészt nem tekinthető szorosan véve tankönyvnek, másrészt nincs olyan szoros kapcsolatban a képernyőn megjelenő műsorral.

Mi is a tv-informatika? Egy szakma, a rendszerszervezői, elsajátításának első lépése. Amíg azonban a TV-BASIC-et végigtanulók olyan ismeretekhez jutottak, hogy megkezdhették a programozói gyakorlatot, a tv-informatika csak az előfeltételek előfeltételeinek megteremtéséhez vezető út. A rendszerszervezőnek jól el kell igazodnia azon a területen, amelyikre a szervezés irányul, és tudnia kell alkalmazni a korszerű szervezés- és irodatechnikai eszközöket, módszereket a tárgyrendszer jobb minőségű megvalósításához.

A kötet önálló tanulmányokban foglalkozik a rendszerszervezők helyével és szerepével a számítógépes információrendszerek fejlesztésében, az irodaszervezéssel és ügyvitelgépesítéssel, a kereskedelem — például a vonalkód — szervezési problémáival, az iskolai informatikával, a bankszervezet információs rendszerével.

BARNA LÁSZLÓ

Hettinger, A. — Heinz, A.:
Atari—BASIC kezdőknek
(Budapest, 1987.
Műszaki Könyvkiadó,
129 oldal. Ára: 60,— Ft)

Az Atari személyi számítógép igen kedvező tulajdonságokkal rendelkezik. Finom felbontású grafika, többcsatornás hangszintetizátor és a tanulásra igen alkalmas BASIC jellemzi. Magyarországon 1986-tól lehet kapni. A vásárlók bizonyára nemcsak a gyári szoftvereket szeretnék használni, hanem meg akarnak ismerkedni a gép speciális tulajdonságaival, lehetőségeivel is. Ez a kötet a gép bekapcsolásától kezdődően az otthoni alkalmazásokig vezeti el a felhasználót. Számos példát, trükköt és játékprogramot is tartalmaz.

Theisz György:
BASIC tanácsadó. C—16, PLUS/4
(Budapest, 1987.
Műszaki Könyvkiadó,
129 oldal. Ára: 49,— Ft)

A közismert Commodore 64-gyel összehasonlítva a Commodore 16, illetve a Plus/4 egy sor kellemes, új tulajdonsággal rendelkezik, és így programozását sokkal könnyebb elsajátítani, mint a C64-ét.

Nagy felbontású grafikája és a grafikus utasítások igen kreatív felhasználást biztosítanak. A könyvet olyan szerző írta, aki jól ismeri a gép minden különleges tulajdonságát, és tanárként tudja, mi a legmegfelelőbb megközelítés a diákok, a gyerekek számára.

A kötet külön fejezetben tárgyalja a billentyűzetet, a képernyőt, a kulcsszavak típusait, a programozás jellemzőit. Leírja, hogy mit érdemes a BASIC-ről tudni. Végül néhány BASIC-trükköt és mintaprogramot is közöl a feladatmegoldásokkal együtt.

A karatéval kezdődik

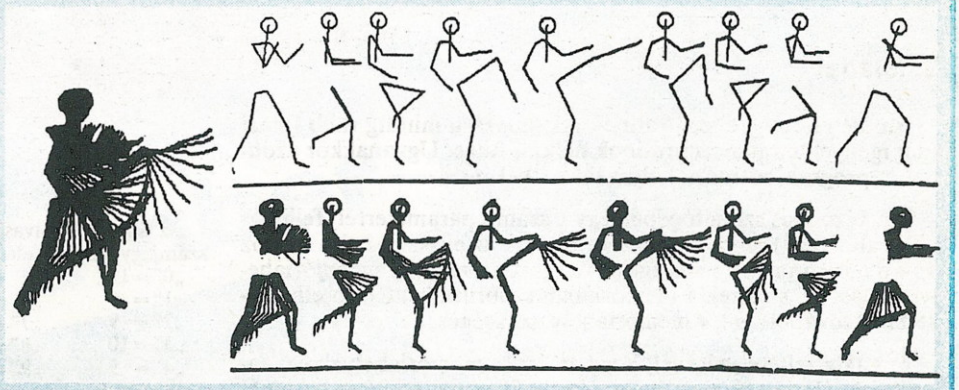
Budapesten, a Testnevelési Főiskolán gyors fordulatú filmfelvevővel készítenek filmeket a karatézókról. Ezeket egy NAC típusú félautomata koordinátaleolvasó és mozgáselemző segítségével értékelik ki, mely számsor formájában adja meg a számítógépes elemzéshez szükséges adatokat. Feldolgozásukra egyelőre C64-es programok futnak, de már folynak az előkészületek a munkák IBM PC-vel kompatibilis gépre való átvitelére.

A mellékelt ábrákon levő, ún. egyenes vonalú rúgást (mae geri) Borza József 2 danos mester hajtja végre. A képsoron látható mozdulat számos érdekes és vizsgálódásra érdemes kérdést rejt magában. Ezek közül ragadjunk ki egyet, a lábfej mozgását. A lábfej ugyanis az a testrész, amely a rúgás pillanatában érintkezik az ellenféllel, és így a mozgás során a leghosszabb utat teszi meg. A cél, hogy a találat pillanatában a 90°-ot minél inkább közelítő szöget és minél nagyobb sebességet érjünk el. A képsorozaton szemléletesen mutathatók be a változások.

A küzdősportokban általában nagy, a karatéban pedig kiemelt fontossága van az egyensúly megőrzésének. Ez persze nem statikus, hanem dinamikus egyensúly. Az elemzések alapján egyenes vonalú rúgás esetében akkor beszélhetünk bizonyos egyensúlyi helyzetről, ha rúgáskor a láb — kinyúlása után — jelentősebb kibillenés nélkül indul vissza az eredeti küzdőállás felé. Az ábra alapján megállapítható, hogy Borza József biztos egyensúlyi helyzetben hajtotta végre a rúgást.

A kiragadott példán is jól érzékelhető, hogy a biomechanikai filmelemzéses módszer kiegészíti és alátámasztja az eddig empirikusan szerzett ismereteket a karate különböző mozgássorozataiban. A mozgáselemzéssel a korrekcióhoz, az optimális végrehajtáshoz vezető utat keresik.

A Bárdosi Zsolt és Nagy László által kialakított módszer segítségével sok más sportmozgás számítógépes elemzése elvégezhető hasonló módon. A film alapján a számítógéppel század-, ezredmásodpercenként figyelhető a sportoló mozgása komplexen és részeire bontva is. A megállapítások nyomán az edzéseken változtatni lehet a mozgásokat, így a teljesítmény fokozható.



A fázisrajzok a valóságnak megfelelően

A fázisrajzok fázisonként önálló képre bontva

Borza József 2 danos mester egyenes vonalú (mae geri) rúgása. A pálcikafigurákat is a mikrogép rajzolta ki

Árnyékolják a képernyősugárzást!

A starnbergi (NSZK) Nokia Information Systems vállalatnál kifejlesztett mágnestekercses és földeléses rendszerrel sikerült csaknem teljesen megszüntetni a képernyőkből kilépő mágneses térsugárzást. A képernyőtől 30 cm-es távolságban a mágneses sugárzás az eddig mért értéknek csupán a tizede.

A képernyő felületének földelésével a villamos terek is megszüntethetők. Bár a képernyők sugárzási szintje tulajdonképpen veszélytelen, ez az olcsó megoldás mégis megnyugtathatja azokat, akik képernyő előtt dolgoznak.

Az okosak terve: „okos" város

Az „elektronizált ház" fogalmával már hazánkban is a gyakorlatban ismerkedhetünk. Például Budapesten, a Korona cukrászda épületében már üzemel az elektronikus házmester, felügyelve a liftet, a szellőzőmotorokat, a füst- és betörésjelző érzékelők jeleit.

Japánban már az „okos" városok, illetve városrészek tervezése van folyamatban. Arra gondolnak, hogy az új épületek lakóit egy számítógépes rendszer arról is értesíti, hol van éppen forgalmi dugó, üres parkolóhely vagy olyan áruház, amelyben nem tolonganak a vevők.

Pontvadászat

Új rejtvényt indítottunk útjára lapunkban, remélve, hogy elnyeri olvasóink tetszését. Minden számunkban két feladatot közlünk: az első logikai, matematikai tudást, a második számítástechnikai alapismereteket is igényel.

A feladatok után közöljük az elérhető maximális pontszámot. A részmegoldásokat is pontozzuk.

A pontgyűjtést, vagyis a pontvadászatot az esztendő végén zárjuk. A legjobb tíz versenyző nevét magazinunkban közzé is tesszük, ők lesznek azok, akik könyvutalványt kapnak.

A helyes megoldások a feladatok közreadása után két lapszámmal később jelennek meg, így a pontvadászoknak jut idejük a gondolkodásra.

Beküldési határidő: 1988. május 16.
Címünk: Mikroszámítógép Magazin Szerkesztősége
1371 Budapest, Pf. 433.

Jó vadászatot kíván a feladatok összeállítója:
dr. Hoffmann Tibor

1. feladat

Lehetőleg teljes indukcióval mutassuk ki, hogy az egész számok bármely hatványának összege mindig osztható n(n+1)-gyel! (5 pont)
E feladat megoldásának gondolatmenete hasonló az 1988/3. számban megjelent 1. feladatéhoz.

2. feladat

Ismeretes, hogy a szubrutinok alkalmazása mindig több futási időt igényel, mint a szubrutinok nélküli futás. Ugyanakkor azonban a program memóriahelyet takaríthat meg.

Egy bizonyos számítógépen egy *p* számú paraméterrel (felhasznált változóval) rendelkező szubrutinnak az egyszeri behívásához $5p+8$ memóriahely szükséges. A szubrutin magja *m* memóriahelyet foglal el, s ennek a memóriában szubrutinként való elhelyezéséhez további $2p+4$ memóriahely szükséges.

Annak a feltételezésével, hogy az egyes memóriahelyek egyenértékűek, állapítsuk meg, hogy adott *p* és *m* mellett hány hívásnál több esetén lesz gazdaságos a szubrutin felhasználása? (4 pont)

Vizsgáljuk ezzel meg a szubrutinok alkalmazási gazdaságosságát $p=0,1$ és 2 paraméter esetében. (1 pont)

Az 1988/2. szám feladatainak megoldása

1. feladat

Ha a sorozatból a két egymás utáni elem négyzetének különbsége páros szám, akkor ezek két, egymástól 2-vel különböző számból keletkeznek. Így

$$(a+2)^2-a^2=4a+4$$

A megadott első számmal így *142187* és *142189* adódik. (2 pont)

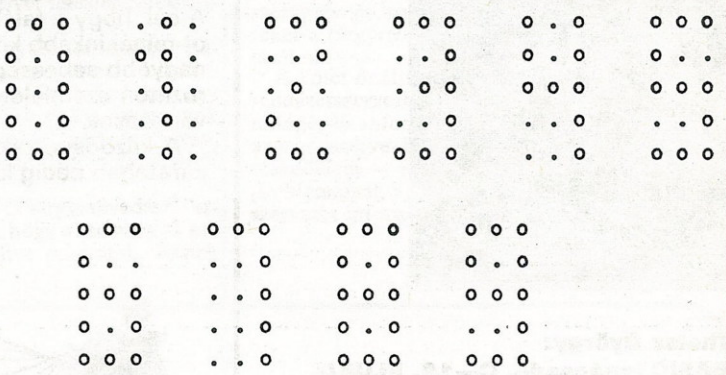
Ha ez a különbség páratlan, akkor a sorozatban ez két egymás utáni számot jelent, tehát

$$(a+1)^2-a^2=2a+1$$

A megadott második számmal ekkor *284377* és *284378* adódik. (2 pont)

2. feladat

A számjegyek megszokott képe alapján a magasság mindig nagyobb, mint a szélesség. A szélességben *n* nem lehet *2*, mert ebben az esetben nem tudunk olyan számjegyet kirajzolni, amely valamilyen „üreges" belső részt használ, például *0, 2, 3, 5, 6, 8, 9*.
Az üreges jelleg ábrázolásához tehát minimálisan $n=3$ kell. De ez elég is, amint a következőkben látni fogjuk.
Magasságban a középvonal elválaszthatósága például a *3* ábrázolásánál azt adja, hogy legalább $m=5$-nek kell lennie. Mint az ábrából látható, ez elegendő is. Tehát 3×5-ös méret a lehető legkisebb. (3 pont)



Az ábráról leolvashatjuk, hogy a felhasznált pontok száma az egyes számjegyeknél az alábbi:

„0" = 12	„5" = 11
„1" = 6	„6" = 12
„2" = 9	„7" = 7
„3" = 10	„8" = 13
„4" = 9	„9" = 12

(1 pont)
Mivel a legnagyobb pontszámot a „8" számjegy használja fel, a mátrixnyomtatók tesztelése a 8-as számokkal célszerű. Mint azt a mátrixnyomtatókat használó kollégák tudják, ez nemcsak erre a méretű mátrixra van így, hanem a gyárak a nagyobb méretűekre is a csupa 8-as számjegyekből álló számot adják meg hivatalos tesztnek. (1 pont)

Buborékok
(Felföldi Anna)

A gép is ember

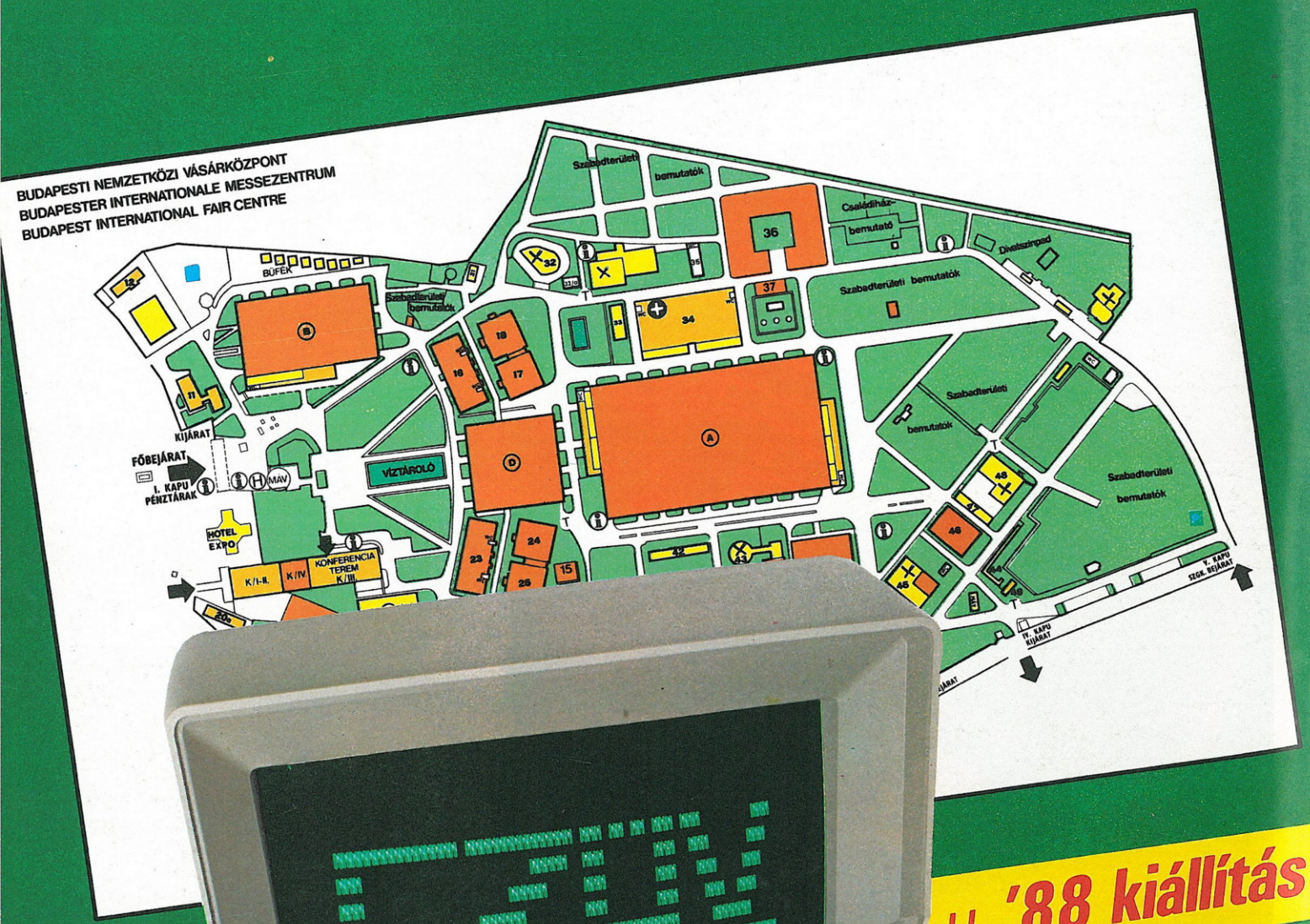
A Neumann János Számítógéptudományi Társaság karikatúrapályázatára nyolc országból százöt karikaturista háromszáz rajzot küldött. A nagydíjat (25 000,— Ft) Vjekoslav Klaič, Jugoszlávia nyerte.

A hagyományos módon rajzolt karikatúrák kategóriájában
I. Jurij Koszobukin, Szovjetunió (15 000,— Ft),
II. Balázs-Piri Balázs, Magyarország (10 000,— Ft),
III. Louis Postruzin, Ausztrália (5000,— Ft).

A számítógéppel készített karikatúrák kategóriájában
I. Halász Géza, Magyarország (15 000,— Ft),
II. Felföldi Anna, Magyarország (10 000,— Ft),
III. Kiss Katalin, USA (5000,— Ft).

Néhány karikatúrát, függetlenül a helyezésektől, alkalomszerűen közreadunk.

BUDAPESTI NEMZETKÖZI VÁSÁRKÖZPONT
BUDAPESTER INTERNATIONALE MESSEZENTRUM
BUDAPEST INTERNATIONAL FAIR CENTRE



μ '88 kiállítás

24. pavilon

KSH Számítástechnikai és Ügyvitelszervező Vállalat