

if

**CALCULATOARE
PERSONALE**

Nr. 17

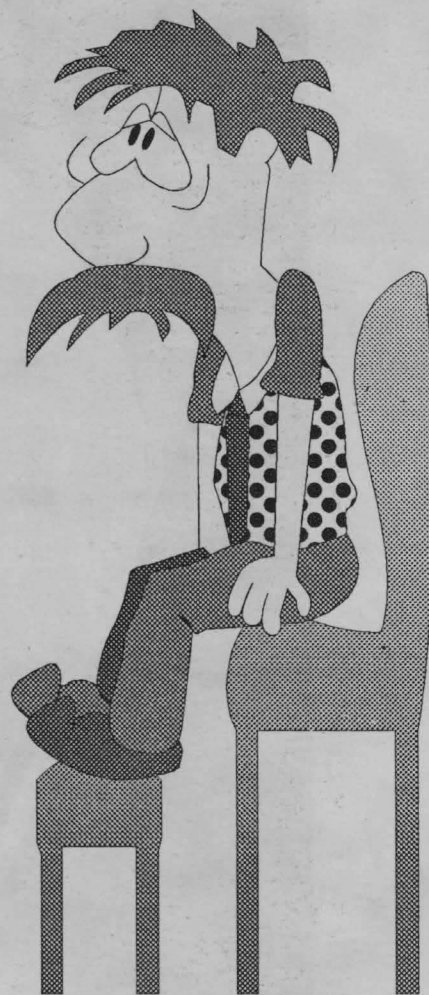
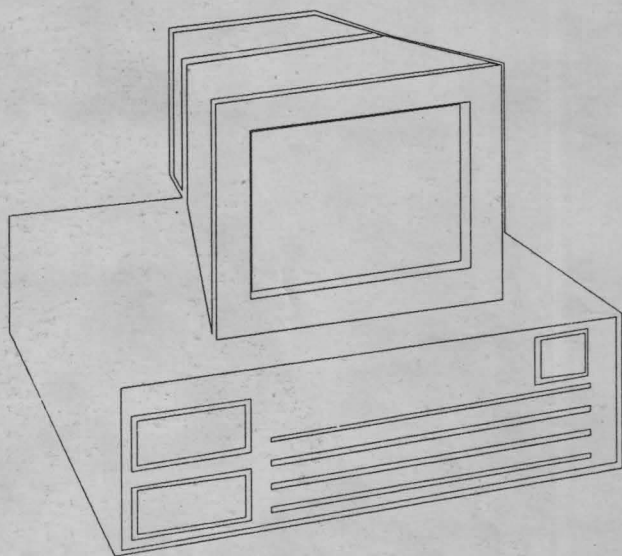
500 lei

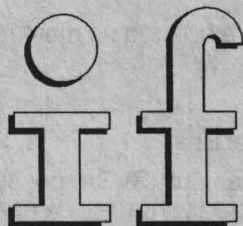
REVISTĂ EDITATĂ DE Micro ATCI S.R.L. TÂRGU-MUREȘ

ISSN 1220-1529



PRESS ANY KEY TO CONTINUE
OR
ANY OTHER KEY TO EXIT...





revistă de informatică

editată de firma



Director:

ing. Dumitru Dunca

Redacția:

ing. Cristian Nagy

ing. Mihai Beer

mat. Ioan Cozac

Keresztes Laszlo

Adresa redacției:

Micro ATCI - revista "if"

C.P. 64, O.P. 1

4300 Târgu-Mureș

Tel/Fax: 0954-31660

Manuscrisele originale sau listingurile de programe sunt primite cu plăcere, cu condiția să nu fi fost publicate în altă parte.

Materialele nepublicate nu se restituie.

Însfârșit, după o (cam lungă) perioadă, un nou număr "if". Sperăm că articolele din această apariție vor reuși să compenseze măcar într-o oarecare măsură întârzierea.

În acest număr al revistei veți găsi mai multe articole referitoare la harddiscuri și discuri optice. Credem că este o temă de interes destul de larg. Articolul despre efectele speciale în cinematografie este ales pentru a oferi câteva informații și pentru a crea un moment de destindere.

Ne face o mare plăcere să putem prezenta și o aplicație ceva mai îndrăzneată elaborată la noi în țară (vezi articolul despre gestiunea producției pe PC) care atacă un domeniu sensibil și neglijat: lansarea și urmărirea producției în firme care produc în serie.

Vă mai atragem atenția și asupra articolului cu privire la elaborarea aplicațiilor pornind de la module standardizate. Este un mod de abordare care poate ușura enorm munca de proiectare și de întreținere. Dacă nu credeți, încercați! Deoarece sursele tuturor modulelor menționate în articol ar ocupa mult prea mult spațiu tipografic, nu reproducem decât o parte. Cei interesați le pot obține pe toate de la redacție (probabil că ar fi preferabilă soluția cunoscută a unei dischete).

Pentru numărul viitor avem deja pregătite câteva articole foarte interesante despre o altă temă în vogă: procesoarele RISC.

Dacă aveți alte preferințe sau idei nu ezitați să ne scrieți. Mai ales dacă ideile se concretizează în materiale care să fie publicate în "if".

În încheiere vă asigurăm de dorința noastră (dublă de eforturi susținute) de a avea o apariție ceva mai regulată.

Cristian Nagy

Apariția acestui număr a fost susținută parțial de către

**FUNDAȚIA SOROS PENTRU
O SOCIETATE DESCHISĂ**

Noutăți de la IBM

Disc optic reinscriptibil IBM 3431

Drive-ul de disc optic reinscriptibil IBM 3431 Model 001 este un drive extern de sine stătător care suportă scriere și citire. El poate fi folosit pentru baze de date și pentru stocarea temporară a fișierelor mari cum ar fi cele folosite de produsele multimedia.

Caracteristicile sale sunt:

- interfață SCSI standard
- tehnologie avansată ce oferă fiabilitate sporită
- 650 MB per disk utilizând sectoare de 1024 octeți (numai sub OS/2 cu High Performance Optical File system - HPOFS)
- 595 MB per disc utilizând sectoare de 512 octeți (sub DOS și OS/2 cu File Allocation Table - FAT)
- rată de transfer de până la 612 kB/sec.

IBM PS/2Workgroup Server 85

Sistemele IBM PS/2 Server 85 (Model 9585 - OXA, V01) sunt calculatoare puternice care îmbină performanțele procesorului cu o mare capacitate de stocare a datelor și cu posibilități sporite de extindere a sistemelor. Aceste echipamente sunt proiectate pentru a acoperi cererile pentru servere de rețea, calculatoare host multi-user și aplicații tehnice sau economice puternice.

Caracteristici:

- procesor Intel 486SX-33MHz
- soclu pentru dezvoltare IBM PS/2 cu procesor 486-33/66
- memorie standard de 8 MB - expandabilă la 64 MB pe placa de bază
- discuri de 212 MB sau de 414 MB - spațiu disc de până la 2,63 GB
- adaptoare grafice ce oferă rezoluții SVGA și XGA-2
- SCSI cu cache integrat pe placa de bază
- tehnologie IBM ECC-P - asigură Controlul și Corecția Erorilor folosind memorii SIMM standard.

IBM PS/ValuePoint2

Noile IBM PS/ValuePoint2 6382 și 6384 au fost proiectate pentru toate domeniile - în special pentru cele care cer performanțe ridicate, memorie, grafică

SVGA, și posibilitatea de a adăuga un mare număr de periferice IBM și OEM.

Caracteristici:

tehnologie VESA Local Bus

- interfață video Local Bus SVGA pe 32 de biți 640x480, 800x600, 1024x768, 1280x1024
- 1 MB Video DRAM ca standard pe toate sistemele - opțional 2 MB
- opțiune Cache L2 (128k și 256k) pentru toate modelele
- DOS/Windows sau OS/2 preinstalat
- expandabil la procesoarele Intel Overdrive
- două porturi seriale
- două porturi paralele
- mouse

Mașinile sunt echipate cu diverse procesoare, de la 486SX - 25MHz (2510 DM) la 486DX2 - 33/66 MHz (5470 DM).

IBM Eastern Europe

PC-urile în frunte

Calculatoarele mari consemnează pierderi pe piață

Datorite slabei conjuncturi mondiale a anului 1992, Piața Mondială a computerelor a scăzut, cu 2.5 miliarde \$, la 107 miliarde \$. Pierderi clare au existat în domeniul computerelor mari, o rată de creștere clară a existat pentru PC-uri și s-au manifestat lupte de prețuri și rabaturi masive, ca și scăderi razante de prețuri.

Comerțul mondial cu PC-uri, a crescut în 1992 cu 7,4%, pornind de la 46,5 miliarde \$. Astfel PC-urile și-au "săltat" la 44,5% partea lor din întreaga piață de computere. Dimpotrivă, încasările producătorilor de computere mari au scăzut cu 16%, la 22,4 miliarde \$. Contribuția la piață în 1992 a fost următoarea:

- minisiteme 23,4%
- computere mari 21,6%
- work-stations 8,7%
- supercomputere 1,8%.

La computere mari, contribuția lui IBM a fost de 52,2%. Pe locurile 2 pînă la 4 au urmat ofertanții: Fujitsu

(9,4%), Hitachi (7,5%) și NEC (6,3%). Firmele americane Unisys și Amdahl, au căzut de pe 2 pe 5.

Pentru PC-uri, situația în 1992 s-a prezentat astfel:

- IBM 12,4% (în scădere de la 13,3% în anul anterior)
- Apple 11,2% (cu 1,42% în plus față de anul anterior)
- Compaq 6,6% (+0,6%)
- NEC 5,1% (venind de pe locul 3)
- Dell (venind de pe locul 11, și dublându-și cifra de afaceri față de anul anterior)

Cei care doresc să reziste pieții în anul 1993, trebuie să dovedească în bună măsură, inventivitate și flexibilitate.

dpa New York

High-Risc Miro

Miro oferă un computer RISC (Reduced Instruction Set Computer) complet, pe o placă ce se poate instala într-un slot. Miro și-a câștigat un nume mai ales ca producător de interfețe grafice cu rezoluție ridicată pentru CAD. Cel mai recent produs are la bază o colaborare strânsă cu specialiștii în grafică de la Silicon Graphics (SGI). Pe placă se află (cum altfel ?) și un procesor R3000 produs de MIPS, care este "asistat" de un R3010-FPU, un cache de date de 1KB, un cache de 4KB pentru comenzi și 8 până la 32 MB memorii de lucru. Și totul costă "numai" 5960 DM.

Dezvoltarea softului pentru High-Risc nu pune probleme. Mediul Iris Indigo, de exemplu, asigură portabilitatea software-ului pentru stații de lucru.

Puterea cartelei se va demonstra în special în domeniul 3D-CAD; funcțiile *Render* și *Shading* rulează cu o viteză de 6 ori mai mare decât cu un procesor 486/50. De sub AutoCAD 12, accesul la interfața High-Risc este asigurat de driver-ul numit GTI oferit de Miro. Pentru grafică, High-Risc se folosește de o interfață grafică Miro-Tiga care este accesată printr-un Local-Bus propriu.

Placa-procesor ar fi potrivită și ca platformă, pentru producerea automată a plăcilor de circuit imprimat sau ca procesor pentru fișiere PostScript. Deoarece CPU-ul de pe High-Risc poate lucra total independent de procesorul PC-ului, calea către procesarea paralelă este deschisă.

Rețea de computere fără fir

Firma italiană de computere OLIVETTI a prezentat, la Londra, prima rețea europeană de calculatoare fără fir. Conform datelor furnizate de firmă, sistemul "NET3" poate face legătura, pe frecvență radio, între 30 de PC-uri, care pot să fie așezate la o distanță de maximum 100 de metri. OLIVETTI indică faptul că rețeaua fără fir poate cuprinde calculatoarele de până acum, ea fiind compatibilă cu standardul DOS.

"NET3" lucrează în noul standard "Digital European Cordless Telecommunications (DECT)", pentru transmisiile de date fără fir. Standardul menționat a fost dezvoltat de *European Telecommunications Standard Institute* (ETSI), în colaborare cu Comunitatea Europeană. Conform firmei OLIVETTI, acest standard este recunoscut oficial de Marea Britanie, Olanda și Germania. În celelalte țări europene au fost pornite procese de recunoaștere oficială.

dpa

Primul Simpozion Internațional de INFORMATICĂ ECONOMICĂ

București 19-22 mai 1993

Organizatori și gazde au fost corpul profesoral și studenții Academiei de Studii Economice, și au fost gazde bune și la fel de buni organizatori. Simpozionul a fost organizat pe secțiuni și a avut participare internațională (Germania, Franța, Bulgaria etc.). Integrate simpozionului, au fost amenajate standuri de prezentarea a diferitelor produse soft și hard de profil. Consider că o enumerare a firmelor prezente poate da cititorului nostru o imagine a amplitudinii manifestării. Deci au fost prezente firmele: "Integrator", "Siveco-România", "Mics", "Micro-ATCI"-Tg.Mureș, "Packard Bell", "Microsoft-România", "Computerland", "Scop", "ESSI-International", "Electronum", "Ciel" ș.a.m.d.; unele din firmele enumerate au asigurat și sponsorizarea simpozionului.

Produsele-program prezentate se refereau la domeniul financiar-contabil și al mijloacelor fixe, acestea din urmă fiind prezentate într-o nouă filozofie - a mentenanței, stocurilor și achizițiilor. Această nouă filozofie, acest nou mod de abordare a unei probleme, de altfel nerezolvată la noi, este propus de firma SIVECO-

România. Produsul propus de firmă a fost apoi prezentat miercuri 24 mai la sediul Asociației Generale a Inginerilor din România.

Tot pe linie CASE (Computer Aided Software Engineering) au mers și produsele firmei ESSI-International, care a prezentat un pachet-program de lucru în informatică economică, folosind mediul de programare CLIPPER.

Sfera de cuprindere a simpozionului a fost serios lărgită prin prezentarea de către "INTEGRATOR" a produselor SUN pe stații SPARK 10 și a unei casete video referitoare la domeniul prelucrării mecanice moderne.

Noutăți de ultimă oră în domeniu, se puteau afla în discuții cu oamenii de la "Scop", proaspăt sosiți de la "CeBit"- Hanovra.

Conferință de presă SIVECO-România

26 mai 1993 (la sediul AGIR)

Conferința de presă a fost precedată de un simpozion (continuat și a doua zi) în cadrul căruia a fost prezentat amănunțit produsul COSMAN. Este vorba de un produs de mentenanță, stocuri și achiziții al firmei SIVECO (cu sediul central la Paris și având filiale în Marea Britanie Elveția și România). Domnii Quintin J. Thorn și Carlo Fighera din cadrul grupului Siveco au condus conferința de presă, asistați fiind de doamna Irina Socol de la Siveco România și de domnul Mircea Tiberiu Gruiescu, gazda conferinței.

Vă prezentăm problemele prezentate sau discutate în ordinea în care au apărut în cadrul conferinței de presă:

- Siveco are o cifră de afaceri de 4 milioane \$
- investiția făcută până acum în România este de 500.000 de franci francezi, cuprinzând numai transport și informare
- ideea a apărut în urma unor așa-numite "interviuri" (profesionale) luate la Paris unor ingineri români
- este vorba de 2 milioane de linii de program, care puteau fi scrise pentru un utilizator anume sau, așa cum s-a întâmplat de altfel, să devină un program pentru toată lumea - adică un pachet de programe pentru un domeniu de aplicații

- desigur n-a exista o bază de pornire cum ar fi procesoarele de texte cunoscute, pentru un nou procesor de texte
- pachetul are un grad de generalitate foarte mare
- România a atras atenția prin două lucruri: capacitatea inginerilor de aici, și cunoașterea de către mulți români a limbii franceze
- obiective:
 - creare unui centru de studii
 - clienți: marile uzine dar nu numai
 - transfer maxim de "know-how"
- mod de lucru: module standard
- domeniul mentenanței încă nu este informatizat la noi
- durata de implementare este de 6 luni, produsele fiind utilizate direct de muncitori, iar ergonomia - foarte slabă în domeniu - este substanțial îmbunătățită
- suportul hard este reprezentat de mașini MS-DOS, programele sunt scrise în C, baza de date este una americană de tip RDEB, rețeaua de tip Novell cu peste 50 de posturi de lucru
- politica de prețuri este flexibilă
- produsul este foarte indicat pentru întreprinderi care nu au resurse financiare de rețehnologizare, dar nici nu pot merge mai departe așa cu utilajele
- unde este vândut COSMAN este lider
- acest produs poate fi ușor configurat funcție de particularități
- pregătirea pentru utilizare cuprinde 4 cicluri a câte 5 zile
- strategia: România va dezvolta produsul pentru estul Europei
- domeniile de aplicație încep în producția de avioane și automobile, trec prin cea de parfumuri, ajungând în domeniul întreținerii clădirilor
- cea mai nouă comandă a fost preluată pentru gestiunea deșeurilor.

La cocteil-ul care a urmat, s-a subliniat din nou motivația pe care o au oamenii din domeniu, în România.

Efecte speciale

Producătorii de filme nu se mai mulțumesc să inventeze povești și să-i facă pe oameni să pară altfel decât sunt, ei inventează acum noi lumi cu ajutorul calculatoarelor. Ideile trec din imaginație direct pe marele ecran în câțiva pași - nu chiar atât de simpli, iar realitatea rămâne în urmă.

Un grădinar coboară în iad. În față îl izbesc culori clipitoare, vârtejuri amețitoare și imagini strălucitoare. Ființele umane se prefac brusc în particule care vibrează și se rotesc violent în aer.

Aceasta nu este o experiență cotidiană nici măcar pentru un grădinar dintr-un film. Jobe Smith (interpretat de Jeff Fahey), eroului filmului *The Lawnmower Man* (film ce a costat 5,8 milioane de lire sterline și a avut premiera la 5 iunie 1992), suferă un virulent atac din partea efectelor speciale. Un fost militar plin de intenții rele, pe nume Dr. Angelo (interpretat de Pierce Brosnan), folosind droguri și un costum conectat la un computer, l-a aruncat pe bietul Jobe în lumea de coșmar a Realității Virtuale.

Dintotdeauna au existat fani care au stat la coadă pentru a-și vedea vedeta favorită în orice fel de film, oricât de slab; acum există fani care sunt gata să vadă un film - indiferent de subiectul lui - numai din cauza efectelor speciale. Efectele speciale cinematografice atrag milioane de spectatori și modifică modul de realizare a filmelor.

În cazurile de lipsă acută de inspirație, efectele speciale sunt folosite pentru a acoperi golurile unei intrigi slabe sau pentru a face interesante personajele oneste, dar asta nu constituie o noutate - adesea filme slabe au fost înecate în costume scumpe și lux. (calculatoarele oferă și alt tip de asistență pentru scenariile cu adevărat proaste. Sunt disponibile programe care generează idei, verifică consecvența vocabularului, prelucrează secvențele și chiar calculează costul).

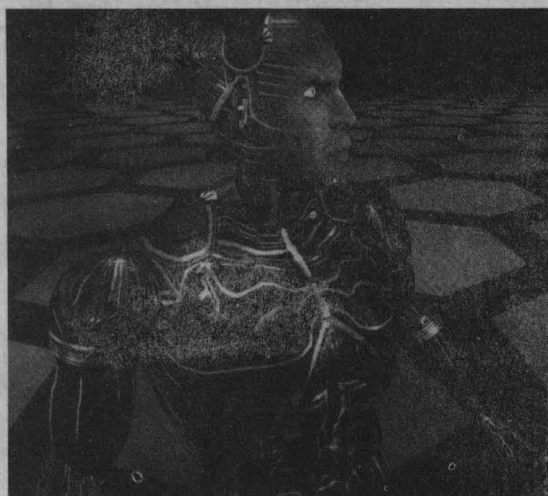
Într-un rol mai inspirat, efectele speciale le oferă regizorilor o nouă lume (de fapt, o infinitate de noi lumi)

cu care să se joace; iar când regizorii văd ce se poate face își schimbă - uneori - ideile inițiale pentru a putea folosi anumite efecte tentante. Mediul poate seduce mesajul.

"Marea schimbare constă în faptul că, în ultimii doi ani, computerele au devenit mult mai rapide" spune Michael Tolson, a cărui companie pe nume Xaos a lucrat o mare parte din efectele speciale din *The Lawnmower Man*. "Au devenit posibile o mulțime de efecte noi, și aceasta poate afecta linia scenariilor. Mulți regizori au nevoie să vadă exemple de ce se poate realiza pentru a le integra în construcția filmului."

Încă de la începuturile cinematografului, autorii de filme au lărgit aria posibilităților căutând efecte speciale noi. În anii '30 Fleischer Bros a construit decoruri miniaturale tridimensionale pentru *Popeye* și *Betty Boop* și le-a rotit pe o scenă turnantă pentru a da senzația de profunzime. Foliile de acetat pe care era desenată animația erau poziționate între decor și camera de luat vederi fixă. Personajele nu prea făceau altceva decât mergeau sau alergau pe stradă.

Computerele au făcut muncă mai ușoară. În 1983 Studiourile Disney au contactat compania de animație Magi Synthavision în legătură cu posibilita-





tea combinării personajelor bidimensionale desenate manual cu efecte tridimensionale generate cu computerul. Cu ajutorul acestei tehnici camera din calculator ar putea, de exemplu, să filmeze personajele care se urmăresc unul pe altul printr-o casă. Dispare astfel restricția decorului bidimensional.

Înainte de apariția calculatoarelor erau necesare echipamente incomode. Pentru secvența de început din *Pinocchio* (1940), Studiourile Disney au folosit o cameră specială pentru a ajusta opt nivele de decoruri pe panouri de sticlă așezate la distanțe diferite de cameră. Această cameră se putea deplasa pe verticală la diferite nivele și simula profunzimea câmpului și efectul tridimensional. În zilele noastre, pentru deplasarea unui desen bidimensional în spațiul tridimensional se poate folosi un mic computer. Deoarece el nu filmează un obiect fizic, nu mai există probleme legate de lumini și umbre sau de numărul și mărimea desenelor.

De asemenea, calculatorul poate fi folosit și pentru generarea a două mișcări diferite ale camerei pe aceeași imagine - prin apăsarea unui buton îl putem vedea pe Goofy ieșind pe ușă în timp ce camera face un zoom pe profilul lui. Înainte, această manevră cerea o mare cantitate de muncă manuală.

Pictând cu ajutorul numerelor

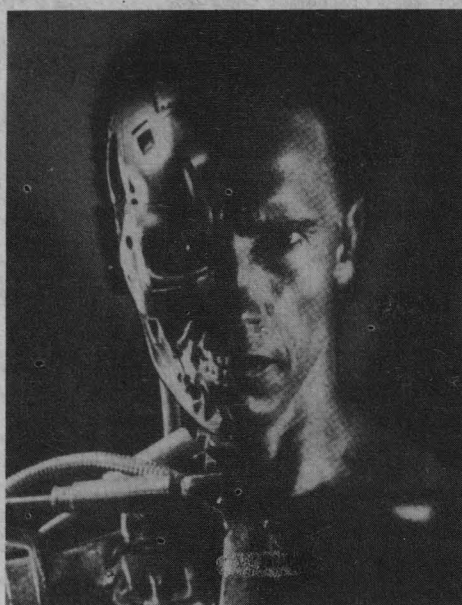
Datorită recentelor progrese din trei domenii importante, s-ar putea ca - în curând -

sunetul clacului să fie înlocuit de țâcănitul tastaturilor, pe măsură ce siliciul concurează celuloizul pentru locul cel mai important în producția de film.

Primul din aceste progrese îl reprezintă capacitatea calculatoarelor de a produce imagini apropiate de realitate. Construirea unei imagini convingătoare a lumii reale - cu infinitatea ei de lumini, umbre și forme - nu este simplă folosind logica strict numerică a unei mașini. A rose is a rose is a rose, dar un calculator ar prefera să-l toace într-o serie de cifre binare. Dar odată ce a refăcut natura, procesul de tocare este evident: inflorșcențele irizate devin blocuri poligonale de culori țipătoare, dealurile vălurite se reduc la linii zimțate.

Totuși, imaginile oferite de calculator suferă o schimbare miraculoasă dacă se folosesc din ce în ce mai multe numere pentru reprezentarea lumii reale. În ultimii ani, capacitatea de memorare și puterea de procesare ale hardului pentru grafică s-au dublat la fiecare 18 luni. Ca urmare, computerele dispun acum de forță pentru a crea imagini complexe uluitoare șubredel modele wire-frame au fost înlocuite cu structuri tridimensionale din ce în ce mai realiste.

Puterea calculatoarelor este folositoare pentru manipularea formelor complexe. Dar de la curba pectoralilor mai este cale lungă până la Arnold Schwarzenegger: tonurile pielii; sclipirile părului său periat cu grijă; felul în care razele apusului se joacă delicat pe nobila sa frunte. Calculatoarele trebuie să proceseze cantități aproape inimaginabile de informație



Arnold Schwarzenegger în Terminator 2

Gândacul inteligent
creat de
Massachusetts
Institute of
Technology



pentru a reproduce astfel de imagini. Arnie ar putea fi o muncă de o viață.

Salvarea softului: pentru a întregi puterea în continuă creștere a hardului, programatorii au creat vaste biblioteci soft de texturi ce includ piatră, lemn și țesături, care oferă o copie fidelă a lumii reale. (Sunt nevoiți să o facă dacă nu vor să scadă frumusețea filmului). Ei au dezvoltat algoritmi eficienți (numiți *ray-tracing*) care calculează cum se comportă lumina pe suprafețe, direcția din care cade, unghiul, cât se absoarbe, cât se reflectă și se refractă, care alte obiecte din vecinătate absorb sau reflectă lumina. Formulele matematice complexe conlucrează cu procesoare puternice pentru a găsi calea cea mai scurtă către imaginea perfectă. Când obiectele se mișcă într-o animație, același soft poate să calculeze din nou valorile de iluminare oferind imagini realiste fie ale lumii reale, fie ale unei fantezii debordante.

Mișcări iscusite

A doua dezvoltare importantă din ultimii ani a fost sofisticarea crescândă a tehnicilor de animație.

Producătorii de soft au integrat date anatomice exacte și mișcări ale articulațiilor în rutine de animație. Când un personaj generat de calculator ridică un deget, se calculează automat mișcarea pentru restul brațului și pentru cap. Aceasta se numește animație

ierarhică și ea poate fi văzută în cazul robotului T1000 din *Terminator 2* (1991). Articulațiile se mișcă corect-sub piele, și au fost luate în considerare gravitația și frecarea.

Personajele generate de calculator se pot mișca chiar și de capul lor. Departamentul de animație de la Massachusetts Institute of Technology a creat un gândac: acțiunile sale se bazează pe mișcările roboților și pe date despre gândacii reali; i s-a creat chiar și "inteligentă". El poate să "vadă" și să evite obiectele, și să încetinească în fața obstacolelor.

Cea de-a treia direcție de dezvoltare le propulsează pe primele două în lumea producției de film. Având în laborator secvențele pe film și având pe disc secvențele generate de calculator, mai este necesar să le punem cap la cap. Pe vremuri, aceasta presupunea proceduri complicate la montaj. Astăzi, puterea unui mainframe permite trecerea imaginilor de pe celuloid pe siliciu, combinarea lor cu secvențele de pe calculator, și trecerea rezultatului pe film. Filme ca *Terminator 2* și *The Lawnmower Man* constituie un indiciu clar asupra modului în care vor fi folosite calculatoarele - în viitor - în industria filmului, alăturând materiale generate pe calculator cu secvențe filmate pe viu. Sunt în curs de dezvoltare sisteme care permit ca totul să se realizeze pe computer în loc să se monteze manual efecte speciale în negativul filmului.



Cum s-a realizat fața lui "Lawnmower Man"

În pofida vrăjitoriilor tehnologice, *The Lawnmower Man* nu este atât de avansat încât să se descurce fără actori. CyberJobe, reprezentarea computerizată a lui Jeff Fahey, a fost creat de compania de efecte speciale Angel Studios. Grafica pe calculatoare utilizează trei tipuri de rutine: modelare - crearea formei de bază; animație - mișcare și iluminare

pe o anumită durată de timp; și rendering - combinarea celor de mai sus și transpunerea lor pe film sau pe banda video.

În *The Lawnmower Man* a fost construită mai întâi figura lui CyberJobe - punctele și liniile de control; calculatorul poate completa detaliile, după care figura este prelucrată pentru a defini culorile și texturile.

Apoi proiectantul stabilește mișcările. Se aleg cadrele cheie, iar

programul calculează cadrele intermediare. Pentru fiecare cadru redat, se calculează exact figura, obiectele, iluminările și culorile. Acest proces poate dura câteva ore pentru un cadru; când cadrul este gata, el este înregistrat pe film. Autorii animației au folosit peste 70 de fotografii reprezentând mișcările faciale și vocale ale lui Fahey pentru a sincroniza sunetul.

Către sfârșitul anilor '90 oamenii vor putea să ducă filmele într-un atelier (așa cum facem astăzi cu filmele fotografice), să plece de acolo cu un disc pentru calculator, să prelucreze imaginile acasă sau la birou și să ducă discul înapoi la acel atelier pentru a trece filmul pe bandă video.

Această cooperare dintre camera de luat vederi și calculator a devenit din ce în ce mai eficientă ca uemare a dezvoltărilor cunoscute de controlul deplasării camerelor din anii '70. Fiecare mișcare a lor era controlată de calculator, ceea ce-i dădea spectatorului senzația că participă la acțiune. Acum, control computerizat înseamnă că datele de la cameră pot fi transmise programelor de modelare și animație, astfel încât elementele generate digital să se potrivească exact cu acțiunea filmată pe viu.

Toate acestea vor schimba și cinematografele. Numărul filmelor tridimensionale care cer ochelari roșii și verzi este în creștere, la fel ca și cel al utilizării sunetului stereo digital; sunt în curs de perfecționare sisteme de feedback de la public.

De exemplu: spectatorii țin în mână bețe din material plastic în două culori: roșii pe o parte și verzi pe cealaltă parte; imaginile sunt înregistrate de camere video care transmit informațiile unor PC-uri care proiectează apoi matricea pe ecran. Acest sistem poate fi folosit pentru votare (da sau nu) sau pentru controlul imaginilor. La congresul de grafică pe calculator Siggraph, ținut la Las Vegas în 1991, 4000 de spectatori au jucat o partidă de ping-pong pe ecran, fiecare paletă fiind controlată de 2000 de oameni.

La Venice, în California, există un loc numit *Mr. Film* - pe a cărui firmă stă scrisă caracterizarea de "teatru virtual". Cu ajutorul unui mouse și a unei mânuși (legate, evident, la un calculator) pot fi controlate mișcările lui Silver Suzy - o fată care practică surfing.

În ciuda acestor progrese, din ce în ce mai mulți oameni vor viziona filme acasă, pe televizoare cu ecran mare. Probabil că vom putea zburda printre clipuri de mărimea unei cărți de joc pentru a ne alege filmul, sau vom putea tasta o scenă, un gen sau un

Eșantionarea vedetelor

Utilizarea unor persoane celebre pentru a realiza animație pe propriul PC nu este chiar atât de îndepărtată. Va deveni din ce în ce mai ușor să se comprime imaginile actorilor din filme în formă digitală, să fie puse de discuri floppy și apoi să fie aduse pe ecranul calculatorului.

O dată ce aveți pe Madonna sau pe Arnie pe monitor, puteți folosi softuri de animație pentru a-i pune să facă orice doriți. Se pot genera secvențe întregi, deși aceasta ar cere foarte mult timp și sute de mega octeți de spațiu pe disc în cazul unui PC standard. Ar mai fi necesar un sistem rapid de comprimare a datelor pentru a putea canaliza toată informația către ecran suficient de repede pentru animarea imaginilor. Va apare tentația de a folosi imaginile pentru a face filme "de casă", create pe noile calculatoare multimedia, salvate pe discuri hard sau optice și puse înapoi pe un film.

Aceasta deschide un nou front de luptă pentru copyright, așa cum a făcut-o sampling-ul în industria muzicii. Autorii de filme lipsiți de scrupule ar putea utiliza imagini a căror proprietate nu o dețin drept sursă ieftină de material brut; ei ar putea să modifice aceste imagini pînă cînd devin aproape de nerecunoscut, sau ar putea - pur și simplu - să facă filme ilegale "distribuind" în ele actori și actrițe care ar fi foarte (folosind un eufemism) iritați dacă ar ști.

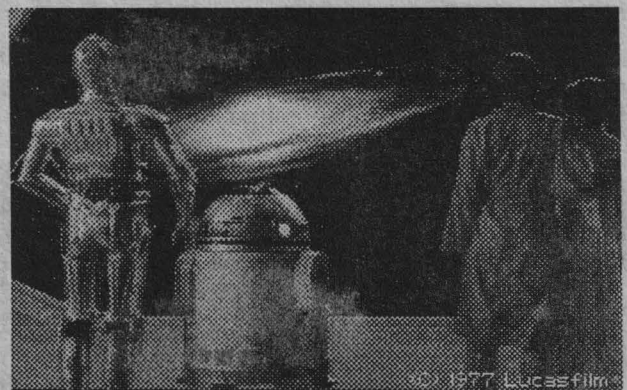


actor. Apoi filmul ales va fi selectat din baza de date și transmis (industria video studiază deja această posibilitate) prin fibre optice calculatorului nostru de acasă, care va fi conectat la un televizor cu ecran mare. Filmului i-ar fi necesare circa cinci secunde să ne parvină.

Am putea chiar să vizionăm filmele interactiv, punând întrebări personajelor sau rulând așa-numite scenarii "what-if" (ce se întâmplă dacă ...) - cum ar fi *Hamlet* cu happy-end.

Dar cel puțin vedetele vor fi din carne și oase. Mai este cale lungă pînă cînd un personaj generat de calculator să ne păcălească așa încît să credem că este o ființă umană sau un peisaj digital să pară natural. Teoretic, toate acestea ar trebui să fie posibile în secolul viitor: dar în ce măsură ne încântă perspectiva, este o altă întrebare. Oricum, la vremea respectivă, s-ar putea ca vizionarea filmelor acasă să însemne personaje holografice plimbându-se prin camera noastră de zi.

Traducere:
ing. Cristian Nagy



Gestiunea producției pe PC

1. Introducere

Răspândirea deosebită a microcalculatoarelor personale compatibile IBM-PC în toate sectoarele vieții economice și sociale impune necesitatea elaborării unor programe performante care să rezolve multiplele probleme practice aferente acestor sectoare.

Se constată o dezvoltare a activității de elaborare de programe de către diferite colective de specialiști renumiți în societăți comerciale care au ca activitate de bază producerea de software original sau adaptarea unor pachete străine la particularitățile legislației românești.

Una din aceste firme este SOFT SERVICE S.R.L. din Brașov, a cărei activitate principală o constituie elaborarea de sisteme informatice în domeniul gestiunii economice.

Strategia acestei firme este următoarea: realizarea unei sistem informatic integrat care să rezolve problemele majore ale societăților comerciale și - independent de aceasta - elaborarea unor programe la solicitarea diverșilor beneficiari de lucrări informatice.

În continuare va fi prezentată preocuparea de realizare a sistemului informatic integrat pentru conducerea proceselor economice din societăți comerciale - cu precădere din domeniul industriei - și realizările obținute până în momentul de față.

În etapele deja parcurse, au fost cooptați specialiști cu experiență practică în domeniul analizei de sistem și a proiectării de sisteme informatice.

Ideile de bază în realizarea sistemului informativ sunt următoarele:

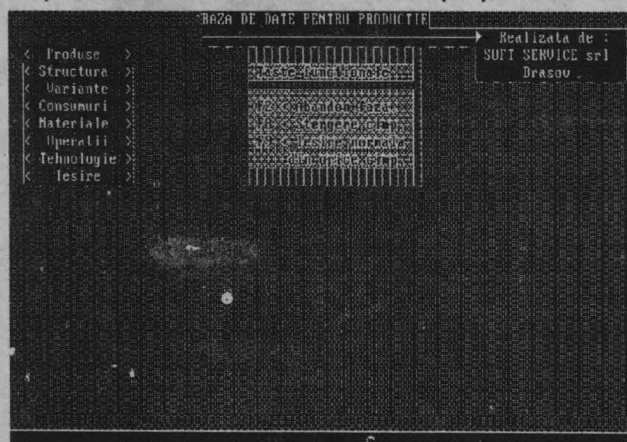
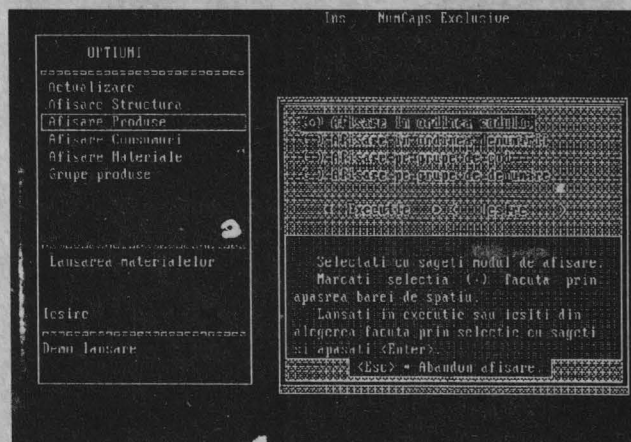
- constituirea unei baze de date unice, ca element integrator;
- realizarea în etape;
- abordarea cu precădere a problemelor din subsistebul producției;
- introducerea în sistem a unor funcții care să asigure strategia și tactica optimă în realizarea obiectivului.

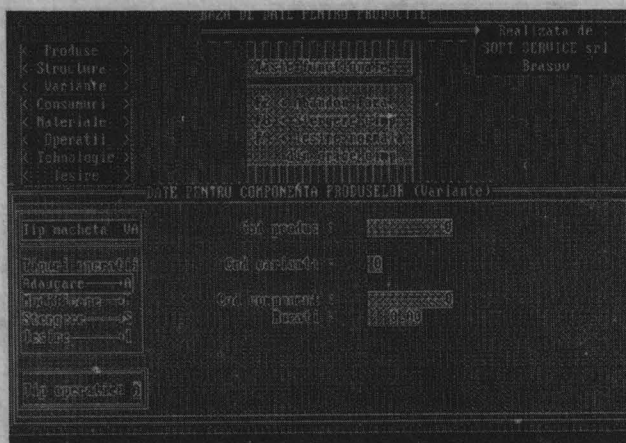
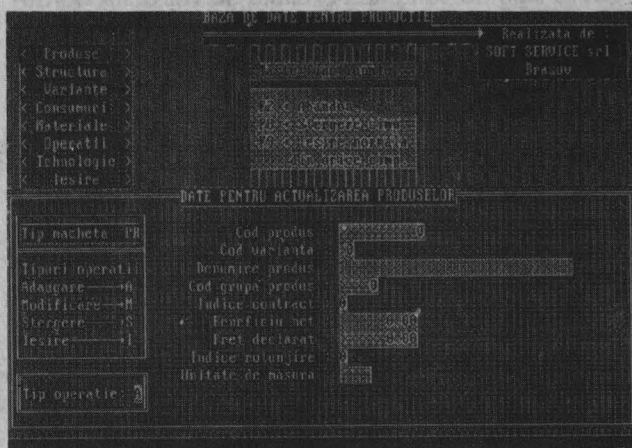
2. Prezentarea subsistemului informatic BD-LANS (baza de date - lansare în producție).

Pentru realizarea unui sistem informatic complex, care să rezolve multitudinea de probleme tehnico-economice cu care se confruntă o societate comercială cu activitate de producție în domeniul industriei, este necesară proiectarea bazei de date unice, ca element integrator al sistemului informatic.

La proiectarea bazei de date trebuie analizate relațiile obiective (existente în realitate) dintre datele care vor fi prelucrate stabilindu-se repartizarea lor în fișiere precum și legăturile dintre ele, în funcție de aceste relații.

Complexitatea deosebită a sistemului informatic integrat impune cu necesitate proiectarea și realizarea bazei de date în etape succesive. Unele din aceste etape au fost parcurse prin realizarea, în decursul timpului, a unor sisteme informatice pe platforme de





diferite tipuri, folosind instrumente de programare diverse (BOMP, SCF, SOCRATE, PRENOM, etc.)

În continuare va fi prezentat un nucleu al bazei de date care conține datele strict necesare activității de lansare în fabricație, implementat pe calculatoare compatibile IBM-PC.

Lansarea în fabricație este o subactivitate de bază a activității de programare a producției, ocupând un loc în aval de programarea propriuzisă și în amonte de execuția și urmărirea îndeplinirii programelor de producție. Putem defini lansarea în fabricație ca fiind ansamblul de activități legate de elaborarea documentației economice, multiplicarea și difuzarea acesteia și declanșarea execuției sarcinilor repartizate pe fiecare loc de muncă din cadrul verigilor structurale ale societății comerciale.

Lansarea în fabricație - la momentul oportun - a tuturor produselor incluse în programul (optim) de fabricație este o obligație majoră, contribuind la buna desfășurare a procesului producției. Ea reprezintă liantul care leagă valc:ile potențiale cuprinse în documentele economice de realizarea efectivă a producției finite.

Lansarea în producție reprezintă un mijloc de verificare a dimensionării cheltuielilor normate pe unitatea de produs (a cheltuielilor materiale și a celor cu munca vie), în principal, cuprinse în documentația tehnologică.

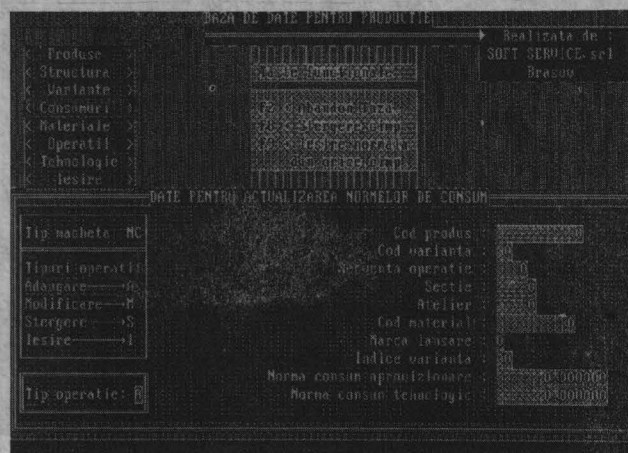
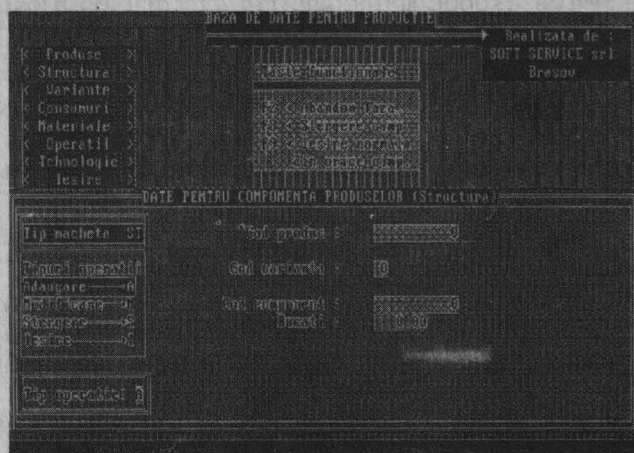
Metoda efectivă de lansare în fabricație utilizată în continuare va fi metoda *pe comenzi*.

Cu ajutorul calculatorului electronic se poate elabora documentația de lansare în fabricație într-un mod operativ și eficient; mai mult, există posibilitatea reținerii informațiilor cuprinse în documentele de lansare elaborate în vederea urmăririi operative a abaterilor care apar - pe parcursul realizării programului de producție - față de valorile din momentul lansării.

Aplicațiile succesive care constituie lansarea materialelor și manoperei sunt:

1. calculul necesarului de repere de fabricat.
2. elaborarea borderoului, a bonurilor și a fișelor limită de materiale.
3. elaborarea borderoului de manoperă și a bonurilor de lucru (sau a altor documente specifice situației concrete) și vor fi descrise succint în continuare.

1. Calculul necesarului de fabricat





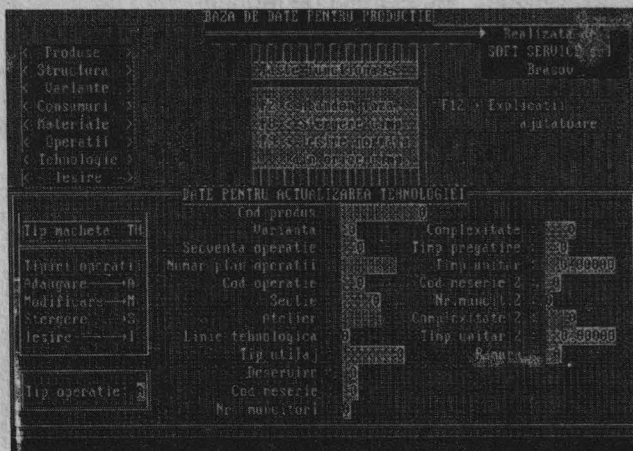
Datele de intrare ale acestei aplicații - cu excepția celor incluse în nucleul bazei de date - sunt conținute în notele de lansare emise de serviciul *Programarea, lansarea și urmărirea producției*: codul produsului (eventual și al variantei de produs), numărul comenzii și cantitatea de produs lansată în fabricație, pentru fiecare din produsele care constituie programul de fabricație.

Cu ajutorul structurii pe nivele a produselor, memorată în fișierul de structură din nucleul bazei de date, se obține necesarul de repere de fabricat printr-o procedură de descompunere a produselor în repere (explozie) și o cumulare pe comenzi a cantităților de repere identice.

2. Elaborarea borderoului, a bonurilor și a fișelor de material

Pe baza necesarului de fabricat obținut în etapa anterioară și a datelor înregistrate în fișierele de consumuri și materiale, se determină cantitatea cumulată de material necesară în două variante:

- la nivelul întregii comenzi,
- la nivelul reperului în cadrul comenzii.



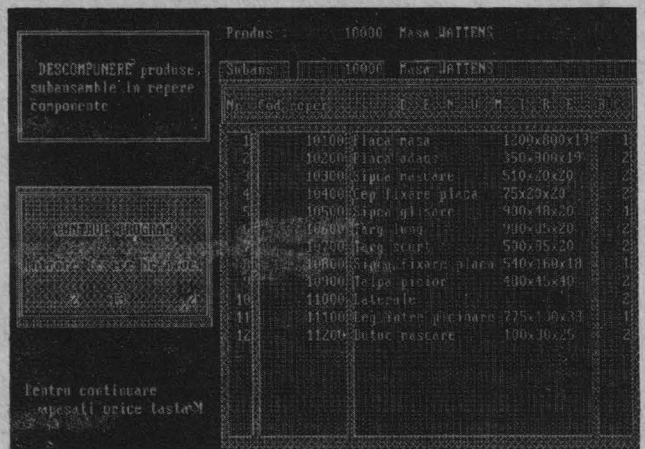
În funcție de varianta solicitată se vor edita borderoul de materiale (ca situație recapitulativă a operației de lansare) și bonurile sau fișele limită de materiale. Conținutul acestor documente poate fi analizat pe baza exemplului prezentat.

3. Elaborarea borderoului manoperă și a bonurilor de lucru

Modalitatea efectivă de lansare a manoperei este foarte diferită de la o firmă la alta. Din acest motiv nu a fost exemplificată această etapă; este necesară adaptarea pachetului de programe la cazul concret.

Informațiile care conduc la elaborarea documentelor specifice lansării materialelor trebuie cuprinse - în principal - în fișierul de tehnologie. O structură mai răspândită a acestui fișier este prezentată în etapa de afișare a ecranelor de prelucrare a datelor. Se propune aici o noțiune - *complexitatea operației* - care poate înlocui vechile elemente de calcul pentru manoperă (rețea tarifară, categorie, treaptă).

În principiu, complexitatea reprezintă un procent pe o scară în care treapta de plecare este salariul minim.



3. Pachetul de programe BD-LANS

Pe baza suportului teoretic prezentat mai sus, a fost elaborat pachetul de programe BD-LANS care rezolvă încărcarea nucleului bazei de date și lansarea materialelor și manoperei.

Structura nucleului bazei de date a fost prezentată deja, informațiile cuprinse în fișierele care îl compun putând fi deduse din ecranele de încărcare a datelor.

Programele sunt scrise în FOXPRO 2.0 și au performanțe deosebite în ceea ce privește asigurarea corectitudinii datelor introduse (încă din faza de încărcare a informațiilor în fișierul de structură se evidențiază neconcordanțele logice - evitarea buclărilor) și în ceea ce privește viteza. Timpul de răspuns redus este asigurat în special de programul de descompunere pe

Cp	Cat Cp	Cp	Ct	Pr	Pa
10000	0	Mesa WITTENS	0	0	0,00
10109	0	Placa masa	1200x800x19	0	0,00
10101	0	Panou placa	1200x800x13	0	0,00
10102	0	Furnir fata	1200x800	0	0,00
10103	0	Furnir dos	1200x800	0	0,20
10104	0	Elind cant lung	1200x19x2	0	0,00
10105	0	Elind cant scurt	800x19x2	0	0,00
10200	0	Placa adanc	350x800x19	0	0,00
10201	0	Panou placa	350x800x10	0	0,00
10202	0	Furnir fata	350x800	0	0,20
10203	0	Furnir dos	350x800	0	0,20
10204	0	Elind cant lung	800x19x2	0	0,00
10205	0	Elind cant scurt	400x19x2	0	0,00
10300	0	Sijca nascuta	510x20x20	0	0,00
10400	0	Top fixare placa	75x20x20	0	0,00
10500	0	Sijca glisiera	900x40x20	0	0,00
10600	0	Targ lung	900x45x20	0	0,00
10601	0	Furnir fata	900x45	0	0,00
10602	0	Furnir dos	900x45	0	0,20
10700	0	Targ scurt	500x45x20	0	0,00
10701	0	Furnir fata	500x45	0	0,00

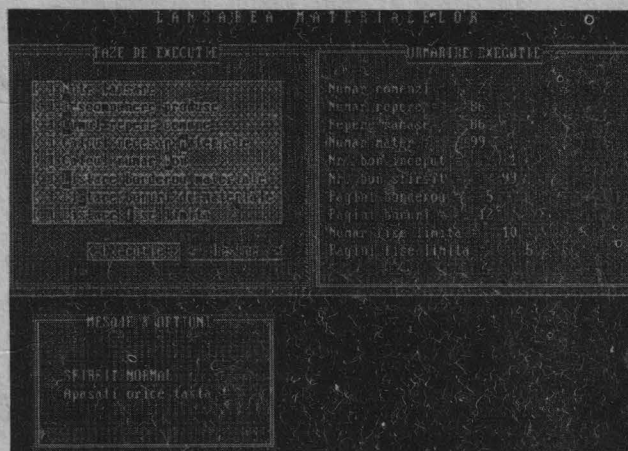
nivele (maximum 10 nivele), care constituie piesa de rezistență a aplicației.

Numărul de nivele poate fi mărit, înregistrându-se o scădere corespunzătoare a vitezei programului de descompunere. În cazurile practice întâlnite până în prezent, numărul de 10 nivele a fost acoperitor chiar pentru produse de complexitate aparent ridicată. De exemplu motoreta *Hoinar* are reperele dispuse pe 6 nivele.

Experimentarea funcționării subsistemului a fost realizată la Societatea Comercială *COMPAL* din Covasna care are ca principală activitate producția de mobilă. Există aici o preocupare deosebită pentru introducerea metodelor moderne de conducere și organizare - inclusiv utilizarea unei rețele de calculatoare - impulsionată de domnul director Csegezi Laszlo.

Exemplificările din acest articol au fost făcute cu date culese la firma menționată.

Pe baza informațiilor încărcate în nucleul bazei de date au fost realizate și alte aplicații pentru maximizarea profitului în funcție de resursele existente la un



Societatea comerciala COMPAL ITSA LIMITA CONSUM MAT. Covasna Nr.fisa: 4

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 200 Nr.comanda: 1111111111111111
Reper: 500011 Denumire:Panou PAL

Material Cod: 10011001
Denumire: PAL 10m cl.B

Cant.neces: 40.10 Pret: 475.00 Valoare: 30804.36
Cant.eliberata: 00:00 Cod abateri: Stare bon

Valabilitate pe luna Control preventiv

Data si sematurii ceruta / Zilnic / Cumlat / Predare / Primire

Cantitate transferata pe fisa limita nr.....

Societatea comerciala COMPAL ITSA LIMITA CONSUM MAT. Covasna Nr.fisa: 4

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 200 Nr.comanda: 1111111111111111
Reper: 500101 Denumire:Panou PAL

Material Cod: 10011001
Denumire: PAL 10m cl.B

Cant.neces: 88.10 Pret: 475.00 Valoare: 30804.36
Cant.eliberata: 00:00 Cod abateri: Stare bon

Valabilitate pe luna Control preventiv

Data si sematurii ceruta / Zilnic / Cumlat / Predare / Primire

Cantitate transferata pe fisa limita nr.....

Societatea comerciala COMPAL DON DE CONSUM MATERIALE Covasna Nr.bon: 1

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 100 Nr.comanda: 1111111111111111
Reper: 500501 Denumire:Panou PAL

Material Cod: 10011201
Denumire: PAL 12m cl.B

Cant.neces: 50.06 Pret: 423.00 Valoare: 22820.43
Cant.eliberata: 00:00 Cod abateri: Stare bon

Data si sematurii / Predator / Primitor / Control preventiv

Societatea comerciala COMPAL DON DE CONSUM MATERIALE Covasna Nr.bon: 2

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 100 Nr.comanda: 1111111111111111
Reper: 500801 Denumire:Panou PAL

Material Cod: 10011601
Denumire: PAL 16m cl.B

Cant.neces: 11.11 Pret: 452.00 Valoare: 5022.22
Cant.eliberata: 00:00 Cod abateri: Stare bon

Data si sematurii / Predator / Primitor / Control preventiv

Societatea comerciala COMPAL DON DE CONSUM MATERIALE Covasna Nr.bon: 3

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 100 Nr.comanda: 1111111111111111
Reper: 500501 Denumire:Panou PAL

Material Cod: 10011602
Denumire: PAL 16m cl.B

Cant.neces: 41.11 Pret: 440.00 Valoare: 18088.00
Cant.eliberata: 00:00 Cod abateri: Stare bon

Data si sematurii / Predator / Primitor / Control preventiv

Societatea comerciala COMPAL DON DE CONSUM MATERIALE Covasna Nr.bon: 6

Cod predator / Cod primitor / Cost creditor / Cost debitor
0 / 0 / 200 / 500/00

Produs/Lucrare: Bancheta "WIT"
Cod Doc.lansate 200 Nr.comanda: 1111111111111111
Reper: 500901 Denumire:Panou PAL

Material Cod: 10012202
Denumire: PAL 22m cl.B

Cant.neces: 20.04 Pret: 500.00 Valoare: 10022.20
Cant.eliberata: 00:00 Cod abateri: Stare bon

Data si sematurii / Predator / Primitor / Control preventiv

moment dat și pentru elaborarea strategiei optime de aprovizionare cu materii prime și materiale.

Cei interesați pot contacta colectivul care a elaborat pachetul de programe la Societatea Comercială *SOFT SERVICE S.R.L.* str. Zizinului nr. 87, sc.A, ap. 3, Brașov.

Fiz. ec. Constantin Dumitriu.

O privire asupra harddiscurilor

Toată lumea îl folosește dar nimeni nu-l poate defini, sau mai bine zis nu-i cunoaște "viața interioară". Importanța lui iese în evidență abia atunci când apare o defecțiune. Este vorba de harddisk, acea memorie magnetică rotativă cu capacitate și viteză apreciabile.

Principiul de funcționare a unui harddisk nu este greu de înțeles, deoarece se aseamănă cu cel al dischetei. Ca și la dischete, biții sunt traduși în schimbări de sens ale fluxului magnetic. Aceasta înseamnă că particulele magnetice au câte o polaritate pentru "0", respectiv "1", diferite între ele.

În ce privește dimensiunile, harddisk-urile se diferențiază net de dischete. (Fig. 1)

Spre deosebire de dischete - care funcționează la 300 rot/min, harddisk-urile (mai exact discurile din acestea), se rotesc cu 3600 - unele model chiar cu 5000 - de rotații pe minut. Este de înțeles că stratul magnetic nu poate fi aplicat pe un suport asemănător cu cel al dischetei, deoarece la asemenea rotații acesta ar începe să fluture. Suportul mecanic al harddisk-urilor îl reprezintă discuri de aluminiu (uneori de sticlă). Pentru mărirea capacității, pe același ax se montează mai multe discuri, unul peste altul (Fig. 2). Radial față

de ax, asemănător pick-up-ului, pe un braț mobil, sunt dispuse capetele de înregistrare-redare - și anume câte unul pentru fiecare față de disc.

O atingere (contact nemijlocit) între cap și disc - foarte temutul *Headcrash* - ar fi mortală la această turație, rezultând distrugerea capului și pierderea datelor. Acest accident poate fi evitat prin efect Bernoulli. La această turație, între disc și cap se formează o pernă de aer, astfel încât capul "plănează" deasupra discului la o distanță de un micrometru. Astfel se explică sensibilitatea harddisk-urilor la orice tip de șocuri. Praful, amprente și firele de păr căzute între cap și disc au proporții imense în acest caz. De aceea harddisk-urile se fabrică în medii protejate la praf, iar carcasa lor este ermetică sau echipată cu filtre ultrafine. Înaintea fiecărui transport discurile trebuie "parcate". La acest procedeu - care poate fi executat automat în momentul decuplării de la rețea (funcția *autopark*), sau inițiat manual, prin lansarea unui program (*ship* sau *park*) - capetele de citire-scriere sunt deplasate într-o zonă care nu conține date. Acolo capetele pot ateriza pe disc (*crashing zone* = zonă de aterizare), fără pericol de deteriorare. Suplimentar, capetele sunt fixate rigid în poziția de parcare și - în cazul în care sunt manipulate corespunzător - este evitată deteriorarea lor. Interesantă este funcția *autopark* a unităților de disc moderne. Deconectat fiind de la rețea, discul se mai rotește o vreme, din inerție. Un generator trans-

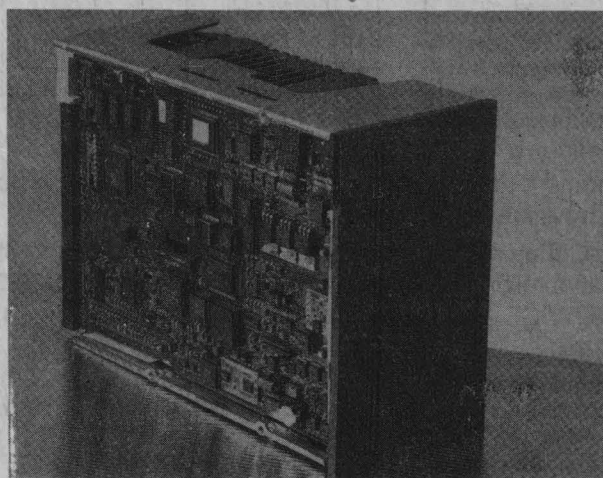


Fig. 1 Micropolis 1548 de format 5,25", are o capacitate de 1,7GBytes

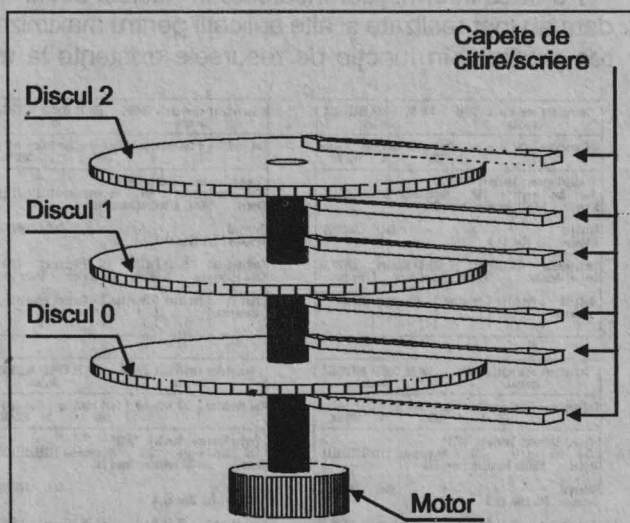


Figura 2 - Structura unui harddisk

Despre BPI și FPI

Așa-numita densitate de înregistrare se măsoară în BPI (*Bits Per Inch*). La ora actuală valorile uzuale sunt de aproximativ 40.000 BPI (vezi tabela 1). O unitate la fel de uzuală este *Flux Per Inch* (FPI), însemnând "schimbări de flux pe inch", și indică de câte ori se poate modifica orientarea particulelor magnetice pe distanța de un inch. Din motive obiective, distanța dintre două schimbări de flux nu poate fi sub o anumită limită. Acest lucru este determinat de faptul că altfel interacțiunea dintre doi biți alăturați ar fi prea intensă și ar avea loc schimbări de polaritate, deci pierderi de date. Înșă, cu cât valorile BPI respectiv FPI sunt mai mari, cu atât încap mai multe date pe harddisk.

formă această energie de rotație într-un curent electric, care este folosit apoi pentru parcare capetelor în zona nepericuloasă. Spre deosebire de unitățile de dischete, discurile din unitatea de disc se rotesc continuu pentru a se evita pierderile de timp la fiecare pornire; rapiditatea, care este avantajul harddisk-urilor, s-ar pierde. De aceea vom trata acest punct forte al harddisk-urilor: rapiditatea. Chiar și primele unități, care aveau un timp de acces de 80 ms, au fost un progres enorm. Timpul mediu de acces, la cele mai evoluat modele la ora actuală, este sub 10 ms. Unitățile de dischetă au timpii de acces între 150 și 200 ms. Deoarece stratul magnetic și capetele sunt de calitate superioară, harddisk-urile au și capacitatea mai mare. Pe când discheta memorează maximum 2,88 Mbyte, pentru harddisk 1 GByte (1 GByte = 1024 MByte) nu reprezintă nici o problemă. De asemenea harddisk-ul este net superior și în ce privește rata de transfer a datelor. Pe când discheta asigură 30-40 Kbyte pe secundă, chiar și un harddisk lent atinge 200 KByte pe secundă. Cele medii ating 600-700 KByte pe secundă, iar de la 1 MByte în sus începe clasa "high-end", care ajunge până la valoarea de 2 MByte pe secundă (toate aceste valori au fost determinate în condițiile lipsei memoriei cache).

Organizarea datelor

Pe suprafața magnetică, prin formatare *low-level*, se instalează piste de date sub forma unor cercuri concentrice. Pistele unui pachet de discuri aflate pe aceeași verticală se numesc generic cilindri. Cu cât pistele sunt mai apropiate, cu atât crește capacitatea discului. Densitatea pistelor se măsoară în TPI (*Tracks Per Inch* = piste pe inch) Deoarece circumferința pistelor centrale este mai mică decât a celor de la periferie, ele au și capacitatea mai mică. La harddisk-urile mai vechi, pista centrală stabilea numărul de sectoare (1 sector = 512 Bytes de date) și pe celelalte piste.

ZBR creează simțitor mai mult spațiu

La scurt timp, s-a elaborat un procedeu de înlăturare a risipei de spațiu pe pistele periferice care sunt, cum am văzut, mai lungi. Cuvântul magic este *Zone-Bit-Recording* sau ZBR. Discul este compartimentat în mai multe grupe de piste. În cazul ideal s-ar calcula valoarea optimă pentru fiecare pistă. În acel caz efortul controler-ului ar spori mult deoarece - înaintea fiecărui acces - ar trebui să se calculeze câte sectoare cuprinde pista care urmează să fie citită. Pentru a reduce efortul de calcul, mai multe piste sunt încadrate într-o "zonă" cu un număr egal de sectoare pe pistă. Deci procedeu este un compromis între câștigul de viteză și cel de spațiu. În figura 3 apare un exemplu de aplicare a acestui procedeu.

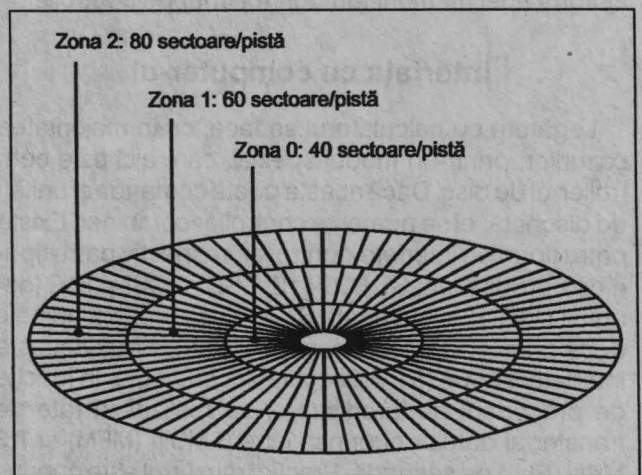


Figura 3

La Zone-Bit-Recording, pistecele centrale au mai puține sectoare decât cele periferice

În zona zero pot fi memorate cele mai puține date, în zona a doua cele mai multe. Modelul LPS 120 AT al firmei Quantum, cuprinde 16 zone cu 44 de sectoare pe pistă (central), respectiv 87 sectoare pe pistă (periferie). Astfel acest harddisk - cu înălțimea de 25,4 mm și un singur disc de 3,5" - ajunge la o capacitate impresionantă de 120 MByte. Modelul LPS 450 (2290 TPI, 50000 BPI, 37900 FCI) al aceluiaș producător - având aceleași dimensiuni, echipat însă cu 5 discuri, are o capacitate - după formatare - de 425 Mbyte. Ar putea deci fi unul dintre favoriții acestui domeniu.

Factorul Interleave

Deoarece primele harddisk-uri erau prea rapide pentru computer-ele momentului respectiv, sectoarele nu puteau fi citite succesiv. În timp ce, de exemplu, sec-

torul 1 a fost citit și evaluat, sectorul 2 a trecut deja de capul de citire și a putut fi citit doar la următoarea rotație. Pentru a evita această întârziere (ar fi fost necesare 17 ture pentru cele 17 sectoare ale pistei unui disc MFM), s-a recurs la un artificiu. Între două sectoare aflate în succesiune logică, se lasă un interstițiu. În acest interstițiu se memorează alt sector al pistei. Acest procedeu se numește *Interleave* (intercalare). *Factorul Interleave* indică numărul de sectoare fizice intercalate între două sectoare logice, și numărul de rotații necesare citirii unei piste. La un *Interleave* de 1:3, sectorul doi este la trei sectoare după sectorul 1 și pentru citirea pistei sunt necesare 3 rotații. Ideal ar fi un factor de *Interleave* de 1, la care sectoarele logice se succed nemijlocit și pista poate fi citită la o singură rotație. Combinații moderne controller-harddisk ajung la această performanță fără probleme, de exemplu cu ajutorul unei memorii-tampon (cache) pe controller.

Interfața cu computer-ul

Legătura cu calculatorul se face, ca în majoritatea cazurilor, printr-un modul special, care aici este controller-ul de disc. Dacă acesta poate comanda și unități de dischetă, el se numește controller combinat. Există patru tipuri de interfețe-controller, respectiv patru tipuri de harddisk-uri: ST 506/412, ESDI, SCSI, și IDE (denumit și AT-Bus). Interfața ST 506/412 a primit numele după primul harddisk Seagate, care a utilizat această interfață ST 506 (ST = *Seagate Technology*). În funcție de procedeu de înregistrare, se realizează rate de transfer al datelor cuprinse între 5 MBit/s (MFM) și 7,5 MBit/s (RLL) pe secundă. Practic transferul este cuprins între 200 și 600 de KByte pe secundă. Interfața ST 506/412 poate fi recunoscută după cabluri: cablul de transmisie a datelor cu 20 de fire și cablul de comandă cu 34 de fire, acesta din urmă fiind realizat din cablu-panglică. Interfața fiind serială, rata transferului de date este destul de redusă. Modelul standard este controller-ul WD 1003 al firmei Western Digital, cu care sunt compatibile multe tipuri de controller-e.

O versiune mai evoluată este *Enhanced Small Device Interface* (ESDI). Aici se lucrează cu aceleași tipuri de cabluri și, de asemenea, în mod serial. Rata de transfer se situează între 10 și 15 MBit/s, sensibil mai mare decât la ST. Atât ST 506/412 cât și ESDI sunt evoluții specifice PC-urilor și suportă maximum două harddisk-uri simultan. Interfața universală la această oră este *Small Computer System Interface*, care a fost standardizată în 1982. SCSI - sau mai evoluata SCSI-II - nu sunt numai interfețe pure pentru harddisk-uri. La ele se pot racorda scanner-e, unități CD-ROM și multe altele.

Până la 7 unități (adică 7 harddisk-uri) pot fi conectate într-o ordine arbitrară, dar ultima trebuie prevăzută

Discuri umede

Dacă distanța de 1 micrometru, dintre capul de citire-scriere și disc pare minusculă, tehnicienilor de harddisk-uri li se pare încă foarte mare deoarece, cu cât distanța devine mai mică, cu atât crește densitatea de înregistrare. Dacă distanța scade la jumătate, densitatea crește de patru ori. Se pare că firma Conner s-a apropiat considerabil de acest vis, utilizând în locul pernei de aer o peliculă de lichid (*Liquid-Disk*), metoda fiind brevetată. Această metodă are două mari avantaje: pelicula de lichid este mai subțire decât perna de aer - ceea ce duce la o creștere apreciabilă a capacității, iar pe de altă parte pelicula de ulei se manifestă și ca un tampon, între cap și disc - ceea ce înlătură pericolul impactului (*headcrash*). Deocamdată sistemul funcționează doar la dimensiuni extrem de reduse ale discului, deoarece la discuri cu diametrul mai mare viteza periferică sporită în zonele exterioare ar deteriora pelicula lichidă. Ideea provine de la o firmă pentru lubrifianți speciali, care a fost pur și simplu cumpărată de Conner. Încă nu se știe când vor apărea asemenea discuri pe piață.

cu o rezistență de capăt. Deoarece fiecare unitate are adresa ei formată dintr-un număr, datele ajung la destinație fiind transferate de la o unitate la alta, până la unitatea prevăzută. Alt avantaj este faptul că interfața SCSI este larg răspândită - se găsește și la calculatorul Next al lui Steve Jobs și la Apple Macintosh. Inteligența acestei interfețe paralele nu o posedă controller-ul fiecărei unități în parte. Spre deosebire de ST 506/412, nu controller-ul comandă procesele - ca de exemplu formatarea, ci calculatorul emite doar instrucțiunea "format harddisk", restul fiind preluat de controller-ul unității respective, în timp ce calculatorul își vede de treaba lui. De aceea ar fi mai corect să se numească modul *Host-Adapter* în loc de controller. *Host-Adapter*-ul adaptează doar semnalele electrice ale SCSI-ului și bus-ul ISA/EISA/MCA din sistem.

SCSI-interfața universală

În cazul unui harddisk, electronica de comandă se află pe unitatea de disc. Avantajul este că producătorul poate optimiza foarte mult legătura între harddisk și controller și *host-adapter*-ul nu mai are prea multă influență asupra ratei de transfer. Dezavantajul este prețul ridicat al componentelor SCSI, deoarece fiecare unitate își "cară" și propria interfață. Harddisk-urile SCSI pot fi recunoscute după cablul cu 50 de fire, cu lungimea de până la 6 m. Rata maximă de transfer este de până la 5 MBytes/secundă, valoare care practic nu este atinsă decât foarte rar. *Host-adapter*-ul SCSI cel mai răspândit este un produs al firmei americane ADAPTEC; el se numește 1542B (controller combinat) și costă în jur de 500 DM. Produsele SCSI sunt în prezent rezervate profesioniștilor datorită prețului ridi-

cat, date fiind și performanțele. Să nu trecem cu vederea nici efortul sporit necesar la instalarea SCSI-ului. De multe ori sunt necesare driver-e suplimentare - care pot provoca incompatibilitate, sau sunt necesare rezistențele terminale care au mai fost amintite. Chiar și o modificare a succesiunii unităților poate crea probleme, contrar specificațiilor SCSI. Nu se recomandă instalarea SCSI-ului de către începători, fiind necesară o oarecare experiență, dacă nu chiar profesionalism. Dar aceasta rezultă automat și din prețul ridicat.

În mod deosebit a atras atenția în ultimul timp cea mai nouă interfață, *Integrated Device Electronics* (IDE), numită adesea și *interfață AT-Bus*. Cea din urmă denumire rezultă din faptul că această tehnologie este accesibilă doar posesorilor de AT (adică de la 80286 încolo), deoarece interfața se bazează pe Bus-ul de 16 biți al AT-ului. Dar nici o regulă fără excepții: de curând a apărut adaptarea IDE pentru PC-uri cu procesoare 8086/8088, pierzându-se însă din viteză. Ca și SCSI, și IDE este o interfață paralelă și necesită doar un *host-adaptor*. Inteligența, adică controller-ul, este parte integrată harddisk-ului și poate fi acordată cu unitatea respectivă. IDE se pretează însă numai la harddisk-uri (mai nou și la streamer-e) și cuprinderea ei este redusă la două (cu adaptoare IDE mai scumpe, până la patru) discuri.

Avantajul față de SCSI este prețul redus, la performanțe comparabile (posibil 2MByte/sec, media aproximativ 1 MByte/sec). Dar nu numai discurile sunt mai avantajoase; un *host-adaptor* IDE cu controller pentru dischetă costă numai 30 DM. Nici modelele cu interfețe seriale și paralele integrate nu sunt mult mai scumpe.

În încheiere, se poate spune că viitorul este al interfețelor SCSI și IDE. Harddisk-urile ST 506/412 vor dispărea de pe piață, iar ESDI-urile vor avea dificultăți - din cauza prețului ridicat - în a se impune în fața standardelor IDE și SCSI.

Procedee de înregistrare

Cum sunt traduși pe suport magnetic biții și octeții?

Prin diferite moduri de codificare, informațiile 0 și 1 sunt transformate în particule magnetice orientate diferit; orientarea stînga/dreapta ar putea corespunde polilor magnetici N/S. În acest sens, capul de citire/scriere este prevăzut cu o bobină. Dacă această bobină este străbătută de un curent electric, ea creează un cîmp magnetic cu polaritatea în funcție de sensul curentului, care magnetizează corespunzător suprafața discului. Deoarece nici harddisk-urile nu au turația perfect constantă, trebuie memorate - în afara datelor - și așa-numitele informații de tact. Acestea asigură scrierea și citirea la aceeași viteză, com-

pensând eventualele abateri datorate variațiilor turației (imaginați-vă haosul care s-ar produce la lungimi diferite ale fiecărui sector în parte). Dacă informațiile de tact sunt înregistrate, în combinație cu informațiile memorate, pe fiecare suprafață de disc, se vorbește despre *Embedded Sector Server*, adică informațiile de sincronizare sunt cuprinse în sectoarele de date. Se poate folosi însă o față a pachetului de discuri inclusiv pentru informațiile de tact, pe când celelalte se utilizează exclusiv pentru date. Prima metodă este mai elegantă și este preferabilă. Dacă s-ar succeda alternativ biții 0 și 1, s-ar putea renunța la tactul separat, utilizând de exemplu un 1 pentru sincronizare. Problemele sunt însă șirurile de biți identici (de exemplu 000 și 11111). Aceste așa-numite curse (în engleză runs), necesită un efort considerabil de sincronizare. Modurile de codificare utilizează diferite metode de sincronizare.

La început a fost procedeul NRZ (*No Return to Zero*). Aici se schimbă magnetizarea numai dacă biții de date se schimbă, de la 0 la 1 sau invers (Fig. 4)

Dacă se succed mai mulți biți identici, nivelul curentului de înregistrare nu se modifică. La o serie de 1, curentul de înregistrare rămâne constant la nivel high (1), fără a reveni - între biți - la low (0); de aici vine denumirea procedurii. Nivelul curentului de înregistrare corespunde exact nivelului semnalului digital, deci și biților de memorat. Din păcate procedeul, deși

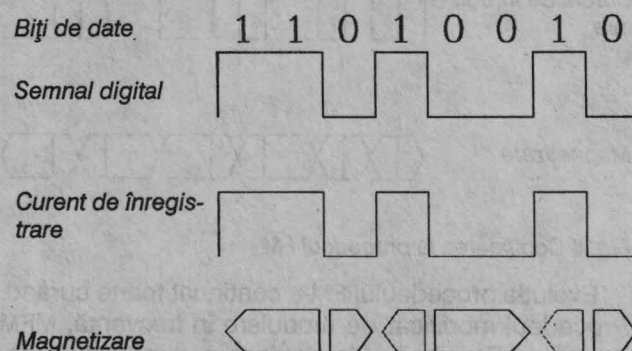


Figura 4 - Exemplu de codificare pentru procedeul NRZ

este simplu, are un dezavantaj însemnat: deoarece pot interveni șiruri de 0 sau 1 arbitrar de lungi, pretențiile referitoare la sincronizare sunt foarte mari și șirul de biți nu conține informații de tact (*Embedded Sector Server* imposibil); semnalele de tact trebuie depuse pe o suprafață separată a pachetului de discuri, blocând înregistrarea datelor pe aceasta.

Procedeul de modulare în frecvență FM (*Frequency Modulation*) a integrat pentru prima dată informațiile

Fundamente

de tact. Biții de date sunt transformați în șiruri de biți FM, deoarece trebuie adăugat tactul. Un 1 se transformă în 11, iar un 0 în 10. Prima cifră servește drept informație de tact, a doua reprezentând bitul de date. Alura curentului de înregistrare din figura 5, poate fi descrisă astfel: dacă o grupă de biți (o grupă de biți cuprinde un bit de date și eventual informații de tact) conține șirul de biți FM "10", aceasta înseamnă "păstrarea curentului"; dacă conține șirul "11" înseamnă "modificarea curentului", iar curentul se modifică în interiorul grupei. Prima cifră indică nivelul curentului la începutul grupei, a doua nivelul la sfârșitul grupei (pe linia întreruptă). După cum observați în figura 5, rezultă multe domenii magnetice scurte.

Deoarece distanța ("d"), dintre două schimbări de flux, nu poate fi arbitrar de mică, densitatea de înregistrare este limitată. În plus, procedeul consumă mult spațiu, deoarece fiecare grupă conține informații de tact. De aceea, codificarea FM se numește *Single Density Format*, adică format de înregistrare simplă densitate.

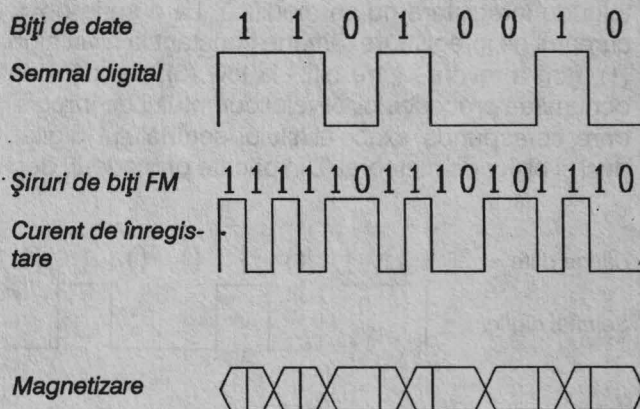


Fig. 5 Codificarea la procedeul FM

Evoluția procedeului FM a continuat foarte curând. Procedeul modificat de modulare în frecvență, MFM (*Modified Frequency Modulation*), a dublat capacitatea de memorare. Tactul se memorează numai în grupele care ele însele, sau cele precedente, nu conțin cifra 1. Un bit de date conținând cifra 1 se transformă în șirul 01 la MFM, 0 transformându-se în 00. Dacă această grupă este precedată de alta 00, al doilea bit de date cu zero-uri se transformă în 10, pentru a menține stabil tactul de citire. Înaintea fiecărei cifre de 1 din șirul MFM se produce o modificare a curentului de înregistrare, zonele diferite de magnetizare fiind mai puține și mai lungi (Fig. 6). Aici procedeul FM ar necesita 12 schimbări de flux. Pentru codificarea șirului de biți din exemplu procedeul MFM necesită doar 6, ceea ce confirmă dublarea de capacitate amintită.

În special harddisk-urile ST 506/412 utilizează această metodă, la care pe fiecare pistă sunt cuprinse 17 sectoare. Deoarece viteza acestui procedeu este redusă, producătorii au elaborat un alt procedeu.

Procedeul RLL urmărește limitarea șirurilor de zero-uri (*RLL = Run-Length-Limited*). La procedeul *RLL-(2,7)*, între doi biți "1" se află minimum doi și maximum șapte biți "0". Nou la acest procedeu este faptul că la codificare se ține seama de biții care urmează celui de codificat - adică se ține seama de context. Tabelul următor arată cum se transformă biții de date în șirul

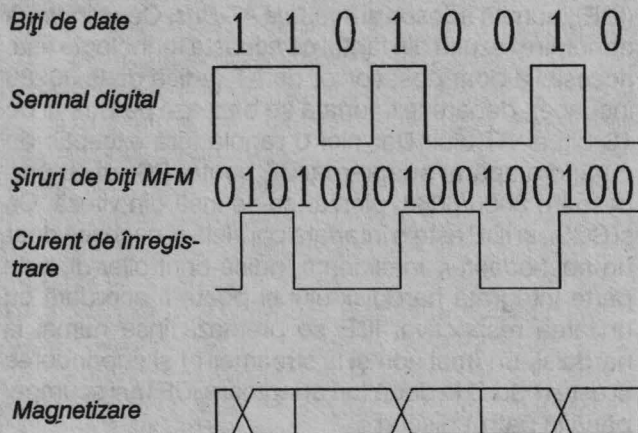


Fig. 6 Codificarea MFM reprezintă o metodă mai evoluată

RLL.

Șirul de biți din figura 7 se transformă în trei pași: 11 devine în cod RLL 010, următoarele două 010 devin fiecare 001000. Biții de date 11010010 s-au transformat în șirul RLL în: 1100>001000>001000, din cei 8 biți inițiali au rezultat 16. Iese în evidență numărul mare de biți aflați pe nivel "0". Curentul de înregistrare se modifică ca și la MFM, înaintea fiecărui bit "1" din șirul RLL. Rezultă patru grupe de magnetizare. La prima vedere, pare paradoxală afirmația că RLL economisește spațiu pentru că s-a mărit doar numărul de biți

Bit	Context	Cod RLL-(2,7)
1	0	10 00
1	1	01 00
0	00	10 0100
0	10	00 1000
0	11	00 0100
0	010	00 001000
0	011	00 100100

Tabelul reprezintă o codificare RLL-(2,7)

Despre pitici și uriași

Probabil nici o altă unitate periferică nu a avut un salt calitativ asemănător harddisk-ului. De fiecare dată când experții au prevăzut sfârșitul evoluției acestora, au fost contraziși, căci în continuare harddisk-urile devin mai mici (dimensional) și mai rapide, simultan crescând și capacitatea. Astfel ouceresc domenii rezervate până nu de mult, din motive de gabarit, memoriilor cu semiconductoare - acestea din urmă fiind și mult mai scumpe. Cel mai recent exemplu este harddisk-ul de 1,8" al firmei Conner, având formatul PCMCIA și mărimea unei cărți de credit. Având capacitatea de 32 MByte, acesta poate înlocui tradiționalele module PCMCIA-Flash-RAM, prețul pe bit fiind inferior celui al concurentului său bazat pe siliciu. Astfel harddisk-urilor li se deschide lumea Palm-top-urilor și a Handheld-urilor.

Un pas mai înainte l-a făcut Hewlett-Packard, cu al său Kittyhawk. Doar de 1,3" - adică aproximativ cât moneda de 100 de lei - și cu o greutate de doar 32 de grame, acest pitic are totuși o capacitate de 20 MBytes. Nu este de mirare că un ceasornicar, și nu Hewlett-Packard, produce această miniatură.

Caracteristicile tehnice sunt absolut comparabile cu cele ale "fraților mai mari". Discul se rotește cu turația record de 5400 rot/min, timpul mediu de acces fiind de 18 ms, iar rata de transfer de 0,9 MBytes pe secundă. În ce privește rezistența mecanică, Kittyhawk-ul depășește cu mult discurile mai mari. Hewlett-Packard vede utilizarea acestor discuri nu numai la Palm-top-uri și Notebooks, ci și la imprimante, fax-uri și copiatoare, folosindu-l drept memorie tampon.

Tehnicienii de la HP se gândesc și la alte aplicații. În jocul unității harddisk de 5,25", vor să echipeze mai multe unități de 1,3" (în spațiul rezervat celei de 5,25"-ar putea îngloba 60 de bucăți de Kittyhawk), drept



Disk-Array-System, pentru a spori securitatea datelor pentru server-ele rețelelor.

Inițial XT-ul lui IBM a fost dotat cu un harddisk de 5,25" care avea o capacitate de 10 MBytes. Discurile moderne au, la același gabarit, o capacitate de 3 GBytes, adică de 300 de ori mai mult; chiar și unitățile de 3,5" au depășit de mult limita de 1 GByte. Discurile de 8" - cândva foarte răspândite, sunt în retragere, fiind înlocuite de *Disk-Array-urii*.

de la 8 la 16; repetăm că distanța dintre schimbările de flux nu poate depăși o limită inferioară.

Acum urmează artificii propriu-zis: deoarece în cod RLL-(2,7), unui bit "1" îi urmează cel puțin doi biți "0", se poate spori densitatea de înregistrare înregistrându-se șirul "100" pe cea mai mică porțiune posibilă. Referitor la biții de date, 1,5 biți pot fi înregistrați pe porțiunea minimă de 1 bit, ca la celelate procedee. Astfel se explică economia de spațiu crescută cu 50 % a codului RLL-(2,7) față de MFM. Procedeele sunt utilizate la discuri mai recente, de tip ST 506/412 și la modelele noi IDE. Să nu vă mirați dacă citiți undeva despre RLL-(1,7), deoarece este vorba de o variantă a lui RLL-(2,7), la care "1" este urmat de un singur "0" în loc de două.

Corectare erorilor

Nici harddisk-urile nu sunt imune la erori. Pentru a evita erorile, se memorează - o dată cu datele - o sumă de control. La citirea datelor, se calculează din nou suma de control, comparându-se cu cea înregistrată pe disc. Dacă cele două sume coincid, este foarte probabil ca totul să fie în regulă. În caz contrar, harddisk-ul semnalizează controller-ului o eroare. Controller-ul comandă repetarea citirii, deoarece este posibilă

o ușoară deplasare a capului de citire-scriere (*Seek Error*). De obicei această problemă este rezolvabilă, în acest caz vorbindu-se despre *Soft-Error* - adică o eroare "ușoară". În cazul că mai multe repetări eșuează, este vorba despre *Hard-Error*. Atunci probabil că o porțiune a stratului magnetic este deteriorată și datele, din păcate, s-au pierdut. De fapt harddisk-urile au - din fabricație - asemenea defecțiuni, care sunt însă imprimate pe partea superioară a carcasei. Aceste porțiuni defecte trebuie specificate la formatarea low-level, pentru a se evita de la bun început utilizarea lor. De regulă sunt între 0 și 20 de sectoare și nu există motive de panică. Harddisk-urile IDE (care se formatează exclusiv cu instrucțiunea DOS "format") sunt prevăzute pe fiecare pistă cu unul sau mai multe sectoare de rezervă, pentru a fi utilizate în locul celor defecte.

Viteza

Un criteriu foarte important de evaluare al unui harddisk este, pe lângă densitatea de înregistrare, timpul mediu de acces. Acesta se măsoară în milisecunde și se compune din timpul de deplasare și timpul de latență. Timpul de deplasare indică perioada necesară deplasării capului de citire/scriere până în dreptul

Fundamente

pistei cerute. Timpul de latență reprezintă intervalul de timp care se mai consumă până când sectorul cerut ajunge în dreptul capului (discul se rotește continuu, probabilitatea ca sectorul să treacă chiar la momentul potrivit prin fața capului fiind foarte mică). Producătorii încearcă neîntrerupt să reducă timpul de acces. Lucrul

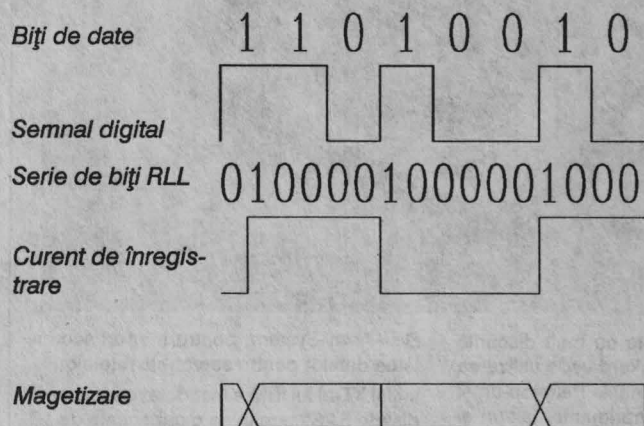


Fig. 7 Codificarea RLL $(-2,7)$ este actualmente cea mai răspândită

acesta se poate realiza printr-o deplasare mai rapidă a capetelor, deplasare care - în dreptul pistei căutate - trebuie amortizată mai intens și necesită un timp mai mare pentru a se "liniști". Se scurtează prin aceasta foarte mult *Setting-Time*-ul, astfel ca faza de amortizare să nu consume întregul câștig de timp. Timpul de latență poate fi scurtat prin mărirea turației discului deoarece cu cât turația este mai mare, cu atât sectorul căutat va fi mai repede în dreptul capului. O consecință pozitivă a măririi turației este și mărirea ratei de transfer, pista fiind citită mai rapid (se presupune un factor de interleave de 1:1). O metodă mai recentă este utilizarea unui al doilea braț cu capete de citire/scriere (citire diametral opusă celuilalt cap). La acest procedeu - dezvoltat de firma Conner - se depășește acel braț care este mai apropiat de pista căutată. În continuare, cele două brațe lucrează independent; deci unul poate, de exemplu, efectua o citire, pe când celălalt efectuează o scriere. În acest scop trebuie memorate pentru scurt timp o mulțime de date într-o memorie intermediară.

În comparație cu procesorul, circuitele RAM sunt lente, dar în comparație cu harddisk-urile ele sunt ultrarapide. De aceea este avantajos ca datele utilizate frecvent să fie citite de pe disc și memorate într-o memorie intermediară RAM, de unde se pot citi - în caz de necesitate - mult mai rapid. În acest sens, se ia în considerare *proprietatea de localizare* a programelor care spune că, pentru părți de program tocmai utilizate, probabilitatea de a fi din nou apelate, este foarte

mare. Se încearcă chiar pronosticarea datelor pe care le va solicita calculatorul la următorul acces (*Look-Ahead-Cache*). Deoarece numărul reușitelor depășește numărul ratărilor datorate unor algoritmi sofisticăți, memoriile cache aduc o sporire a vitezei și în cazul accesului și în ce al ratei de transfer. Harddisk-urile moderne au integrate, de obicei, memorii cache de 32 până la 512 KByte, împărțite în mai multe segmente, și toate accesesele de citire (mai nou și cele de scriere) sunt trecute prin această memorie-tampon. Cea mai scumpă soluție în domeniul harddisk-cache-urilor a fost și va fi *cache-controller*-ul (mai corect spus *host-adapter*-ul, la IDE și SCSI). Se vorbește că RAM-Cache-urile multor modele 386 sau 486 vor fi realizate hardware, complet transparent pentru soft, adică insesizabil pentru soft.

Majoritatea acestor controller-e posedă propriul procesor, care preia întregul transfer de date dinspre și înspre disc.

La acestea se poate folosi un Cache de 512 KBytes până la 16 MBytes. Procesorul integrat pe modul preia întreaga muncă a CPU-ului, având acces direct la memoria calculatorului. Acest lucru poate crea probleme dacă datele din cache nu corespund cu cele din memoria de lucru, adică nu este asigurată consistența datelor. Cu cât cache-ul este mai mare, cu atât paguba este mai mare în cazul căderilor de tensiune a rețelei.

Mai avantajoase - și de multe ori și mai rapide - sunt programele cache. Soluțiile exclusiv software utilizează o parte a memoriei de lucru pentru cache. Cu toate că la programele cache CPU-ul preia din nou comanda, viteza este totuși mai mare decât la controller-e cu procesor propriu. Motivul este foarte simplu: se pierde timp cu procedeele de *soft-cache*, în schimb cache-controller-ul trebuie să transporte date cu ajutorul *I/O-Bus*-ului. Din păcate, Bus-ul are un tact de numai 8 MHz (mai rar până la 12 MHz), pe când accesul la memoria de lucru, se realizează cu frecvența de tact a microprocesorului. În altă ordine de idei, în caz de incompatibilitate, programele cache pot fi deconectate fără probleme, ca să numai vorbim de prețul lor redus. Aici sunt suficiente un spațiu rezervat în RAM și unul din programe. Prețurile componentelor RAM fiind în scădere, multe programe cache pot fi procurate pe piața Shareware (de exemplu *Hyperdisk*).

În afară de acestea, începând cu varianta 5.0 a lui MS-DOS, se livrează și cache *smartdrive*-uri care conlucrează optim și cu Windows. În ansamblu, aceasta este soluția rapidă și accesibilă ca preț indicată pentru sisteme la domiciliu.

Matthias Günter
Traducere: ing. Mihai Beer

Organizarea harddisc-urilor

Structuri de discuri

O componentă nelipsită a oricărui PC este harddisk-ul. În această privință, merită să aruncați o privire în culise. Cel puțin atunci când - din greșală - ștergeți un fișier sau când apare o eroare de citire, informațiile suplimentare pot fi de mare folos.

Un harddisk tipic este compus dintr-un pachet de discuri, pe care sunt înregistrate magnetic informațiile în fișiere. Discurile sunt acoperite cu un film subțire, conținând particule metalice sau de oxid. Intrând sub acțiunea unui câmp magnetic, acestea se orientează după una sau două direcții. Suprafața fiecărui disc este împărțită în inele concentrice (cilindri). La rândul lor, cilindrii sunt divizați în zone de memorare numite sectoare. La majoritatea PC-urilor, un sector ocupă 512 Bytes.

Când se vorbește despre tehnologia harddisk-urilor, cel mai frecvent cuvânt este "capete". Numărul capetelor de citire/scriere este egal cu cel al fețelor pe care se face înregistrarea datelor. La majoritatea harddisk-urilor numărul de capete este dublul celui al discurilor. Așa-numiții cilindri formează, începând de la centrul discului până la periferia sa, o serie de inele

înguste. Sectoarele sunt segmente ale acestora, în cadrul fiecărui cilindru. Capacitatea în octeți a harddisk-ului se calculează în felul următor:

capacitate = număr de capete x număr de cilindri x număr de sectoare pe pistă x 512.

Înainte de a putea inițializa un harddisk, în Sistemul MS-DOS, trebuie să fie îndeplinite trei condiții. În primul rând discul trebuie formatat *low-level*. Acest mod de formatare stabilește poziția sectoarelor și cilindrilor pe disc. În al doilea rând, harddisk-ul trebuie partiționat, adică împărțit în mai multe unități logice, care în MS-DOS sunt adresate separat, fiind și tratate separat ca unități fizice de sine-stătătoare. În al treilea rând, harddisk-ul trebuie formatat *high-level*. Acest procedeu generează structuri de date, cum ar fi FAT-ul (File Allocation Table) sau directorul principal. Informațiile despre sectoare (ID) sunt înregistrate în sectorul *ID-Header*. Acesta conține, de obicei, numărul capului, numărul sectorului, numărul de cilindri (piste) și adresa sectoru lui cu informațiile despre acestea. În afara de acestea, aici mai apar teste pentru depistarea erorilor la nivel de Header. Controller-ul harddisk-ului folosește aceste informații pentru căutarea într-un anumit sector.

Offset-ul Înregistrărilor din tabelul de partiții

	00hex	01hex	02hex	03hex	04hex	05hex	06hex	07hex	08-0Bhex	0C-0Fhex
01BEhex	B1	H	S	C	S1	H	S	C	SP	N
01CEhex	B1	H	S	C	S1	H	S	C	SP	N
01DEhex	B1	H	S	C	S1	H	S	C	SP	N
01EEhex	B1	H	S	C	S1	H	S	C	SP	N

Primul cap, sector și cilindru

Ultimul cap, sector și cilindru

Mărimea partiției în sectoare

Simbol	Semnificație
B1	boot indicato
00hex	=inactiv
80hex	=partiție acivă
H	număr cap
S	număr sector
C	număr cilindru
S1	System Indicator
00hex	=neutilizat
01hex	=MSDOS(până la 16 MByte)
04hex	=MSDOS(16 până la 32 MByte)
05hex	=partiție DOS extinsă
06hex	=MSDOS(peste 32 MByte)
07hex	=OS/2 High Performance File System)
SP	număr de sectoare anterioare acestei partiții
N	mărimea partiției

Fig. 1. Formatul tabelului principal de partiții

Fundamente

Numerotarea se va face funcție de diferenții factori care vor influența ulterior, modul de utilizare și timpul de acces al harddisk-ului. Unul din acești factori este cel de *Interleave*. Acesta adaptează turația discului la viteza cu care controller-ul poate transfera date înspre sau dinspre harddisk. Procedeu interleavé separă sectoare ale discului numerotate crescător, astfel încât sectoarele 2 și 3, de exemplu, se pot afla la o distanță de 4 sectoare între ele. Sectoarele trebuie dispuse astfel deoarece nu toate computerele reușesc, la viteza de rotire a discului, să citească un sector, să înregistreze datele în memorie, fiind și pregătite pentru citirea următorului sector. Dacă pentru aceasta controller-ul trebuie să aștepte o rotație completă a discului, se pierde timp. Dacă sectoarele sunt depărtate, se poate stabili distanța între două sectoare care se succed logic, astfel încât capul de citire să se afle în dreptul sectorului următor exact în momentul în care controller-ul este pregătit pentru un nou transfer de date. Acest procedeu garantează viteza de transfer maximă. Formatarea low-level marchează pe disc și sectoarele deteriorate, care nu vor mai putea fi folosite pentru memorarea datelor.

MS-DOS permite împărțirea harddisk-ului în partiții. Acestea sunt tratate ca harddisk-uri separate și sunt adresate separat. Există trei tipuri de partiții: partiția primară DOS, partiții Extended-DOS și partiții non-DOS. O partiție primară DOS este prima partiție DOS aplicată pe harddisk, și este singura din care poate fi lansat DOS. Ea este asimilată unei unități fizice independente. În MS-DOS un harddisk poate avea maximum patru partiții.

Versiunile MS-DOS de la 3.3 în sus sprijină partițiile Extended-DOS, care permit împărțirea harddisk-urilor mai mari în unități logice. Partițiile DOS pot fi împărțite în atâtea unități logice câte permite spațiul pe harddisk. La un sistem tipic, cu un singur harddisk și o partiție primară DOS (unitatea C), o partiție extinsă DOS poate cuprinde până la 23 de unități logice (de la D la Z). Așa-numitele partiții non-DOS pot fi utilizate de alte sisteme de operare, cum ar fi Novell Netware. În aceste cazuri, MS-DOS nu recurge la aceste partiții. MS-DOS 4.0 și 5.0 permit unități logice de până la 2 GByte. La această capacitate, o unitate fizică poate fi împărțită în până la 24 de unități logice, ceea ce simplifică foarte mult lucrul. La partiționarea unui harddisk, MS-DOS începe instalarea cu octetul 1BE hex (446 în zecimal) din tabelul de partiții principal. Formatul acestui tabel este reprezentat în figura 1.

Acest tabel conține patru înregistrări de câte 16 biți, fiecare înregistrare fiind împărțită în 10 câmpuri de lungimi diferite. În aceste câmpuri sunt înscrise toate informațiile necesare descrierii unei partiții a harddisk-ului:

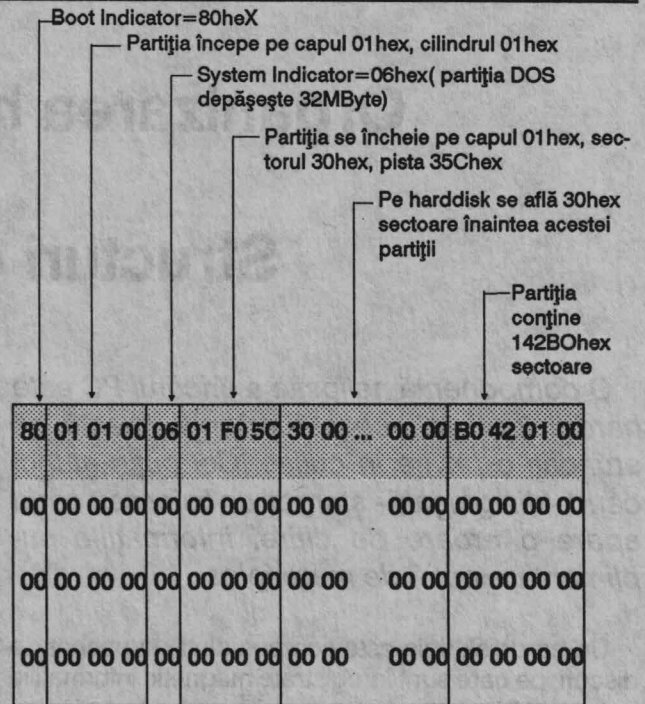


Fig. 2. O serie de înregistrări într-un tabel de partiții.

- capul, sectorul și cilindrul
- începutului și sfârșitului partiției
- mărimea partiției, în sectoare
- informații despre modul de formatare și sistemul instalat
- informația dacă partiția de pe care pornește sistemul este cea activă

Nu este necesar ca partițiile să aibă o anumită mărime sau poziție pe disc. Octeții sectoarelor și pistelor nu pot fi interpretați direct ca valori de 8 biți. Cei 6 biți inferiori indică numărul actual al sectorului (0 până la 63), pe când cei 2 biți superiori - împreună cu cei 8 biți ai câmpului cilindrului (pistei), generează numărul acestuia, folosind 10 biți (0 până la 1023). Deci MS-DOS "vede" harddisk-uri cu 1024 de piste a câte 64 de sectoare; harddisk-urile care nu pot atinge aceste limite, trebuie "ajutate" de driver-e speciale, sau echipate cu controller-e care să facă adaptarea la limitele stabilite pentru sectoare și cilindri. Bitul indicator al sistemului (offset 04 hex), indică sistemul instalat pe partiția respectivă și, în cazul partițiilor DOS, și tipul partiției. Partițiile DOS primare sunt marcate, funcție de mărime, cu 01hex, 04hex sau 06hex. În mod normal partițiile sub 16 MByte se notează cu 01hex, cele între 16 și 32 MByte cu 04 hex, iar cele peste 32 de MByte cu 06hex. Valoarea 05hex marchează o partiție Extended-DOS. Toate celelalte valori pozitive marchează partiții non-DOS. Număul 00hex indică o partiție încă

nedefinită. Partiția activă în sectorul Boot, conține valoarea 80hex. Este permisă existența unei singure partiții cu această valoare. Dacă PC-ul "se trezește" pe harddisk, un program auxiliar scurt citește în RAM sectorul care conține tabelul principal al partițiilor, executând apoi un alt program scurt, aflat la începutul acestuia. Acest program caută în tabel valoarea 80hex, în câmpul Boot Indicator. Primul sector cu informația căutată este transferat în memoria de lucru, acesta instalând o rutină de execuție (Bootstrap-Loader) care încarcă MS-DOS (sau un alt sistem conținut în partiție) de pe harddisk. Dacă un harddisk urmează să conțină mai mult decât o unitate logică, trebuie instalată o partiție Extended-DOS. Diferența dintre partiția DOS primară și Extended-DOS, constă în faptul că cea din urmă poate fi divizată în mai multe unități logice, fapt nepermis la partiția DOS primară. Numerotarea pentru primul cap, primul sector și primul cilindru, ale unei partiții Extended-DOS, nu marchează poziția unei partiții fizice, ci indică alt tabel de partiții, care descrie un volum logic. Dacă în partiția Extended-DOS sunt definite unități logice suplimentare, a doua înregistrare în acest pseudo-tabel conține adresa altui tabel, în care sunt cuprinse informațiile referitoare la următoarea unitate logică. Tabele de partiții se pot succeda în numărul dorit, pentru descrierea tuturor unităților logice. fig 2

Hexdump-ul din figura 2 reprezintă seria de înregistrări dintr-o tabelă de partiții aparținând unui harddisk de 40 MByte, care cuprinde o partiție DOS primară de 20 MByte și o partiție Extended-DOS de 20 MByte. Aceasta din urmă este împărțită în două unități logice de câte 10 MByte.

Tabelul principal de partiții conține inscripții pentru o partiție de tip 04hex și pentru una de tip 05hex. Numerotarea primului cap, sector și cilindru se înscrie în înregistrarea făcută pentru Extended-DOS. Programul care citește sectorul care începe astfel: "cap 00hex, sector 41hex și pistă ABhex", găsește tabelul de partiții nr.2. Aici apar două înregistrări: una pentru o partiție de tip 01hex și una pentru o partiție de tip 05hex. Partiția de tipul 01hex este prima unitate logică a lui Extended-DOS. Dacă programul citește primul sector al partiției de tip 05hex, găsește un al treilea tabel de înregistrări, cu o singură înregistrare de tip 01hex. Această partiție este cea de-a doua unitate logică.

După ce harddisk-ul a fost partiționat, diferitele volume pot fi formate high-level. După formare, fiecare volum MS-DOS are, în esență, următoarea structură:

- sector Boot
- File Allocation Table (FAT)
- una sau mai multe copii ale FAT-ului

- director principal
- zone pentru date și subdirectoare.

Boot-Sector-ul se instalează în primul sector al unui volum, pentru a fi ușor găsit de către MS-DOS. Tabelul 1 descrie structura sa. El conține informații referitoare la mărimea și structura volumului, componenta cea mai importantă fiind Bootstrap-Loader-ul care încarcă MS-DOS-ul. După pornirea PC-ului, BIOS-ul încarcă în memorie primul sector fizic al harddisk-ului și execută codul respectiv, adică încarcă sistemul de operare MS-DOS. La începutul sectorului Boot se află o instrucțiune JMP din codul-program propriu-zis, care începe la adresa 1Ehex. La versiunea Short, care cuprinde doar 2 MByte, se adaugă în cod o instrucțiune NOP pentru a umple spațiul până la următoarea înregistrare. În câmpul în care apare producătorul se poate înscrie orice, deoarece aceste informații nu se valorifică. Următoarele înregistrări se referă la parametri fizici ai volumului, și sunt valorificate de către funcțiile întreruperii 13hex. Aceste funcții utilizează driver-e de unități sau programe utilitare pentru citirea și înregistrarea datelor. De aceea aceste înregistrări se mai numesc și *Bloc BIOS de parametri*. Rutina Boot se poate extinde peste mai multe sectoare dacă cei 482 Byte rezervați nu sunt suficienți. Apoi acești primi 482 de Bytes trebuie reîncărcați de către prima parte a programului. În acest caz înregistrarea conține, începând cu adresa 0Ehex, numărul sectoarelor în cauză, inclusiv sectorul Boot. La toate versiunile DOS până la 5.0 spațiul din sectorul de Boot este suficient pentru Bootstrap-Loader, astfel încât înregistrarea cu adresa 0Ehex are conținutul "1". Ca să puteți crea un nou fișier, MS-DOS trebuie să știe care sectoare ale volumului mai sunt libere. Aceste informații sunt cuprinse în File Allocation Table (FAT), aceasta urmând nemijlocit sectorului Boot (respectiv unor sectoare rezervate). FAT conține numeroase înregistrări, care corespund unor grupuri de sectoare logic succesive, denumite *cluster-e*. Pentru a afla câte sectoare compun un cluster, se citește înregistrarea de la adresa 0Dhex în sectorul Boot. La un harddisk de tip AT, un cluster este compus, de regulă, din patru sectoare, exceptând cazul când volumul depășește capacitatea de 128 MByte. Deoarece în FAT sunt rezervați maximum 16 biți pentru înscrierea unui cluster, pot fi gestionate maximum 65536 cluster-e. La patru sectoare pe cluster și 512 Bytes pe sector, rezultă exact acei 128 MByte ca mărime maximă a unui volum. MS-DOS gestionează volume mai ample prin faptul că, în funcție de mărimea volumului, cuprinde un număr crescut de sectoare într-un cluster. Astfel la măriri între 128 și 256 MByte, 8, între 256 și 512, 16, între 512 MByte și 1 GByte, 32, iar între 1 GByte și 2 GBytes, 64 de sectoare sunt cuprinse într-un cluster. Genialitatea acestui procedeu constă în faptul că structura

Fundamente

FAT nu este influențată și în plus, ea poate fi gestionată în toate cazurile, de aceeași rutină. Formând cluster-ele, se evită o fragmentare excesivă a fișierelor. În

Mărimea inscripțiilor în cadrul FAT depinde, începând cu versiunea 3.0 a lui DOS, de numărul cluster-elor de administrat. Până la 4096 de cluster-e

0	Media-Descriptor	F8FF
1	octeți "lest"	FFFF
2		
3		000A
4		0006
5		
6		0007
7		000D
8		
9		
A		FFF8
B		
C		
D		0003
E		
F		

Intrarea în director a fișierului DEMO.EXE

00	D	E	M	O
04	32	32	32	32
08	E	X	E	0
0C				
10				
14			B7	5E
18	B3	18	4	0
1C	70	13	0	0

Fig. 3. Înlănțuirea inscripțiilor FAT, exemplificată

schimb acest lucru nu poate fi evitat la lucrul cu un disc pe care se creează și de pe care se șterg foarte des fișiere. Atâta timp cât volumul este proaspăt creat, fișierele pot fi scrise pe disc unul după altul. După prima ștergere, golul produs trebuie completat. Dacă fișierul nou este mai mare decât cel vechi, trebuie fragmentat cel puțin în două părți. Ca exemplu vom folosi un caz extrem, fără formare de cluster-e, la care un fișier de 100 de sectoare, este fracționat în 100 de sectoare fizic distanțate între ele. Întârzierile care intervin numai din poziționarea de 100 de ori a capului de citire sunt inadmisibile. Deci prin formarea cluster-elor se garantează totuși faptul că, patru sectoare fiind citite consecutiv, capul trebuie poziționat numai de 25 de ori. Câștigul de viteză este cu atât mai mare, cu cât numărul de sectoare pe cluster este mai mare. Împărțirea pe cluster-e are însă și un dezavantaj: cu creșterea numărului de sectoare pe cluster crește risipa de spațiu. Imaginați-vă ce se întâmplă numai la un fișier de 100 de Bytes, cum ar fi "config.sys". Fără cluster-e, se risipesc 412 Bytes, deoarece un sector cuprinde 512 Bytes. Cu patru sectoare pe luster, se risipesc deja $3 \times 512 + 412 = 1948$ iar cu 64 de sectoare/cluster chiar $63 \times 512 + 412 = 32668$ Bytes.

cer 12 biți, mai multe cer 16 biți. Deoarece la 4 sectoare pe cluster și 4096 cluster-e se gestionează doar 8 MByte, la ora actuală FAT-urile sunt, aproape în exclusivitate, de 16 biți. Pentru a determina numărul de biți pe inscripție FAT, trebuie doar împărțită valoarea de la adresa 13hex (total sectoare pe volume) din Boot-sector, la inscripția de la adresa 0Dhex (sectoare/cluster). Dacă rezultatul este mai mic sau egal cu 4096, este vorba de inscripții de 12 biți, în caz contrar de inscripții de 16 biți. Primele două inscripții FAT sunt rezervate și cuprind 3 octeți (12 biți) sau 4 octeți (16 biți). Primul byte conține așa-numitul *Media-Descriptor*, care furnizează informații referitoare la formatul volumului și suportul datelor. În principiu, la harddisk-uri are valoarea F8hex. Următorii doi sau trei octeți conțin valoarea de "lest": 225.

Sectoarele unui fișier din cadrul FAT sunt dispuse într-o listă înlănțuită, și se evaluează astfel: în inscripția de director a fișierului respectiv, începând cu adresa 1Ahex, se află numărul primului cluster; inscripția FAT corespunzătoare conține numărul următorului cluster; la rândul ei inscripția FAT a acestuia corespunde, din nou, numărului următorului cluster, etc., până la inscripția FAT corespunzătoare ultimului cluster, care cuprinde marcarea sfârșitului. În cazul unui FAT de 16

Adresa	Conținut	Lungime și tip
+00hex	Instrucțiune de salt la rutina Boot	3octeți
+03hex	Producător și număr al versiunii	8 octeți
+0Bhex	Octeți pe sector	1 cuvânt
+0Dhex	Sectoare pe cluster	1 octet
+0Ehex	Numărul sectoarelor rezervate	1 cuvânt
+10hex	Numărul de File Allocation Tables(FATs)	1 octet
+11hex	Număr de inscripții în directorul principal	1 cuvânt
+13hex	Număr de sectoare în volum	1 cuvânt
+15hex	Media-Descriptor	1 octet
+16hex	Număr de sectoare pe FAT	1 cuvânt
+18hex	Sectoare pe cilindru	1 cuvânt
+1Ahex	Număr de capete	1 cuvânt
+1Chex	Distanța la care se află primul sector din volum față de primul sector pe suportul de memorie	1 cuvânt
+1Ehex-1FFhex	Rutina Boot	482 octeți

Tabelul 1 Conținutul unui sector Boot

biți, valorile sunt între FFF8hex și FFFFhex. Un cluster cu erori se marchează cu codul FFF7hex, iar unul liber cu 0000hex. Dacă este vorba despre de un cluster rezervat, de exemplu unul pentru FAT însuși, inscripția FAT corespunzătoare are valoarea FFF0hex până la FFF6hex. Figura 3 reprezintă, printr-un fișier fictiv "de-

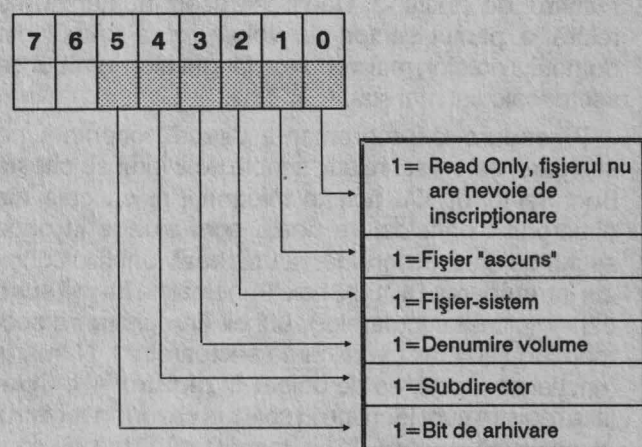


Fig. 4. Attributbyte-ul definește tipul inscripționării directorului

mo.exe" cuprinzând 5 cluster-e, înlanțurarea inscripțiilor FAT.

Deoarece, în principiu, în caz de deteriorare a FAT-ului se distruge toate datele (datele există ca atare în sectoare, dar lipsește contextul), MS-DOS permite programului de formatare crearea uneia sau mai multor copii ale FAT-ului, administrate în paralel de MS-DOS. Dacă se distruge FAT-ul primar, acesta poate fi înlocuit folosind instrucțiunea CHKDSK împreună cu una din copii.

În cadrul fiecărui volum, FAT este urmat de directorul principal. Acesta se compune din mai multe sectoare, al căror număr este consemnat în Boot-sector începând de la adresa 11hex. Fiecare sector conține până la 16 inscripții de fișiere de câte 32 de Bytes. Deoarece mărimea sectorului principal este stabilită prin formatare, ea nu poate fi extinsă dinamic. Acesta este motivul pentru care este mai bine să se lucreze cu mai multe subdirectoare. Tabelul 2 prezintă structura unei inscripții de director.

Primii 8 Bytes conțin denumirea fișierului; în caz de necesitate se umple spațiul cu blank-uri. Interesant este procedeul de ștergere a unui fișier. Datele nu sunt șterse fizic, ci este modificat doar primul octet al denumirii fișierului în E5hex. Astfel se recunoaște, la crearea unui nou fișier DOS, că inscripția respectivă poate fi modificată. Dacă utilizați UNDELETE sub MS-

Adresă	Conținut	Lungime și tip
+00hex	Denumirea fișierului	8 octeți
+08hex	Extensia fișierului	3 octeți
+0Bhex	Atributul fișierului	1 octet
+0Chex	Rezervat	10 octeți
+16hex	Ora	1 cuvânt
+18hex	Data	1 cuvânt
+1Ahex	Primul cluster al fișierului	1 cuvânt
+1Chex	Mărimea fișierului	1 dublu cuvânt

Tabelul 2 Structura unei inscripții în director

DOS 5.0, programul vă solicită prima literă din denumirea fișierului și apare: "a fost modificat în E5hex". Atâta timp cât nu înregistrați un alt fișier în locul celui vechi, acesta din urmă poate fi reconstituit. Dacă primul simbol este "E5hex", MS-DOS îl înlocuiește cu 05hex. Rămâne un secret al creatorilor lui MS-DOS de ce nu au ales direct codul 05hex pentru fișiere șterse. Sfârșitul directorului principal este marcat de un octet "zero". Octetul atribut (fig 4) definește tipul inscripției în director.

Interesant este bitul de arhivare care la fiecare acces de scriere, este adus la "1". Un program de salvare a datelor cum este, de exemplu *backup* îl poate readuce la zero, iar o "salvare" în continuare poate stabili dacă, de la ultimul acces, fișierul a fost sau nu modificat; dacă da, atunci bitul de arhivare ia din nou valoarea "1".

În final se poate determina mărimea fișierului, cu relația:

$$\text{mărime} = \text{Word1} + 65536 \times \text{Word2},$$

"Word1" reprezentând cei 16 biți inferiori, iar "Word2" biții superiori din Dword-ul de la adresa 1Chex din director.

Dacă bit-ul 4 din octetul atribut este ocupat, inscripția din director este un subdirector. Acesta este gestionat dinamic de FAT (ca și datele de altfel) astfel încât, teoretic, poate deveni nelimitat de mare. La crearea unui subdirector MS-DOS rezervă inițial doar un cluster. Pe parcursul extinderii, se generează o înlănțuire de cluster-e (cum s-a explicat mai sus) gestionate de FAT. Inscripția din directorul principal indică - exact ca la fișiere - primul cluster. Inscripțiile din subdirector au aceeași structură cu cele din directorul principal. Fiecare subdirector conține semnele: "." pentru directorul actual, și ".." pentru cel ierarhic superior. Aceste inscripții nu pot fi șterse cu comanda `del`.

Domeniul propriu-zis al datelor, care cuprinde și subdirectoarele, începe după directorul principal. Numărul primului sector logic al domeniului datelor poate fi determinat prin următorul calcul:

Nr. primului sector de date = nr. sectoare rezervate + nr. sector FAT x nr. FAT-uri + nr. de sectoare ale directorului principal

Cei trei termeni pot fi determinați pe baza inscripțiilor din sectorul de Boot astfel: numărul de sectoare din directorul principal se determină prin împărțirea la 32 a numărului de inscripții (adresa 11hex), numărul de sectoare FAT prin înmulțirea dintre inscripția de la adresa 10hex cu cea de la adresa 16hex, iar numărul sectoarelor rezervate se găsește la adresa 0Ehex. Primul sector al unui fișier anume îl puteți determina în felul următor: se reduce cu două numărul de cluster-e, deoarece primele două inscripții conțin Media-Descriptorul și "lest-ul", iar rezultatul se înmulțește cu numărul de cluster-e. Deoarece acest număr nu se referă la primul sector din volum, ci la primul din domeniul datelor, mai trebuie adăugat aici numărul de sector calculat mai sus.

Procedura la programarea directă, orientată pe sectoare, este deci relativ simplă. Mai întâi se citește Boot-Sector-ul. Cu aceste informații și cu cele ale directorului principal se poate apoi ajunge la orice sector de date. Programarea utilizează funcțiile 00hex din întreruperea BIOS 13hex. Principalele funcții sunt: 02hex (citirea sectoarelor), 03hex (înregistrarea sectoarelor) și 04hex (verificarea sectoarelor). Numărul funcției se predă, ca de obicei, în registrul AH. Celelalte registre sunt identice în cele trei cazuri. În registrul AL se predă numărul de sectoare (1 până la 128) care trebuie citite (înregistrate, verificate). În registrul DL trebuie să apară numărul harddisk-ului (80hex pentru primul instalat pe PC, 81hex pentru al doilea), iar în DH

numărul capului de înregistrare/redare. Registrul CH preia numărul cilindrului iar CL numărul primului sector care urmează a fi citit (înregistrat, verificat).

Aici iese în evidență faptul că un harddisk poate avea mai mult decât 256 de cilindri, chiar dacă într-un registru de 8 biți pot fi înregistrate doar numere de la 0 la 255. Deoarece nici un cilindru nu conține mai mult de 63 de sectoare, biții 7 și 8 păstrează valoarea "0". De aceea le-au fost alocați biții 8 și 9 ai unui număr de cilindru format din 10 biți. Pentru numărul de cilindru, lucrurile stau astfel:

Biții 0- 7: registrul CH, biții 0- 7

Biții 8- 9: registrul CL, biții 6- 7

Perechea de regiștri ES:BX trebuie să indice întotdeauna memoria-tampon a datelor (buffer-ul). Iată două exemple în acest sens. În primul exemplu ur-

```
data      db 512                ; buffer date
start:    mov  ah,02             ;cod pentru citire sector
          mov  al,20             ; se citesc 20 de sectoare
          mov  dl,80h           ; discul Nr. 1
          mov  dh,02            ; capul Nr. 2
          mov  ch,231           ; bitii 0-7 ai numarului de cilindru 999
          mov  cl,128+64+3      ; bitii 8+9 ai nmarului de cilindru 999
          ; si numarul sectorului
          mov  ds,es            ; segmentul bufferului de date
          mov  dx,offset data    ; offsetul bufferului de date
          int  13h; apelarea functiei
```

Listing 1. Un program care citește 20 de sectoare de pe hard-disc.

```
data      db 512                ; buffer date
start:    mov  ah,02             ;cod pentru scriere sector
          mov  al,10            ; se scriu 10 sectoare
          mov  dl,81h           ; discul Nr. 2
          mov  dh,05            ; capul Nr. 5
          mov  ch,44            ; bitii 0-7 ai numarului de cilindru 300
          mov  cl,0+64+20       ; bitii 8+9 ai nmarului de cilindru 300
          ; si numarul sectorului
          mov  ds,es            ; segmentul bufferului de date
          mov  dx,offset data    ; offsetul bufferului de date
          int  13h; apelarea functiei
```

Listing 1. Un program care scrie 10 sectoare de pe hard-disc.

mează să fie citite 20 de sectoare, începând cu cilindrul 999, sectorul 3, și capul 2 al primului harddisk instalat (listing 1). În exemplul al doilea se înregistrează cu date din memoria-tampon 10 sectoare, începând cu cilindrul 300, sectorul 20, capul 5 al celui de-al doilea harddisk (listing 2). Dacă operația decurge fără greșală, toate funcțiile întorc un Carry-Flag poziționat în 0. Dacă apare o eroare, Carry-Flag este poziționat în 1, iar registrul AH conține codul numeric al erorii.

Frank Riemenschneider

Traducere și adaptare: ing. Mihai Beer

Medii de memorare optice

În așteptarea MO-urilor

Pe lângă mediile de memorare bazate pe feromagnetism de până acum, au fost expuse pe piață, în ultimul timp, și procedee de înregistrare magneto-optice (MO). În valul de răspândire tot mai largă a aplicațiilor Multimedia, cercetătorii pieței pronosticează mediilor de memorare optice perspective demne de luat în seamă.

Ca medii de memorare optice, se face deosebirea între CD-ROM-uri, WORM-uri și unități MO. În timp ce CD-ROM-urile reprezintă o memorie care poate fi doar citită, un WORM (*Write Once Read Multiple*), poate fi înscris o dată. Unitatea MO se bazează, dimpotrivă, pe înregistrarea magneto-optică, datele putând fi salvate sau șterse de câte ori se dorește.

CD-ROM - o memorie care poate fi numai citită

CD-ROM-ul este primul apărut dintre mediile optice, care a cunoscut o răspândire largă în domeniul neprofesionist. Aceasta rezultă, pe de o parte din prețurile hardware care scad puternic, iar pe de alta din numărul de aplicații care le susține, aflat în continuă creștere. Ca și CD-ul audio, CD-ROM-ul este placat numai pe o parte. Suprafața acestui mediu de memorare constă dintr-un strat de policarbonat artificial, un strat de aluminiu reflectant și un strat de lac de protecție. Pe suprafața plăcii se trage o urmă spirală, asemănător discurilor audio normale, chiar în momentul fabricării. Cu ajutorul unui laser, se ard niște adâncituri, în spirala acesteia, la anumite distanțe. Spre deosebire de discurile audio, aici spirala se desfășoară din interior spre exterior. Urmă spirală se potrivește foarte bine pentru înregistrarea secvențială a cantităților mari de date. Procedeele permite producătorilor folosirea aceluiași instalații pentru producerea CD-ROM-urilor, ca și cele utilizate la producerea CD-urilor audio.

Oricum, avantajul acesta este anulat de timpul de acces crescut. Urmele concentrice (gen harddisk), permit un acces mai rapid, deoarece sectorul corespunzător este mai ușor de localizat, dată fiind echidistanța unei urme față de mijlocul discului. Acest procedeu este esențial mai complex la o urmă spirală. În procesul de citire, urma este explorată de o rază laser foarte concentrată. Dacă raza nimereste într-o adâncitură, urmează o absorbție; dacă nu, raza este reflectată și detectată de o foto-celulă. Astfel datele sunt memorate binar, așa cum este cunoscut de la mediile de până acum. Dacă raza este reflectată starea este 0, dacă însă are loc o absorbție, starea este 1. Adânciturile sunt desemnate drept *Pits*, iar locurile care rămân neschimbate, drept *Lands*. Timpii de acces ai unei unități CD-ROM se află undeva între 300 și 400 de ms. O îmbunătățire în acest sens, a fost obținută prin procedeele *Continuos-Read*. Aici sunt citite deja date, în tampon, în timp ce datele citite mai înainte sunt transferate computerului, prin magistrală. La tehnologia CD-ROM, cerințele în ce privește securitatea datelor sunt esențial mai mari, decât la CD-urile audio. Mai mult de 10%, din spațiul de memorare

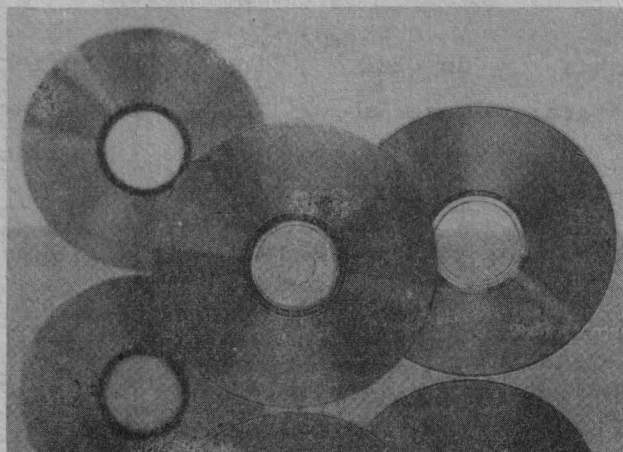


Fig 1: Capacitatea CD-ROM-ului este între 540 și 630 MBytes per disc, ceea ce corespunde la 270.000 până la 317.000 de pagini A4 scrise

disponibil este ocupat cu gestiunea datelor și testarea erorilor. Un avantaj al CD-ROM urilor este costul de fabricație redus. Dacă se iau în considerare numai costurile de multiplicare, la o serie mică, producerea unui pachet-program care cuprinde numai câteva dischete, este mai scumpă decât producția CD-ROM. Un alt avantaj constă în faptul că în orice unitate CD-ROM pot fi citite, în mod normal, și CD-uri audio. Rularea CD-ului audio se face aici sub controlul computerului. Se pot obține și unități CD-ROM externe, care oferă confortul de operare al aparaturii CD audio, iar utilizând un acumulator, pot fi folosite ca CD-Player mobil. Parametrii de capacitate ai acestor unități sunt însă, de regulă, sub cei ai unităților care au fost concepute numai pentru CD-ROM. Pe baza formatului de înregistrare și a vitezei de rotație, rata de transfer a datelor este aproape identică la toate unitățile CD-ROM. Ca domeniu de implementare, fără îndoială că CD-ROM-ului i se oferă, distribuirea software-ului. Aplicațiile zilelor noastre devin tot mai cuprinzătoare, un număr crescut de dischete-HD de instalare, nemaifiind o raritate. Deci, ce poate fi mai indicat decât oferirea soft-ului pe CD-ROM-uri? Cu acestea pot fi rezolvate mai multe probleme, dintr-o lovitură: pe de-o parte CD-ROM-ul oferă, față de purtătoarele magnetice de date, o siguranță a datelor fundamental mai mare, astfel încât software-ul original nefalsificat stă oricând la dispoziție; pe de alta, metoda face casă bună cu copyright-ul producătorilor. Tehnologia CD-ROM este ideală pentru aplicații Multimedia: fotografiile color, desene, sound-uri și, înainte de toate, imagini animate care cer capacități de memorare imense (fig 1)

Chiar în domeniul Multimedia, unitățile CD-ROM sunt livrate foarte des împreună cu plăci de Sound. Pe unitățile externe sunt disponibile carcasa în care pot fi găzduite mai multe unități CD-ROM, astfel încât să poată fi accesate corespunzător cantități mari de date. Schimbătoarele-CD se oferă în special pentru folosirea în rețele. Mai nou se oferă și CD-uri inscriptibile o singură dată, CD-R-urile (R = recordable). Acestea sunt sensibil mai scumpe decât pendant-urile lor doar citibile. Perspectivele CD-R-urilor sunt roze, datorită compatibilității lor cu CD-ROM-urile standard. CD-ROM-urile sunt bune în sensul că producătorul a stabilit din timp normele obligatorii, corespunzătoare procedurii de înregistrare. Acest *High-Sierra-Standard* stabilește în ce ordine trebuie dispuse datele pe CD. Standardul răspândit în Europa (150-9660), a fost dezvoltat din *High-Sierra-Standard*. În cele ce vor urma, aplicații care au fost concepute pentru *High-Sierra-Standard*, sunt rulabile și sub 150-9660. Invers însă nu este posibil.

Ca interfață pentru unități CD-ROM, s-a dezvoltat adaptorul SCSI, ca standard, datorită flexibilității și capacității sale (fig 2)

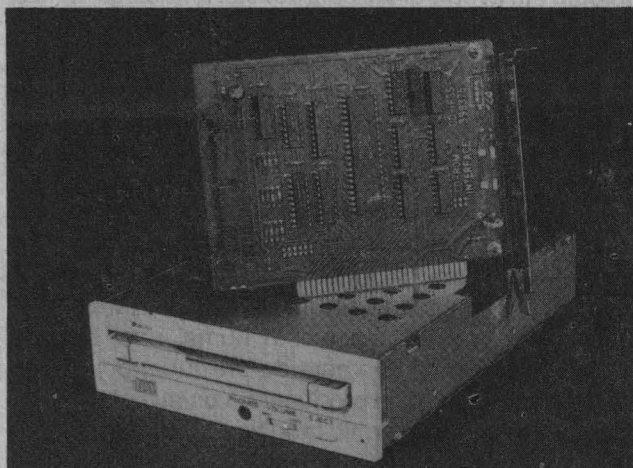


Fig 2 Unitățile care se bazează pe tehnologie de memorare optică, sunt comandate, de cele mai multe ori, printr-o interfață SCSI

SCSI oferă și perspectiva așezării în linie a mai multor aparate SCSI, cum ar fi hardisk-ul, streamer-ul și scanner-ul.

Între unitățile CD-ROM existente, se găsesc și variante avantajoase în ce privește costul, care pot fi cuplate prin interfață serială, respectiv paralelă. Spre deosebire de CD-ROM, mediul de memorare a lui WORM, poate fi reînscris cel puțin o dată. Astfel se apropie mai mult, de mediile de memorare magnetice tradiționale. Discurile de 12", pot memora câțiva GBytes, cele mai mici, de 5,25" atingând oricum o capacitate de circa 650 MBytes. Pe unități *Write-Once*, pot fi înregistrate date, până la 8 GBytes. dar nu numai din cauza mărimii mediilor de memorare, unitățile WORM, sunt livrate numai ca memorii externe. Deoarece plăcile pot fi folosite în sisteme asemănătoare Musicbox-urilor, posibilitățile de extindere sunt nelimitate. Cantitățile de date, puse astfel la dispoziție, pot atinge domeniul Terabytes (1 TeraByte = 1024 GBytes). Acest ordin de mărime depășește de departe, pe cel al suporturilor de date obișnuite. Bucuria unei capacități foarte mari, este știrbită însă, de timpii de acces relativ mari.

WORM - (re) inscriptil o dată

Spre deosebire de CD-ROM, unitățile WORM dispun de două laser-e: un laser puternic, pentruscriere, și unul slab, pentru citire. Pentru a înscris datele, o rază laser explorează suprafața. Reflexia este redirecțată și transformată corespunzător modificărilor luminii, în date binare. Modul de funcționare a discurilor o

dată inscriptibile, bazate pe sticlă, se sprijină pe tehnologia *Pit*. În cursul aplicării acestei tehnologii, o rază laser este dirijată spre stratul de înregistrare, acesta încălzind locul respectiv. Prin această modificare a temperaturii, se provoacă procese de evaporare și reacții termice, care produc în vecinătatea lor "văi" respectiv "culmi". În procesul de citire, o rază laser de intensitate mai redusă este reflectată și apoi prelucrată. Prelucrarea reflexiei se face împrăștiind lumina pe frontiera bulei magnetice (bulele magnetice sunt zone, sau domenii cilindrice, din memoriile magnetice *bubble*, având aceeași formă de magnetizare - n.t.). Deși acest procedeu servește numai unei singure înscriseri, căldura și diferența de presiune, pot șterge informații. Procesul de producere a bulei (fig 3), necesită un strat metalic termic-modificabil, care este înconjurat de un strat de polimer și de unul de sticlă.

Spre deosebire de tehnologia *Pits*, placa are cu totul altă construcție în ce privește crearea bulei. Deasupra unui strat reflectant, se găsește un strat opac. Acest strat este ars de către laser, în locurile corespunzătoare, la citire. Deoarece procesul de scriere, la unități optice se face întotdeauna prin procedee optice, pericolele cunoscute până acum la sistemele magnetice, cum ar fi prăbușirea capului de scriere-citire, pe suprafața discului (așa-numitul *Headcrash*), sunt de domeniul trecutului. Numai praful se poate depune cu timpul. Astfel, la unele aparate, sunt integrate perii pentru lentile, laser-ul fiind permanent curat. Mai ales benzile magnetice și arhivele-microfișă ar trebui din când în când înlocuite cu unități *WORM*, dată fiind capacitatea de memorare enormă a acestora, și timpii de acces substanțial mai reduși. Deoarece da-

tele de pe *WORM* nu pot fi șterse, domeniul principal de aplicații al acestor unități, este arhivarea.

Alături de mediile reinscriptibile, *WORM* își vor păstra dreptul de a exista, mai ales dacă ne gândim la faptul că, pe piața arhivărilor, se înregistrează rate de creștere mari. *WORM* sunt mai puțin sensibile la factorii de mediu decât *CD-R*-urile. Deoarece pot fi înscrise numai o singură dată și numai o dată șterse, ele eșuează ca memorii de masă în adevăratul sens al cuvântului. Discurile magneto-optice sunt deci cea mai bună alternativă.

În tehnologia magneto-optică este vorba de o combinație a tehnicii magnetice cu cea optică. Un procedeu de înregistrare care lucrează după principiul digital trebuie să producă două stări diferite într-un punct al mediului de înregistrare, și să le poată apoi deosebi la citire. Acest lucru este obținut cu raze laser, diferite ca putere, folosind și un electromagnet.

Inscripționarea unui disc magneto-optic se face printr-o rază laser bogată în energie. Această rază încălzește punctele de înscrisere până la temperatura Curie, specifică materialului respectiv. Dacă aceasta este atinsă, este suficient atunci un câmp magnetic exterior slab, pentru a modifica direcția de magnetizare a materialului din care este construit suportul de date. Spre deosebire de celelalte procedee descrise până aici, la procedeu acest, informațiile sunt literalmente șterse. Pe cealaltă parte a plăcii optice se găsește un magnet, care produce un câmp cauzator al unei noi orientări magnetice a zonei de înscrisere (fig 4).

Aici magnetizarea este perpendiculară pe suprafața plăcii. Însă, la puteri mari ale laser-ului, apare pericolul ca zonele de graniță să fie și încălzite și influențate de câmpul magnetic. De aceea raza laser trebuie să fie focalizată precis. Puterea laser-ului folosit, se găsește în domeniul miliwaților, iar temperatura Curie la câteva sute de grade Celsius.

Ca și pe suporturile de date de până acum, pe discul magneto-optic, apar două stări magnetice diferite: pozitiv și negativ. Cele două stări sunt deosebite, prin polarizarea creată de reflexia razei laser, mai săracă în energie. Raza reflectată este polarizată, funcție de poziția magnetilor (se vorbește de așa-numitul efect-Kerr). Conținutul de informații din zonele care nu sunt încălzite de raza laser rămâne neafectat de câmpul magnetic.

Deci, la un disc magneto-optic, de fiecare dată se face mai întâi o ștergere, abia apoi începând procesul propriu-zis de scriere. Acest lucru explică și de ce un proces de scriere pe un disc magneto-optic este mai mare consumator de timp, față de suporturile de date de până acum.

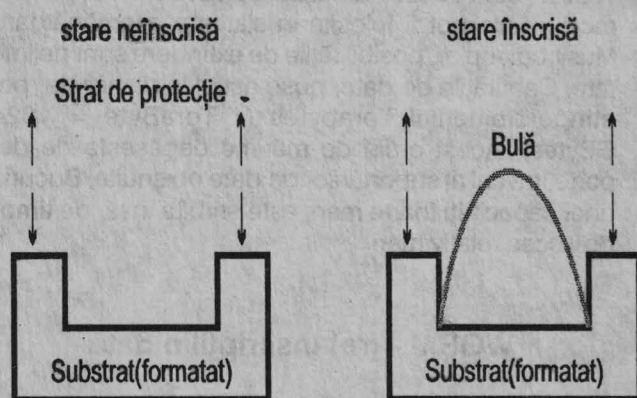


Fig 3 Procedeu de producere a bulei este des folosit la *CD-ROM*-uri. Aici, o rază laser, se ocupă cu producerea unui *Pit*.

Alternativa: memorii magneto-optice

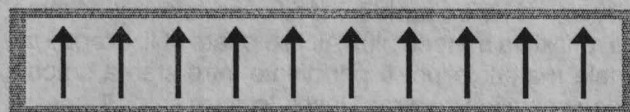
Tehnologia magneto-optică reclamă o tehnică optică complicată pentru capul de scriere-citire. Acest lucru afectează negativ mărimea și masa capului. Un cap mare și greu nu se lasă manipulat rapid, influențând astfel viteza de acces. Acest fapt explică de ce unitățile magneto-optice au performanțe mai reduse, în legătură cu procesele de scriere și citire, decât harddisk-urile.

Unităților magneto-optice li se poate asocia denumirea de "hibrid", între sistemele pur magnetice și cele pur optice. Avantajele discurilor magneto-optice sunt la îndemână: în formatul de 5,25", pot înregistra date între 600 MBytes și 1 GByte. Discurile mai mici, tot de acest tip, cu dimensiunea de 3,5", cuprind și ele 128 MBytes. În plus, discurile magneto-optice sunt insensibile, spre deosebire de harddisk-uri, la câmpurile magnetice; aceasta deoarece structura magnetică a unui anume punct nu poate fi modificată decât dacă locul respectiv este încălzit la temperatura necesară. De aici, garantarea securității datelor pe perioadă mai lungă. Dar și alte influențe exterioare, cum ar fi umiditatea, căldura sau manipularea neglijentă, nu deteriorează acest tip de discuri. O serie de discuri descriabile, sunt concepute pentru o siguranță a datelor între 10 și 100 de ani.

Domeniile de implementare a unităților magneto-optice rezultă din avantajele lor, cantitățile mari de date memorate nefiind o problemă. Cine și-a încercat deja mâna, pe scanner-e color cu profunzimea culorii de 24 de biți, și-a dorit desigur, cu pasiune, și o unitate de memorie corespunzătoare. Aici, mai mulți MBytes de date sunt atinși rapid, și chiar cel mai mare harddisk devine la un moment dat neîncăpător. Din acest motiv, discurile MO se oferă ca purtătoare de date, fără doar și poate, în domeniul mediilor de tipărire. Acest lucru este valabil și pentru CAD și grafică. Dar discul MO este remarcabil și în ce privește folosirea lui ca mediu de Back-up, de arhivare, el fiind, în plus, și ușor de scos din unitate și apoi transportat.

Phase Change Tehnology

În procesul de schimbare a fazei (*Phase Change Technology*), nu este vorba de o nouă tehnologie în sine. Nou este numai domeniul de utilizare. Până acum, acest procedeu cunoscut deja de decenii, a fost folosit - în principal - la unitățile WORM profesionale. Mai nou își găsește utilizare și la plăcile optice reinscriptibile. La această tehnologie este vorba de un procedeu de înregistrare pur optic, care se descurcă fără electromagneți. Datele vor fi scrise și citite prin



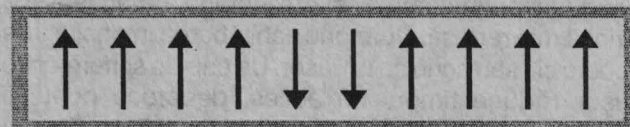
1) Stare de bază



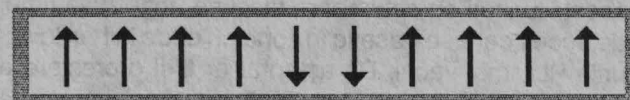
2) Încălzitor laser



3) Orientare (crearea unui câmp magnetic)



4) Răcire (în câmpul creat)



5) Memorare (câmp decuplat)

Fig 4 Un câmp magnetic exterior, se îngrijește de orientarea corespunzătoare, la mediile optice de memorare

raze laser de puteri diferite; la scriere, raza laser este multiplicată de mai mult de zece ori.

Pentru claritate: puterea laser este atunci între 18 și 20 de miliwați. Aceștia sunt suficienți pentru a încălzi stratul corespunzător până la punctul de topire. Locurile prin care "calcă" laser-ul de scriere își modifică reflexia, ceea ce se manifestă la explorarea cu laser-ul de citire. La procedeu cu schimbare a fazei, diferențierea între cele două intensități se face prin fotocelule, spre deosebire de tehnologia magneto-optică unde este determinată de polarizarea luminii. Dacă după reflexie intensitatea a scăzut foarte mult, atunci se recunoaște starea 1. Dacă raza laser este reflectată fără pierderea intensității, aceasta descrie starea 0. La

Fundamente

scriere, raza puternică modifică structura atomică prin încălzire și răcire rapidă, modificând astfel capacitatea de reflexie a materialului. Aici se poate utiliza faptul că unele materiale pot fi pendulate între starea amorfă și cea cristalină cu o rază laser. În stare amorfă, aceste materiale dovedesc un grad de reflexie evident redus, starea aceasta (starea 1) fiind obținută prin încălzire până la punctul de topire. Pentru a readuce particulele în starea cristalină, este suficientă folosirea unei raze laser mai slabe. Prin reducerea căldurii, este redusă libertatea de mișcare a particulelor, ele reasezându-se în structura cristalină (pentru noi starea 0). Diferențele între intensitățile de reflexie sunt aicifundamental mai mari decât diferențele de la polarizare. Spre deosebire de procedeul magneto-optic, cel cu schimbarea fazei lucrează cu o viteză evident mai mare. Acest lucru derivă din faptul că, înaintea scrierii pe placă, informația deja existentă trebuie mai întâi ștearsă.

Supportul optic de date poate fi descris deja după o singură rotație, spre deosebire de procedeul magneto-optic unde sunt necesare permanent, trei rotații. Această (din urmă) unitate optică este mai redusă ca preț decât cea magneto-optică, capul de scriere-citire fiind mai puțin complex și prezentând, de aceea, și o masă mai redusă. Doar prin schimbarea intensității se poate citi sau scrie cu un laser. Un cap de scriere-citire ușor reduce timpul de acces, deoarece poate fi condus evident mai rapid spre locul corespunzător pe placă. În ce privește timpul de acces, este clar că posibilitățile tehnice nu au fost epuizate, chiar dacă la aceste unități cu schimbare fazei se obțin deja timpi de acces, care se găsesc în zona celor de la harddisk-urile RLL mai vechi. De așteptat ar fi și o creștere a turației, iar în domeniul densității de înregistrare se obțin succese importante. Abia atunci când timpii de acces vor deveni comparabili cu cei ai memoriilor magnetice, procedeele nemagnetice vor intra în competiție directă cu acestea.

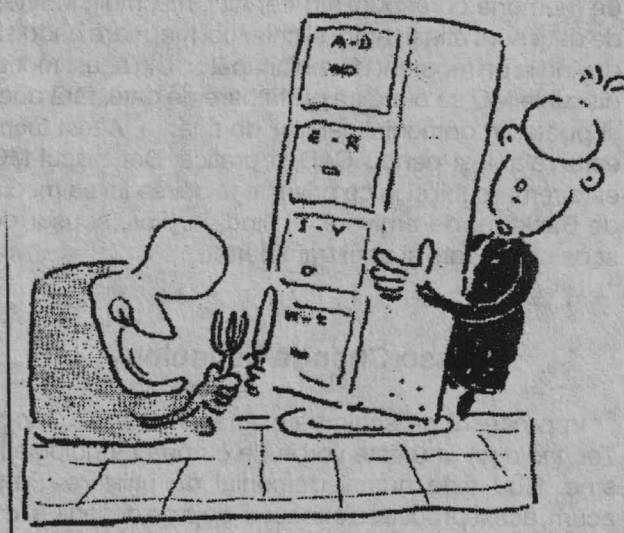
În ce privește capacitatea, la mediile optice, se deosebesc două procedee de înscriere. Pe de-o parte își găsește utilizare procedeul CAV (*Constant Angular Velocity*), aici turația plăcii rămânând constantă, și de aceea densitatea datelor este mai mare în domeniile interioare decât la periferie. Pe de altă parte, este procedeul CLV (*Constant Linear Velocity*), cu o folosire mai eficientă a suprafeței plăcii. Aici densitatea informațiilor este aceeași, atât pe urmele interioare, cât și pe cele periferice. Capacitatea de memorare, în general mai mare, este echilibrată - din păcate - de timpul de acces crescut, deoarece pe parcurs turația trebuie schimbată și se cere și o organizare mai complexă a datelor.

Oricum sensul dezvoltării este, fără îndoială, spre unități optice reinscriptibile. Viitorul aparține unităților combinate, care să poată citi atât WORM-uri cât și

CD-uri, cât și Phase-Change-uri; acestea vor trebui, în plus, să poată și înscrie diferite tipuri de suporturi. Rate de creștere înregistrează și *Jukebox*-urile, care pot gestiona mecanic mai multe discuri. Oricum, în privința *Jukebox*-urilor, în față stau CD-urile. Și în ceea ce privește procedeele de producție, cercetare va înainta, așteptându-se prețuri scăzute și capacități crescute. În nici un caz mediile de memorare optice, nu sunt singurele pentru care cercetarea merge mai departe. Harddisk-urile devin din ce în ce mai rapide și oferă capacități mai mare, la gabarit în scădere. Tot așa, în domeniul arhivării, dezvoltarea unităților de bandă magnetică progresează. Însăși mult-apreciată dischetă nu este exceptată de la progres; ea va asigura o capacitate de câteva ori mai mare decât cea din zilele noastre. Din acest motiv, nu va exista o tehnologie ultimativă. Utilizatorul va trebui să-și cântărească alegerea, funcție de domeniul de implementare. Se profilează o reunire de tehnologii, nicidecum o concurență a lor.

Matthias Günther

Traducere:
ing. Mihai Beer



Reușită în a cincea dimensiune

Cu Pentium, noua creație în sectorul procesoarelor compatibile 80x86, revine și această familie de procesoare - după o perioadă de absență - în rândul microchip-urilor de capacități mari. Câtă putere de calcul îi face cadou PC-ului aceasta nouă "lovitură" de la Intel, prezintă acest articol.

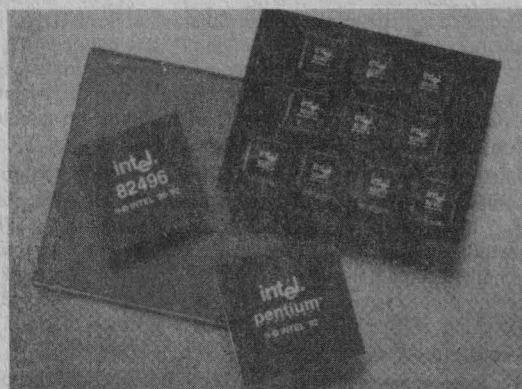
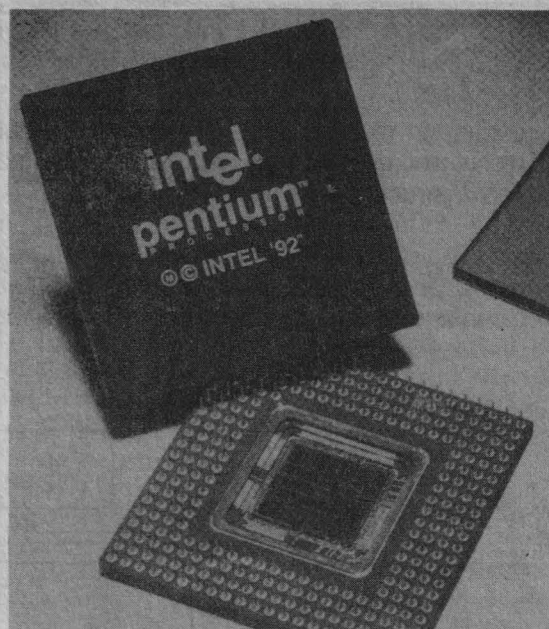


Fig. 1 - Aproape incredibil arată noul procesor de la Intel. Alături noua serie de chipuri pentru Pentium.

În sfârșit a "scăpat" ceva. După luni întregi de atmosferă secretoasă s-a permis unui număr de "aleși", să arunce o privire mai exactă asupra "coroanei" tuturor procesoarelor Intel (figura 1). Ceea ce s-a prezentat jurnaliștilor de specialitate a fost un CPU care asigură PC-ului pătrunderea în domeniul zonelor de putere ale stațiilor de lucru moderne.

Superscalar - din doi facem unul

De mai multă vreme este cunoscut faptul că noul zar aruncat de Intel - cum era și de așteptat - nu va purta numele de 586. Această denumire s-ar fi potrivit în ordinea 8086, 80286, 80386 și 486; și totuși, după certuri cu concurența asupra drepturilor, a devenit clar că Intel nu va putea să-și protejeze numele sub forma unei combinații de cifre. Altfel ar fi putut fiecare firmă să copieze produsul denumindu-l "586". De aceea cei de la Intel s-au hotărât rapid să părăsească drumul



obișnuit de alegere a numelui și au botezat noul procesor cu numele, protejat prin drept de autor, de Pentium.

Dar nu numai ca nume se deosebește Pentium de premergătorii săi. Și din punct de vedere tehnic el explorează teren nou (figura 2). Este vorba aici de procesorul cu arhitectură "superscalară". În loc de o unitate de integrare, Pentium are două Integer-Units prin care să prelucreze, la o trecere, două comenzi (în cazul cel mai favorabil). Pentru aceasta se vor încărca din code-cache două comenzi succesive de fiecare dată în cele două Pipelines ale unităților de integrare, urmând prelucrarea în paralel.

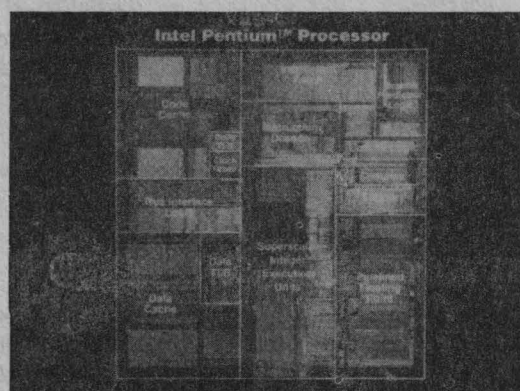


Fig 2 Pe 300 milimetri pătrați de siliciu sunt înghesuite cele 3,1 milioane de tranzistoare

Dar acest lucru nu este suficient. La o instrucțiune de salt condiționat sau necondiționat, Pentium încearcă să aibă pregătită instrucțiunea următoare

corectă (pentru salt, respectiv continuare fără salt). De acest lucru se îngrijește o unitate numită BPL (*Branch Prediction Logic*).

Ea conține în memorie ramificări ale programului, pentru a putea prevedea deja instrucțiunea corectă la repetarea instrucțiunii de salt. Dacă predicția este corectă, a doua pipeline poate să își dea drumul neîntârziat. Acest truc rafinat îi aduce lui *Pentium* cel mai mare câștig de capacitate față de 486. Figura explică construcția internă a procesorului *Pentium*.

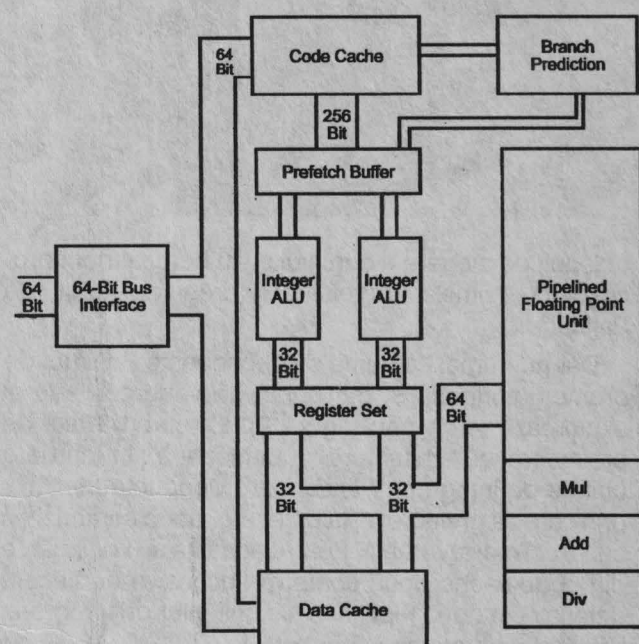


Fig 3 Desenul arată, în mare, construcția internă a procesorului. Semnificative sunt cele două unități de integrare caracteristice arhitecturii superscalare.

BPL prezice

Cine lucrează atât de repede are nevoie și de o încărcare rapidă a comenzilor următoare. Alimentarea pipeline-urilor este preluată de o unitate *Prefetch* de 32 de octeți, împărțită în două, care la rândul ei "crează" dintr-un cache de comenzi, de 8 KByte. Pentru ca ambele Pipeline să primească simultan comenzi, cache-ul de comenzi permite accesul simultan la conținutul său.

Total independent de cache-ul de comenzi, lucrează cache-ul de date al lui *Pentium*. Prin aceasta

Concurența RISC

La începutul anului 1991 producătorii de vârf din domeniul hardware și software, s-au așezat la o masă pentru a stabili un numitor comun în ce privește posibilitatea folosirii, pe căi simple, a softurilor existente și pe alte platforme hardware. Acestei inițiative numită ACE (*Advanced Computing Environment*) i-au aparținut, între alții, Microsoft, Intel, MIPS, Sun și DEC. Chiar dacă această reuniune a eșuat datorită unor frecșuri interne, a avut cel puțin un efect: generația sistemelor de operare viitoare, Windows NT de la Microsoft, va putea să ruleze nu numai pe procesoare Intel. În paralel cu versiunea pentru PC, Microsoft dezvoltă și o versiune corespunzătoare ACE. Aceasta este funcțională nu doar pe procesoare MIPS - R3000, R4000 și R4400, ci rulează și pe procesoarele Alpha ale firmei DEC precum și pe computerele SPARC de la Sun. Deci software-ul NT trebuie doar compilat pe tipul de procesor specific, după care va lucra la fel pe un Intel 486 ca pe un Alpha. Dacă se compară datele venite de la concurenții lui Intel cu cele ale lui *Pentium* - noul născut - situația nici nu este chiar atât de roză.

Tochmai a fost etalat procesorul Sparc al lui Sun. El este cel mai ușor comparabil cu *Pentium* în ce privește capacitatea. La procesorul Sparc este vorba despre un CPU eminent RISC, care este înlma care bate în workstation-urile Sun. Capacitatea la calculul cu întregi este undeva sub cea a lui *Pentium*; calculele în virgulă mobilă se fac, dimpotrivă, mai rapid.

Mult mai greu îi este lui *Pentium* să rivalizeze cu superprocesoarele de la MIPS și DEC. Ambele lucrează cu o frecvență de tact de 150 MHz. Aceasta are ca rezultat teoretic, datorat arhitecturii superscalare, o capacitate de 300 mips. Dar și la Specmark - Benchmark-ur' mai realiste - este bătut *Pentium* de cele două procesoare de 64 de biți. Atât Alpha de la DEC cât și R4400 ajung, cu peste 100 Spearmarks, la aproape de două ori valoarea pentru *Pentium*.

Cu toate acestea sensul comparării directe a două procesoare care aparțin unor lumi diferite, stă oarecum sub semnul întrebării. Aceasta deoarece factori cum ar fi rata de transfer a datelor pe magistrală, organizarea periferiei și suport prin sistemul de operare, au în practică o foarte mare influență asupra vitezei reale a computerului. Măsura o dau aici sigur Workstation-urile de la Silicon Graphics care lucrează cu R4400, dar care se și mișcă în cu totul alte domenii de preț. În această privință Intel nu este în pericol de a fi presat de piață; marea masă a utilizatorilor se interesează încă de computere cu capacitate satisfăcătoare și cu preț acceptabil. Dacă Windows NT va putea schimba ceva în acest sens vom mai vedea.

s-a înlăturat un punct foarte slab al lui 486. Acum un capăt al unei pipeline poate citi din cache-ul de comenzi, pe când celălalt capăt scrie în cache-ul de date. În acest caz, pentru 486 ar fi fost necesar un Waitstate (o stare de așteptare).

Și la cod recursiv (care se schimbă prin el însuși) consistența datelor se păstrează. De aceasta se îngrijește o unitate proprie, care verifică permanent dacă conținutul unui cache mai este actual sau dacă se găsește deja în forma actualizată în celălalt cache.

În ce privește frecvența de tact, nu există noi recorduri. *Pentium* va apărea la început într-o variantă de

60 MHz și într-una de 66 MHz. Astfel, în această privință nu este depășit purtătorul steagului care este 486DX/2-66. Faptul că *Pentium* lucrează, cu toate acestea, de două ori mai repede se datorează în primul rând arhitecturii superscalare.

Viteză dublă în ciuda păstrării frecvenței

Aproape toate instrucțiunile lui *Pentium* sunt pe durata unui singur ciclu de ceas. Pentru comenzile complexe ale procesoarelor 80x86, el dispune de o unitate specială, care păstrează compatibilitatea cu vechile procesoare Intel. Deci la întrebarea dacă *Pentium* are un CPU RISC sau CISC se poate răspunde foarte clar: "DaNu".

Deși *Pentium* este un procesor de 32 de biți, el poate accesa memoria de lucru pe 64 de biți. Astfel, la o singură trecere pot fi transferați 8 octeți din memorie în cache. Prin aceasta atinge o rată de transfer al datelor de maximum 528 MByte pe secundă. În comparație cu acesta, 486 lucrează cu 160 MByte pe secundă. La bus-ul de intrare/ieșire nu se schimbă nimic. El este limitat în continuare la 32 de biți.

Pentium - minunea calculului

După părerea celor de la Intel, tehnica procesoarelor pe 32 de biți este suficientă în viitorul apropiat. Mai mult decât cei 4 Gbyte de memorie adresabilă fizic, nu va putea nimeni să înglobeze în computerul său, în anii care vor urma. Transferul I/O este și el destul de puternic prin bus-ul local. Absoluta necesitate a unui procesor de 64 de biți nu se întrevide momentan la Intel.

Cea mai mare creștere a capacității a obținut coprocesorul integrat. El lucrează acum ca pipeline cu 8 nivele cu o unitate de adăuie, multiplicare și diviziune implementată hardware în mod corespunzător. Un bus de 64 de biți asigură un transfer de date rapid. Spre deosebire de 486 rezultă un câștig în performanță de cinci până la zece ori mai mare. Chiar și în cazul mediilor de aplicații cu calcule intensive cum sunt CAD, grafică 3D sau calcul tabelar acest lucru este remarcabil. Un computer-prototip pe care Intel l-a prezentat la prima demonstrație, a arătat într-un mod impresionant creșterea de capacitate a noului CPU. Un demo care "vrăjea" o grafică 3D în timp real, pe ecran, a mers cu o viteză obișnuită doar pe workstations. Și Windows a profitat foarte mult de pe urma acestui procesor de mare capacitate.

În calculatorul pregătit pentru demonstrație lucra, acoperit de un radiator gigantic, un *Pentium* pe 66 MHz, pe o placă-procesor separată (figura 4). Avea la dispoziție 32 MBytes de memorie internă. Comunica-

rea între CPU și cartela grafică ATI-Match-32 se făcea prin bus-ul local PCI, specificat de Intel.

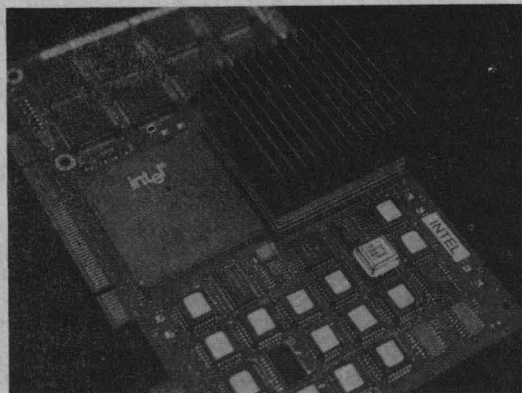


Fig 4 Figura arată placa-procesor a calculatorului pentru demonstrație. *Pentium* este ascuns sub radiatorul uriaș.

Pentru a exploata la maxim potențialul lui *Pentium*, trebuie ca și software-ul să fie adaptat arhitecturii superscalare. De aceea Intel a dat mare importanță dezvoltării de compilatoare potrivite pentru noul procesor. Software-ul compilat pe acestea rulează vizibil mai repede decât soft-ul MS-DOS de până acum. Însă și soft-ul vechi, de pe procesoarele 80x86, codat cu acestea, nu-și pierde din capacitate.

O atenție deosebită a acordat Intel temei *Multiprocessing*. Primul pas, fără creștere de capacitate, a fost funcționarea a două procesoare *Pentium* într-un mod de observare. Aici unul din procesoare preia calculul propriu-zis iar celălalt verifică corectitudinea lucrului. Dacă apar neconcordanțe este trimis un mesaj corespunzător. Această funcție de siguranță este gândită, înainte de toate, pentru file-server-e.

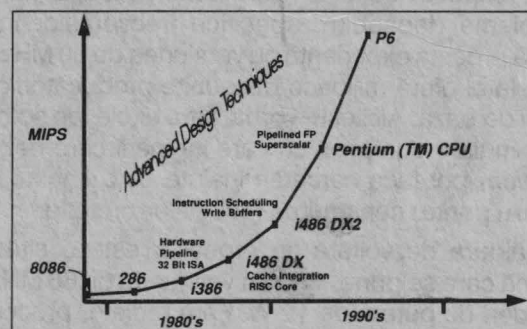


Fig. 5 - Capacitatea lui *Pentium* comparată cu cea a predecesorilor săi. De pe acum să ne bucurăm de următorul (P6).

Ce se va întâmpla cu 486 ?

Chiar dacă există *Pentium*, Intel nu vrea - pe nici o cale - să încheie dezvoltarea lui 486, deoarece chiar în acest domeniu lucrează neobosit concurența pentru a rupe monopolul originalului. Dacă se iau ca bază cifrele actuale de vânzare, Intel nu poate renunța în nici un caz la această piață. În ultimul an au fost mai mulți cumpărători de 486 decât de 386. Nici *Pentium* nu va modifica mult din această realitate, deoarece el va fi rezervat sistemelor scumpe. Cei de la Intel se gândesc acum la căi de creștere a capacității lui 486. Ceea ce este bun pentru *Pentium* este pentru 486 doar puțin - un Cache intern mărit. Astfel și 486 va obține un Cache de memorie de 16 KByte, pentru îmbunătățirea performanțelor sale.

O cale de dezvoltare în continuare vede Intel în tehnica *Clock-Tripler*. În locul vitezei duble, noi versiuni ale lui 486 vor lucra cu o viteză triplă a magistralei. Deci să ne bucurăm gândul unor CPU-uri cu tactul de 75 și 100 MHz.

Concurenței îi este greu în momentul de față să apară pe piață cu un 486 cinstit. Oricum AMD are un 486 funcțional pe masa de lucru, dar procesul pierdut de curând în favoarea lui

Intel nu le permite utilizarea microcodului lui 486. Până când AMD apare cu un cod propriu vor trece desigur câteva luni.

Cyrix oferă de mai mult timp un procesor marcat Cx486DLC. Aici este vorba de un 386SX cu un Cache intern de numai 1 KByte. Îi lipsește de asemenea coprocesorul matematic.

Mai aproape de model se află noua creație 486S2-50 al lui Cyrix. Și acesta, ca și 486DX/2, se servește de o tehnică *Clock-Doubler*. Aceasta înseamnă că intern lucrează cu 50 MHz, iar pe extern are 25 MHz. Cache-ul intern de 2 KB a părut cam slăbuț în comparație cu cel de 8 KB al originalului. Totuși, algoritmi de Cache rafinați pot suplini această lipsă. Ca și lui 486SX și lui Cyrix 486S2-50 îi lipsește coprocesorul matematic. Dar încă înaintea sfârșitului lui 1993 va urma un prețios 486 DX.

Și IBM dezvoltă o versiune proprie de 486 DX. Cu o frecvență de tact de 100 MHz și 16 KB Cache intern va intra în competiție "super-486".

Astfel lui Intel îi rămâne puțin timp pentru a se culca pe lauri, deoarece deândată ce Cyrix, AMD și ceilalți vor avea un produs finit, vor atrage prin preț - așa cum au făcut deja la 386 - pentru a-și tăia o felie mai mare din cozonacul procesoarelor.

Și pentru modul multiprocesor real are *Pentium* pregătite câteva "features". Ele afectează, în primul rând, chestiuni ale consistenței datelor. Aici sunt implementate hardware câteva funcțiuni care scutesc mult din efortul programatorilor de multiprocessing-software. Întrebarea dificilă dacă datele din cache mai sunt actuale sau nu, sau dacă un alt procesor tocmai prelucrează un nou rezultat, capătă un răspuns - în multe cazuri - prin hardware. Aceasta nu numai că economisește efort de programare, dar crește și performanța.

Grijii și-au făcut cei de la Intel, în legătură cu frecvența de tact. Oricum pe bus-ul de procesor există 60 MHz respectiv 66 MHz - care determină apariția unor probleme deosebite, specifice frecvențelor înalte. După proasta experiență cu versiunea de 50 MHz a lui 486, Intel oferă mijloace deosebite producătorilor de plăci de bază. Aici este vorba, între altele, de software de simulare complex, cu care inginerii care dezvoltă *Pentium* pot face cercetări înainte de a cheltui banii pentru construirea modelelor-prototip.

Căldura dezvoltată de procesor este o altă problemă care se pune. Oricum versiunea de 66 MHz are pierderi de putere de 12 W. Fără radiator procesorul nu poate disipa cantitatea de căldură în mediu. Intel îl lasă pe producătorul de computere să-și construiască propria variantă de disipare. Procesorul este livrat fără radiator.

Cât costă *Pentium*, Intel nu vrea să desconspire. În schimb dă asigurări că prețul per Mips va fi chiar mai mic decât la 486. Aprecierile precaute se învârt în jurul

cifrei de 2000 DM pe bucată. Computerele corespunzătoare scriu prețul cu cinci cifre.

Rezumat

Cu noile procesoare Intel a dat un zar bun. Din nou își întrece vizibil predecesorul: noul procesor are capacitate dublă față de un 486 DX/2-66 (figura 5).

Pentru aplicații grafice și domeniul CAD probabil că acest salt de capacitate a fost dorit de mult, însă cea mai mare a utilizatorilor nu folosesc această performanță. Pentru ei este mai importantă o placă grafică mai rapidă sau un hard-disc, decât o creștere a capacității CPU-ului. De această realitate sunt convingși și cei de la Intel. De aceea ei pun accentul în continuare pe eforturile legate de 486. Doar câteva sute de mii de procesoare *Pentium* vor părăsi în acest an halele de producție - o cifră aproape jenantă în comparație cu cele 30 de milioane de 486 produse în același timp. Astfel în anii 1993 și 1994 486 rămâne numărul 1.

Dar acestea se pot schimba drastic sub noul sistem de operare Windows NT. Primele impresii ale noii versiuni Windows sunt foamea de capacitate. Chiar și cu hardware rapid el se chinuie. Utilizatorii frenetici de NT vor forma piața mare de *Pentium*. Ei vor fi urmați, pe baza experienței de mai târziu, de cei care vor dori software eficient nu unul care să facă doar abuz de resurse.

Traducere:
ing. Mihai Beer

Lanțuri aruncate în aer

Programarea lui *Extended Memory*

Acum 13 ani, când PC-urile apăreau pe piață, cei 640 de KBytes ai memoriei de lucru cu care erau echipate păreau aproape gigantici. Pe măsura dezvoltării în continuare a software-ului, a reieșit faptul că, pentru multe aplicații, 640 de KBytes sunt pur și simplu prea puțin. Acest articol arată, cum puteți "arunca în aer lanțurile" limitării memoriei și cum puteți folosi așa-numita *Extended Memory*, pe AT sau 386.

Din păcate sistemul de operare MS-DOS, dată fiind conceperea sa pentru procesorul 8086/8088 folosit în PC-uri, nu poate adresa decât 1 MByte de memorie. Deoarece din acesta, 384 KBytes sunt rezervați pentru memoria-ecran a cartelelor grafice, BIOS și extensiile BIOS, MS-DOS are la dispoziție numai 640 KBytes.

Pentru a sparge granița de 640 KBytes, specifică MS-DOS-ului, există două procedee. Pe de-o parte așa-numita *Expanded Memory*, după standardul LIM-EMS, elaborat împreună de Lotus, Intel și Microsoft, iar pe de alta, *Extended Memory*, care rezultă inevitabil din construcția unui PC-AT sau 386. *Expanded Memory* extinde memoria aflată la dispoziția MS-DOS-ului, utilizând porțiunile nefolosite ale domeniului de memorie rezervată (cei 384 KBytes), despre care vorbeam mai sus. Procedeele EMS (*Expanded Memory System*) intercalează în poziții neocupate din interiorul memoriei, așa-numite "pagini de memorie", având mărimea de câte 16 KBytes. Aici se cer, oricum, și plăci EMS speciale, care lucrează cu o logică de comandă deosebită, pentru a putea intercala și apoi scoate paginile de memorie în și din spațiul de memorie dorit. Administrarea paginilor de memorie ale lui *Expanded Memory* este preluată de un driver EMM (*Expanded Memory Manager*), care pune la dispoziție mai multe funcții prin intermediul întreruperii 67 hex. *Expanded Memory* poate fi însă și simulată. Pentru acesta există emulatoare EMS speciale, care preschimbă *Expanded Memory*, în EMS software compatibil pe harddisk sau în *Extended Memory*. Deoarece procesorul 80286 a

unui AT, în contrast cu 8086/8088, poate adresa 16 MBytes de memorie, iar procesorul 80386 poate adresa 4 GBytes, pe aceste clase de computere poate fi instalată memorie suplimentară. Aceasta trece dincolo de granița de 1 MByte și se numește *Extended Memory*. De fapt *Extended Memory* este prevăzută pentru sisteme de operare care nu sunt limitate la cei 640 KBytes ai lui MS-DOS, cum sunt de exemplu OS/2, XENIX sau UNIX, care fiecare pot gestiona 16 MBytes de memorie.

Oricum, cu limitările de rigoare, se poate accesa și sub MS-DOS, domeniul de memorie de deasupra limitei de 1 MByte. Limitări, deoarece accesul controlat, cu simț de răspundere, la *Extended Memory*, presupune o programare destul de stufoasă. Motivul este, din nou (cum ar putea fi altfel !?) concepția lui MS-DOS.

În mod fundamental, MS-DOS poate fi folosit numai într-un singur mod, care poate adresa maximum 1 MByte de memorie. La procesorul 8086/8088, aceasta mai este încă de înțeles, deoarece el nu poate adresa mai multă memorie cu circuitele sale. Dar 80286 și 80386, pot de fapt mai mult. Aici MS-DOS poate fi manevrat numai în *Real Mode*, care emulează cu mare precizie numai proprietățile lui 8086/8088. În principiu, un 80286 sau un 80386 în *Real Mode* nu sunt altceva decât un 8086/8088 deosebit de rapid, astfel încât nici aici nu se poate face o adresare peste limita de 1 MByte.

Altfel stau lucrurile în așa-numitul *Protected Mode*, care a fost adăugat la 80286 și 80386. Acest mod de funcționare este de fapt noutatea, în comparație cu procesorul 8086/8088. *Protected Mode* permite izolarea logică a domeniilor de adrese pentru mai multe programe. Printr-un schimb rapid între aceste programe, este posibilă rularea simultană (mai bine zis aparent simultană), a mai multor programe.

În *Protected Mode*, pot fi folosite 24 (la 80286) respectiv 32 (la 80386) de căi de adresare, deci câteva căi în plus față de *Real Mode*. Astfel pot fi adresați pînă la 16 MBytes, respectiv 4 GBytes de memorie de lucru. Deci în *Protected Mode*, accesul la *Extended Memory*

Memoria extinsă

nu pune nici un fel de probleme; aici pot fi adresați, în plus față de domeniul convențional de memorie, mai mulți MBytes. Cum pot fi folosite aceste capacități, când MS-DOS funcționează totuși numai în *Real Mode*? Acest lucru este făcut posibil, prin faptul că sub MS-DOS, puteți comuta și în *Protected Mode*, poziționând un *statusbit* special (flag), în cuvântul de stare al mașinii. Prin aceasta puteți accesa memoria extinsă după cum doriți, și apoi să reveniți în *Real Mode*. Acest procedeu funcționează într-adevăr, folosirea sa fiind oricând, mult mai greoaie decât pare la prima vedere. Cea mai simplă sarcină este comutarea în modul protejat. Pentru aceasta, 80286 și 80386 au la dispoziție comanda în cod-mașină "lmsw", cu care comutarea se face mai lejer. Greutățile apar atunci când această comandă nu este cea din urmă de executat. Dacă încercați să comutați în *Protected Mode* numai la comanda "lmsw", de regulă computerul va "cădea". Aceasta se datorește faptului că mulți regiștri și multe structuri de date, care nu aveau nici o importanță în *Real Mode*, pot fi de mare importanță pentru execuția programului în *Protected Mode*.

Dacă ați trecut și peste acest hop, vă aflați în fața problemei accesului propriu-zis la memoria extinsă. Acest lucru nu mai este dificil când dispuneți de suficiente cunoștințe de programare. ale lui 80286 și 80386.

O problemă reală este însă ultimul pas: revenirea în *Real Mode* (pentru a duce mai departe MS-DOS-ul) pentru că, spre deosebire de comutația din *Real* în *Protected Mode*, pentru 80286 nu există comandă care să determine revenirea din *Protected* în *Real*. De fapt cei care au dezvoltat procesorul 80286 au pornit de la ideea că nimeni nu se va gândi la așa ceva, cu toate că modul protejat oferă mult mai multe posibilități decât cel real. Abia la dezvoltarea lui 80386 s-a revenit asupra acestei omisiuni, prin faptul că a fost implementată o nouă comandă pentru comutare din modul protejat în cel real. Totuși singura metodă care permite o întoarcere din *Protected* în *Real* la 80286, este oarecum "brutală": procesorul trebuie supus unui Reset, deoarece după un Reset, procesorul pornește întotdeauna în *Real Mode*. Dar după un Reset, computerul începe de fiecare dată cu autotestul, care în acest caz este de nedorit. Mai degrabă, ar trebui ca programul să se întoarcă nemijlocit în locul care urmează comenzii Reset. Așa cum puteți constata din această scurtă prezentare, există o serie întreagă de greutăți când se încearcă trecerea în mod protejat și apoi în real. Din fericire nu trebuie să vă îngrijiiți dumneavoastră de toate amănunțele în ce privește programarea, deoarece BIOS-ul unui AT sau unui PC-386 conține câteva funcții, pentru accesul la memoria extinsă.

Cu aceste funcții se poate determina mărimea memoriei extinse, poate fi comutat procesorul în modul

protejat și celulele de memorie din *Extended Memory* pot fi înscrise sau citite. Aceste funcții sunt puse la dispoziție drept subfuncții ale întreruperii BIOS 15hex. Figura 1 reunește funcțiile și parametrii corespunzători.

Funcția 87hex: transmiterea cuvintelor între memoria de lucru și <i>Extended Memory</i>	
Intrare	AH = 87hex CX = numărul de cuvinte de transmis ES = adresa segmentului unei GDT (Global Descriptor Table) SI = adresa offset a unei GDT
Ieșire	bit de stare-carry (0: nici o eroare, 1: eroare) AX=cod de eroare (0: nici o eroare)
Funcția 88hex: calculul mărimii memoriei extinse	
Intrare	AX = 88 hex
Ieșire	AX= mărimea în KBytes a memoriei extinse
Funcția 89hex: comutare în mod protejat	
Intrare	AH = 89 hex
Ieșire	fără

Figura 1:
Rutinele întreruperii 15hex care lucrează în modul protejat

Prin funcțiile sale, întreruperea 15hex poate face, în principiu orice lucru care are legătură cu modul protejat. De un interes deosebit sunt primele două funcții ale acestei întreruperi. În acest context, funcția 89 hex este neinteresantă deoarece, așa cum am menționat, de îndată ce este apelată această funcție, computerul se prăbușește, dacă anterior nu au fost utilizați anumiți regiștri. Funcția 88 hex este foarte ușor de manipulat. Cu aceasta puteți determina mărimea memoriei extinse. După apelul acestei funcții se stabilește mărimea, în KBytes, a memoriei extinse și este întoarsă în registrul AX. Cele două programe-exemplu scurte, "exramchk.pas" (listing 1) și "exramchk.c" (listing 2), demonstrează utilizarea funcției 88 hex în Turbo-Pascal și Turbo-C. Ambele programe constată câtă memorie extinsă este instalată și întorc un mesaj corespunzător. Ceva mai complicat este lucrul cu funcția 87 hex. Cu această funcție pot fi transferate blocuri de memorie, atât din memoria de lucru convențională în *Extended Memory*, cât și invers. Așa cum arată figura 1, funcția 87 hex, așteaptă adresa de segment și de offset, a unei GDT. Pentru a explica semnificația acestei GDT, trebuie să revenim puțin. Procesoarele 80286 și 80386 au fost dezvoltate cu scopul de a produce condițiile inițiale hardware pentru un mod de lucru multitasking efectiv. Pe lângă mecanismele de protecție a domeniului de adrese ale unui program față de un altul, aparțin aici și procese de

salvare pe disc, la nevoie, de programe întregi, părți de programe sau domenii de date, când memoria de lucru este folosită în alte scopuri. Dacă atunci sunt necesare din nou acele părți de program sau acele date, ele sunt aduse înapoi în memorie de pe harddisk. Acest procedeu presupune existența dinainte a unei adresări a memoriei, cu totul diferite de cea din *Real Mode*. De exemplu, pentru a citi un Byte în *Real Mode*, folosiți o adresă de segment și una de offset, care împreună formează adresa fizică a celulei de memorie referite.

În *Protected Mode*, un asemenea acces direct nu este posibil deoarece, dacă aici un program ar putea utiliza adrese fizice, ar fi posibil pe de-o parte să se acceseze și spațiul altui program, iar pe de alta, procesorul nu ar putea să aibe grijă ca programele salvate în afară, să fie citite din nou în memorie înaintea unei încercări de acces. De aceea, în modul protejat nu pot fi folosite adrese fizice, ci adrese logice. Transformarea adreselor logice în adrese fizice se face prin intermediul unei unități de gestiune a memoriei (*Memory Management Unit*, pe scurt MMU), integrate în procesor.

Deci conținutul unui registru segment nu va fi înțeles în *Protected Mode* direct ca adresă, ci ca "sector". Acesta este un indicator spre o tabelă în care se găsesc, din nou, "descriptori". În sfârșit, un descriptor descrie parametrii fizici ai unui segment de memorie, deci a adresei fizice, a lungimii și priorităților de acces ale acestuia. Tabela în care se găsesc descriptorii este denumită Tabelă de Descriptori. Un program care rulează în modul protejat, folosește selectori pentru a accesa domenii de memorie. MMU-ul calculează descriptorii cu ajutorul acestor selectori, astfel încât procesorul să poată accesa fizic celulele de memorie. Dacă un segment de memorie a fost transferat pe un suport de date extern, este la dispoziție un marcaj corespunzător, în descriptor. La un acces la acest segment, MMU recunoaște că domeniul de memorie nu este încă la dispoziție în memoria de lucru, și încarcă segmentul corespunzător de pe suportul de memorie. Prin selectoare și descriptorii procesorul, respectiv MMU-ul său, poate deci supraveghea în mod simplu toate accesele la segmentele de memorie.

La folosirea întreruperii 15hex, pentru schimbul de date între memoria de lucru convențională și cea extinsă, procesorul este comutat în modul protejat. Însă mai întâi trebuie încărcată o tabelă de descriptori, pentru a putea accesa ambele domenii de memorie (deci atât memoria de lucru cât și memoria extinsă). De aceea întreruperea așteaptă adresa unei tabele globale de descriptori, în poziția din memorie definită prin combinația de regiștri ES:SI. Tabela globală de descriptori este valabilă pe întregul sistem de operare, nu numai pentru un program singular (în acest ultim

caz, ar fi o tabelă locală de descriptori). Figura 2 redă construcția unui GDT.

00 hex	rezervat	8 Byte
08 hex	Tabela Globală de Descriptori	8 Byte
10 hex	adresă-sursă	8 Byte
18 hex	adresă-destinație	8 Byte
20 hex	segment de cod BIOS	8 Byte
28 hex	segment de stivă BIOS	8 Byte

Figura 2: Construcția unei GDT (Global Descriptor Table)

Pentru aplicația următoare, sunt importante numai câmpurile "adresă-sursă" și "adresă-destinație". Toate celelate câmpuri sunt completate independent, la apelul lor, efectuat de către BIOS. Ambele câmpuri conțin descriptori. Prin aceștia trebuie dată adresa fizică a segmentelor de memorie care să servească drept sursă, respectiv drept destinație, a transferului. Fiecare descriptor are lungimea totală de 8 Bytes. Figura 3 redă construcția unui descriptor.

În câmpul *Adresa fizică a segmentelor de memorie*,

00 hex	Lungimea segmentelor	2 Byte
02 hex	Adresa fizică a segmentelor de memorie	3 Byte
05 hex	Nivelele de prioritate ale accesului	1 Byte
06 hex	Rezervat	2 Byte

Figura 3: Construcția unui descriptor, în interiorul unei GDT

trebuie să indicați adresa fizică a segmentului de memorie de transferat. Pentru această adresă sunt prevăzuți 24 de biți, putând fi deci reprezentată prin 3 Bytes. Câmpul *lungimea segmentelor* indică lungimea domeniului de memorie de transferat. Dar, deoarece în apelul BIOS numărul de cuvinte de transferat trebuie oricum dat, BIOS-ul AT-urilor și PC-urilor 386 nu poate face deosebirea între diferitele programe care fac acces la *Extended Memory*. În măsura în care trans-

Memoria extinsă

miteți corect parametrii, BIOS-ul execută transferul de date cerut. Ce se întâmplă însă dacă mai multe programe folosesc simultan memoria extinsă?

Vi se poate întâmpla, de exemplu, ca un program-ajutător rezident, împreună cu un alt program-utilizator, să depună concomitent date, în același domeniu din memoria extinsă, pentru că "nu știu unul de altul". Începând cu versiunea 3.3 a lui MS-DOS, lucrurile merg și mai rău: aici se poate găzdui chiar RAM-diskul în *Extended Memory*, iar începând cu versiunea 4.0, pot fi găsite aici fișiere-tampon sau zone-tampon. Dar cine garantează că aceste programe nu folosesc exact domeniul din memoria extinsă pe care l-ați ales pentru memorarea datelor dumneavoastră? Primul adaos la soluție constă în împiedicarea unui program de a utiliza memoria ocupată de o altă aplicație. Pentru a ajunge la o concluzie, trebuie să vă întrebați de unde va ști un program care parte de memorie o poate folosi și care nu. Pentru aceasta, programul cere mărimea memoriei extinse prin funcția 88 hex a întreruperii 15 hex. Mărimea întoarsă de această funcție, va fi apoi folosită integral sau parțial.

Deci un program nu folosește memorie neautorizată de funcția 88 hex, deoarece pornește de la ideea că, peste valoarea determinată, nu mai există spațiu disponibil în memoria extinsă. Atunci, pentru a proteja un domeniu de memorie de accesul unui al doilea program, trebuie să vă îngrijiți ca BIOS-ul să nu întoarcă mărimea reală a memoriei extinse, ci numai domeniul din memorie care mai este încă disponibil. Pentru aceasta, trebuie să redirecționați vectorul întreruperii 15 hex (semnificând adresa rutinei de tratare a întreruperii aparținătoare), spre o rutină proprie.

Fie cazul unui program pe care l-ați scris dumneavoastră, care este foarte "avid" de spațiu de memorie, necesitând 640 Kbytes de memorie extinsă. Acum, dacă AT-ul sau PC-ul 386 dumneavoastră dispune în total de 1 MByte de memorie extinsă, funcția 88 hex întoarce, în mod corespunzător, valoarea 1024. Deci, programul dumneavoastră trebuie să redirecționeze întreruperea 15 hex spre o rutină proprie, care să intercepteze funcția 88 hex care să întoarcă o mărime a memoriei, redusă cu 640 Kbytes; în acest caz, rămân 384 Kbytes de memorie extinsă. Un program următor, va percepe o memorie extinsă de 384 Kbytes când apelează funcția 88 hex, el nici macăr neîncercând să acceseze domeniul de memorie de deasupra (în acest caz, cei 640 Kbytes rezervați de dumneavoastră), acesta fiind deci protejat la acces. Avantajul acestui procedeu este că poate funcționa de mai multe succesiv deoarece al doilea program care utilizează memoria extinsă poate, la rândul lui, să redirecționeze vectorul întreruperii pentru a "amăgi" un program care urmează, cu o mărime a memoriei extinse și mai redusă.

O problemă apare totuși, în cadrul acestei metode, atunci când trebuie încheiat un program. Atunci domeniul de memorie rezervat de program în *Extended Memory*, ar trebui re-eliberat. Teoretic acest lucru poate fi obținut redirectând vectorul întreruperii 15 hex, din nou pe rutina inițială din BIOS.

Ce se întâmplă însă dacă, între timp, s-au băgat în vector alte programe? Rutinele-service de întrerupere sunt și ele "scoase", iar problema inițială reapare: un program proaspăt încărcat, pornește iar de la premiza că are la dispoziție întreaga memorie extinsă. În afară de aceasta, o a doua problemă nu a fost încă rezolvată: ce se întâmplă oare, cu programele-ajutătoare interne ale lui MS-DOS, când folosesc *Extended Memory*? Un exemplu în acest sens, este driver-ul pentru RAM-disk. Acesta se numește *vdisk.sys* sau *ram-drive.sys*, și poate să găzduiască un RAM-disk în memoria extinsă, începând cu versiunea 3.3 a lui MS-DOS.

Înainte de a utiliza *Extended Memory*, trebuie neaparat să verificați, dacă nu este deja instalat un RAM-disk. Dacă este cazul, atunci trebuie să determinați mărimea RAM-disk-ului. După aceea, puteți utiliza, pentru aplicațiile dumneavoastră, memoria aflată deasupra acestui domeniu. Pentru a stabili dacă este instalat un RAM-disk, trebuie știut cum funcționează, de fapt, acesta: un RAM-disk "schimbă direcția" vectorului de întrerupere 19 hex, spre o rutină proprie. În mod normal, întreruperea 19 hex este folosită pentru Boot-area sistemului de calcul. În principiu, noua rutină a driver-ului de RAM-disk, nu face altceva decât să redea controlul, service-rutinei "vechii" întreruperii 19 hex. Aceasta înseamnă că funcționarea computerului nu este limitată aici în nici un sens.

Ca orice vector de întrerupere, noul vector pentru întreruperea 19 hex, constă dintr-o adresă de offset și una de segment. Adresa segmentului, redă începutul segmentului de memorie, pentru noua rutină de tratare a întreruperii, iar adresa de offset indică spre prima comandă de executat din această rutină.

De cele mai multe ori într-un asemenea caz, adresa de offset este 0, deoarece începutul segmentului este concomitent și începutul rutinei. În cazul de față însă, startul rutinei de întrerupere nu este la începutul segmentului. Mai degrabă, acolo sunt depuși niște octeți de recunoaștere, prin care se poate constata dacă este, sau nu instalat un RAM-disk. Abia după aceea vine prima, și în același timp ultima, comandă, a noii rutine, comandă care apelează din nou vechea rutină a întreruperii 19 hex. Deci, întregul efect al noii rutine-service pentru întreruperea 19 hex, constă în faptul că, la începutul rutinei sunt așezați niște octeți de recunoaștere, care indică un RAM-disk instalat. Figura 4 arată, ca exemplu, primii 32 de Bytes ai întreruperii 19

	Conținut hexazecimal	Conținut zecimal
00hex	59 22 29 00 00 00 00 00	Y=).....
08hex	00 00 00 00 00 00 00 00
10hex	00 00 00 00 00 00 00 18
18hex	1C 21 3D 10 DF 10 4F 01	.! = . _ 0.

Figura 4 Primii 32 de Bytes ai întreruperii 19hex, fără RAM-disk

	Conținut hexazecimal	Conținut zecimal
00hex	00 00 E7 19 00 08 A9 00	..t...È.
08hex	D4 00 01 00 00 00 00 00	ș.....
10hex	00 00 56 44 49 53 4B 20	..VDISK
18hex	20 56 33 2E 33 28 00 00	V3.3(..

Figura 5 Primii 32 de Bytes ai întreruperii 19hex, cu 128 KBytes de RAM-disk în memoria extinsă

hex, când nu este instalat RAM-disk-ul, iar figura 5, primii 32 de Bytes, când în *Extended Memory* este instalat, pe 128 KBytes, un RAM-disk.

Această stare o obțineți, prin următoarea linie, introdusă în *config.sys*:

```
device=vdisk.sys128/e
```

Așa cum bine se poate vedea în cele două figuri, prezența unui RAM-disk este foarte ușor de stabilit. Pe când la o întrerupere 19 hex, cu RAM-disk-ul instalat, octeții 12 până la 1C hex conțin textul "VDISKV3.3", în aceeași octeți, cu RAM-disk-ul neinstalat, sunt doar semne disparate, în locul unui text inteligibil.

Chiar dacă poziția exactă în care se găsește textul în rutina de întrerupere diferă de la MS-DOS la MS-DOS, puteți să fiți siguri că un RAM-disk este instalat, deîndată ce textul "VDISK" se găsește undeva între cei 32 de Bytes citiți de la începutul segmentului întreruperii 19 hex. Oricum, nu este greu de scris o rutină care să determine, după metoda descrisă, dacă este sau nu instalat un RAM-disk.

Începutul RAM-disk-ului din *Extended Memory*, poate fi preluat, de asemenea, din primii 32 de Bytes ai rutinei de tratare a întreruperii 19 hex. Dacă în figura 5 urmăriți octeții 2C până la 2E, veți descoperi valorile 00, 00 și 01. Acești 3 Bytes reprezintă o adresă de 24 de biți, deci trebuie citiți împreună drept 010000hex (format Intel). Aceasta este adresa de start propriu-zisă a RAM-disk-ului. Deci RAM-disk-ul începe cu primul Byte de după limita de 1MByte. Dacă acum, consultați primii Bytes ai RAM-disk-ului, adică octeții începând cu celula de memorie

010000hex, găsiți acolo Boot-Block-ul RAM-disk-ului. Figura 6 arată conținutul primilor 32 de Bytes ai RAM-disk-ului.

În primul rând recunoașteți, cu ajutorul octeților de la 03 la 0A, că domeniul de memorie este efectiv ocupat de RAM-disk, pentru că aici se găsește, din nou, șirul de caractere "VDISK3.3".

În octeții 0B până la 1D, urmează blocul de parametri BIOS. În acesta sunt informații despre mărimea sectorului, mărimea cluster-ului, numărul de FAT-uri ș.a.m.d.

Blocul de parametri BIOS ai RAM-disk-ului, din figura 6, indică faptul că RAM-disk-ul lucrează cu 128 Bytes pe sector, folosește un sector pe cluster, conține o tabelă FAT, permite maximum 64 de intrări în directorul principal și deține o capacitate totală de 1024 de sectoare, adică 128 Kbytes.

Un mare interes prezintă următorii 2 Bytes, 1E și 1Fhex. Aceștia conțin (în Kbytes), adresa primului Byte din memoria extinsă, care nu mai este ocupat de RAM-disk. În figura 6 vedeți valorile 04 și 80hex care, recalculate, au ca rezultat cifra 480hex. De aceea, primul Byte liber de după RAM-disk, începe cu adresa 1024 + 128 Kbytes. Dacă citiți ambii Bytes cu adresele 1E și 1Fhex, puteți să stabiliți relativ simplu, unde începe domeniul liber al memoriei extinse. Aceasta vă deschide și posibilitatea marcării memoriei extinse drept ocupată, crescând dumneavoastră înșivă această valoare. Un program ulterior care ar dori de asemenea să folosească memoria extinsă, pornește de la ideea că, domeniul rezervat în plus de dumneavoastră este ocupat de RAM-disk, și îl lasă neatins.

Pe scurt, puteți utiliza Boot-Block-ul RAM-disk-ului, pentru a vă asigura că nici un alt program nu va "cotrobăi" prin datele dumneavoastră din *Extended Memory*.

Pentru aceasta stabiliți mai întâi dacă este instalat un RAM-disk, căutând în primii 32 de Bytes ai rutinei întreruperii 19 hex, șirul de caractere "VDISK". Dacă găsiți acest șir de caractere, atunci, în următorul pas, căutați dacă RAM-disk-ul începe în *Extended Memory*.

	Conținut hexazecimal	Conținut zecimal
00hex	00 00 00 56 44 49 56 4b	...VDISK
08hex	33 2E 33 80 00 01 01 00	3.3Ç....
10hex	01 40 00 00 02 FE 06 00	.@...#..
18hex	08 00 01 00 00 00 80 04Ç.

Figura 6 Primii 32 de Bytes ai RAM-disk-ului

Memoria extinsă

Dacă este așa, citiți mărimea din Byte-ul 1E și din 1Fhex, însumați domeniul pe care îl doriți rezervat pentru programul dumneavoastră, și salvați noua valoare, din nou în Boot-Block-ul RAM-disk-ului. În sfârșit puteți utiliza, din memoria extinsă de deasupra RAM-disk-ului, domeniul de memorie dorit, pentru rutinele și programele proprii.

După cum vedeți, există două procedee diferite de acces la memoria extinsă. Pe de-o parte, redirectând întreruperea 19 hex la o rutină proprie, astfel încât programele următoare vor fi induse în eroare, în sensul că ar fi mai puțină memorie instalată decât este realmente. Pe de altă parte, rezervându-vă memorie în *ExtendedMemory*, prin Boot-Block-ul unui RAM-disk. Deosebirea dintre cele două procedee este semnificativă: la metoda prin întreruperea 19 hex, memoria extinsă este ocupată "de sus în jos", iar la manipularea Boot-Block-ului RAM-disk-ului, dimpotrivă, "de jos în sus".

Cea mai bună este, ca de obicei, calea de mijloc: după start, programul dumneavoastră va determina pentru început mărimea memoriei extinse, prin întreruperea 15 hex. Apoi cercetează dacă există RAM-disk instalat în memoria extinsă. Dacă este instalat, se determină mărimea lui. Pentru aceasta, programul trebuie să pornească de la ideea că valoarea cea mai mică din cele determinate, este adevărata mărime a memoriei extinse disponibile - și anume disponibilă deasupra RAM-disk-ului. Acest domeniu de memorie puteți să-l folosiți, după aceea, fără probleme. Oricum, înainte de aceasta, mai trebuie să vă îngrijiți ca memoria să nu fie folosită de alte programe. Pentru aceasta, scrieți o rutină nouă pentru întreruperea 19 hex care, în locul valorii reale, întoarce o mărime a memoriei extinse, diminuată cu cantitatea de memorie pe care o folosiți dumneavoastră, când este apelată funcția 88 hex.

La fel, trebuie să adaptați și mărimea "oficială" a RAM-disk-ului, astfel încât să pară că memoria ocupată de dumneavoastră din *Extended Memory*, este ocupată tot de RAM-disk. Dacă nu este instalat deloc RAM-disk-ul, trebuie să ocupați un domeniu din memoria extinsă care să simuleze Boot-Block-ul unui RAM-disk adevărat. Astfel sunteți sigur că un program următor, care vrea de asemenea să utilizeze memoria extinsă și care pentru stabilirea mărimii memoriei folosește numai metoda RAM-disk-ului, nu se va suprapune din greșală peste datele dumneavoastră. Prin acest procedeu, datele dumneavoastră vor fi păstrate în memoria extinsă, atât de sigur cât este posibil.

Martin Althaus

Traducere:
ing Mihai Beer

```
/*
Nume: exramchk.c
Limbaaj: QuickC, TurboC si MS-C incepind cu versiunea 4.0
Cartela grafica: oricare
Particularitati: fara
*/

/*-----
EXRAMCHK.C

Calculeaza marimea memoriei extinse la
AT-uri sau PS/2 de la modelul 5.0 in
sus . Pentru aceasta este apelata func-
tia 88hex a intreruperii 15hex
-----*/

# include "dos.h"
int extram()
{
/*-----
aceasta functie intoarce marimea lui
Extended Memory
-----*/
union REGS cpu; /* register-variable*/
cpu.h.ah=0x88; /* funcia 88hex*/
cpu.h.al=0; /* stergere AL*/
int86( 0x15, &cpu, &cpu); /* apel
BIOS*/
return cpu.x.ax; /* intoarcerea valo-
rii*/
}

main()
{
if (extram() == 0)
{
printf(" \naparatul dvs este un
PC\n");
printf(" sau nu este instalata me-
moria extinsa\n");
}
else
{
printf( "\n sistemul dvs de-
tine");
printf( "%d KByte ", extram);
printf( "Extended Memory.\n\n");
}
}
}
```

"exramchk.c" folosește funcția 87 hex pentru a determina mărimea memoriei extinse.

Nume exramchk.pas
 Limbaj QuickPascal si TurboPascal ince-
 pind cu versiunea 4.0
 Cartela grafica oricare
 Particularitati fara

program exramchk;

(* -----
 Calculeaza marimea memoriei extinse, la
 AT-uri sau PS/2, de la modelul 5.0 in
 sus. Pentru aceasta este apelata func-
 tia 88hex a intreruperii 15hex.
 -----*)

uses dos; (* Biblioteca de accese ale
 CPU-ului*)

```
var
cpu: registers;
(* variabila pentru acces-BIOS*)
begin
writeln;writeln;
cpu.ah:=136; (* functia 88hex *)
intr($15,cpu); (* lansarea intrerupe-
rii*)
if (cpu.ax=0) or (cpu.flags and 1 = 1)
then (* comutare*)
begin
writeln ( 'sau aparatul dvs. este un
PC,');
writeln ( 'sau nu este instalata memo-
ria extinsa,');
writeln;
end
else
write ('sistemul dvs dispune de ',
cpu.ax, ' KBytes Extended Memory.');
```

"exramchk.pas" este versiunea Pascal a programului
 de determinare a mărimii memoriei extinse

MOVERAM.ASM

```
; Muta un bloc de memorie de 100 de
; octeti de la adresa 3000:0000 hex
; din memoria conventionala la adresa
; 0000:0000 hex in memoria extinsa
;
```

```
stiva segment para stack
dw 100h dup (?)
stiva ends
```

```
;
date segment para 'data'
; rezervat pt. GDT
gdt db 16 dup(0);
; Adresa sursei:
; lungimea segmentului
dw 0ffffh
; Adresa 030000 hex
db 0,0,3
; Drept de acces
db 97
; Rezervat
db 0,0
; Adresa destinatiei:
; Lungimea segmentului
dw 0ffffh
: Adresa 100000 hex
db 0,0,10h
; Drept de acces
db 97
; Rezervat
db 0,0
; Restul GDT-ului (stiva si
; segment de cod)
db 16 dup (0)
date ends
;
program segment para 'code'
assume cs:program,ds:date,
ss:stiva
start proc far
; Incarca adresa segmentului de date
in DS
mov ax,date
mov ds,ax
; 1000 octeti
mov cx,500
;Functia 87 hex
mov ah,87h
; DS pe stiva
push ds
; ES = DS
pop es
; Adresa GDT-ului
mov si,offset gdt
; Intrerupere
int 15h
; Terminarea programului
mov ax,4c00h
int 21h
start endp
program ends
end start
```

Rutina "moveram.asm" mută 1000 de octeți (500
 cuvinte) din memoria convențională în memoria
 extinsă.

Module fundamentale și module standard în dBASE și Foxpro

Vă prezentăm o aplicație standard, care utilizează limbajul celor două medii care dau titlul articolului. Aplicația aceasta nu face uz de meniurile puse la dispoziție de dBASE IV sau Foxpro2, fiind utilă în cazurile în care separația dintre utilizator și programator este suficient de profundă. În schimb, se ia în considerare principiul după care un program nu va fi niciodată în întregime rescris, ci se vor lua module testate în alte aplicații. Aceste module vor forma apoi infrastructura noii aplicații.

Reflecții de început

Să presupunem că un program dBASE, respectiv Foxpro, are două niveluri. Primul este reprezentat de un meniu-linie, iar al doilea de subprogramele aflate în legătură cu punctele acestuia. De fapt, cum se scrie un program? ... Metodistii studiază problema în cele mai mici amănunte - ceea ce este foarte important la lucrul în echipă, dar cere disciplină și profesionalism. Practicienii, în schimb, lucrează direct pe instrucțiuni - reinventează roata - ca apoi să constate că totul trebuie reluat de la început. Metoda prezentată aici pornește de la module fundamentale care definesc un mediu, în care modulele standard să poată lucra, ea impunând un stil de lucru disciplinat.

Programare înseamnă de fapt, transformarea unei idei într-un program. Acest lucru s-ar putea realiza în cinci pași:

- 1) Definirea problemei
- 2) Proiectarea soluției
- 3) Transcrierea în cod
- 4) Documentarea programului
- 5) Testarea

Al 2-lea pas conduce spre definirea punctelor meniului-linie principal. Pasul al 4-lea își dă măsura importanței când se fac modificări, și mai ales când și

altcineva trebuie să înțeleagă programul. Aceasta deoarece, după o pauză determinată de diverse motive, trece mult timp până înțelegi din nou programul, iar programele nedocumentate ale altora sunt aproape imposibil de înțeles. Dat fiind că testarea cere mult timp, ea nefăcându-se numai separat pe module ci și global, cu valori minime și maxime, cu valori eronate și apoi cu date reale, metoda își arată eficiența mai ales la al acest al 5-lea pas.

Structura programului

Un concept în care programele au întotdeauna aceeași construcție nu trebuie să fie neaparat rigid, ceea ce vom demonstra aici. Aplicațiile cu baze de date cer întotdeauna două tipuri de prelucrare, necesare și suficiente: prelucrarea interactivă și prelucrarea batch. Astfel, schema unei aplicații dBASE, propusă de noi, este prezentată în figura 1.

În partea de inițializare se face declararea variabilelor, deschiderea fișierului cu proceduri (detalii despre acesta vor apărea la momentul potrivit), și poziționa-

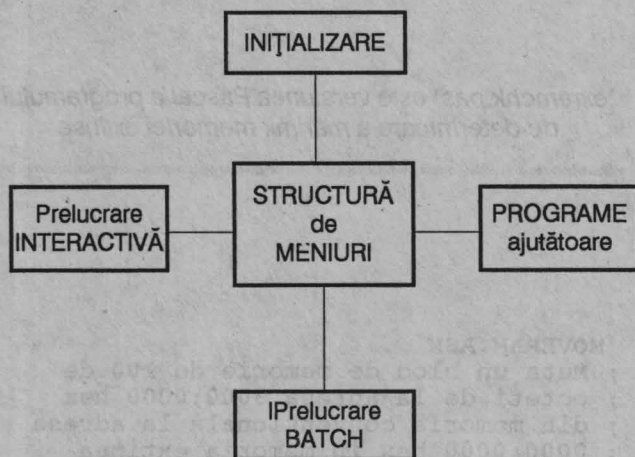


Figura 1

rea "comutatoarelor" dBASE (comenzile SET ...). Modulele : "inițializări" și "structura de meniuri" sunt așa-numitele module fundamentale, iar modulele "prelucrare interactivă", "prelucrare batch" și "programe auxiliare", sunt așa-numitele module standard. De ce fundamentale și de ce standard? ... Fundamentale pentru că sunt baza aplicației - dacă vrei și în ce privește înțelegerea superficială a ei, și standard deoarece creează și servesc un adevărat standard; ceea ce este particular fiecărei aplicații se completează apoi, ca într-un checklist. Ați recunoscut desigur în acest stil de lucru, pe cel al generatoarelor de programe, acestea din urmă fiind totuși mai rigide. Avantajele conceptului prezentat aici sunt: timp de dezvoltare și testare reduși, posibilități de eroare reduse și o mai mare transparență. Două observații vor completa cele spuse până acum. Prima este că un program în care fiecare parte este scrisă după o altă "filozofie", este foarte greu de înțeles, iar a doua este apropierea metodei de față de conceptele din domeniul CASE (*Computer Aided Software Engineering*), adică multă analiză și metodică.

Cu comenzile dBASE - respectiv Foxpro - se poate rezolva aproape orice problemă descrisă de un algoritm. Dar fiecare programator are de rezolvat sarcini care se repetă. Acestea pot fi rezolvate prin proceduri. Foarte des apare de exemplu întrebarea, adresată utilizatorului, dacă dorește sau nu efectuarea unei anumite acțiuni (decizie interactivă). Este preferabil ca aici să se scrie o procedură sau o funcție, decât să apară același modul de decizie, altfel scris în fiecare parte de program, "umflând" nejustificat codul. Apoi procedura sau funcția este găzduită într-un fișier numit "fișier cu proceduri", care în aplicația noastră apare ca modul PROCEDUR.PRG. Dacă la începutul programului apare comanda SET PROCEDURE TO PROCEDUR, această procedură de decizie poate fi apelată ca orice comandă dBASE. Modulul cu inițializări - în care apar variabilele de declarat PUBLIC, se poziționează comenzile SET și apare SET PROCEDURE TO PROCEDUR - este INIT.PRG. Sistemul de meniuri este controlat de MAIN.PRG. La aceste trei module fundamentale: PROCEDUR.PRG, INIT.PRG și MAIN.PRG, se adugă modulul fundamental MASCA.PRG, care reafixează ecranul principal - la revenirea în meniu din fiecare modul aflat în legătură cu punctul de meniu respectiv. În plus, modulul MASCA.PRG poate conține și comenzi utile în acest caz, cum ar fi cea de închidere a tuturor fișierelor-bază de date.

Modulele standard sunt MAINxy.PRG - care conține prelucrări în dialog cu utilizatorul și o structură de meniuri de nivel 2, și MAINPRN.PRG folosit pentru tipărirea diverselor rapoarte sau pentru "filtrarea" unei baze de date.

Există, după cum se vede, reguli după care se atribuie denumirea modulelor. Acestea sunt:

- patru litere care sugerează funcția
- două litere pentru nivelul în sistemul de meniuri.

Vom intra în câteva amănunte în ce privește aplicarea celei de-a doua reguli. MAIN11 este un modul care este apelat din primul meniu pull-down care apare la primul punct al meniului-linie principal. CAUTĂ11 este un subprogram de căutare din MAIN11. TIPĂ12 este un subprogram de tipărire lansat de al doilea punct al primului meniu pull-down menționat mai sus, ș.a.m.d.. A șaptea cifră poate fi folosită când sunt necesare, de exemplu, mai multe programe de tipărire: TIPĂ12a, TIPĂ12b, TIPĂ12c, iar a opta rămâne liberă pentru cazuri mai deosebite.

Modulele auxiliare sunt MAIN61.PRG pentru deschiderea tuturor bazelor de date și ștergerea - cu instrucțiunea PACK de altfel mare consumatoare de timp - a articolelor marcate pentru ștergere, și MAIN62.PRG care reface fișierele INDEX la căderi de tensiune, când sortarea în cadrul fișierelor este afectată. Aceste două module formează punctul 6 de meniu principal. Punctele de meniu principal 1-5 sunt rezervate pentru aplicații, iar punctul 8 este rezervat pentru gestiunea culorilor, a parametrilor și a variabilelor. Gestiunea culorilor folosite de un program nu este o problemă, dacă sunt prevăzute de la bun început variabile pentru acestea. Modificarea variabilelor pentru culori se face în modulul MAINCOLO.PRG. Variabilele globale sunt gestionate de MAINVARI.PRG. Dacă de exemplu punctul doi al meniului principal nu este ocupat dar dorim să introducem o aplicație nouă la acest punct, apar modificări și în MAINVARI.PRG. Variabilele locale, denumite aici parametri, vor fi gestionate în MAINPARA.PRG. Acesta are o construcție similară cu a lui MAINVARI, dar nu se modifică la un modul proaspăt integrat.

Din cauza limitărilor concepției lui dBASE, respectiv a lui Foxpro, valorile pentru culori, parametri și variabile, trebuie salvate în fișiere DBF - adică în CULOARE.DBF, PARAM.DBF și VARIABLE.DBF, care trebuie neapărat să existe pe suportul pe care sunt celelalte module, pentru a putea lansa în execuție întreaga aplicație asupra căreia ne concentrăm. VARIABLE.DBF își capătă structura numai într-o aplicație concretă.

Cu aceasta, părțile de program sunt: fișier cu proceduri, program de inițializare, program principal, mască principală, modul interactiv, modul batch, program de ștergere a articolelor vechi, program de reindexare, gestiunea culorilor, gestiunea parametrilor și gestiunea variabilelor. Schema completată apare în figura 2.

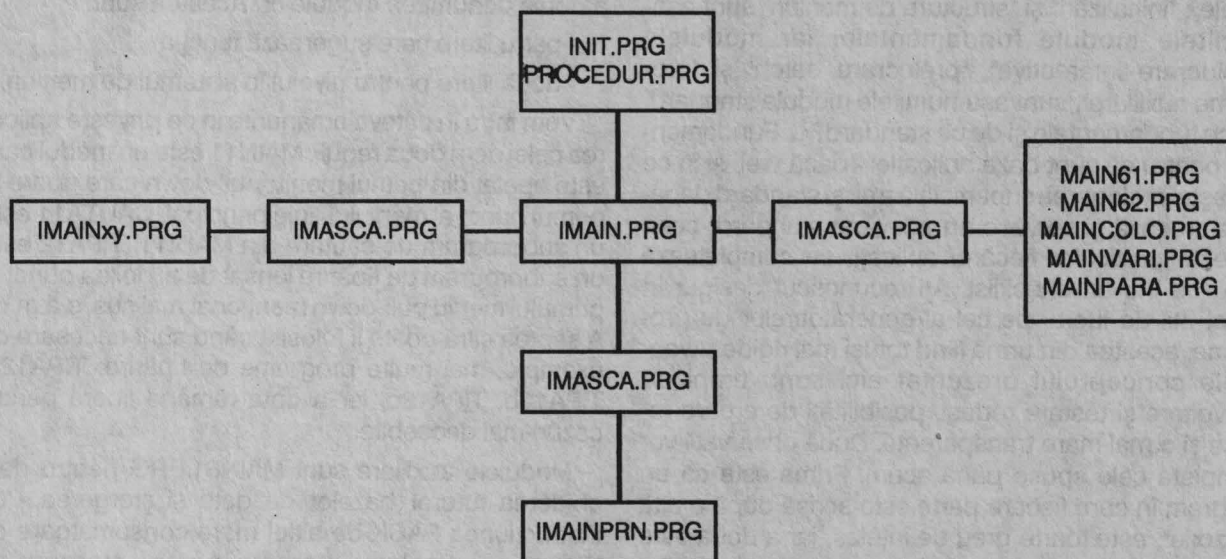


Figura 2

```

*
*MAIN.PRG Version 1.0 22.04.1990 16:44
*
* Programul controleaza meniurile pt.
* gestiunea adrese
*
* Programe si proceduri apartinatoare
* Fisiere accesate
* Descrierea variabilelor
* Modificari
**** COPYRIGHT B.F.E. - Soft****
do init          && Incarcare va-
riabile si parametri SET
do initfile      && Creare fisiere
inexistente si indexi
set clock off    && Decuplare ceas
do culoare with 1
clear
rind=1
do linie with 1
rind=22
do linie with 1
*** Initializarea valorii pt. coloane**
store 0 to VColoana1
store 0 to VSpalte2
store 0 to VColoana3
store 0 to VColoana4
store 0 to VColoana5
store 0 to VColoana6
store 0 to VColoana7
store 0 to VColoana8
store 0 to VColoana9
*** Textele de meniu din scheletul/
structura meniului
    
```

```

store "" to VM_Text1
store "" to VM_Text2
store "" to VM_Text3
store "" to VM_Text4
store "" to VM_Text5
store "Intretinere fisiere" to VM_Text6
store "" to VM_Text7
store "Parametri" to VM_Text8
store "Sfirsit" to VM_Text9
*** Calculul valorii pt. coloane - Cal-
culul blanc-urilor
if len(trim(VM_Text1))>0
    VColoana2=1
endif
if len(trim(VM_Text2))>0
    VColoana3=1
endif
if len(trim(VM_Text3))>0
    VColoana4=1
endif
if len(trim(VM_Text4))>0
    VColoana5=1
endif
if len(trim(VM_Text5))>0
    VColoana6=1
endif
if len(trim(VM_Text6))>0
    VColoana7=1
endif
if len(trim(VM_Text7))>0
    VColoana8=1
endif
if len(trim(VM_Text8))>0
    VColoana9=1
endif
    
```

```

endif
*** Calculul valorilor coloanelor
VColoana1=0
VColoana2=VColoana-
na2+len(trim(VM_Text1))+VColoana1
VColoana3=VColoana-
na3+len(trim(VM_Text2))+VColoana2
VColoana4=VColoana-
na4+len(trim(VM_Text3))+VColoana3
VColoana5=VColoana-
na5+len(trim(VM_Text4))+VColoana4
VColoana6=VColoana-
na6+len(trim(VM_Text5))+VColoana5
VColoana7=VColoana-
na7+len(trim(VM_Text6))+VColoana6
VColoana8=VColoana-
na8+len(trim(VM_Text7))+VColoana7
VColoana9=VColoana-
na9+len(trim(VM_Text8))+VColoana8
*** Ciclul principal****
define menu main
* define pad main1 of main prompt
"&VM_Text1" at 0,VColoana1
* define pad main2 of main prompt
"&VM_Text2" at 0,VColoana2
* define pad main3 of main prompt
"&VM_Text3" at 0,VColoana3
* define pad main4 of main prompt
"&VM_Text4" at 0,VColoana4
* define pad main5 of main prompt
"&VM_Text5" at 0,VColoana5
* define pad main6 of main prompt
"&VM_Text6" at 0,VColoana6
* define pad main7 of main prompt
"&VM_Text7" at 0,VColoana7
* define pad main8 of main prompt
"&VM_Text8" at 0,VColoana8
* define pad main9 of main prompt
"&VM_Text9" at 0,VColoana9
* on pad main1 of main activate popup
main1
* on pad main2 of main activate popup
main2
* on pad main3 of main activate popup
main3
* on pad main4 of main activate popup
main4
* on pad main5 of main activate popup
main5
* on pad main6 of main activate popup
main6
* on pad main7 of main activate popup
main7
* on pad main8 of main activate popup
main8
* on pad main9 of main activate popup
main9
*
* define popup main1 from 1,VColoana1
* define bar 1 of main1 prompt "";
* message ""
* define bar 2 of main1 prompt "";
* message ""
* define bar 3 of main1 prompt "";
* message ""
* define bar 4 of main1 prompt "";
* message ""
* define bar 5 of main1 prompt "";
* message ""
* define bar 6 of main1 prompt "";
* message ""
* define bar 7 of main1 prompt "";
* message ""
* define bar 8 of main1 prompt "";
* message ""
* define bar 9 of main1 prompt "";
* message ""
on selection popup main1 do main1
*
define popup main2 from 1,VColoana2
* define bar 1 of main2 prompt "";
* message ""
* define bar 2 of main2 prompt "";
* message ""
* define bar 3 of main2 prompt "";
* message ""
* define bar 4 of main2 prompt "";
* message ""
* define bar 5 of main2 prompt "";
* message ""
* define bar 6 of main2 prompt "";
* message ""
* define bar 7 of main2 prompt "";
* message ""
* define bar 8 of main2 prompt "";
* message ""
* define bar 9 of main2 prompt "";
* message ""
on selection popup main2 do main2
*
define popup main3 from 1,VColoana3
* define bar 1 of main3 prompt "";
* message ""
* define bar 2 of main3 prompt "";
* message ""
* define bar 3 of main3 prompt "";
* message ""
* define bar 4 of main3 prompt "";
* message ""
* define bar 5 of main3 prompt "";
* message ""
* define bar 6 of main3 prompt "";
* message ""
* define bar 7 of main3 prompt "";
* message ""
* define bar 8 of main3 prompt "";
* message ""
* define bar 9 of main3 prompt "";
* message ""
on selection popup main3 do main3
*
define popup main4 from 1,VColoana4

```

Baze de date

```
* define bar 1 of main4 prompt "";
* message ""
* define bar 2 of main4 prompt "";
* message ""
* define bar 3 of main4 prompt "";
* message ""
* define bar 4 of main4 prompt "";
* message ""
* define bar 5 of main4 prompt "";
* message ""
* define bar 6 of main4 prompt "";
* message ""
* define bar 7 of main4 prompt "";
* message ""
* define bar 8 of main4 prompt "";
* message ""
* define bar 9 of main4 prompt "";
* message ""
on selection popup main4 do main4
*
define popup main5 from 1,VColoana5
* define bar 1 of main5 prompt "";
* message ""
* define bar 2 of main5 prompt "";
* message ""
* define bar 3 of main5 prompt "";
* message ""
* define bar 4 of main5 prompt "";
* message ""
* define bar 5 of main5 prompt "";
* message ""
* define bar 6 of main5 prompt "";
* message ""
* define bar 7 of main5 prompt "";
* message ""
* define bar 8 of main5 prompt "";
* message ""
* define bar 9 of main5 prompt "";
* message ""
on selection popup main5 do main5
*
define popup main6 from 1,VColoana6
define bar 1 of main6 prompt "Sterge
date";
message "Datele marcate pt. stergere
sint sterse definitiv"
define bar 2 of main6 prompt "Rein-
dexare";
message "Reconstruire fisiere index"
* define bar 3 of main6 prompt "";
* message ""
* define bar 4 of main6 prompt "";
* message ""
* define bar 5 of main6 prompt "";
* message ""
* define bar 6 of main6 prompt "";
* message ""
* define bar 7 of main6 prompt "";
* message ""
* define bar 8 of main6 prompt "";
* message ""
* define bar 9 of main6 prompt "";
* message ""
* define bar 9 of main6 prompt "";
* message ""
on selection popup main6 do main6
*
define popup main7 from 1,VColoana7
* define bar 1 of main7 prompt "";
* message ""
* define bar 2 of main7 prompt "";
* message ""
* define bar 3 of main7 prompt "";
* message ""
* define bar 4 of main7 prompt "";
* message ""
* define bar 5 of main7 prompt "";
* message ""
* define bar 6 of main7 prompt "";
* message ""
* define bar 7 of main7 prompt "";
* message ""
* define bar 8 of main7 prompt "";
* message ""
* define bar 9 of main7 prompt "";
* message ""
on selection popup main7 do main7
*
define popup main8 from 1,VColoana8
define bar 1 of main8 prompt "Culori";
message "Stabilire culori pt. monito-
rul color"
define bar 2 of main8 prompt "Varia-
bile";
message "Stabilirea valorilor speci-
fice programului"
define bar 3 of main8 prompt "Parame-
trii";
message "Gestiune parametrarii"
* define bar 4 of main8 prompt "";
* message ""
* define bar 5 of main8 prompt "";
* message ""
* define bar 6 of main8 prompt "";
* message ""
* define bar 7 of main8 prompt "";
* message ""
* define bar 8 of main8 prompt "";
* message ""
* define bar 9 of main8 prompt "";
* message ""
on selection popup main8 do main8
*
define popup main9 from 1,VColoana9
define bar 1 of main9 prompt "Sfirsit";
message "Inapoi in Sistemul de Ope-
rare"
define bar 2 of main9 prompt "dBase";
message "Inapoi in dBase"
on selection popup main9 do main9
activate menu main
clear
return
```

```

procedure main1
do case
  case bar() = 1
*   do mask with .f. && Salvare masca
*   do main11
*   do mask with .t.
  case bar() = 2
*   do mask with .f. && Salvare masca
*   do main12
*   do mask with .t.
  case bar() = 3
*   do mask with .f. && Salvare masca
*   do main13
*   do mask with .t.
  case bar() = 4
*   do mask with .f. && Salvare masca
*   do main14
*   do mask with .t.
  case bar() = 5
*   do mask with .f. && Salvare masca
*   do main15
*   do mask with .t.
  case bar() = 6
*   do mask with .f. && Salvare masca
*   do main16
*   do mask with .t.
  case bar() = 7
*   do mask with .f. && Salvare masca
*   do main17
*   do mask with .t.
  case bar() = 8
*   do mask with .f. && Salvare masca
*   do main18
*   do mask with .t.
  case bar() = 9
*   do mask with .f. && Salvare masca
*   do main19
*   do mask with .t.
  otherwise
    ?? chr(7)
endcase

procedure main2
do case
  case bar() = 1
*   do mask with .f. && Salvare masca
*   do main21
*   do mask with .t.
  case bar() = 2
*   do mask with .f. && Salvare masca
*   do main22
*   do mask with .t.
  case bar() = 3
*   do mask with .f. && Salvare masca
*   do main23
*   do mask with .t.
  case bar() = 4
*   do mask with .f. && Salvare masca
*   do main24
*   do mask with .t.
  case bar() = 5
*   do mask with .f. && Salvare masca
*   do main25
*   do mask with .t.
  case bar() = 6
*   do mask with .f. && Salvare masca
*   do main26
*   do mask with .t.
  case bar() = 7
*   do mask with .f. && Salvare masca
*   do main27
*   do mask with .t.
  case bar() = 8
*   do mask with .f. && Salvare masca
*   do main28
*   do mask with .t.
  case bar() = 9
*   do mask with .f. && Salvare masca
*   do main29
*   do mask with .t.
  otherwise
    ?? chr(7)
endcase

*
procedure main3
do case
  case bar() = 1
*   do mask with .f. && Salvare masca
*   do main31
*   do mask with .t.
  case bar() = 2
*   do mask with .f. && Salvare masca
*   do main32
*   do mask with .t.
  case bar() = 3
*   do mask with .f. && Salvare masca
*   do main33
*   do mask with .t.
  case bar() = 4
*   do mask with .f. && Salvare masca
*   do main34
*   do mask with .t.
  case bar() = 5
*   do mask with .f. && Salvare masca
*   do main35
*   do mask with .t.
  case bar() = 6
*   do mask with .f. && Salvare masca
*   do main36
*   do mask with .t.
  case bar() = 7
*   do mask with .f. && Salvare masca
*   do main37
*   do mask with .t.
  case bar() = 8
*   do mask with .f. && Salvare masca
*   do main38
*   do mask with .t.
  case bar() = 9
*   do mask with .f. && Salvare masca
*   do main39
*   do mask with .t.

```

Baze de date

```
        otherwise
        ?? chr(7)
endcase

procedure main4
do case
    case bar() = 1
    * do mask with .f. && Salvare masca
    * do main41
    * do mask with .t.
    case bar() = 2
    * do mask with .f. && Salvare masca
    * do main42
    * do mask with .t.
    case bar() = 3
    * do mask with .f. && Salvare masca
    * do main43
    * do mask with .t.
    case bar() = 4
    * do mask with .f. && Salvare masca
    * do main44
    * do mask with .t.
    case bar() = 5
    * do mask with .f. && Salvare masca
    * do main45
    * do mask with .t.
    case bar() = 6
    * do mask with .f. && Salvare masca
    * do main46
    * do mask with .t.
    case bar() = 7
    * do mask with .f. && Salvare masca
    * do main47
    * do mask with .t.
    case bar() = 8
    * do mask with .f. && Salvare masca
    * do main48
    * do mask with .t.
    case bar() = 9
    * do mask with .f. && Salvare masca
    * do main49
    * do mask with .t.
    otherwise
    ?? chr(7)
endcase
*
procedure main5
do case
    case bar() = 1
    * do mask with .f. && Salvare masca
    * do main51
    * do mask with .t.
    case bar() = 2
    * do mask with .f. && Salvare masca
    * do main52
    * do mask with .t.
    case bar() = 3
    * do mask with .f. && Salvare masca
    * do main53
    * do mask with .t.
    case bar() = 4
    * do mask with .f. && Salvare masca
    * do main54
    * do mask with .t.
    case bar() = 5
    * do mask with .f. && Salvare masca
    * do main55
    * do mask with .t.
    case bar() = 6
    * do mask with .f. && Salvare masca
    * do main56
    * do mask with .t.
    case bar() = 7
    * do mask with .f. && Salvare masca
    * do main57
    * do mask with .t.
    case bar() = 8
    * do mask with .f. && Salvare masca
    * do main58
    * do mask with .t.
    case bar() = 9
    * do mask with .f. && Salvare masca
    * do main59
    * do mask with .t.
    otherwise
    ?? chr(7)
endcase

procedure main6
do case
    case bar() = 1
    do mask with .f. && Salvare masca
    do main61
    do mask with .t.
    case bar() = 2
    do mask with .f. && Salvare masca
    do main62
    do mask with .t.
    case bar() = 3
    * do mask with .f. && Salvare masca
    * do main63
    * do mask with .t.
    case bar() = 4
    * do mask with .f. && Salvare masca
    * do main64
    * do mask with .t.
    case bar() = 5
    * do mask with .f. && Salvare masca
    * do main65
    * do mask with .t.
    case bar() = 6
    * do mask with .f. && Salvare masca
    * do main66
    * do mask with .t.
    case bar() = 7
    * do mask with .f. && Salvare masca
    * do main67
    * do mask with .t.
    case bar() = 8
    * do mask with .f. && Salvare masca
    * do main68
    * do mask with .t.
```

```

case bar() = 9
* do mask with .f. && Salvare masca
* do main69
* do mask with .t.
  otherwise
    ?? chr(7)
endcase

procedure main7
do case
  case bar() = 1
* do mask with .f. && Salvare masca
* do main71
* do mask with .t.
  case bar() = 2
* do mask with .f. && Salvare masca
* do main72
* do mask with .t.
  case bar() = 3
* do mask with .f. && Salvare masca
* do main73
* do mask with .t.
  case bar() = 4
* do mask with .f. && Salvare masca
* do main74
* do mask with .t.
  case bar() = 5
* do mask with .f. && Salvare masca
* do main75
* do mask with .t.
  case bar() = 6
* do mask with .f. && Salvare masca
* do main76
* do mask with .t.
  case bar() = 7
* do mask with .f. && Salvare masca
* do main77
* do mask with .t.
  case bar() = 8
* do mask with .f. && Salvare masca
* do main78
* do mask with .t.
  case bar() = 9
* do mask with .f. && Salvare masca
* do main79
* do mask with .t.
  otherwise
    ?? chr(7)
endcase

procedure main8
do case
  case bar() = 1
    do mask with .f. && Salvare masca
    do mainfarb && Farbwerte ein-
tellen
    set clock off && Uhr ausschalten
    do farbe with 1
    clear
    zeile=1
    do linie with 1
    zeile=22
    do linie with 1
  case bar() = 2
    do mask with .f. && Salvare masca
    do mainvari && Variablenwerte
    verwalten
    do mask with .t.
  case bar() = 3
    do mask with .f. && Salvare masca
    do mainpara && Parameterver-
waltung
    do mask with .t.
  case bar() = 4
* do mask with .f. && Salvare masca
* do main84
* do mask with .t.
  case bar() = 5
* do mask with .f. && Salvare masca
* do main85
* do mask with .t.
  case bar() = 6
* do mask with .f. && Salvare masca
* do main86
* do mask with .t.
  case bar() = 7
* do mask with .f. && Salvare masca
* do main87
* do mask with .t.
  case bar() = 8
* do mask with .f. && Salvare masca
* do main88
* do mask with .t.
  case bar() = 9
* do mask with .f. && Salvare masca
* do main89
* do mask with .t.
  otherwise
    ?? chr(7)
endcase
*
procedure main9
do case
  case bar() = 1
    set color to w/
    clear
    quit
  case bar() = 2
    set color to w/
    clear
    return to master
  otherwise
    ?? chr(7)
endcase

* MAIN61.PRG Version 1.0 22.04.1990
16:44
*
* Programul sterge toate datele mar-
cate pt. stergere
* Programe si proceduri apartinatoare

```

Baze de date

```
* Fisiere accesate
* Descrierea variabilelor
* Modificari
**** COPYRIGHT B.F.E. - Soft****
do golit          && Sterge ecran
do titel with "Stergere"
do data
do culoare with 2
#@ 4, 2 say "Datele marcate pt. ster-
gere vor fi iremediabil sterse"
do culoare with 1
do maideparte with "Stergere date ve-
chi"
if .not. maideparte
    return
endif
do mesaj with "Va rog putina rabdare"
*** Lista fisierelor
*use <nume fisier>
*** Lista fisierelor
run del.?dx >nul  && Sterge fisiere
index
do initfile      && Reconstruire fis.
index
deactivate window mesaj
return

*
* MAIN62.PRG    Version 1.0 22.04.1990
16:44
*
* Programul reconstruieste fisierele
index
* Programe si proceduri apartinatoare
* Fisiere accesate
* Descrierea variabilelor
* Modificari
**** COPYRIGHT B.F.E. - Soft****
do golit
do titel with "Reindexare"
do data
do culoare with 2
@ 4, 2 say "Programul sterge fisierele
index (sortarile)"
@ 5, 2 say "si le reconstruieste ulte-
rior (resortare)"
@ 6, 2 say "Folositi acest program
atunci cind computerul"
@ 7, 2 say "dintr-o data nu mai ga-
seste datele introduse,"
@ 8, 2 say "lucru care se poate intim-
pla dupa o cadere a sistemului"
do culoare with 1
do maideparte with ""
if .not. maideparte
    return
endif
run del.?dx >nul  && Sterge fisiere
index

do initfile      && Reconstruire
fisiere index
return

* MASK.PRG    Version 1.0 22.04.1990
20:29
*
* Programul arata Masca de introduceri
* Programe si proceduri apartinatoare
* Fisiere accesate
* Descriere variabile
*
* afisare - va fi predat ca parametru
* if .t. - afisare continut anterior
al ecranului
* if .f. - salvare continut actual al
ecranului
* Modificari
**** COPYRIGHT B.F.E. - Soft****
parameters afisare
do culoare with 1
if afisare      && Resetare valori
    close databases  && Inchidere fi-
siere
    set margin to 0  && Asezare mar-
gine pe 0
    set console on  && Iesire pe
ecran cuplata
    set device to screen  && Iesire pe
ecran cuplata
    set print off    && Decuplare im-
primanta
    store .f. to tiparit  && Comutato-
rul pt. tiparire pus pe .f.
    store ctod(" . . ") to dela  &&
Resetare data calendar
    store ctod(" . . ") to pinala
&& Resetare data calendar
    set clock off    && Decuplare ceas
restore screen from mask
else
    save screen to mask
do culoare with 1
clear
rind=1
do linie with 1
rind=22
do linie with 1
endif
return

* INIT.PRG Versiunea 1.0 22.04.1990
20:58
*
* Programul incarca/initializeaza va-
riabilele comune
* Valabil pentru toate programele
* Programe si proceduri apartinatoare
* Fisiere accesate
```

```

* param - contine parametrii
* culoare - contine pozitionarea culo-
rilor
* Descrierea variabilelor
* Modificari
**** COPYRIGHT B.F.E. - Soft****
clear all          && Sterge toate
variabilele
*** Initializari generale
set talk off      && Nici un mesaj
set blocksize to 8 && Marimea cimpu-
rilor Memo
set date german   && Data in sistem
german
set deleted on    && Inlaturarea ar-
ticolelor sterse
set intensity off && Nici o afisare
in video-invers
set clock to 1,69 && Pozitia (pe
ecran) a ceasului
set hours to 24   && Afisaj pe 24
de ore
set clock off     && "Scoaterea"
ceasului
set status off    && Decuplarea afi-
sarii liniei de stare
set scoreboard off && Decuplarea afi-
sarii sistem
set instruct off  && Decuplarea li-
niei cu indicatii
set escape off    && Decuplarea in-
treruperii cu Escape
set safety off    && Fara mesaj la su-
prascr. fis. vechi
*** Deschiderea fisierului cu proce-
duri*
set procedure to procedure && Deschi-
dere fisier cu proceduri
select 1
use param
do param
close databases  && Inchidere fi-
sier cu parametru
*** Initializare generala a variabile-
lor*
public dela, pinala
&& Interogare diferita in timp
store ctod(" . . ") to dela, pinala
public introducere && Pentru o noua
introducere
public decautat    && Notiune comuna
de cautat
store "" to decautat
public tiparit     && Tiparire
public mai departe && Pt. procedura
mai departe
public VNuma       && Numele de cau-
tat
public VNrcлие     && Nr. clientu-
lui, de cautat

public nrarticol   && Intoarce alege-
rea facuta
store 0 to nrarticol
public tasta       && Numarului tas-
tei apasate
store 0 to tasta
public alegere     && Alegere din me-
niuri
store "" to alegere
public cautare     && Comutator pt.
cautare
public VData       && Variabila pt.
data interna
store date() to VData && Init./in-
carcare cu data curenta
public Lastkey     && Trebuie sa poa-
ta fi predata mai departe
public intrerupere && Intreruperea
tiparirii
store .f. to intrerupe
return

* INITFILE.PRG Version 1.0 7.03.1990
6:55
*
* Programul produce fisiere index
*
* Programe si proceduri apartinatoare
* Fisiere accesate
* Descrierea variabilelor
* Modificari
*
**** COPYRIGHT B.F.E. - Soft**** *
*if .not. file('<nume fisier>')
* use <numefisier>
* index on <cimpuri> tag <nume>
*endif
*
close databases    && Golire toate
domeniile
return

```

ing. Mihai Beer

*Din motive care țin, evident, de economisirea spațiu-
lui tipografic, nu prezentăm sursele tuturor module-
lor. Cei care doresc le pot obține de la readacție.*

Lumea culorilor VGA

(modurile în 256 culori)

Până acum am arătat cum se programează cartela VGA pentru a obține diferite moduri în 256 culori. Am programat astfel modurile 320x240 și 360x480. În programele care foloseau aceste moduri am folosit doar câteva culori, și de aceea în acest articol prezentăm un program care demonstrează unele posibilități de a obține efecte vizuale. În acest fel ne putem face o idee despre lumea culorilor VGA.

Deoarece până acum am folosit pentru majoritatea programelor modul 320x240, de această dată prezentăm o noutate: modul 360x360, pe care l-am obținut din modul 360x480 prin reprogramarea câtorva regiștri ai controlorului VGA, și anume: rezoluția pe verticală (360 linii în loc de 480), și centrarea pe verticală a imaginii în spațiul de afișare. Ce putem face cu acest mod în 256 culori?

Să observăm diferențele față de modul 360x480:

- raportul dintre densitatea pixelilor pe orizontală și cea pe verticală este de aproximativ 0.77 (modul 360x360) față de 0.56 (modul 360x480);
- un calcul simplu ne arată că acest mod permite paginarea memoriei ecran; în acest caz se poate lucra cu două pagini, rămânând un spațiu de memorie nefolosit de 368 octeți în fiecare din cele patru plane de memorie (în total 1472 octeți).

Față de modul 320x240 observăm că rezoluția este ceva mai mare, în schimb densitatea pixelilor nu este aceeași pe orizontală și pe verticală. Ambele moduri pot fi folosite cu succes pentru a realiza programe de animație după modelul prezentat în articolul din nr. 4/1992. Modelul va fi îmbunătățit într-un articol viitor, folosind câțiva regiștri cu funcții speciale ai controlorului VGA.

Prezentarea unui algoritm FloodFill

Au fost prezentate până acum următoarele moduri în 256 culori: 320x240, 360x360 și 360x480. Toate aceste moduri sunt nestandard, ele nefiind cunoscute de driver-ele uzuale (ex: EGAVGA.BGI, mouse etc). De altfel, EGAVGA.BGI nu cunoaște nici măcar modul

320x200, singurul mod standard în 256 culori. Nu putem avea deci acces la funcțiile din biblioteca grafică lucrând în aceste moduri, și de aceea aceste funcții trebuie să le construim cu forțe proprii. Pentru acest motiv prezentăm în cadrul acestui articol un model de procedură FloodFill, care va utiliza modul 360x360x256 prezentat mai sus. În legătură cu algoritmul folosit în cadrul acestui program trebuie făcute câteva precizări:

- procedura este puternic recursivă, de aceea stiva se epuizează extrem de rapid; dacă zona de umplut este destul de mare, poate apărea eroarea "Stack overflow error";
- această problemă poate fi rezolvată rescriind procedura ca și cum limbajul nu ar cunoaște recursivitatea, folosind în acest scop memoria liberă aflată la dispoziție;
- în cazul de față, pentru o rezoluție de 360x360 avem nevoie de maximum 360x360x2x2 (=518400) octeți pentru memorarea coordonatelor tuturor punctelor parcurse, în cazul în care se memorează coordonata fiecărui punct pe 4 octeți;
- se poate folosi o metodă destul de simplă de comprimare, folosindu-se pentru memorarea coordonatelor unui punct 3 octeți; în acest caz avem nevoie de maximum 360x360x3 (=388800) octeți.

mat. Ioan Cozac

NGRAFICA.PAS

```
unit ngrafica; {$D-,L-,S-}
interface uses dos;
const linmax=360; colmax=360;
procedure nsetgraph;
```

```

procedure nwritepixel(x,y:word;
p:byte);
function nreadpixel(x,y:word):byte;
procedure nsetcolor(c,r,g,b:byte);
procedure ngetcolor(c:byte; var
r,g,b:byte);
procedure nterminate;

implementation

var regs:registers; i,j,k:word;
    me:array[0..linmax-1,0..colmax div
4 - 1] of byte absolute $a000:0;

const valori:array[0..9] of byte=
($6d,$59,$5a,$8e,$62,$8b,$0b,$3e,0,$40);

const valorj:array[0..7] of byte=
($ad,$8c,$67,$2d,0,$e7,$04,$c3);

procedure nsetgraph;
begin
    regs.ax:=19; intr(16,regs);
    portw[$3c4]:=$604;
    portw[$3c4]:=$100;
    port[$3c2]:=$87;
    portw[$3c4]:=$300;
    port[$3d4]:=$11;
    port[$3d5]:=port[$3d5] and $7f;
    for i:=0 to 9 do
        begin
            port[$3d4]:=i;
            port[$3d5]:=valori[i];
        end;
    for i:=0 to 7 do
        begin
            port[$3d4]:=i+16;
            port[$3d5]:=valorj[i];
        end;
    portw[$3c4]:=$f02;
    fillchar(me,linmax*colmax div
2,#0);
end;

procedure calcule(x,y:word);
begin
    k:=x and 3;
    i:=y;
    j:=x div 4;
end;

procedure nwritepixel(x,y:word;
p:byte);
begin
    calcule(x,y);
    port[$3c4]:=2;
    port[$3c5]:=1 shl k;

```

```

    me[i,j]:=p;
end;

function nreadpixel(x,y:word):byte;
begin
    calcule(x,y);
    port[$3ce]:=4;
    port[$3cf]:=k;
    nreadpixel:=me[i,j];
end;

procedure nsetcolor(c,r,g,b:byte);
begin
    with regs do
        begin
            ah:=16; al:=16;
            bh:=0; bl:=c;
            ch:=g; cl:=b;
            dh:=r;
        end;
    intr(16,regs);
end;

procedure ngetcolor(c:byte; var
r,g,b:byte);
begin
    with regs do
        begin
            ah:=16; al:=21;
            bh:=0; bl:=c;
            intr(16,regs);
            g:=ch; b:=cl; r:=dh;
        end;
end;

procedure nterminate;
begin
    regs.ax:=3; intr(16,regs);
end;

end.

NMOD.PAS

uses ngrafica,dos,crt;
var i,j,k,l:integer;

function minim(a,b:integer):integer;
begin
    if a
        then minim:=a
        else minim:=b
end;

function maxim(a,b:integer):integer;

```

Laborator

```

begin
  if ab
    then maxim:=a
    else maxim:=b
end;

begin
  nsetgraph;

  for i:=0 to 44 do
    begin
      nsetcolor(i+1,0,0,i+19);
      nsetcolor(i+46,i+1,i+1,63);
    end;

  for i:=179 downto 0 do
    begin
      k:=i+(180-i)*2-1;
      l:=(179-i) div 2 + 1;
      for j:=0 to (180-i)*2-1
do
  begin
    nwritepixel(i+j,i,l);
    nwritepixel(i,i+j,l);
    nwritepixel(k,i+j,l);
    nwritepixel(i+j,k,l);
  end;
  delay(i div 2);
end;

  for i:=1 to 45 do
    for j:=1 to 45 do
      begin
        nsetcolor(j,minim(i,46-j),
          minim(i,46-j),
          minim(j+19,63));
        nsetcolor(j+45,maxim(j-
i,0),
          maxim(j-i,0),
          maxim(64-i,
          64-j));
      end;

      nsetcolor(255,0,0,0);
      for i:=179 downto 0 do
        begin
          for j:=0 to (180-i)*2-1 do
            begin
              k:=i+(180-i)*2-1;
              nwritepixel(i+j,i,255);
              nwritepixel(i,i+j,255);
              nwritepixel(k,i+j,255);
              nwritepixel(i+j,k,255);
            end;
            delay(i div 2);
          end;
        end;
      end;

  for i:=0 to 63 do
    begin
      nsetcolor(255,i,0,0);
      delay(60);
    end;
  for i:=0 to 63 do
    begin
      nsetcolor(255,63,i,i);
      delay(60);
    end;
  for i:=0 to 59 do
    begin
      nsetcolor(i+1,i+4,0,0);
      nsetcolor(i+61,63,0,i+4);
      nsetcolor(i+121,63-
i,0,63);
      nsetcolor(i+181,0,0,63-
i);
    end;
  for i:=0 to 359 do
    begin
      for j:=0 to i do
        nwritepixel(j,i-j,
          j div 3 + (i-j) div
          3 + 1);
      delay((359-i) div 8);
    end;
  for i:=0 to 359 do
    begin
      for j:=i to 359 do
        nwritepixel(j,359+i-
          j, j div 3 + (359+i-j)
          div 3 + 1);
      delay(i div 8);
    end;
  for i:=240 to 255
do nsetcolor(i,0,
  (i-240)*2+30,0);
  for i:=359 downto 0 do
    begin
      for j:=0 to 359-i do
        nwritepixel(j,i+j,
          (i+1) div 24 + 240);
      for j:=i to 359 do
        nwritepixel(j,j-i,
          (i+1) div 24 + 240);
      delay(i div 8);
    end;
  nterminate;
end.

```

NFILL.PAS

```

{$M 65520,0,655360}
uses ngrafica;

var culoarefond:byte;

procedure nfloodfill(x,y:word; c:byte);
procedure fillzone(x,y:word);
begin
  nwritepixel(x,y,c);
  if nreadpixel(x,y-1)=culoarefond
    then fillzone(x,y-1);
  if nreadpixel(x-1,y)=culoarefond
    then fillzone(x-1,y);
  if nreadpixel(x,y+1)=culoarefond
    then fillzone(x,y+1);
  if nreadpixel(x+1,y)=culoarefond
    then fillzone(x+1,y);
end;
begin
  culoarefond:=nreadpixel(x,y);
  fillzone(x,y);
end;

procedure oline(xi,xf,y:word; c:byte);
var x:word;
begin
  if xixf then
    begin
      x:=xi;
      xi:=xf;
      xf:=x;
    end;
  for x:=xi to xf do nwrite-
pixel(x,y,c);
end;

procedure vline(x,yi,yf:word; c:byte);
var y:word;
begin
  if yiyf then
    begin
      y:=yi;
      yi:=yf;
      yf:=y;
    end;
  for y:=yi to yf do nwrite-
pixel(x,y,c);
end;

procedure nrectangle(xi,yi,xf,yf,
cul:integer);
begin
  oline(xi,xf,yi,cul);
  oline(xi,xf,yf,cul);
  vline(xi,yi,yf,cul);
  vline(xf,yi,yf,cul);

```

```

end;

begin
  nsetgraph; nsetcolor(5,63,31,0);
  nrectangle(1,1,358,358,5);
  nsetcolor(12,31,63,0);
  nsetcolor(9,31,0,63);
  nsetcolor(6,47,0,63);
  nrectangle(140,140,210,210,5);
  nfloodfill(180,180,12);
  nrectangle(10,10,80,80,5);
  nfloodfill(30,30,9);
  nrectangle(250,250,320,320,5);
  nfloodfill(300,300,6);
  nrectangle(140,20,210,90,5);
  nfloodfill(180,45,6);
  nrectangle(140,250,225,320,5);
  nfloodfill(175,275,9);
  nrectangle(10,250,75,325,5);
  nfloodfill(15,275,12);
  nrectangle(280,25,320,125,5);
  nfloodfill(300,50,12);
  nrectangle(15,160,90,225,5);
  nfloodfill(25,175,6);
  nrectangle(275,150,325,240,5);
  nfloodfill(280,175,9);
  nrectangle(120,120,225,225,5);
  nsetcolor(255,63,63,63);
  nfloodfill(125,125,255);
  readln;
  nterminate;
end.

```

Noi sunete din IBM-PC

Vă propunem mai jos un mic program, scris în C, care generează zgomote "simpatice" cu ajutorul difuzorului instalat pe orice calculator compatibil IBM-PC. El ar putea constitui fie nucleul unor rutine proprii de zgomote, fie o metodă sigură de a vă exaspera colegii de birou.

Programul newsound.c a fost compilat cu Borland C 3.0.

Cristian Nagy

```
#include <dos.h>
#include <stdlib.h>

void noise(int f, int b, int t)
{
    int i;

    b = f >> b;
    f -= b >> 1;
    for( i = 0; i < t; i++)
        sound(f+ random(b));
}

void soundmix(int f1, int f2, int t)
{
    sound(f1);    delay(t);
    sound(f2);    delay(t);
}

int main(void)
{
    int L;

    /* 1 */
    for( L = 10000; L >= 20; L--)
        noise( L, 1, 1 );
    /* 2 */
    for( L = 100; L <= 1000; L++ )
        soundmix( L, 5000, 2);
    for( L = 1000; L >= 100; L-- )
        soundmix( L, 250, 1 );
    /* 3 */
    for( L = 0; L < 3; L++ )
        soundmix( 1000, 800, 500 );
    /* 4 */
    for( L = 1000; L >= 100; L-- )
        noise( L, 0, 3 );
    for( L = 8000; L <= 12000; L++ )
        noise( L, 0, 1 );
    nosound();    /* Nu uita! */
    return 0;
}
```

Afișarea programelor din memorie

Sistemul de operare poate încărca în memorie, în vederea execuției, două tipuri de programe: din fișiere cu extensia COM și EXE. Pentru ambele tipuri, încărcătorul de programe va alege o porțiune din memorie cu mărimea de 10 (hex) paragrafe (1 paragraf = 10 hex octeți), numită *Prefixul Segmentului Program* - PSP, care va constitui baza memoriei alocate programului. Putem recunoaște un PSP după instrucțiunea cu care începe - INT 20H (terminare program).

Deasupra PSP va fi încărcat un segment de cod (pentru programele de tip COM) sau mai multe segmente (cod, date, stivă) pentru programele de tip EXE.

Încărcătorul va alocă fiecărui program încărcat o porțiune de memorie numită *context*, în mărime de până la 32K. Adresa contextului va fi scrisă în PSP, la offset-ul 2C hex.

Ne propunem să afișăm toate programele EXE și COM existente la un moment dat în memorie, precum și spațiul ocupat de fiecare.

În acest scop folosim funcția 52H a întreruperii 21H, care ne furnizează (printre altele) și adresa primului bloc de memorie folosit de programele încărcate.

La începutul fiecărui bloc de memorie (context, PSP, cod) se află un header de control având următoarea structură:

Tipul blocului poate fi "M" - pentru un bloc intermediar, sau "Z" - pentru ultimul bloc alocat. Dacă blocul este intermediar, putem găsi următorul bloc peste N paragrafe, unde N este mărimea blocului (în header, offset 3).

Deținătorul blocului poate fi un alt bloc (de exemplu deținătorul unui bloc de tip PSP este blocul de cod al programului respectiv). Dacă blocul nu are deținător, în această poziție apare 0.

Blocul de memorie efectiv are mărimea precizată, în header la offset-ul 3, și începe în paragraful imediat următor (baza header + 10 hex).

Spațiul rezervat conține, în cazul blocurilor fără deținător, chiar numele programului aflat în memorie. O altă metodă pentru aflarea numelui programului este de a căuta în blocul context deținut de acel program, dar nu dă întotdeauna rezultate.

Propun în continuare un program bazat pe informațiile de mai sus. În plus, se folosește întreruperea BIOS 12H pentru aflarea memoriei convenționale totale. Pentru o funcționare corectă, el nu va fi lansat direct din IDE, ci din linia de comandă DOS.

```
#include <dos.h>
#include <stdio.h>

#define Int20 0x20cd

struct MCB
{
    char type;
    int owner;
    unsigned size;
    char res[3];
    char name[8];
    int psp;
} far *s;

void main(void)
{
    int memkb;

    asm int 12h
    asm dec ax
    asm mov memkb,ax
    asm push es
    asm mov ah,52h
    asm int 21h
    asm mov ax,es:[bx-2]
    asm pop es

    s = MK_FP( _AX, 0 );
    printf(" Paragraph Owner Bytes
Program\n");
    printf(" ----- ---- ----
-----\n");
    while( s->type != 'Z' )
    {
        if( s->psp == Int20 )
            printf( " %04x %04x %6u
%-8Fs\n",FP_SEG( s ), s->owner,
(s->size) << 4, s->name );
        else
        {
            printf( " %04x %04x %6u
", FP_SEG( s ),s->owner, (s-
>size) << 4 );
            if(s->owner) printf( "\n" );
            else printf( "--Free--\n" );
        }
        s = MK_FP( FP_SEG(s) + s->size+1,
    )
    }
    printf( "Available memory: %d
kbytes\n\n", memkb - (_psp >> 6) );
}
```

Schimbător de paletă pentru VGA monocrom

Pentru interfața VGA există multe programe și jocuri care expun pe ecran o adevărată demonstrație de culori. Însă dacă aveți un monitor monocrom, pot să apară probleme la reprezentarea unui număr mare de culori. Motivul este că majoritatea acestor monitoare reprezintă numai semnalele pentru verde, ca nuanțe de gri. Părțile roșii și albastre ale imaginii rămân invizibile.

"VGA2GRI" încearcă să ofere o soluție. După lansare, programul se instalează rezident în memorie. El interceptează întreruperea 10 hex (întreruperea video) și se informează cu privire la schimbarea modului video și la accesul la paleta de culori a interfeței VGA. În ambele cazuri, se execută mai întâi întreruperea 10 hex originală. Apoi se modifică regiștrii microprocesorului și se apelează din nou întreruperea 10 hex pentru a schimba paleta de culori în nivele de gri. Pentru aceasta, se apelează o funcție a BIOS-ului cu parametri:

AH = 10 hex

AL = 1B hex.

Se vor înscrie aceleași valori în cele trei palete de culori, obținându-se o bună reprezentare - pe un monitor monocrom - a culorilor sub formă de nivele de gri.

Unele programe ocolesc întreruperea 10 hex, lucrând direct cu interfața VGA. Deoarece programul prezentat nu "vede" astfel de operații, a fost prevăzută o comutare manuală cu ajutorul combinației de taste *Alt - Shift dreapta - Shift stânga*.

Programul poate fi "creat" cu utilitarul *debug* sub forma unui fișier COM:

```
debug < vga2gri.deb > vga2gri.lst
```

Apoi, în "vga2gri.lst" se pot vedea mesajele debugger-ului. Programul se lansează fără parametri (eventual chiar din AUTOEXEC.BAT). Funcția are nevoie de câteva secunde pentru a recalcula paleta, timp în care ecranul va fi șters.

În program veți găsi, după instrucțiunea de salt la partea de instalare, șase instrucțiuni NOP. Ele servesc la rezervarea spațiului în care se vor salva vectorii vechilor întreruperi.

Helmut Vogler

Traducere și adaptare:

ing. Mihai Beer

Tips & Tricks

```
' Program: vga2gri.deb
' Functi: modificarea paletii de culori
' Limbaj: debug vga2gri.deb

n vga2gri.com
a
jmp 165
nop
nop
nop
nop
nop
nop
pushf
cs:
call far [100]
mov ax,101b
mov bx,0
mov cx,100
cs:
jmp far [100]
cmp ah,0
je 108
cmp ah,b
je 108
cmp ah,10
jne 117
cmp al,2
jbe 108
cmp al,10
jb 117
cmp al,13
jbe 108
jmp 117
push ds
push ax
mov ax,40
mov ds,ax
mov al,[17]
and al,b
cmp al,b
pop ax
pop ds
jne 160
push ax
push bx
push cx
mov ax,101b
mov bx,0
mov cx,100
pushf
cs:
call far [100]
pop cx
pop bx
pop ax
cs:
jmp far [104]
mov ax,0
mov es,ax
mov ax,cs
mov ds,ax
es:

mov ax,[40]
mov [100],ax
es:
mov ax,[42]
mov [102],ax
es:
mov ax,[70]
mov [104],ax
es:
mov ax,[72]
mov [106],ax
mov ax,cs
pushf
cli
es:
mov [42],ax
es:
mov [72],ax
mov ax,11c
es:
mov [40],ax
mov ax,139
es:
mov [70],ax
popf
mov ax,101b
mov bx,0
mov cx,100
pushf
cs:
call far [100]
mov ah,9
mov dx,1c0
int 21
mov dx,165
int 27
db 'Comutator VGA-Mono instalat'
db a,d
db '-- HotKey: ALT+SHIFT-ri'
db '+SHIFT-le'
db a,a,'$'

rcx
238
w
q
```

Editorial	3
----------------------------	----------

Noutăți

Disc optic reinscriptibil IBM 3431	4
IBM PS/2Workgroup Server 85	4
IBM PS/ValuePoint2	4
Calculatoarele mari consemnează pierderi pe piață	4
București 19-22 mai 1993	5
26 mai 1993 (la sediul AGIR)	6

Efecte speciale

Efecte speciale	7
Pictând cu ajutorul numerelor	8
Mișcări iscusite	9
Cum s-a realizat fața lui "Lawnmower Man" 10	
Eșantionarea vedetelor	11

Aplicații

Gestiunea producției pe PC	12
1. Introducere	12
2. Prezentarea subsistemului informatic BD-LANS (baza de date - lansare în producție).	12
3. Pachetul de programe BD-LANS	14

Fundamente

O privire asupra harddiscurilor	16
Organizarea datelor	17
ZBR creează simțitor mai mult spațiu	17
Factorul Interleave	17
Interfața cu computer-ul	18
SCSI-interfața universală	18
Procedee de înregistrare	19
Corectare erorilor	21
Viteza	21
Organizarea harddisc-urilor	
Structuri de discuri	23
Medii de memorare optice	30

CD-ROM - o memorie care poate fi numai citită	30
WORM - (re) inscriptil o dată	31
Alternativa: memorii magneto-optice	33
Phase Change Tehnology	33

Hardware

Reușită în a cincea dimensiune	35
Superscalar - din doi facem unul	35
BPL prezice	36
Viteză dublă în ciuda păstrării frecvenței	37
Pentium - minunea calculului	37
Rezumat	38

Memoria extinsă

Lanțuri aruncate în aer	
Programarea lui	
<i>Extended Memory</i>	39

Baze de date

Module fundamentale și module standard în dBASE și Foxpro	46
Reflecții de început	46
Structura programului	46

Laborator

Lumea culorilor VGA (modurile în 256 culori)	56
Prezentarea unui algoritm FloodFill	56



str. Gheorghe Doja nr. 36
C.P. 64, O.P. 1
4300 TÂRGU-MUREȘ
Tel./Fax: 0954-31660

va edita, în curând, un interesant și util
manual de utilizare pentru

Windows 3.1

Cum este alcătuită cartea

Introducere

Partea introductivă oferă informații de bază despre Windows: o privire de ansamblu asupra funcțiilor, a hardware-ului necesar, precum și informații de culise asupra memoriei.

Esențialul în Windows

Această parte conține funcțiile cele mai importante și interesante, prezentate prin exemple și numeroase ilustrații. Veți învăța operarea de bază și funcțiile avansate ale programului.

Teme care ne conduc mai departe

Acest capitol se concentrează asupra unei colecții de sfaturi practice și trucuri privind mediul Windows.