Ára: 31 Ft

μ mikro számítógép magazin

ÚJ GESTA HUNGARORUM?

1990/2

8. ÉVFOLYAM
1990/2. SZÁM

mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

**Címlapunk:
Velekei József Lajos
munkája**

90 – 6720 – Nyírségi Nyomda, Nyíregyháza
Felelős vezető:
Jáger Zoltán

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 115-4250

Levélcím:
1371 Budapest Pf. 433

Kiadja:
MTESZ Neumann János Számítógéptudományi Társaság
1054 Budapest, Báthori u. 16.

Levélcím:
1368 Budapest 5. Pf. 240

Telefon: 132-9349

Felelős kiadó:
Tóth Istvánné ügyvezető főtitkárhelyettes

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

Megjelenik havonta.
Egy szám ára 31,— Ft
Előfizetési díj:
egy évre 372,— Ft
fél évre 186,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, Pf. 149
és a Magyar Média
1932 Budapest, Pf. 279
88—1135

INDEX: 25 629
ISSN 0236-6088

TARTALOM

GESTA, Magyarország története 1990/91-ben

Javaslat az angol Domesday projekt itthoni megismétlésére

„Minden gyereknek lesz egy otthoni számítógépe. Ez a gép összekapcsolható lesz egy hálózaton keresztül a saját iskolájának a számítógépével, a helyi tanulórendszerrel, valamint a világ bármelyik adatbázisával. Ráadásul kompaktlemezeket tartalmazó elektronikus tanulócsomagokat fognak használni, interaktív audio- és videoanyagokkal. Az elektronikus üzenetváltási lehetőség kapcsolatot fog teremteni a tanárok és a diákok — és ami még fontosabb — a diákok és diákok között. (...) A diákok a házi feladatot mind elektronikus, mind pedig írott formában elkészíthetik. (...)

A gyerekek majd azért mennek az iskolába, hogy játsszanak egymással, hogy társadalmi életet éljenek, sportoljanak, (...), táncoljanak (...) stb. Röviden, az otthon lesz a tanulás helye, míg az iskola a játéké."

(Tom STONIER: Computers and the Future of Education. ECCE'88 — Lausanne)

Egyszer az egyik barátom azt rótta fel bűnömül, hogy sok mindent még félig kész állapotban „kifecsegek", aztán, ha nem sikerült valamit megcsinálnom, éghet a bőr a képemen. Ez nálam valóban így is van, rettenetesen szeretem ugyanis a kollektív munkát, amikor van egy félig „megsült" gondolat, aztán összejön néhány „bolond" (nak mondott) ember, elkezd beszélgetni, egymás után szállnak fel az ötletrakéták, és a végén létrejön valami, amit, ha van hozzá elegendő lelkes ember, meg lehet csinálni. Ha persze az eredeti ötlet senkit sem lelkesít, illetve nincs, aki leüljön és összedugja másokkal a fejét, akkor jön a katasztrófa, és az ötlet a saját hamvába hull.

Nem mondanék sokat az angol Domesday projektről, amelyet az angol oktatásügy kezdeményezésére a Domesday kódex születésének a 900. évfordulójára az angol iskolák diákjai összehoztak, ti. a Mikroszámítógép Magazin 1988. februári számában a projektről egy rövid leírást már egyszer közöltünk. Azoknak, akik nem olvasták az írást, röviden annyit, hogy 1086-ban Hódító Vilmos felkérte az ország írástudóit, ragadjanak tollat, ecsetet, papírt, írják és rajzolják le környezetüket, hogyan élnek, mit csinálnak, milyenek a társadalmi körülmények, hogyan ünnepelnek, hogyan gyászolnak, milyen állatok vannak a környezetükben, hogyan gazdálkodnak, és még sok minden mást, amit a korabeli Anglia megörökítésére méltónak tartanak. Így jött létre egy könyv, amelynek szinte minden lapját más írta, egy adott időszak Angliájának tökéletes leírása.

Ezt a Hódító Vilmos-i tettet ismételte meg az angol kormány, amikor meghirdette ismét a Domesday projektet, de már az 1980-as évek csúcstechnikáját alkalmazva, ami nagyjából azt jelentette, hogy az információt (írást, rajzokat, térképeket, fényképeket és videofelvételeket) számítógépen gyűjtötték össze és szerkesztették meg, a válogatott, lektorált, tördelt és adatbázisba rendezett anyagot két videolemezen rögzítették.

Amikor ezt a történetet az egyik barátomnak elmondtam, és az elbeszélésben körülbelül itt tartottam, már kívülről fújta a folytatást: „ ... és ugyebár most elhatároztad, hogy ezt itt nálunk is meg kell csinálni. Még mielőtt folytatnád, arra válaszolj, hogy mindehhez ki adja a pénzt?"

A kérdést persze megválaszolatlanul hagytam, és elmondtam a hazai megvalósításra vonatkozó terveimet, amit nyilvánvalóan a hazai technikai adottságokhoz kellett igazítanom, azaz az eléggé nem szidható Commodore gépekkel felszerelt iskolákhoz, a gyakorlatilag elérhetetlen videolemezjátszókhoz, ugyanakkor azonban figyelembe vettem a nagyon gyorsan kialakítható, videotex alapú iskolai számítógéphálózatot. Még mielőtt bárki elhamarkodott kritikával illetné elmebeli állapotomat, én valóban azt állítom, hogy nagyon rövid időn belül — két évre becsülöm ezt a periódust — Magyarországon valamennyi iskola összekapcsolható lesz részben az ún. „Közművelődési és oktatási adatbázissal", részben pedig egymással.

Ennek a nagyvonalú tervnek az alapja egy hármas együttműködés, az Eszterlánc Rt., a Müszi és a SZÁMALK között, ti. ez a három cég megvásárolta a grazi IIG-től (Institut für Informationsverarbeitung Graz) az ún. szoftver-dekódert, amelynek a segítségével bármelyik PC vagy klón egy modemen keresztül, mindenfajta hardverkiegészítés nélkül a VTX erőforrás-számítógépekkel kommunikálhat. Az együttműködők — dicséretes módon — elhatározták, hogy a szoftvert az iskoláknak majdnem ingyen, névleges áron adják, a gazdálkodó szervezeteknek viszont „árban", így a kettő majd kiegyenlíti egymást, és ami a leglényegesebb, az iskolákat egy, nálunk igen fejlettnek számító kommunikációs lehetőséghez juttatják. A GESTA projektet erre a technikai lehetőségre terveztem, a videotex-hálózat lehet az az eszköz, amivel az iskolák tanulói és persze a tanárai, sőt a tanulók hozzátartozói is összegyűjthetik az információt és tárolhatják azt.

A GESTA projekthez — szerencsére — még a történelmi dátumok is jól illeszkednek. A mi Domesday kódexünk ugyanis a Gesta Hungarorum, amelyről az Új magyar lexikon azt írja, hogy 1091—92 körül keletkezett, így az általam megírásra javasolt periódus végén, 1991-ben mi az elektronikus módon szerkesztett és kiadott GESTA HUNGARORUM-mal ünnepelhetjük az eredeti kódex 900 éves évfordulóját. A művelődési miniszternek írt javaslatomban az is szerepel, hogy Magyarország életében az elkövetkezendő két év korszakos jelentőségű lesz, mind politikai, mind gazdasági szempontból. Milyen izgalmas pedagógiai kísérlet lesz majd ezeket a változásokat diákok által írt esszék formájában cenzúrázatlanul viszontlátni, azt, hogy hogyan is tükröződnek ezek a változások a különböző korosztályokhoz tartozó gyerekek lelkében.

Az iskolák VTX terminálokkal való ellátása nyilvánvalóan nem megy egyik napról a másikra, ezért a projekt a meglévő, különböző típusú iskola-számítógépeknek, főleg a Commodore-oknak a projektekbe való bekapcsolásával is számol. Valószínű ugyanis, hogy a diákok egy része az általa kiválasztott témát a videotex-hálózathoz nem kapcsolt saját vagy iskolai számítógépen dolgozza fel, és lemezen küldi be. Nyilvánvalóan a szerkesztőbizottság ezeket a lemezeket is fogadja és átfordítja PC-formátumra.

A projekt egyes szerkesztőbizottságait el kell látnunk például scannerekkel is, ti. még mindig nagyon sok olyan diák van, aki egyáltalán nem jut számítógéphez. Ezek a diákok nagy valószínűséggel hagyományos módon, például írógépen készítik el dolgozataikat, amelyeket nagyon jó lenne gépi úton a tárolóba beolvasni. Ebben remélem, hogy volt intézetem, az SZKI segíteni fog, és ingyen vagy nagyon névleges áron rendelkezésünkre bocsátja a most már igazán világhírű Recognita szövegfelismerő rendszert. Persze a legkedvesebb adathordozónk a telefondrót lesz, amelyen a szöveget és a rajzokat a legegyszerűbben lehet a központi tárba továbbítani. Amire még szükség van, azt már szinte meg sem kell említeni: például a képdigitalizáló és persze a központi VTX számítógép, amelynek a beszerzésére az első lépések már megtörténtek.

A GESTA projektről elbeszélgettem néhány osztrák kollégával is, akik annyira lelkesültek a tervért, hogy igen komolyan gondolkoznak az osztrák projekt elindításáról. Miután a mi VTX hálózatunk majdnem teljesen kompatibilis az osztrák VTX hálózattal, igen célszerű lenne a két hálózat és nyilvánvalóan a két projekt összekapcsolása is. Ha még azt is figyelembe vesszük, hogy ennek a közös projektnek a „végeredményét" a Bécs—Budapest világkiállítás alkalmából is használni lehet, akár a kiállítás információs rendszerének az egyik adatbankjaként, akkor nem is olyan elképzelhetetlen, hogy a projektet a kormány is támogatni fogja.

Arról sem szabad elfeledkeznünk, hogy az osztrák, a német és a luxemburgi posta november elején összekapcsolta VTX rendszereit, és így az osztrák és magyar rendszernek az összekötésével megnyílik a lehetőség további két országgal való kooperációra. Ha még ők is csatlakoznak a GESTA projekthez, akkor már egy egész csinos közép-európai adatbázis elérésére lesz lehetőségük a VTX-előfizetőknek, különösen akkor, ha mindegyik ország az adatbázisból kiválasztott anyagokból létrehoz egy angol nyelvű változatot, amelyet nemzeti modulokból lehetne fokozatosan egyre tovább építeni.

A GESTA projektet csak akkor lehet megvalósítani, ha az adatgyűjtés, a feldolgozás és tárolás valódi mozgalommá válik. Az jár a fejemben, hogy minden tisztességes mozgalom egy megkülönböztető jelvényt ad a tagjainak, hogy az összefogásnak legyen bizonyos külső megnyilvánulása is. Én nem hiszem, hogy a GESTA mozgalom tagjai jobb jelvényt találnának, mint egy Anonymus-figurát, amelyet mindenkinek eljuttathatnánk, aki már beküldött valamilyen információt a szerkesztőségnek.

Talán még annyit érdemes elmondani, hogy a GESTA tervezésébe érdemes lesz bevonni a Domesday projekt szervezőit is, sőt talán ki sem kellene találnunk egy új adatstruktúrát, el kellene fogadnunk az általuk használt rendszert, és akkor nagyon egyszerűen kialakítható lenne a rendszeres kapcsolat a már említett három országon kívül Angliával is.

Végezetül még egyszer az anyagiakról. Az a szándékom, hogy egy alapítványra teszek javaslatot, amelyből elsősorban nem a mozgalom készkiadásait lehetne fedezni, hanem inkább a GESTA-ba író, rajzoló, fényképező és videózó diákok, tanárok, ifjak és még ifjabbak munkáját lehetne megjutalmazni, mert én nagyon erősen hiszek abban, hogy ez a mozgalom elindítható, és sikeresen be is fog fejeződni a GESTA HUNGARORUM elkészültének 900. évfordulójára.

Kovács Győző

PROLÓGIA

II. RÉSZ

Napjainkban a programozás egyik legizgalmasabb kalandja a Prolog. Előzményei az 1970-es évek elejére nyúlnak vissza, amikor is a kutatás arra irányult, hogy a programozási munka nagyobb részét a számítógép végezze el. A Prolog ennek a sokéves kutatómunkának az eredménye. Az első hivatalos Prolog-verziót a marseilles-i egyetemen fejlesztették ki Alain Colmerauer vezetésével az 1970-es évek elején. Ők adták a nevét is (PROgramming in LOGic). Ez sokkal eredményesebb és hatékonyabb lett, mint az addig ismert legtöbb programozási nyelv, köztük a Pascal és a BASIC. Például egy adott alkalmazás Prologban általában tízszer kevesebb programsort igényel, mint Pascalban. Olyan nyelv gondolatai körvonalazódtak, amely szabályok és tények alapján dolgozik, látszólag önállóan oldja meg a problémát. Ma a Prolog talán a legfontosabb eszköze a mesterséges intelligenciát alkalmazó programozásnak és a szakértői rendszereknek.

A PROGRAMOK VÉGREHAJTÁSA

A cél két fontos, a nyelvbe beépített mechanizmus révén érhető el. Az egyik a mintaillesztés, amely két kifejezés (az elérendő cél és a rendelkezésre álló szabályok egyike) közötti, argumentumonkénti összehasonlításra szolgál. Az illesztés két esetben lehet sikeres: ha mindkét illesztendő argumentum konstans, és azok pontosan megegyeznek, vagy ha az egyik argumentum változó, de a másiknak van határozott értéke, tehát vagy konstans, vagy értékkel bíró változó.

A másik mechanizmus a backtrack algoritmus, amely a megoldáshalmaz összes elemét igyekszik megtalálni. Ha a lehetséges összes (jó és rossz) megoldást egy megoldási (döntési) fa ágaiba és leveleibe helyezzük, akkor az említett algoritmus a fa módszeres bejárását végzi, minden megoldás után visszalépve az utolsó elágazásra, ahonnan újra próbálkozik másik megoldást találni. Ez egy változó szempontjából azt jelenti, hogy új értékhez csak abban az egyetlen elágazási pontban juthat, ahol először határozott értéket kapott.

A célkifejezésről előbb-utóbb kideríti a Prolog rendszer, hogy az adott körülmények között teljesül-e vagy nem, illetve milyen feltétellel teljesülhet. A célkifejezés vizsgálatának kétféle módja van.

— Ha a célkifejezésben lévő szabály csak konstansokat tartalmaz, akkor a rendszernek nincs más dolga, mint addig helyettesítgetni ezeket a tényállításokat a beépített backtrack algoritmus segítségével, amíg a szabály nem teljesül, amikor is válaszként közli, hogy TRUE. Ellenkező esetben, tehát ha az összes lehetőség kimerült, és még mindig nem volt sikeres a keresés, a válasz FALSE.

— Ha a célkifejezésben van változó, akkor a rendszernek az a feladata, hogy a szabályok segítségével keressen legalább egy olyan értéket, amely kielégíti a célt. Bizonyos körülmények között az összes megoldást megkapjuk.

EGY EGYSZERŰ ELDÖNTÉSI FELADAT

Most érkeztünk el arra a pontra, ahol mutatnunk kellene már egy-két működő programot, mert ellankad az olvasó figyelme. Az első néhány programocska úgynevezett ,,programozási tétel" lesz. Egyszerű, a mindennapi programozási munkában sűrűn előforduló alapalgoritmusokról van szó, megoldásuk bevezet a Prolog programozásba.

Az eldöntés programozási tétele arra ad algoritmust, hogy egy adatstruktúrában van-e T tulajdonságú elem. Lássunk egy kész megoldást:

```
domains
   elem = symbol
   lista = elem*
predicates
   bennevan (elem, lista)
clauses
   bennevan (E, [E:_]).
   bennevan (E, [_:Lv]) if bennevan (E,Lv).
```

A *domains* szekció deklarál egy elem nevű, tetszőleges szimbólumból álló elemi objektumot, valamint egy lista nevű, az ,,elem" típusú tagokból álló listát. A lista vagy sorozat olyan adatszerkezet, amely már ismert típusú tagokból áll; ezek közül az első a lista feje, a többi a lista vége. Csak a fejben lévő elem érhető el közvetlenül. A lista jelölése lehet zárt és lehet bontott. Az első esetben csak egy szokványos változónév reprezentálja. A felbontott írásmód esetén szögletes zárójelekkel határoljuk, és a fejétől a végét a ¦ jel választja el. A lista végét vagy egy változónév vagy újabb kibontott lista képviselheti. Az ,,üres lista" jele [ ]. A listák legvégén implicite mindig üres lista áll. Így a lista feje egy elem, a vége pedig minden esetben maga is lista.

Az adatszerkezet definíciója tehát rekurzív!

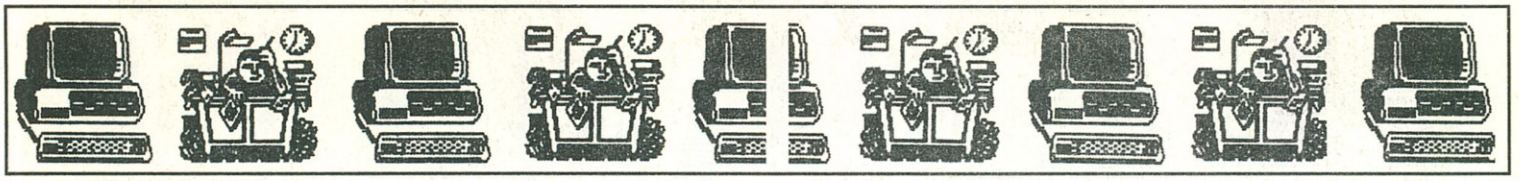
A Prolog legtermészetesebb adatstruktúrája a lista, így a feladatot egy kicsit egyszerűsítve úgy fogalmazzuk át, hogy van-e a listában egy bizonyos elem?

A *predicates* egyetlen szabályt deklarál, amelynek a neve ,,bennevan", két argumentummal, amelyek közül az első ,,elem" típusú, a második pedig ,,lista" típusú. A deklaráció után ragaszkodnunk kell a paraméterek ilyen sorrendjéhez.

Az a tény, hogy *nincs goal,* vagyis a programban meghatározott célkifejezés, azt eredményezi, hogy egy RUN parancsra megkérdezi a célt. Ezt beadva fog csak működni a program.

A *clauses* szekció tartalmazza a ,,bennevan" szabályait. Az első szabállyal nincs is semmi gond; az elem típusú, tehát szimbólumot tartalmazó E változó benne van a listában, ha az E a lista feje, akármi is áll a lista további részében. Aki írt már rekurzív programot, annak a második szabály nem különösebben meglepő: az E elem benne van a listában, ha a lista elsőn kívüli részében benne van. Ezzel a lehetséges eseteket szétválasztottuk, hiszen egy lista a lista fejéből és a lista elsőn kívüli elemeiből álló listából áll. A szabály nem rekurzív első sora elintézi az első esetet, a második, rekurzív sor pedig a másodikat.

Procedurálisan úgy képzelhetjük el a program működését, hogy megnézi a lista első elemét, és ha az egyenlő a keresett szimbólummal, akkor a ,,bennevan"-ság teljesül. Ha nem, akkor keres egy újabb szabályt, ami most azt mondja ki, hogy akkor teljesül a ,,bennevan"-ság, ha a lista maradék részére az teljesül. Tehát újra kezdi a vizsgálatot, már egy rövidebb listával, és természetesen újból az első szabályt próbálja alkalmazni. Ha valaha teljesül az egyezés, akkor (örömében...) visszalép a rekurzió egy magasabb szintjére, ahol most



pontot talál. Ennek az a jelentése, hogy nincs további teendő, tehát ismét visszalép, egészen addig, amíg a parancs szintig el nem ér. Itt kiírja, hogy True, ami angolul „igaz"-at jelent.

Ha a sorozatos rekurzív hívások által annyira lerövidül a lista, hogy már nincs benne semmi, csak az üres lista, akkor (bánatosan) konstatálja, hogy nincs meg a keresett elem a listában, majd a rekurzív hívásokból visszalépve azt írja a képernyőre, hogy False. Szemléltetésül bemutatom a fenti program futását a

Goal: bennevan ("f",["a","b","f","c"])

céllal, a TRACE bekapcsolásakor adott visszajelzésen keresztül:

```
CALL: bennevan("f",["a","b","f","d"])
REDO: bennevan("f",["a","b","f","d"])
CALL: bennevan("f",["b","f","d"])
REDO: bennevan("f",["b","f","d"])
CALL: bennevan("f",["f","d"])
RETURN:*bennevan("f",["f","d"])
RETURN: bennevan("f",["b","f","d"])
RETURN: bennevan("f",["a","b","f","d"])
True
```

Most egy olyan célállításról kérdezzük a gép véleményét, amely szemmel láthatóan kudarccal fog végződni:

Goal: bennevan("z",["a","b","c"])

A nyomkövetés eredménye az alábbi lista:

```
CALL: bennevan("z",["a","b","c"])
REDO: bennevan("z",["a","b","c"])
CALL: bennevan("z",["b","c"])
REDO: bennevan("z",["b","c"])
CALL: bennevan("z",["c"])
REDO: bennevan("z",["c"])
CALL: bennevan("z",[ ])
REDO: bennevan("z",[ ])
FAIL: bennevan("z",[ ])
False
```

Megjegyzés. Prologban is lehet ugyanarra a feladatra többféle megoldás, ez általában a programozótól függ. A végeredmény eleganciája viszont már sokat sejtet a megoldás jóságáról és a programozó gyakorlatáról, de főleg a képességéről. Gyanítom, hogy az én itt közreadott programjaim sem mindig a legtökéletesebbek, és lehet azoknál jobbakat írni.

ELEM TÖRLÉSE LISTÁBÓL

Az első és legfontosabb lépés meghatározni a feladat kívánt értelmezését. Világos, hogy egy listából az E elem minden előfordulását törlő programnak három argumentuma lesz: a törlendő E elem, az E elemet akár többször is tartalmazó L1 lista és az E elemet sehol sem tartalmazó L2 lista. Az alkalmas reláció alakja tehát: torol(L1,X,L2). Szöveges értelmezése pedig az, hogy L2 minden esetben Ł1, elhagyva E minden előfordulását. Például a torol ([a,b,c,b],b,L) feladat megoldása L=[a,c,].

A program procedurális nyelvekben feltétlenül ciklusos megoldást követelne. Prologban a ciklusokat rekurzióval valósítjuk meg.

Vegyük elő most a *procedurális* gondolkodásmódunkat! Kezdjük a rekurzív részszel. A listákra általánosan használt rekurzív forma az [E;Lista]. Két lehetséges eset van: az egyik, amikor E a törölni kívánt elem, a másik, mikor nem az. Az első esetben E-t rekurzív úton törölve a Lista listából kapjuk a megoldást. Az alkalmas szabály tehát:

```
torol([E;Lista],E,Maradek])
if torol(Lista,E,Maradek).
```

Gondolkodásmódot váltva, a szabály *deklaratív* olvasata ez: E-nek [E;Lista]-ból való törlése Maradek-ot ad eredményül, ha E Lista-ból való törlésének eredménye Maradek. A feltétel, hogy a lista feje és a törlendő elem ugyanaz, a szabály fejében levő E változó által biztosított, hiszen ha a szabály igaznak bizonyul, akkor azonos változónév helyén csak azonos értékek állhatnak.

A második esetben, ahol a törlendő elem különbözik F-től, azaz a lista fejétől, egyszerű. A kívánt eredmény egy lista, amelynek a feje F és farka az E elemtől megtisztított Lista.

```
torol([F;Lista],E,[F;Maradek])
if E<>F and torol (Lista,E,Maradek).
```

A szabály deklaratív olvasata: E törlése [F;Lista]-ból [F;Maradek], ha E különbözik F-től és E Lista-ból való törlésének eredménye Maradek. Az előző szabállyal ellentétben a feltétel, hogy a lista feje és a törölni kívánt elem különbözik, explicit módon ki van fejezve a szabály törzsében, különben az E és az F változók felvehetnék ugyanazt az értéket is.

Az alapeset (vészkijárat...) triviális. Üres listából nem lehet elemet törölni, és az elvárt eredmény szintén az üres lista. Ez adja az utolsó szabályt:

```
torol([ ],_,[ ]).
```

Az aláhúzásjel a középső paraméter helyén azt jelenti, hogy itt akármi állhat, a szabály szempontjából közömbös. Így lehet kivédeni a rendszer egy hibaüzenetét, amely szerint a torol([ ],E,[ ]). szabályban az E csak egyszer szerepel, így definiálatlan. A teljes program tehát:

```
domains
  elem = symbol
  lista = elem*
predicates
  torol(lista, elem, lista)
clauses
  torol([E;Lista],E,Maradek]) if torol(Lista,E,Maradek).
  torol([F;Lista],E,[F;Maradek]) if E<>F and torol (Lista,E,Maradek).
  torol([ ],_,[ ]).
```

RENDEZÉS PERMUTÁLÁSSAL

A listák rendezésének logikai specifikációja az, hogy keressük a lista olyan permutációját, amely rendezett. Ez azonnal leírható, mint egy logikus program. Az alapreláció: rendperm(Xs,Ys), ahol Ys az Xs elemeiből álló növekvően rendezett lista:

```
rendperm(Xs,Ys) if permutacio (Xs,Ys)
and rendezett (Ys) and !.
```

A felkiáltójel (cut) letiltja a további megoldások keresését, hiszen most annak nincs értelme.

A rendezés legfelső szintjét lebontottuk, a felülről lefelé haladó programfejlesztés szellemében. Most már csak a *permutacio* és a *rendezett* szabályokat kell definiálnunk. Annak eldöntése, hogy egy lista növekvően rendezett-e, kifejezhető az alábbi két szabállyal. Az első kijelentés szerint az egyelemű lista szükségszerűen rendezett. A másik szabály szerint egy lista rendezett, ha az első eleme kisebb vagy egyenlő, mint a második, és a maradék lista a második elemtől kezdve rendezett.

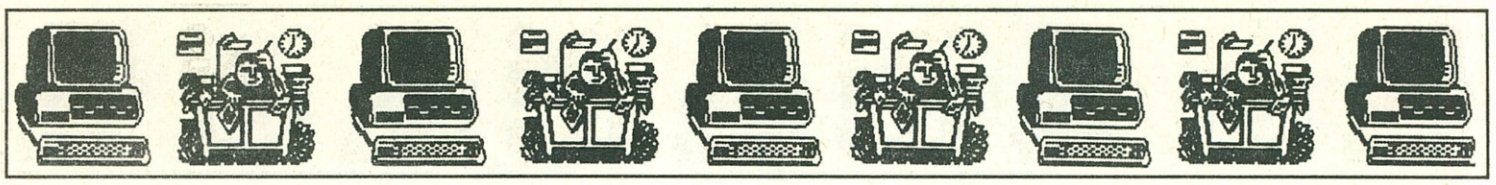
```
rendezett([_]).
rendezett([X;[Y;Ys]]) if X<=Y and rendezett ([Y;Ys]).
```

A permutálást végző program még érdekesebb. A permutálási eljárás egy megoldása az lehet, hogy kiemelünk egy elemet a listából, azt a lista elejére helyezzük, majd rekurzívan permutáljuk a maradék listát. A második, nem rekurzív kijelentés szerint az üres lista egyetlen permutációja önmaga.

```
permutacio(Xs,[Z;Zs]) if kiejt(Z,Xs,Ys)
and permutacio(Ys,Zs).
permutacio([ ],[ ]).
```

Szólni kell a *kiejt* szabályról. Ennek szerepe az, hogy a lista elejétől kezdve a vizsgálatot, az első kívánt elemet kiemelve adja vissza a bemenő listát. Tehát ez nem pontosan azonos azzal a programmal, amely az összes előfordulást kitörli, csak hasonlít rá. Deklaratív olvasásban így hangzik: E-t kiemelhetjük az [E;Lista] listából a Lista visszaadásával, vagy E-t kiemelhetjük az [F;Lista1]-ből, ha megadjuk az [F;lista2] listát, ahol a Lista2 egyenlő az E elem Lista1-ből való kiemelésének eredményével.

Érdemes most áttekinteni a teljes programot:



```
domains
  elem=symbol
  lista=elem*
predicates
  permutacio(lista,lista)
  kiejt(lista,elem,lista)
  rendezett(lista)
  rendperm(lista,lista)
clauses
  rendperm(Xs,Ys) if permutacio(Xs,Ys) and rendezett(Ys) and !.

  permutacio(Xs,[Z¦Zs]) if kiejt(Xs,Z,Ys) and permutacio(Ys,Zs).
  permutacio([ ],[ ]).

  kiejt([E¦Lista],E,Lista).
  kiejt([F¦Lista1],E,[F¦Lista2]) if kiejt(Lista1,E,Lista2).

  rendezett([_]).
  rendezett([X¦[Y¦Ys]]) if X<=Y and rendezett([Y¦Ys]).
```

Futtatáskor például a következő célt adhatjuk meg:

Goal: rendperm([c,a,b],R)

A válasz remélhetőleg ez lesz:

R=[a,b,c]

1 Solution (1 megoldás született)

A permutációs rendezés sajnos nem túl szerencsés megoldás, mert a permutációk előállítása csak 1/n!-sal növeli a rendezettség elérésének valószínűségét lépésről lépésre. A permutációk sorában ,,egyszer csak" felbukkan az ,,igazi", és ennek ugyanakkora a valószínűsége, mint bármely másik permutációnak.

GYORS RENDEZÉS (QUICKSORT)

Az egyik leghatékonyabb rendezési algoritmus a Quicksort. Elve az, hogy egy elemet kiválasztva, kettéosztjuk a listát nála kisebb és nála nagyobb elemekből álló részlistákra, majd ezeket rendezzük, végül újra egyesítjük a kisebbek listáját, a kiemelt elemet és a nagyobbak listáját. Már az elve is rekurzív, így igazán hatékonyan csak a rekurziót megengedő programozási nyelvekben valósulhat meg.

Kezdjük az ismerkedést a *szetval* szabályrendszerrel, amelynek az a feladata, hogy az első paraméterként megadott [F,L1] lista F elemét elhelyezze vagy a kisebb-egyenlők közé (1. szabály), vagy az F-nél nagyobbak közé (2. szabály) attól függően, hogy melyik reláció teljesül. Mielőtt azonban elhelyezné, egy rekurzív hívásra kényszeríti önmagát, hogy az L2 és az L3 listák már rendelkezésre álljanak. Ha az első paraméter az üres lista, akkor persze nincs mit tenni, két üres listát ad vissza eredményként.

Másik szabálycsoportunk az *osszerak* nevű, amely szintén ravasz módon dolgozik. Feladata, hogy egyesítsen két listát. Az is megengedett, hogy azonos elemek legyenek az eredménylistában, ellenben ügyelni kell a sorrendre, amit úgy old meg, hogy sorozatos rekurzív hívásokkal ,,kiüríti" az első argumentumban adott listát, oda helyezi a nagyobbak listáját (ezzel teljesül a 2. szabály), majd az első lista végéről kezdve a rekurzív lépésekből sorra visszatérve alakítja ki az eredményt, az elemeket egyenként hozzáírva az eredménylista elejére. Az utolsó rekurzív hívásból való visszatérés után a harmadik, mondhatnánk ,,eredményparaméterben" lesz az összesített lista. Maga a program az alábbi:

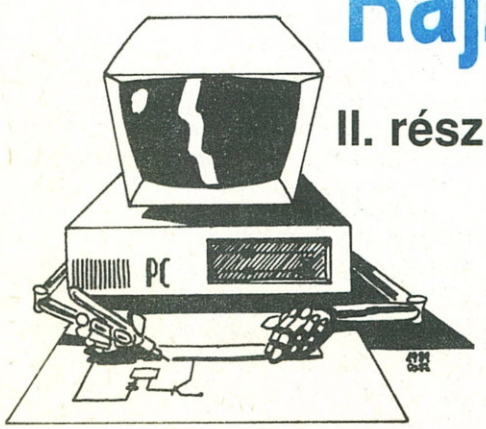
```
domains
  elem = symbol
  lista = elem*
predicates
  rendquick(lista,lista)
  szetval(lista,elem,lista,lista)
  osszerak(lista,lista,lista)
clauses
  szetval([F¦L1],E,[F¦L2],L3) if F<= E and szetval(L1,E,L2,L3).
  szetval([F¦L1],E,L2,[F¦L3]) if F> E and szetval(L1,E,L2,L3).
  szetval([ ],_,[ ],[ ]).

  osszerak([E¦L1],L2,[E¦L3]) if osszerak (L1,L2,L3).
  osszerak([ ],Lista,Lista).

  rendquick([E¦L1],L2) if szetval (L1,E,Kis,Nagy) and
                          rendquick(Kis,Kisrend) and
                          rendquick(Nagy,Nagyrend) and
                          osszerak(Kisrend,[E¦Nagyrend],L2).
  rendquick([ ],[ ]).
```

Makány György

Rajzoljunk számítógéppel!



II. rész — Újat rajzoljunk vagy javítsunk?

A rajzok módosításának és átszerkesztésének egyebek mellett jó eszközei az ún. blokkműveletek. A blokkműveletek a rajz egyes, általunk megválasztott egységeinek kezelésére szolgálnak. A megjelölt blokkokat többek között mozgatni, fájlba írni, onnan behozni, másolni is lehet.

A blokkműveletek megismeréséhez hívjuk meg a főmenüből a BLOCK parancsot. Ekkor a képernyőn megjelenik a BLOCK almenü:

MOVE
DRAG
FIXUP
GET

SAVE
IMPORT
EXPORT

Ezek közül a MOVE a még huzalozás nélküli elemek mozgatására alkalmazható célszerűen. Ha huzalozott rajzokon kívánunk blokkokat mozgatni, a MOVE-val mozgatott blokkban lévő alkatrészekhez csatlakozó vezetékek is elmozdulnak úgy, hogy a vezetékkötések meg is szakadnak. Ezt a hibát eddigi ismereteink birtokában már ki tudjuk javítani (a PLACE főparancsban a WIRE funkcióval a hiányzó vezetékeket pótolhatjuk), de a vezetékekkel együtt történő mozgatást kedvezőbben is meg lehet oldani. Ezt később, a DRAG alparancs tárgyalásánál fogjuk bemutatni.

A blokkmozgatás tanulmányozásához hívjunk be egy huzalozás nélküli alkatrészeket tartalmazó rajzfájlt (mi az *előző számban* megjelent cikk 1. ábráján látható rajzot fogjuk használni; ha ilyen fájlunk nincs, készítsünk egy egyszerű rajzot)! Ehhez válasszuk a főparancsmenüből a QUIT-et, majd az almenüből az INITIALIZE alparancsot. A feltett kérdésre írjuk be a fájlnevet, ennek eredményeképpen az ábrán látható rajz betöltődik. Zoomozással állítsuk be a számunkra kedvező méretarányt, majd hívjuk a BLOCK főparancsot és ezután a MOVE alparancsot! Ekkor a következő menü jelenik meg:

Begin
Find
Jump
Zoom
escape

A következő lépés a mozgatni kívánt blokk megjelölése. A blokk a munkalapon egy téglalap alakú terület, amelyet egy átlójának két végpontjával határozhatunk meg. Vigyük a kurzort a tervezett terület valamelyik ,,sarkához". Válasszuk a menüből a Begin funkciót. Ekkor a menüben a Begin helyett End jelenik meg. Mozgassuk a kurzort az átló másik végpontjára és válasszuk az End-et! Ezzel a blokk kijelölése megtörtént, amit a munkalapon is láthatunk: a kijelölt blokk keretben jelenik meg a képernyőn. A menüben is változás van: az End helyett Place áll. A kerettel határolt területet a kurzorbillentyűkkel a kívánt helyre mozgathatjuk, majd a Place-szel ott ,,letehetjük". Ezt az állapotot az 1. ábra mutatja.

Huzalozott rajzokon a blokkok (például az előbbi példában egy tároló) mozgatása a vezetékkötések megtartásával a DRAG alparancs alkalmazásával oldható meg.

A feladat elvégzéséhez töltsük be az előbb említett 1. ábrán látható rajzot. Ha ez nincs meg lemezes fájlban, készítsünk egy egyszerű, vezetékeket is tartalmazó kapcsolást. Válasszuk a főmenüből a BLOCK főparancsot, majd a DRAG alparancsot. Mint ahogy ezt a MOVE gyakorlása során tettük, jelöljük ki a mozgatni kívánt területet. Legyen ez most a rajzon legalul lévő tároló és ,,környéke". Toljuk el a blokkot balra (10-12 raszterponttal — azaz a kurzor ugyanennyi lenyomásával). Már a mozgatás során látható, hogy a vezetékek eredeti csatlakozópontjaikat megtartva, ,,gumiszálként" változtatják helyzetüket. A kívánt pozíció elérésekor a blokkot a Place-szel rögzítsük. Ezt az állapotot szemlélteti a 2. ábra.

A vezetékek ,,rendbetételéhez" a BLOCK menü FIXUP alparancsa használható. Ennek meghívásával a következő menühöz jutunk:

Pick
Find
Jump
Zoom
escape

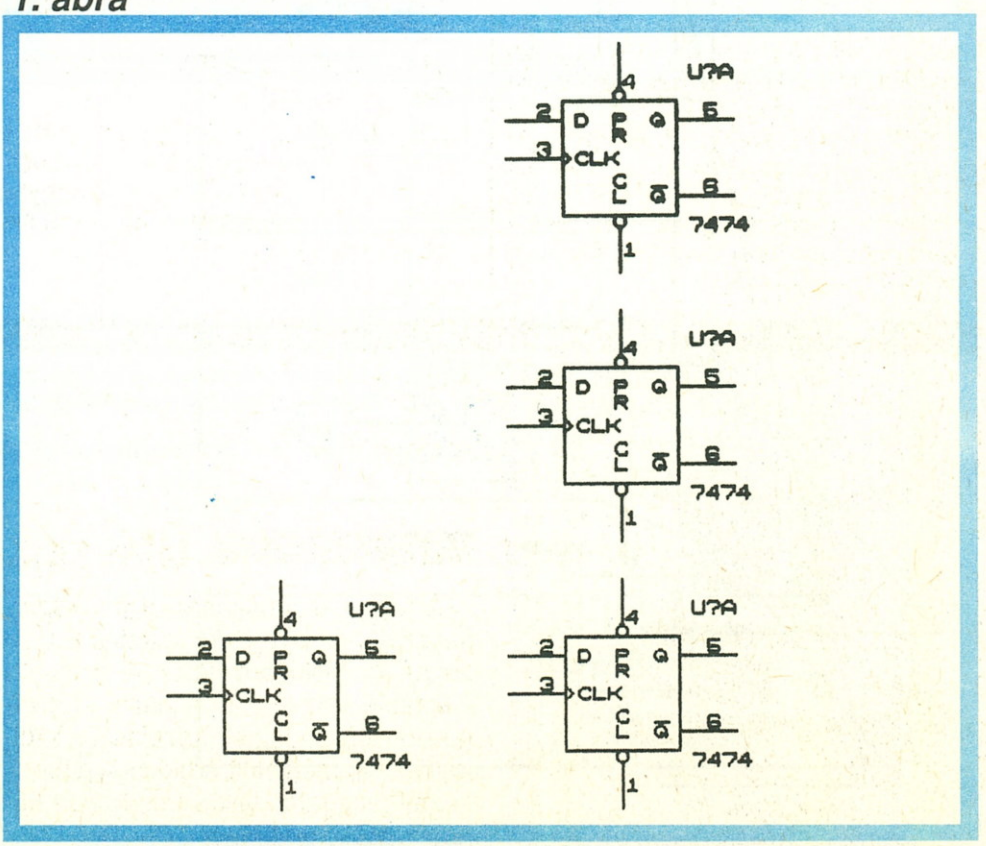
A kurzor mozgatásával álljunk egy olyan pontra, ahol a csatlakozó vezetéket javítani kívánjuk, majd hívjuk a Pick-et! Ekkor egy kételemű menühöz jutunk, amelyből válasszuk a Pick one-t! Az eredmény egy újabb menü, amely arra kérdez rá, hogy a csatlakozási ponthoz kapcsolt melyik vezetéket kívánjuk elmozdítani. Az OrCAD a felkínált vezetéket szaggatottan rajzolja az ernyőre. Ha ez megfelel, a menüben lépjünk a This-re, ha nem, akár a Next, akár a Previous segítségével új vezetéket jelölhetünk ki.

Ha a választott vezetéket kijelöltük a This-szel, az ernyő felső részén lévő FIXUP menü a következő módon változik:

Drop
End
Find
Zoom
escape

Most elkezdhetjük a vezeték mozgatását a kurzorvezérlő billentyűkkel. Ha a kívánt alakot elértük, a vezeték véglegesítése az End-del történhet (ekkor a képernyőn a régi vezeték eltűnik). Ha az új vezetékelhelyezésben nemcsak egy,

1. ábra



hanem több töréspontot kívánunk, ne az End-et, hanem a Drop-ot alkalmazzuk, csak az utolsó töréspontnál rögzítsük a helyzetet az End segítségével. Ha az összes vezetéket rendbetettük, a 3. ábrán látható kapcsolási rajzhoz jutunk.

A BLOCK menü következő eleme, amelyet bemutatunk, a SAVE. Ennek az alparancsnak a segítségével a már ismertetett módon kijelölt blokkot az OrCAD blokkmemóriájába tárolhatjuk (mint ahogyan egyszerre csak egyetlen blokkot jelölhetünk ki, a blokkmemória is csak egyetlen blokkot tárol).

A blokkmemóriából a tárolt blokkot a GET alparanccsal tölthetjük vissza a képernyőre. A visszahozott blokk a kurzorvezérlő billentyűkkel mozgatható a munkalapon, majd a kívánt helyre Place-szel helyezhető el. Eddig a dolog a MOVE alparancs hatásához hasonló, de lényeges különbség, hogy a blokkot további mozgatással újabb helyekre helyezhetjük le. Ez a funkció azonos a GET főparancs kapcsán már megismertekkel (a könyvtárból GET-tel lehívott alkatrészt is több helyre tehettük le anélkül, hogy a GET főparancsból ki kellett volna lépni).

Az EXPORT alparanccsal egy megjelölt blokkot másolhatunk fájlba. A blokk kijelölésének befejezésekor (End) a program a fájl nevét kérdezi. Ennek megadása során a szokásos DOS-formátumban meghajtó és útvonal (Path) is kijelölhető: például a:\orcad\f.sch.

Az IMPORT alparancs egy lemezes fájlnak a munkalapra töltését végzi. Kiadása után az OrCAD megkérdezi a fájlnevet; a meghajtót és útvonalat itt is megadhatjuk. Ha a megadott néven nincs fájl, hibajelzés keletkezik (a fájl megnyitása lehetetlen). Ha a fájl létezik, a képernyő felső részén egy menü jelenik meg:

Place
Find
Jump
Zoom
escape

Ekkor a kurzort arra a helyre kell mozgatnunk, ahová a betöltendő blokk bal felső sarkát kívánjuk pozicionálni. Ha most a menüből a Place-t választjuk, a lemezen tárolt blokk a megjelölt helyre kerül be. Ha a blokk a munkalapon nem fér el, az OrCAD megkérdezi, törölje-e a lapra már rá nem férő részt. Ha a kérdésre Yes választ adunk, a betöltés megtörténik, ha No-val válaszolunk, az IMPORT-ot az OrCAD figyelmen kívül hagyja, és visszatér a főparancsmenü „fölé".

Az EXPORT és IMPORT alparancsok alkalmazása különösen akkor hasznos, ha nagyobb részeket kívánunk egyik munkalapról a másikra átvinni.

Ha az OrCAD megismerésében, illetve a programmal segített rajzolás műveleteinek elsajátításában tovább kívánunk lépni, valószínűleg a DELETE főparancsra van a legnagyobb szükség. Ennek segítségével a rajz egyes részei törölhetők. A DELETE hívásakor a következő rövid menühöz jutunk:

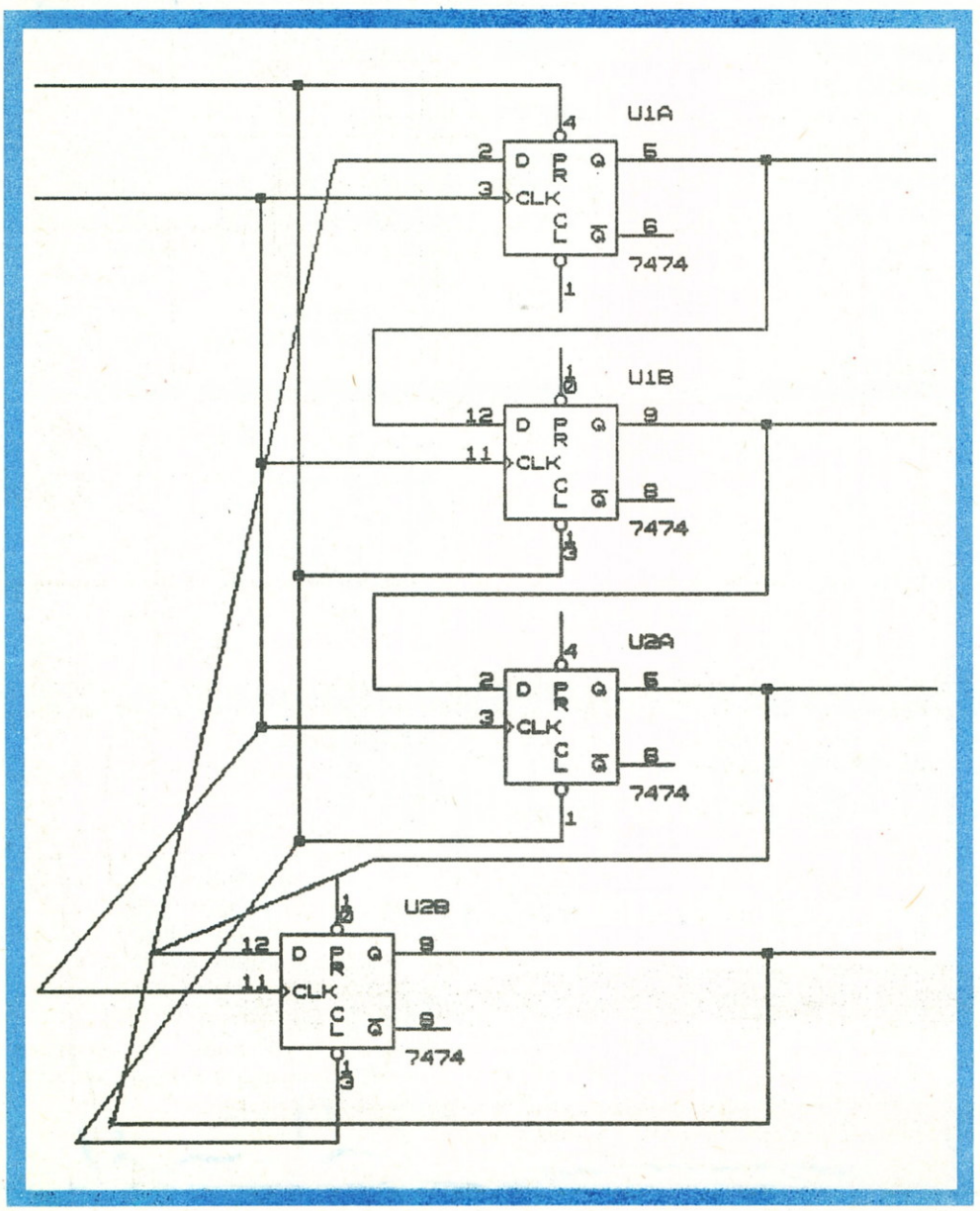
OBJECT
BLOCK
UNDO

Az OBJECT választása után egy újabb menüt kapunk a következő elemekkel:

Delete
Jump
Find
Zoom
escape

Az OBJECT-tel egy áramköri elemet (alkatrészt, egységet) tudunk a rajzról törölni. Vigyük a kurzort az alkatrész „belsejébe", és válasszuk a menüből a Delete-et! Az elem a rajzról törlődik, de a vezetékek vonalai a kapcsolódási pontokig megmaradnak. Ha a kurzort rossz

2. ábra



helyre állítottuk, hibajelzés keletkezik (itt nincs mit törölni). A szerkesztés ilyenkor az Enter lenyomásával folytatható.

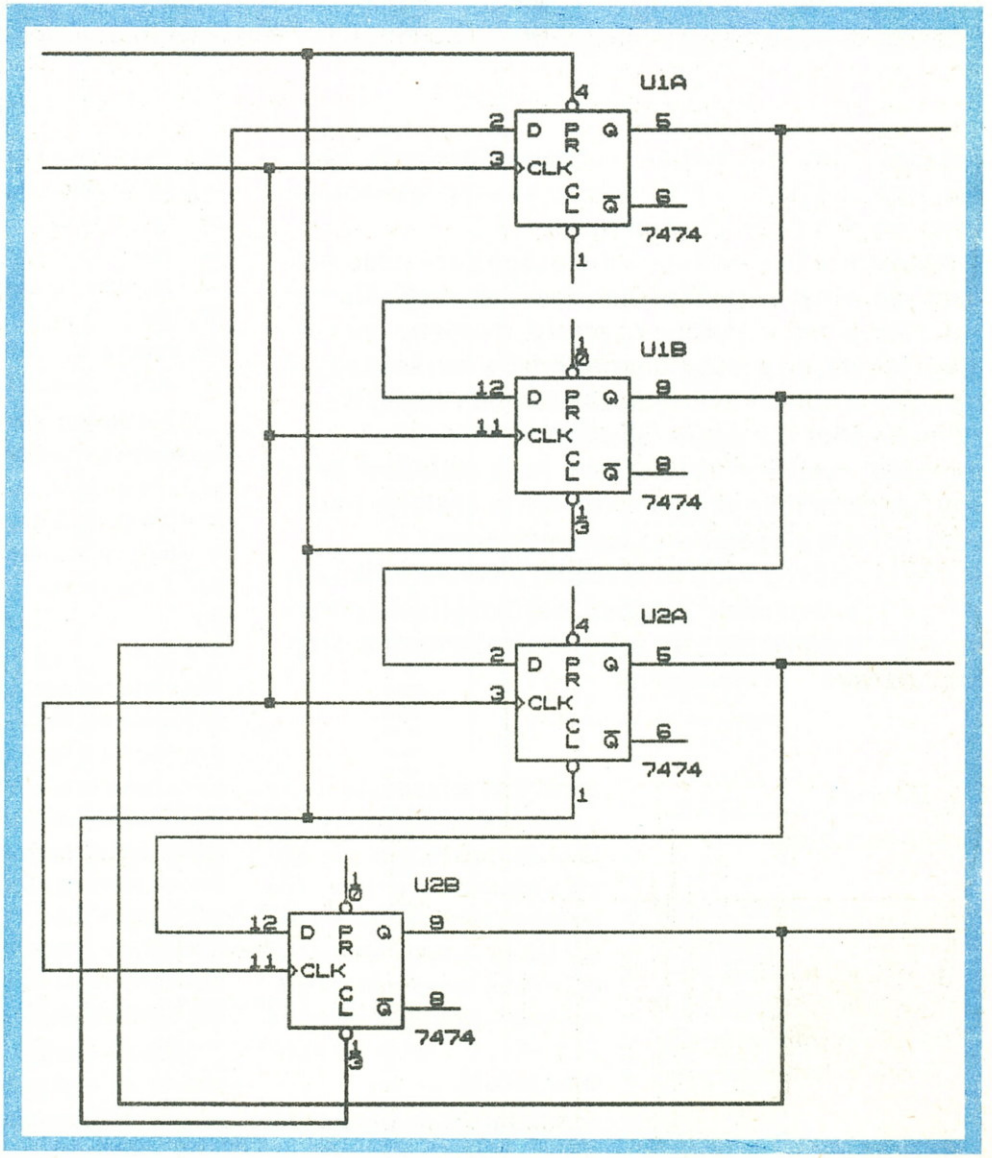
A BLOCK kiválasztása után szintén egy újabb menühöz jutunk:

Begin
Find
Jump
Zoom
escape

A már ismertetett módon kijelölhetjük a törölni kívánt blokkot. Az End hatására a blokk törlődik. Ha vezetékek is vannak benne, az OrCAD természetesen ezeket is törli. Ha egy kapcsolási elem egy része is szerepel a blokkban, az egész elem törlődik. A blokktörlés alkalmazható a rajzunkon esetleg már rajta lévő feliratok törlésére is.

Az UNDO a legutolsó törlés előtti állapotot állítja vissza akkor is, ha a főparancsmenüből már más műveletet is elvégeztünk, például GET-tel alkatrészt helyeztünk le, vagy a PLACE-szel vezetékeket rajzoltunk. A megfogalmazásból az is kiderül, hogy ha a DELETE főparancs elhagyása nélkül többször töröltünk, például az OBJECT funkcióban több elemet egymás után, az UNDOval az összes törlés hatása megszüntethető, azaz minden elem visszahozható a munkalapra.

Mayer Tamás



3. ábra

Hozzászólás

Lapunk 1988/10. számában cikket jelentettünk meg Könyvtárak és fájlok védelme címmel. Ehhez fűzött értékes észrevételeket Katona Zoltán miskolci olvasónk. Ezeket, amelyek a cikkben ismertetett és valóban egyszerű, korántsem tökéletes védelem helyett egy jobb megoldást mutatnak be, az alábbiakban közöljük.

A cikkben közölt megoldás megkerülhető, és ehhez még a PC TOOLS-ra sincs szükség, ha az azonosító végén elhelyezett és az általa ismert kódot az Alt+számbillentyűk segítségével adja meg a gép kezelője. Bár a könyvtárak nem, de a fájlok ** specifikációval továbbra is törölhetők maradnak.

Jobb megoldás, ha a cikkben említett FFH kódot nem az azonosító végére, hanem az attribútumbájtba tesszük. Ez a lemez-tartalomjegyzék 11. bájtja (0..7. bájt: könyvtár- vagy fájlazonosító, 8..10. bájt: kiterjesztés, 11. bájt: attribútum).

Az attribútumbájt szerkezete a következő:

Bit	Jelentés
0	csak olvasható
1	rejtett
2	rendszerfájl
3	kötetazonosító
4	tartalomjegyzék
5	archív
6	
7	

Ha a fentieket végiggondoljuk, egyszerűen belátható, hogy az FFH attribútummal a rendszer valóban nem tud mit kezdeni (a fájl vagy könyvtár egyszerre rendszerfájl, kötetazonosító, könyvtár, tartalomjegyzék stb.)

Kevésbé zavaró, ha FFH helyett értelmesebb értékeket adunk meg. Például: könyvtár+rejtett=12H, vagy archív+rendszer+rejtett+csak olvasható=27H. Ennek a megoldásnak fő előnye, hogy a CHKDSK vagy a különféle segédprogramok (például a PC TOOLS R5.1-hez tartozó COMPRESS) is elfogadják az „értelmesebb" megjelölést, míg az FFH attribútummal „nem tudnak megbirkózni", bár utóbbi tény a védelem hatásosságát nem korlátozza.

Az ismertetett módon hozzáférhetetlenné tett fájlokat és könyvtárakat egykönnyen sem törölni, sem felülírni nem lehet.

FELADATOK — MEGOLDÁSOK

Sorozatunkat elsősorban középiskolásoknak szánjuk, de reméljük, hogy minden olvasónknak tanulási lehetőséget és szórakozást nyújt.
A feladatok a Nemes Tihamér országos számítástechnikai verseny színvonalának felelnek meg. Minden esetben olyat választunk, amely röviden, gyorsan megoldható, de a megoldáshoz ötletre van szükség. A megoldást mindig a következő számban közöljük.
Mivel a változatosságra törekszünk, különböző programozási nyelveket használunk. Az is előfordul, hogy egy feladatra több programnyelven is közlünk megoldást, ezzel is elősegítve az ismeretszerzést.
A szerkesztőség várja az olvasók, a versenyzők leveleit. A legötletesebb program beküldőjét könyvutalvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

20. feladat: Disassembler

Írjon disassembláló programot egy tetszőleges processzorra, tetszőleges nyelven! Törekedjen arra, hogy a program más assembly nyelv disassemblálására könnyen átírható legyen.

Megoldás

Azt elég gyorsan el tudjuk dönteni, hogy feltételes utasítások sorával sem gyors, sem flexibilis programot nem lehet írni. Ugyanakkor egy egyszerű táblázattal sem dolgozhatunk, mert sok assembler utasításnak van paramétere, amely külön feldolgozást igényel. A megoldást egy olyan táblázat jelenti, amely lehetővé teszi az utasítások csoportokra osztását, és az egyes csoportok speciális kezelését.

A disassembláló programot egy egyszerű processzor, a 8048-as alapján mutatom be. Ennek az utasításkészletét csak néhány csoportba kell osztani. Az immediate adatot tartalmazó utasításokon kívül csak az ugróutasítások igényelnek speciális kezelést. Ezeknél az adatot, illetve a címet az utasítás második bájtjából kell kiszámolni. Ezenkívül a regisztercímzést használó utasítások kerültek egy csoportba. Ha az azonos sztringeket összevonni tudó fordítóprogramot használunk (ilyen például a Turbo-C 2.0), akkor rövidebb kódot kapunk.

Az IBM PC-re Turbo-C 2.0 nyelven megírt program fő részét a codes táblázat képezi. Ennek definíciója:

```
const struct {
char *name;
byte type;
} codes [256];
```

A táblázat n-edik eleme (codes[n]) tartalmazza az n kódhoz tartozó utasítás nevét (codes[n].name) és csoportba sorolását (codes[n].type). A csoporttípus alapján lehet a szükséges paraméterekkel kiegészíteni az utasításnevet. Ez határozza meg azt is, hogy az utasítás hány bájtos.

Mielőtt a disassembláló szubrutint átnéznénk, tekintsük át a főprogram működését!

A bejelentkezés és a használati utasítás kiírása után a program egy 64 kbájtos területet foglal le a far heapből, és itt tárolja a beolvasott kódot. Mivel ez a szegmensregiszteres címzés miatt nem is olyan egyszerű, egy külön szubrutin (binit) végzi el. A szubrutin a lefoglalt buff tömböt 0xFF bájtokkal tölti fel.

A betöltendő fájl megadására lehetőség van parancssorból vagy az L utasítással. A kettő igen hasonlóan történik. Ofszetcím mindkét esetben megadható, így a fájlt tetszőleges címre tölthetjük.

Ezenkívül még két utasítása van a programnak: a D dump és az U disassemble. A parancssorok feldolgozása egy végtelen ciklusban történik. A két parancs feldolgozása hasonló. Ha paraméter nélkül használjuk a parancsot, akkor az előző end címtől ugyanannyi kód feldolgozása történik. A dump, illetve a disassemble funkciót a dump és a dis eljárások végzik.

Nézzük először a dump eljárást, ez az egyszerűbb! A cím után 16 bájt hexadecimális kiírása következik, majd a start pointer 16-ot visszalép, és ugyanezt a 16 bájtot ASCII-ben is kiírja, a nem nyomtatható karaktereket "."-tal helyettesítve. Az eljárás billentyűlenyomással félbeszakítható, vagy end cím elérésekor befejeződik.

A dis eljárás sokban hasonlít a dump szubrutinhoz, bár itt a kiíratás nem közvetlenül, hanem line sztringen keresztül történik. Erre azért van szükség, mert kétbájtos utasítások esetén a második bájt hexadecimális kódját be kell szúrni az utasítás neve elé.

Ha az utasítás NOP típusú, azaz nem igényel különleges elbánást, akkor a rutin első utasítása elvégzi a disassemblálást, a kapott sort már csak ki kell írni. A cím, a kódbájt és az utasításnév kerül line-ba. Más típusú utasításokat még ki kell egészíteni. Ez történik a switch utasítás különböző case ágaiban. Ha az utasítás kétbájtos, a start pointer is itt növelhető.

A rutin végén az előkészített sor kiírása, majd a pointer növelése valósul meg. Az eljárás a dumphoz hasonlóan billentyűlenyomással félbeszakítható, vagy end cím elérésekor befejeződik.

Más nyolcbites processzor nyelvének disassemblálására a program könnyen átírható. Egyszerű utasításkészletű processzor — például 8031 — esetén csak a táblázatot kell módosítani és egy-két új típust felvenni. Bonyolultabb processzor — mint a Z80 — újabb táblázatokat és prefixek alapján a megfelelő táblázat gondos kiválasztását igényli. 64 kbájtnál nagyobb címtartományt címezni tudó processzor nyelvének disassemblálását a program nem támogatja, ahhoz a puffer kezelését meg kellene változtatni.

21. feladat: Kifejezés-kiértékelő

Írjon programot, amely egy zárójeles kifejezés értékét kiszámítja. A kifejezés a következő műveleti jeleket tartalmazza:

*,/ elsődleges,

+,- másodlagos.

Az elsődleges műveletek hajtandók végre először. ZX-Spectrum BASIC-ben a VAL függvény alkalmazása nem megoldás!

Hogyan lehetne a programot további műveleti jelekkel bővíteni?

Pintér Gábor

```
/* dis48m.c */

/* 8048 disassembler              V1.2m  */
/*  Írta  Pinyó                 11-01-89 */

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <ctype.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>

/* Ezzel tölti ki az üres helyet. */
#define FILL 0xFF

/* Hasznos tipusok: */
typedef unsigned char   byte;
typedef unsigned int    word;

/* A puffer: */
/* 64kbyte-os tömb a far heap-ben */
byte far *buff;

/* Sorhossz. */
#define LLEN 133

/*************************************/

/* A puffer létrehozása és
   inicializálása. */
void binit(){
  register word i;

/* 64kbyte + 16 byte lefoglalása. */
  if (!(buff = farmalloc(0x10010L))){
    printf("Not enough memory.\n");
    exit(0);
  }
/* Következő paragrafushatárra
   illesztés. */
  buff = (byte far *)
  ((((unsigned long int)(buff)) +
  ((unsigned int)(buff)>>4)+0x10000L)
  & 0xFFFF0000L);
/* A puffer törlése. */
  i = 0;
  do {
    buff[i] = FILL;
  } while (++i);
} /* binit */

/*************************************/

/* Start-tól end-ig disassemblál. */
void dis(word start, word end){

/* Vége jelző. */
  char v = '\0';
/* A disassemlált sor. */
  char line[LLEN];
  int pos;
/* Jump/call cím.*/
  word addr;

/* Disassembler táblázat. */
#define NOP  '\x0'
#define IMM  '\x1'
#define CALL '\x2'
#define PAGE '\x3'
#define RR   '\x4'

const struct {
  char *name;
  byte type;
} codes[256] = {
/*  00  */ {"NOP"              ,NOP  },
/*  01  */ {"DB\t001H"         ,NOP  },
/*  02  */ {"OUTL\tBUS,A"      ,NOP  },
/*  03  */ {"ADD\tA"           ,IMM  },
/*  04  */ {"JMP"              ,CALL },
/*  05  */ {"EN\tI"            ,NOP  },
/*  06  */ {"DB\t006H"         ,NOP  },
/*  07  */ {"DEC\tA"           ,NOP  },
/*  08  */ {"INS\tA,BUS"       ,NOP  },
/*  09  */ {"IN\tA,P1"         ,NOP  },
/*  0A  */ {"IN\tA,P2"         ,NOP  },
/*  0B  */ {"DB\t00BH"         ,NOP  },
/*  0C  */ {"MOVD\tA,P4"       ,NOP  },
/*  0D  */ {"MOVD\tA,P5"       ,NOP  },
/*  0E  */ {"MOVD\tA,P6"       ,NOP  },
/*  0F  */ {"MOVD\tA,P7"       ,NOP  },
/*  10  */ {"INC\t@R0"         ,NOP  },
/*  11  */ {"INC\t@R1"         ,NOP  },
/*  12  */ {"JB0\t"            ,PAGE },
/*  13  */ {"ADDC\tA"          ,IMM  },
/*  14  */ {"CALL"             ,CALL },
/*  15  */ {"DIS\tI"           ,NOP  },
/*  16  */ {"JTF\t"            ,PAGE },
/*  17  */ {"INC\tA"           ,NOP  },
/*  18  */ {"INC\t"            ,RR   },
/*  19  */ {"INC\t"            ,RR   },
/*  1A  */ {"INC\t"            ,RR   },
/*  1B  */ {"INC\t"            ,RR   },
/*  1C  */ {"INC\t"            ,RR   },
/*  1D  */ {"INC\t"            ,RR   },
/*  1E  */ {"INC\t"            ,RR   },
/*  1F  */ {"INC\t"            ,RR   },

/*  20  */ {"XCH\tA,@R0"       ,NOP  },
/*  21  */ {"XCH\tA,@R1"       ,NOP  },
/*  22  */ {"DB\t022H"         ,NOP  },
/*  23  */ {"MOV\tA"           ,IMM  },
/*  24  */ {"JMP"              ,CALL },
/*  25  */ {"EN\tTCNTI"        ,NOP  },
/*  26  */ {"JNT0\t"           ,PAGE },
/*  27  */ {"CLR\tA"           ,NOP  },
/*  28  */ {"XCH\tA,"          ,RR   },
/*  29  */ {"XCH\tA,"          ,RR   },
/*  2A  */ {"XCH\tA,"          ,RR   },
/*  2B  */ {"XCH\tA,"          ,RR   },
/*  2C  */ {"XCH\tA,"          ,RR   },
/*  2D  */ {"XCH\tA,"          ,RR   },
/*  2E  */ {"XCH\tA,"          ,RR   },
/*  2F  */ {"XCH\tA,"          ,RR   },

/*  30  */ {"XCHD\tA,@R0"      ,NOP  },
/*  31  */ {"XCHD\tA,@R1"      ,NOP  },
/*  32  */ {"JB1\t"            ,PAGE },
/*  33  */ {"DB\t033H"         ,NOP  },
/*  34  */ {"CALL"             ,CALL },
/*  35  */ {"DIS\tTCNTI"       ,NOP  },
/*  36  */ {"JT0\t"            ,PAGE },
/*  37  */ {"CPL\tA"           ,NOP  },
/*  38  */ {"DB\t038H"         ,NOP  },
/*  39  */ {"OUTL\tP1,A"       ,NOP  },
/*  3A  */ {"OUTL\tP2,A"       ,NOP  },
/*  3B  */ {"DB\t03BH"         ,NOP  },
/*  3C  */ {"MOVD\tP4,A"       ,NOP  },
/*  3D  */ {"MOVD\tP5,A"       ,NOP  },
/*  3E  */ {"MOVD\tP6,A"       ,NOP  },
/*  3F  */ {"MOVD\tP7,A"       ,NOP  },

/*  40  */ {"ORL\tA,@R0"       ,NOP  },
/*  41  */ {"ORL\tA,@R1"       ,NOP  },
/*  42  */ {"MOV\tA,T"         ,NOP  },
/*  43  */ {"ORL\tA"           ,IMM  },
/*  44  */ {"JMP"              ,CALL },
/*  45  */ {"STRT\tCNT"        ,NOP  },
/*  46  */ {"JNT1\t"           ,PAGE },
/*  47  */ {"SWAP\tA"          ,NOP  },
/*  48  */ {"ORL\tA,"          ,RR   },
/*  49  */ {"ORL\tA,"          ,RR   },
/*  4A  */ {"ORL\tA,"          ,RR   },
/*  4B  */ {"ORL\tA,"          ,RR   },
/*  4C  */ {"ORL\tA,"          ,RR   },
/*  4D  */ {"ORL\tA,"          ,RR   },
/*  4E  */ {"ORL\tA,"          ,RR   },
/*  4F  */ {"ORL\tA,"          ,RR   },

/*  50  */ {"ANL\tA,@R0"       ,NOP  },
/*  51  */ {"ANL\tA,@R1"       ,NOP  },
/*  52  */ {"JB2\t"            ,PAGE },
/*  53  */ {"ANL\tA"           ,IMM  },
/*  54  */ {"CALL"             ,CALL },
/*  55  */ {"STRT\tT"          ,NOP  },
/*  56  */ {"JT1\t"            ,PAGE },
/*  57  */ {"DB\t057H"         ,NOP  },
/*  58  */ {"ANL\tA,"          ,RR   },
/*  59  */ {"ANL\tA,"          ,RR   },
/*  5A  */ {"ANL\tA,"          ,RR   },
/*  5B  */ {"ANL\tA,"          ,RR   },
/*  5C  */ {"ANL\tA,"          ,RR   },
/*  5D  */ {"ANL\tA,"          ,RR   },
/*  5E  */ {"ANL\tA,"          ,RR   },
/*  5F  */ {"ANL\tA,"          ,RR   },

/*  60  */ {"ADD\tA,@R0"       ,NOP  },
/*  61  */ {"ADD\tA,@R1"       ,NOP  },
/*  62  */ {"MOV\tT,A"         ,NOP  },
/*  63  */ {"DB\t063H"         ,NOP  },
/*  64  */ {"JMP"              ,CALL },
/*  65  */ {"STOP\tTCNT"       ,NOP  },
/*  66  */ {"DB\t066H"         ,NOP  },
/*  67  */ {"RRC\tA"           ,NOP  },
/*  68  */ {"ADD\tA,"          ,RR   },
/*  69  */ {"ADD\tA,"          ,RR   },
/*  6A  */ {"ADD\tA,"          ,RR   },
/*  6B  */ {"ADD\tA,"          ,RR   },
/*  6C  */ {"ADD\tA,"          ,RR   },
/*  6D  */ {"ADD\tA,"          ,RR   },
/*  6E  */ {"ADD\tA,"          ,RR   },
/*  6F  */ {"ADD\tA,"          ,RR   },

/*  70  */ {"ADDC\tA,@R0"      ,NOP  },
/*  71  */ {"ADDC\tA,@R1"      ,NOP  },
/*  72  */ {"JB3\t"            ,PAGE },
/*  73  */ {"DB\t073H"         ,NOP  },
/*  74  */ {"CALL"             ,CALL },
/*  75  */ {"ENT0\tCLK"        ,NOP  },
/*  76  */ {"JF1\t"            ,PAGE },
/*  77  */ {"RR\tA"            ,NOP  },
/*  78  */ {"ADDC\tA,"         ,RR   },
/*  79  */ {"ADDC\tA,"         ,RR   },
/*  7A  */ {"ADDC\tA,"         ,RR   },
/*  7B  */ {"ADDC\tA,"         ,RR   },
/*  7C  */ {"ADDC\tA,"         ,RR   },
/*  7D  */ {"ADDC\tA,"         ,RR   },
/*  7E  */ {"ADDC\tA,"         ,RR   },
/*  7F  */ {"ADDC\tA,"         ,RR   },

/*  80  */ {"MOVX\tA,@R0"      ,NOP  },
/*  81  */ {"MOVX\tA,@R1"      ,NOP  },
/*  82  */ {"DB\t082H"         ,NOP  },
/*  83  */ {"RET"              ,NOP  },
/*  84  */ {"JMP"              ,CALL },
/*  85  */ {"CLR\tF0"          ,NOP  },
/*  86  */ {"JNI\t"            ,PAGE },
/*  87  */ {"DB\t087H"         ,NOP  },
/*  88  */ {"ORL\tBUS"         ,IMM  },
/*  89  */ {"ORL\tP1"          ,IMM  },
/*  8A  */ {"ORL\tP2"          ,IMM  },
/*  8B  */ {"DB\t08BH"         ,NOP  },
/*  8C  */ {"ORLD\tP4,A"       ,NOP  },
/*  8D  */ {"ORLD\tP5,A"       ,NOP  },
/*  8E  */ {"ORLD\tP6,A"       ,NOP  },
/*  8F  */ {"ORLD\tP7,A"       ,NOP  },

/*  90  */ {"MOVX\t@R0,A"      ,NOP  },
/*  91  */ {"MOVX\t@R1,A"      ,NOP  },
/*  92  */ {"JB4\t"            ,PAGE },
/*  93  */ {"RETR"             ,NOP  },
/*  94  */ {"CALL"             ,CALL },
/*  95  */ {"CPL\tF0"          ,NOP  },
/*  96  */ {"JNZ\t"            ,PAGE },
/*  97  */ {"CLR\tC"           ,NOP  },
/*  98  */ {"ANL\tBUS"         ,IMM  },
/*  99  */ {"ANL\tP1"          ,IMM  },
/*  9A  */ {"ANL\tP2"          ,IMM  },
/*  9B  */ {"DB\t09BH"         ,NOP  },
/*  9C  */ {"ANLD\tP4,A"       ,NOP  },
/*  9D  */ {"ANLD\tP5,A"       ,NOP  },
/*  9E  */ {"ANLD\tP6,A"       ,NOP  },
/*  9F  */ {"ANLD\tP7,A"       ,NOP  },

/*  A0  */ {"MOV\t@R0,A"       ,NOP  },
/*  A1  */ {"MOV\t@R1,A"       ,NOP  },
/*  A2  */ {"DB\t0A2H"         ,NOP  },
/*  A3  */ {"MOVP\tA,@A"       ,NOP  },
/*  A4  */ {"JMP"              ,CALL },
/*  A5  */ {"CLR\tF1"          ,NOP  },
/*  A6  */ {"DB\t0A6H"         ,NOP  },
/*  A7  */ {"CPL\tC"           ,NOP  },
/*  A8  */ {"MOV\tR0,A"        ,NOP  },
/*  A9  */ {"MOV\tR1,A"        ,NOP  },
/*  AA  */ {"MOV\tR2,A"        ,NOP  },
/*  AB  */ {"MOV\tR3,A"        ,NOP  },
/*  AC  */ {"MOV\tR4,A"        ,NOP  },
/*  AD  */ {"MOV\tR5,A"        ,NOP  },
/*  AE  */ {"MOV\tR6,A"        ,NOP  },
/*  AF  */ {"MOV\tR7,A"        ,NOP  },

/*  B0  */ {"MOV\t@R0"         ,IMM  },
/*  B1  */ {"MOV\t@R1"         ,IMM  },
/*  B2  */ {"JB5\t"            ,PAGE },
/*  B3  */ {"JMPP\t@A"         ,NOP  },
/*  B4  */ {"CALL"             ,CALL },
/*  B5  */ {"CPL\tF1"          ,NOP  },
/*  B6  */ {"JF0\t"            ,PAGE },
/*  B7  */ {"DB\t0B7H"         ,NOP  },
/*  B8  */ {"MOV\tR0"          ,IMM  },
/*  B9  */ {"MOV\tR1"          ,IMM  },
/*  BA  */ {"MOV\tR2"          ,IMM  },
/*  BB  */ {"MOV\tR3"          ,IMM  },
/*  BC  */ {"MOV\tR4"          ,IMM  },
/*  BD  */ {"MOV\tR5"          ,IMM  },
/*  BE  */ {"MOV\tR6"          ,IMM  },
/*  BF  */ {"MOV\tR7"          ,IMM  },

/*  C0  */ {"DB\t0C0H"         ,NOP  },
/*  C1  */ {"DB\t0C1H"         ,NOP  },
/*  C2  */ {"DB\t0C2H"         ,NOP  },
/*  C3  */ {"DB\t0C3H"         ,NOP  },
/*  C4  */ {"JMP"              ,CALL },
/*  C5  */ {"SEL\tRB0"         ,NOP  },
/*  C6  */ {"JZ\t"             ,PAGE },
/*  C7  */ {"MOV\tA,PSW"       ,NOP  },
/*  C8  */ {"DEC\t"            ,RR   },
/*  C9  */ {"DEC\t"            ,RR   },
/*  CA  */ {"DEC\t"            ,RR   },
/*  CB  */ {"DEC\t"            ,RR   },
/*  CC  */ {"DEC\t"            ,RR   },
/*  CD  */ {"DEC\t"            ,RR   },
/*  CE  */ {"DEC\t"            ,RR   },
/*  CF  */ {"DEC\t"            ,RR   },

/*  D0  */ {"XRL\tA,@R0"       ,NOP  },
/*  D1  */ {"XRL\tA,@R1"       ,NOP  },
/*  D2  */ {"JB6\t"            ,PAGE },
```

```
/*  D3  */ {"XRL\tA"          ,IMM   },
/*  D4  */ {"CALL"            ,CALL  },
/*  D5  */ {"SEL\tRB1"        ,NOP   },
/*  D6  */ {"DB\t0D6H"        ,NOP   },
/*  D7  */ {"MOV\tPSW,A"      ,NOP   },
/*  D8  */ {"XRL\tA,"         ,RR    },
/*  D9  */ {"XRL\tA,"         ,RR    },
/*  DA  */ {"XRL\tA,"         ,RR    },
/*  DB  */ {"XRL\tA,"         ,RR    },
/*  DC  */ {"XRL\tA,"         ,RR    },
/*  DD  */ {"XRL\tA,"         ,RR    },
/*  DE  */ {"XRL\tA,"         ,RR    },
/*  DF  */ {"XRL\tA,"         ,RR    },

/*  E0  */ {"DB\t0E0H"        ,NOP   },
/*  E1  */ {"DB\t0E1H"        ,NOP   },
/*  E2  */ {"DB\t0E2H"        ,NOP   },
/*  E3  */ {"MOVP3\tA,@A"     ,NOP   },
/*  E4  */ {"JMP"             ,CALL  },
/*  E5  */ {"SEL\tMB0"        ,NOP   },
/*  E6  */ {"JNC\t"           ,PAGE  },
/*  E7  */ {"RL\tA"           ,NOP   },
/*  E8  */ {"DJNZ\tR0,"       ,PAGE  },
/*  E9  */ {"DJNZ\tR1,"       ,PAGE  },
/*  EA  */ {"DJNZ\tR2,"       ,PAGE  },
/*  EB  */ {"DJNZ\tR3,"       ,PAGE  },
/*  EC  */ {"DJNZ\tR4,"       ,PAGE  },
/*  ED  */ {"DJNZ\tR5,"       ,PAGE  },
/*  EE  */ {"DJNZ\tR6,"       ,PAGE  },
/*  EF  */ {"DJNZ\tR7,"       ,PAGE  },

/*  F0  */ {"MOV\tA,@R0"      ,NOP   },
/*  F1  */ {"MOV\tA,@R1"      ,NOP   },
/*  F2  */ {"JB7\t"           ,PAGE  },
/*  F3  */ {"DB\t0F3H"        ,NOP   },
/*  F4  */ {"CALL"            ,CALL  },
/*  F5  */ {"SEL\tMB1"        ,NOP   },
/*  F6  */ {"JC\t"            ,PAGE  },
/*  F7  */ {"RLC\tA"          ,NOP   },
/*  F8  */ {"MOV\tA,"         ,RR    },
/*  F9  */ {"MOV\tA,"         ,RR    },
/*  FA  */ {"MOV\tA,"         ,RR    },
/*  FB  */ {"MOV\tA,"         ,RR    },
/*  FC  */ {"MOV\tA,"         ,RR    },
/*  FD  */ {"MOV\tA,"         ,RR    },
/*  FE  */ {"MOV\tA,"         ,RR    },
/*  FF  */ {"MOV\tA,"         ,RR    },

};

  do {
/* A cím, az első byte és
   az utasítás név kiírása. */
    pos = sprintf(line,
          "%04X %02X        %s",
          start,buff[start],
          codes[buff[start]].name);
/* Tipustól függő folytatás. */
    switch (codes[buff[start]].type){
      case IMM:{
/* Immediate operandus kiírása. */
        if (start++ == end) v = '\1';
        sprintf(line+pos,",#0%02XH",
                buff[start]);
        sprintf(line+8,"%02X",
                buff[start]);
        *(line+10) = ' ';
        break;
      }
      case CALL:{
/* Jump/call cím számítása és kiírása */
        addr = ((buff[start] >> 5) << 8)
                + buff[start+1];
        sprintf(line+pos,"\t0%04XH",
                addr);
        if (start++ == end) v = '\1';
        sprintf(line+8,"%02X",
                buff[start]);
        *(line+10) = ' ';
        break;
      }
      case PAGE:{
/* Lapon belüli feltételes ugrás. */
        addr = (start & 0xFF00)
               + buff[start+1];
        sprintf(line+pos,"0%04XH",addr);
        if (start++ == end) v = '\1';
        sprintf(line+8,"%02X",
                buff[start]);
        *(line+10) = ' ';
        break;
      }
      case RR:{
/* Regiszter. */
        sprintf(line+pos,"R%1d",
                buff[start] & '\7');
        break;
      }
    }
/* Az előkészített sor
   tényleges kiírása. */
    printf("%s\n",line);
    if (start++ == end) v = '\1';
/* Billentyűlenyomással megszakítható */
    if (kbhit()){
      getch();
      v = '\1';
    }
  } while (!v);
} /* dis */

/*************************************/

/* Dump start-tól end-ig. */
void dump(word start, word end){
  register int i;
  char v = '\0';

  do {
/* A cím kiírása. */
    printf("%04X ",start);
/* 16 byte kiírása hexadecimálisan. */
    for (i = 0; i<16; i++){
      printf(" %02X",buff[start++]);
    }
    printf("  ");
/* Ugyanennek a 16 byte-nak a
   kiírása ASCII kódban. */
    start -= 16;
    for (i = 0; i<16; i++){
      if (end == start) v = '\1';
      printf("%c",
      (((buff[start] & '\x7F') >= ' ') ?
      (buff[start] & '\x7F') :
      ('.')));
      start++;
    }
    printf("\n");
/* Billentyűlenyomással megszakítható */
    if (kbhit()){
      getch();
      v = '\1';
    }
  } while (!v);
}

/*************************************/

main(){
/* Bejelentkezési üzenet. */
  char *log[] = {
"8048 disassembler     V1.2m"
  "       Pinyó     11-01-1989",
"",
"  Használata::",
"    dis48m [[path]név"
  " [offset]] [>output_file]",
"",
"  Érvényes parancsok:",
"    L name[ offset]  - Fájl betöltése",
"    D [start[ end]]  - Dump",
"    U [start[ end]]  - Disassemble",
"    ^C               - Kilépés",
"",
NULL
};
  char **l = log;
/* Input fájl. */
  char inname[LLEN];
  FILE *infile;
  int bb;
/* Fájl betöltési offset. */
  word offs = 0;
/* Parancssor. */
  char c;
  char line[LLEN];
  char *ll;
/* Start és end cím. */
  word start = 0;
  word end = 0x10;
  word est;

/* Bejelentkezés. */
  while (*l){
    fprintf(stderr,"\t%s\n",*l++);
  }

/* A puffer létrehozása
   és inicializálása. */
  binit();
/* Ha a parancssorban fájl definiált,
   annak megnyitása. */
  if (_argc > 1){
    strcpy(inname,_argv[1]);
    if (!(infile = fopen(inname,"rb"))){
      fprintf(stderr,
  "Input fájl megnyitás sikertelen.\n");
      exit(0);
    }
    sscanf(_argv[2],"%x",&offs);
    for (;;){
      if ((bb=getc(infile)) == EOF)
        break;
      buff[offs++] = bb;
    }
    fclose(infile);


  for (;;){
/* A parancssor beolvasása. */
    fprintf(stderr,">");
    gets(line);
    ll = line;
    while (isspace(c = *ll++));
/* Elágazás a parancstól függően. */
    switch (c & '\x5F'){
/* Nem parancs. */
      default:{
        fprintf(stderr,
         "Nem létező parancs.\n");
        break;
      }
/* CR-ek kihagyása.  */
      case '\r': case '\n': case '\0':
        break;
/* Fájl betöltése. */
      case 'L':{
        offs = 0;
        sscanf(ll,"%s%x",inname,&offs);
        if (!(infile =
                fopen(inname,"rb"))){
          fprintf(stderr,
  "Input fájl megnyitás sikertelen.\n");
          break;
        }
        for (;;){
          if ((bb=getc(infile)) == EOF)
            break;
          buff[offs++] = bb;
        }
        fclose(infile);
        break;
      }
/* Dissassemble. */
      case 'U':{
        sscanf(ll,"%x%x",&start,&end);
        dis(start,end);
        est = end - start;
        start = end;
        end += est;
        break;
      }
/* Dump. */
      case 'D':{
        sscanf(ll,"%x%x",&start,&end);
        dump(start,end);
        est = end - start;
        start = end;
        end += est;
        break;
      }
    }
  }
```

Hangos Commodore Plus/4

HANGJEGY	H.REG.ERETEK	AKT.FREKVENCIA	$FF0E-$FF0F	$FF13 $FF12	SK BK GK
C-1	185	132,2	B9	7C	C0 C4 C8
C#1	235	140,5	EB	7C	C0 C4 C8
D-1	278	148,6	16	7D	C1 C5 C9
D#1	318	157,1	3E	7D	C1 C5 C9
E-1	353	165,2	61	7D	C1 C5 C9
F-1	395	176,3	8B	7D	C1 C5 C9
F#1	432	187,3	B0	7D	C1 C5 C9
G-1	465	198,3	D1	7D	C1 C5 C9
G#1	494	209,2	EE	7D	C1 C5 C9
A-1	521	220,4	09	7E	C2 C6 CA
A#1	550	233,9	26	7E	C2 C6 CA
H-1	576	247,5	40	7E	C2 C6 CA
C-2	605	264,6	5D	7E	C2 C6 CA
C#2	629	280,7	75	7E	C2 C6 CA
D-2	651	297,3	8B	7E	C2 C6 CA
D#2	671	314,1	9F	7E	C2 C6 CA
E-2	688	330	B0	7E	C2 C6 CA
F-2	709	352	C5	7E	C2 C6 CA
F#2	727	373,3	D7	7E	C2 C6 CA
G-2	744	396	E8	7E	C2 C6 CA
G#2	759	418,4	F7	7E	C2 C6 CA
A-2	772	440	04	7F	C3 C7 CB
A#2	787	467,8	13	7F	C3 C7 CB
H-2	800	495	20	7F	C3 C7 CB
C-3	814	528	2E	7F	C3 C7 CB
C#3	826	560	3A	7F	C3 C7 CB
D-3	837	592,9	45	7F	C3 C7 CB
D#3	847	626,4	4F	7F	C3 C7 CB
E-3	855	656,1	57	7F	C3 C7 CB
F-3	866	701,8	62	7F	C3 C7 CB
F#3	875	744,2	6B	7F	C3 C7 CB
G-3	884	792	74	7F	C3 C7 CB
G#3	891	833,7	7B	7F	C3 C7 CB
A-3	898	880	82	7F	C3 C7 CB
A#3	905	931,8	89	7F	C3 C7 CB
H-3	912	990	90	7F	C3 C7 CB
C-4	918	1046,1	96	7F	C3 C7 CB

SK:SAJAT KARAKTERKESZLETNEL
BK:A GEP BELSO KARAKTERKESZLETENEK HASZNALATAKOR (BEJELENTKEZESKOR)
GK:GRAFIKUS KEP HASZNALATAKOR

(C) BY BAM TEAM '1989 BUDAPEST

Senki sem állítja, hogy a C Plus/4-es gép zenei géniusz lenne. Mégis, akit sorsa egy Plus/4-eshez vezetett, természetesen szeretné kihasználni a gép zenei adottságait, bármilyen szerények is azok. Sajnos a géppel foglalkozó szakkönyvekben éppen ez a fejezet a legrövidebb, és a könyvek sok esetben a felhasználó szempontjából érdektelen információkat tartalmaznak. Például a Felhasználói Kézikönyv és a Programozói Útmutató az NTSC rendszerben érvényes hangregiszterértékeket közli!

Tapasztalataim szerint az a képlet is hibás, amelyet arra az esetre mellékeltek, ha a felhasználó országában — azaz nálunk is — PAL rendszerűek a tv-készülékek. Számításaim szerint a képlet a következő:

hangregiszterérték = 1024-(110880/frekvencia)

Súlyosabb hiba, hogy egy könyv sem közli a félhangok hangregiszterértékeit.

Ezeken a problémákon próbál segíteni a következő táblázat, amely természetesen PAL rendszerben érvényes.

Szabolcsi Szabolcs

A funkcióbillentyűk titka,

avagy hogyan használjunk nyolcnál több funkcióbillentyűt a C64-en?

```
.,C000  78       SEI
.,C001  A9 C0    LDA #$C0
.,C003  A0 15    LDY #$15
.,C005  8C 14 03 STY $0314
.,C008  8D 15 03 STA $0315
.,C00B  A9 00    LDA #$00
.,C00D  85 FE    STA $FE
.,C00F  8D 14 C0 STA $C014
.,C012  58       CLI
.,C013  60       RTS
.,C014  00       BRK
.,C015  AD 8D 02 LDA $028D
.,C018  A6 C5    LDX $C5
.,C01A  E4 FE    CPX $FE
.,C01C  D0 05    BNE $C023
.,C01E  CD 14 C0 CMP $C014
.,C021  F0 43    BEQ $C066
.,C023  E0 03    CPX #$03
.,C025  90 4D    BCC $C074
```

```
.,C027  E0 07    CPX #$07
.,C029  B0 49    BCS $C074
.,C02B  86 FE    STX $FE
.,C02D  29 07    AND #$07
.,C02F  A0 00    LDY #$00
.,C031  D9 88 C0 CMP $C088,Y
.,C034  D0 03    BNE $C039
.,C036  20 69 C0 JSR $C069
.,C039  C8       INY
.,C03A  C0 07    CPY #$07
.,C03C  D0 F3    BNE $C031
.,C03E  A5 C5    LDA $C5
.,C040  38       SEC
.,C041  E9 03    SBC #$03
.,C043  AA       TAX
.,C044  BC 98 C0 LDY $C098,X
.,C047  84 FF    STY $FF
.,C049  B9 00 C1 LDA $C100,Y
.,C04C  20 7A C0 JSR $C07A
.,C04F  30 06    BMI $C057
.,C051  E6 FF    INC $FF
.,C053  A4 FF    LDY $FF
.,C055  D0 F2    BNE $C049
.,C057  A9 20    LDA #$20
```

```
.,C059  20 7A C0 JSR $C07A
.,C05C  AD 8D 02 LDA $028D
.,C05F  8D 14 C0 STA $C014
.,C062  A9 00    LDA #$00
.,C064  85 CF    STA $CF
.,C066  4C 31 EA JMP $EA31
.,C069  48       PHA
.,C06A  18       CLC
.,C06B  A5 C5    LDA $C5
.,C06D  79 90 C0 ADC $C090,Y
.,C070  85 C5    STA $C5
.,C072  68       PLA
.,C073  60       RTS
.,C074  EE 14 C0 INC $C014
.,C077  4C 66 C0 JMP $C066
.,C07A  48       PHA
.,C07B  29 7F    AND #$7F
.,C07D  AE 86 02 LDX $0286
.,C080  20 13 EA JSR $EA13
.,C083  20 B6 E6 JSR $E6B6
.,C086  68       PLA
.,C087  60       RTS
.,C088  00       BRK
```

Bizonyára sok C64-tulajdonos panaszkodott már — gondolok azokra, akik nemcsak játszanak a gépen, hanem egy kicsit programoznak is —, hogy kevés a gépen található 8 funkcióbillentyű. A megoldás nagyon egyszerű: használjuk fel a CTRL és COMMODORE billentyűket is (a SHIFT-et eredetileg is használhatjuk). Ha megszámoljuk a variációkat, akkor látjuk, hogy a lehetséges 32-ből csak 28-at tudunk kihasználni, mert a SHIFT+COMMODORE együttes lenyomásának beépített funkciója van.

Az alábbi program ezt a megoldást mutatja be egy megszakítórutinon keresztül. A program tehát akkor is futhat, amikor BASIC-ben programozunk. A program $C000-tól kezdődik, így nem foglal le helyet a BASIC RAM-ból; 28 funkcióbillentyűt használ, minden <F> billentyűhöz egy BASIC utasítás van rendelve.

Rövidítések: CTRL->CT; SHIFT->SH; COMMODORE->C=

Brumi

```
    <F1> - PRINT      <SH+F1> - GOTO      <C=+F1> - LOAD
    <F3> - INPUT      <SH+F3> - GOSUB     <C=+F3> - SAVE
    <F5> - READ       <SH+F5> - POKE      <C=+F5> - OPEN
    <F7> - DATA       <SH+F7> - PEEK      <C=+F7> - CLOSE
<CT+F1> - RUN    <CT+SH+F1> - LIST   <CT+C=+F1> - FOR
<CT+F3> - STOP   <CT+SH+F3> - CONT   <CT+C=+F3> - NEXT
<CT+F5> - SYS    <CT+SH+F5> - CMD    <CT+C=+F5> - IF
<CT+F7> - END    <CT+SH+F7> - NEW    <CT+C=+F7> - THEN
     <CT+C=+SH+F1> - RESTORE   <CT+C=+SH+F5> - WAIT
     <CT+C=+SH+F3> - VERIFY    <CT+C=+SH+F7> - DIM
```

A funkcióbillentyűk táblázata

Csalás a véletlennel

A mai világban mindenre oda kell figyelni, mert lépten-nyomon becsapnak minket. Így van ez a számítástechnikában is, s már a véletlenben sem bízhat az ember. Különösen akkor nem, ha még ráadásul kiderül, hogy igencsak befolyásolható az is, amiről ezt egyáltalán nem tételezné fel az ember.

Nos, elég gyakran előfordul, hogy szükségünk van egy véletlenszámra. A mi esetünkben ez legyen most 2 és 12 között. Az ember rögtön azt mondja, hogy ezt stílszerűen két dobókockával való dobással kellene szimulálni. Ha ezt tesszük, meg is lesz az eredménye, ugyanis ebben az esetben sokkal nagyobb valószínűsége van annak, hogy hetet dobunk, mint annak, hogy kettőt vagy tizenkettőt. Elvileg egyenlő esélyre volna szükségünk, amit egy darab nagyobb oldalszámú „kockával" el is lehetne érni. A két módszer közötti különbséget jól szemlélteti az *1.* és a *2. ábra,* amely az eloszlásfüggvényt ábrázolja. (Ha három kockával dobnánk, még nagyobb lenne a valószínűsége, hogy közbülső értéket kapunk.)

Mindezt csak azért említettem meg, mert számítógépprogramjaink írása közben igen gyakran esünk hasonló hibába. Figyeljünk tehát arra, hogy véletlenszámokat oszthatunk, szorozhatunk, s hozzá is adhatunk valamit, de két véletlenszámot ne adjunk össze, mert az az előbbi torzulást vonja maga után.

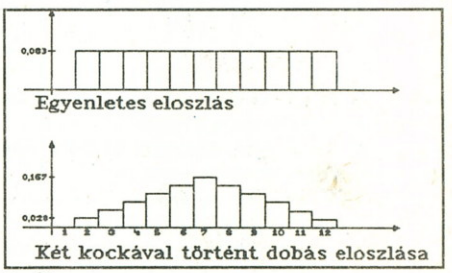
Persze ezt a lehetőséget ki is lehet használni, ha olyan programot akarunk készíteni, ami saját magunknak kedvez. Ekkor azonban az eloszlási görbét laposítani kell, hogy ez a kis csalás ne legyen feltűnő. Mivel például véletlenszámot nem lehet összeadni, ezért egy kis trükköt kell alkalmazni. Módosítsuk a fenti példát úgy, hogy az egyik véletlenszám 1 és 8, a másik 1 és 4 között változzon. A görbe csúcsát máris levágtuk. További hasonló fogások alkalmazásával egyéb kunkorokat is ki tudunk alakítani, s ha kihasználjuk azt, hogy mindez egy számítógépen történik, s azzal még egyéb műveleteket is el tudunk végeztetni, elég sokféle lehetőséget kínálhatunk magunknak.

Van azonban még egy dolog, amit nem szabad figyelmen kívül hagyni: mégpedig az, hogy a számítógép véletlenszám-generátora sem lineáris. A példa kedvéért nézzük meg a C64 és az IBM Turbo Pascal véletlenszám-generátorát. C64-en kétszer tízezer és egyszer százezer számot generáltam 0 és 1 között. A tartományt tíz részre osztva, a *táblázatban* és a *3. ábrán* látható értékek keletkeztek. IBM-en szerencsére nagyobbak voltak a lehetőségek, így ott egy százezres és egy egymilliós kísérletet csináltam. Érdekes módon az így keletkezett görbék igen hasonlítottak egymásra, amiből azt a következtetést lehet levonni, hogy a különböző programjaink által generált véletlenszámok is hasonló valószínűségi eloszlásúak lesznek.

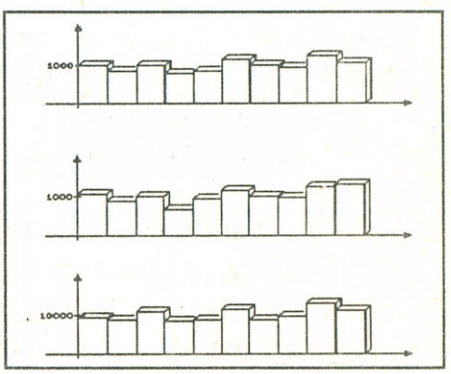
Ha ezt a két dolgot figyelembe vesszük, igen hatásosan tudunk majd csalni saját programjainkban.

Bártfai Barnabás

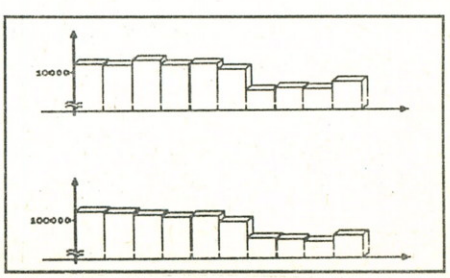
gép	C64	C64	C64	IBM	IBM
kísérlet	10000	10000	100000	100000	1000000
0.0-1.0	1007	1021	9958	10667	107340
0.1-0.2	952	976	9872	10614	107035
0.2-0.3	998	1001	1063	10776	106677
0.3-0.4	927	788	9831	10617	106339
0.4-0.5	956	945	9881	10672	106562
0.5-0.6	1074	1137	10124	10113	99494
0.6-0.7	996	990	9968	9057	91725
0.7-0.8	969	982	9987	9133	91654
0.8-0.9	1092	1109	10217	9088	91578
0.9-1.0	1029	1051	10099	9263	91596



1. ábra



2. ábra



3. ábra

Az EXDOS rendszerváltozói

A következő összeállítást azoknak az Enterprise-tulajdonosoknak ajánljuk, akiknek gépükhöz EXDOS lemezvezérlőjük is van. Az EXOS rendszerváltozóit az EXOS műszaki leírás és a Felhasználói kézikönyv egyaránt tartalmazza. Az EXOS rendszerbővítő modulok azonban deklarálhatnak egyéb rendszerváltozókat, amint ezt az EXDOS is teszi. Az alábbiakban ismertetjük az általa létrehozott rendszerváltozókat.

Szám	Röv.név	Funkció
64	ROM_ EXDOS	Annak a szegmensnek a száma, ahol az EXDOS program elérhető
65	IS_PO	Az ISDOS által használt szegmensek
66	IS_P1	
67	IS_P2	
68	IS_P3	
69	ECHO	On (0) A batch-fájl végrehajtása során minden üzenetet kiír a képernyőre (#255)
70	VERIFY	On (0) Az EXDOS ellenőrzést végez minden lemezre írás után
71	DEF_UNIT	(1-26) Az alap-meghajtó száma
72	BOOT_DRIVE	(1-26) Az ISDOS rendszer újratöltésére kijelölt meghajtó száma
73	STEP_RATE	A meghajtók fizikai léptetési ideje 0: 6 ms, 1: 12 ms, 2: 20 ms, 3: 30 ms
74	DSK_CHK	On (0) lemezellenőrzés minden fájl megnyitásakor
75	IN_ERROR	Annak a csatornának a száma, ahol az EXDOS a hibakijelzés visszajelzését intézi
76	OUT_ERROR	
77	IN_CLI	Annak a csatornának a száma, ahol az EXDOS a parancssorértelmezőt használja
78	OUT_CLI	
79	DT_FORM	EXDOS-dátum és -időformátum
80	ER_MISS	On (0) hibakijelzés engedélyezve
81	AB_ERR	Lemez-hibakód az .ABORT hibánál
82	DSK_ERR	Hiba az utolsó ISDOS parancsban
83	CLI_PROT	Az ISDOS parancssor-értelmezőt védő flag
84	RND-0	32 bites véletlenszám-generátor a lemez azonosításához a DISK I/0 műveleteknél
85	RND-1	
86	RND_2	
87	RND_3	
88	CLI_EN	On (0) Az EXDOS parancssorértelmező engedélyezve
89	DCH_DIS	Off (255) Az EXDOS azt hiszi, hogy lemezt nem cserélt, választ kapott a berendezéskezelőtől
90	FAST_VID	On (0) ISDOS gyors képernyőkezelő a tranziens utasítások részére
91	CD_PROMPT	On (0) Az aktuális útvonal kiíródik a promptban.

Az EXDOS létrehoz még nyolc rendszerváltozót (92-99). Kérjük a kedves olvasót, hogy ha információja van ezekről, írjon nekünk.

KO-GA

Variációk a képernyőre avagy

Néha még a gépi kód is lassú!

Ha az ember demo programot ír, vagy hosszú, keserves munkával írt programját látványosabbá akarja tenni, nem árt, ha van kéznél egy-két olyan képernyőkezelő rutin, amire bizton számíthat. Megírtam néhány ilyen assembly rutint, melyeket, remélem, sokan tudnak majd hasznosítani.

Mit is tudnak ezek? A 32768-as címtől elhelyezkedő, 6144 bájt hosszú, színek nélküli képernyőt a 16384-es címtől kezdve — ami a képernyőterület kezdete — valami furfangos módon bemásolják. Nagyképűen azt is mondhatjuk, hogy a Spectrumunk videotechnikát produkál. Akkor hát lássuk az első rutint!

NYIT RUTIN

A képátvitel a képernyő közepén kezdődik, majd lefelé és felfelé egyszerre folytatódik, miközben az eredeti képernyő folyamatosan törlődik. A rutin a 40008-as címtől kezdve helyezkedik el a memóriában, és innen is hívható. Figyelem! A 40000-40007-ig terjedő részen a program változókat tárol! *(Folytatjuk)*

KisPiroska Zoltán

```
    ORG  40008
    LD   HL,18559
    LD   (40000),HL
    LD   HL,20351
    LD   (40002),HL
    LD   B,4
L0  PUSH BC
    LD   HL,(40000)
    LD   B,32
L1  INC  HL
    DJNZ L1
    LD   (40000),HL
    LD   HL,(40000)
    DEC  H
    LD   (40004),HL
    LD   HL,(40002)
    LD   (40006),HL
    LD   B,8
L2  PUSH BC
    LD   HL,(40004)
    INC  H
    LD   (40004),HL
    LD   DE,(40004)
    LD   HL,(40004)
    LD   B,64
L3  INC  H
    DJNZ L3
    LD   BC,32
    LDDR
    LD   DE,(40006)
    LD   HL,(40006)
    LD   B,64
L4  INC  H
    DJNZ L4
    LD   BC,32
    LDDR
    HALT
    LD   HL,(40006)
    DEC  H
    LD   (40006),HL
    POP  BC
    DJNZ L2
    LD   HL,(40002)
    LD   B,32
L5  DEC  HL
    DJNZ L5
    LD   (40002),HL
    POP  BC
    DJNZ L0
    LD   HL,20479
    LD   (40000),HL
    LD   HL,18431
    LD   (40002),HL
    LD   B,8
L6  PUSH BC
    LD   HL,(40000)
    LD   B,32
L7  INC  HL
    DJNZ L7
    LD   (40000),HL
    LD   HL,(40000)
    LD   (40004),HL
    LD   HL,(40004)
    DEC  H
    LD   (40004),HL
    LD   HL,(40002)
    LD   (40006),HL
    LD   B,8
L8  PUSH BC
    LD   HL,(40004)
    INC  H
    LD   (40004),HL
    LD,  HL,(40004)
    LD   B,64
L9  INC  H
    DJNZ L9
    LD   DE,(40004)
    LD   BC,32
    LDDR
    LD   HL,(40006)
    INC  H
    LD   (40004),HL
    LD   HL,(40004)
    LD   B,64
L10 INC  H
    DJNZ L10
    LD   DE,(40006)
    LD   BC,32
    LDDR
    HALT
    LD   HL,(40006)
    DEC  H
    LD   (40006),HL
    POP  BC
    DJNZ L8
    LD   HL,(40002)
    LD   B,32
L11 DEC  HL
    DJNZ L11
    LD   (40002),HL
    POP  BC
    DJNZ L6
    RET
    END
```

TOP LISTA

Osztály, géptípus	Játék programok	géptípus	Felhasználói programok	géptípus
IBM AMIGA	Milleneum	Amiga	Ventura Professional 2.0	IBM
	3D Tetris	IBM	GEM Artline	IBM
	Kick Off	Amiga	AMI	IBM
C=128 C=64 C+4	Heroes of the Lance	C-64	GEOS 2.1	C-128
	Neuromancer	C-64	GeoPublish	C-64
	Pool of Radiance	C-64	Printfox	C-64
Enterp Spectr. TVC	How to be complete b.	Spect.	———	
	Bard's Tale III.	Spect.	———	
	Knightmare	Spect.	———	

bitkiller

Listánkat felhasználói, illetve játékprogramokból állítjuk össze. A legjobbakat, legérdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprise-ra, TVC-re, Atarira és IBM-re készült programrangsorokat várunk havonta.

Címünk:
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
Diákszerkesztőség

BASIC-bővítések Commodore 16-ra

5. rész

Az előző részekben az ERASE, WINDOW parancsokkal ismerkedtünk meg. Most egy újabb parancsra kerül sor, melynek neve: HOLD.

A szó jelentése tartás, felfüggesztés. Többször előfordul, hogy a program futását valamilyen okból lassítani kell, esetleg várakozó ciklusokat kell beépíteni a programba, például fényújság, villogtatás esetén. A várakozást ilyenkor egy FOR-NEXT ciklussal oldhatjuk meg.

Most nem a futásidő növekedése lehet probléma, hanem a program hossza, ami több késleltető programrész használatánál nyúlik, mint az a bizonyos rétes. Bár gyakorlott programozók ezt egy paraméterezhető késleltető szubrutinnal megoldják, de ez is hosszabb, mint egyetlen parancs. Kézenfekvő tehát, hogy bevezessünk egy parancsot a késleltetésre. Ez a HOLD parancs. Szintaxisa: HOLD X, ahol X egy 0-65535 közé eső szám. A késleltetési időt a t= x/100 képlettel kapjuk meg, másodpercben. Kiszámolható, hogy így közel 11 órás késleltetés is elérhető, ami azért abszurdum. 1 másodperc a késleltetése a HOLD 100 parancsnak.

A paraméter — a WINDOW parancshoz hasonlóan — nemcsak konstans lehet, hanem változó, kifejezés is. Például a HOLD 1000, az A=1000:HOLD A és a HOLD 100*10 mind egyenértékű, és 10 másodperces késleltetést valósít meg.

A program az *ábrán* látható. Működése egyszerű. Minden megszakításkor egy előre beállított számlálót csökkent. Ha a számláló 0-ra ér, jelenti a késleltetés végét. Ha a paraméter beolvasásakor valamilyen hibát tapasztal, akkor annak megfelelő hibaüzenetet ad. A rutin beágyazódik a megszakítási rendszerbe, függetlenül attól, hogy van-e még a megszakító rutinnak egyéb feladata.

Kádár Sándor

```
ass -16: pass 1 2
2000                    .opt p
2000                    *=   $2000
2000          tok       =    $030c
2000          detok     =    $030e
2000          vegre     =    $0310
2000          chrget    =    $0473
2000          irq       =    $0314
2000          xbe       =    $9d84
2000          vesszo    =    $9491
2000          frnum     =    $9314
2000          egesz     =    $a327
2000          be1       =    $0b
2000          be2       =    $49
2000          be3       =    $23
2000 a9 1f              lda  #<tok1
2002 8d 0c 03           sta  tok
2005 a9 20              lda  #>tok1
2007 8d 0d 03           sta  tok+1
200a a9 33              lda  #<detok1
200c 8d 0e 03           sta  detok
200f a9 20              lda  #>detok1
2011 8d 0f 03           sta  detok+1
2014 a9 3f              lda  #<vegre1
2016 8d 10 03           sta  vegre
2019 a9 20              lda  #>vegre1
201b 8d 11 03           sta  vegre+1
201e 60                 rts
                      ;
                      ;
201f 48       tok1      pha
2020 a9 20              lda  #>tabla
2022 a0 4f              ldy  #<tabla
2024 20 07 8a           jsr  $8a07

2027 68                 pla
2028 90 06              bcc  vege
202a a5 0b              lda  be1
202c 48                 pha
202d 4c d6 89           jmp  $89d6
2030 4c 6c 89 vege      jmp  $896c
                      ;
                      ;
2033 aa       detok1    tax
2034 84 49              sty  be2
2036 a0 20              ldy  #>tabla
2038 84 23              sty  be3
203a a0 4f              ldy  #<tabla
203c 4c 9c 8b           jmp  $8b9c
                      ;
                      ;
203f 38       vegre1    sec
2040 e9 80              sbc  #$80
2042 0a                 asl  a
2043 a8                 tay
2044 b9 55 20           lda  innen+1,y
2047 48                 pha
2048 b9 54 20           lda  innen,y
204b 48                 pha
204c 4c 73 04           jmp  $0473
                      ;
                      ;
204f 48 4f 4c tabla     .asc "holD"
2053 00                 .byte 0
                      ;
                      ;
2054 55       innen     .byt <megy-1
2055 20                 .byt >megy

                      ;
2056 20 14 93 megy      jsr  frnum
2059 a5 66              lda  $66
205b 30 47              bmi  hiba
205d a5 61              lda  $61
205f c9 91              cmp  #$91
2061 b0 41              bcs  hiba
2063 20 27 a3           jsr  egesz
2066 a5 64              lda  $64
2068 85 e1              sta  $e1
206a a4 65              ldy  $65
206c 84 e0              sty  $e0
206e 78                 sei
206f ad 14 03           lda  irq
2072 8d 59 05           sta  $0559
2075 ad 15 03           lda  irq+1
2078 8d 5a 05           sta  $055a
207b a9 aa              lda  #<irq1
207d 8d 14 03           sta  irq
2080 a9 20              lda  #>irq1+1
2082 8d 15 03           sta  irq+1
2085 58                 cli
2086 a5 91    cik1      lda  $91
2088 c9 7f              cmp  #$7f
208a f0 08              beq  veg
208c a5 e1              lda  $e1
208e d0 f6              bne  cik1
2090 a5 e0              lda  $e0
2092 d0 f2              bne  cik1
2094 78       veg       sei
2095 ad 5a 05           lda  $055a
2098 8d 15 03           sta  irq+1

209b ad 59 05           lda  $0559
209e 8d 14 03           sta  irq
20a1 58                 cli
20a2 18                 clc
20a3 60                 rts
20a4 4c 1c 99 hiba      jmp  $991c
20a7 6c 59 05 tov       jmp  ($0559)
20aa a5 e0    irq1      lda  $e0
20ac 10 04              bpl  a1
20ae c6 e0              dec  $e0
20b0 d0 f5              bne  tov
20b2 c6 e0    a1        dec  $e0
20b4 10 f1              bpl  tov
20b6 c6 e1              dec  $e1
20b8 4c a7 20           jmp  tov
2000-20bb
```

20 éves a UNIX

II. rész

Miért késik mindmáig a világmegváltás?

Már 20 éves? No, akkor már elég öreg lehet, ha a számítástechnika fél évszázados korszakához és rohanó élettempójához viszonyítjuk. Mégis COCOM-listán van! Szoftverfejlesztő rendszerként 20 évesen is „High Technology Product"-nak számít! Jó, tudom. A 20 év alatt a Unixszal is történt azért valami. Kezdeti gyors fejlődése azonban hamar alábbhagyott. Generális megújhodását ma éppen legjobban dicsért tulajdonsága, a hordozhatóság gátolja. A hordozhatóság és a szabványosság kedvéért, nekem úgy tűnik, elvtelenül gúzsba kötötték. Nem véletlen, hogy a szakma nagyjai (az IBM, a DEC stb., szóval a „hetek") nem bírták tovább nézni ezt a vergődést, és az Open Software Foundation (OSF) keretében új fejlesztési irányt próbálnak indítani

Az Unix születésekor szülőatyját egyáltalán nem világmegváltó eszmék gyötörték. Egyszerűen csak szeretett volna egy programfejlesztést minél gyorsabban befejezni. A Bell Lab hírhedt volt arról, hogy ott mindenféle számítógéptípus előfordul, és az ott dolgozó kutatók állandó kapcsolatban álltak az egyetemek és gyárak kutatóival. Az egyik ilyen balsikerű kapcsolat szüleménye volt a nagygépes MULTICS operációs rendszer.

A MULTICS-ot arra tervezték, hogy a programozók online módon tudjanak programot fejleszteni egy olyan nagyszámítógépen, amelyen korábban csak kötegelt üzemmódban lehetett dolgozni. A kötegelt üzemmódtól azért igyekeztek valamilyen módon megszabadulni a programozók, mert a fejlesztési munkákban az interaktivitás hiánya katasztrofálisan rossz hatású volt. Hogy ez mennyire igaz, arra álljon itt egy hirdetés, amelyet saját szememmel olvastam: „Duplázza meg programozói kapacitását, vegyen ... on line rendszert!". A hirdető szerintem még nem is ígért túl sokat.

Nos, ebből látható, hogy mit vártak a MULTICS-tól a programozók, de sajnos a MULTICS a várakozásokat nem teljesítette. Lassú és a méregdrága hardverbázis miatt drága volt. Sok felhasználó egyidejű kiszolgálása gyakorlatilag ábránd maradt. A MULTICS-projektben részt vevő, majd alkalmazásából kiábránduló Ken Thompson azért megtanult egy-két olyan elvet, ami egy jó programfejlesztő környezetben nélkülözhetetlen. Minden rosszban van valami jó is, a Unix kidolgozását ezekből kiindulva tudta megkezdeni. Maga a Unix elnevezés is a MULTICS nevéből ered, amit egy esti baráti szópóker-partin találtak ki. A Uni azt hangsúlyozta, hogy a fejlesztési munkákban az egyén munkája fontosabb, mint a hardver kihasználtságának növelése multiprogramozás útján, de a fejlesztés kárára. Az X a CS kiejtéséből maradt meg — a tréfa betetőzéseként.

Az eddigiek talán segítenek megmagyarázni, hogy a Unix a születésekor miért vonult el a tudósok elefántcsonttornyába. Ken Thompson tulajdonképpen ugyanazt mondta a Unix megalkotásával, amit Arkhimédész mondott a gall katonáknak: „Ne zavarjátok a köreimet!". Zűrzavarban nem lehet a lényegre koncentrálni. Egy IBM kutatási jelentésből kitűnik, hogy a programozói munka hatásfoka három másodpercnél hosszabb válaszidőknél már a tizedére esik. Emberünk agya elkezd elkalandozni! Ennek a jelentésnek az eredménye, hogy az IBM egyes kutatóhelyein ma több személyi számítógépet helyeznek el, mint ahányan ott dolgoznak.

A „csendes sarok" a Bell Labban a már emlegetett, kimustrált PDP-7-es gép mellett jött össze Ken Thompsonnak, aki akkorra a fejlesztésre szánt költségeknek jól a nyakára hágott a drága nagygépes MULTICS-környezetben. A szituáció lehet, hogy sokak számára ismerős a saját gyakorlatából. Ilyenkor kell vigyázni, nehogy újabb idétlen döntéssel súlyosbítsunk a helyzetünkön. Ken Thompsonnak azonban valószínűleg nemigen volt más alternatívája, de döntésének eredménye, a PDP-7-es bázis, mint már láttuk, alig két év alatt kihúzta alóla a talajt. Ez nem kis büntetés volt, de a kutyaszorítóból úgy mászott ki, hogy az már világmegváltásnak bizonyult. Persze, főleg csak a szoftverfejlesztők világában.

A Unix sokáig körözött a Bell Labon és az egyetemeken belül, amíg kikerült a „nagyközönséghez". Amivel azonban egy tudós vénájú alkat sikeresen dolgozik, az a hétköznapi emberek körében általában alig válik be. Ez a nóta megint ismerős lehet. Az ember nem képes akármekkora IQ-különbséget átlépni. Egyáltalán az IQ-ra oda kell figyelni. Amikor például a híres nyugatnémet Starfighterek katasztrófáinak okát kutatták, éppen az ellenkező irányú problémára derült fény. Feltűnt ugyanis, hogy Amerikában nem olyan sűrűek a Starfighter-balesetek. Amíg azonban Amerikában a pilóták kiválasztásakor egy alsó és felső IQ-korlátot írtak elő, addig a nyugatnémeteknél a legmagasabb IQ-t elérőket válogatták ki. Nos, a túl magas IQ-val rendelkező pilóták agya vészhelyzetben „túlpörgött", a szabványos eljárás helyett elkezdtek okoskodni, ami a vesztüket okozta.

Felmerül egy kérdés. Vajon miért kellett a PDP-n új operációs rendszert írnia Ken Thompsonnak? Talán a DEC-nek nem volt szoftvere a gépekhez? A baj az, hogy túl sokféle is volt, de egyik sem volt kompatibilis a másikkal. A DEC gépek azzal a szlogennel indultak világhódító útra a bigott nagyszámítógépes korszak kellős közepén, hogy ne vegyenek több számítógép-erőforrást, mint amennyi a feladat megoldásához feltétlenül szükséges. Abban a korban, amikor egy számítógépet csak úgy tudtak elképzelni, hogy egy hatalmas légkondicionált termet kell bebútorozni

szépre festett nagy szekrénnyekkel, a DEC ajánlata teljesen szokatlan volt.

A DEC azonban jó érzékkel találta meg a filozófiájának megfelelő alkalmazói kört, az ipari folyamatok vezérlését. A folyamatok vezérlése akkoriban elég egyszerű algoritmusokat jelentett, ami tényleg elég kevés számítógép-erőforrással is kiszolgálható volt. A DEC ezért elkezdett lecsupaszított gépeket gyártani, amelyekben volt egy kis szóhosszúságú processzor (a mért jelek pontossága úgyis kicsi volt!), és valami minimális tár az üzemindításkor egyszer betöltendő program és az adatok tárolására. A perifériát szellemesen rendkívül ökonomikusan oldották meg telexírógéppel, amely egyszerre négy perifériát biztosított: írógép, valamint lyukszalag be- és kimenetet. A programot lyukszalagon tárolták, és így arról kellett betölteni.

Ebben a helyzetben nagygépes értelemben vett operációs rendszerre egyszerűen nem volt szükség. Később persze az ipari környezet étvágya mohón nőni kezdett, de a konfigurációk igen erős eltérése miatt egységes operációs rendszer készítése lehetetlen volt. A minigépes korszak történeti körülményei tehát egy alapszoftver-anarchiához vezettek. Ken Thompsonnak a DEC-hagyatékot éppúgy ki kellett hajítania az ablakon, mint a MULTICS-ot. A Unix ezzel tulajdonképpen világmegváltó lehetőséget kapott már az induláskor, hiszen alkalma lett volna rendet teremteni a DEC-szoftver háza táján. De nem így történt. A DEC-et a Unix nem érdekelte. Nem értette meg a jelentőségét, és ez talán nagy hiba volt.

A Unix az első világmegváltási kísérletről lemaradt, a DEC fejlesztői pedig egymással konkurálva fejlesztgették tovább saját operációs rendszereiket. Ez az elbonyolító korszak sajnos most a VAX/VMS-nél tart. Sovány vigasz, hogy amikor a nyolcvanas évek elején már szinte minden nagy beszállt a Unix-ringbe — amelyek közül a legnagyobb pszichológiai hatású a Hewlett Packard cég döntése volt —, végre a DEC is felvette szoftverpalettájára a Unixot Ultrix néven.

Amikor az Unix először kikerült a Bell Labból, akkor a labor szoftvert még nem forgalmazhatott. A 6-os verziójú Unix tehát nem azért került ki az egyetemekre forrásnyelvvel együtt, mert a Bell Lab a piacon addig teljesen szokatlan politikával akart indulni, hanem azért, mert nem tudott volna eleget tenni garanciális kötelezettségeinek (nálunk ez a probléma ugyan kit izgatna?). Alternatív megoldás tehát csak az lehetett, hogy az ügyfél maga javítsa a hibákat, ha tudja. Ehhez volt nélkülözhetetlen a forrásnyelvi anyag.

Persze az egyetemeken koncentrálódott nagy szellemi kapacitásnak óriási kihívás volt a forrásnyelvű Unix, így nem véletlen, hogy azonnal nekiestek a módosításának. Ezek rendszerint valami javítási kísérletből indultak. A Bell Lab ezt a beruházást végül is elég nagy haszonnal zárta. A 7. verzióban már szerepelt az egyetemekről érkező módosítások színe-java. A 6. verzió volt alapja az egyik legerősebb egyetemi fejlesztőhely, a Berkeley Egyetem (University of California, Berkeley: UCB) Unix-változatainak: 16 bites: bsd 2.x, 32 bites: bsd 4.x (bsd=berkeley system distribution).

Az összes többi lényeges fejlesztés már a 7. verzióból indult, tehát elég lényeges pont a Unix fejlődésében. Nekem az a véleményem, hogy innen kezdett a Unix ismét a világ-meg-nem-váltó irányba indulni, mert létrejött egy sereg inkompatibilis változat. A Unix-hívők ezt a korszakot éppen fordítva, a Unix világhódító útjának hitték. A Unix a nagygépektől a mikrogépekig minden gépkategóriára átterjedt, ami látszólag növekedést mutat. Amikor azonban a Bell Lab az AT&T felosztása után végre bekerülhetett a szoftverpiacra, meglepetéssel tapasztalhatta, hogy az akkor még háborítatlan uralmat gyakorló nagyszámítógépes piac figyelemre sem méltatja a System II. néven ajánlgatott nagygépes verziót. A DEC hozzáállása már ismert, így a minigépes piacon is lanyha volt az érdeklődés. Az AT&T és a Bell Lab kénytelen volt az újonnan felvett mikrogépes piacon bedobni egyszülött fiát System V.-ként. Ez végre sikert hozott.

Ekkor minden Unix-hívő úgy gondolta, hogy a mikrogépes fronton a Unix végre meg fogja váltani a világot. Ember tervez, IBM végez. Mint ismeretes, az IBM PC-k nem Unixszal születtek meg. Ezt egyértelműen indokolta az induláskor rendkívül szűk konfiguráció (16 k-s változat is volt, hát hogy lehet abban Unixot elképzelni?). Az IBM PC-k ezért a CP/M-től kapták az indító munícióit. Ez határozta meg az MS-DOS kiinduló paramétereit.

A PC-k hardverkonfigurációja talán még annál is gyorsabb ütemben nőtt, mint annak idején ez a minigépeknél történt. A fejlődés alig két év alatt kinőtte az eredeti CP/M korlátokat, de a Unix hardverigényét még mindig nem tudta teljesíteni. Közbenső megoldás született: az MS-DOS 2.0-s verziója egy sereg dolgot ripsz-ropsz átvett a Unixból, hogy a legsürgetőbb konfigurációs korlátokat le tudja győzni (már egy 5 Mbájtos kis XT winchesterén is tarthatatlan volt, hogy az egyetlen katalógusban csak 255 állomány lehet, tehát kellett a Unix hierarchikus katalógusfája!). Mire aztán a PC-vonal 1984-ben kitermelte a Unixok hardverigényeit megoldó AT-ket, az MS-DOS már óriási szoftverarzenállal rendelkezett a piacon. Hiába kérte meg az IBM a Microsoftot, hogy gyorsan adaptálja saját Unix-változatát, a Xenixet az AT-re, a Unix már lekéste a vonatot, és máig is késésben van. A világmegváltás ismét nem sikerült.

A Xenixszel elég sok baja volt az IBM-nek. Míg az MS-DOS környezetekkel alig kellett foglalkoznia, a Xenix installációk folyton panaszkodtak valami miatt. Az IBM tehát konstatálta a Unix gyengéit, ezért nem erőltette túlzottan a fejlesztését. Így a Xenix is úgy járt, mint az első Unix-változatok. Egyik-másik alkalmazó buzgóbb fejlesztőnek bizonyult, mint a Microsoft vagy az IBM. Ezek közül ma kiemelkedő a már említett Santa Cruz Operation cég, amely a Santa Cruz-i Egyetem kutatóiból verbuválódott. Az SCO Xenix nevet ma minden magára adó unixos ismeri.

Az SCO Xenix kifejezetten jó Unix-környezetet nyújt, különösen a 386-os PC-ken, amelyeken megoldott az MS-DOS és a Xenix közötti kölcsönös kapcsolat. Az SCO Xenix ezzel legalább menteni tudja az MS-DOS világban feltornyosult szoftverarzenált. Xenix /386 alatt ráadásul több MS-DOS-feladat is futhat, alkalmas a legkülönbözőbb hálózati kapcsolatokra, és ma már a Xenix alkalmazói szoftverajánlat is elég jó (például az MS-WORD, a FOXBASE, a Porfolio csomag irodai alkalmazásokhoz, az X-WINDOW CAD/CAM-hoz stb.). Egy kisebb környezetben, AT/386-ot vagy PS/2-70/80-at „vezénylő" gépekkel, valamint 256-os AT/XT-s (MS-DOS-emulációra már alkalmatlan) profi munkaállomásokkal, esetleg tökéletes védelmet már nem biztosító XT-kkel és olcsó terminálok seregével felszerelt Xenix hálózattal ma is többre jutunk, mint egy LAN-nal, ami előszeretettel elrabolja legjobb gépünket server céljára, és aztán kínlódhatunk butácska XT munkaállomásokon.

A jövő a LAN-ok integrálódása a Unixszal. A Novell cég óriási lépést tett ebbe az irányba a portábilis, C nyelvre átírt Advenced NetWare (ANW) kibocsátásával. A hatékonyság növeléséhez azonban más fontos dolgokat is elleshetne a Unix az ANW-től, így a gyorsabb működést megengedő lemezterület-kiosztási módszert és a sokkal hatékonyabb katalóguskezelést. Ezek azonban kényes pontok, mert sajnos éppen a Unix legszabványosabb elemeit érintik. Vajon ki nyeli le először a békát a rossz szabványelemek kidobásával?

Az eddigi szubjektív elemzésen túl nekifogtam egy keményebb, tudományos igényű elemzésnek is. Ennek célja a Unix gyenge pontjainak céltudatosabb kipuhatolása. A legutoljára született operációs rendszerek, akárhogy nézzük is a dolgot, a PC-s operációs rendszerek. Ezekben csapódhatott le tehát az operációs rendszerek készítésének eddigi tapasztalata. A Unixot ezért a PC-s operációs rendszerek körében hasonlítottam össze, már amennyire a felvett operációs rendszerek jellemzőit sikerült megtalálnom vagy plauzibilis alapon hihetőnek véltem. A legvizenyősebb terület a Concurrent DOS, amit *a következő számban* tanulmányozható táblázat CDOS-ként rövidít. A homályosságban ezt követi a CP/M-86.

Zsadányi Pál

HIRSCHMANN-ia

Jelen ismertetésünkkel (reklámmal?) nem bicskanyitogatás elérése a célunk. NSZK-beli tudósítónk, Ernst Demianiuk csak azt szerette volna bemutatni, hogy a — reméljük nem túl távoli — jövőben mi fog „begyűrűzni" hozzánk is.

Az 1989-ben Berlinben megrendezett nemzetközi híradástechnikai kiállítás slágere az első nyugatnémet DFS-KOPERNIKUS nevű távközlési műhold volt. Ez kettős polarizációjú tv-műsorok sugárzására alkalmas, két különböző frekvenciasávban.

A nálunk is jól ismert HIRSCHMANN cég e műsorok vételére alkalmas, 85 cm átmérőjű parabolaantennát fejlesztett ki, melynek fejegysége lehetővé teszi az említett függőleges, illetve vízszintes polarizációjú jelek vételét a 11, valamint a 12,5 GHz-es tartományban, mono és sztereo üzemmódban. A kívánt üzemmód kiválasztását mikroprocesszoros vezérlőegység végzi.

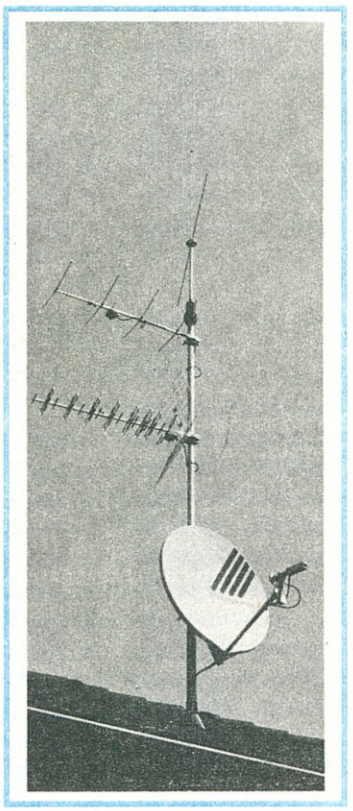
A tavaly június 6-án pályára bocsátott KOPERNIKUS műhold 12 német nyelvű tv-programot és 16 rádióműsort sugároz a digitális kép-, illetve hangminőségnek megfelelően. Ez év tavaszán lövik fel a DFS-2 nevű szatellitot, így lehetővé válik, hogy a DFS-1 telefonbeszélgetési és adatátviteli feladatokat láthasson el.

A HIRSCHMANN által gyártott 55 cm-es parabolaantenna is megfelelne a 12,5 GHz-es adások vételére, de a SAT 1, az RTL PLUS, az ARD 1 PLUS, valamint a 3SAT adók műsorai a 11 GHz-es frekvenciatartományban csak a 85 cm-es HIRSCHMANN parabolaantennával foghatók tökéletesen.

Akik nyaralásuk időtartama alatt sem szeretnének kedvenc tv-műsoraikról lemondani, és ráadásul még olyan szerencsések is, hogy lakókocsival indulhatnak útnak, azok felszerelhetik azt az e műsorok vételére alkalmas HIRSCHMANN vevőantennával.

A berendezést, amely magába foglalja a parabolaantennát, a polarizációs konvertert, valamint azt az egységet, amely lehetővé teszi az űrből érkező jelek fogadását a hagyományos tévékészülékek számára, 2100 DEM-től lehet megvásárolni.

A HIRSCHMANN megfelelő rendszereket biztosít a többi műhold (ASTRA, EUTELSAT, INTELSAT stb.) adásának vételéhez is.



Rezidens vírusdetektor

Az alábbi program, mint a neve is mutatja, egy vírusfelismerő program. Abban különbözik más hasonló programoktól, hogy nem végez teljes körű vírusvizsgálatot a lemezen lévő összes programon. Ehelyett állandó védelmet nyújt a Péntek 13, a Potyogós és a Reteelős vírusok ellen. Ezeket a vírusokat csak felismeri, de nem irtja ki őket.

A vírusok felismeréséről

A felismerés nem bizonyos bájtok keresésével, hanem vírusspecifikus megszakításhívások (különleges kóddal meghívott DOS-megszakítások) figyelésével történik. Ez gyorsítja a program működését. A Potyogósnak és a Péntek 13-nak vannak rezidens — futás után is tárban maradó — részei. A vírus amikor elindul, megvizsgálja, hogy bent van-e már a tárban, és ebben a pillanatban buktatható le, mert egy különleges kóddal hívja meg a DOS-vektort, INT 21H-t. A kód Péntek 13 esetén AH=E0H, Potyogós esetén AX=4BFFH. Ha tehát egy vírusfertőzött program meghívja a DOS-vektort ezekkel a kódokkal, akkor az RVD felismeri, és figyelmeztet, hogy ezt vagy azt a fajta vírust észlelte. Péntek 13 vírus esetén meggátolja a fertőzött program végrehajtását, a Potyogóst viszont engedi tovább futni, mert a DI regiszterben 55AAH-val visszatérve a megszakításból „át lehet verni" a Potyogóst.

A Potyogós itt ugyanis azt hiszi, hogy már bent van a tárban, és rátér a megfertőzött eredeti program futtatására. Magyarul: nem marad bent a tárban, nem tud fertőzni, és még a fertőzött program is hibátlanul lefut (lásd a Védőoltás című cikket a Mikromagazin 1989/9. számában).

A Resetelős vírust már bonyolultabb felismerni. Itt nem kerülhető el a részletesebb ellenőrzés. A Resetelős AH=40H-val hívja meg a DOS-vektort. Ez a DOS állományíró funkciója. Ha a vírus ezt a funkciót hívja meg, és a regiszterek egy bizonyos értéket vesznek fel, továbbá a kimentendő bájtok megegyeznek egy mintával, akkor az RVD Resetelős vírust jelez, és megszakítja a program végrehajtását.

A már tárban lévő vírusok

A program figyelmeztet, ha a Péntek 13 vagy a Potyogós már a tárban van. A felfedezés módja: a DOS-vektort meghívjuk a két különleges kóddal, és ha valamelyik vagy mindkét vírus bent van, akkor rájuk jellemző különleges kóddal válaszolnak. Az RVD ezek alapján lép akcióba.

Vírusfertőzés-védelem

Az RVD felfedezi, ha őt is vírustámadás éri. Ez a védelem kontrollösszeg-számításon alapul. Kontrollösszeget úgy számol, hogy betölti önmagát, és bájtonként végighaladva, a program összes bájtját hozzáadja a DX regiszterhez. Mivel a DX túlcsordulásával nem foglalkozik, egy 16 bites kontrollösszeget kap. Ezt összehasonlítja a program írásakor készített kontrollösszeggel, és ha nincs egyezés, akkor figyelmeztet a veszélyre. De ez még semmi! Ha megfertőzték, képes magából kiirtani a vírust. A védelem elvéből adódóan a legkülönfélébb vírusok sem okoznak neki nehézséget.

Bármilyen vírust ki tud irtani magából, nemcsak a Péntek 13-at, a Potyogóst és a Resetelőst. Kihasználja, hogy a vírusok általában, rejtőzködési szempontok miatt, lefutásuk után átadják a vezérlést a megfertőzött programnak, amelynek még tökéletesen kell futnia. Emiatt az RVD-t vírusfertőzés után eredeti állapotába visszaállíthatjuk. A visszaállítás és ezzel a vírusirtás elve: a program a fertőzés felismerése után a tárból kimenti önmagát RVD.COM néven, előtte azonban a fertőzött állapotát átnevezi VIRULENS.RXX-re. A név kiterjesztésében az X-ek helyén különféle véletlenszám-generátorral előállított alfanumerikus karakterek állhatnak.

Ezt az esetleges többször előforduló vírusfertőzések dokumentálására használja. (Szerintem manapság felelőtlenség ilyen önvédelemmel nem rendelkező szoftvert írni.) A program begépelése a mellékelt BASIC nyelvű beíróprogrammal végezhető, amit a GWBASIC vagy a QuickBASIC című programok elfogadnak. A BASIC-beíró begépelése után lehet a hexadecimális kódban megadott számokat beadni a PC-beírónak. A legelső sorban levő két bájt az RVD hossza, először ezt kell begépelni. Utána jönnek az RVD bájtjai, minden sor végén ellenőrzőösszeggel. A beíróprogram a beolvasott adatokat 16 bájtonként felírja az RVD.COM nevű állományba.

ifj. Zsadányi Pál

```
5 cls:print "PC-prg beiro v1.0":print
10 open "o",#1,"rvd.com":dim a(16)
20 gosub 500:ho=a:gosub 500:ho=ho+a*256:print
30 eo=0:print "Soreleje:":for e=1 to 16:gosub 500
40 eo=(eo+a) and 255:a(e)=a:gosub 900:if e=8 then print
50 next e:print" ";:gosub 500
60 if a<>eo then print "Az ellenorzo osszeg hibas, kere
sort ujra!":goto 30
70 for e=1 to 16:print #1,chr$(a(e));:ho=ho-1:if ho=0 t
close #1:end
80 next e:print ho" byte van meg hatra.":print:goto 30
500 input r$:a=val("&h"+r$):return
900 for f=1 to e:print right$("0"+hex$(a(f)),2)" ";
910 if f=8 then print "- ";
920 next:print:return
```

```
9E 04
E9 09 01 52 56 44 3A 20 - 70 31 33 20 76 69 72 75  F3
73 0D 0A 24 52 56 44 3A - 20 41 62 6F 72 74 61 6C  B9
6F 6D 20 61 20 70 72 67 - 20 76 65 67 72 65 68 61  C8
6A 74 61 73 61 74 2E 0D - 0A 24 52 56 44 3A 20 72  A8
65 73 65 74 65 6C 6F 20 - 76 69 72 75 73 0D 0A 24  85
52 56 44 3A 20 70 6F 74 - 79 6F 67 6F 73 20 76 69  C9
72 75 73 2C 20 6E 65 6D - 20 68 61 67 79 6F 6D 20  AB
74 65 72 6A 65 64 6E 69 - 21 0D 0A 24 00 00 00 00  B1
53 80 FC FF 75 05 B8 34 - 12 5B CF 80 FC E0 75 17  58
8C C8 8E D8 BA 03 01 B4 - 09 CD 21 BA 14 01 B4 09  AF
CD 21 B8 00 4C CD 21 80 - FC 40 75 3B 81 F9 88 02  50
75 18 8B DA 81 3F 51 BA - 75 10 81 7F 04 FC 8B 75  42
09 8C C8 8E D8 BA 3A 01 - EB CD 83 F9 05 75 15 8B  06
DA 81 3F EA F0 75 0D 81 - 7F 02 FF 00 75 06 80 7F  71
04 F0 74 DD EB 20 90 80 - FC 4B 75 1A 3C FF 75 16  FC
1E 50 52 8C C8 8E D8 BA - 50 01 B4 09 CD 21 5A 58  E2
1F BF AA 55 5B CF 5B 2E - FF 2E 7C 01 BA 69 03 B4  14
09 CD 21 BE 00 01 BF 9E - 05 B9 9E 04 F3 A4 E8 A0  92
00 73 07 B4 09 CD 21 EB - 47 90 A1 85 05 39 06 9C  ED
05 74 2D BA DF 04 B4 09 - CD 21 E8 E9 00 73 05 B4  EB
09 EB 2D 90 E8 7A 00 73 - 07 B4 09 CD 21 EB 21 90  D4
A1 85 05 39 06 9C 05 74 - 07 BA FF 04 B4 09 CD 21  EE
B4 FF CD 21 3D 34 12 75 - 0C BA B0 04 B4 09 CD 21  BE
B8 00 4C CD 21 B8 21 35 - CD 21 89 1E 7C 01 8C C0  5E
A3 7E 01 B8 FF 4B CD 21 - 81 FF AA 55 75 07 BA 8D  54
04 B4 09 CD 21 B4 E0 CD - 21 80 FC E0 74 07 BA 6E  30
04 B4 09 CD 21 BA 80 01 - B8 21 25 CD 21 BA CF 04  63
B4 09 CD 21 BA 0C 02 B1 - 04 D3 EA 42 B8 00 31 CD  DD
21 B8 00 00 A3 85 05 BA - 87 05 B9 00 00 B0 00 B4  69
```

```
3D CD 21 8B D8 73 06 BA - 74 05 EB 49 90 B9 B8 0B  7A
BA 9E 05 B4 3F CD 21 73 - 06 BA 74 05 EB 37 90 8B  27
C8 8B F8 80 BD 9D 05 1A - 75 02 49 4F 83 F9 00 74  43
19 49 49 8B 16 85 05 BE - 9E 05 26 8A 04 34 56 32  A7
E4 03 D0 46 E2 F4 89 16 - 85 05 B4 3E CD 21 73 05  54
BA 74 05 C3 F8 C3 E4 00 - 24 0F 04 41 A2 99 05 E4  31
00 24 0F 04 41 A2 9A 05 - BA 87 05 BF 8F 05 B4 56  5C
CD 21 BA 87 05 B9 00 00 - B0 01 B4 3C CD 21 8B D8  DF
73 06 BA 74 05 EB 11 90 - BA 9E 05 B9 9E 04 B4 40  E4
CD 21 72 04 B4 3E CD 21 - C3 52 56 44 20 52 65 73  3D
69 64 65 6E 74 20 56 69 - 72 75 73 20 44 65 74 65  EF
63 74 6F 72 20 56 31 2E - 35 0D 0A 5B 43 5D 20 43  37
6F 70 79 6C 65 66 74 20 - 31 39 38 39 20 62 79 20  19
50 61 75 6C 20 5A 73 61 - 64 61 6E 79 69 20 48 47  A4
35 42 49 52 0D 0A 4B 4B - 56 4D 46 20 49 6E 66 20  05
49 2E 2F 33 2E 0D 0A 53 - 68 61 72 65 77 61 72 65  C0
2C 20 66 72 65 65 20 73 - 6F 66 74 77 61 72 65 0D  86
0A 0A 46 65 6C 69 73 6D - 65 72 65 6D 20 61 20 70  2E
31 33 2C 20 61 20 70 6F - 74 79 6F 67 6F 73 20 65  3A
73 20 61 20 72 65 73 65 - 74 65 6C 6F 73 20 76 69  E9
72 75 73 6F 6B 61 74 20 - 61 20 70 72 67 2D 6B 65  F0
6E 2C 20 64 65 20 6E 65 - 6D 20 69 72 74 6F 6D 20  4E
6B 69 20 6F 6B 65 74 2E - 0D 0A 45 73 7A 72 65 76  6B
65 73 7A 65 6D 2C 20 68 - 61 20 62 65 6E 74 20 75  97
6C 20 76 61 6C 61 6D 65 - 6C 79 69 6B 20 61 20 6D  C9
65 6D 6F 72 69 61 62 61 - 6E 2E 0D 0A 0A 24 52 56  C9
44 3A 20 61 20 6D 65 6D - 6F 72 69 61 62 61 6E 20  5A
76 61 6E 20 61 20 50 31 - 33 21 0D 0A 24 52 56 44  E2
3A 20 61 20 6D 65 6D 6F - 72 69 61 62 61 6E 20 76  8C
61 6E 20 61 20 70 6F 74 - 79 6F 67 6F 73 0D 0A 24  2F
52 56 44 3A 20 4D 61 72 - 20 69 6E 73 74 61 6C 6C  7D

61 6C 76 61 20 76 61 67 - 79 6F 6B 21 0D 0A 24 52  03
56 44 20 69 6E 73 74 61 - 6C 6C 65 64 0D 0A 24 52  07
56 44 3A 20 4D 65 67 66 - 65 72 74 6F 7A 6F 74 74  FE
20 65 67 79 20 76 69 72 - 75 73 21 07 0D 0A 24 52  73
56 44 3A 20 4F 52 49 41 - 53 49 20 48 49 42 41 21  10
20 4E 45 4D 20 54 55 44 - 4F 4D 20 4B 49 49 52 54  4C
41 4E 49 20 4D 41 47 41 - 4D 42 4F 4C 20 41 20 56  0F
49 52 55 53 54 21 0D 0A - 20 53 75 72 67 6F 73 65  D7
6E 20 74 6F 72 6F 6C 6A - 20 6C 65 2C 20 65 73 20  5D
63 73 65 72 65 6C 6A 20 - 6B 69 20 65 67 79 20 65  C6
72 65 64 65 74 69 20 52 - 56 44 2D 72 65 21 07 07  BC
07 0D 0A 24 52 56 44 3A - 20 44 4F 53 20 45 72 72  B7
6F 72 0D 0A 24 00 00 52 - 56 44 2E 43 4F 4D 00 56  6B
49 52 55 4C 45 4E 53 2E - 52 56 44 00 78 A3 00 00  57
```

Programozási fogások és melléfogások

Legutóbb — olvasói észrevételek hatására — azzal a kérdéssel foglalkoztam, hogy érdemes-e a BASIC szubrutinokat a program elejére tenni. Most, szintén az olvasóktól kapott levelek nyomán, az e sorozat keretében elkövetett, talán nem anynyira programozási, mint inkább didaktikai baklövéseimről írok.

A Magazin 1989. augusztusi számában többek között a *Mikrovilág* 1989/8. számában közölt — az amerikai *RUN* magazinból átvett — *Form Writer* menükezelésével foglalkoztam. Csak utaltam arra, hogy az eredeti program helytelenül kezeli a nullás menüpontot, de ezt a hibát a módosított változatban is benne hagytam. Pásthy Béla Tatabányáról az eredeti programrészt az *1/a. listán* látható módon javította. (Módosítását aláhúzással jelzem.) Sajnos figyelmen kívül hagyta, hogy a VAL függvény feleslegessé válik, azért a 8210-es sor az *1/b. lista* szerint egyszerűsíthető. A levélből az is kiderül, hogy az általam adott módosítás javítása sokkal nehezebb, de szerintem felesleges is. Mindig az egyszerűbb megoldást kell keresni, még ha nem is sikerül hamar megtalálni.

A szeptemberi számban olyan esetet elemeztem, amelyben a gépi kód használata felesleges, nem növeli a program hatékonyságát, csupán a terjedelmét növeli. Az általam javasolt megoldásban — melyet az ottani 2. listán mutattam be — *látszólag* minden rendben van. Dr. Hack Frigyes levélben hívta fel a figyelmemet a FOR...NEXT ciklus alkalmazásának veszélyeire. Emlékeztetőül a *2/a. listán* idézem programomnak azt a részletét, amelynek 442-446-os soraiban található az ominózus ciklus. Figyelmen kívül hagytam a szabályt, amelyet sajnos a szakirodalom is méltatlanul mellőz, s ritka kivételként Ligeti Gábor és Szervánszky György *A ZX Spectrum programozása* című, a SZÁMALK-nál 1985-ben kiadott könyvének 40. oldalán olvasható: „A ciklusból általában kiugrani sem szabad, de lehet, mert ezt a BASIC nem tudja ellenőrizni. Ajánlatos megszokni, hogy *a ciklust csak a NEXT utasításnál hagyjuk el.*"

Milyen veszélyeket rejt a ciklusból való kiugrás? Az egyik a különböző ciklusok összegabalyodása, aminek eshetőségét a NEXT utáni ciklusváltozó elhagyása még tovább növeli. A másik, hogy a verem betelik, s ez *OUT OF MEMORY* hibaüzenettel a program leállásához vezethet. A szeptemberi számban megjelent teljes programot tanulmányozva megállapíthatjuk, hogy egyik veszély sem fenyeget: nincs több ciklus, ami összekeveredhetne, és a választható menüpontok kizárják a verem betelésének lehetőségét. Mi hát a baj? Az, hogy aki a bemutatott algoritmust átveszi, sokkal bonyolultabb programokba építi be, ami által az említett hibák előfordulásának valószínűsége sokszorosára nő. Olyan módszereket kell bemutatni, amelyek minden esetben biztonságosan működnek, a kockázatosak legfeljebb elrettendő példa gyanánt.

Mindezek figyelembevételével a *2/b. listán* látható módszer alkalmazását ajánlom. Remélem, senkit nem zavar, hogy az IF...GOTO utasításokkal megvalósított ciklus lassúbb a FOR...NEXT ciklusnál, hiszen — éppúgy, mint a BASIC változatnak az eredeti gépi kódéval szembeni lassúsága — a használatban egyáltalán nem érzékelhető.

A szeptemberi számban említettem még, hogy az ott bemutatott algoritmus változatlanul alkalmazható bármely Commodore gépre, a VC20-tól a C128-on az INSTR függvénnyel a *2/c. listán* látható módon *egyszerűbben* megoldható a feladat. C64-en a Simon's BASIC-ben a PLACE függvény nyújt lehetőséget hasonlóra, de vigyázzunk, mert a paraméterek sorrendje fordított az INSTR-hez képest.

Az ON utasításokban GOTO helyett GOSUB is lehet, a program struktúráját ennek megfelelően kell kialakítani. A GOSUB számomra rokonszenvesebb, saját programjaimban is ezt használom. A bemutatott példákon a kiindulási programhoz való nagyobb hasonlóság érdekében választottam az ON GOTO-t.

Végezetül Hack Frigyes hozzájárulásával a *3. listán* idézek leveléből egy rövid programot, amely a ciklusok összekeveredését mutatja be. A programot először a 25-ös sor nélkül kell lefuttatni, majd azzal kiegészítve. (A csillagokat természetesen nem kell beírni.) Kipróbálásakor — bár sejtettem, hogy mire számíthatok — az eredmény láttán nagyon meglepődtem. Érdemes azt is megnézni, mi történik, ha a NEXT után a ciklusváltozókat szabályosan beírjuk.

Barna László

1/a. lista

```
     ...
8200 GOSUB 100:ON VAL(A$)GOTO 8300,8350,
     8370,8390,8430
8210 IF VAL(A$)=0 AND A$="0" GOTO 3000
8220 GOTO 8000
     ...
```

1/b. lista

```
          ...
       8210 IF A$="0" GOTO 3000
          ...
```

2/a. lista

```
     ...
440 GET A$
442 FOR A=1 TO 3
444 IF A$=MID$("(F1)(F3)(F5)",A,1)  THEN
    450
446 NEXT : GOTO 440
450 ON A GOTO 460,470,480
     ...
```

2/b. lista

```
     ...
440 GET A$
442 A=1
444 IF A$=MID$("(F1)(F3)(F5)",A,1)  THEN
    450
446 A=A+1 : IF A<4 GOTO 444
450 ON A GOTO 460,470,480 : GOTO 440
     ...
```

2/c. lista

```
     ...
440 GET A$
450 ON INSTR("(F1)(F3)(F5)",A$) GOTO 460
    ,470,480 : GOTO 440
     ...
```

3. lista

```
      10 FOR K=1 TO 10
      20   FOR B=1 TO 10
***   25     IF B>5 THEN 50  ***
      30     PRINT B;
      40   NEXT
      50   PRINT K
      60 NEXT
      70 END
```

Időkijelzés és adatforgalom

Kísérletezzük ki, ha másképp nem megy!

Két programot teszünk közzé, az egyiket a hasznossága, a másikat az érdekessége miatt. Szívesen vennénk, ha a szerző által leírt problémákat valaki meg tudná válaszolni.

Elsőnek egy időkijelző rutin működését írom le. Sokat olvastam már a gép felhasználói megszakításáról, azonban arról, hogy hosszabb rutinokat — mintegy 100 bájt — hová lehet elhelyezni a memóriában, nem találtam leírást. Elkezdtem hát keresgélni olyan címeket, amelyek nem érintik a BASIC-területet, így elég átírni a User-Interrupt jelzőbájtját, valamint találunk mögötte száz bájtnyi szabad területet. Ezenkívül a 0-ás lapon kellett keresnem, mivel ez mindig be van lapozva. Végül találtam egy területet, amelyet a BASIC nem ront el. Hangsúlyozom, hogy választásomnak semmiféle tudományos alapja nincs, csupán programokat futtattam BASIC-ben, így kísérleteztem ki ezt a helyet.

A rutin nagyon egyszerű működésű, assembler listájából *(1. lista)* minden kiderül. Érdekességképpen megjegyzem, hogy a programban az idő olvasását először EXOS 32-vel akartam elvégezni, de ez valamilyen szemetet jelenített meg az állapotsorban, magyarázatot nem találtam. A programhoz az egyszerűség kedvéért készítettem BASIC-betöltőt is *(2. lista)*, amely a programot a futás címére helyezi és aktivizálja.

Betöltés után a BASIC programot törölhetjük is, az időkijelzőt semmi sem rontja el, kivéve, ha kilépünk a BASIC-ből. A kijelzés vezérlése a következőképpen történik.

Bekapcsolás: SPOKE 255,49134,6

Kikapcsolás: SPOKE 255,49134,0

A 49133-134 a User-Interrupt címe, amelyre ilyen módon írunk 0600H értéket, ahol a programunk elhelyezkedik. (Olvasóink figyelmébe ajánljuk az 1989/9. számban megjelent SPOKE-SPEEK című cikket. A szerk.)

Másik programomat *(3. lista)* a szükségessége miatt készítettem. Feladata: biztosítani az adatforgalmat a BASIC és az ASMON között, mágnesszalagon keresztül. Azoknak készült, akik gépi kódú programokat akarnak BASIC-be építeni és elhelyezni CODE parancsok segítségével. Két üzemmódja van: MAGNÓ ➡ BASIC és BASIC ➡ MAGNÓ. Ezek közül választani kell a futás elején. Lássuk részletesen.

MAGNÓ ➡ BASIC

Ha az ASMON-nal létrehozunk egy kódot, amelyet megfelelő futási címről indítva (kétnyelvű gépen 4827=120 BH) magnóra mentünk az „S" paranccsal, a kód visszaolvasható BASIC-be. A programrész beolvasás után hexadecimálisra alakítja a kódot és CODE utasítások segítségével képernyőre viszi.

A futás végén ezeken kell végigmenni az ENTER-rel, s így épülnek be a programba. Ezután a 100-as sor ALLOCATE értékét a programnak megfelelően lecsökkentjük, majd 5000-től töröljük a programot. Ezzel készen áll a gépi kódú rutinunk, melyhez már csak a BASIC részeket kell hozzáírni.

A program érdekessége a beolvasásban rejlik. Mivel nem tudjuk, hány bájt fog érkezni a magnóról, a beolvasást végtelenített DO-LOOP ciklus végzi. A legutolsó bájt hibát jelezne, ezt azonban egy HANDLER blokk „elkapja", és befejezi a kiírást a zárójelek bezárásával. A program az első CODE blokknak az ELEJE nevet adja, erre a kimentésnél lehet szükség.

BASIC ➡ MAGNÓ

Ezt a részt akkor lehet használni, ha például disassemblálni akarunk egy általunk nem ismert kódot. Ekkor a program betöltése után 110-től írhatjuk a kódokat. Egyetlen kikötés, hogy az első CODE blokknak az ELEJE nevet kell adnunk.

Munkánk végén indulhat a program, a feltett kérdésre elég egy gombot megnyomnunk, ha rendesen dolgoztunk. A vége cím-

1. lista

```
        ; IDŐKIJELZŐ PROGRAM ASSEMBLER LISTÁJA
        ;
        ORG 600H   ; A FORDÍTÁST 4000H MEMÓRIA OFSZETTEL VÉGEZTEM
        ;
        IN A(0B2H)
        LD (PORT),A     ;2-es lap mentése
        LD A,255
        OUT (0B2H),A    ;rendszerlap a 2-es lapra
        LD A,(0BF74H)   ;óra
        LD DE,KIIR
        CALL TOLT
        LD A,(0BF73H)   ;perc
        LD DE,KIIR+3
        CALL TOLT
        LD A,(0BF72H)   ;másodperc
        LD DE,KIIR+6
        CALL TOLT
        LD HL,(0BFF6H)  ;állapotsor eleje
        LD DE,18
        ADD HL,DE       ;kezdő pozició
        EX DE,HL
        LD HL,KIIR
        LD BC,8
        LDIR            ;kiírás az állapotsorba
        LD A,(PORT)
        OUT (0B2H),A    ;eredeti 2-es lap vissza
        RET
        ;
    ;TOLT rutin bemenetei:
    ; A  - kiirandó szám BCD kódban
    ; DE - kiirás pozicioja
        ;
TOLT    LD HL,MUNKA     ;munkaterület a rotációhoz
        LD (HL),A
        LD B,2          ;ciklusváltozó
TOLT2   LD A,0
        RLD             ;rotáció A és (HL) között
        ADD A,30H       ;szám elökészitése kiiráshoz
        EX DE,HL
        LD (HL),A       ;szám töltése a kiirandó mezőbe
        INC HL
        EX DE,HL
        DJNZ TOLT2
        RET
        ;
KIIR    DEFB "00:00:00" ;kiirandó mező
PORT    DEFB 0
MUNKA   DEFB 0
        ;
        END
```

2. lista

```
IDOKIJELZO PROGRAM BASIC BETOLTOJE

  100 ALLOCATE 100
  110 CODE ORA=HEX$("DB,B2,32,55,06,3E,FF,
D3,B2,3A,74,BF,11,4D,06,CD,3A,06,3A,73,BF,
11,50,06,CD,3A,06,3A,72,BF,11,53,06,CD,3A,
06,2A,F6,BF,11,12,00,19,EB,21,4D,06,01,08,
00")
  120 CODE =HEX$("ED,B0,3A,55,06,D3,B2,C9,
21,56,06,77,06,02,3E,00,ED,6F,C6,30,EB,77,
23,EB,10,F4,C9,30,30,3A,30,30,3A,30,30,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00"
)
  130 FOR I=0 TO 90
  140   POKE 1536+I,PEEK(ORA+I)
  150 NEXT I
  160 SPOKE 255,49134,6
```

3. lista

```
ADATKONVERTALO PROGRAM

  100 ALLOCATE 2000
 5000 CODE VEGE=HEX$("0")
 5005 TEXT 40
 5010 PRINT "MAGNO ➡ BASIC : 1"
 5020 PRINT "BASIC ➡ MAGNO : 2"
 5030 INPUT B
 5040 IF B=1 THEN 5150
 5050 IF B<>2 THEN 5030
 5060 PRINT "ELEJE,VEGE KESZ?"
 5070 GET B$
 5080 IF B$="" THEN 5070
 5090 OPEN #10:"TAPE:KOD" ACCESS OUTPUT
 5100 FOR I=ELEJE TO VEGE
 5110   PRINT #10:CHR$(PEEK(I));
 5120 NEXT I
 5130 CLOSE #10
 5140 STOP
 5150 OPEN #10:"TAPE:" ACCESS INPUT
 5160 LET CC=100
 5170 DO
 5180   LET CC=CC+10
 5200   PRINT CC;" ";
 5201   IF CC=110 THEN
 5202     PRINT "CODE ELEJE=HEX$(""";
 5203   ELSE
 5204     PRINT "CODE =HEX$(""";
 5205   END IF
 5210   FOR I=1 TO 50
 5220     GET #10:B$
 5230     CALL KONV
 5240     PRINT B$;
 5245     IF I<50 THEN PRINT ;",";
 5250   NEXT I
 5255   PRINT """)"
 5260 LOOP
 5270 STOP
 5280 DEF KONV
 5285   WHEN EXCEPTION USE HIBA
 5290     LET B=ORD(B$)
 5295   END WHEN
 5300   LET B1=INT(B/16):LET B2=B-16*B1
 5310   LET B1=B1+48:LET B2=B2+48
 5320   IF B1>57 THEN LET B1=B1+7
 5330   IF B2>57 THEN LET B2=B2+7
 5340   LET B$=CHR$(B1)&CHR$(B2)
 5350 END DEF
 5360 HANDLER HIBA
```

mel nem kell foglalkozni, ezt az 5000-es sor tartalmazza.

Ezután a kód ELEJE-től a VEGE-ig magnóra mentődik KOD névvel. Ha más néven akarunk menteni, akkor a választott nevet írjuk be az 5090-es sorba a KOD helyett. Az így létrehozott kódot az ASMON be tudja olvasni az „R" utasítás segítségével, majd kezelni tudja tetszés szerint.

Hámori György

Mankó a gépi kódhoz

A mellékelt programmal nem törekszem babérokra, csak remélem, hogy segítséget nyújtok azoknak az Enterprise-tulajdonosoknak, akik most kezdenek a BASIC nyelv után gépi kódú programozással foglalkozni. A programmal a különböző számrendszerek közötti átváltást lehet elvégezni és kinyomtatni. Ez kezdőknek komoly segítséget jelenthet.

A program felépítése:

101—114	Alapbeállítások
115—135	Menürutin
500	Bináris átalakító
508—516	Kiíratórutinok
517	HEX-átalakító
526	Billentyűbeolvasás
534	Képernyőtörlés, -nyomtatás
543	Nyomtatórutin
549	Bináris beolvasás
562	HEX-beolvasás
576	Decimális beolvasás
590	Üzenőrutin

Borsos József

Minden kedden 17-től 20 óráig
HCC ENTERPRISE klub
a VSZM
Közösségi Házban
(Bp. XI., Fehérvári út 120.)
Klubvezető: Romvári Gábor
Telefon: 1810-950/473

```
100 PROGRAM BIN-HEX-DEC CONVERTER BY B'WARE
101 DATA 98,121,32,32,66,39,87,65,82,69,32
102 STRING A$*16,W$*1,H$*4
103 NUMERIC W,DEC,B,H,I,Z,S
104 TEXT 80
105 SET VIDEO MODE 1:SET VIDEO COLOR 0:SET VIDEO X 42:SET VIDEO Y 3
106 OPEN #220:"video:"
107 SET #220:PALETTE 0,91:DISPLAY #220:AT 2 FROM 1 TO 3
108 PRINT #220,AT 2,11:"16 BIT BINARY";TAB(42);"HEXA";TAB(68);"DEC"
109 PRINT #220,AT 3,14:"( H L )"
110 SET VIDEO MODE 1:SET VIDEO COLOR 0:SET VIDEO X 22:SET VIDEO Y 2
111 OPEN #250:"video:"
112 DISPLAY #250:AT 26 FROM 1 TO 1
113 SET #102:PALETTE 91,0:SET CURSOR COLOR 0
114 LET S=6
115 DO
116   PRINT AT 6,7:"                    ";TAB(40);"    ";TAB(65);"      "
117   LET S=S+1:CALL SP
118   IF S>22 THEN CALL P
119   CLEAR #250
120   SET #250:PALETTE 10,255
121   PRINT #250,AT 1,3:"B BIN       H HEXA       D DEC       C COPY"
122   CALL KEY
123   SELECT W
124   CASE 66,98
125     CALL BIN0
126   CASE 72,104
127     CALL HEX0
128   CASE 68,100
129     CALL DEC0
130   CASE 67,99
131     CALL C
132   CASE ELSE
133     LET S=S-1
134   END SELECT
135 LOOP
500 DEF BINH
501   LET B=DEC:LET Z=32768:LET A$=""
502   IF B/Z>=1 THEN LET A$=A$&"1":LET B=B-Z:GOTO 504
503   LET A$=A$&"0"
504   LET Z=Z/2
505   IF Z>=1 THEN GOTO 502
506   CALL PRB
507 END DEF
508 DEF PRH
509   PRINT AT S,40:H$
510 END DEF
511 DEF PRD
512   PRINT USING "%%%%#",AT S,65:DEC
513 END DEF
514 DEF PRB
515   PRINT AT S,7:A$(:8);" ";A$(9:)
516 END DEF
517 DEF HEXA
518   LET B=DEC:LET H=4096:LET H$=""
519   LET I=INT(B/H+48)
520   IF I>57 THEN LET I=I+7
521   LET H$=H$&CHR$(I)
522   LET B=B-INT(B/H)*H:LET H=H/16
523   IF H>=1 THEN GOTO 519
524   CALL PRH
525 END DEF
526 DEF KEY
527   LET W$=INKEY$
528   IF W$="" THEN
529     SOUND PITCH 100,DURATION 1
530     GOTO 527
531   END IF
532   LET W=ORD(W$)
533 END DEF
534 DEF P
535   CLEAR #250
536   SET #250:PALETTE 255,10
537   PRINT #250,AT 1,3:"ANY KEY  =>  clear the screen      C COPY"
538   CALL KEY
539   IF W=99 OR W=67 THEN CALL C
540   CLEAR #102
541   LET S=7
542 END DEF
543 DEF C
544   CLEAR #250
545   SET #250:PALETTE 4,255:PRINT #250,AT 1,20:"COPY"
546   COPY
547   LET S=S-1
```

```
548 END DEF
549 DEF BIN0
550   CLEAR #250
551   SET #250:PALETTE 1,255
552   PRINT #250,AT 1,3:"16 BIT BINARY => HEXADECIMAL => DECIMAL"
553   LET A$="":PRINT AT 6,7:"!!!!!!!! !!!!!!!!"
554   LET DEC=0:LET Z=32768
555   FOR X=1 TO 16
556     CALL KEY
557     IF W$<>"1" AND W$<>"0" THEN GOTO 556
558     LET DEC=DEC+(W-48)*Z:LET Z=Z/2:LET A$=A$&W$:CALL PRB
559   NEXT
560   CALL HEXA:CALL PRD
561 END DEF
562 DEF HEX0
563   CLEAR #250
564   SET #250:PALETTE 5,255
565   PRINT #250,AT 1,3:"HEXADECIMAL => DECIMAL => 16 BIT BINARY"
566   PRINT AT 6,40:"!!!!":LET H$="":LET DEC=0
567   FOR X=3 TO 0 STEP-1
568     CALL KEY
569     IF W$<"0" OR W$>"f" OR(W$>"9" AND W$<"a") THEN GOTO 568
570     IF W$>="a" THEN LET DEC=DEC+(W-87)*16^X:GOTO 572
571     LET DEC=DEC+(W-48)*16^X
572     LET H$=H$&UCASE$(W$):CALL PRH
573   NEXT X
574   CALL BINH:CALL PRD
575 END DEF
576 DEF DEC0
577   CLEAR #250
578   SET #250:PALETTE 9,255
579   PRINT #250,AT 1,3:"DECIMAL => HEXADECIMAL => 16 BIT BINARY "
580   PRINT AT 6,65:"!!!!!":LET DEC=0
581   FOR X=4 TO 0 STEP-1
582     CALL KEY
583     IF W=32 THEN LET W$="0"
584     IF W$<"0" OR W$>"9" THEN GOTO 582
585     LET DEC=DEC+VAL(W$)*10^X:PRINT AT S,69-X:W$
586   NEXT X
587   IF DEC>65535 THEN PRINT AT S,65:"ÄÄÄÄÄ":GOTO 577
588   CALL BINH:CALL HEXA
589 END DEF
590 DEF SP
591   RESTORE
592   FOR X=0 TO 10
593     READ B
594     SPOKE 255,16075+X,B
595   NEXT
596 END DEF
```

Mi a manó?

Akinek nem telik sztereo erősítő berendezésre vagy költséges hangmodulátorra, próbálja ki a walkmanhez is használható fejhallgatót. A kazettára mentéshez is használható OUTPUT-hoz kell csatlakoztatni. Így a lehető legteljesebben érvényesül a számítógép sztereo hatása.

●

A kecskeméti Szalvay Mihály Úttörő- és Ifjúsági Otthonban működő Enterprise Klub (Kecskemét, Széchenyi u. 7.) jelentkezett szerkesztőségünknél. Jelenlegi taglétszámuk 30 fő. Várják a további jelentkezőket — mi pedig híreket a további EP kluboktól.

●

Katona László olvasónk küldött a „Mi a manó?" részére Enterprise gépre vonatkozó érdekességeket. Ezek egyike: ha a szövegszerkesztő üzemmódban megtelik a memória, akkor a gép figyelmeztetés nélkül törli a szöveg elejét vagy a végét, attól függően, melyik van éppen távolabb a kurzortól.

●

A program „szöveg" utasítása után zárójelben felsorolt változók nulla, illetve üres karakter értéket kapnak. Az idézőjelek között üres karakter is lehet. Az értékadás csak akkor működik, ha ez a program első sora.

Következő számunkban folytatjuk Katona László észrevételeinek ismertetését, és további tanácsokat, ötleteket várunk a „Mi a manó?"-ba.

Gaetsch Günterné rajza

Sorba rendezés

A Magazin 1989/10. számában a Feladatok — megoldások sorozatban Pintér Gábor a sorba rendezéssel foglalkozott. Kóta Béla olvasónk az Enterprise-osok részére is készített IS-BASIC-ben egy hatékony rendezőprogramot. Az algoritmus a vastagon nyomtatott programrészben található.

```
100 PROGRAM "tape-2:rendez"
110 ! IS-BASIC. Kóta Béla. 1989.
120 INPUT PROMPT "kezdö sorszám =":
    SK
130 INPUT PROMPT "vég   sorszám =":
    SV
140 NUMERIC VALAMI(SK TO SV)
150 !-értékadás-
160 RANDOMIZE
170 PRINT
180 FOR I=SK TO SV
190   LET VALAMI(I)=RND(10000)
200 NEXT I
210 CALL KIIRO:CALL RENDEZ(VALAMI):
    CALL KIIRO
220 END
230 !
240 DEF KIIRO
250   PRINT
260   FOR I=SK TO SV
270     PRINT I,VALAMI(I)
280   NEXT I
290 END DEF
300 !
310 DEF RENDEZ(REF X)
320   NUMERIC I,J,K,C
330   LET N1=LBOUND(X):LET N2=
      UBOUND(X)
340   FOR I=N1 TO N2
350     LET K=I:LET C=X(I)
360     FOR J=I+1 TO N2
370       IF X(J)<C THEN LET C=X(J):
          LET K=J
380     NEXT J
390     LET X(K)=X(I):LET X(I)=C
400   NEXT I
410 END DEF
```

Átszámozunk

Az 1989. áprilisi Magazinban olvashattunk az Enterprise BASIC programok átsorszámozásáról. Ennek tanulmányozására szolgál egy relokálható BASIC-utasításbővítő.

A lista szerinti BASIC-töltő program készíti el szalagra az utasításbővítőt. A modult betöltve a BASIC felismer két parancsmódú új utasítást. Az NLIST hatására a LIST, DELETE, RENUMBER parancsokkal programunk nem érhető el. Más eset, ha egy blokk nevére hivatkozunk. Az így kimentett program később is megtartja e tulajdonságait.

A YLIST a helyreállító parancs.

A modul betölthető a BASIC program tárba vitele előtt és utána is. A meleg reset hatására nem törlődik, ezért feleslegesen ne töltsük be többször.

Haluska László

```
100 PROGRAM "NO_LIST_BASICTOLTO"
110 OPEN #1:"NO_LIST" ACCESS OUTPUT
120 LET C=-9464
130 READ IF MISSING 170:A$
140 LET C$=HEX$(A$(:2)):LET B$=HEX$(A$(3:))
150 LET C=MOD(C+ORD(B$)+256*ORD(C$)+1,65536)
160 PRINT #1:B$;C$;:GOTO 130
170 FOR X=1 TO 13
180 PRINT #1:C$;
190 NEXT
200 CLOSE #1
210 IF C THEN PRINT "HIBA TORTENT A DATABEOLVASASNAL!"
220 DATA 0200,00A8,0000,0000,0000
230 DATA 0000,0000,0000,9676,40C6
240 DATA 5927,4678,3D56,C42C,800F
250 DATA 060C,3800,CE0A,902C,0000
260 DATA 0105,0080,3810,0100,8005
270 DATA A727,44F9,4E0A,1226,654A
280 DATA 2048,3A01,CA7F,5220,30C9
290 DATA 5392,002A,0558,C0D0,020C
300 DATA 0064,453F,A041,F413,00FC
310 DATA B25B,A51F,0BE0,ACF4,7E02
320 DATA 409C,D32F,0980,EDB2,862D
330 DATA 4680,5823,0804,0400,C9BA
340 DATA C0C0,A068,0946,9E75,C05A
350 DATA 7862,44F9,58ED,1D88,570F
360 DATA 652B,7EC2,885D,0201,F233
370 DATA 40E8,3803,CE0A,972D,AA8B
380 DATA FF74,1A86,902D,5818,FD00
390 DATA 002F,A11F,C290,90C0,3661
400 DATA 4E1D,B366,8008,9598,9E39
410 DATA 868D,00F1,72C4,1C38,868B
420 DATA 4103,C470,2E6C,00C7,000A
```

A Devil's Lair németesítése

A korai fejlesztésű Enterprise játékprogramok egy részére jellemző, hogy nem futnak kétnyelvű gépeken (például a Snooker), vagy ha futnak is, akkor hibásan (mint a Devil's Lair), illetve némelyik csak angol módban (Colossal Game, Devil's Lair). Ennek az az oka, hogy a fejlesztők túlzottan kihasználták az angol gépek sajátosságait.

A Devil's Lairnél a gépi kódban megnyitandó videolap memóriacímét vették „fixnek"; nem az EXOS-on keresztül érték el, hanem empirikusan megállapított értéket POKE-oltak be a BASIC-ből. A megoldás az, hogy megnyitjuk a kérdéses videolapot, az EXOS-szal lekérdezzük a címét, és ezt írjuk be a két memóriabájtba. Az új BASIC-betöltő elvégzi még az angol módba állást is, hiszen a fentebb említett POKE-ok egy német módban aktív rutint tesznek tönkre. A csak angol módban futó programoknál az angol módba állást SET 144,255 utasítással végezzük el EXT " UK" helyett, mivel az előbbi is angol nyelvű üzeneteket eredményez, de anélkül, hogy a billentyűzetet és a karakterképeket átdefiniálná.

Ha tehát a Devil' s Lair eredeti BASIC-betöltőjét kicseréljük a *listán* láthatóval, mind angol, mind kétnyelvű EP-okon hibátlanul futó programot kapunk.

Racskó Tamás

```
100 PROGRAM "DEVIL'S_LAIR"
110 ALLOCATE 1000
120 WHEN EXCEPTION USE ENGEP
130   SET 144,255
140 END WHEN
150 SET KEY CLICK OFF
160 SET SPEAKER OFF
170 CLEAR SCREEN
180 PRINT AT 10,12:"DEVIL'S LAIR"
190 PRINT AT 14,6:"COPYRIGHT
LORICIELS MAY 1985"
200 CODE XXXX=HEX$("00,00,00,00,00,
00,00,00,00,00,00,00,00,00")
210 CODE LOAD=HEX$("00,3E,FD,D3,B3,
3E,FE,D3,B1,3E,6A,11")
220 CODE AAAA=WORD$(LOAD)&HEX$("F7,
01,11,00,C0,01,00,2E,3E,6A,F7,06,3E,6A,
F7,03,C3,2A,C1")
230 CODE VADDR=HEX$("3E,01,06,03,F7,
0B,60,69,C9")
240 SET VIDEO MODE 1
250 SET VIDEO COLOR 1
260 SET VIDEO X 40
270 SET VIDEO Y 27
280 OPEN #1:"VIDEO:"
290 LET ES=USR(VADDR,0)
300 CLOSE #1
310 POKE 4810,MOD(ES,256)
320 POKE 4811,(INT(ES/256) BAND 31
BOR 64)
330 CALL USR(LOAD,0)
340 HANDLER ENGEP
350   CONTINUE
360 END HANDLER
370 END
```

Csipkés RND

Az alábbi kis program az RND (végtelenszám-generáló függvény) segítségével „csipkét" rajzol a képernyőre. A program a szinusz, koszinusz és a PI (3,14...) segítségével felrajzol egy pontokból álló kört. A FOR-NEXT cikluson belül definiált x és y értékek csak abban térnek el az xx és yy értékektől, hogy az e értékkel, azaz e * PI/lépésközzel előbbre halad a „körpályán". A PLOT x, y; xx, yy egy egyenest rajzol, és a csipke így alakul ki, a vonal elfordulásával.

Érdemes a programon belül az értékeket megváltoztatni, kísérletezni, mert így a program megértése könnyebbé válik és új, talán érdekesebb csipkék is kialakulhatnak. Például x = a + b * cos i esetében, ahol a = a kör középpontja, b = a kör sugara.

Az x-en és az xx-en belül a b-t, azaz a sugarat azonos mértékben megváltoztatva, ellipszist kapunk.

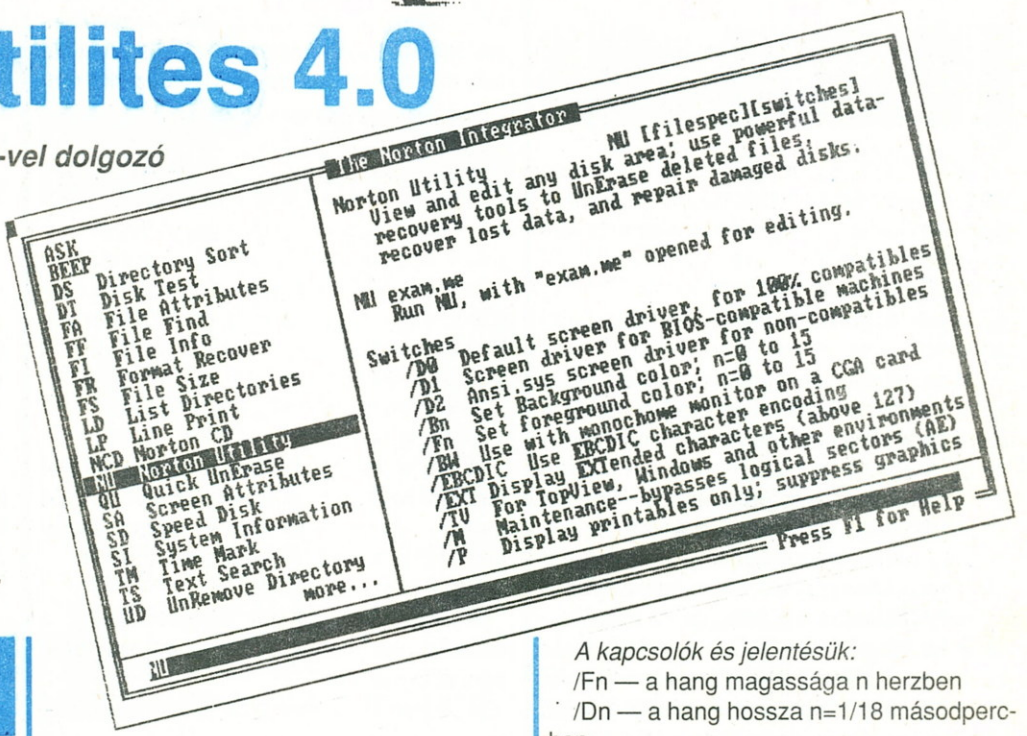
Papp Miklós

```
10 GRAPHICS HIRES 4
17 SET INK WHITE
20 LET SZ=INT(10+RND*40)
30 LET E=INT(20+RND*35)
70 FOR I=0 TO PI*2 STEP PI/SZ
80   LET X=600+350*COS(I)
81   LET XX=600+350*COS(I+E*PI/SZ)
90   LET Y=350+350*SIN(I)
91   LET YY=350+350*SIN(I+E*PI/SZ)
110  PLOT X,Y;XX,YY
120 NEXT I
130 GOTO 20
```

PROGRAM ISMERTETŐ

Norton Utilites 4.0

E program birtokában az IBM PC-vel dolgozó felhasználó olyan eszközre tesz szert, amely munkáját nagymértékben megkönnyíti. A programcsomagban található parancsfájlok, programok segítségével a törölt fájlok visszaállításától kezdve a hanggenerálásig nagyon sokféle funkció ellátása lehetséges. A következőkben részletesen ismertetjük a programcsomag egyes elemeinek használatát.



ASK.EXE

Interaktív (párbeszédes) batch-fájlok létrehozásánál alkalmazható. Megadási módja: ASK „üzenet", [billentyűlista]. A segítségével létrehozott batch-fájl működése a következő. Megjeleníti az üzenetet a képernyőn, majd erre az üzenetre választ vár. Ha a válaszként lenyomott billentyű szerepel a billentyűlistában, akkor az ASK átadja a vezérlést a batch-fájl megfelelő helyére, a billentyű ERRORLEVEL kódjának segítségével. Az ERRORLEVEL szintek a következőképpen alakulnak: az elsőnek megadott billentyű szintje 1, a másodikként megadotté 2, és így tovább. A parancsok végrehajtása az ERRORLEVEL értékétől függő ágra jut úgy, hogy a legmagasabb szintű hajtódik először végre, majd az ezt követő szintre jut a vezérlés, legutoljára az ERRORLEVEL 1 szint hajtódik végre. További magyarázat helyett nézzünk egy példát!

A fájl neve legyen MINTA.BAT. Tartalma:

```
echo off
cls
ask „Biztos be akar lépni?", ni
if errorlevel 2 goto run
if errorlevel 1 goto quit
:run
cd user
goto ki
:quit
echo Ha nem, hát nem!
:ki
```

Nézzük a működést! Meghíva a MINTA-t, a következő történik. A képernyőn a parancsok nem fognak megjelenni (echo off), képernyőtörlés (cls) után megjelenik az üzenet. Erre az üzenetre az „n", illetve az „i" gomb lenyomásával lehet válaszolni. Bármilyen más billentyű lenyomásakor sípolással figyelmeztet, hogy a billentyű nincs a megengedett válaszok között. Ha „i"-t nyomunk, akkor a user nevű alkönyvtárat megnyitja, majd befejezi a program futását. Ha „n"-et nyomunk, akkor kiírja az echo utáni szöveget, majd szintén befejeződik a program futása.

BEEP.EXE

Megadási módja: BEEP [kapcsolók] vagy BEEP [fájlnév].

A program lejátssza a kapcsolók segítségével megadott hangot vagy hangok sorozatát. A hangokat meg lehet adni egy külön fájlban (ezt hívjuk hangfájlnak) vagy egy parancssorban. Egy példa: BEEP /F100 /D36 / W18 /R3. Ezt a parancssort begépelve a következő fog történni. Egy hangsorozatot hallunk, melynek a jellemzői: a hang magassága 100 Hz (/F100), ideje 2 másodperc (/D36), 1 másodperc szünet (/W18), majd ezt háromszor még megismétli (/R3).

BEEP hangfájl

Ha így adjuk meg, lejátssza a hangfájlban megadott hangok sorozatát. A hangok listáját ugyanúgy kell megadni, mint a parancssorban. Ha csak a BEEP-et adjuk meg, egy kb. 1/2 másodperces, 800 Hz-es hangot fogunk hallani.

A kapcsolók és jelentésük:

/Fn — a hang magassága n herzben

/Dn — a hang hossza n=1/18 másodpercben

/Rn — a hang ismétlése n-szer

/Wn — hang(ok) közötti szünet megadása n=1/18 másodpercben

DS.EXE

(Directory Sort)

Segítségével a lemez vagy az alkönyvtár katalógusában szereplő állományokat rendezhetjük több szempont szerint.

Megadási módja: DS [alkönyvtárnév] vagy DS rendezési kulcs(ok) [alkönyvtárnév] [kapcsoló].

A rendezés lehetséges egy vagy több alkönyvtárban név (N), kiterjesztés (E), létrehozási idő (T), dátum (D), nagyság (S) szerint vagy ezek kombinációjaként.

A programot futtathatjuk interaktív vagy full screen editor módban. Interaktív lesz a futás, ha a rendezési kulcsokat a parancssorba írjuk. Ellenkező esetben full screen módban jelenik meg a program. Ha csak a DS-t adjuk meg, akkor az aktuális alkönyvtár jelenik meg full screen módban a képernyőn, ha megadjuk az alkönyvtár nevét is, akkor a kért katalógust szerkeszthetjük.

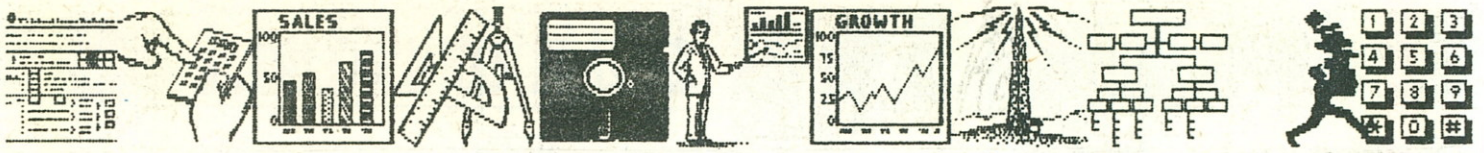
Példák:

DS NE \util \norton /S

Az így megadott parancs rendezi a \util\norton elérésű alkönyvtárat fájlnév (N) és kiterjesztés (E) szerint. Az /S-sel megadott rendezési elv szerint a subdirectoryt is rendezi.

DS D—T—

Rendezi az alkönyvtárat úgy, hogy a rende-



zést a legújabb állományokon kezdi. Tehát a hozzánk legközelebbi időpontban létrehozott állomány lesz a listában elöl.

Rendezési kulcsok:

N — név; D — dátum; E — kiterjesztés; S — nagyság; T — idő. Ha a megadott kulcs után "—"-t teszünk, akkor a kulcs szerinti rendezés az alapértelmezett rendezés ellentéte lesz.

Alapértelmezés: névnél, kiterjesztésnél az angol ábécében elöl álló, nagyságnál a nagyobb, időnél a régebbi dátumú állomány lesz az első.

Kapcsoló és jelentése:

/S — a rendezés alkönyvtárra is vonatkozzon.

Ha az előzőekben leírtaknak megfelelően a full screen editor módot választjuk, akkor a parancs kiadása után egy képernyő jelenik meg.

A felső sorban a program neve, a meghajtó azonosítója, az alkönyvtár neve látható. Ez alatt az alkönyvtárban található az állományok neve, kiterjesztése, nagysága, az utolsó változtatásuk dátuma és ideje. A képernyő jobb oldalán a "Sort by" felirat alatt a rendezési kulcs, az "Order" felirat alatt a sorrend beállítása lesz a kiválasztás után látható. Ez alatt vannak a rendezési kulcsok és a rájuk vonatkozó parancsok. A képernyő legalsó sorában a rendezési kulcsok szerinti rendezésre, egyes állományok mozgatására, a könyvtár rendeltetésének felcserélésére és a directory lemezre írására vonatkozó parancsok kaptak helyet.

A rendezési kulcsok megadása

A program bejelentkezése után, ha a Tab billentyűt leütjük, a rendezési kulcsot vagy kulcsokat a megfelelő nagybetű leütésével adhatjuk meg. Ennek hatására a "Sort by" felirat alatt a rendezési kulcsot, az "Order" alatt a rendezési sorrendet fogjuk látni. A rendezési sorrend alatt " " jelenik meg. A " " az alapértelmezés szerinti rendezést jelenti, ami az alábbiaknak felel meg:

— Névnél a betűrend szerinti rendezést jelent, az angol ábécének megfelelően.

— Kiterjesztésnél a kiterjesztések ábécé szerinti rendezését jelenti.

— Az idő és dátum szerinti rendezésnél a régebbi állományok kerülnek előre.

— Nagyság szerinti rendezéskor csökkenő nagyság szerint lehet rendezni.

Ha a kulcs megadása után a "—" jelet adjuk, akkor az alapértelmezésbeli rendezéssel ellentétesen fog rendezni a program.

A "Clear sort order" parancsot a "C" leütésével adhatjuk ki, ami törli a beadott kulcsokat.

A "Movesort entry" parancsot az "M" billentyű leütésével aktivizálhatjuk. Hatására a képernyő bal felére térhetünk vissza. Ugyanezt a Tab-bal is elérhetjük. Ha valamelyik megadott kulcsot törölni szeretnénk, akkor álljunk a kurzormozgató billentyűk segítségével a törlendő kulcsra, majd nyomjuk meg az "Esc" billentyűt.

Re-sort:

Ha megadtuk a rendezési kulcsot, kulcsokat, az "R" billentyű lenyomásával aktivizálódik a parancs, aminek hatására a könyvtár tartalomjegyzékét a megadottak szerint rendezi.

Move file(s):

Ennek a parancsnak a segítségével egyes állományokat mozgathatunk oda, ahova akarjuk. Az "M" gomb lenyomására aktivizálható. Használata a következő. Ráállunk a mozgatni kívánt fájl nevére, kurzormozgatók segítségével, majd megnyomjuk a szóközbillentyűt. Ezzek kijelöltük mozgatásra a fájlt. A kurzormozgatók segítségével oda visszük a fájl nevét, ahová szeretnénk, majd Entert adunk.

Change sort order:

A már rendezett, illetve rendezetlen könyvtár alatti bejegyzések ennek a parancsnak a hatására fordított sorrendben lesznek láthatók. Aktivizálása a "C" leütésével történik.

Write changes to disk:

A képernyőn látható fájl-elrendezést a lemezre rögzíthetjük. A "W" gomb lenyomására aktivizálódik.

Az editorból az "Esc" billentyűvel léphetünk ki.

DT.EXE

(Disk Test)

A programmal a lemezt vagy az egyes fájlokat ellenőrizhetjük, fizikai hibák szempontjából. A hibás clusterokon lévő adatokat átmozgatja a kijelölt clusterokra.

Megadási módja: DT [d:] [fájlnév] [kapcsolók] DT

Így megadva az aktuális lemezt ellenőrzi interaktív módon. DT ELLEN.COM: így az ELLEN. COM fájlt ellenőrzi, és a kérdéses clustereket áthelyezi.

Kapcsolók és jelentésük:

/B — a lemezt és fájlokat is ellenőrzi

/Cn — az n-edik cluster "rossz" jelzéssel való megjelölése

/Cn — az n-edik cluster "rossz" jelzésének törlése

/D — teljes lemez vizsgálata

/F — csak a fájlokat ellenőrzi

/LOG — a teszt eredménye printerre vagy LOG kiterjesztésű fájlba kerül

/M — a kérdéses clusterokat biztonságos helyre teszi

/S — az alkönyvtárakat is ellenőrzi

FA.EXE

(File Attribute)

E programmal a fájl-attribútumokat a képernyőre listázhatjuk, beállíthatjuk, törölhetjük.

Megadási módja: FA [fájlnév] [opciók] [kapcsolók]

Például: FA /r/ hid+

Ha nincs megadva fájlnév, akkor az opciók és kapcsolók hatása az összes fájlra vonatkozik. Ebben a példában először az összes olyan fájlt megkeresi, amelynek "csak olvasható" (read only) attribútuma van, majd ezeket a fájlokat rejtett (hidden) fájlokká alakítja.

Opciók és jelentésük.

/A — archív

/R — read only (csak olvasható)

/HID — hidden

/SYS — system (rendszer)

Ha az opciók után + jel van, akkor beállítódik az attribútum, ha — jel van, akkor az opció által meghatározott attribútum törlésre kerül.

Kapcsolók és jelentésük:

/CLEAR — az összes attribútumot törli

/P — képernyőnkénti listázás beállítása, a lista további része egy billentyű lenyomására kerül a képernyőre

/S — az alkönyvtárban lévő fájlokon is beállítja, illetve törli az attribútumokat

/U — azok a fájlok kerülnek listázásra, amelyeknek az attribútuma megváltozott

/T — van-e olyan fájl, amelynek változott az attribútuma?

FF.EXE

(File Find)

A lemez összes alkönyvtárában megkeresi a megadott állományt, és megadja az elérési útját. A parancs képes a rejtett (hidden) fájlok helyének kijelzésére is.

Megadási módja: FF [d:] [fájlnév] [kapcsolók]

Ha a fájlnév után nem adjuk meg a kiterjesztést, akkor az összes olyan állomány elérési útját megadja, amelynek a megadott név a neve.

Kapcsolók és jelentésük:

/A — a keresés az egész lemezre kiterjed

/B — a képernyőnkénti listázás

/W — megegyezik a DOS /W opciójával

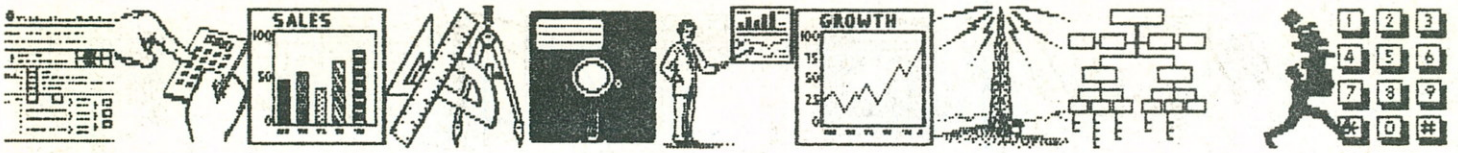
FI. EXE

(File Info)

Ez a program lehetővé teszi, hogy a fájljainkhoz különböző, számunkra fontos megjegyzéseket fűzzünk, ezeket a megjegyzéseket megváltoztassuk, illetve megnézhessük. A megjegyzéseket az FI.EXE könyvtáranként egy állományba teszi, ennek az állománynak a neve FILEINFO.FI.

Megadási módja: FI [fájlnév] [megjegyzés] [kapcsolók]

Ha csak az FI parancsot adjuk ki, akkor a könyvtárban lévő fájlokat és az ezekhez fű-



zött megjegyzéseket listázza a parancs. A megjegyzések fájlokhoz fűzésének két módja van:

/FI fájlnév megjegyzés

/FI fájlnév /E

Így megadva a parancsot "full screen" módon fűzhetjük hozzá, illetve módosíthatjuk a megjegyzéseket az állományokhoz. A megjegyzés maximum 65 karakter hosszú lehet.

Kapcsolók és jelentésük:

/C — csak azok az állományok kerülnek listázásra, amelyekhez van megjegyzés

/D — megjegyzés törlése

/E — „full screen" mód beállítása

/L — a teljes megjegyzés megjelenítése, ugyanis alapértelmezésben a megjegyzés első 35 karakterét írja ki a parancs

/N — az /E kapcsolóval együtt használatos akkor, ha nem 100 százalékig IBM-kompatibilis gépünk van

/P — a képernyőnkénti listázás beállítása

/PACK — sűríti a FILEINFO.FI állományt (opcionális)

/S — listázáskor az aktuális könyvtárból nyíló alkönyvtárnak a fájljait is listázza

FR.EXE

(Format Recover)

A programmal visszaállíthatjuk a véletlenül formattált winchester tartalmát. Ez a program két részből áll. Először a winchesterből elmenti azokat az információkat, amelyek feltétlenül szükségesek a visszaállításhoz (/SAVE), majd újraformáz a mentett információk segítségével. Ahhoz, hogy a program vissza tudja állítani a winchester tartalmát, rendszeresen a SAVE opcióval kell futtatni. A SAVE hatására a lemezen egy FRECOVER.DAT nevű fájl kerül a lemezre, aminek a nagysága 70 százalékig telített lemezen 64 kbájt körül van. Ezért ajánlott az AUTOEXEC.BAT feladatai közé ennek a programnak a lefuttatását is beiktatni a SAVE opcióval.

Megadási módja: FR [d:] [/SAVE]

Kapcsoló és jelentése:

/SAVE — elmenti a szükséges adatokat a lemez visszaállításához.

FS.EXE

(File Size)

A program segítségével a képernyőre listázhatjuk a megadott könyvtárban vagy lemezen lévő állományok nagyságát, valamint a listázott fájlok által elfoglalt terület nagyságát bájtokban, illetve a használható lemezterület százalékában, valamint az aktuális meghajtóban lévő lemez összkapacitását.

Megadási módja: FS [fájlnév] [meghajtó:] [kapcsolók]

Kapcsolók és jelentésük:

/P — képernyőnkénti listázás

/S — az alkönyvtár tartalmát is listázza

/T — csak az összesített információk jelennek meg.

LD.EXE

(List Directories)

A lemezen lévő könyvtárakról szöveges vagy grafikus listát jelenít meg. Elsősorban akkor használhatjuk ezt a programot, ha ki szeretnénk nyomtatni ezt a struktúrát vagy egy fájlt szeretnénk elhelyezni valamelyik könyvtárba. A grafikus fastruktúra megjelenítéséhez, valamint könyvtárváltáshoz az NCD ide vonatkozó funkcióját használja ez a program.

Grafikus könyvtárstruktúra nyomtatón való megjelenítése: LD/G> prn

Megadási módja: LD [d:] [elérési út] [kapcsolók]

Kapcsolók és jelentésük:

/A — a lemezen lévő összes könyvtár listája

/G — grafikus megjelenítés

/N /G — együtt, akkor használjuk, ha nem IBM-kompatibilis printerünk van

/P — képernyőnkénti listázás beállítása

/T — összesített információk (csak a fájlok száma és nagysága).

LP. EXE

(Line Print)

Segítségével szöveges állományokat nyomtathatunk az általunk megadott formátumra vagy egy másik állományba.

Megadási Módja. LP fájlnév [cél fájlnév] [kapcsolók]

Kapcsolók és jelentésük:

Kapcsoló	Jelentés	Alapértelmezés
/N	sor számozása	kikapcsolt
/Tn	felső margó n	3 sor
/Bn	alsó margó n	5 sor
/Ln	bal margó n	5 karakter
/Rn	jobb margó n	5 karakter
/Mn	sorok száma n	66 sor
/Wn	lap szélessége n	85 inch
/Pn	lapsorszámozás kezdete n	1
/Sn	soremelés	1
/80	80 inches nyomtató	bekapcsolt
/132	sűrített nyomtatás	kikapcsolt
/EXT	kibővített karakterkészlet	kikapcsolt
/MEADERn	nyomtatófej szintje n	1
/EBCDIC	EBCDIC kód használata	kikapcsolt

Az /N, /80, /132, /EXT és a /EBCDIC kapcsolók alapértelmezéstől különböző állítása a parancssorban való megadásukkal történik.

/SET: fájlnév — Lotus-fájl megadása

NCD.EXE

(Norton Change Directory)

Ez a program, három, könyvtárakkal kapcsolatos műveletet egyesít, nevezetesen a könyvtárváltást (CD), új könyvtár létrehozását (MD), valamint könyvtár törlését. Ezeket a funkciókat kétféleképpen adhatjuk meg: vagy parancssorban, vagy full screen editor módban.

Megadási módja: NCD [könyvtárnév, vagy a könyvtárnév első néhány karaktere] [kapcsoló], vagy NCD MD könyvtár, vagy NCD RD könyvtár. Az utolsó két parancs új könyvtárat hoz létre, illetve könyvtárat töröl.

Példa a könyvtárváltásra:

NDC nor

Ezzel a paranccsal bármelyik könyvtárban is vagyunk, a norton nevű alkönyvtárba kerülünk. Ha több könyvtárnév kezdő karakterei megegyeznek, akkor a kívánt alkönyvtár nevét végig meg kell adni.

Kapcsoló:

/R — újraolvassa a lemez könyvtárstruktúráját.

Erre azért van szükség, mert a program egy fájlban, a TREEINFO.NCD fájlban tárolja a lemez könyvtárstruktúráját. Ha ezt az opciót megadjuk, akkor felfrissíti a fájlt.

Full screen editor mód megadása az NCD begépelésével történik. A képernyőn látványosan jelenik meg a lemez könyvtárstruktúrája. Az éppen használt könyvtár neve inverzben látszik. A képernyő alsó részén jelenik meg a használt könyvtár neve az elérési úttal, a legalsó sorban a kiadható parancsok kaptak helyet.

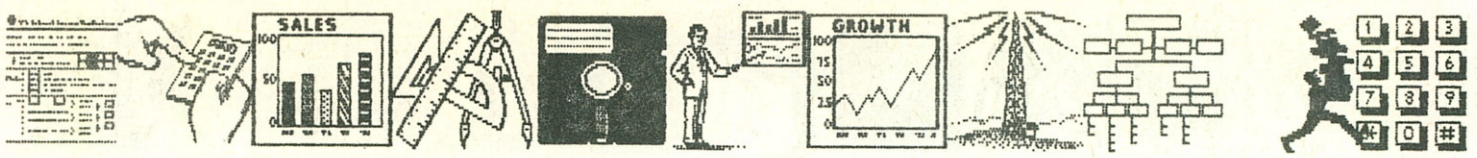
Könyvtárváltás: a kurzormozgató gombokkal a használni kívánt könyvtár nevére állunk, majd entert adunk.

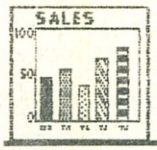
Új könyvtár nyitása (Make directory): a kurzormozgató gombokkal arra az alkönyvtárra állunk, amelyből egy új könyvtárat szeretnénk létrehozni, majd lenyomjuk az "M"-et. Ezután beírhatjuk a könyvtár nevét.

Könyvtár törlése (Remove directory): a könyvtár kiválasztásánál ugyanúgy járunk el, mint az előzőeknél, de az "R"-et ütjük le.

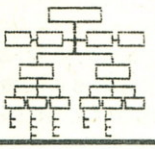
Visszaállás arra a könyvtárra, ahonnan a programot hívtuk (Original directory): az "O" lenyomására az intenzíven látható könyvtár nevére állhatunk.

E program csak azért hasznos, mert látványosan mutatja meg a lemez könyvtárstruktúráját. A programmal megvalósítható funkciók a DOS-ban is megtalálhatók, a program semmivel nem egyszerűsíti le ezeknek a DOS-funkcióknak a használatát. **K.L.**





VEGYES



A PC-tervezés dilemmái

Merre visz az út?

Új architektúrák

Az asztali PC-k nagyívű fejlődése mintha lelassult volna: többé nem elegendő a csupán nagygépes elveket kölcsönözni a fejlődéshez, de a nagygép típusú problémák és megoldásuk egy további fejlődés előhírnökei. Valószínűleg e problémák megoldása nyomán a személyi szuperszámítógépek új és izgalmas korszaka következik majd.

A 8 bites rendszerek tervezése egyszerű volt: nem jelentett nagy gondot egy tucatnyi IC összekötése. Ez még a 8086-os processzorral készült 16 bites rendszerekre is igaz. A 386-os processzorbázison alapuló rendszereket azonban már nem lehet egy sajtcédulán megtervezni, különösen nem, ha az 25 MHz-es frekvencián működik. A sebesség növekedésével a PC-tervezőknek ugyanazokat a problémákat kell megoldaniuk, amelyeket korábban a nagyobb gépek tervezőinek.

Miután a számítógépek teljesítményét első közelítésben az utasításvégrehajtási idővel lehet jellemezni, e teljesítményt az órajel növelésével lehet fokozni. Az órajel frekvenciájának növelése azonban számos más problémát is felvet, melyek közül a legnagyobb az ún. „von Neumann-féle szűk keresztmetszet".

A szűk keresztmetszetet a processzor sebességi korlátja okozza, az az idő, amely alatt az adatokat és utasításokat a memóriából elő tudja venni. Sok ötlet és erőfeszítés kellett a probléma megoldásához már a nagy rendszerekben is.

A nagygépes tervezők sohasem rendelkeztek elegendően gyors memóriával, a processzorok mindig gyorsabbak voltak. A mikroprocesszorok fejlődésében pedig ennek az ellenkezője volt igaz. Amikor a mikroprocesszorok megjelentek, a félvezetős memóriák gyártása kiforrott technológia szerint történt, és miután mindkettőt ugyanazzal a technológiával gyártották, a memória és a processzor sebessége napjainkig lépést tartott egymással.

Mára megváltozott a helyzet. A processzorok — a 80386-os is — 25 MHz-cel képesek működni, de ez a sebességnövekedés túlnőtt a memóriák hozzáférési idején, így a PC-tervezők ismét a régi problémával néznek szembe.

A szűk keresztmetszet bármilyen feloldásának a hozzáférés gyorsítását és a memóriához fordulás számának csökkenését is magába kell foglalnia. A hozzáférési idő növelésének egy lehetséges egyszerű módja a nagyobb sebességű áramkörök gyártását biztosító technológiák alkalmazása a memóriagyártásban. A statikus RAM-chipek gyorsabbak, mint a dinamikus RAM-ok, a TTL és ECL memóriák még ennél is gyorsabbak. A gyorsabb chipeknél gond a kisebb memóriakapacitás, a magasabb ár és a nagyobb fogyasztás.

A kisebb memóriakapacitás azt eredményezi, hogy több chipet kell felhasználni, ez pedig megnöveli a memória és a processzor közötti távolságot. Az eszköz nagyobb sebességéből adódó előnyök egy része így elvész. A nagyobb fogyasztás miatt nagyobb tápegység kell, és drágább hűtés a nagyobb teljesítmény okozta hő eltávolításához. A másik lehetőség csökkenteni a memóriához fordulások számát. Ezt úgy lehet elérni, hogy vagy hatékonyabb utasításkészletet kell készíteni, vagy minden memóriához fordulás alkalmával nagyobb mennyiségű adatot kell elővételezni (fetch). Például az összetett utasítás kiküszöböli a néhány tucat memória vagy regiszterhez fordulás szükségességét, és emiatt sokkal gyorsabb is. Egyszerűen fogalmazva: a 32 bites adat elérése gyorsabb, mint a 8 bites adatot négyszer kiolvasni.

A tervezők mindkét technikát felhasználták a processzorteljesítmény javítására. Sajnos a nagyobb utasításkészlet és a szélesebb adatmező nagyobb chipméretet igényel, és a gyártási technológia határát már elérték a 80386-os méretű chipekkel.

A másik út a memória-hozzáférési sebesség növelése: sokkal gazdaságosabban használjuk ki az egyes utasításciklusok alatti időt azzal, hogy a következő memóriahelyekhez az alatt az idő alatt fordulunk, amíg a processzor egy utasítást végrehajt.

Ezt a technikát „pipeline-nak" (=láncolás, összefűzés) hívjuk, ami azt jelenti, hogy az adathozzáférési idő majdnem olyan hosszú lehet, mint a processzor végrehajtási ciklusa. Így a végrehajtási arányt legfeljebb csak megduplázni lehet. Mindazonáltal a pipeline módszert már a 8086-osnál is felhasználták.

A gyorsabb memória használata nem biztos, hogy a leggazdaságosabb megoldás. De nincs szükség a teljes rendszerben nagy sebességű memória használatára. A legtöbb gyors memória-hozzáférést egy viszonylag kis memóriaterületnél biztosítják. A processzorchipben már néhány általános célú regiszter kialakítása is észrevehető sebességnövekedést ad.

Ezt a szemléletet tovább lehet fejleszteni, ha egy kisméretű, de igen gyors memóriamezőt — mondjuk 64 kbájtot — alakítunk ki a teljes, több Mbájtos memóriamezőn kívül. A legtöbb programban a memóriához fordulás kb. 80 százaléka ehhez az ún. „cache" (magyarul talán gyors memóriának fordítható) területhez történik.

A cache memóriát tulajdonképpen minden nagyszámítógép-rendszernél megtalálhatjuk, és a csúcs mikroprocesszoroknál is elkezdték az alkalmazását. Az Inmos Transputer, amely egy nagy sebességű szupermikro, tetézi ezt azzal, hogy a néhány kbájtos, nagy sebességű cache memóriát magán a processzorchipen helyezi el.

Az IBM 64 kbájtos cache memóriákat használ a 80386 alapú, 25 MHz-es órajellel működő

Model 70-A21 PC-iben. Ez egy olyan tényező, melyet a jövőben egyre gyakrabban fogunk tapasztalni a mikroprocesszoros rendszerekben.

A cache memória segítségével csak akkor jutunk sebességnövekedéshez, ha azt jól és hatékonyan használjuk fel. Ezt egy speciális hardver beépítésével érhetjük el, amely ellenőrzi a cache tartalmát. Amikor hozzá akarunk férni egy a memóriában lévő szóhoz, az mindenkor automatikusan betöltődik a cache-be a közvetlen szomszédaival együtt. A cache vezérlő ezt a blokkot címével együtt tárolja, így együttesen érhetők el. A processzor először a cache vezérlőhöz fordul, vajon a szükséges cím és tartalma a cache-ben rendelkezésre áll-e; ha nem, akkor normális memóriaciklust hajt végre.

Virtuális memória

A virtuális memória használatakor az operatív memória a program számára mint cache használható. Ennek az az előnye, hogy a programozó a létező RAM-területnél hosszabb programot is írhat.

Ahogy a cache memória, a virtuális memória is speciális hardvert igényel a hatásos működtetéshez. Ez az áramkör a programoknak vagy adatoknak azt a részét tölti be a memóriába, amely éppen szükséges. A nem használt szegmenseket lemezen tárolja, és a RAM-ból törli. A memória hierarchikus felépítése a lemeztől a processzorregiszterig az *1.ábrán* látható.

A virtuális memóriarendszereknek sok előnyük van. Ha a programozó vagy a program tudja, hogy a használható memória mérete majdnem végtelen, a program méretére vonatkozó minden kényszer megszűnik, a nagy adatbázisok látszólag a RAM-ban lehetnek, és a programok olyan nagyok, amekkorát a programozó csak akar. A virtuális memória legnagyobb előnye azonban mégiscsak az, hogy a programokat és az adatokat cserélgetni lehet a gépek között, akkor is, ha azok memóriakonfigurációja különböző.

A virtuális memóriát nagygépeken már régóta alkalmazzák, de csak manapság terjedt ki a mikroprocesszoros számítógépeken. A 80386-oshoz és a 68020-ashoz hasonló processzorok memóriakezelő áramkörökkel rendelkeznek, amelyek hatásosan szervezik és kezelik a virtuálismemória-rendszereket. Ez tehát egy másik olyan tulajdonsága a közepes és nagygépeknek, amelyet sikeresen átültettek a PC-kbe is.

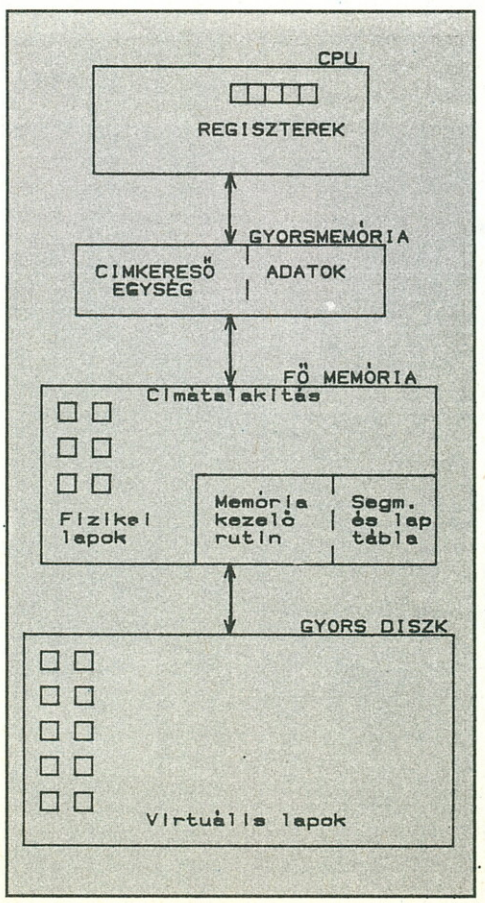
A sok nagygépes tervezési technika adaptációja a mikroprocesszorok és a mikroprocesszor bázisú rendszerek utolsó generációjába jelzi, hogy az ezekhez tartozó technológiai szint is fejlődik.

A szűk keresztmetszet nem fog megszűnni — a memóriák egyre gyorsabbak lesznek, de sohasem eléggé gyorsak. A cache memóriaelve csak a probléma átmeneti megoldását jelenti. Nem szabad olyan 50 MHz-es processzort gyártani, amely idejének nagy részét a memóriára való várakozással tölti; a processzorchipbe helyezett memóriával lehetne segíteni ezen a dolgon, de itt is korlátokba ütközünk: egyrészt a chip méretébe, másrészt a fizika törvényeibe. Az sem csökkenti a korlátot, ha az adat méretét növeljük. Ha 32-ről 64-re növeljük a bitszámot — a teljesítményt ez is csak megduplázza (ugyanakkor megkettőzzük azokat az elemeket, amelyek a széles adatbuszt kezelik). A 32 bit minden bizonnyal a maximális adatméret a PC-n.

A spektrum másik végén ott áll még az a lehetőség, hogy tökéletesítsük a tárolóeszköz-elérési sebességet. A lemez egyre gyorsabb és nagyobb lesz. Már nem utópia a 600 Mbájtos PC-s winchester, 20 ms elérési idővel. Láthatunk olyan rendszereket, amelyek nagy, gyors fixlemezes meghajtókat használnak. Néhány félvezetőgyártó előrejelzése szerint a 16 Mbites DRAM-oké a jövő.

De mindezek mellett hol marad a felhasználó olthatatlan vágya: olcsón még nagyobb számolási teljesítményt kapni? Két válasz van — az egyik az egyidejű, a másik az elosztott végrehajtás. Már mindkét technikát eredményesen alkalmazzák a teljesítmény fokozására, mind a nagygépek, mind a minigépek világában.

1. ábra



Multiprocesszoros rendszerek

A multiprocesszoros rendszerben a PC több processzort tartalmaz, ezek mindegyike megfelel egy 25 MHz-es 80386-os processzornak, néhány Mbájtos RAM-mal. Az egész rendszer egymás közötti és a felhasználóval történő kommunikációja egy vezélőprocesszor felügyelete alatt zajlik. A program futása logikailag úgy van felosztva, hogy az egyes részeken a processzorok egyidejűleg dolgoznak. Az ilyen gép teljesítménye durván egyenlő a felhasznált processzorok számával. A multiprocesszorral bármilyen teljesítményű számítógép építése lehetséges — ilyen az az asztali szuperszámítógép is, amelyet a Meiko Computing Surface már meg is épített. Azoknak a felhasználóknak, akik egyre nagyobb és nagyobb számolási teljesítmény után sóvárognak, ez a típus egy logikus választási lehetőséget nyújt.

A legtöbb megvásárolható rendszerben szokásos megoldás a megosztott feldolgozás alkalmazása és az erőforrások megosztása több gép között, amely számos közepes teljesítményű PC összekapcsolását jelenti, esetleg mini-, közép-, vagy éppen nagygépekkel közös hálózatba kötve. Ez a hálózat az egész földgömbre is kiterjedhet. A hálózatba kötött rendszerek biztosítják a számítógépek erőforrásként való használatát, a kialakított adattároló, visszakereső és kommunikációs programok segítségével.

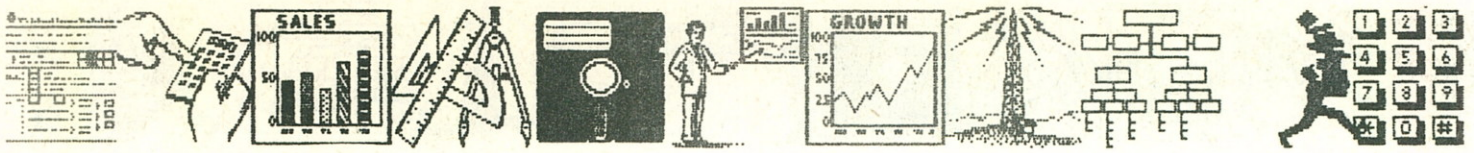
PS/2: a nagyra nőtt mikro

Nem egyszerű feladat a számítógép-tervezés, amely az állandó teljesítménynövelés szükségességével néz szembe. Ez összefügg a nagyobb teljesítmény felé ható két különálló irányzat létezésével. Építhető egy célirányos multiprocesszoros rendszer, vagy összpontosíthatunk egy sokkal egyszerűbb hálózatorientált rendszerre. A probléma az, hogy a PC tömegtermék: szükségképpen általános célú számítógép, amelyet gazdaságosan kell gyártani és a lehető legalacsonyabb áron eladni.

A problémát az IBM a PS/2 megtervezésével próbálta megoldani, amely lehetővé teszi egy teljesen köznapi gép (standard AT) vagy kicsi, de potenciálisan egy nagy teljesítményű rendszer részeként való üzemelést.

Valójában az IBM ebben a gépben egy csomó dolgot kipróbált. Éppúgy konfigurálható nagy teljesítményű mikroprocesszoros rendszerként, mint egy egyszerű hálózati rendszer elemeként, és lépéseket tettek a rendszer további felgyorsítására a bevitel és kivitel működésében jelentkező késleltetések kiküszöbölésével.

A B/K-késleltetés komoly gond a processzor



teljesítménye szempontjából, és jelentősen hozzájárul a lelassuláshoz. A PC-ben a sebességet tekintve két kritikus eszköz van: a lemezmeghajtó és a video-képernyő. Más eszközök, mint a billentyűzet vagy az egér, melyek csak bevitelre szolgálnak, nem rendelkezhetnek olyan válaszidőkkel, melyek gyorsabbak, mint a felhasználó. A lemezmeghajtó leglényegesebb késleltetését a fejhez tartozó mechanikus mozgás okozza. A berendezés nagyon gyors lehet, ha a fej éppen a megfelelő átviendő blokk fölött áll. A video-képernyő által okozott késleltetés egyszerűen a szükséges váltások és frissítések óriási mennyisége miatt van. A legegyszerűbb válasz a fenti problémára talán az lehetne, hogy gyorsítsuk fel a B/K-eszközöket, de ezt könnyebb mondani, mint megtenni. Ennél jobb ötlet az, melyet a legtöbb nagygépnél alkalmaznak. El kell venni a B/K-funkciókat a processzortól. Egy speciális B/K-processzor — mint például a lemezmeghajtó — tudja kezelni a B/K-t, mialatt a processzor mást csinálhat.

Ez természetesen a koprocesszor-elv továbbgondolása úgy, hogy az már nem egy alárendelt processzor többé. Vállalhat teljes felügyeletet mind a rendszerben, mind a memória felett. Az IBM ezeket az intelligens B/K-folyamatokat csatornáknak (channel) nevezi, és a nagygépektől vitte át a PS/2 sorozatú gépekre.

A PS/2-nél használatos Micro Channelt ugyanakkor nem szabad összetéveszteni a hagyományos PC-n található busszal. A hagyományos busz esetén a processzor elérheti a memóriát és a memóriába ágyazott B/K-eszközöket; nem tervezték arra, hogy megengedjék más eszközöknek a buszvonalak ellenőrzését. Ha megkísérelnénk két processzorral vezérelni a hagyományos buszt, az eredmény az lesz, hogy a busz leragad, és a rendszer szétesik. (El lehet érni, hogy a busz ne ragadjon le, de ahhoz nagyon sok pótlólagos áramkörre van szükség.) A PS/2 egy buszvezérlő segítségével éri el, hogy több versengő eszköz rajta lehessen a buszvonalakon. Ez nemcsak az intelligens perifériákat, hanem a processzor többszörözését is támogatja.

A Micro Channel Architecture (MCA) ezért a legnagyobb előrelépés a PC-tervezésben. Lecsökkenti a B/K-k késleltetését, és lehetővé teszi az intelligens B/K-hardver használatát. Az MCA fokozza az adatbiztonságot a rendszeren belül, mert a B/K mindig rendszerhibák potenciális forrása volt. Bármilyen B/K-porttal, ha közvetlenül a memóriaterületbe ágyazzuk, az egész rendszert tesszük sebezhetővé. Ha a Micro Channelt használjuk, intelligens B/K-vezérlőkkel átléphetjük ezt a problémát: minden hiba ellenőrizhető lesz már a „csatorna" szintjén, nem pedig a memóriában.

A Micro Channel igen kiváló módszer arra, hogy nagy mennyiségű adatot (blokkot) gyorsan átvihessünk a memória és a B/K-eszközök között (lemez, video-kijelző). Lényegében a PS/2-t úgy tervezték, hogy másodpercenként több mint 20 millió karaktert képes mozgatni. Az adatátvitelnek ez a fajtája nem igényel semmilyen processzorfelügyeletet.

Természetesen az MCA előnye és összhangja függ a gépen futó szoftvertől is, ebből fakad az OS/2 operációs rendszer fontossága.

Igen érdekes, hogy a PS/2 sorozatú gépek tervezése milyen figyelemre méltó azonosságot mutat a nagy IBM korai időszakának nagygépeivel. A 2. *ábrán* egy ikerprocesszoros nagy IBM rendszert láthatunk, de az sem lehetetlen, hogy egy PS/2 gépet egy második processzorral.

SAA és hálózatkialakítás

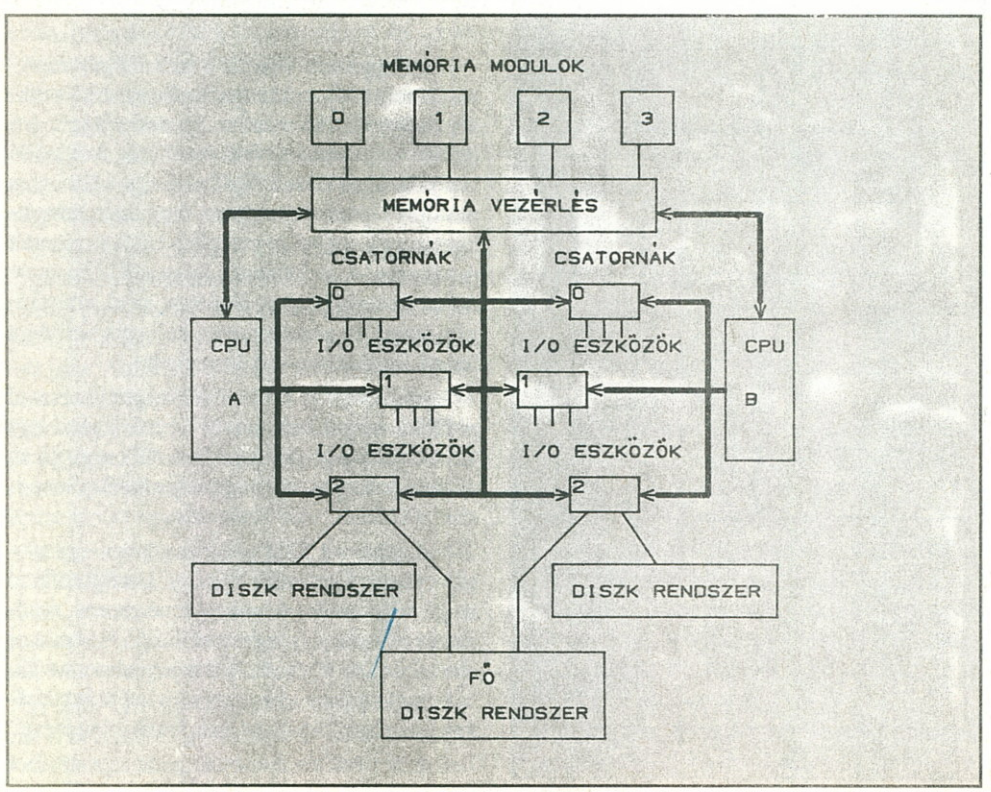
Az architektúrát illetően a nagygépek, a minigépek és PC-k fejlesztési iránya nem egészen azonos. A legtöbb számítógépgyártó eltökélt szándéka, hogy egy közös architektúra kialakítását célozza meg a különböző méretű gépeknél. Az elképzelés az, hogy a felhasználó egyszerűen tudja bővíteni (méretben) gépét anélkül, hogy ki kelljen dobnia a már meglévő szoftvereibe fektetett munkát. Lehetőséget kell adni a felhasználóknak, hogy a különböző méretű gépeiket könnyen hálózatba köthessék. A PC-felhasználók számára az a legnagyobb jelentőségű, hogy az IBM termékek rendszerfelépítése egységesített. Ezt hívják SAA-nak (System Application Architecture). Ez gondoskodik az átmenetről a három legnagyobb IBM-környezet: a System/370, a System/3x és a Personal System/2 között. A közös tervezési filozófiának négy tényezője van: a hasonló felhasználói elérési mód, a közös programozási felület, a közös kommunikációs támogatás és utoljára a talán legfontosabb: a közös alkalmazások.

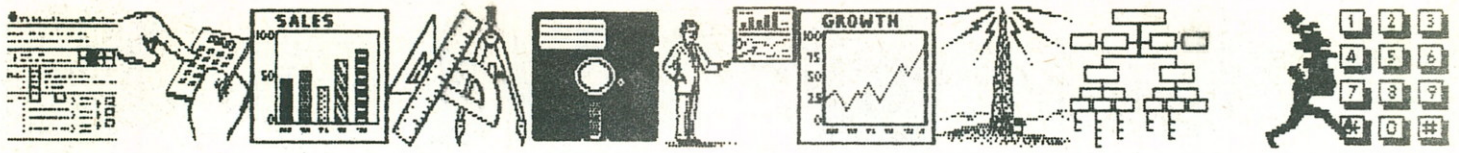
Az SAA nem egyszerűen egy termék: tervezési protokollok és filozófiák rendszere. Használható multiprocesszoros hálózatok kialakítására, amelyek különböző méretű gépeket foglalnak magukba. Az SAA filozófia mind a szoftver, mind a hardver szintjén alkalmazható. Akkor lesz igazán jó hatásfokú, ha mindkettőt egyszerre felhasználjuk.

Bár az SAA nemrég jött divatba, ez az a termék, melyen az IBM már évek óta dolgozik. Sok elemét azonban már eddig is beépítették — például a 80286 és különösen a 80386 tervezésénél vették figyelembe az SAA igényeit. Ennek egy része volt az IBM-től az Intel felé áramló szaktudás, aminek az eredménye a közös IBM/Intel mérnöki tervezőcsoport.

Az új processzorok nélkülözhetetlenek voltak az SAA PC-elemeinek elkészítésében. Kellett egy gyors, nagy teljesítményű processzor, nagy memóriatartomány elérésének lehetőségével, ami nagy teljesítményű konkurens (egyidejű) programvégrehajtást is biztosít (multi-

2. ábra





tasking system). Az ehhez szükséges jellemzőkről gondoskodtak a 286-os és 386-os típusokban.

Az egyidejűség — egyidejű végrehajtás — az SAA filozófiában fontos, lévén ez a nagy teljesítményű hálózat alapvető eleme. A PC-t a hálózathoz csatolva bizonyára képes használni azt a hálózatot önállóan, bármely alkalmazói programból. Ezt már minden PC-s hálózati rendszer tartalmazza, de két árnyoldala van, melyek gátolják a PC-hálózatok használatát a kritikus alkalmazások esetében.

Az első árnyoldal az az egyszerű mód, ahogyan az illetéktelen felhasználó elérheti a PC-ket a hálózaton keresztül; a jelszó nem igazi védelem egy védelemtörő hacker ellen. Másodszor, a beérkező adat bármilyen meghibásodása a futó program és a hibás adat programbeli kölcsönhatására vezethet.

Mindkét eset gyenge oldala abból fakad, hogy a felhasználói és a hálózati rendszerprogramok ugyanazon a memóriaterületen vannak, és megszakításos vezérléssel kapcsolgatunk a két program között. A probléma megoldható az egyidejű végrehajtású környezettel, ahol is a processzort úgy tervezték, hogy az egyes programok (felhasználások) különböző memóriablokkokban legyenek. Így aztán egy „hardverfal" keletkezik a gépen futó különböző eljárások között. Egy ilyen falat biztosít a 286-os és 386-os processzor ún. védett (protected) memóriakezelése *(3. ábra)*.

A „védett módú" egyidejű végrehajtási rendszerben a felhasználó számára lehetővé válik egy vagy több program futtatása és ugyanúgy a hálózat kialakítása is. Egy nagy teljesítményű processzor esetében a feladatok között az idők meghatározásával a felhasználónak nem kell törődnie. A védett módú egyidejű végrehajtású gépek úgy fognak viselkedni, mintha két vagy több elkülönített gép lenne, melyek egymással kapcsolatban vannak egy kábelen keresztül.

Egy védett módú konkurens végrehajtású számítógép viselkedésében nem más, mint egy kis hálózat. Minden folyamat úgy viselkedik, mint egy független gép, és közöttük a kommunikációs lánc szoftver útján jön létre, pontosan úgy, mintha az egy összekábelezett hálózaton történne. Ez a hasonlóság a kulcsa az elosztott feldolgozást végző rendszerek létrehozásának.

Ha az ilyen gépeket hálózatba kapcsoljuk, a hálózat többi gépe további folyamatoknak látszik. Ez azt jelenti, hogy a hálózat minden egyes gép bővítésévé válik, és nem adatmozgatást jelent a gépek között. Nincs indok arra, hogy az egyik felhasználó ne futtathasson egy programot egy másik gépen. A második gép teljesítménye nem csökken, és az egész úgy tűnik az első gép felhasználójának, mintha programja a saját gépén futna.

A hálózatoknak ez a típusa — óriási teljesítmény lehetőségével — kezd nagyon hasonlatossá válni egy párhuzamos feldolgozást végző rendszerhez. Ha a hálózatban egy nagygép vagy egy „nagy" minigép van, akkor minden felhasználó ennek a gépnek a teljesítményét potenciálisan használhatja saját PC-jének teljesítménynövelésére. A jelenlegi OS/2 verzió nem támogatja az ilyen típusú rendszereket. Bizonyára a későbbi változatok tartalmazni fogják ezt is. Ugyanakkor van egy létező operációs rendszer, amely támogatja az erőforrások ilyen megosztását: ez a Unixból származtatott QNX.

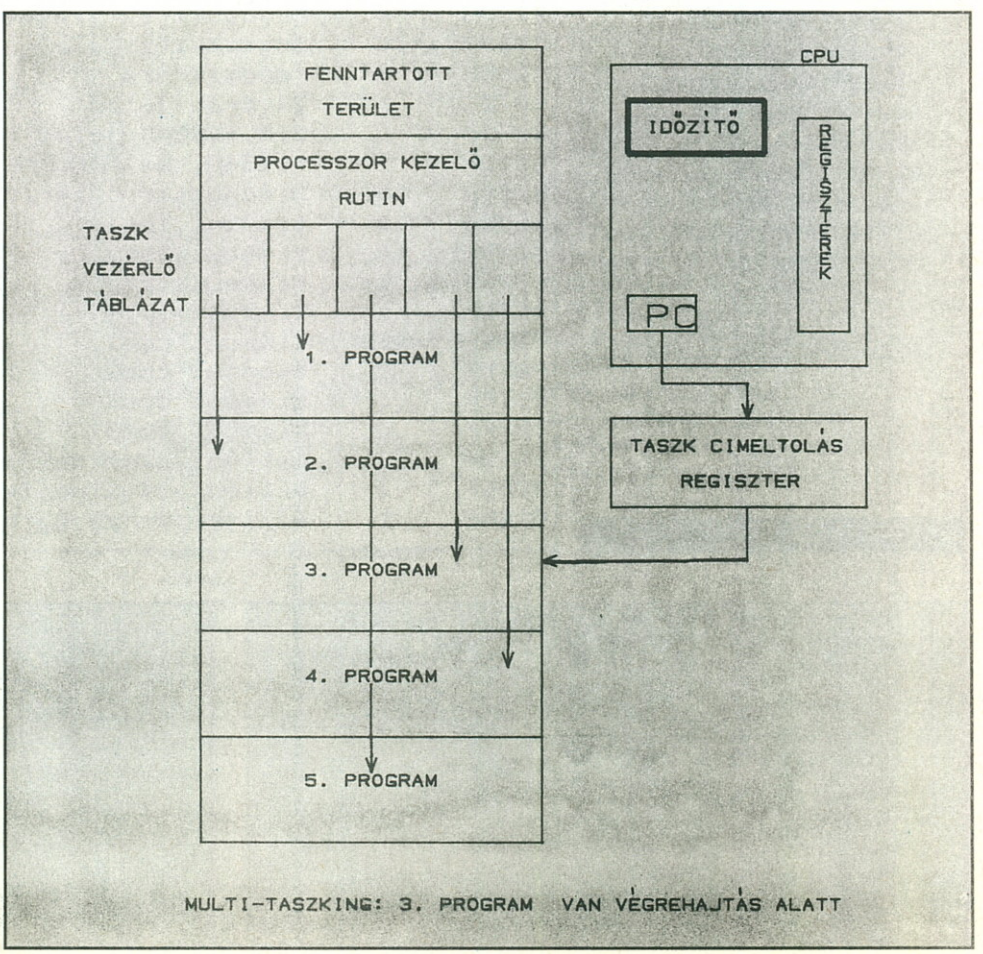
Következtetések

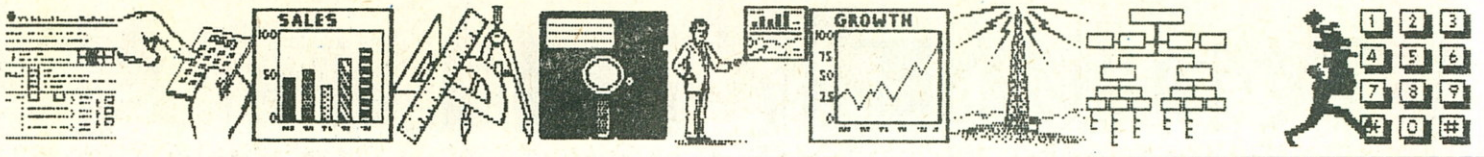
Úgy tűnik, az általános irány a számítástechnikában az, hogy azok a felhasználások fognak fejlődni, amelyek kihasználják a gép teljes teljesítményét (lehetőségeit), és még 10 százalékkal többet azok különleges jellemzőiből. Mostanáig a félvezetőgyártásban való jártasság irányította a folyamatokat a még nagyobb teljesítményű személyi számítógépek felé. A technológia azonban elérte saját határát, és ezért a továbbiakban nem várhatunk már ilyen gyors fejlődést.

A határok, amelyekkel a PC-tervezők szembenéznek, ugyanazok, mint amelyek előtt a nagygépek tervezői álltak 10 évvel ezelőtt. A megoldások is hasonlóak: valójában a különbség minimálissá vált a különálló nagygép és a különálló PC között. Csak a különböző félvezető-technológiákban van eltérés. Ez a tendencia a számítógépipart új, eddig ismeretlen vizekre irányítja. Az asztali szuperszámítógép és a nagy teljesítményű PC-s óriás hálózatok fognak dominálni, képessé válva egyedüli működésre vagy egy nagy, párhuzamos közös processzorként üzemelni. Ennek a rendszernek a jellemzői és képességei jórészt még ismeretlenek. A PC-fejlesztők sem bízhatnak a nagygépek fejlődésének lemásolásában. A helyzet izgalmas és ingerlő. A játszma visszakanyarodott a számítástechnikához.

(A Personal Computer World 1989. novemberi számának 176—181. oldalán megjelent „The next generation" c. cikk alapján)

3. ábra





Ki itt belépsz...

Mikor az IBM-PC XT, AT számítógépet bekapcsoljuk, egy csak olvasható tárolóban (EPROM-ban) tárolt program, az ún. ROM-BIOS futása indul el. A ROM-BIOS-ban lévő szubrutinok beállítják a gép perifériáinak (képernyő, lemezmeghajtók száma és típusa, nyomtató, soros vonalak, billentyűzet stb.) paramétereit, és ezek a memória 400H címén (0040H:0000H) kezdődő és 100H hosszúságú ún. BIOS-RAM munkaterületen tárolódnak. Táblázatunk az itt található változókról ad felvilágosítást.

BIOS-RAM-belépések

Mindegyik cím a BIOS-RAM szegmensre vonatkozik (0040H).

Cím	Bájt	Tartalom
0000H	2	cím COM1 (F8 03=03F8H)
0002H	2	cím COM2 (F8 02=02F8H)
0004H	2	cím COM3
0006H	2	cím COM4
0008H	2	cím LPT1
000AH	2	cím LPT2
000CH	2	cím LPT3
000EH	2	cím LPT4
0010H	2	DIP-kapcsoló az alaplemezről
0012H	1	FLAG a diagnosztizáló szoftvernek
0013H	2	RAM-méret kbájt-ban
0015H	2	hibakód a diagnosztizáló szoftvernek
0017H	3	billentyü-státusz (0017=shift-status)
001AH	2	billentyüpuffer 1
001CH	2	billentyüpuffer 2
001EH	32	billentyüpuffer (15+1 szó)
003EH	11	floppyvezérlö-státusz
0049H	1	videodisplay mód
004AH	2	karakter/sor
004CH	2	video-RAM mérete bájtban
004EH	2	video-RAM címe
0050H	16	kurzorpozíció (képernyöoldal 2 bájt)
0060H	2	kurzorméret
0062H	1	képernyöoldalak száma
0063H	2	6845 videovezérlö címe
0065H	1	videovezérlö jelzöbájtja
0066H	1	palettaszín
0067H	4	kezdöcím és szegmens egy ROM-hoz
006BH	1	megszakításjelzö
006CH	4	rendszeróra
0070H	1	24 órás jelzö
0071H	1	CTRL-BREAK-jelzö
0072H	1	=1234H, ha CTRL-ALT-DEL volt
0074H	4	winchester-vezérlö státusz-bájt
0078H	1	timeout lpt1
0079H	1	timeout lpt2
007AH	1	timeout lpt3
007BH	1	timeout lpt4
007CH	1	timeout com1
007DH	1	timeout com2
007EH	1	timeout com3
007FH	2	timeout com4
0080H	2	billentyüzet-puffer kezdöcíme (=001E)
0082H	2	billentyüzet-puffer végcíme (=003E)
0100H	1	AT 1=képernyö-hardcopy (PRTSC)

Grafika

Az Amiga programozása assembly nyelven III.

A sorozat előző részében tömören ismertettem az Amiga grafikus üzemmódjait. Leírtam, hogyan lehet a ROM-rutinok felhasználásával megnyitni screeneket és ablakokat. Ennek kapcsán szó volt a NewScreen és a NewWindow struktúrák szerepéről és alkalmazásáról. Azt ígértem, hogy az IDCMP (Intuition Direct Communications Message Port) alkalmazásáról még szó lesz.

Az Amiga többfeladatos (multitasking) gép, tehát a programok nemcsak egymás után, hanem egyidejűleg is futhatnak. Az egyes programok egymással nincsenek szinkronban, így egyikük sem tudja a másikról, hogy éppen hol tart. A feladatok szinkronizálására az Amiga jelzőbiteket használ. Ezek gyűjtő elnevezése: signal.

Amikor egy új feladat elindul, a rendszer a rendelkezésére bocsát 32 darab signalt (ez 1longword). A feladat ezek után képes bármelyik signalra várakozni. Ilyenkor amíg a signal értéke 0, a feladat ún. waiting state-ben van, ami azt jelenti, hogy futása fel van függesztve. Mihelyt a signal értéke 1 lesz, a feladat visszakerül eredeti, futó állapotába. A 32 signalból 16 a rendszer számára van lefoglalva, 16-ot pedig saját céljaira használhat fel a programozó. (A szabad bitek helyét és számát nyilvántartja a rendszer.)

Ha a számítógép bármilyen aktivitást észlel a külvilág felől — például lenyomtunk egy billentyűt, elmozdítottuk az egeret —, egy ún. message-et (üzenetet) hoz létre. A message egy olyan struktúra (adatsorozat), amely leírja az illető eseményt. A message létrejöttéről — az esemény bekövetkezéséről — pedig a fent említett signalok valamelyike ad hírt. A message-hez az ún. MsgPort struktúra segítségével juthatunk hozzá.

Az IDCMP az MsgPort speciális fajtája, és szorosan összefügg az ablakkezeléssel. Ennek a portnak a segítségével szerezhetünk tudomást mindarról, ami egy adott ablakkal kapcsolatban történik. Lehetőségünk van arra, hogy a rendszert „ráállítsuk" bizonyos események figyelésére. Azt, hogy a rendszer mit figyeljen (az IDCMP osztályát), a NewWindow struktúra IDCMP_Flags nevű elemébe kell beírnunk. Ha például arról akarunk tudomást szerezni, hogy a felhasználó az ablakot be akarja csukni a pointerrel, akkor a $200 értéket kell az IDCMP_Flags helyére beírni. Ha ezt megtettük, az IDCMP signal bitje csak akkor fog jelezni, amikor az illető ablakot valaki be akarja csukni.

Az IDCMP_Flags változóban azoknak az IDCMP osztályoknak az összegét kell elhelyezni, amelyek bekövetkezéséről

1. lista

1.LISTA

```
;KONSTANS-DEKLARÁCIÓK
GetMsg:                 equ     -$174
ReplyMsg:               equ     -$17a
Wait:                   equ     -$13e
IDCMPCLOSEWINDOW:       equ     $200
WINDOWCLOSE:            equ     8
IDCMP_Flags:            equ     IDCMPCLOSEWINDOW
Flags:                  equ     WINDOWCLOSE+WINDOWDRAG+ACTIVATE
                                .
                                .
                                .
                                .
                                .
                                .
;EZT A RÉSZT KELL BEILLESZTENI ABBA A PROGRAMBA, AMELYET
;VÁRAKOZTATNI AKARUNK
Sleep:  bsr     WaitForIDCMP
        cmpi.l  #IDCMPCLOSEWINDOW,Class
        bne.s   Sleep
                                .
                                .
                                .
                                .
                                .
                                .
;SZUBRUTINOK
WaitForIDCMP:
        move.l  ExecBase,a6     ;az Exec báziscíme
        bsr.s   WaitForSignal   ;várakozás a signal-ra
        move.l  a5,a0           ;az MsgPort címe
        jsr     GetMsg(a6)      ;az üzenet átvétele
        tst.l   d0              ;volt üzenet?
        beq.s   NoMsg           ;nem volt, kilép
        move.l  d0,a1           ;az IntuiMessage címe
        move.l  20(a1),Class    ;a 20-ik elem az osztály
        jsr     ReplyMsg(a6)    ;válaszolni is kell
NoMsg:  rts
WaitForSignal:
        move.l  Window,a0       ;a Window struktúra címe
        move.l  86(a0),a5       ;a 86-ik elem a UserPort
        move.b  15(a5),d1       ;az mp_SigBit elővétele
        clr.l   d0              ;az mp_SigBit megszabta
        bset    d1,d0           ;helyen 1-es álljon!
        jsr     Wait(a6)        ;a task vár a signal-ra
        rts
                                .
                                .
                                .
                                .
                                .
                                .
;VÁLTOZÓK
Class:          dc.l    0       ;az IDCMP osztályt tárolja
```

visszajelzést akarunk kapni. Az alábbi IDCMP osztályok léteznek:

SIZEVERIFY equ 1 ; az ablak méretét meg akarják változtatni

NEWSIZE equ 2 ; az ablak mérete megváltozott

REFRESHWINDOW equ 4 ; az ablak tartalma megsérült, ezért frissítésre szorul

MOUSEBUTTONS equ 8 ; az egér megnyomása

MOUSEMOVE equ $10 ; az egér elmozdítása

GADGETDOWN equ $20 ; egy gadget kiválasztása

GADGETUP equ $40 ; egy gadget elengedése

REQSET equ $80 ; egy requestert nyitottak ki az illető ablakban

MENUPICK EQU $100 ; egy menüpont kiválasztása

CLOSEWINDOW equ $200 ; az ablak becsukása

RAWKEY equ $400 ; billentyűzet-input

REQVERIFY equ $800 ; requestert akarnak kinyitni ebben az ablakban

REQCLEAR equ $1000 ; a kinyitott requester már megszűnt

MENUVERIFY equ $2000 ; menüt akarnak megjeleníteni ebben az ablakban

NEWPREFS equ $4000 ; a Preferences struktúra tartalma megváltozott

DISKINSERTED equ $8000 ; lemez behelyezése

DISKREMOVED equ $10000 ; lemez kivétele

WBENCHMESSAGE equ $20000 ; Workbench üzenet

ACTIVEWINDOW equ $40000 ; az ablak aktivizálása

INACTIVEWINDOW equ $80000; az ablak inaktivizálása

VANILLAKEY equ $200000 ; a Set Mapnak megfelelő RAWKEY

INTUITICKS equ $400000 ; óra, amely másodpercenkét kb. 10-et „ketyeg"

Lássunk végre egy konkrét példát! Az *1. lista* egy olyan programrészletet mutat be, amely az ablak becsukását — vagyis a becsukó szimbólum aktivizálását — érzékeli. A múlt havi példaprogramból és a továbbiakban bemutatásra kerülő kis programból is az ESC billentyű lenyomásával lehet kilépni. Az 1. listán látható részlettel kulturáltabbá tehetjük programjainkat, amelyek ezután már az ablak kikapcsolására fognak befejeződni.

A Sleep nevű ciklust programunknak arra a helyére kell beszúrni, ahol az ablak kikapcsolására várakozni akarunk. A WaitFor IDCMP szubrutin első dolga az, hogy meghívja a WaitForSignal rutint. Ez utóbbi elsőként az ablak Window struktúrájának (ezt az ablak megnyitásakor a rendszer készítette el) egyik elemét veszi elő. Mégpedig az ún. UserPort struktúra kezdőcímét, ez a cím található a Window 86-os ofszetjén.

A UserPort struktúra ugyanolyan felépítésű, mint a korábban már említett MsgPort. Minden MsgPort struktúra 15. bájtja azt adja meg, hogy az illető porthoz hányadik signal bit

2.LISTA

```
ExecBase:           equ     4
OpenLibrary:        equ     -$228
CloseLibrary:       equ     -$19e
OpenScreen:         equ     -$c6
CloseScreen:        equ     -$42
OpenWindow:         equ     -$cc
CloseWindow:        equ     -$48
Move:               equ     -$f0
Draw:               equ     -$f6
SetRGB4:            equ     -$120
SetAPen:            equ     -$156
SetBPen:            equ     -$15c
SetDrMd:            equ     -$162
HIRES:              equ     $8000
CUSTOMSCREEN:       equ     $01
IDCMP_Flags:        equ     0
WINDOWSIZING:       equ     $0001
WINDOWDRAG:         equ     $0002
ACTIVATE:           equ     $1000
Flags:              equ     WINDOWSIZING+WINDOWDRAG+ACTIVATE

Start:
        move.l  ExecBase,a6         ;az Exec báziscíme
        clr.l   d0                  ;verziószám: 0
        lea     IntuitionName,a1    ;mutató az Intuition nevére
        jsr     OpenLibrary(a6)     ;az Intuition megnyitása
        move.l  d0,IntuitionBase    ;a báziscím elmentése
        beq     NoIntuition         ;ha nem sikerült, kilép
        clr.l   d0                  ;verziószám: 0
        lea     GfxName,a1          ;mutató a Graphics nevére
        jsr     OpenLibrary(a6)     ;a Graphics megnyitása
        move.l  d0,GfxBase          ;a báziscím elmentése
        beq     NoGfx               ;ha nem sikerült, kilép
        move.l  IntuitionBase,a6    ;az Intuition báziscíme
        lea     NewScreen,a0        ;mutató a NewScreen-re
        jsr     OpenScreen(a6)      ;a screen létrehozása
        move.l  d0,NWScreen         ;beírja a NewWindow-ba
        lea     NewWindow,a0        ;mutató a NewWindow-ra
        jsr     OpenWindow(a6)      ;az ablak kinyitása
        move.l  d0,Window           ;a Window struktúra mutatója
        move.l  d0,a0               ;előkészítés
        move.l  50(a0),RastPort     ;mutató a RastPort-ra
;A GRAFIKAT ELŐÁLLÍTÓA PROGRAMTÖARZS
        move.l  GfxBase,a6          ;a Graphics báziscíme
        move.l  NWScreen,a0         ;a Screen struktúra kezdőcíme
        add.l   #44,a0              ;a ViewPort kezdőcíme A0-ba
        move.l  a0,ViewPort         ;... de el is kell menteni
        moveq   #1,d0               ;1-es palettaszín
        moveq   #15,d1              ;R=15
        moveq   #0,d2               ;G=0
        moveq   #0,d3               ;B=0
        jsr     SetRGB4(a6)         ;a szín beállítása
        move.l  ViewPort,a0         ;a ViewPort kezdőcíme A0-ba
        moveq   #2,d0               ;2-es palettaszínről lesz szó
```

```
      moveq    #0,d1              ;R=0
      moveq    #15,d2             ;G=15
      moveq    #0,d3              ;B=0
      jsr      SetRGB4(a6)        ;a szín beállítása
      move.l   RastPort,a1        ;a RastPort báziscíme
      moveq    #1,d0              ;1-es palettaszín
      jsr      SetAPen(a6)        ;ő lesz az APen
      moveq    #2,d0              ;2-es palettaszín
      jsr      SetBPen(a6)        ;ő lesz a BPen
      move.w   #$fff0,34(a1)      ;LinePrtn (vonalmintázat)
      moveq    #1,d0              ;JAM2-es üzemmód
      jsr      SetDrMd(a6)        ;az üzemmód beállítása
      move.l   #50,d0             ;a kurzor x koordinátája
      move.l   #20,d1             ;a kurzor y koordinátája
      jsr      Move(a6)           ;a kurzor pozicionálása
      move.l   #200,d0            ;a végpont x koordinátája
      move.l   #80,d1             ;a végpont y koordinátája
      jsr      Draw(a6)           ;a vonal meghúzása
Wait:
      cmpi.b   #116,$bfec01       ;az ESC közvetlen figyelése
      bne      Wait               ;(ide szúrható be az 1.lista)
      move.l   IntuitionBase,a6   ;az Intuition bázisa
      move.l   Window,a0          ;a Window struktúra
      jsr      CloseWindow(a6)    ;az ablak bezárása
      move.l   NWScreen,a0        ;a Screen struktúra
      jsr      CloseScreen(a6)    ;a screen megszüntetése
      move.l   ExecBase,a6        ;az Exec báziscíme
      move.l   GfxBase,a1         ;a Graphics báziscíme
      jsr      CloseLibrary(a6)   ;bezárja a Graphics-ot
NoGfx:
      move.l   IntuitionBase,a1   ;az Intuition bázisa
      jsr      CloseLibrary(a6)   ;bezárja az Intuition-t
NoIntuition:
      illegal                     ;kilépés kizárással

IntuitionName:    dc.b   "intuition.library",0
GfxName:          dc.b   "graphics.library",0
even
NewScreen:        dc.w   0,0,640,256,2
                  dc.b   2,3
                  dc.w   HIRES,CUSTOMSCREEN
                  dc.l   0,ScreenTitle,0,0
ScreenTitle:      dc.b   "Screen",0
even
NewWindow:        dc.w   100,25,350,125
                  dc.b   2,1
                  dc.l   IDCMP_Flags,Flags,0,0,WindowTitle
NWScreen:         dc.l   0,0
                  dc.w   50,30,350,160,15
WindowTitle:      dc.b   "Ablak",0
even
IntuitionBase:    dc.l   0
 GfxBase:         dc.l   0
 Window:          dc.l   0
 RastPort:        dc.l   0
 ViewPort:        dc.l   0    ;a ViewPort struktúra kezdőcíme
```

2. lista

tartozik (e bájt neve: mp_SigBit). Szubrutinunk lehívja ezt, és előállít egy olyan 32 bites értéket, amelyben az mp_SigBit által definiált bitpozícióban 1-es áll. A Wait nevű ROM-rutin megállítja a feladatunkat mindaddig, amíg a DO regiszterben megadott signal meg nem érkezik.

Ezután programunk visszatér a WaitFor IDCMP szubrutinba. Meghívja az Exec könyvtár GetMsg nevű rutinját, amely a beérkezett üzenet — ennek neve jelen esetben IntuiMessage — kezdőcímét adja vissza. A GetMsg bemeneti paramétereként az AO címregiszterbe kell tölteni az MsgPort címét, ezt most a UserPort címe.

Megkaptuk tehát az üzenet kezdőcímét, amely az IntuiMessage nevű struktúra kezdőcíme. Az IntuiMessage 20-as ofszetjén találjuk meg azt a 32 bites értéket, amely a bekövetkezett esemény mibenlétét adja meg. Ezt gondosan el is mentjük. Végül az üzenetek átvételéhez hozzátartozik a válaszadás. Ez annyiból áll, hogy az üzenet kezdőcímét az AI címregiszterbe töltve meghívjuk az ExecReplyMsg rutinját. A Sleep ciklusnak most már csak ki kell értékelnie a kapott IDCMP osztályt.

Megjegyzés: olyankor, amikor csak egyféle IDCMP osztályt engedélyezünk a NewWindow-ban, nincs is szükség a bekövetkezett IDCMP osztály vizsgálatára. Én a program bővíthetősége miatt írtam be ezt a tesztelést.

Már létre tudunk hozni screeneket és ablakokat, jó lenne ezek után rajzolni is beléjük. Ebben a Kickstart-ROM Graphics szubrutinkönyvtára nyújt óriási segítséget. Vegyük sorra a legérdekesebbeket!

A feladat az legyen, hogy egyenes, szaggatott (!) vonalat kell húznunk az (50,50) pont és a (200,120) pont között. A megoldás a *2. listán* látható.

Először be kell állítanunk a rajzolat színét. Ki kell választanunk, hogy a paletta hányadik színével akarunk rajzolni. Ezt a palettaszínt a SetRGB4 rutinnal változtathatjuk meg a kívánt színűre, ahogy azt két hónappal ezelőtt bemutattam. Most pedig közölnünk kell a rendszerrel a rajzoláshoz kiválasztott szín sorszámát.

A rajzolat színének számát a SetAPen rutinnal definiálhatjuk. A SetAPen meghívása előtt AI-be az ún. RastPort struktúra kezdőcímét kell tölteni. Ezt a címet a Window struktúrában, az 50-es ofszeten találjuk meg. D0-nak pedig a szín sorszámát kell tartalmaznia az aktuális palettában.

Itt kell megemlítenem, hogy a Graphics könyvtár háromféle rajzolatszínt különböztet meg:

APen: a tényleges rajzolatszín

BPen: a rajzolat háttérszíne (például szaggatott vonal esetében ebben a színben jelennek meg a vonalközök)

AOLPen: a foltbetöltő rutin ezt a színt tekinti határnak (Areafill OutLine Pen)

Ha tehát meg akarjuk szabni a szaggatott vonal közeinek színét, ezt a SetBPen szubrutinnal megtehetjük. A SetBPen

ugyanolyan bemeneti paramétereket kíván, mint a SetAPen.

A vonalrajzoláshoz felhasznált bitminta a RastPortban található. A RastPort struktúra 34-es ofszetjén álló 16 bites értéket (neve: LinePtrn) kell tehát beállítanunk úgy, hogy a végeredmény szaggatott vonal legyen. Írjunk ide $fff0-át! Ennek eredményeként a vonalhúzásnál 12 rajzolatszínű pontot 4 háttérszínű pont fog követni.

Hogy a helyzet még bonyolultabb legyen, a Graphics négyféle rajzolási módot különböztet meg, és nekünk most ezt is be kell állítani. A négy üzemmód a következő:

JAM1: normál rajzolás az APen-nel (0)

JAM2: ugyanaz, mint a JAM1, de ahol a LinePtrn-ben 0 áll, ott a BPen-t használja a rajzoláshoz (1)

COMPLEMENT: logikai KIZÁRÓ- VAGY (XOR) műveletet képez a már meglévő rajzolattal (2)

INVERSVID: inverz rajzolás (például négyelemű paletta esetén a 0-ásból 3-as, az 1-esből 2-es szín lesz, és fordítva (4)

Mi meg szeretnénk jeleníteni a vonalközöket is, ezért a JAM2 üzemmódot választjuk. Az üzemmódok magyarázata után zárójelben álló számok azok a kódok, amelyeket a SetDrMd rutin számára meg kell adnunk a beállításhoz. A SetDrMd szubrutin tehát a rajzolás módjának beállítására szolgál, és az üzemmód kódját a D0 regiszterbe kell betölteni. Az Al-nek emellett a RastPort kezdőcímét kell tartalmaznia.

A következő lépés az, hogy a Move rutinnal beállítjuk a grafikus kurzort az (50,50) koordinátára. A Move számára a RastPort kezdőcímét az Al-ben, a kurzor x koordinátáját a D0-ban, y koordinátáját a Dl regiszterben kell átadni.

És végre itt a beteljesülés, a Draw szubrutinnal meghúzzuk az áhított vonalat. Persze előbb az Al-be a RastPort címét, a D0-ba a vonal végpontjának x koordinátáját, Dl-be pedig az y koordinátáját kell töltenünk.

Ezt a programot is kiegészíthetjük az 1. listán bemutatott IDCMP rutinnal, én csupán a hellyel való takarékosság miatt alkalmaztam megint a billentyűzet figyelését.

Azok számára, akik a C nyelvet használják, felsorolom a fenti grafikus függvényhívások formátumát:

SetAPen (rp, pen)

SetBPen (rp, pen)

SetDrMd (rp, mode)

Move (rp, x, y)

Draw (rp, x, y)

SetRGB4 (vp, n, r, g, b)

Jó tudni, hogy a Draw után a grafikus kurzor a vonal végére kerül, ami jelentősen megkönnyíti a folytonos vonalak, sokszögek rajzolását. A sokszögek rajzolásának azonban van egy ennél is egyszerűbb módja. De erről majd a következő részben.

Tóth Zoltán

C az Amigán

Már elég régóta folynak a mesterséges intelligenciával kapcsolatos kutatások, és létezik a PROLOG, procedurális programnyelveket azonban — úgy tűnik — még jó ideig használni kell. A rövidebb programoknál még széles skáláról választhatunk. Lehet a BASIC-et vagy felhasználói program makronyelvét használni, ha egyéni problémát megoldó, egyszer-kétszer lefuttatott programról van szó. Aki a gép és a saját tudását akarja csillogtatni, írhat teljesen assemblyben. (Azért a ROM-rutinok direkt meghívását nem tanácsolom.) A sort még lehetne folytatni.

Mikor azonban a probléma komplexebbé válik, a program hossza növekszik, a legtöbb nyelv kényelmetlenné válik. Még belegondolni is rossz, hogy például egy DTP (Desktop Publishing = kiadványszerkesztő) programot teljesen assemblyben kellene megcsinálni. Az operációs rendszer és a gyári programok azt mutatják, hogy a feladatok nagy részénél a C a leghatékonyabb nyelv, és hozzátehetjük, nem csak az Amigán. Rendelkezik a legtöbb algoritmikus nyelveknél megszokott döntő tulajdonsággal (strukturáltság, könnyen kezelhető adatszerkezetek stb.), és kifinomult fordítóval rendkívül megközelíti az assembly sebességét. A hibrid programozással pedig a kritikus pontoknál betoldhatunk assembly részleteket.

Azoknak, akik mindezek ellenére ösztönösen kerülik a C-t — bízom benne, hogy kevesen vannak —, csak azt tudom mondani, „sajnos" az eredeti Commodore-dokumentáció is C-t használ az operációs rendszer ismertetésére, így hát kénytelenek némi ismeretet szerezni a C-ről. Ezzel együtt szeretnék megnyugtatni mindenkit: a C nyelv megismerése a megértéshez kellő szinten nagyon rövid idő. Ehhez magyar nyelven is több könyv nyújt segítséget. Akinek már van némi tájékozottsága a programnyelvek területén, annak ajánlom B.W. Kernigham — D.M. Ritchie: A C programozási nyelv c. könyvét.

A következőkben a speciális, Amigára vonatkozó ismeretekkel foglalkozom. A két legelterjedtebb C-implementáció a Lattice és az Aztec. Forrásszinten lényegében kompatibilisek, csak használatukban mutatkozik több különbség. Mindkét fordítóhoz készült keresztfejlesztő, amely lehetőséget nyújt arra, hogy PC-n készítsük el a tárgykódot, és mindkét fordítónak létezik PC-változata is. Az Aztec C egyik nagy előnye, hogy először assemblyre fordít. A Commodore cég a Lattice-compilert támogatja. Az Aztec C híveinek csak a fordítási folyamatot kell kiismerniük, a többi közölt ismeret használható az Aztecnél is. A linker használatát pedig minden fordítóprogramra épülő nyelvet használó programozónak ajánlatos megismernie.

Először szeretnék néhány hiedelmet eloszlatni. A Mikromagazin 1989/7. számában szerepelt, hogy az Amiga C programozása bonyolult. Ez kategorikusan nem igaz, mert ha valaki nem használja ki a speciális lehetőségeket, hanem a szabványos könyvtári rutinokra és a magas szintű szolgáltatásokra támaszkodik, „észre sem fogja venni", hogy a programja többfeladatos környezetben fut. Egy ilyen programban azonban nem igazán valósul meg az az alapkövetelmény, hogy a program könnyen használható legyen, amihez már hozzászoktak az Amiga-felhasználók.

Az igazi gond azonban nem az, hogy hogyan csináljuk, hanem az, hogy mit csináljunk — aranyszabály az Amiga programozásában. Szinte nincs olyan probléma, amelyre ne jutna eszébe az embernek két-három kivitelezhető, praktikus megoldás a lehetőségek dzsungelében. Ez a jellemző a programszerkezetben is érvényesül; ha egy tipikus Amiga program forráslistáját megnézzük — az első ijedtség után — megállapíthatjuk, hogy a program

túlnyomó része adatstruktúrákat iniciálizál, include fájlokat szerkeszt be. A B/K tevékenység, amelynek elfogadható kivitelezése más rendszerekben sok munkával jár, általában nagyrészt azt jelenti, hogy a program átpasszolja a struktúrákat, helyesebben azok címeit az operációs rendszernek.

Amíg végre (nem) működik a program...

Általában amikor elkezdünk egy gépet programozni, az első nehézségek még akkor jelentkeznek, miközben forráskódból megpróbálunk futtatható programot csinálni, ezek után általában kiderül, hogy nem működik a program, és kezdhetjük elölről. Az sem mindegy, hogy mivel készült a forrásszöveg. A programszerkesztők (editorok) között — rendkívüli képességeinél és gyorsaságánál fogva — abszolút favorit a CygnusEd Professional. Aki a használatának alapjaira negyedóra alatt nem jön rá, annak a programozáshoz általában nem is érdemes hozzáfognia. Kisebb javításakor megteszi a Ed vagy más egyszerűbb editor is, mivel a CygnusEd memóriafoglalása elég tekintélyes. Miután kiementettük a forráskódot egy .c kiterjesztésű fájlba, jöhet a fordítás. A Lattice C két menetben végzi a munkát: az LC1 előállít egy közbenső, .q kiterjesztésű fájlt, az LC2 ebből állítja elő a tárgykódot, .o kiterjesztéssel. Újabb verziók tartalmaznak egy LC néven futó programot, amely összefogja a program elkészítésének lépéseit. Az LC1 és LC2 estében alkalmazott opciókat a fájlnév előtt kell megadni. A -cw kivételével átlagos fordításnál nincs túlzottan szükségünk semelyik opcióra. Az opciók közül a fontosabbak, a 3.03-as verziót alapul véve, az alábbiak.

LC1

- b. A static és az extern tárolási osztályú adatokat az A5 vagy az A6 regiszterek segítségével éri el, ez áthelyezhető kód készítéséhez szükséges.

- c <karakterek>. Ennek az opciónak a használatával főként különböző kompatibilitási problémákat győzhetünk le, amelyek akkor jelentkeznek, ha más gépre készült forráskódot akarunk lefordítani. A karakterek a következők lehetnek:

c — lehetővé teszi megjegyzések egymásba skatulyázását;

d — használhatjuk a $ jelet az azonosítókban;

m — használhatunk többkarakteres karakterkonstansokat;

s — a fordító a megegyező sztringeket csak egyszer tárolja,

u — a char definíciókat unsigned char-nak veszi;

w — kikapcsolja annak a figyelmeztetésnek a küldését, amely akkor keletkezik, ha nem definiáljuk, hogy egy függvény int-tel tér vissza.

— d. A hibakereséshez szükséges információkat kapcsolja a kódhoz.

— d <sz>. Az <sz> szimbólum értékét fordítás közben állíthatjuk be.

— d <sz> =<x>. Az <sz> szimbólumnak az <x> értékét adja.

— i <út>. Meghatározza, hogy hol keresse a fordító az include fájlokat, alapesetben ez az INCLUDE: directory.

— 1. Minden adatot duplaszóhatárra illeszt.

— o <név>. Beállítja az eredményül kapott fájl nevét <név>.q-ra.

— p. A fordító nem fordítja le a forrásszöveget, csak az előfeldolgozó által szolgáltatott fájlt állítja elő. p kiterjesztéssel.

— x. Felszólítja a fordítót, hogy külső hivatkozásoknak (extern) ne foglaljon le helyet, mert egy másik forráskódban lesznek definiálva.

LC2

— o <név>. Beállítja az eredményül kapott fájl nevét <név>.o-ra.

— r. Az összes függvényhívás PC relatív lesz. Áthelyezhető kód készítéséhez szükséges.

— v. A fordító nem kapcsolja a tárgykódhoz a veremmutatót ellenőrző kódrészt, ennek hatására a program kb. 20 százalékkal gyorsabb lesz, és minden függvényhívásnál felszabadul 14 bájt. Ezt az opciót akkor ésszerű használni, ha már végeztünk a program belövésével.

A fordítás(ok) eredményeként kapott tárgykódo(ka)t a linker szerkeszti össze, így nemcsak több menetben készíthetjük el programunkat, hanem több nyelvet is használhatunk. Az AmigaDOS töltőprogramja lehetőséget ad overlay programszerkezet létrehozására, így ezt a linker is támogatja. Az overlay szerkezetű programoknál nem a teljes kód van a memóriában, hanem csak azok a részek, amelyekre éppen szükség van. Az egyes részek ki- és betöltögetését a program számára láthatatlan overlay supervisor végzi. Mivel overlay szerkezetre csak igen nagy programoknál van szükség, nem tárgyalom a kialakítás mikéntjét. Overlay technikát használ például a Deluxe Paint.

A linker fájlneve Alink, későbbi felülről kompatibilis verziókban Blink. Az Alink meghívásának formáját megnézhetjük az Alink? paranccsal. Ahol több fájl is megadható, az egyes fájlneveket pluszjellel vagy vesszővel kell elválasztani. Semelyik argumentumnál nincs kijelölve default út, tehát úgy kell őket megadni, mintha egy közönséges parancsban szerepelnének. A paraméterek jelentése:

FROM. A felhasználandó tárgykódok. A felhasznált tárgykódok mellett meg kell adni a LIB:c.o fájlt is.

TO. A készítendő fájl. Ha ezt az argumentumot nem adjuk meg, nem készül futtatható fájl.

WITH. A paraméterfájl, amelyet a parancssorban előforduló argumentumok megadására használhatunk. A paraméterfájlban minden sornak az itt felsorolt kulcsszavakkal kell kezdődnie (FROM, TO stb.). Ez után állhat a kívánt argumentum, és végül pontosvesszővel elválasztva egy megjegyzés. Ha egy argumentum mind a parancssorban, mind a paraméterfájlban szerepel, a parancssorban lévő az érvényes. A paraméterfájl alkalmazásával megmenekülünk a parancssor ismételt begépelésétől.

VER. Meghatározza, hogy hová küldje a linker az üzeneteit; ha ezt nem definiáljuk, az üzenetek a standard kimenetre kerülnek, amely általában az aktuális ablak.

LIB vagy LIBRARY. Meghatározza a felhasználandó scanned könyvtárakat (scanned libraries). A scanned (letapogatott) szó arra utal, hogy az ilyen típusú könyvtárakból azok a részek kerülnek a programfájlba, amelyekre hivatkozást talál a linker a tárgykódok valamelyikében. Az ilyen típusú könyvtáraknak a használat módjában semmi közük nincs az operációs rendszer rutinjait tartalmazó rezidens könyvtárakhoz (resident libraries). Az 1c.lib tartalmazza a szabványos C függvényeket (printf(), strcpy() stb.). Az amiga.lib pedig áthidalja azt a problémát, hogy a C a vermen keresztül adja át a függvényparamétereket, az operációs rendszer pedig a regisztereket használja. A fenti két könyvtárat mindig célszerű megadni a LIBRARY argumentumnál. A scanned könyvtárak a Lib: directoryban helyezkednek el.

Az amiga.lib-bel kapcsolatban fontos szólni a bázismutatókról. A könyvtárak eléréséhez használt bázismutatókat mindig a megadott névvel (IntuitionBase, GfxBase stb.), függvényen kívül kell definiálni, mivel ezekre hivatkozások vannak az amiga.lib-ben. Ennek kapcsán megjegyzem, hogy a kis- és nagybetűket a C-ben nem lehet szabadon helyettesíteni egymással.

MAP. Meghatározza a map fájl nevét, amely tájékoztatja a programozót az egyes programrészekben definiált szimbólumokról. A map normál ASCII szövegfájl. Ha nem adjuk meg a MAP paramétert, nem készül ilyen fájl.

XREF. Hasonló a map fájlhoz, azzal a különbséggel, hogy minden egyes programrészhez felsorolja azokat a külső szimbólumokat, amelyekre hivatkozunk.

WIDTH. A sorhosszúság megadására szolgál, amelyet a map és xref fájl elkészítésénél használ a linker.

Az összeszerkesztés során különböző jelzéseket kaphatunk. A linker leggyakrabban azért ad hibajelzést, mert egy szimbólumot vagy kétszer, vagy egyszer sem definiáltunk, illetve nem pontosan adtuk meg a parancssort. Ha egyéb üzenettel találkozunk, az sajnos azt jelzi, hogy a fordító hibás tárgykódot generált — feltéve, hogy valóban tárgykódokat adtunk meg paraméterként.

IDEAMENT

A TASWORD THREE program karakterkészletének módosítása

A Spectrum-használók körében több szövegszerkesztő program ismert. Előnyeikről, hátrányaikról nem sokat érdemes beszélni, hiszen „minden cigány a maga lovát dicséri". Én a fenti programot használom teljes megelégedettséggel. Mindent tud, amire szükségem van. Még többet is! Minden szolgáltatását ki sem tudom használni.

Nézzük először a programot magát. A TASWORD III csak magnetofonnal nem működik, mivel egyszerre nem fér az összes programblokk és a szövegfájl a memóriába. Eredetileg microdrive-ra írták, de láttam már floppys változatot is. Nézzük a blokkokat:

TASCODE 1	47847,10239
TASCODE 2	27392,10239
TASCODE 3	23296,256
TASCTRL	25000,2391
TASTABLE	37888,6656

Induláskor a TASCODE 2-t nem tölti be, és szövegszerkesztő üzemmódban jelentkezik be. Az egyéb funkciókhoz egy STOP utasítással töltjük be a TASCODE 2 blokkot. Innen BASIC-be is kilépheténk. Ilyenkor írható be a nyomtatóhoz a csatorna („b"!) megnyitása. A visszatérés RUN. A parancsmódból Return utasítással töltetjük be a TASCODE 1-et, és léphetünk vissza a szövegszerkesztő módba.

Szolgáltatások

A szövegmező maximálisan 128 karakter hosszú sorokból állhat. A margó tetszőlegesen állítható be. A képernyő vagy 64, vagy 32 karaktert jeleníthet meg egy sorban. A sorok száma a TASWORD II-vel ellentétben tetszőleges, amíg a memória be nem telik, ami hozzávetőlegesen 15 k üres helyet még kijelez ugyan ilyenkor, de ha nem hagyunk kb. 2 k-t szabadon, akkor a szövegszerkesztő funkciók nem működnek, „File full" üzenetet kapunk.

A program a normál és grafikus (nyomtatóvezérlő) karaktereken kívül egy „második" karakterkészletet is tartalmaz, a space-től m-ig. Ez a 77 karakter magában foglal szinte mindent, amire egy európai nyelvet használónak szüksége lehet. Lehetőség van a nyomtatóvezérlő és a második karakterkészlet help információinak átírására, így a „mankó" oldal csak a számunkra szükséges információt jeleníti meg.

A kurzor a más szövegszerkesztőknél megszokott módon betűnként, szavanként, bekezdésenként, oldalanként mozgatható minden irányban.

A szerkesztés történhet nyomdai tükör formájában vagy anélkül, akár bekezdésenként vagy soronként utólag, ami javítás esetén segítség. Sorokat lehet jobbra, balra mozgatni, középre helyezni, szétvágni. Blokkokat lehet áthelyezni, egyszer vagy többször másolni, törölni.

Van szövegkereső üzemmód, melynek hibája, hogy a grafikus karaktereket és a második karakterkészlet betűit nem fogadja el, így alig használható.

Tabulátora tetszőlegesen állítható. A sor két végét automatikusan jelzi, ezenkívül 19 tabulálási pont állítható be.

A kisbetűkből nagyot, illetve a nagyokból kicsit tud csinálni, de sajnos a második karakterkészlet betűivel ez nem működik, ezért az ékezetes betűket egyenként kell átírni.

Szükség esetén jelölhetők az oldalak egy szaggatott vonallal, és van egy Form feed karakter, amely nyomtatáskor lapdobást eredményez.

A parancsmód szolgáltatásai több részletben állnak rendelkezésre.

Az első menü

— Nyomtatás
— Nyomtatás microdrive-on levő adatbázis segítségével
— Szöveg kimentése microdrive-kazettára, magnószalagra
— Szöveg betöltése microdrive-kazettáról, magnószalagról
— Szöveg hozzátöltése microdrive-kazettáról, magnószalagról
— Visszatérés a szöveghez
— Programmódosítás
— A TASWORD III kimentése
— Szöveg törlése microdrive-kazettáról
— Kilépés BASIC-be

```
  10 REM A DEMO kazettan levo karaktertervezo program modositasa
                dr.GRESZ MIKLOS 1989
  20 INPUT "a file cime:";b
  30 INPUT "a file hossza:";l
  40 INPUT "a karakterek cime:";x: LET Q=256*INT (x/256): LET W=x-Q
 620 LET r$="a": GO TO 1400: REM GO TO 7300
1301 REM
1304 REM
1310 REM
1311 REM
1312 REM
4410 INPUT "hova? ASCII ";ASCII
4412 PRINT AT 21,0;ASCII
4414 LET CIM=Q+(ASCII-32)*8+W
4430 POKE CIM+i,r(i+1): REM POKE USR r$+i,r(i+1)
6000 REM
6030 FOR i=32 TO 63
6070 FOR i=32 TO 63
6080 POKE 23607,(Q/256-1): POKE 23606,W: PRINT CHR$ i;
6095 POKE 23606,0: POKE 23607,60: PRINT
7020 REM
7025 REM
7030 REM
7035 REM
7040 REM
7042 REM
7050 REM
7060 REM
7065 GO SUB 6200: PRINT "Ird be a kimentendo file nevet!"
7220 REM
7230 REM
7240 REM
7302 INPUT "Honnan? ASCII ";ASCII
7304 PRINT AT 21,0;ASCII
7306 LET CIM=Q+(ASCII-32)*8+W
7320 LET b=PEEK (CIM+y)
8000 REM
8010 REM
8020 REM
8040 REM
9000 CLEAR : SAVE "character" LINE 1
9010 REM
9100 REM
9110 REM
9120 REM
9130 REM
9140 REM
9150 REM
9160 REM
9400 REM
9500 REM
9510 STOP
```

Nyomtatóüzemmódok

— Első sor
— Utolsó sor
— Másolatok száma
— Sorköz
— Laponként/folyamatosan
— Fejléc nyomtatása
— Lábjegyzet-nyomtatása
— Oldalszámozás
— Bal margó

} Minden oldalra ugyanazt a beírt sort nyomtatja

Az egyszerre a memóriába bele nem férő szövegek címeit egy $ karakter után megadva, mindegyiket kinyomtatja egyetlen print utasítással. Akár különböző microdrive-okról. Megfelelően definiált adatokat — akár a MASTERFILE segédprogrammal átalakítva, amit szintén megtalálunk az eredeti kazettán — felhasznál nyomtatáskor. Például egy levelet 20 címre egyszer kell megírni, ha a címjegyzék megfelelő formában rendelkezésre áll. A program 20-szor a 20 különböző címmel tudja kiírni. Megfelelő utasítással a leveleket sorszámozza, sőt megadott paraméterek alapján (például irányítószám, megadott betű) válogat a címlistáról, és csak a feltételeket teljesítő címekre nyomtatja ki a levelet.

Programmódosítás

Oldalformátum: sorok száma
fejléc sorainak a száma
lábjegyzet sorainak a száma
nyomtatóvezérlő kódok: CR, LF, FF

A normál karakterek nyomtatóvezérlő kódjai (max. 3 kód).

A nyomtatóvezérlő karakterek (a 16 grafikus karakter + A—P, max. 5 kód).

A 2. karakterkészlet nyomtatóvezérlő kódjai, max. 3 kód.

A program paramétereinek megváltoztatása:

Tintaszín 1
Tintaszín 2
Papírszín
Keretszín 1
Keretszín 2
Kurzor formája
Bal margó
Jobb margó
Billentyűhang

Látható, hogy a program többet tud, mint amennyit használni lehet. De van egy-két hiányossága. Most egyről szólok, mivel ez tűnt számomra a legzavaróbbnak. A nyomtatónak küldendő kód minden betűnél megváltoztatható. Ez így szinte tökéletes.

Szinte. Mert mi van akkor, ha a nyomtatóm tud olyan karaktert, amelyet a program nem? Például az enyém ismer néhány görög betűt, tud táblázatot szerkeszteni. Ezek nem szerepelnek a második karakterkészletben. Tehát lássak

```
  10 REM Karaktertervezo program Modositotta dr.GRESZ MIKLOS 1989
  20 INPUT "a file cime:";b
  30 INPUT "a file hossza:";l
  40 INPUT "a karakterek cime:";x: LET Q=256*INT (x/256): LET W=x-Q
  50 BORDER 6: PAPER 6: INK 0: CLS
  90 DIM r(8)
 100 DIM c(4)
 110 DATA 5,8,6,7
 120 FOR i=1 TO 4: READ c(i): NEXT i
 500 LET xpo=4: LET ypo=12
 600 GO SUB 8100
 620 LET r$="a": GO TO 1400: REM GO TO 7300
1000 GO SUB 8100
1170 PRINT "Use the arrow keys with or      without CAPS SHIFT to move the  flas
hing cursor about the grid."
1172 PRINT "The cursor square is filled-in  when CAPS SHIFT is pressed and  is c
leared otherwise."
1174 PRINT "Press Q to quit this stage."
1200 LET x=-1: LET y=-1
1210 OVER 1
1220 GO SUB 5200
1224 PRINT AT yp,xp; FLASH 1;" "
1230 GO SUB 4000
1240 OVER 0
1250 GO SUB 4200
1300 GO SUB 6200
1320 GO SUB 4400
1330 GO SUB 6000
1340 GO SUB 6200
1400 PRINT FLASH 1;"Press:"
1410 PRINT "Q to quit this program"
1420 PRINT "S to save characters on tape"
1430 PRINT "A to alter the pattern"
1440 PRINT "C to clear grid and start again"
1450 PRINT "P to pick up a character"
1500 POKE 23560,0
1505 LET k$=CHR$ PEEK 23560
1510 IF k$="q" THEN GO TO 9500
1520 IF k$="s" THEN GO TO 7000
1530 IF k$="a" THEN GO SUB 6200: GO TO 1170
1540 IF k$="c" THEN GO TO 1000
1550 IF k$="p" THEN GO TO 7200
1590 GO TO 1505
4000 REM Design character
4005 PAPER 2
4010 GO SUB 5300
4011 IF m=1 THEN GO TO 4100
4012 IF (x+1)*(y+1)*(x-8)*(y-8)=0 THEN LET m=7
4015 PRINT AT yp,xp; PAPER m;" "
4020 GO SUB 5400
4030 GO SUB 5200
4035 PRINT AT yp,xp; FLASH 1;" "
4040 GO TO 4010
4100 PRINT PAPER 7;AT yp,xp; FLASH 0;" "
4150 PAPER 6: RETURN
4200 REM Set r(8)
4210 LET y=ypo-8
4220 FOR j=1 TO 8
4230 LET sum=0: LET x=xpo
4250 FOR i=1 TO 8
4260 LET sum=2*sum
4270 LET p=INT (ATTR (y,x)/8)
4275 IF p=0 THEN LET sum=sum+1
4280 LET x=x+1
4290 NEXT i
4300 LET r(j)=sum
4310 LET y=y+1
4320 NEXT j
4340 RETURN
4400 REM Put r(8) into usr r$
4410 INPUT "hova? ASCII ";ASCII
4412 PRINT AT 21,0;ASCII
4414 LET CIM=Q+(ASCII-32)*8+W
4420 FOR i=0 TO 7
4430 POKE CIM+i,r(i+1): REM POKE USR r$+i,r(i+1)
4440 NEXT i
4450 RETURN
5000 REM Draw grid
5010 PAPER 7: FOR j=ypo-9 TO ypo: PRINT AT j,xpo-1;"          ": NEXT j: PAPER 6
5020 LET xo=8*xpo: LET yo=8*(22-ypo)
5050 FOR j=0 TO 64 STEP 8
5060 PLOT xo,yo+j: DRAW 64,0
5070 NEXT j
5080 FOR j=0 TO 64 STEP 8
5090 PLOT xo+j,yo: DRAW 0,64
5100 NEXT j
5110 RETURN
5200 REM Screen coords from x,y
5210 LET xp=xpo+x
5220 LET yp=ypo-8+y
5230 RETURN
5300 REM Poll keyboard for cursor control keys
5301 LET key=PEEK 23560: POKE 23560,0
5302 LET m=1
5304 IF key=CODE "q" THEN RETURN
5310 LET m=7
5320 LET c=key-48
5330 IF c>=5 AND c<=8 THEN RETURN
5340 LET m=0
5345 LET c=c+41
5350 IF c<1 OR c>4 THEN GO TO 5300
```

mást a képernyőn, mint amit a nyomtatóm ki fog írni? Meg lehet szokni, de nem igazán jó. Vagyis jöhet a dolog lényege! (Ha pedig még nem írtam volna már több százezer karaktert a programmal, akkor némelyik magyar ékezetes betűt is áthelyezném. De már megszoktam így. Ez a nyomtató viszont új, ezért kerestem a megoldást.)

A TASWORD THREE több programrészből áll. Ezek között van a TASTABLE nevű is. A karakterek ebben találhatók. Mégpedig három készlet. A sima, normál karakterek a ROM-ban vannak, tehát kell a 64 karakteres képernyőhöz egy készlet és a második készlethez a normál és a „fél" karakterek. Az adatok:

TASTABLE 37888,6656

sima 1/2:1 készlet 38400

második 1/2:1 készlet 39424

második 1:1 készlet 40704

Tehát a saját készlet kialakításához az utóbbi kettőhöz kell hozzányúlni. Erre a célra kiválóan alkalmas a demo kazettán található Charactergenerator program, némi módosítás után. Ha a mellékelt lista szerint átalakítjuk a programot, bármilyen egyedi tervezésű karaktert beírhatunk a készletbe. Természetesen a megfelelő kezdőcím átírásával (a 40-es programsorban) a ROM-ban bárhol elhelyezkedő karakterkészlet módosítható. A kezdőcímet többször át kell most is írnunk, mivel az egyszerűség kedvéért egyszerre csak 32 karaktert látunk és módosíthatunk.

Gresz Miklós

```
5360 LET c=c(c)
5390 RETURN
5400 REM Change x,y as a function of c
5410 IF c<>5 THEN GO TO 5430
5420 IF x=-1 THEN RETURN
5425 LET x=x-1: RETURN
5430 IF c<>8 THEN GO TO 5450
5440 IF x=8 THEN RETURN
5445 LET x=x+1: RETURN
5450 IF c<>6 THEN GO TO 5470
5460 IF y=8 THEN RETURN
5465 LET y=y+1: RETURN
5470 IF y=-1 THEN RETURN
5480 LET y=y-1: RETURN
6000 REM
6010 INK 2
6020 PRINT AT 0,0;
6030 FOR i=32 TO 63
6040 PRINT CHR$ i;
6050 NEXT i
6060 INK 0: PRINT
6070 FOR i=32 TO 63
6080 POKE 23607,(Q/256-1): POKE 23606,W: PRINT CHR$ i;
6090 NEXT i
6095 POKE 23606,0: POKE 23607,60: PRINT
6096 RETURN
6200 REM Clear lower screen
6210 PRINT AT ypo,0
6220 FOR y=ypo TO 20
6230 PRINT OVER 0;"                                "
6240 NEXT y
6245 PRINT AT ypo+3,0;
6250 RETURN
7000 REM Save on tape
7010 GO SUB 6200
7065 GO SUB 6200: PRINT "Ird be a kimentendo file nevet!"
7066 INPUT LINE r$
7070 SAVE r$CODE b,l
7090 GO TO 1340
7200 REM Pick up character
7210 GO SUB 6200
7300 OVER 1
7302 INPUT "Honnan? ASCII ";ASCII
7304 PRINT AT 21,0;ASCII
7306 LET CIM=Q+(ASCII-32)*8+W
7310 FOR y=0 TO 7
7320 LET b=PEEK (CIM+y)
7330 LET s=128
7400 FOR x=0 TO 7
7410 LET t=INT (b/s)
7420 GO SUB 5200: PRINT AT yp,xp; PAPER 7*NOT t;" "
7430 LET b=b-t*s
7440 LET s=s/2
7470 NEXT x
7480 NEXT y
7490 OVER 0: GO TO 1340
8100 OVER 0: GO SUB 6000: GO SUB 5000: OVER 1: GO TO 6200
9000 CLEAR : SAVE "character" LINE 1
9510 STOP
```

Mikroelektronika, mikroszámítógépek

Nyitott rendszerű felsőfokú képzés

A mikroelektronika tudományának gyakorlati alkalmazása egyre nagyobb szerepet kap mindennapi életünkben. Nemzetközi tapasztalatok előrevetítésével más most állíthatjuk, hogy a századfordulóig a mikroszámítógépet szinte minden állampolgár használni fogja.

Minden felelős munkakört betölteni szándékozó vagy éppen betöltő szakembernek egyre inkább szüksége van arra, hogy saját szakmai ismeretei mellett a mikroelektronika eredményeit is alkalmazni tudja.

Az egyén és a társadalom szempontjából egyaránt hasznos és viharos gyorsaságú változásoknak vagyunk tanúi. A korszerű kutató, fejlesztő, gyártó eszközökben és magukban a gyártott termékekben: az ipar, a mezőgazdaság, az orvosi elektronika, az oktatás, a nyelvtanulás, szinte valamennyi tudományos kutatás, az adminisztráció stb. különböző területein napról napra követhetően bővül a mikroszámítógépek alkalmazása.

Tizenkilenc hazai egyetem/főiskola az LSI Oktatóközpont (1033 Budapest, Hévízi út 6/E, telefon: 180-5712) működésével most indítja azt a nyitott rendszerű felsőfokú képzést, amely az egyéni érdeklődési körnek megfelelően lehetőséget nyújt a mikroelektronikát, mikroszámítógépet alkalmazó mérnök, üzemmérnök, tanár, mezőgazdasági gépészmérnök, orvos stb. képzésére. Felvételi vizsga nincs, de a követelmény a kiadandó diploma szintjének megfelelő.

A hallgató egyéni tanulással sajátíthatja el az ismereteket; a módszertani segítséget nyújtó tankönyvek sora már megjelent, illetve folyamatosan megjelenik. Az egyes tantárgyak tananyagrészeinek videoszalagra vett előadásait többször megnézhetik — ezeket a budapesti, illetve vidéki konzultációs központoktól kölcsönözni is lehet — igénybe vehetők a 3-5 napos vizsgaelőkészítő tanfolyamok, és még tovább folytathatnánk.

Aki a jövőbe néz, aki előre akar lépni, aki úgy érzi, hogy a jövő technikáját már most érdemes elsajátítani, azok számára ezen a szakterületen a képzés kapuit nagy igyekezettel próbáljuk kitárni.

Németh István
LSI ATSZ

A Solarsoft kínálatából

Mint arról már 1989/11-es számunkban hírt adtunk, új szoftverkategória jelentkezett a magyar piacon is: a shareware. A Cédrus Kisszövetkezet Magyarországon elsőként vállalkozott arra, hogy a vékony pénztárcájú, PC-hez, PC-kompatibilis géphez csak alkalomszerűen hozzájutó, tanulni, illetve ötletet meríteni kívánó számára is megfelelő áru szoftvertermékeket forgalmazzon. Úgy véljük, olvasóink között is sokan akadnak, akik csak ezt a szoftvert tudják megfizetni, ezért határoztuk el, hogy rendszeresen közzétesszük a választást, eligazodást megkönnyítő Solarsoft-katalógus információit.

NÉV:
QBTOOLS
SZERZŐ:
Többek, 1986.
LEÍRÁS:
Kiegészítések, szubrutinok a QBASIC 2.0 használatához.
— BASDELUX. LIB — 8 funkció
— CHKQB — display-teszt
— TSTAVAIL — lemezen levő szabad hely tesztelése
— MASTER 1. és 2. — szubrutingyűjtemények
— QBX206 — QB 2.0 keresztreferencia v. 2.06
— TUTOR-QB — útmutatók a QB 2.0 használatához
— BASIC példaprogramok
DOKUMENTÁCIÓ:
Felkommentározott forrásprogramok.
KONFIGURÁCIÓ: —

NÉV:
BASWIND and WINDOW TOOLS
SZERZŐ:
Dave Evers, USA, 1988.
LEÍRÁS:
Ablaktechnika a QBASIC 4.0 használatához. A lemezről hiányzik a BW4DEMO.ARC és a BWTOOLS.ARC.
— BASWIND4.QLB
— BASWIND4. LIB
— BASWIND4.EXE
—WNDTOOL4.LIB
A ∗.BAS és a ∗ SUB fájlok CSAK ezek használatát mutatják be! A BYTE magazinból írták ki a FASTPRT.ASM-et (Fast printing modul — 1987) és a NEWSCRN.ASM-et (Screen Save and Restore).
DOKUMENTÁCIÓ:
Forrásszövegek kommentározva.
KONFIGURÁCIÓ: —

NÉV:
Batch File Utilities és Extended Batch Language v. 3.09, 1988.
SZERZŐ:
Public Software Library, USA, 1983. és Frank Canova, Seaware Corp., Delray Beach, USA, 1988.
LEÍRÁS:
A Batch File Utilities a DOS 2.0-tól felfelé készült batch fájlokból áll. Kisegítő fájlok, hívott .ASM forrásprogramok, .INC, .REM kódok és .COM-ok (lefordított .ASM-ek).
Az Extended Batch Language a DOS korszerű kiterjesztése. A BATFUNC1.COM program az első indítás után rezidens marad. A BAT.COM csak egy behívó.
Néhány lehetőség:
— sajátmenü-készítés
— DOS-shell
— auto run
— sztring, operátorok, lebegőpontos ábrázolás alkalmazása
— elérési utak, rendszerkörnyezet
(beep, call, cls, color, exit, return, shell, stack, type, skip, trace, on error, resume, seek, locate stb., továbbá: dos vars, global vars, default drive, environment vars)
DOKUMENTÁCIÓ:
Részletes felhasználói tudnivalók a BATDOC, BATDOC2 állományokban találhatók, amelyek egyben részletes HELP-ek is.
KONFIGURÁCIÓ:
—
FONTOS:
A programot nyomatékosan ajánljuk!

NÉV::
MODULA 2 — Fitted Modula Compiler v. 1.3
SZERZŐ:
Fitted Software Tools, USA, 1988.
LEÍRÁS:
A Modula-2 nyelv a Pascalhoz hasonlít. A szoftverhez nem adják az ED.EXE szövegszerkesztőt és az M2COMP.EXE fordítót!
Az EXEFILES.ARC kibontásakor kapjuk meg a többi .EXE fájlt (MC.EXE, M2LINK.EXE, EDCONFIG.EXE, GENMAKE.EXE, DBG2MAP.EXE)
A HUGE.ARC és a LARGE.ARC tartalmazza az ∗.M20 típusú ∗ .LIB object-eket; a DEFFILES.ARC a ∗.DEF típusú definíciós fájlokat, a TALK.ARC pedig a ∗.MOD típusú .LIB forrás-modulokat.
DOKUMENTÁCIÓ:
Az FMODULA2.DOC teljes leírást ad a programok használatáról, installásáról. A nyelv szintaxisáról semmit nem írnak.
KONFIGURÁCIÓ:
A programok winchestert vagy 720 kbájtos, illetve 1,2 Mbájt kapacitású floppyt igényelnek.

NÉV:
MS QuickBasic Tools
SZERZŐ:
Több cég és személy (összeállító: Computer Solution GmbH.)
LEÍRÁS:
A Microsoft QuickBasic-hez írt fejlesztést támogató program- és szubrutingyűjtemény.
MYED 2.02 — szövegszerkesztő
EGA Utility — segédprogramok EGA-monitorokhoz
ADVBA99 — kibővített funkciókönyvtár
QBWARE — interfészkönyvtár
QBTOOLS — kiegészítések, szubrutinok
FONTOS:
A részletesebb leírás a különálló lemezeknél található meg.

NÉV:
A/86 Macro Assembler és D/86 Debugger v. 3.18
SZERZŐ:
Eric Isaacson, Bloomington, USA, 1988.
LEÍRÁS:
Az assemblerben írt programok fordítása az A/86-tal. Szerkesztéshez LINKER nincs! (A DOS LINK-re hivatkozik!)
XREF — keresztreferencia-készítő program
EXMAC — makrokiterjesztést kezelő program
Az assembly nyelv, illetve a lefordított programok vizsgálatához nyújt segítséget a debugger program (monokróm!).
A beépített HELP az F10 gombbal bármikor hívható.
DOKUMENTÁCIÓ:
Teljes használati útmutató az A00.DOC.—A17.DOC és a D00.DOC—D10.DOC fájlokban.
KONFIGURÁCIÓ: —

NÉV:
DOSMENUE v. 1.2
SZERZŐ:
Georg Huonker, 1987.
LEÍRÁS:
Egyszerű menürendszer. 30 parancs hajtatható végre. 15 batch program és 15 DOS-parancs választható ki, melyek editálhatók a felhasználó igényeinek megfelelően.
Megkíméli a felhasználót a hosszú, több paraméteres parancssorok beírásától, segítségével egyszerűen indíthatók kisebb batch programok.
HASONLÓ: AUTOMENU.
DOKUMENTÁCIÓ: Németül.
KONFIGURÁCIÓ: —

ADOK—VESZEK—CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: kereskedelmi tevékenységet folytatóknak gépelt soronként (60 karakter) 100,- Ft, másoknak az első sor 50,- Ft, minden további sor 20,- Ft. Kérjük, hogy a hirdetés díját az NJSZT 1054 Bp., Báthori u. 16. címére fizessék be, rózsaszínű postautalványon (jelölve, hogy apróhirdetés), a befizetést igazoló szelvényt pedig csatolják a hirdetéshez. Hirdetéseiket a szerkesztőség címére várjuk (1371 Bp., Pf.: 433). Az NJSZT tagjai továbbra is kedvezményesen hirdethetnek (az első sor ingyenes), de kérjük, hogy adják meg tagsági számukat. Azokat a hirdetéseket, amelyek a hónap első napjáig megérkeznek, már a hónap végén viszontláthatják lapunkban.

Adok

Eladó: **Amiga 500**-as számítógép, **Commodore 1541-II** típusú drive, vagy IBM részegységekre cserélem. Ajánlatokat levélben kérem a következő címre: Fehér Emil, Budapest, Késmárki utca 10. 1118

Amiga programok és 3.5", 5.25" NoName DSDD lemezek 10 db 1250/390 Ft-os áron eladók. Keresztes Gábor, Budapest, Laky köz 11. 1142. Tel.: 164-3452

AAAAMIGA!! Törjük le a pénzért másolókat! Legújabb nyugati törésű Amiga programjainkat a postai költségek megtérítése mellett bárkinek szívesen átmásoljuk! Válaszboríték ellenében 4-5 napon belül listát küldünk! Cím: Amiga Barátok Klubja, Sopron, Pf.: 308, 9401. Új tagok jelentkezését szívesen fogadjuk!

Amiga 500 számítógéphez szabvány (720 k) floppy illesztését A/D átalakító építését, MIDI interfész készítését vállalom. Ürmössy Zoltán, Edelény, Bódva u. 20. 3780

Amigások! ADOK-VESZEK-CSERÉLEK! A legfrissebb törésű programok! (pl. SPACE-ACE). Kódokkal, leírásokkal! Listát küldök és várok! Takács Gábor, Nyíregyháza, Új u. 11. 4400

Amiga programokat adunk és cserélünk. A legújabb programok! Olcsón és gyorsan. Válaszborítékért listát küldünk. AMIGA GBM, Szekszárd, Pf.: 259, 7101

Az országban a legolcsóbb! **Amiga 500** programok 40 Ft-ért. Magyar nyelvű Amiga szakkönyvek megrendelhatők! Ifj. Haár László, Budapest, Dráva u. 11. VII.30. 1133. Tel.: 1732-008

Atari 800XL, 1050 drive és több száz program (100 disk) egy joystick-al eladó. Tel.: 114-7151

Atari 800XL, magnóval, joystickkal, sok programmal, bő szakirodalommal eladó. Irányár 14 000 Ft. Szíjártó Zoltán, Pécs, Gyenis L. u. 9. 7632

C16-os, **C Plus/4**-es programok nagy választékban eladók (12 Ft/db) és cserélendők. Kérésre listát küldök. Domján Viktor, Kecskemét, Mikszáth Kálmán körút 16. 6000

C Plus/4, magnó, 1541/II-es floppy, 2 db joystick, 30 db lemez, programok és szakkönyvek eladók. Irányár: 30 000 Ft. Egri László, Békésszentandrás, Pacsirta u. 6. 5561

új **C Plus/4** tartozékokkal olcsón eladó. Tel.: 1-352-894

C Plus/4, magnó, 1 db joystick, 250 program, szakkönyvek eladók 10 000 Ft-ért. Gyalog László, Budapest, Kosárfonó u. 10. 1191

C Plus/4-es programok nagyon olcsón (8 Ft/db) eladók. Válaszborítékért listát küldök. Pápes Zoltán, Maglód, Ady E. u. 14. 2234

C64 játékprogramokat eladok(25 Ft/db). Válaszborítékért listát küldök. (Adathordozó szükséges.) Személyesen is. Vári Tamás, Monor, Ady E. u. 9. IV/2. 2200

Commodore 64, datasett, 1541-es floppy, lemezek, kazetták programokkal, sok kiegészítő és irodalom eladó. Tel.: 1159-507 este és szopmbat délután.

C64 floppy drájvval 21 000 Ft, 15 darabos 2100 Ft névértékű könyvtár 1500 Ft, 16 db SpectrumV. 480 Ft, 50 lemez színvonalas, új játék 4500 Ft, 25 lemez felhasználói, szintén minőségiek, 2200 Ft. A programok adatai adatbázisokban, nyomtatásban. Várhegyi István, Nyírbátor, Derzsi u. 31. 4300

Commodore 64-es színvonalas programok eladók kazettára! 4 és 7 Ft/db. 700 db programról listát küldök! Bohács Tibor, Nagykálló, Petőfi út 8. 3420. Tel.: Nagykálló 346.

C64-re POOL OF RADIANCE eladó (8 oldal), 100 Ft. Herczeg Ferenc, Újszász, Wesselényi út 1/A. 5052

Szintetizáló program **Commodore 64**-re! Dob + 2 szólam, dallammemória és egyéb lehetőségek. Zenei képzettséget nem igényel. Kívánságra díjmentes tájékoztatót küldök. A program ára (leírással) kazettán 449, lemezen 499 Ft utánvéttel. Erdélyi Zsolt, Sződ, Dózsa Gy. út 261. 2134

C64-hez Turbo cartridgek eladók. Eprom égetést vállalok 2716-27256-ig. Borbás Dénes, Borsodszentgyörgy, Főút 10. 3623

C64-hez 1764-es RAM-bővítő (256 k-s) modul 11 000 Ft-ért eladó. Mészáros Tamás, Miskolc, Kondor B. u. 18. 3535

4 éves **C64** floppyval 30 db lemez, 2 joystick, szakkönyvek: 30 000 Ft. Habony Zsolt, Miskolc, Szepesi u. 8. 3525. Tel.: (46) 77-131

C64 Datasett másolómodul 500 Ft-ért. Válaszborítékért grafikus tájékoztató! Tóth Lajos, Székesfehérvár, Vöröshadsereg út 82. 8000

C64-re felhasználói- és játékprogramok eladók (15 Ft/db) lemezre és kazettára, vagy lemezzel együtt (90 Ft/lemez). Pankász Viktor, Pécs, 39-es dandár u. 5/b. 7633

+++ COMMODORE /!-ESEK FIGYELEM !!!+++ C64-re 2-3 hetes szuper szinvonalú nyugati DEMOK-JÁTÉKOK igen olcsón kaphatók!!! Farkas Gábor, Mezőnyárád, Lenin út 140. 3421. Tel.: Mezőnyárád II. Üdvözlettel: TJC.-TLU.!!! WRITE TO!

C128, C64, C Plus/4, C16-ra a legújabb 1989-es programok eladók. Keresztfalvi János, Budapest, Doberdó út 4. 1034. kérésre listát küldök!

Commodore 128D eladó. Három üzemmód (C128, C64, CP/M). Beépített 1571-es lemezegységgel. Két darab joystickkal, 30 db szakkönyvvel 30 000 Ft-ért. Egy 128 K-s bővítő (RAM) 5000 Ft. Nyolcvan db DS/DD DISK, rajta kétötöd rész felhasználói (több leírással), háromötöde játékprogramokkal. Vegyesen a három üzemmódhoz. Csak a lemezek árát kérem, ami 8000 Ft!!
Egy MICROVOX-64 magyarul beszélő bővítő 6500 Ft. Commodore 1084 szines (szuper) monitor, 40/80 oszlopos kijelzéssel, 14"-os méretbe, CGA-felbontás, CVBS/RGB és EURO csatlakozókkal 30 000 Ft. Egyben történő vásárlásnál engedménnyel. Megegyezhetünk. Fődi József, Szeged, Kállay Albert u. 6-10. 6720

C128 D (Beépített VC 1571) eladó. Ára 38 000 Ft + ajándékok: 1 db joystick, 10 db lemez programokkal, szakirodalom. Keresem: Graphic Booster 128 programot. Rácz Krisztina, Budapest, Tenkes u. 7/a. 4.em.15. 1225. Tel.: 1-181-200/1315 mellék.

Enterprise programokat eladok, cserélek (10-20 Ft), kb. 600 db van. Fazekas Dániel, Komló, Viola u. 31. 7300

Enterprise programok olcsón eladók, de veszek és cserélek is. Eladó egy 48 k-s ZX-Spectrum tanítható joystick-illesztővel, szakkönyvekkel, interfésszel és 300 programmal. Sváb László, Budapest, Pára u. 8. 1108. Tel.: 177-4243

Enterprise-osok figyelem!
Több száz program rendkívül olcsón eladó. Tetszése, kívánsága szerint választhat a régebbi és a legújabb játék- és felhasználói programok közül. Listát adok (válaszboríték ellenében, de személyesen is). Üzenetet hagyni, programot rendelni lehet a 1848-899-es telefonszámon, Tóth Gusztáv névre. Cím: Tóth Gusztáv, Budapest, XV. ker. Nádastópark 32. 1152. Cserélek is.

Enterprise 128 bő irodalommal + joystickkal + 17 kazetta minőségi programmal + magnóval 13 000 Ft-ért eladó. Tel.: (62) 29-491.

Enterprise 128 + magnó + programok + könyvek; Philips monochrome monitor; JVC GR-A30E video kamkorder eladó. Tel.: 48/17-495

Enterprise programok eladók. 10-50 Ft. Válaszborítékért listát küldök. Zemen László, Budapest, Kada u. 141. fsz. 9. 1104

Enterprise programokat olcsón eladok ill. cserepartnereket keresek. Minden levélre válaszolok. Czibere Lajos, Debrecen, Hámán Kató u. 1. III/12. 4027

Enterprise prg-ok nagy választékban lemezen, kazettán, garantáltan jó minőségben a legolcsóbb áron eladók. Csere programok is érdekelnek. Jeligém: GYORSASÁG, MINŐSÉG. Válaszborítékért listát küldök. Barabás Barbara, Bonyhád 3, Pf.: 25. 7153

Enterprise programok eladók! 3.5" lemezre is! Válaszborítékért listát küldök. Gyurta Viktor, Budapest, Szövőszék u. 14. 1108. Tel.: 177-5695

Enterprise programok eladók 10-50 Ft. Válaszborítékért listát küldök. Érdeklődés kizárólag levélben. Leleszné, Budapest, Delej u. 51. XV. 1h. 25. 1089

Enterprise programokat adok és cserélek. A legújabb programok olcsón! Válaszborítékért listát küldök. Sándor József, Bonyhád, Bezerédi út 41. 7150

Enterprise programokat adok-veszek, megfelelő partner esetén cserélek. Csere esetén listát, vásárlás esetén válaszborítékot kérek! Ifj. Sala István, Budapest, Hársfa sétány 17. V.30. 1203

Enterprise eredeti EX-DOS lemezvezérlő eladó. Ár: 10 000 Ft. Postán utánvéttel elküldöm. Légrádi Béla, Bp. Kada u. 118. 1104

Enterprise 128 k + magnó + könyvek + programok + joystick 16 000 Ft-ért és Primó A32 + tápegység + könyv + programok 4000 Ft-ért eladó. Jakó Attila, Gyál, Bocskai u. 119. Tel.: 1479-633 este 6-tól 8-ig.

Eladó **Enterprise 128**, 720 k floppy, 256 k RAM (+256 k betehető) 2x32 K EPROM (EXDOS, ASMON, VAR, +32 k betehető), két ROM-os kapcsolható BASIC kártya, kettős joystick csatlakozó, szines video kimenet, szakkönyvek, programok együtt eladó (35 000 Ft) Levélben: Dalos Tibor, Paks, Építők u. 22. 7030. Tel.: (75) 18-644

Hőstabilizált 48 k-s **ZX-Spectrum** monitorral, magnóval, interfésszel, joystickkal, több száz programmal, szakirodalommal egyben vagy darabonként olcsón eladó. Érdeklődni lehet hétvégén: Baráth Endre, Ajka, Korányi Frigyes u. 22. 8400

Spectrum 128 eladó + joystick, interfész, magnó sok program. Érdeklődni: Schön Roland, Békéscsaba, Tanácsköztársaság út 29-33. 5600

Spectrum emulátor Enterprise számítógépre eladó. Érdeklődni lehet: Budavölgyi László, Budapest, Szövetség u. 43. I.14. 1074

ZX-Spectrumra (48 k) és C64-re a legújabb játékok olcsón eladók. Szabó Gyula, Biatorbágy, Ipar u. 11. 2051

ZX-Spectrum (48 k) tartozékokkal: ZX printer, Turbo joystick, ZX interfész, 1 db joystick, sok játékprogram, szakirodalom, újságok - lehetőleg együtt - olcsón eladók. Árajánlatot levélben kérek. Molnár Ádám, Budapest, Muhar u. 57. 1028

A legolcsóbb! Német oktató-fordító-szótár programot 3600 szóval C64-re 950 Ft-ért lemezen utánvéttel küldök. Kérjen tájékoztatót! Csik László, Újszász, Dózsa Gy. u. 23.
DCD-80 grafikus nyomtatásra is alkalmas printer, Spectrum illesztéssel (univerzális interfész), csak 9989 Ft-ért eladó. Boros Péter, Győr, Kodály u. 40. 9023. Tel.: (96) 17-470 (este).
Final III Cartridge eladó! Tel.: 1848-845

-- HI C/! CRACKERS ! --
Ismered a POWER PACK-ot? Hát a II. változatát!? JEE! MINDENT kihoz a C64-ből amit tud! SŐT!! A legjobb demo szerelhető össze vele !! Ára lemezen 450 Ft. És ha ez nem érdekelne - bár nem hiszem - de meg akarod kapni a legújabb törésű programokat irj!!! Kótai Balázs, Sopron, Margitbányiai u. 37/a. 9400 X team GREETINGS TO FBI ILC READS ABC GWC ... all we do, is for you

Teljes HC konfiguráció elfogadható áron eladó: Commodore Plus/4 alapgép 1551-es (gyors) lemezegységgel és ékezetes betűkkel NLQ minőségben író Seikosha SP-180VC nyomtatóval, igényes szoftverrel: DELTEX, SUPERBASE, PED, TEDPAINT, stb. Érdeklődni a 183-5965 telefonon lehet egész nap.

3.5"-os márkás kétoldalas floppylemezek (Maxell, Kodak) eladók. 100% errer free! Ár: 1790-2190 Ft között. Kívánságra Amiga programokkal együtt is, külön felár nélkül. Raiz Tamás, Budapest, Győri út 8. 1123. Tel.: 1-566-941

5.25" Philips M0105 lemezek eladók. 100 Ft/db. C64 programokkal 120 F/db. Válaszborítékért listát küldök! Márton Gábor, Boglárlelle, Sallai u. 31. 8638. Tel.: 84/50-584

Veszek

Megvételre keresek C64-hez VC 1541 vagy VC 1541 II. floppyt. Ajánlatokat ármegjelöléssel címemre kérek: Sebestyén Károly, Hajdúnánás, Mártírok út 11. 4080

C64-et vásárolnék. Cím: Horváth András, Tata, Komáromi u. 18. 2/4. 2/1. 2890

Keresem az új (lapos) C64 kapcsolási rajzát. Viszlai Balázs, Múcsony, Kossuth út 81. 3744

C Plus/4 Vennék egyet, ha nem túl drágán adod. Gál Miklós 1-125-290/3842

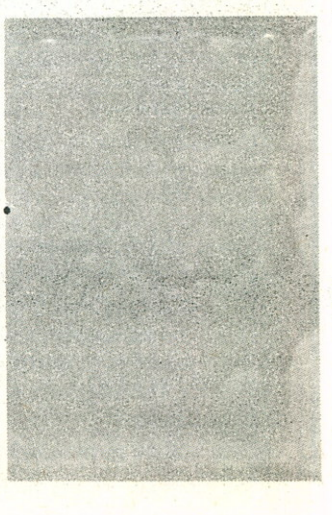
Amiga 500-at keresek olcsón (alapgépet). Érdeklődni: Savanya Zsolt, Szeged, Széchenyi tér 4.sz. 6720. Tel.: (62) 26-844

Atari 800XL programokat veszek - cserélek - eladok kazettán és mágneslemezen. Szluka János, Budapest, Fehérvári út 239. VI.em. 18. 1116. Tel.: 1-627-312

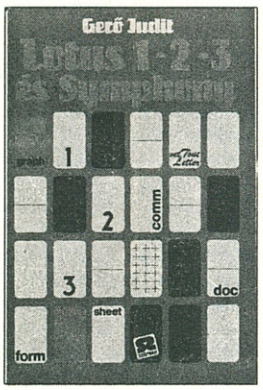
Ibanez (Japán) klasszikus gitáromat eladnám (16 000), vagy elcserélném C64-hez tartozó floppyra. Weisz János, Olasz, Kossuth L. u. 1. 7745

Cserélek

Új **Atari** 1029 típusú nyomtatómat Enterprise kompatibilisre cserélném, esetleg eladnám. Töltési Imre, Székesfehérvár, Lövölde u. 4. 9/2. 8000



Gerő Judit
LOTUS 1-2-3 és SYMPHONY
(Budapest, 1989. SZÁMALK, 196 oldal.
Ára: 256,— Ft)



Az IBM XT és AT, valamint a velük kompatibilis gépek térhódításával párhuzamosan egyre népszerűbbek az olyan alkalmazói programok, amelyek használata nem tételez fel különösebb számítástechnikai, programozói előképzettséget, sőt mindinkább igazodik a felhasználók igényeihez. Az egyik ilyen lehetséges célterület a táblázatkezelés. Az előre „gyártott" és „felokosított" táblázatkezelő programok jelentősen csökkentik és egyszerűsítik a különféle kimutatások készítésével kapcsolatos teendőket. A táblázatkezelő rendszerek nemcsak az adatok rögzítésére, táblázatba rendezésére és kiértékelésére használhatók, hanem már a tervezés folyamán is segítséget nyújthatnak, ehhez azonban ismerni kell szolgáltatásaikat, működésüket.

A kötet címadó programjai közül a LOTUS 2.0 és a SYMPHONY 1.1-es verzióját tárgyalja. Ez a részben oktató jellegű kézikönyv elsősorban a két termék felhasználóinak szól. Akik az egyes lépéseket, utasításokat ki is próbálják, hatásukat a képernyőn figyelemmel kísérhetik.

Spectrum játék és program 5.
(Budapest, 1989.
LSI ATSZ, 179 oldal.
Ára: 189,— Ft)

A kiadó népszerű „Spectrum játék és program" sorozatának legújabb kötete az előzőekhez hasonlóan épül fel. Közli a játékleírásokat, a POKE-okat és egy mikrolexikont. A felhasználói rész a hardverismeretekkel foglalkozik.

Angster Erzsébet — Béres Erzsébet:
Turbo Pascal
(Budapest, 1989.
SZÁMALK, 194 oldal.
Ára: 418,— Ft.)

A Turbo Pascal nyelv K. Jensen és N. Wirth által definiált, magas szintű, strukturált programnyelv, a Standard Pascal bővített változata, amely kihasználja az utóbbi években elterjedt személyi számítógépek lehetőségeit. Olyan önálló kompakt rendszer, amely saját szövegszerkesztővel, saját fordítóprogrammal, saját programfuttatási, tesztelési környezettel rendelkezik. Gyors, kényelmes programfejlesztési eszközként — azon felül, hogy lehetőséget nyújt Turbo Pascal programok létrehozására, fordítására, futtatására és tesztelésére — olyan programmodult is előállíthatunk vele, amely a keretrendszer nélkül, közvetlenül az operációs rendszer alatt is futtatható.

A kötet az IBM PC MS-DOS, PC-DOS operációs rendszer alatt működő, 3.01 verziójú Turbo Pascal rendszer használatára vonatkozó legfontosabb információkat foglalja össze: a rendszer generálásától kezdve, a nyelvi leíráson keresztül a leggyakrabban használandó általános és speciális megoldásokig, kiegészítve a rendszerhez kapcsolható ún. Tool-box állománykezelés lehetőségeivel.

A könyv előképzettségként feltételezi a DOS operációs rendszer és legalább egy magas szintű nyelv ismeretét, valamint Pascal nyelvi alapfogalmakat.

A kötet végén olyan mintaprogramok és feladatok találhatók, amelyek témakörönként igyekeznek bemutatni és begyakoroltatni a nyelv használatát.

„Az év szoftvere" Magyarországon

A honi számítógépes szaklapok szerkesztői már többször belefogtak, hogy a külföldi laptársak értékeléseihez hasonlóan hírveréssel támogassák meg az arra érdemes hazai szoftvereket — de ez eddig nem sikerült. Valahogy nem olyan a magyar számítástechnikai piac, hogy simán eldönthetnénk, melyik is „Az év szoftvere".

Egy ilyen kísérlet mégis érdekes, mert arra is rávilágít, hogy a fejlesztő cégek mennyire törekszenek termékeik népszerűsítésére és reklámozására. Hiszen amelyik szoftver a vita során elkerülte a zsűri figyelmét, valószínűleg nem kapott elegendő publicitást a szaksajtót rendszeresen tanulmányozó újságírók között, és ezzel párhuzamosan az sem vélhető, hogy a vásárlók — a felhasználók — tájékozottak lennének értékükről: előnyeikről és korlátaikról. Sőt, sokszor egyáltalán nem is tudnak róluk.

Az 1989-re vonatkozó minősítés, vagyis a legjobb szoftverekként rangsoroló szavazás előkészítéséről szólva önmagában is üdvös tény, hogy ez az első konkrét összefogás volt a számítógépes szaksajtó maroknyi specialistájának együttműködésére: azaz ne csak külön-külön, hanem közösen tegyenek valamit a szakma múlt évi eredményeinek népszerűsítésére, bemutatására.

Melyek voltak 1989 legfontosabb eseményei, és melyik tehát „Az év szoftvere"?

— A shareware-ek — azaz a szabadon terjeszthető olcsó programok — forgalmazása 1989 nyarán kezdődött a Cédrus Kisszövetkezet által létrehozott SOLARSOFT programközpont keretében.

— A COMPFAIR Számítástechnikai Szakvásár és Kiállítás 1989 folyamán vált rendszeres, önálló rendezvénnyé, állandó szervezőirodával, a COMPEXPO-val.

— A CWI által kidolgozott új, magyar ékezetes karakterelhelyezési szabvány teljes vagy részleges (azaz kettős hosszú ékezetes nagybetűs vagy ezek nélküli) formájában általánossá vált, már a két „különc" cég, az SZKI és a Videoton is ezt alkalmazza.

*

Ígéretes fejlesztésekként olyan szoftvereket sorolunk fel, amelyek minden bizonnyal nagy jövő előtt állnak, de még további munkát igényelnek kereskedelmi termékké válásukhoz.

— A NYELVÉSZ helyesírás-ellenőrző rendszert a SOFTINVEST keretében fejleszti egy független csoport.

— A PC TALKER beszédszintetizátor munkálatai sajnos elakadtak, mert a hírek szerint eddigi forgalmazója, a TECHNOREKORD Kft. nem vállalja a további programfejlesztési költségeket.

— A SZENZOR rugalmas vállalati adatbázis-kezelő utódjaként a Szenzor Kft.-nél folyamatosan fejlesztik a nagy sikerre számító VÉNUSZ adatbázis-kezelőt és alkalmazásgenerátort. Talán 1990 végén már forgalomba is kerülhet.

A vírusvédelem terén az év slágere a PRGDOKI programrendszer volt, amely ingyenes szoftverként vált ismertté az Ázsió Kisszövetkezet révén az 1989-es COMPFAIR szakkiállításon. További verzióit azonban a CÉDRUS SOLARSOFT hozza ki. Programozóik a továbbfejlesztését komplex vírusdetektáló rendszerként tervezik.

— A CHKVIR több verziója szabad szoftverként terjedt el, utolsó változatát a PC-Szalon forgalmazza. Az elmúlt év leggyakoribb vírusai ellen hatásos.

— DXU2 a neve a Műszertechnika ingyenes szoftverének; a „potyogós" vírus ellen fejlesztették ki. Ma már túlhaladott, de úttörő volt a hazai, szabadon terjeszthető szoftverek között.

*

Az ékezetes betűket kiíró, billentyűátdefiniáló programok közül a legújabb az UNILAB Kft. shareware programja. Még a Ventura Publisher Professional is eltűri! A másik, klasszikusnak számító program, a KLAVGEN, kereskedelmi termék — de ez ma már korszerűtlen.

*

Az egyéb programok és kiegészítők közül az AMI ANGOL—MAGYAR SZÓTÁR (SZKI HARDWARE Laboratórium) inkább még csak biztató kezdeményezés, azonban már kereskedelmi termék. Viszont sokat segít a felhasználókon a HP LaserJethez készített, magyar ékezésű sokféle betűkészlet, amely a REALCO terméke.

*

Más a helyzet azoknál a felhasználói szoftvereknél, amelyek bizony nem alacsony árúak, és a kereskedelmi forgalomban kaphatók. E programrendszereket már valóban lehetett részletesen tesztelni. A szerkesztők választása a következő termékekre esett:

— a QUATTRO adatbázis-kezelő magyar változata (Novotrade);

— a GEM rendszer (GEM Desktop, Draw Plus, 1st Word) magyar változata (az SZKI és a Novotrade közösen forgalmazza);

— a TOPCAD magyar tervezőrendszer, Apple Macintosh gépre (Graphisoft Rt.);

— a Recognita Plus lapolvasó szoftver (SZKI);

— a Xerox Ventura Publisher Professional Extension V.2.0 magyar változata (SZKI);

— a Wordstar 5.0 magyar verziója (Microsystem).

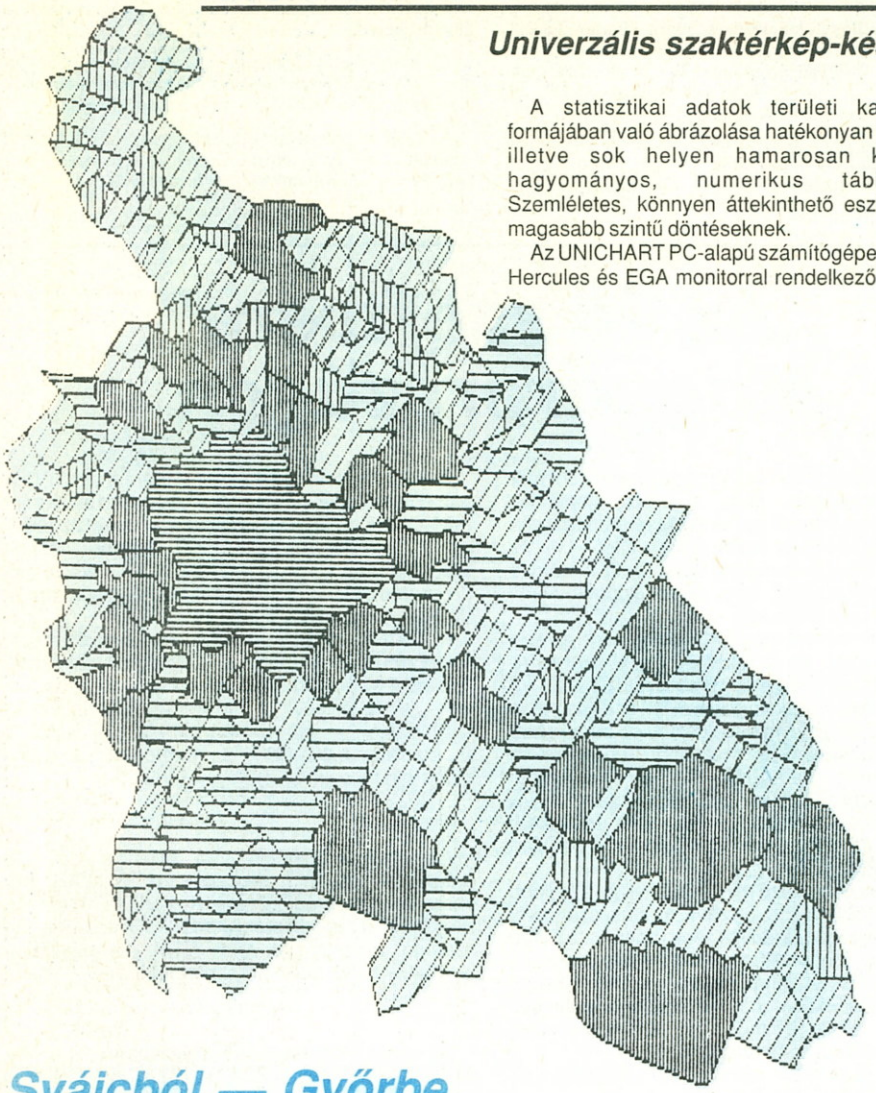
(1989. december 22-i értékelés)

CHIP Számítógép Magazin
Computer Panorama
Digitart Stúdió Egyesület
Floppy.lap
Mikroszámítógép Magazin

Az egyes szoftverek bővebb ismertetésére a későbbi számainkban visszatérünk. (A szerk.)

UNICHART

Univerzális szaktérkép-készítő program döntések előkészítéséhez



A statisztikai adatok területi kartogramok formájában való ábrázolása hatékonyan támogatja, illetve sok helyen hamarosan kiváltja a hagyományos, numerikus táblázatokat. Szemléletes, könnyen áttekinthető eszköze ez a magasabb szintű döntéseknek.

Az UNICHART PC-alapú számítógépes rendszer, Hercules és EGA monitorral rendelkező IBM XT-n, AT-n futtatható. Biztosítja a szöveges adatbázis információtartalmának hagyományos lekérdezését, csoportosítását, tematikus térképeket készít Magyarországról a kijelölt információk alapján.

Az UNICHART egyik alkalmazása az ÁFÉSZ és az Oktatáskutató Intézet közösen kifejlesztett UNICHART-OKI rendszere a népességben megtalálható iskolázottsági magatartások jellemzéséhez és az iskolázottságban bekövetkező változások feltárásához biztosít számítógépes támogatást a KSH településsoros adatai alapján.

A program a felhasználó igényének megfelelően előállított adatbázisból a szűréssel, osztályozással leválogatott paraméterek alapján printerre listáz, lemezes adatállományt, tematikus térképet, statisztikai diagramot készít az adatokkal kapcsolatos jelenségek elemzésére és a változások feltárására.

A grafikus megjelenítés alapja Magyarország 1:500 000 méretarányú digitális közigazgatási térképe. A statisztikai információk ábrázolása ezen felületi módszerrel történik. A szoftver biztosítja a LOTUS- és a FRAMEWORK-adatcserét.

Svájcból — Győrbe

Öt IBM PC/AT-vel kompatibilis számítógéppel, nyomtatókkal, írásvetítőkkel és egyéb kiegészítő eszközökkel rendezett be számítógépes termet a győri Révai Miklós Gimnáziumban egy magyar származású svájci vállalkozó, Albert Friedery, aki maga is egykor a nagymúltú győri középiskola diákja volt. Jelenleg egy számítógép-összeszereléssel és forgalmazással foglalkozó svájci székhelyű részvénytársaság többségi részvényese. A mintegy kétmillió forint értékű gépparkot kettős céllal adta át. Egyrészt az alma mater iránti megbecsülése jeléül hozzá kíván járulni az oktatási intézmény mai diákjainak számítógépes képzéséhez. Másrészt a bemutatóterem az üzletszerzést is segíti: a vállalkozó ugyanis győri székhelyű magyar—svájci részvénytársaság létrehozásán dolgozik.

Esküvő

Az Egyesült Államokban bemutatott Computerized Bride Guide (számítógépes esküvő-útmutató) szoftver azt ígéri, hogy a vőlegényt és a menyasszonyt kellő időben és stressz nélkül segíti az oltár elé. Ennek a fontos napnak az előkészítése ugyanis sok időbe és pénzbe kerül. A vendégek névsorának összeállítása, a megrendelések és az összes, elengedhetetlen apróságok beszerzése mind irányíthatóvá és ellenőrizhetővé válik. A program segítséget nyújt a meghívók írásához, ültetési rendet javasol, egy elektronikus határidőnapló pedig emlékezteti a jegyespárt a fontos határidőkre, figyelemmel kíséri az előlegeket és a költségvetéseket.

Unirobot

Elkészült az első hazai fejlesztésű robottargonca az Unirobot Kft.-nél. Az ipari számítógéppel vezérelt robottargonca akkumulátorral működik, s újratöltés nélkül 16 órát üzemel egyhuzamban. A kocsi egytonnás terhet tud önműködően szállítani a számítógép által meghatározott útvonalon. A vezérlő számítógép utasítja a robotot, hogy milyen útvonalon haladjon, hol álljon meg, milyen terhet vegyen fel, s azt hova szállítsa. Kezelőszemélyzet nélkül, önállóan dolgozik, így jól alkalmazható raktárakban, textil- és élelmiszeripari üzemekben, kórházakban. Előnyösen alkalmazható egészségre ártalmas technológiák környezetében, kémiai anyagok, vegyszerek, sugárzó és robbanóanyagok szállítására, valamint olyan helyeken, ahol az emberi tűréshatárnál melegebb vagy hidegebb van. A közeljövőben megkezdődik a sorozatgyártás.

Apple

A kaliforniai Apple cég termékei először kerültek Közép-Európába hivatalosan, jogtisztán: október közepén nyitották meg az Apple Centert a II. kerületi Borbolya utca 3-ban. Az amerikai konszern osztrák képviselete három hónap alatt rendezte be objektumát, amelynek impozáns épülete osztrák, illetve magyar magáncégek tulajdonában van.

Több célt szolgál az Apple Centerben felállított technológiai park: oktatást, fejlesztést és elsősorban művészeteket, tudományos kutatókat érintő munkahelyteremtést. A kreatív, alkalmazott videografika technikai háttériparának meghonosításával jelentős segítséghez juthatnak a magyar művészek. A kft. munkatársai máris részt vesznek például — a magyar filmgyártó vállalatokkal és reklámfilmkészítő szakemberekkel karöltve — videofilmek előállításában. Az Apple IIcx-hez csatlakoztatott színes Tektronix lézernyomtatóval a nyomdatechnikai alkalmazásokat támogatják.

Számítógépes kapcsolat

A Medimpex Gyógyszerkülkereskedelmi Vállalat közvetlen kikötői számítógépes kapcsolatot épített ki és tart fenn a brémai kikötővel. Ezzel már 1992-re készülnek, amikor a közös piaci országok közötti információkat nemzetközi adatátviteli hálózaton keresztül, kizárólag elektronikus úton lehet majd továbbítani. Az információs rendszer magyarországi rendszergazdája a Kopint-Datorg. Mód nyílik egyúttal arra is, hogy ezt a szolgáltatást a Medimpex — megfelelő díjazás ellenében — más magyar külkereskedelmi vállalat számára is lehetővé tegye. A rendszer előnye a naprakészebb, pontosabb információközvetítés, az idő- és költségmegtakarítás, valamint ezek eredményeként a biztosabb döntéshozatal.

Floppyland a Belvárosban

Újabb és újabb üzletek, bemutatótermek nyitása jelzi, hogy a számítástechnikai termékek árusítását formailag is egyre inkább áthatják a kereskedelmi jellegzetességek. Ahhoz, hogy egy terméket el lehessen adni, ma már elengedhetetlenül hozzátartozik a bemutatás lehetősége, eladó és vevő közvetlen találkozása a termék „jelenlétében". Jól segíti a folyamatot a polcról levehető szoftverek egyre bővülő köre, ugyanakkor a hardverértékesítésnél még ma is a hírnév a legdöntőbb vásárlási szempont. „Megirigyelve" azokat a hasznos tapasztalatokat, amelyekre a Magazin Diákrovatának munkatársa tett szert, bolyongva a számítástechnikai cikkeket forgalmazó üzletekben, elhatároztam, ellátogatok néhány helyre, elsősorban az újdonságokra koncentrálva, s élményeimet megosztom az olvasókkal is.

Első utam a Váci utca 84-be vezetett. Itt, a divatos utca Dimitrov tér felőli végén nyitotta meg Floppyland néven vadonatúj üzletét a Cédrus Kisszövetkezet, amely új, eredeti ötleteivel már eddig is jó néhányszor magára irányította a szakmai közvélemény figyelmét.

Sky és Tandon

Az üzlet — nevének kissé leszűkítő jelentésével ellentétben — nem csupán szoftverértékesítéssel foglalkozik; széles körű hardverválasztékát több forrásból is meríti. Ilyenek többek között a Sky Computer termékei vagy az Omikron Kisszövetkezettel közösen forgalmazott Tandon gépek. Ízelítőül álljon itt néhány „fogás" a hardvermenüből:

	Garancia nélkül,	Garanciával
Turbo XT 256/F360/MG	47 400 Ft	53 100 Ft
Turbo XT 640/F360/CG	73 100 Ft	81 900 Ft
Turbo AT 1024/W20/F1.2/MG	113 200 Ft	126 800 Ft
Turbo AT 1024/W80/F1.2/VGA	234 800 Ft	263 000 Ft
386 2048/W20/F1.2/MG 20-25 MHz	208 400 Ft	233 400 Ft
386 2048/W80/F1.2/VGA 20-25 MHz	330 100 Ft	369 700 Ft
41256-07 RAM chip (9 db),	8 000 Ft	9 000 Ft
80 287 koprocesszor (10 MHz)	32 900 Ft	36 800 Ft
80 387-17 koprocesszor	64 900 Ft	72 700 Ft
ARCNET hálózati kártya	9 400 Ft	10 500 Ft
Color grafikus kártya	3 800 Ft	4 300 Ft
Color monitor, Philips 8833	27 300 Ft	30 600 Ft
Tandon 386/20-110		589 000 Ft
Tandon 386/33-330		999 000 Ft

Szünetmentes tápegységek is szerepelnek a terméklistán a Fiskars cég kínálatából. Ezek ára teljesítménytől függően 12 000 és 4 115 000 Ft között alakul.

Polaroid, Solarsoft és a többiek

Mint a Polaroid cég magyarországi disztribútora, a Cédrus Kisszövetkezet itt, ebben az üzletében is nagy választékot kínál a Polaroid termékekből (a Váci utcai bolt látja el áruval a jogosított budapesti és vidéki viszonteladókat is). A Polaroid lemezek ára (10 db) 1800-tól 4300 forintig terjed, a monitorelőtétek átlagára 6000 forint körüli.

A bolt szoftverválasztékában természetesen előkelő hely jut a „saját" termékcsaládnak, a Solarsoft Programkönyvtár darabjainak (a katalógus közlését 1989/12. számunkban kezdtük meg). Várhatóan már a közeljövőben megjelenhetnek a Floppyland polcain az első magyar shareware-ek, freeware-ek és public domain szoftverek: a Cédrus vállalkozott rá, hogy támogatja, bátorítja ezeket a kezdeményezéseket. Begyűjti, rendszerezi és terjeszti a három kategóriába tartozó szoftvereket, újszerű lehetőséget kínálva a fejlesztők marketing-elképzeléseihez. És mindezt szinte lemezáron.

A Solarsoft mellett teret kaptak az üzletben a kész programrendszerek. Íme néhány példa:

FOXBASE 2.10	45 900 Ft
dBASE IV fejlesztői változat	139 900 Ft
dBASE IV Lan Pack	119 900 Ft
Framework III 1.1	64 900 Ft
Framework III Lan Pack	99 900 Ft
Multimate Advantage II 4.0	44 900 Ft
Draw Appleuse 1.0	42 900 Ft
Workbench/2	298 000 Ft

Norton

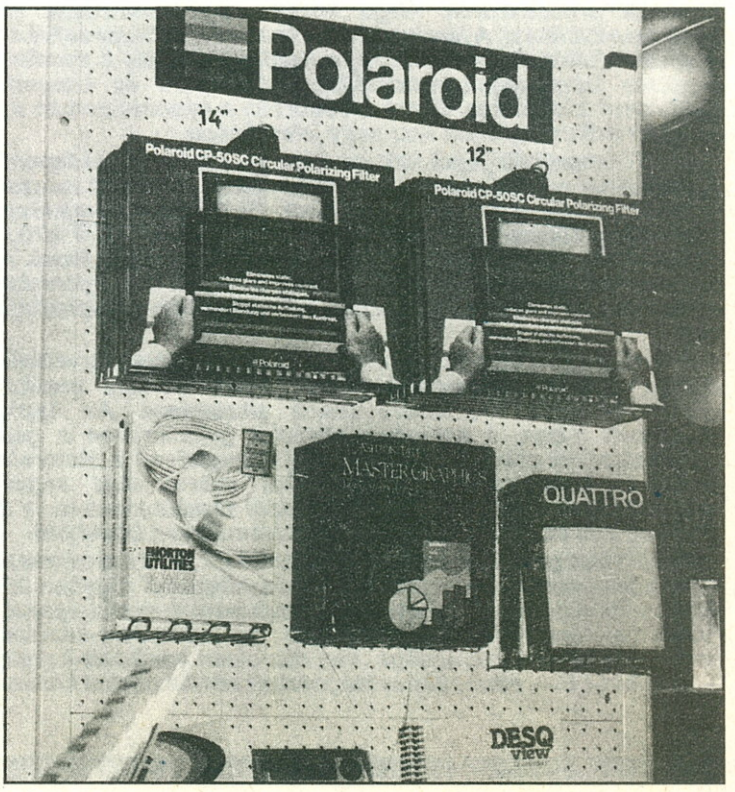
Egy tengerészsapka és egy mentőöv. Előbbi minden bizonnyal a DOS körüli navigációt, utóbbi pedig a „quick help"-et jelképezi. A két szimbólum a Norton Commander 3.0-ás, illetve a Norton Utilities 4.5-ös verziójának ismertetőjén szerepel. Aki korábban már megismerkedett a Peter Norton Computing Inc. termékeivel, bizonyára örömmel fogadja az új verziók kínálta bővített szolgáltatásokat is. A Commander View opciója — nem kevesebb, mint 26-féle formátum kezelésére képes — két szenzációs többlettel is szolgál: a képernyő két paneljének egyikén kijelölt fájl a másik panelen tanulmányozható, és bepillanthatunk a PCX kiterjesztésű grafikus állományokba is. A Commander Mail nevű szolgáltatás egyelőre még túlzott újdonságnak tűnik a magyarországi postai átviteli lehetőségek ismeretében, de a Commander Link — amelynek révén fájlok vihetők át laptopra és vissza — már nálunk is jól használható megoldás.

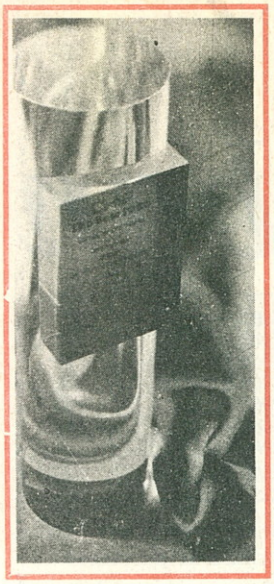
A Norton Utilities 4.5-ös verziója öt vadonatúj programegyüttest is tartalmaz — a Norton Disk Doctort, a Safe Formatot, a Norton Control Centert, a Batch Enhancert és a Disk Information nevű programot — azon kívül, hogy a korábbi verziókban szereplő elemeket is korszerűsítették a fejlesztők. Közülük például a Safe Format a hajlékony-, a Norton Disc Doctor pedig a merevlemez „egészségéért" felelős.

A két programcsomag együtt 29 900 forintba kerül a Floppylandben, külön-külön a Commander 19 600, a Utilities pedig 14 000 forintért vásárolható meg.

Összegezve tapasztalataimat: a Floppyland, arculatát és a választék bőségét tekintve, minden bizonnyal kedvelt zarándokhelye lesz — ha már most nem az! — a professzionális számítástechnika iránt érdeklődőknek.

V. J.





Az év számítógépe

Mint előző számunkban a hírekben már beszámoltunk róla, a hazánkban is ismert német számítástechnikai folyóirat, a CHIP magazin az elmúlt esztendőben is odaítélte Az év számítógépe kitüntető címeket. Ezt az elismerést szakújságírók szavazatai alapján lehet elnyerni. Egy szerkesztőség legfeljebb öt jelölt között, kategóriánként 200 pontot oszthat szét, de egyetlen jelöltre is adhatja a voksát. A lap értékítélete nem mellékes a gyártók számára, hiszen jelentősen befolyásolja a felhasználók választását egy-egy új gép beszerzésekor.

Talán ennek a lapnak köszönhető, hogy a Commodore 64 — valójában régen túlhaladott otthoni számítógép — még a múlt évben is vezette az értékesített gépek listáját Európában.

Az egyes kategóriákat két fő szempont szerint határozták meg. Az egyik a gép központi egységének, a mikroprocesszornak a típusa. Az IBM-kompatibilis gépek között tovább osztották a mezőnyt: a kézben tartható, azaz hand held, a hordozható, azaz portable és a közöttük lévő, úgynevezett laptop kategóriára.

Jelentős a fejlődés a processzorok között. Az Intel 8086-os processzorával talán már nincs is korszerű gép a piacon, csak az NDK és Kína gyártja nagyobb tömegben az ennek megfelelő berendezéseket. XT-n már régen a 8088-as Intel processzorral, illetve ennek kis fogyasztású CMOS integrált áramköri technológiával készített 80C88-as változatával szerelt gépeket értenek a szakemberek. A japán NEC cég is megjelent az XT központi egységeként a NEC V20, valamint a NEC V30-as processzorral. Az Intel 80286-os processzor az AT gépek szabványa lett. Megjelentek az ennél többre képes Intel 80386-os és az Intel 80486-os központi egységek is, amelyekben elegendő teljesítménytartalékok rejlenek a következő évekre. E processzorok legfontosabb előnye — ezért terjedhetnek el olyan gyorsan —, hogy egy fejlettebb processzoron, kompatibilis operációs rendszert feltételezve, változtatás nélkül futtathatók a korábbi központi egységre írt programok kódjai.

A másik szabványteremtő cég a Motorola, amelynek 68000-es és 68300-as típusú központi egységei ideálisak a grafikus adatok feldolgozását előtérbe helyező számítástechnikai rendszerek számára. Egyre erőteljesebb a törekvés, hogy az Intel és a Motorola processzorok világa között megvalósíthassák az adatkompatibilitást. A Macintosh megtette a kezdeti lépéseket az Appletalk hálózati rendszerével, s kezdetét vehette a mindkét géptípus lehetőségeit maximálisan kihasználó és integrált rendszerek kialakítása. Ennek elsősorban a nyomdaiparban és az elektronikus sajtóban lesz nagy jelentősége.

Az XT-kompatibilis gépek kategóriájában — Intel 8088-as központi egységekkel rendelkeznek — a legjobbnak az IBM PS/2 sorozat legkisebb tagja, a Model 30 bizonyult. Ez 8 MHz-es órajelével fizikailag is kétszer gyorsabb a megszokott XT-nél (4,77 MHz). 640 kbájtos operatív tára az alaplapon 2,64 Mbájtra bővíthető. A merevlemez mérete 20 Mbájt, a tesztek szerint 2,5-szer gyorsabb a hagyományos XT-nél. 720 kbájtos, 3,5 inches floppymeghajtója van.

A Motorola 68300-as központi egység valódi 32 bites szervezésű. A vele épített berendezések igazi professzionális grafikai munkaállomások készítésére teszik alkalmassá. Az Apple Macintosh II-ben, e kategória abszolút győztesében a gép „gondolkodási sebességét" jelentősen megnöveli a Motorola 68802-es lebegőpontos matematikai műveleteket végző célprocesszora. Operatív memóriája már alapkiépítésben is 2 Mbájtos, ami az alaplapon 8 Mbájtig folyamatosan bővíthető.

Az első Mac gépekhez csak extra tartozékként járt a merevlemez. Most a felhasználó választhat, hogy pénztárcájától függően 30, 80 vagy 160 Mbájtos winchesterrel kívánja megvásárolni a gépet. Ennek a tervezőrendszereknél és az adatbázis-kezelők használatánál van jelentősége. Van már olyan adatbázis-kezelő — az ORACLE —, amely képes megosztott adatbázisokat kezelni a legkülönbözőbb nagygépes operációs rendszerek alatt, a PC-kkel és a Mac gépekkel vegyes hálózatban is.

Még két gép volt esélyes ebben a kategóriában: Steve Jobb furcsa felépítésű „kockája", a NEXT, valamint az ATARI MEGA ST, amely már közelebb áll a mi világunkhoz, hiszen Magyarországon is ismerik.

Még mindig a 68000-es központi egységre épülnek a régebbi Motorola processzoros gépek. Ebben a kategóriában a Commodore cég vitte el a pálmát 210 pontos eredményével — erősen megosztott mezőnyben — az Amiga 500-zal. Ennek a gépnek az órajele 7,16 MHz. Munkamemóriája 512 kbájt, amely csak azért mondható jónak egy grafikára specializált gépnél, mert operációs rendszere nem itt, hanem egy 245 kbájtos ROM-ban foglal helyet. 800 Mbájtos, 3,5 inches floppyval kapható. Az Amiga DOS operációs rendszer egyszerű kezelhetőséget biztosít. Ára kifejezetten a fiatalok első komolyabb gépévé teszi, kiszorítva a Commodore 128-ast. Grafikai képességei ennek az Amigának a maga kategóriájában jók. 16 színben 640x400-as képfelbontást (jobb, mint az IBM EGA!), s ezenkívül még két üzemmódot ad a felhasználóknak. Az egyik 320x200 képpontos felbontással 64 szín, a 640x256 képpontos felbontás esetében pedig fekete-fehér képmegjelenítést biztosít.

A hordozható (portable) IBM-kompatibilis gépek kategóriájában az első helyezés egy 80386-os processzorra alapozott gépnek jutott. Az amerikai Zenith cég első, valóban csúcsszínvonalú terméke a Zenith Turbosport 386, 2 Mbájtos operatív tára az alaplapon további egy Mbájttal bővíthető. További növelés is lehetséges, de ebben az esetben az újabb memóriachipeket egy külön bővítőkártyán lehet csak elhelyezni.

Az amerikai gyár terméke teljes egészében alacsony energiafelvételű CMOS áramkörökből épül fel, így akár telepről is üzemeltethető. Ez érthető, hiszen a fejlesztéséül szolgáló alapgépet mintegy két éve a hadsereg céljaira kezdték fejleszteni. Merevlemeze 40 Mbájt kapacitású, 28 ms hozzáférési idejű. Ez ellensúlyozza azt a hátrányt, hogy a gép viszonylag lassú, két órajele 12, illetve 8 MHz. Képernyője háttérvilágítású (backlight), nagy kontrasztú folyadékkristályos megjelenítő (LDC Supertwist). A beépített grafikai kártya a 640x400-as CGA kétszeres letapogatású (doublescan) üzemmódjának felel meg. Ez utóbbi néhány program futásánál némi gondot okozhat.

A valóban hordozható, ún. laptop gépek esetében már csak az Intel 386-os gép indulhat eséllyel a szakemberek kegyeiért folyó versenyben. A két éve fogalmazott Toshiba 5100 most végre befutott. Korábban irreálisan magas ára miatt nem nyerte el a felhasználók szimpátiáját. A 8,7 kg tömegű berendezés valóban hordozható. 20-tól 100 Mbájtig mindenféle — kívánság szerinti — merevlemez beleépíthető. Bár belső hajlékonylemez-egysége 3,5 inches, 1,44 Mbájtos, lehetőség van hagyományos külső 1,2 Mbájtos, AT-szabványú lemezegység csatlakoztatására is. A grafikai kártya a VGA szabványnak felel meg, azzal az eltéréssel, hogy a színeket egy 16 fokozatú szürkeskálára transzponálja a plazmapaneles képernyőn. Lehetőség van azonban egy külső, hagyományos színes VGA monitor csatlakoztatására is.

Országváltás következett be a hagyományos gépek kategóriájában. Az eddig hagyományosan jelenlévő Compaq céget egy másik amerikai cég, a Dell Computer Inc. ütötte ki. Dee 325-ös típusszámot viselő modelljük lett az első. Szinte mindent tud, amit egy Intel 80386-alapú gépnek tudnia kell. 32 kbájtos gyorsítótárral (cache memória), 25 MHz-es órajellel, valódi 32 bites szervezésű lapozható EMS memóriával rendelkezik. Teljes a Unix-, a Xenix- és a DOS-kompatibilitása, amely vonzóvá teheti a felhasználók körében nem is oly olcsó gépet.

Már megjelentek a 40 MHz körüli órajellel rendelkező berendezések, így várható, hogy ez a gép nem őrzi meg három évig vezető helyét úgy, mint annak idején a Compaq tette. S csendben megjelent egy új processzorcsalád is: az Intel 80486. A fejlődés ismét felgyorsult, s reméljük, az árversenynek mi is hasznát fogjuk látni a nem is oly távoli jövőben.

Kis János