if

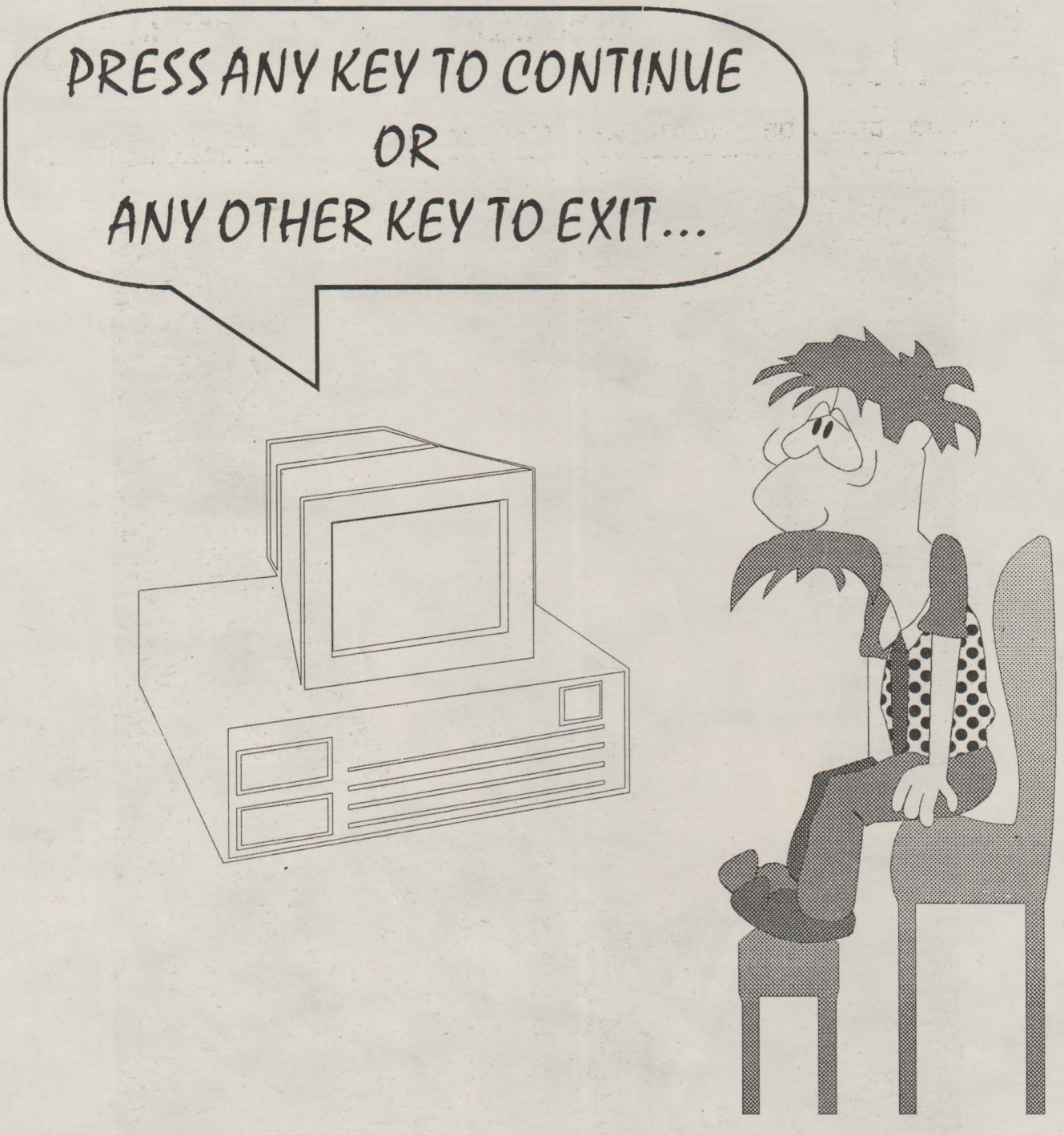
CALCULATOARE
PERSONALE

Nr. 17

500 lei

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

ISSN 1220-1529


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

if

revistă de informatică

editată de firma

micro ATCI S.R.L.

Director:
ing. Dumitru Dunca

Redacţia:
ing. Cristian Nagy
ing. Mihai Beer
mat. Ioan Cozac
Keresztes Laszlo

Adresa redacţiei:
Micro ATCI - revista "if"
C.P. 64, O.P. 1
4300 Târgu-Mureş
Tel/Fax: 0954-31660

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

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

Ne face o mare plăcere să putem prezenta şi o aplicaţie ceva mai îndrăzneaţă elaborată la noi în ţară (vezi articolul despre gestiunea producţiei pe PC) care atacă un domeniu sensibil şi neglijat: lansarea şi urmărirea producţiei în firme care produc în serie.

Vă mai atragem atenţia şi asupra articolului cu privire la elaborarea aplicaţiilor pornind de la module standardizate. Este un mod de abordare care poate uşura enorm munca de proiectare şi de întreţinere. Dacă nu credeţi, încercaţi! Deoarece sursele tuturor modulelor menţionate în articol ar ocupa mult prea mult spaţiu tipografic, nu reproducem decât o parte. Cei interesaţi le pot obţine pe toate de la redacţie (probabil că ar fi preferabilă soluţia cunoscută a unei dischete).

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

Dacă aveţi alte preferinţe sau idei nu ezitaţi să ne scrieţi. Mai ales dacă ideile se concretizează în materiale care să fie publicate în "if".

În încheiere vă asigurăm de dorinţa noastră (dublată de eforturi susţinute) de a avea o apariţie ceva mai regulată.

Cristian Nagy

Apariţia acestui număr a fost susţinută parţial de către

FUNDAŢIA SOROS PENTRU O SOCIETATE DESCHISĂ

Noutăţi 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 şi citire. El poate fi folosit pentru baze de date şi pentru stocarea temporară a fişierelor mari cum ar fi cele folosite de produsele multimedia.

Caracteristicile sale sunt:

- interfaţă SCSI standard
- tehnologie avansată ce oferă fiabilitate sporită
- 650 MB per disk utilizând sectoare de 1024 octeţi (numai sub os/2 cu High Performance Optical File system - HPOFS)
- 595 MB per disc utilizând sectoare de 512 octeţi (sub DOS şi 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 - 0XA, V01) sunt calculatoare puternice care îmbină performanţele procesorului cu o mare capacitate de stocare a datelor şi cu posibilităţi sporite de extindere a sistemelor. Aceste echipamente sunt proiectate pentru a acoperi cererile pentru servere de reţea, calculatoare host multi-user şi aplicaţii teehnice 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 - spaţiu disc de până la 2,63 GB
- adaptoare grafice ce oferă rezoluţii SVGA şi XGA-2
- SCSI cu cache integrat pe placa de bază
- tehnologie IBM ECC-P - asigură Controlul şi Corecţia Erorilor folosind memorii SIMM standard.

IBM PS/ValuePoint2

Noile IBM PS/ValuePoint2 6382 şi 6384 au fost proiectate pentru toate domeniile - în special pentru cele care cer performanţe ridicate, memorie, grafică SVGA, şi posibilitatea de a adăuga un mare număr de periferice IBM şi OEM.

Caracteristici:

tehnologie VESA Local Bus

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

Maşinile 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 piaţă

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

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

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

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

(9,4%), Hitachi (7,5%) şi NEC (6.3%). Firmele americane Unisys şi Amdahl, au căzut de pe 2 pe 5.

Pentru PC-uri, situaţia 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 faţă de anul anterior)
- Compaq 6,6% (+0,6%)
- NEC 5,1% (venind de pe locul 3)
- Dell (venind de pe locul 11, şi dublându-şi cifra de afaceri faţă de anul anterior)

Cei care doresc să reziste pieţii în anul 1993, trebuie să dovedească în bună măsură, inventivitate şi 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 şi-a câştigat un nume mai ales ca producător de interfeţe grafice cu rezoluţie ridicată pentru CAD. Cel mai recent produs are la bază o colaborare strânsă cu specialiştii în grafică de la Silicon Graphics (SGI). Pe placă se află (cum altfel ?) şi 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 şi 8 până la 32 MB memori de lucru. Şi totul costă "numai" 5960 DM.

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

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

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

Reţea de computere fără fir

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

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

dpa

Primul Simpozion Internaţional de INFORMATICĂ ECONOMICĂ

Bucureşti 19-22 mai 1993

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

Produsele-program prezentate se refereau la domeniul financiar-contabil şi al mijloacelor fixe, acestea din urmă fiind prezentate într-o nouă filozofie - a mentenanţei, stocurilor şi achiziţiilor. 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 Asociaţiei Generale a Inginerilor din România.

Tot pe linie CASE (Computer Aided Software Engineering) au mers şi 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 staţii SPARK 10 şi a unei casete video referitoare la domeniul prelucrării mecanice moderne.

Noutăţi de ultimă oră în domeniu, se puteau afla în discuţii cu oamenii de la "Scop" , proaspăt sosiţi de la "CeBit"- Hanovra.

Conferinţă de presă SIVECO-România

26 mai 1993 (la sediul AGIR)

Conferinţa de presă a fost precedată de un simpozion (continuat şi a doua zi) în cadrul căruia a fost prezentat amănunţit produsul COSMAN. Este vorba de un produs de mentenanţă, stocuri şi achiziţii al firmei SIVECO (cu sediul central la Paris şi având filiale în Marea Britanie Elveţia şi România). Domnii Quintin J. Thorn şi Carlo Fighera din cadrul grupului Siveco au condus conferinţa de presă, asistaţi fiind de doamna Irina Socol de la Siveco România şi de domnul Mircea Tiberiu Gruiescu, gazda conferinţei.

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

- Siveco are o cifră de afaceri de 4 milioane $
- investiţia făcută până acum în România este de 500.000 de franci francezi, cuprinzând numai transport şi informare
- ideea a apărut în urma unor aşa-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, aşa cum s-a întâmplat de altfel, să devină un program pentru toată lumea - adică un pachet de programe pentru un domeniu de aplicaţii
- 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 atenţia prin două lucruri: capacitatea inginerilor de aici, şi cunoaşterea de către mulţi români a limbii franceze
- obiective:
  - creare unui centru de studii
  - clienţi: marile uzine dar nu numai
  - transfer maxim de "know-how"
- mod de lucru: module standard
- domeniul mentenanţei 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 substanţial îmbunătăţită
- suportul hard este reprezentat de maşini MS-DOS, programele sunt scrise în C, baza de date este una americană de tip RDEB, reţeaua de tip Novell cu peste 50 de posturi de lucru
- politica de preţuri este flexibilă
- produsul este foarte indicat pentru Întreprinderi care nu au resurse financiare de retehnologizare, dar nici nu pot merge mai departe aşa cu utilajele
- unde este vândut COSMAN este lider
- acest produs poate fi uşor configurat funcţie de particularităţi
- pregătirea pentru utilizare cuprinde 4 cicluri a câte 5 zile
- strategia: România va dezvolta produsul pentru estul Europei
- domeniile de aplicaţie încep în producţia de avioane şi automobile, trec prin cea de parfumuri, ajungând în domeniul întreţinerii clădirilor
- cea mai nouă comandă a fost preluată pentru gestiunea deşeurilor.

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

Efecte speciale

Producătorii de filme nu se mai mulţumesc să inventeze poveşi şi să-i facă pe oameni să pară altfel decât sunt, ei inventează acum noi lumi cu ajutorul calculatoarelor. Ideile trec din imaginaţie direct pe marele ecran în câţiva paşi - nu chiar atât de simpli, iar realitatea rămâne în urmă.

Un grădinar coboară în iad. În faţă îl izbesc culori clipitoare, vârtejuri ameţitoare şi imagini strălucitoare. Fiinţele umane se prefac brusc în particule care vibrează şi se rotesc violent în aer.

Aceasta nu este o experienţă 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 şi a avut premiera la 5 iunie 1992), suferă un virulent atac din partea efectelor speciale. Un fost militar plin de intenţii rele, pe nume Dr. Angelo (interpretat de Pierce Brosman), folosind droguri şi un costum conectat la un computer, l-a aruncat pe bietul Jobe în lumea de coşmar a Realităţii Virtuale.

Dintotdeauna au existat fani care au stat la coadă pentru a-şi 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 şi modifică modul de realizare a filmelor.

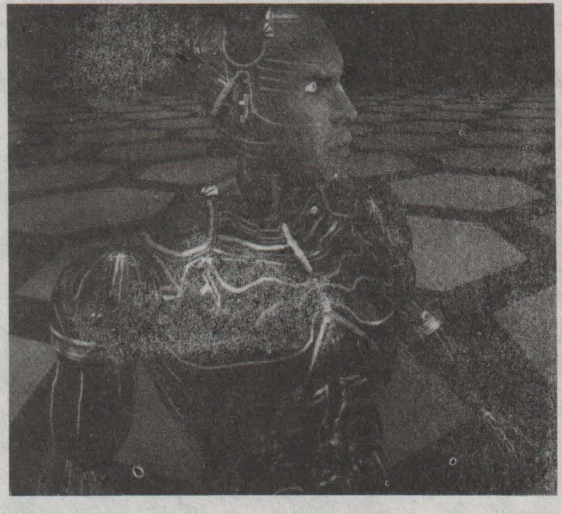
În cazurile de lipsă acută de inspiraţie, efectele speciale sunt folosite pentru a acoperi golurile unei intrigi slabe sau pentru a face interesante personajele oneste, dar asta nu constituie o noutate - adesea filme slabe au fost înecate în costume scumpe şi lux. (calculatoarele oferă şi alt tip de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

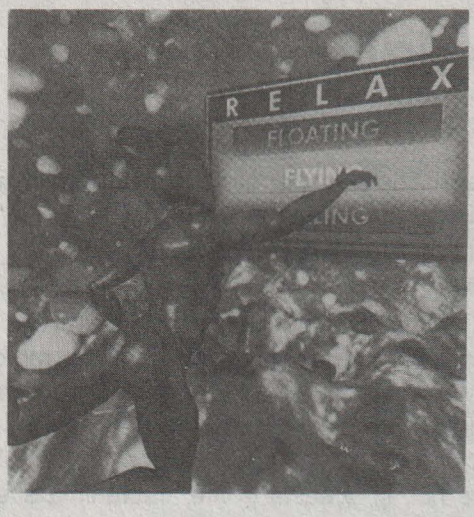
Într-un rol mai inspirat, efectele speciale le oferă regizorilor o nouă lume (de fapt, o infinitate de noi lumi) cu care să se joace; iar când regizorii văd ce se poate face îşi schimbă - uneori- ideile iniţiale pentru a putea folosi anumite efecte tentante. Mediul poate seduce mesajul.

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

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

Computerele au făcut munca mai uşoară. În 1983 Studiourile Disney au contactat compania de animaţie 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 restricţia decorului bidimensional.

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

De asemenea, calculatorul poate fi folosit şi pentru generarea a două mişcări diferite ale camerei pe aceeaşi imagine - prin apăsarea unui buton îl putem vedea pe Goofy ieşind pe uşă 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 ţăcănitul tastaturilor, pe măsură ce siliciul concurează celuloidul pentru locul cel mai important în producţia 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 şi forme - nu este simplă folosind logica strict numerică a unei maşini. 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: inflorscenţele irizate devin blocuri poligonale de culori ţipătoare, dealurile vălurite se reduc la linii zimţate.

Totuşi, 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 şi puterea de procesare ale hardului pentru grafică s-au dublat la fiecare 18 luni. Ca urmare, computerele dispun acum de forţă pentru a crea imagini complexe uluitoare şubredele 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 Schwarzenhegger: tonurile pielii; sclipirile părului său periat cu grijă; felul în care razele apusului se joacă delicat pe nobila sa frunte. Calculatoarele trebuie să proceseze cantităţi aproape inimaginabile de informaţie

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 viaţă.

Salvarea softului: pentru a întregi puterea în continuă creştere a hardului, programatorii au creat vaste biblioteci soft de texturi ce includ piatră, lemn şi ţesături, care oferă o copie fidelă a lumii reale. (Sunt nevoiţi să o facă dacă nu vor să scadă frumuseţea filmului). Ei au dezvoltat algoritmi eficienţi (numiţi *ray-tracing*) care calculează cum se comportă lumina pe suprafeţe, direcţia din care cade, unghiul, cât se absoarbe, cât se reflectă şi 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 mişcă într-o animaţie, acelaşi soft poate să calculeze din nou valorile de iluminare oferind imagini realiste fie ale lumii reale, fie ale unei fantezii debordante.

Mişcări iscusite

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

Producătorii de soft au integrat date anatomice exacte şi mişcări ale articulaţiilor în rutine de animaţie. Când un personaj generat de calculator ridică un deget, se calculează automat mişcarea pentru restul braţului şi pentru cap. Aceasta se numeşte animaţie ierarhică şi ea poate fi văzută în cazul robotului T1000 din *Terminator 2* (1991). Articulaţiile se mişcă corect-sub piele, şi au fost luate în considerare gravitaţia şi frecarea.

Personajele generate de calculator se pot mişca chiar şi de capul lor. Departamentul de animaţie de la Massachusetts Institute of Technology a creat un gândac: acţiunile sale se bazează pe mişcările roboţilor şi pe date despre gândacii reali; i s-a creat chiar şi "inteligenţă". El poate să "vadă" şi să evite obiectele, şi să încetinească în faţa obstacolelor.

Cea de-a treia direcţie de dezvoltare le propulsează pe primele două în lumea producţiei de film. Având în laborator secvenţele pe film şi având pe disc secvenţele 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 secvenţele de pe calculator, şi trecerea rezultatului pe film. Filme ca *Terminator 2* şi *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 secvenţe 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 faţa 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; animaţie - mişcare şi iluminare pe o anumită durată de timp; şi rendering - combinarea celor de mai sus şi transpunerea lor pe film sau pe banda video.

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

Apoi proiectantul stabileşte mişcările. Se aleg cadrele cheie, iar programul calculează cadrele intermediare. Pentru fiecare cadru redat, se calculează exact figura, obiectele, iluminările şi culorile. Acest proces poate dura câteva ore pentru un cadru; când cadrul este gata, el este înregistrat pe film. Autorii animaţiei au folosit peste 70 de fotografii reprezentând mişcările faciale şi vocale ale lui Fahey pentru a sincroniza sunetul.

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

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

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

De exemplu: spectatorii ţin în mână beţe din material plastic în două culori: roşii pe o parte şi verzi pe cealaltă parte; imaginile sunt înregistrate de camere video care transmit informaţiile 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, ţinut 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 şi a unei mănuşi (legate, evident, la un calculator) pot fi controlate mişcările lui Silver Suzy - o fată care practică surfing.

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

Eşantionarea vedetelor

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

O dată ce aveţi pe Madonna sau pe Arnie pe monitor, puteţi folosi softuri de animaţie pentru a-i pune să facă oriçe doriţi. Se pot genera secvenţe întregi, deşi aceasta ar cere foarte mult timp şi sute de mega octeţi de spaţiu pe disc în cazul unui PC standard. Ar mai fi necesar un sistem rapid de comprimare a datelor pentru a putea canaliza toată informaţia către ecran suficient de repede pentru animarea imaginilor. Va apare tentaţia de a folosi imaginile pentru a face filme "de casă", create pe noile calculatoare multimedia, salvate pe discuri hard sau optice şi puse înapoi pe un film.

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

actor. Apoi filmul ales va fi selectat din baza de date şi 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 aşa-numite scenarii "what-if" (ce se întâmplă dacă ...) - cum ar fi *Hamlet* cu happy-end.

Dar cel puţin vedetele vor fi din carne şi oase. Mai este cale lungă până când un personaj generat de calculator să ne păcălească aşa încât să credem că este o fiinţă umană sau un peisaj digital să pară natural. Teoretic, toate acestea ar tebui 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 producţiei pe PC

1. Introducere

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

Se constată o dezvoltare a activităţii de elaborare de programe de către diferite colective de specialişti renumiţi în societăţi comerciale care au ca activitate de bază producerea de software original sau adaptarea unor pachete străine la particularităţile legislaţiei româneşti.

Una din aceste firme este SOFT SERVICE S.R.L. din Braşov, 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 societăţilor comerciale şi - independent de aceasta - elaborarea unor programe la solicitarea diverşilor beneficiari de lucrări informatice.

În continuare va fi prezentată preocuparea de realizare a sistemului informatic integrat pentru conducerea proceselor economice din societăţi comerciale - cu precădere din domeniul industriei - şi realizările obţinute până în momentul de faţă.

În etapele deja parcurse, au fost cooptaţi specialişti cu experienţă practică în domeniul analizei de sistem şi a proiectării de sisteme informatice.

Ideile de bază în realizarea sistemului informativ sunt următoarele:

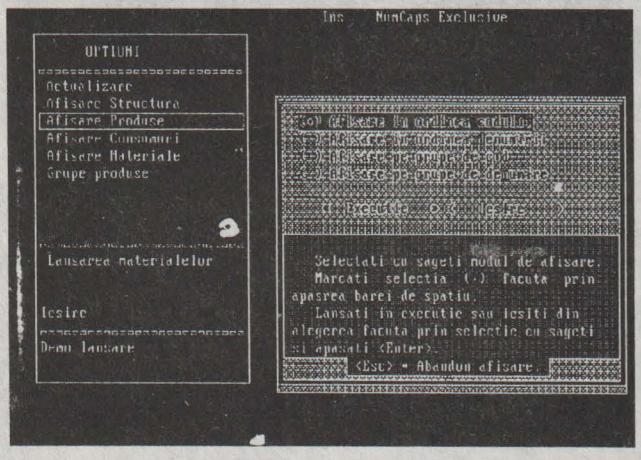
- constituirea unei baze de date unice, ca element integrator;
- realizarea în etape:
- abordarea cu precădere a problemelor din subsistebul producţiei;
- introducerea în sistem a unor funcţii care să asigure strategia şi tactica optimă în realizarea obiectivului.

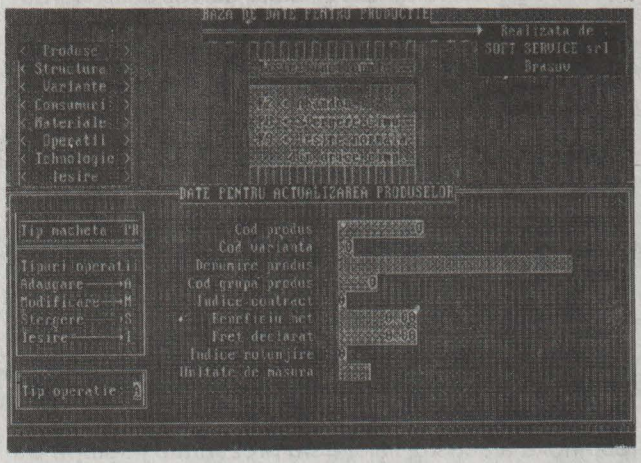
2.Prezentarea subsistemului informatic BD-LANS (baza de date - lansare în producţie).

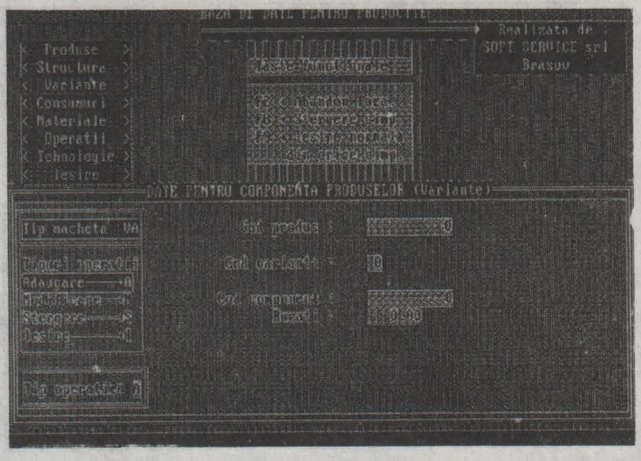
Pentru realizarea unui sistem informatic complex, care să rezolve mulitudinea de probleme tehnico-economice cu care se confruntă o societate comercială cu activitate de producţie în domeniul industriei, este necesară proiectarea bazei de date unice, ca element integrator al sistemului informatic.

La proiectarea bazei de date trebuie analizate relaţiile obiective (existente în realitate) dintre datele care vor fi prelucrate stabilindu-se repartizarea lor în fişiere precum şi legăturile dintre ele, în funcţie de aceste relaţii.

Complexitatea deosebită a sistemului informatic integrat impune cu necesitate proiectarea şi 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 conţine datele strict necesare activităţii de lansare în fabricaţie, implementat pe calculatoare compatibile IBM-PC.

Lansarea în fabricaţie este o subactivitate de bază a activităţii de programare a producţiei, ocupând un loc în aval de programarea propriuzisă şi în amonte de execuţia şi urmărirea îndeplinirii programelor de producţie. Putem defini lansarea în fabricaţie ca fiind ansamblul de activităţi legate de eleborarea documentaţiei economice, multiplicarea şi difuzarea acesteia şi declanşarea execuţiei sarcinilor repartizate pe fiecare loc de muncă din cadrul verigilor structurale ale societăţii comerciale.

Lansarea în fabricaţie - la momentul oportun - a tuturor produselor incluse în programul (optim) de fabricaţie este o obligaţie majoră, contribuind la buna desfăşurare a procesului producţiei. Ea reprezină liantul care leagă valorile potenţiale cuprinse în documentele economice de realizarea efectivă a producţiei finite.

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

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

Cu ajutorul calculatorului electronic se poate elabora documentaţia de lansare în fabricaţie într-un mod operativ şi eficient; mai mult, există posibilitatea reţinerii informaţiilor cuprinse în documentele de lansare elaborate în vederea urmăririi operative a abaterilor care apar - pe parcursul realizării programului de producţie - faţă de valorile din momentul lansării.

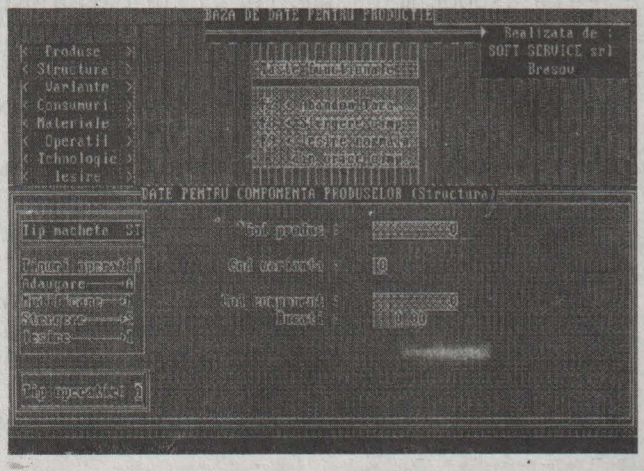
Aplicaţiile succesive care constituie lansarea materialelor şi manoperei sunt:

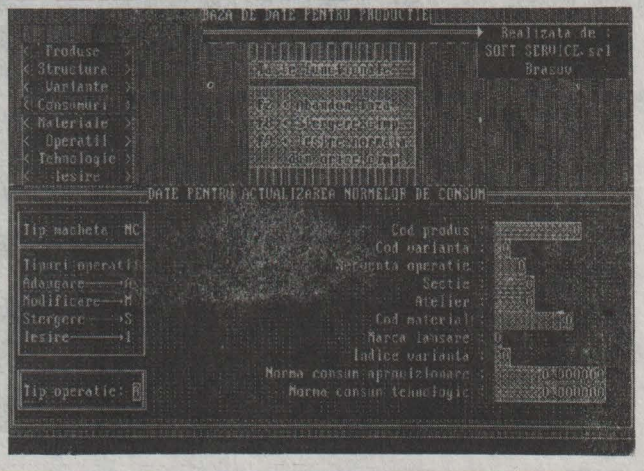
1. calculul necesarului de repere de fabricat.

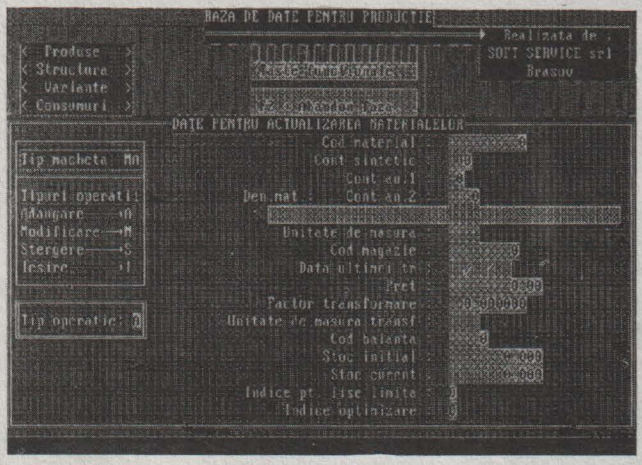
2. elaborarea borderoului, a bonurilor şi a fişelor limită de materiale.

3. elaborarea borderoului de manoperă şi a bonurilor de lucru (sau a altor documente specifice situaţiei concrete) şi vor fi descrise succint în continuoare.

1. Calculul necesarului de fabricat







Datele de intrare ale acestei aplicaţii - cu excepţia celor incluse în nucleul bazei de date - sunt conţinute în notele de lansare emise de serviciul *Programarea, lansarea şi urmărirea producţiei*: codul produsului (eventual şi al variantei de produs), numărul comenzii şi cantitatea de produs lansată în fabricaţie, pentru fiecare din produsele care constituie programul de fabricaţie.

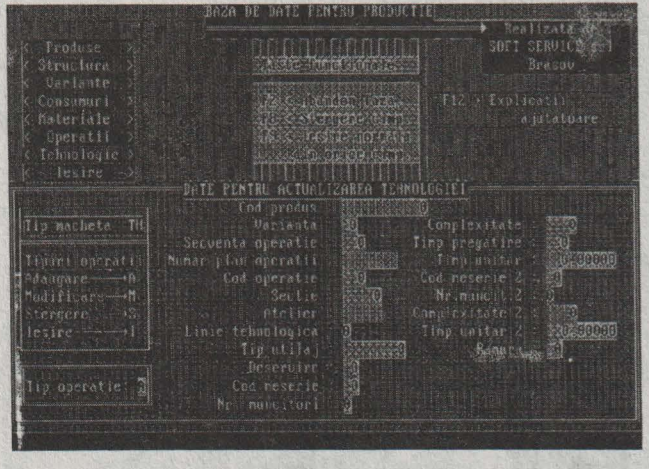
Cu ajutorul structurii pe nivele a produselor, memorată în fişierul de structură din nucleul bazei de date, se obţine necesarul de repere de fabricat printr-o procedură de descompunere a produselor în repere (explozie) şi o cumulare pe comenzi a cantităţilor de repere identice.

2. Elaborarea borderoului, a bonirilor şi a fişelor limită de materiale .

Pe baza necesarului de fabricat obţinut în etapa anterioară şi a datelor înregistrate în fişierele de consumuri şi materiale, se determină cantitatea cumulată de material necesară în două variante:

-la nivelul întregii comenzi,

-la nivelul reperului în cadrul comenzii.



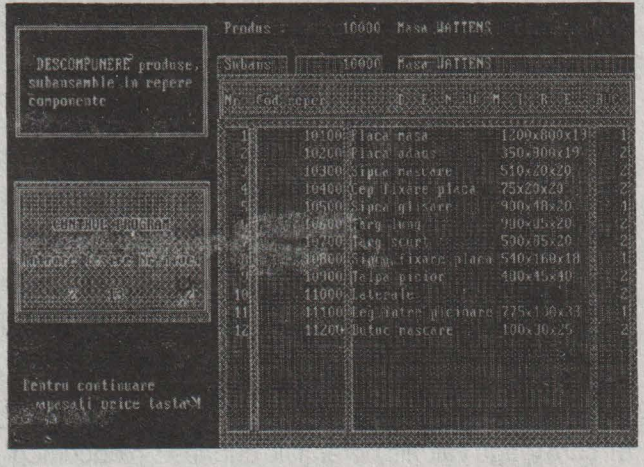
În funcţie de varianta solicitată se vor edita borderoul de materiale (ca situaţie recapitulativă a operaţiei de lansare) şi bonurile sau fişele limită de materiale. Conţinutul acestor documente poate fi analizat pe baza exemplului prezentat.

3. Elaborarea borderoului manoperă şi a bonuri - lor 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.

Informaţiile care conduc la elaborarea documentelor specifice lansării materialelor trebuie cuprinse - în principal - în fişierul de tehnologie. O structură mai răspîndită a acestui fişier este prezentată în etapa de afişare a ecranelor de prelucrare a datelor. Se propune aici o noţiune - *complexitatea operaţiei* - care poate înlocui vechile elemente de calcul pentru manoperă (reţea tarifară, categorie, treaptă).

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

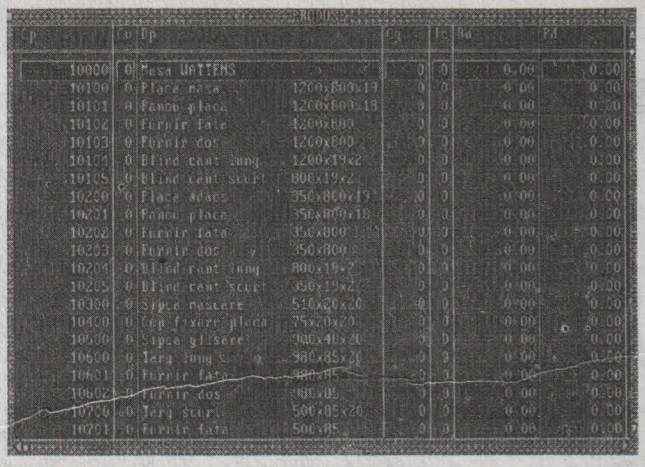


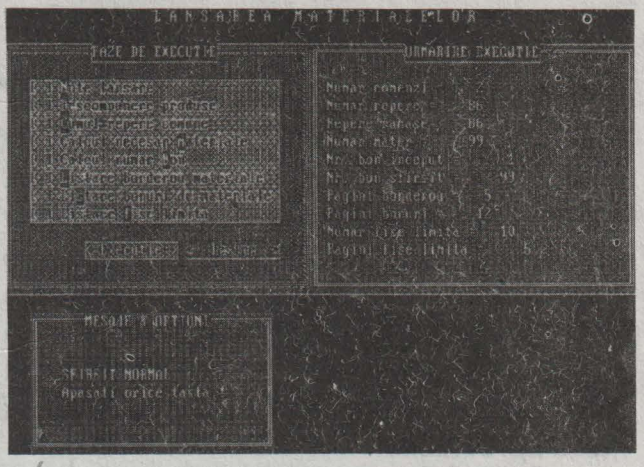
3. Pachetul de programe BD-LANS

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

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

Programele sunt scrise în FOXPRO 2.0 şi au performanţe deosebite în ceea ce priveşte asigurarea corectitudinii datelor introduse (încă din faza de încărcare a informaţiilor în fişierul de structură se evidenţiază neconcordanţele logice - evitarea buclărilor) şi în ceea ce priveşte viteza. Timpul de răspuns redus este asigurat în special de programul de descompunere pe





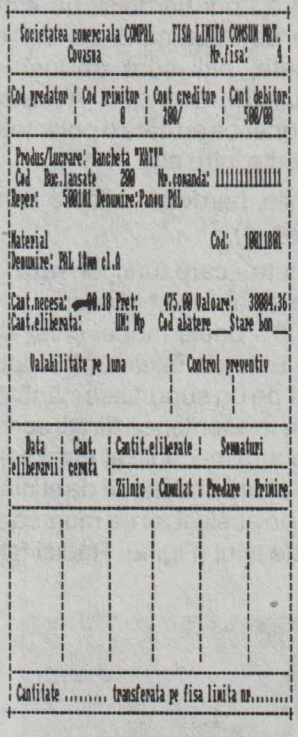
nivele (maximum 10 nivele), care constituie piesa de rezistenţă a aplicaţiei.

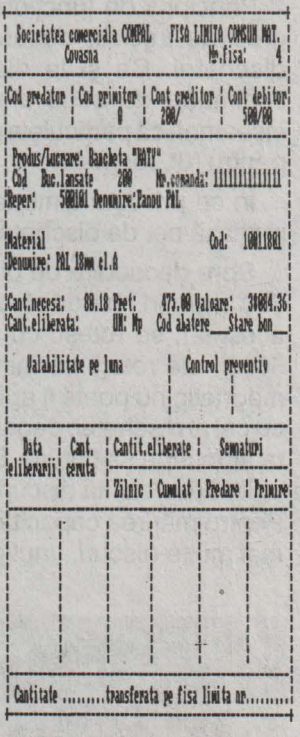
Numărul de nivele poate fi mărit, înregistrându-se o scădere corespunzătoare a vitezei programului de descompunere. În cazurile practice întâlnite până în prezent, numărul de 10 nivele a fost acoperitor chiar pentru produse de complexitate aparent ridicată. De exemplu motoreta *Hoinar* are reperele dispuse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

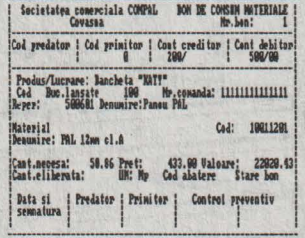
Experimentarea funcţionării subsistemului a fost realizată la Societatea Comercială *COMPAL* din Covasna care are ca principală activitate producţia de mobilă. Există aici o preocupare deosebită pentru introducerea metodelor moderne de conducere şi organizare - inclusiv utilizarea unei reţele de calculatoare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

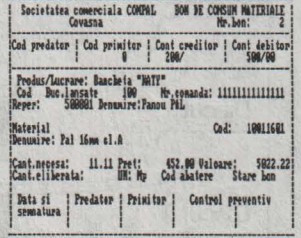
Exemplificările din acest articol au fost făcute cu date culese la firma menţionată.

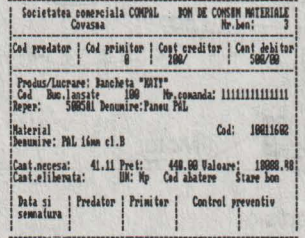
Pe baza informaţiilor încărcate în nucleul bazei de date au fost realizate şi alte aplicaţii pentru maximizarea profitului în funcţie de resursele existente la un moment dat şi pentru eleborarea strategiei optime de aprovizionare cu materii prime şi materiale.

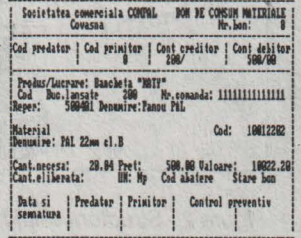












Cei interesaţi pot contacta colectivul care a elaborat pachetul de programe la Societatea Comercială SOFT SERVICE S.R.L..str. Zizinului nr. 87, sc.A, ap. 3, Braşov.

Fiz. ec. Constantin Dumitriu.

O privire asupra harddiscurilor

Toată lumea îl foloseşte dar nimeni nu-l poate defini, sau mai bine zis nu-i cunoaţe "viaţa interioară". Importanţa lui iese în evidenţă abia atunci când apare o defecţiune. Este vorba de harddisk, acea memorie magnetică rotativă cu capacitate şi viteză apreciabile.

Principiul de funcţionare a unui harddisk nu este greu de înţeles, deoarece se aseamănă cu cel al dischetei. Ca şi la dischete, biţii sunt traduşi 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 priveşte dimensiunile, harddisk-urile se diferenţiază net de dischete. (Fig. 1)

Spre deosebire de dischete - care funcţionează la 300 rot/min, harddisk-urile (mai exact discurile din acestea), se rotesc cu 3600 - unele model chiar cu 5000 - de rotaţii pe minut. Este de înţeles că stratul magnetic nu poate fi aplicat pe un suport asemănător cu cel al dischetei, deoarece la asemenea turaţii acesta ar începe să fluture. Suportul mecanic al harddisk-urilor îl reprezintă discuri de aluminiu (uneori de sticlă). Pentru mărirea capacităţii, pe acelaşi ax se montează mai multe discuri, unul peste altul (Fig. 2). Radial faţă de ax, asemămător pick-up-ului, pe un braţ mobil, sunt dispuse capetele de înregistrare-redare - şi anume câte unul pentru fiecare faţă de disc.

O atingere (contact nemijlocit) între cap şi disc - foarte temutul *Headcrash* - ar fi mortală la această turaţie, rezultând distrugerea capului şi pierderea datelor. Acest accident poate fi evitat prin efect Bernoulli. La această turaţie, între disc şi cap se formează o pernă de aer, astfel încât capul "planează" deasupra discului la o distanţă de un micrometru. Atfel se explică sensibilitatea harddisk-urilor la orice tip de şocuri. Praful, amprentele şi firele de păr căzute între cap şi disc au proporţii 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 reţea (funcţia *autopark*), sau iniţiat manual, prin lansarea unui program (*ship* sau *park*) - capetele de citire-scriere sunt deplasate într-o zonă care nu conţine date. Acolo capetele pot ateriza pe disc (*crashing zone = zonă de aterizare*), fără pericol de deteriorare. Suplimentar, capetele sunt fixate rigid în poziţia de parcare şi - în cazul în care sunt manipulate corespunzător - este evitată deteriorarea lor. Interesantă este funcţia *autopark* a unităţilor de disc moderne. Deconectat fiind de la reţea, discul se mai roteşte o vreme, din inerţie. Un generator trans-

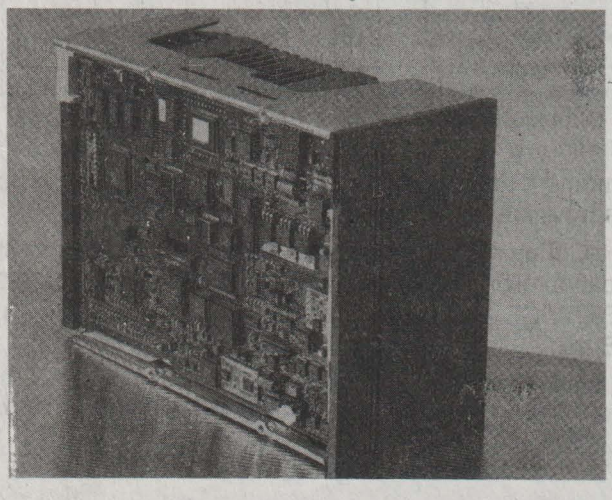


Fig. 1 Micropolis 1548 de format 5,25", are o capacitate de 1,7GBytes

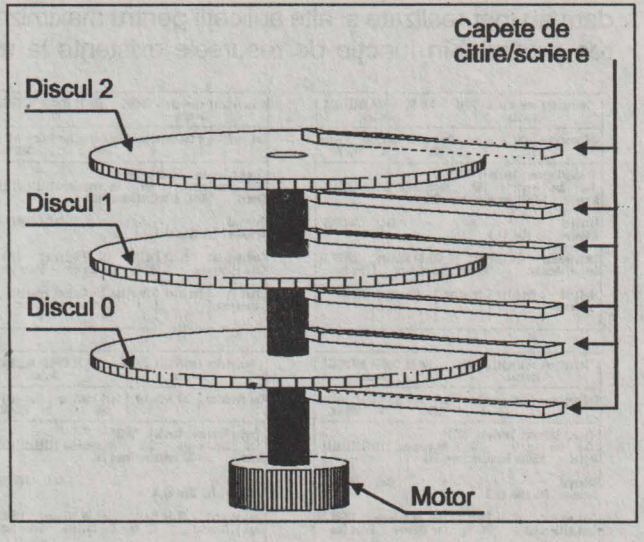


Figura 2 - Structura unui harddisk

Despre BPI şi FPI

Aşa-numita densitate de înregistare 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", şi indică de câte ori se poate modifica orientarea particulelor magnetice pe distanţa de un inch. Din motive obiective, distanţa dintre două schimbări de flux nu poate fi sub o anumită limită. Acest lucru este determinat de faptul că altfel interacţiunea dintre doi biţi alăturaţi ar fi prea intensă şi ar avea loc schimbări de polaritate, deci pierderi de date. Însă, cu cât valorile BPI respectiv FPI sunt mai mari, cu atât încap mai multe date pe harddisk.

formă această energie de rotaţie într-un curent electric, care este folosit apoi pentru parcarea capetelor în zona nepericuloasă. Spre deosebire de unităţile 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 şi primele unităţi, care aveau un timp de acces de 80 ms, au fost un progres enorm. Timpul mediu de acces, la cele mai evoluate modele la ora actuală, este sub 10 ms. Unităţile de dischetă au timpii de acces între 150 şi 200 ms. Deoarece stratul magnetic şi capetele sunt de calitate superioară, harddsik-urile au şi 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 şi în ce priveşte rata de transfer a datelor. Pe când discheta asigură 30-40 Kbyte pe secundă, chiar şi un harddisk lent atinge 200 KByte pe secundă. Cele medii ating 600-700K Byte 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 condiţiile lipsei memoriei cache).

Organizarea datelor

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

ZBR creează simţitor mai mult spaţiu

La scurt timp, s-a elaborat un procedeu de înlăturare a risipei de spaţiu 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 procedeul este un compromis între câştigul de viteză şi cel de spaţiu. În figura 3 apare un exemplu de aplicare a acestui procedeu.

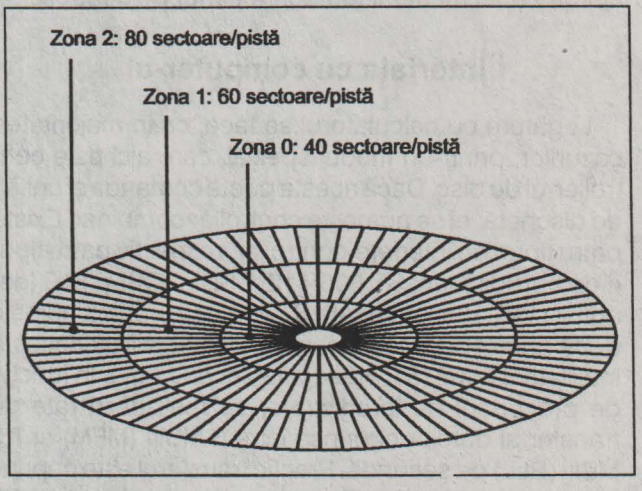


Figura 3

La Zone-Bit-Recording, pistele centrale au mai puţine sectoare decât cele periferice

În zona zero pot fi memorate cele mai puţine 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 înălţimea de 25,4 mm şi un singur disc de 3,5" - ajunge la o capacitate impresionantă de 120 MByte. Modelul LPS 450 (2290 TPI, 50000 BPI, 37900 FCI) al aceluiaş producător - având aceleaşi dimensiuni, echipat însă cu 5 discuri, are o capacitate - după formatare - de 425 Mbyte. Ar putea deci fi unul dintre favoriţii acestui domeniu.

Factorul Interleave

Deoarce 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 şi evaluat, sectorul 2 a trecut deja de capul de citire şi a putut fi citit doar la următoarea rotaţie. 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 interstiţiu. În acest interstiţiu se memorează alt sector al pistei. Acest procedeu se numeşte *Interleave* (intercalare). *Factorul Interleave* indică numărul de sectoare fizice intercalate între două sectoare logice, şi numărul de rotaţii necesare citirii unei piste. La un Interleave de 1:3, sectorul doi este la trei sectoare după sectorul 1 şi pentru citirea pistei sunt necesare 3 rotaţii. Ideal ar fi un factor de Interleave de 1, la care sectoarele logice se succed nemijlocit şi pista poate fi citită la o singură rotaţie. Combinaţii moderne controller-harddisk ajung la această performanţă fără probleme, de exemplu cu ajutorul unei memorii-tampon (cache) pe controller.

Interfaţa 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 şi unităţi de dischetă, el se numeşte controller combinat. Există patru tipuri de interfeţe-controller, respectiv patru tipuri de harddisk-uri: ST 506/412, ESDI, SCSI,şi IDE (denumit şi AT-Bus). Interfaţa *ST 506/412* a primit numele după primul harddisk Seagate, care a utilizat această interfaţă ST 506 (*ST = Seagate Technology*). În funcţie de procedeul de înregistrare, se realizează rate de transfer al datelor cuprinse între 5 MBiţi (MFM) şi 7,5 MBiţi (RLL) pe secundă. Practic transferul este cuprins între 200 şi 600 de KByte pe secundă. Interfaţa ST 506/412 poate fi recunoscută după cabluri: cablul de transmisie a datelor cu 20 de fire şi cablul de comandă cu 34 de fire, acesta din urmă fiind realizat din cablu-panglică. Interfaţa 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 aceleaşi tipuri de cabluri şi, de asemenea, în mod serial. Rata de transfer se situează între 10 şi 15 MBiţi, sensibil mai mare decât la ST. Atât ST 506/412 cât şi ESDI sunt evoluţii specifice PC-urilor şi suportă maximum două harddisk-uri simultan. Interfaţa universală la această oră este *Small Computer System Interface*, care a fost standardizată în 1982. SCSI - sau mai evoluata SCSI-II - nu sunt numai interfeţe pure pentru harddisk-uri. La ele se pot racorda scanner-e, unităţi CD-ROM şi multe altele.

Până la 7 unităţi (adică 7 harddsik-uri) pot fi conectate într-o ordine arbitară, dar ultima trebuie prevăzută cu o rezistenţă de capăt. Deoarece fiecare unitate are adresa ei formată dintr-un număr, datele ajung la destinaţie fiind transferate de la o unitate la alta, până la unitatea prevăzută. Alt avantaj este faptul că interfaţa SCSI este larg răspândită - se găseşte şi la calculatorul Next al lui Steve Jobs şi la Apple MacIntosh. Inteligenţa acestei interfeţe paralele nu o posedă controller-ul fiecărei unităţi în parte. Spre deosebire de ST 506/412, nu controller-ul comandă procesele - ca de exemplu formatarea, ci calculatorul emite doar instrucţiunea "format harddisk", restul fiind preluat de controller-ul unităţii respective, în timp ce calculatorul îşi 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 şi bus-ul ISA/EISA/MCA din sistem.

Discuri umede

Dacă distanţa de 1 micrometru,dintre capul de citire-scriere şi disc pare minusculă, tehnicienilor de harddsik-uri li se pare încă foarte mare deoarece, cu cât distanţa devine mai mică, cu atât creşte densitatea de înregistrare. Dacă distanţa scade la jumătate, densitatea creşte 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 subţire decât perna de aer - ceea ce duce la o creştere apreciabilă a capacităţii, iar pe de altă parte pelicula de ulei se manifestă şi ca un tampon, între cap şi disc - ceea ce înlătură pericolul impactului (*headcrash*). Deocamdată sistemul funcţionează 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 lubrifianţi speciali, care a fost pur şi simplu cumpărată de Conner. Încă nu se ştie când vor apărea asemenea discuri pe piaţă.

SCSI-interfaţa 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 şi controller şi *host-adapter*-ul nu mai are prea multă influenţă asupra ratei de transfer. Dezavantajul este preţul ridicat al componentelor SCSI, deoarece fiecare unitate îşi "cară" şi propria interfaţă. 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-dapter*-ul SCSI cel mai răspândit este un produs al firmei americane ADAPTEC; el se numeşte 1542B (controller combinat) şi costă în jur de 500 DM. Produsele SCSI sunt în prezent rezervate profesioniştilor datorită preţului ridi-

cat, date fiind şi performanţele. 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 rezistenţele terminale care au mai fost amintite. Chiar şi o modificare a succesiunii unităţilor poate crea probleme, contrar specificaţiilor SCSI. Nu se recomandă instalarea SCSI-ului de către începători, fiind necesară o oarecare experienţă, dacă nu chiar profesionalism. Dar aceasta rezultă automat şi din preţul ridicat.

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

Avantajul faţă de SCSI este preţul redus, la performanţe comparabile (posibil 2MByte/sec, media aproximativ 1 MByte/sec). Dar nu numai discurile sunt mai avantajoase; un *host-adapter* IDE cu controller pentru dischetă costă numai 30 DM. Nici modelele cu interfeţe seriale şi paralele integrate nu sunt mult mai scumpe.

În încheiere, se poate spune că viitorul este al interfeţelor SCSI şi IDE. Harddisk-urile ST 506/412 vor dispare de pe piaţă, iar ESDI-urile vor avea dificultăţi - din cauza preţului ridicat - în a se impune în faţa standardelor IDE şi SCSI.

Procedee de înregistrare

Cum sunt traduşi pe suport magnetic biţii şi octeţii?

Prin diferite moduri de codificare, informaţiile 0 şi 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 funcţie de sensul curentului, care magnetizează corespunzător suprafaţa discului. Deoarece nici harddisk-urile nu au turaţia perfect constantă, trebuie memorate - în afara datelor - şi aşa-numitele informaţii de tact. Aceasta asigură scrierea şi citirea la aceeaşi viteză, compensând eventualele abateri datorate variaţiilor turaţiei (imaginaţi-vă haosul care s-ar produce la lungimi diferite ale fiecărui sector în parte). Dacă informaţiile de tact sunt înregistrate, în combinaţie cu informaţiile memorate, pe fiecare suprafaţă de disc, se vorbeşte despre *Embedded Sector Server*, adică informaţiile de sincronizare sunt cuprinse în sectoarele de date. Se poate folosi însă o faţă a pachetului de discuri inclusiv pentru informaţiile de tact, pe când celelalte se utilizează exclusiv pentru date. Prima metodă este mai elegantă şi este preferabilă. Dacă s-ar succeda alternativ biţii 0 şi 1, s-ar putea renunţa la tactul separat, utilizând de exemplu un 1 pentru sincronizare. Problematice sunt însă şirurile de biţi identici (de exemplu 000 şi 111111). Aceste aşa-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 biţii de date se schimbă, de la 0 la 1 sau invers (Fig. 4)

Dacă se succed mai mulţi biţi identici, nivelul curentului de înregistare nu se modifică. La o serie de 1, curentul de înregistrare rămâne constant la nivel high (1), fără a reveni - între biţi - la low (0); de aici vine denumirea procedeului. Nivelul curentului de înregistrare corespunde exact nivelului semnalului digital, deci şi biţilor de memorat. Din păcate procedeul, deşi

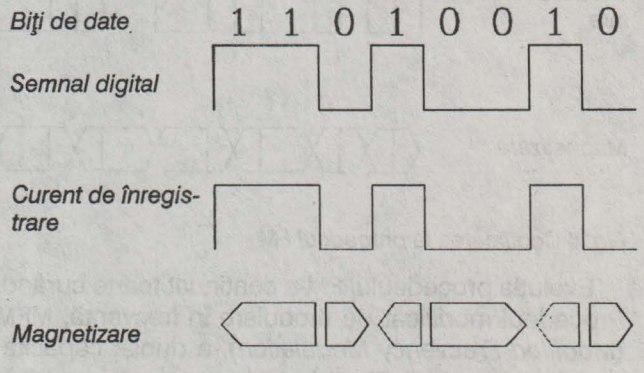


Figura 4 - Exemplu de codificare pentru procedeul NRZ

este simplu, are un dezavantaj însemnat: deoarece pot interveni şiruri de 0 sau 1 arbitrar de lungi, pretenţiile referitoare la sincronizare sunt foarte mari şi şirul de biţi nu conţine informaţii de tact (*Embedded Sector Server* imposibil); semnalele de tact trebuie depuse pe o suprafaţă separată a pachetului de discuri, blocând înregistrarea datelor pe aceasta.

Procedeul de modulare în frecvenţă FM (*Frecvency Modulation*) a integrat pentru prima dată informaţiile

de tact. Biţii de date sunt tranformaţi în şiruri de biţi FM, deoarece trebuie adăugat tactul. Un 1 se transformă în 11, iar un 0 în 10. Prima cifră serveşte drept informaţie de tact, a doua reprezentând bitul de date. Alura curentului de înregistrare din figura 5, poate fi descrisă astfel: dacă o grupă de biţi (o grupă de biţi cuprinde un bit de date şi eventual informaţii de tact) conţine şirul de biţi FM "10", aceasta înseamnă "păstrarea curentului"; dacă conţine şirul "11" înseamnă "modificarea curentului", iar curentul se modifică în interiorul grupei. Prima cifră indică nivelul curentului la începutul grupei, a doua nivelul la sfârşitul grupei (pe linia întreruptă). După cum observaţi în figura 5, rezultă multe domenii magnetice scurte.

Deoarece distanţa ("d"), dintre două schimbări de flux, nu poate fi arbitar de mică, densitatea de înregistrare este limitată. În plus, procedeul consumă mult spaţiu, deoarece fiecare grupă conţine informaţii de tact. De aceea, codificarea FM se numeşte *Single Density Format*, adică format de înregistare simplă densitate.

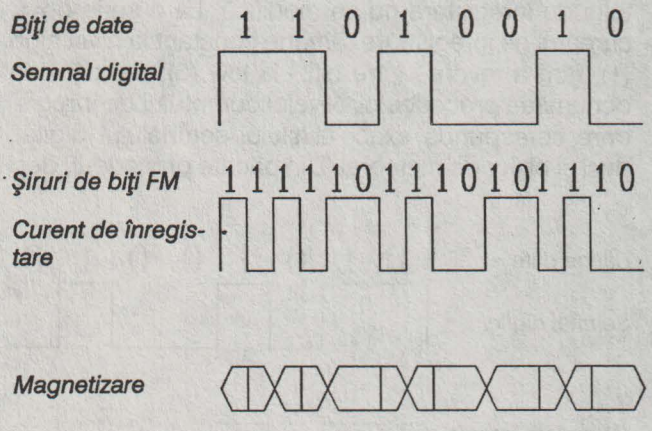


Fig. 5 Codificarea la procedeul FM

Evoluţia procedeului FM a continuat foarte curând. Procedeul modificat de modulare în frecvenţă, MFM (*Modified Frecvency Modulation*), a dublat capacitatea de memorare. Tactul se memorează numai în grupele care ele însele, sau cele precedente, nu conţin cifra 1. Un bit de date conţinând cifra 1 se transformă în şirul 01 la MFM, 0 tranformându-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 menţine stabil tactul de citire. Înaintea fiecărei cifre de 1 din şirul MFM se produce o modificare a curentului de înregistrare, zonele diferite de magnetizare fiind mai puţine şi mai lungi (Fig. 6). Aici procedeul FM ar necesita 12 schimbări de flux. Pentru codificarea şirului de biţi din exemplu procedeul MFM necesită doar 6, ceea ce confirmă dublarea de capacitate amintită. În special harddisk-urile ST 506/412 utillizează această metodă, la care pe fiecare pistă sunt cuprinse 17 sectoare. Deoarece viteza acestui procedeu este redusă, producătorii au eloborat un alt procedeu.

Procedeul RLL urmăreşte limitarea şirurilor de zerouri (*RLL = Run-Length-Limited*). La procedeul *RLL-(2,7)*, între doi biţi "1" se află minimum doi şi maximum şapte biţi "0". Nou la acest procedeu este faptul că la codificare se ţine seama de biţii care urmează celui de codificat - adică se ţine seama de context. Tabelul următor arată cum se transformă biţii de date în şirul

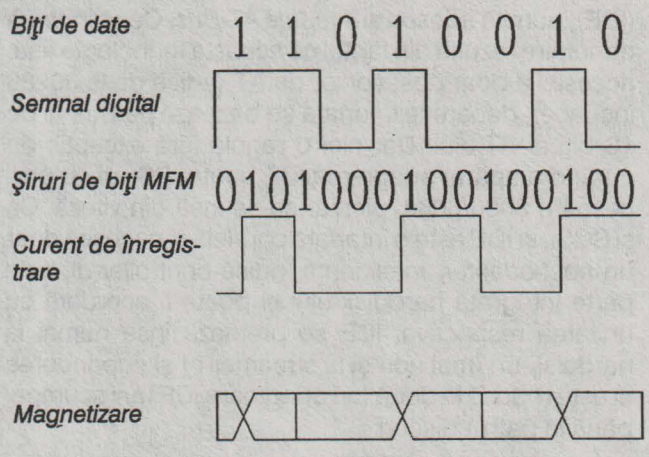


Fig. 6 Codificarea MFM reprezintă o metodă mai evoluată

RLL.

Şirul de biţi din figura 7 se transformă în trei paşi: 11 devine în cod RLL 010, următoarele două 010 devin fiecare 001000. Biţii de date 11010010 s-au transformat în şirul RLL în: 1100>001000>001000, din cei 8 biţi iniţiali au rezultat 16. Iese în evidenţă numărul mare de biţi aflaţi pe nivel "0". Curentul de înregistrare se modifică ca şi la MFM, înaintea fiecărui bit "1" din şirul RLL. Rezultă patru grupe de magnetizare. La prima vedere, pare paradoxală afirmaţia că RLL economiseşte spaţiu pentru că s-a mărit doar numărul de biţi

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 şi uriaşi

Probabil nici o altă unitate periferică nu a avut un salt calitativ asemănător harddisk-ului. De fiecare dată când experţii au prevăzut sfârşitul evoluţiei acestora, au fost contrazişi, căci în continuare harddisk-urile devin mai mici (dimensional) şi mai rapide, simultan crescând şi capacitatea. Astfel ouceresc domenii rezervate până nu de mult, din motive de gabarit, memoriilor cu semiconductoare - acestea din urmă fiind şi mult mai scumpe. Cel mai recent exemplu este harddisk-ul de 1,8" al firmei Conner, având formatul PCMCIA şi mărimea unei cărţi de credit. Având capacitatea de 32 MByte, acesta poate înlocui tradiţionalele module PCMCIA-Flash-RAM, preţul pe bit fiind inferior celui al concurentului său bazat pe siliciu. Astfel harddisk-urilor li se deschide lumea Palmtop-urilor şi 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 - şi cu o greutate de doar 32 de grame, acest pitic are totuşi o capacitate de 20 MBytes. Nu este de mirare că un ceasornicar, şi nu Hewlett-Packard, produce această miniatură.

Caracteristicile tehnice sunt absolut comparabile cu cele ale "fraţilor mai mari". Discul se roteşte cu turaţia 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 priveşte rezistenţa mecanică, Kittyhawk-ul depăşeşte cu mult discurile mai mari. Hewlett-Packard vede utilizarea acestor discuri nu numai la Palmtop-uri şi Notebooks, ci şi la imprimante, fax-uri şi copiatoare, folosindu-l drept memorie tampon.

Tehnicienii de la HP se gândesc şi la alte aplicaţii. În locul unităţii harddisk de 5,25", vor să echipeze mai multe unităţi de 1,3" (în spaţiul rezervat celei de 5,25"s-ar putea îngloba 60 de bucăţi de Kittyhawk), drept

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

Iniţial XT-ul lui IBM a fost dotat cu un harddiskde 5,25" care avea o capacitate de 10 MBytes. Discurile moderne au, la acelaşi gabarit, o capacitate de 3 GBytes, adică de 300 de ori mai mult; chiar şi unităţile de 3,5" au depăşit de mult limita de 1 GByte. Discurile de 8", cândva foarte răspândite, sunt în retragere, fiind înlocuite de *Disk-Array*-uri.

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

Acum urmează artificiul propriu-zis: deoarece în cod RLL-(2,7), unui bit "1" îi urmează cel puţin doi biţi "0", se poate spori densitatea de înregistrare înregistrându-se şirul "100" pe cea mai mică porţiune posibilă. Referitor la biţii de date, 1,5 biţi pot fi înregistraţi pe porţiunea minimă de 1 bit, ca la celelate procedee. Astfel se explică economia de spaţiu crescută cu 50 % a codului RLL-(2,7) faţă de MFM. Procedeul este utilizat la discuri mai recente, de tip ST 506/412 şi la modelele noi IDE. Să nu vă miraţi dacă citiţi 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 uşoară 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 "uşoară". În cazul că mai multe repetări eşuează, este vorba despre *Hard-Error*. Atunci probabil că o porţiune a stratului magnetic este deteriorată şi datele, din păcate, s-au pierdut. De fapt harddisc-urile au - din fabricaţie - asemenea defecţiuni, care sunt însă imprimate pe partea superioară a carcasei. Aceste porţiuni defecte trebuie specificate la formatarea low-level, pentru a se evita de la bun început utilizarea lor. De regulă sunt între 0 şi 20 de sectoare şi nu există motive de panică. Harddisk-urile IDE (care se formatează exclusiv cu instrucţiunea 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 şi se compune din timpul de deplasare şi timpul de latenţă. Timpul de deplasare indică perioada necesară deplasării capului de citire/scriere până în dreptul

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

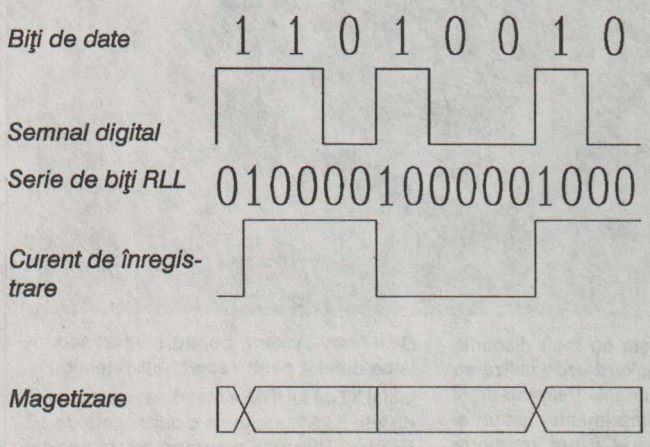


Fig. 7 Codificarea RLL -(2,7) este actualmente cea mai răspândită

acesta se poate realiza printr-o deplasare mai rapidă a capetelor, deplasare care - în dreptul pistei căutate - trebuie amortizată mai intens şi necesită un timp mai mare pentru a se "linişti". Se scurtează prin aceasta foarte mult *Setting-Time*-ul, astfel ca faza de amortizare să nu consume întregul câştig de timp. Timpul de latenţă poate fi scurtat prin mărirea turaţiei discului deoarece cu cât turaţia este mai mare, cu atât sectorul căutat va fi mai repede în dreptul capului. O consecinţă pozitivă a măririi turaţiei este şi 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 braţ cu capete de citire/scriere (citire diametral opusă celuilalt cap). La acest procedeu - dezvoltat de firma Conner - se depalsează acel braţ care este mai apropiat de pista căutată. În continuare, cele două braţe 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 mulţime de date într-o memorie intermediară.

În comparaţie cu procesorul, circuitele RAM sunt lente, dar în comparaţie cu harddisk-urile ele sunt ultrarapide. De aceea este avantajos ca datele utilizate frecvent să fie citite de pe disc şi 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 părţi 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 reuşitelor depăşeşte numărul ratărilor datorate unor algoritmi sofisticaţi, memoriile cache aduc o sporire a vitezei şi în cazul accesului şi în ce al ratei de transfer. Harddisk-urile moderne au integrate, de obicei, memorii cache de 32 până la 512 KByte, împărţite în mai multe segmente, şi toate accesele de citire (mai nou şi cele de scriere) sunt trecute prin această memorie-tampon. Cea mai scumpă soluţie în domeniul harddisk-cache-urilor a fost şi va fi *cache-controller*-ul (mai corect spus host-adapter-ul, la IDE şi SCSI). Se vorbeşte că RAM-Cache-urile multor modele 386 sau 486 vor fi realizate hardware, complet transparent pentru soft, adică insesizabil pentru soft.

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

La acestea se poate folosi un Cache de 512 KBytes pînă 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 consistenţa datelor. Cu cât cache-ul este mai mare, cu atât paguba este mai mare în cazul căderilor de tensiune a reţelei.

Mai avantajoase - şi de multe ori şi mai rapide - sunt programele cache. Soluţiile 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 totuşi mai mare decât la controller-e cu procesor propriu. Motivul este foarte simplu: se pierde timp cu procedeul 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 frecvenţa 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 preţul lor redus. Aici sunt suficiente un spaţiu rezervat în RAM şi unul din programe. Preţurile componentelor RAM fiind în scădere, multe programe cache pot fi procurate pe piaţa Shareware (de exemplu *Hyperdisk*).

În afară de acestea, începând cu varianta 5.0 a lui MS-DOS, se livrează şi cache *smartdrive*-uri care conlucrează optim şi cu Windows. În ansamblu, aceasta este soluţia rapidă şi accesibilă ca preţ 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 privinţă, merită să aruncaţi o privire în culise. Cel puţin atunci când - din greşală - ştergeţi un fişier sau când apare o eroare de citire, informaţiile suplimentare pot fi de mare folos.

Un harddisk tipic este compus dintr-un pachet de discuri, pe care sunt înregistrate magnetic informaţiile în fişiere. Discurile sunt acoperite cu un film subţire, conţinând particule metalice sau de oxid. Intrând sub acţiunea unui câmp magnetic, acestea se orientează după una sau două direcţii. Suprafaţa fiecărui disc este împărţită în inele concentrice (cilindri). La rândul lor, cilindrii sunt divizaţi în zone de memorare numite sectoare. La majoritatea PC-rilor, un sector ocupă 512 Bytes.

Când se vorbeşte despre tehnologia harddisk-urilor, cel mai frecvent cuvânt este "capete". Numărul capetelor de citire/scriere este egal cu cel al feţelor pe care se face înregistrarea datelor. La majoritatea harddisk-urilor numărul de capete este dublul celui al discurilor. Aşa-numiţii 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 octeţi 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 iniţializa un harddisk, în Sistemul MS-DOS, trebuie să fie îndeplinite trei condiţii. În primul rând discul trebuie formatat *low-level*. Acest mod de formatare stabileşte poziţia sectoarelor şi cilindrilor pe disc. În al doilea rând, harddisk-ul trebuie partiţionat, adică împărţit în mai multe unităţi logice, care în MS-DOS sunt adresate separat, fiind şi tratate separat ca unităţi 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. Informaţiile despre sectoare (ID) sunt înregistrate în sectorul *ID-Header*. Acesta conţine, de obicei, numărul capului, numărul sectorului, numărul de cilindri (piste) şi adresa sectoru lui cu informaţiile despre acestea. În afară de acestea, aici mai apar teste pentru depistarea erorilor la nivel de Header. Controller-ul harddisk-ului foloseşte aceste informaţii pentru căutarea într-un anume sector.

Offset-ul Înregistrărilor din tabelul de partiţii

	00hex	01hex	02hex	03hex	04hex	05hex	06hex	07hex	08-0Bh ex	0C-0Fh ex
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 şi cilindru

Ultimul cap, sector şi cilindru

Mărimea partiţiei în sectoare

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

Fig. 1. Formatul tabelului principal de partiţii

Numerotarea se va face funcţie de diferiţii factori care vor influenţa ulterior, modul de utilizare şi timpul de acces al harddisk-ului. Unul din aceşti factori este cel de *Interleave*. Acesta adaptează turaţia discului la viteza cu care controller-ul poate trasfera date înspre sau dinspre harddisk. Procedeul interleave separă sectoare ale discului numerotate crescător, astfel încât sectoarele 2 şi 3, de exemplu, se pot afla la o distanţă de 4 sectoare între ele. Sectoarele trebuie dispuse astfel deoarece nu toate computerele reuşesc, la viteza de rotire a discului, să citească un sector, să înregistreze datele în memorie, fiind şi pregătite pentru citirea următorului sector. Dacă pentru aceasta controller-ul trebuie să aştepte o rotaţie completă a discului, se pierde timp. Dacă sectoarele sunt depărtate, se poate stabili distanţa 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 şi sectoarele deteriorate, care nu vor mai putea fi folosite pentru memorarea datelor.

MS-DOS permite împărţirea harddisk-ului în partiţii. Acestea sunt tratate ca harddisk-uri separate şi sunt adresate separat. Există trei tipuri de partiţii: partiţia primară DOS, partiţii Extended-DOS şi partiţii non-DOS. O partiţie primară DOS este prima partiţie DOS aplicată pe harddisk, şi este singura din care poate fi lansat DOS. Ea este asimilată unei unităţi fizice independente. În MS-DOS un harddisk poate avea maximum patru partiţii.

Versiunile MS-DOS de la 3.3 în sus sprijină partiţiile Extended-DOS, care permit împărţirea harddisk-urilor mai mari în unităţi logice. Partiţiile DOS pot fi împărţite în atâtea unităţi logice câte permite spaţiul pe harddisk. La un sistem tipic, cu un singur harddisk şi o partiţie primară DOS (unitatea C), o partiţie extinsă DOS poate cuprinde până la 23 de unităţi logice (de la D la Z). Aşa-numitele partiţii non-DOS pot fi utilzate de alte sisteme de operare, cum ar fi Novell Netware. În aceste cazuri, MS-DOS nu recurge la aceste partiţii. MS-DOS 4.0 şi 5.0 permit unităţi logice de până la 2 GByte. La acestă capacitate, o unitate fizică poate fi împărţită în până la 24 de unităţi logice, ceea ce simplifică foarte mult lucrul. La partiţionarea unui harddisk, MS-DOS începe instalarea cu octetul 1BE hex (446 În zecimal) din tabelul de partiţii principal. Formatul acestui tabel este reprezentat În figura 1.

Acest tabel conţine patru înregistrări de câte 16 biţi, fiecare înregistrare fiind împărţită în 10 câmpuri de lungimi diferite. În aceste câmpuri sunt înscrise toate informaţiile necesare descrierii unei partiţii a harddisk-ului:

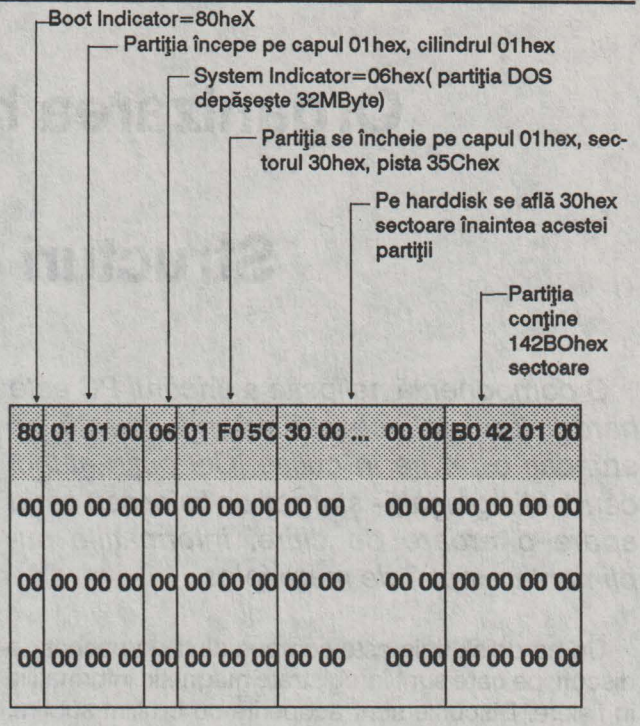


Fig. 2. O serie de înregistrări într-un tabel de partiţii.

- capul, sectorul şi cilindrul
- începutului şi sfârşitului partiţiei
- mărimea partiţiei, în sectoare
- informaţii despre modul de formatare şi sistemul instalat
- informaţia dacă partiţia de pe care porneşte sistemul este cea activă

Nu este necesar ca partiţiile să aibă o anumită mărime sau poziţie pe disc. Octeţii sectoarelor şi pistelor nu pot fi interpretaţi direct ca valori de 8 biţi. Cei 6 biţi inferiori indică numărul actual al sectorului (0 până la 63), pe când cei 2 biţi superiori - împreună cu cei 8 biţi ai câmpului cilindrului (pistei), generează numărul acestuia, folosind 10 biţi (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 şi cilindri. Bitul indicator al sistemului (offset 04 hex), indică sistemul instalat pe partiţia respectivă şi, în cazul partiţiilor DOS, şi tipul partiţiei. Partiţiile DOS primare sunt marcate, funcţie de mărime, cu 01hex, 04hex sau 06hex. În mod normal partiţiile sub 16 MByte se notează cu 01hex, cele între 16 şi 32 MByte cu 04 hex, iar cele peste 32 de MByte cu 06hex. Valoarea 05hex marchează o partiţie Extended-DOS. Toate celelalte valori pozitive marchează partiţii non-DOS. Număul 00hex indică o partiţie încă

nedefinită. Partiţia activă în sectorul Boot, conţine valoarea 80hex. Este permisă existenţa unei singure partiţii cu acestă valoare. Dacă PC-ul "se trezeşte" pe harddisk, un program auxiliar scurt citeşte în RAM sectorul care conţine tabelul principal al partiţiilor, executând apoi un alt program scurt, aflat la începutul acestuia. Acest program caută în tabel valoarea 80hex, în câmpul Boot Indicator. Primul sector cu informaţia căutată este transferat în memoria de lucru, acesta instalând o rutină de execuţie (Bootstrap-Loader) care încarcă MS-DOS (sau un alt sistem conţinut în partiţie) de pe harddisk. Dacă un harddisk urmează să conţină mai mult decât o unitate logică, trebuie instalată o partiţie Extended-DOS. Diferenţa dintre partiţia DOS primară şi Extended-DOS, constă în faptul că cea din urmă poate fi divizată în mai multe unităţi logice, fapt nepermis la partiţia DOS primară. Numerotarea pentru primul cap, primul sector şi primul cilindru, ale unei partiţii Extended-DOS, nu marchează poziţia unei partiţii fizice, ci indică alt tabel de partiţii, care descrie un volum logic. Dacă în partiţia Extended-DOS sunt definite unităţi logice suplimentare, a doua înregistrare în acest pseudo-tabel conţine adresa altui tabel, în care sunt cuprinse informaţiile referitoare la următoarea unitate logică. Tabele de partiţii se pot succeda în numărul dorit, pentru descrierea tuturor unităţilor logice. fig 2

Hexdump-ul din figura 2 reprezintă seria de înregistrări dintr-o tabelă de partiţii aparţinând unui harddisk de 40 MByte, care cuprinde o partiţie DOS primară de 20 MByte şi o partiţie Extended-DOS de 20 MByte. Aceasta din urmă este împărţită în două unităţi logice de câte 10 MByte.

Tabelul principal de partiţii conţine inscripţii pentru o partiţie de tip 04hex şi pentru una de tip 05hex. Numerotarea primului cap, sector şi cilindru se înscrie în înregistrarea făcută pentru Extended-DOS. Programul care citeşte sectorul care începe astfel: "cap 00hex, sector 41hex şi pistă ABhex", găseşte tabelul de partiţii nr.2 . Aici apar două înregistrări: una pentru o partiţie de tip 01hex şi una pentru o partiţie de tip 05hex. Partiţia de tipul 01hex este prima unitate logică a lui Extended-DOS. Dacă programul citeşte primul sector al partiţiei de tip 05hex, găseşte un al treilea tabel de înregistrări, cu o singură înregistrare de tip 01hex. Această partiţie este cea de-a doua unitate logică.

După ce harddisk-ul a fost partiţionat, diferitele volume pot fi formatate high-level. După formatare, fiecare volum MS-DOS are, în esenţă, următoarea structură:

- sector Boot
- File Allocation Table (FAT)
- una sau mai multe copii ale FAT-ului
- director principal
- zone pentru date şi subdirectoare.

Boot-Sector-ul se instalează în primul sector al unui volum, pentru a fi uşor găsit de către MS-DOS. Tabelul 1 descrie structura sa. El conţine informaţii referitoare la mărimea şi 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 şi execută codul respectiv, adică încarcă sistemul de operare MS-DOS. La începutul sectorului Boot se află o instrucţiune JMP din codul-program propriu-zis, care începe la adresa 1Ehex. La versiunea Short, care cuprinde doar 2 MByte, se adaugă în cod o instrucţiune NOP pentru a umple spaţiul până la următoarea înregistrare. În câmpul în care apare producătorul se poate înscrie orice, deoarece aceste informaţii nu se valorifică. Următoarele înregistrări se referă la parametrii fizici ai volumului, şi sunt valorificate de către funcţiile întreruperii 13hex. Aceste funcţii utilizează driver-e de unităţi sau programe utilitare pentru citirea şi înregistrarea datelor. De aceea aceste înregistrări se mai numesc şi *Bloc BIOS de parametri*. Rutina Boot se poate extinde peste mai multe sectoare dacă cei 482 Byte rezervaţi nu sunt suficienţi. Apoi aceşti primi 482 de Bytes trebuie reîncărcaţi de către prima parte a programului. În acest caz înregistrarea conţine, începând cu adresa 0Ehex, numărul sectoarelor în cauză, inclusiv sectorul Boot. La toate versiunile DOS până la 5.0 spaţiul din sectorul de Boot este suficient pentru Bootstrap-Loader, astfel încât înregistrarea cu adresa 0Ehex are conţinutul "1". Ca să puteţi crea un nou fişier, MS-DOS trebuie să ştie care sectoare ale volumui mai sunt libere. Aceste informaţii sunt cuprinse în File Allocation Table (FAT), aceasta urmând nemijlocit sectorului Boot (respectiv unor sectoare rezervate). FAT conţine numeroase înregistrări, care corespund unor grupuri de sectoare logic sucesive, denumite *cluster*-e. Pentru a afla câte sectoare compun un cluster, se citeşte 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 depăşeşte capacitatea de 128 MByte. Deoarece în FAT sunt rezervaţi maximum 16 biţi pentru înscrierea unui cluster, pot fi gestionate maximum 65536 cluster-e. La patru sectoare pe cluster şi 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 funcţie de mărimea volumului, cuprinde un număr crescut de sectoare într-un cluster. Astfel la mărimi între 128 şi 256 MByte, 8, între 256 şi512, 16, între 512MByte şi 1GByte, 32, iar între 1 GByte şi 2 GBytes, 64 de sectoare sunt cuprinse într-un cluster. Genialitatea acestui procedeu constă în faptul că structura

FAT nu este influenţată şi în plus, ea poate fi gestionată în toate cazurile, de aceeaşi rutină. Formând cluster-ele, se evită o fragmentare excesivă a fişierelor. În

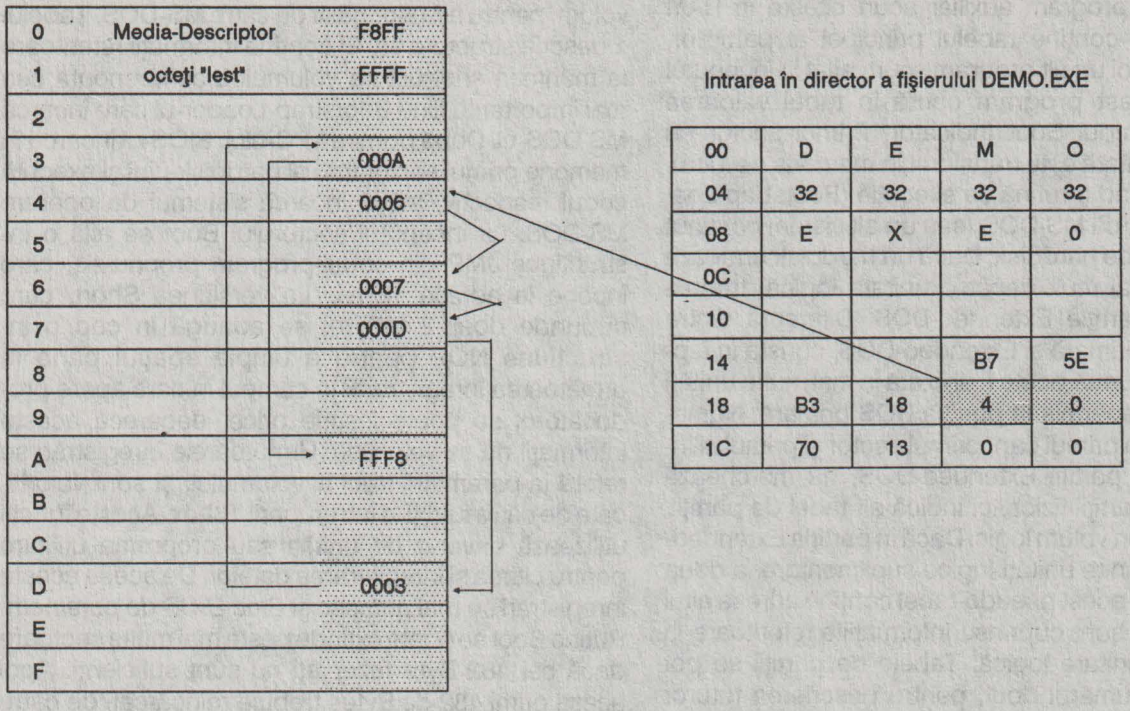


Fig. 3. Înlănţuirea inscripţiilor FAT, exemplificată

schimb acest lucru nu poate fi evitat la lucrul cu un disc pe care se creează şi de pe care se şterg foarte des fişiere. Atâta timp cât volumul este proaspăt creat, fişierele pot fi scrise pe disc unul după altul. După prima ştergere, golul produs trebuie completat. Dacă fişierul nou este mai mare decât cel vechi, trebuie fragmentat cel puţin în două părţi. Ca exemplu vom folosi un caz extrem, fără formare de cluster-e, la care un fişier de 100 de sectoare, este fracţionat în 100 de sectoare fizic distanţate între ele. Întîrzierile care intervin numai din poziţionarea de 100 de ori a capului de citire sunt inadmisibile. Deci prin formarea cluster-elor se garantează totuşi faptul că, patru sectoare fiind citite consecutiv, capul trebuie poziţionat numai de 25 de ori. Câştigul de viteză este cu atât mai mare, cu cât numărul de sectoare pe cluster este mai mare. împărţirea pe cluster-e are însă şi un dezavantaj: cu creşterea numărului de sectoare pe cluster creşte risipa de spaţiu. Imaginaţi-vă ce se întâmplă numai la un fişier 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 ripesc deja 3x512+412=1948 iar cu 64 de sectoare/cluster chiar 63x512+412=32668 Bytes.

Mărimea inscripţiilor în cadrul FAT depinde, începând cu versiunea 3.0 a lui DOS, de numărul cluster-elor de administrat. Până la 4096 de cluster-e cer 12 biţi, mai multe cer 16 biţi. Deoarece la 4 sectoare pe cluster şi 4096 cluster-e se gestionează doar 8 MByte, la ora actuală FAT-urile sunt, aproape în exclusivitate, de 16 biţi. Pentru a determina numărul de biţi pe inscripţie FAT, trebuie doar împărţită valoarea de la adresa 13hex (total sectoare pe volume) din Boot-sector, la inscripţia de la adresa 0Dhex (sectoare/cluster) . Dacă rezultatul este mai mic sau egal cu 4096, este vorba de inscripţii de 12biţi, în caz contrar de inscripţii de 16 biţi. Primele două inscripţii FAT sunt rezervate şi cuprind 3 octeţi (12 biţi) sau 4octeţi (16 biţi). Primul byte conţine aşa-numitul *Media-Descriptor*, care furnizează informaţii referitoare la formatul volumului şi suportul datelor. în pricipiu, la harddisk-uri are valoarea F8hex. Următorii doi sau trei octeţi conţin valoarea de "lest": 225.

Sectoarele unui fişier din cadrul FAT sunt dispuse într-o listă înlănţuită, şi se evaluează astfel: în inscripţia de director a fişierului respectiv, începând cu adresa 1Ahex, se află numărul primului cluster; inscripţia FAT corespunzătoare conţine numărul următorului cluster; la rândul ei inscripţia FAT a acestuia corespunde, din nou, numărului următorului cluster, etc., până la inscripţia FAT corespunzătoare ultimului cluster, care cuprinde marcarea sfârşitului. În cazul unui FAT de 16

Adresa	Conţinut	Lungime şi tip
+00hex	Instrucţiune de salt la rutina Boot	3octeţi
+03hex	Producător şi număr al versiunii	8 octeţi
+0Bhex	Octeţi 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 inscripţii 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	Distanţa la care se află primul sector din volum faţă de primul sector pe suportul de memorie	1 cuvânt
+1Ehex-1FFhex	Rutina Boot	482 octeţi

Tabelul 1 Conţinutul unui sector Boot

biţi, valorile sunt între FFF8hex şi 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 însuşi, inscripţia FAT corespunzătoare are valoarea FFF0hex până la FFF6hex. Figura 3 reprezintă, printr-un fişier fictiv "demo.exe" cuprinzând 5 cluster-e, înlănţurea inscripţiilor FAT.

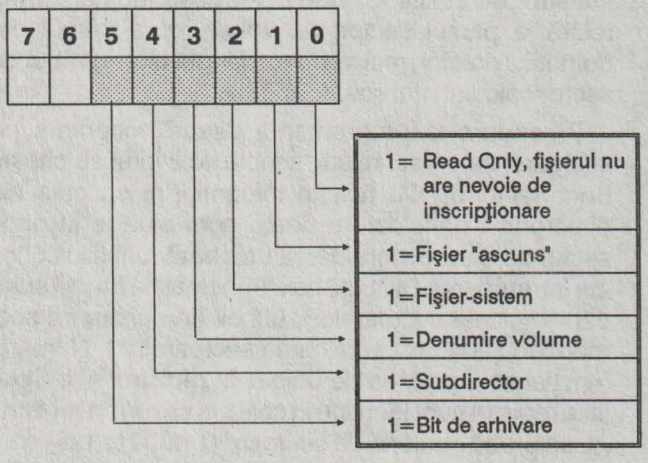


Fig. 4. Attributbyte-ul defineşte tipul inscripţionării directorului

Deoarece, în principiu, în caz de deteriorare a FAT-ului se distrug toate datele (datele există ca atare în sectoare, dar lipseşte 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 instrucţiunea 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 conţine până la 16 inscripţii de fişiere 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 inscripţii de director.

Primii 8 Bytes conţin denumirea fişierului; în caz de necesitate se umple spaţiul cu blank-uri. Interesant este procedeul de ştergere a unui fişier. Datele nu sunt şterse fizic, ci este modificat doar primul octet al denumirii fişierului în E5hex. Astfel se recunoaşte, la crearea unui nou fişier DOS, că inscripţia respectivă poate fi modificată. Dacă utilizaţi UNDELETE sub MS-

Adresă	Conţinut	Lungime şi tip
+00hex	Denumirea fişierului	8 octeţi
+08hex	Extensia fişierului	3 octeţi
+0Bhex	Atributul fişierului	1 octet
+0Chex	Rezervat	10 octeţi
+16hex	Ora	1 cuvânt
+18hex	Data	1 cuvânt
+1Ahex	Primul cluster al fişierului	1 cuvânt
+1Chex	Mărimea fişierului	1 dublu cuvânt

Tabelul 2 Structura unei inscripţii în director

DOS 5.0, programul vă solicită prima literă din denumirea fişierului şi apare: "a fost modificat în E5hex". Atâta timp cât nu înregistraţi un alt fişier în locul celui vechi, acesta din urmă poate fi reconstituit. Dacă primul simbol este "E5hex", MS-DOS îl înlocuieşte cu 05hex. Rămâne un secret al creatorilor lui MS-DOS de ce nu au ales direct codul 05hex pentru fişiere şterse. Sfârşitul directorului principal este marcat de un octe "zero". Octetul atribut (fig 4) defineşte tipul inscripţiei 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, fişierul a fost sau nu modificat; dacă da, atunci bitul de arhivare ia din nou valoarea "1".

În final se poate determina mărimea fişierului, cu relaţia: .

mărime = Word1 + 65536 x Word2,

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

Dacă bit-ul 4 din octetul atribut este ocupat, inscripţia din director este un subdirector. Acesta este gestionat dinamic de FAT (ca şi datele de altfel) astfel încât, teoretic , poate deveni nelimitat de mare. La crearea unui subdirector MS-DOS rezervă iniţial doar un cluster. Pe parcursul extinderii, se generează o înlănţuire de cluster-e (cum s-a explicat mai sus) gestionate de FAT. Inscripţia din directorul principal indică - exact ca la fişiere - primul cluster. Inscripţiile din subdirector au aceeaşi structură cu cele din directorul principal. Fiecare subdirector conţine semnele: "." pentru directorul actual, şi ".." pentru cel ierarhic superior. Aceste inscripţii nu pot fi şterse cu comanda `del`

Domeniul propriu-zis al datelor, care cuprinde şi 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 determinaţi pe baza inscripţiilor din sectorul de Boot astfel: numărul de sectoare din directorul principal se determină prin împărţirea la 32 a numărului de inscripţii (adresa 11hex), numărul de sectoare FAT prin înmulţirea dintre inscripţia de la adresa 10hex cu cea de la adresa 16hex, iar numărul sectoarelor rezervate se găseşte la adresa 0Ehex. Primul sector al unui fişier anume îl puteţi determina în felul următor: se reduce cu două numărul de cluster, deoarece primele două inscripţii conţin Media-Descriptorul şi "lest-ul", iar rezultatul se înmulţeşte 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 citeşte Boot-Sector-ul. Cu aceste informaţii şi cu cele ale directorului principal se poate apoi ajunge la orice sector de date. Programarea utilizează funcţiile 00hex din întreruperea BIOS 13hex. Principalele funcţii sunt: 02hex (citirea sectoarelor), 03hex (înregistrarea sectoarelor) şi 04hex (verificarea sectoarelor). Numărul funcţiei 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 evidenţă faptul că un harddisk poate avea mai mult decât 256 de cilindri, chiar dacă într-un registru de 8 biţi pot fi înregistrate doar numere de la 0 la 255. Deoarece nici un cilindru nu conţine mai mult de 63 de sectoare, biţii 7 şi 8 păstrează valoarea "0". De aceea le-au fost alocaţi biţii 8 şi 9 ai unui număr de cilindru format din 10 biţi. Pentru numărul de cilindru, lucrurile stau astfel:

Biţii 0- 7: registrul CH, biţii 0- 7

Biţii 8- 9: registrul CL, biţii 6- 7

Perechea de regiştri ES:BX trebuie să indice întotdeauna memoria-tampon a datelor (buffer-ul). Iată două exemple în acest sens. În primul exemplu urmează să fie citite 20 de sectoare, începând cu cilindrul 999, sectorul 3, şi 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 operaţia decurge fără greşală, toate funcţiile întorc un Carry-Flag poziţionat în 0. Dacă apare o eroare, Carry-Flag este poziţionat în 1, iar registrul AH conţine codul numeric al erorii.

Frank Riemenschneider

Traducere şi adaptare: ing. Mihai Beer

```
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 citeşte 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.

Medii de memorare optice

În aşteptarea MO-urilor

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

Ca medii de memorare optice, se face deosebirea între CD-ROM-uri, WORM-uri şi unităţi 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 şterse de câte ori se doreşte.

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 preţurile-hardware care scad puternic, iar pe de alta din numărul de aplicaţiipe care le susţine, aflat în continuă creştere. Ca şi CD-ul audio, CD-ROM-ul este placat numai pe o parte. Suprafaţa acestui mediu de memorare constă dintr-un strat de policarbonat artificial, un strat de aluminiu reflectant şi un strat de lac de protecţie. Pe suprafaţa plăcii se trage o urmă spirală, asemănător discurilor audio normale, chiar în momentul fabricării. Cu ajutorul unui laser, se ard nişte adâncituri, în spirala acesteia, la anumite distanţe. Spre deosebire de discurile audio, aici spirala se desfăşoară din interior spre exterior. Urma spirală se potriveşte foarte bine pentru înregistrarea secvenţială a cantităţilor mari de date. Procedeul permite producătorilor folosirea aceloraşi instalaţii pentru producerea CD-ROM-urilor, ca şi 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 uşor de localizat, dată fiind echidistanţaunei urme faţă de mijlocul discului. Acest procedeu este esenţial mai complex la o urmă spirală. În procesul de citire, urma este explorată de o rază laserfoarte concentrată. Dacă raza nimereşte într-o adâncitură, urmează o absorbţie; dacă nu, raza este reflectată şi detectată de o foto-celulă. Astfel datele sunt memorate binar, aşa cum este cunoscut de la mediile de până acum. Dacă raza este reflectată starea este 0 , dacă însă are loc o absorbţie, starea este 1. Adânciturile sunt desemnate drept *Pits*, iar locurile care rămân neschimbate, drept *Lands*. Timpii de acces ai unei unităţi CD-ROM se află undeva între 300 şi 400 de ms. O îmbunătăţire în acest sens, a fost obţinută prin procedeul *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, cerinţele în ce priveşte securitatea datelor sunt esenţial mai mari, decât la CD-urile audio. Mai mult de 10%, din spaţiul de memorare

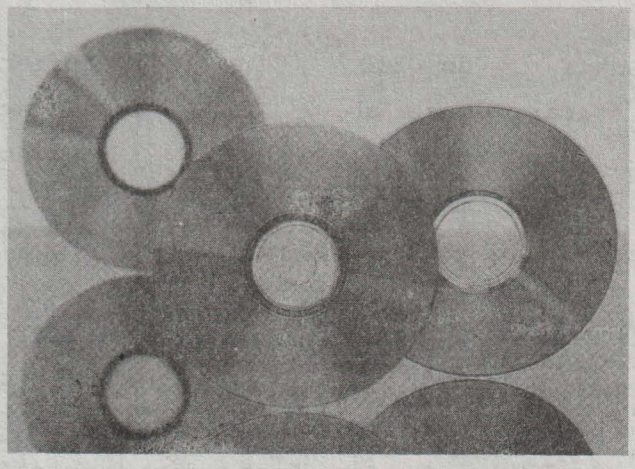


Fig 1: Capacitatea CD-ROM-ului este între 540 şi 630 MBytes per disc, ceea ce corespunde la 270.000 până la 317.000 de pagini A4 scrise

disponibil este ocupat cu gestiunea datelor şi testarea erorilor. Un avantaj al CD-ROM urilor este costul de fabricaţie 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 producţia CD-ROM. Un alt avantaj constă în faptul că în orice unitate CD-ROM pot fi citite, în mod normal, şi CD-uri audio. Rularea CD-ului audio se face aici sub controlul computerului. Se pot obţine şi unităţi 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 unităţi sunt însă, de regulă, sub cei ai unităţilor care au fost concepute special numai pentru CD-ROM. Pe baza formatului de înregistrare şi a vitezei de rotaţie, rata de transfer a datelor este aproape identică la toate unităţile CD-ROM. Ca domeniu de implementare, fără îndoială că CD-ROM-ului i se oferă, distribuirea software-ului. Aplicaţiile 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, faţă de purtătoarele magnetice de date, o siguranţă a datelor fundamental mai mare, astfel încât software-ul original nefalsificat stă oricând la dispoziţie; pe de alta, metoda face casă bună cu copyright-ul producătorilor. Tehnologia CD-ROM este ideală pentru aplicaţii Multimedia: fotografii color, desene, sound-uri şi, înainte de toate, imagini animate care cer capacităţi de memorare imense (fig 1)

Chiar în domeniul Multimedia, unităţile CD-ROM sunt livrate foarte des împreună cu plăci de Sound. Pe unităţile externe sunt disponibile carcaseîn care pot fi găzduite mai multe unităţi CD-ROM, astfel încât să poată fi accesate corespunzător cantităţi mari de date. Schimbătoarele-CD se oferă în special pentru folosirea în reţele. Mai nou se oferă şi 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 compatibilităţii lor cu CD-ROM-urile standard. CD-ROM-urile sunt bune în sensul că producătorul a stabilit din timp normele obligatorii, corespunzătoare procedeului de înregistrare. Acest *High-Sierra-Standard* stabileşte 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, aplicaţii care au fost concepute pentru *High-Sierra-Standard*, sunt rulabile şi sub 150-9660. Invers însă nu este posibil.

Ca interfaţă pentru unităţi CD-ROM, s-a dezvoltat adaptorul SCSI, ca standard, datorită flexibilităţii şi capacităţii sale (fig 2)

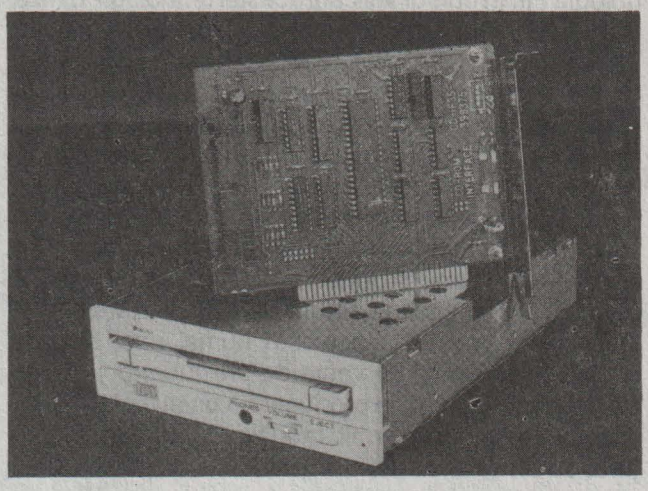


Fig 2 Unităţile care se bazează pe tehnologie de memorare optică, sunt comandate, de cele mai multe ori, printr-o interfaţă SCSI

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

Între unităţile CD-ROM existente, se găsesc şi variante avantajoase în ce priveşte costul, care pot fi cuplate prin interfaţă serială, respectiv paralelă. Spre deosebire de CD-ROM, mediul de memorare a lui WORM, poate fi reînscris cel puţin o dată. Astfel se apropie mai mult, de mediile de memorare magnetice tradiţionale. Discurile de 12", pot memora cîţiva GBytes, cele mai mici, de 5,25" atingând oricum o capacitate de circa 650 MBytes. Pe unităţi *Write-Once*, pot fi înregistrate date, până la 8 GBytes. dar nu numai din cauza mărimii mediilor de memorare, unităţile WORM, sunt livrate numai ca memorii externe. Deoarece plăcile pot fi folosite în sisteme asemănătoare Musicbox-urilor, posibilităţile de extindere sunt nelimitate. Cantităţile de date, puse astfel la dispoziţie, pot atinge domeniul Terabytes (1 TeraByte = 1024 GBytes). Acest ordin de mărime depăşeşte de departe, pe cel al suporturilor de date obişnuite. Bucuria unei capacităţi foarte mari, este ştirbită însă, de timpii de acces relativ mari.

WORM - (re) inscriptil o dată

Spre deosebire de CD-ROM, unităţile WORM dispun de două laser-e: un laser puternic, pentruscriere, şi unul slab, pentru citire. Pentru a înscrie datele, o rază laser explorează suprafaţa. Reflexia este redirectată şi transformată corespunzător modificărilor luminii, în date binare. Modul de funcţionare 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 şi reacţii termice, care produc în vecinătatea lor "văi" respectiv "culmi". În procesul de citire, o rază laser de intensitate mai redusă este reflectată şi apoi prelucrată. Prelucrarea reflexiei se face împrăştiind lumina pe frontiera bulei magnetice (bulele magnetice sunt zone, sau domenii cilindrice, din memoriile magnetice *bubble*, având aceeaşi formă de magnetizare - n.t.). Deşi acest procedeu serveşte numai unei singure înscrieri, căldura şi diferenţa de presiune, pot şterge informaţii. Procesul de producere a bulei (fig 3), necesită un strat metalic termic-modificabil, care este încojurat de un strat de polimer şi de unul de sticlă.

Spre deosebire de tehnologia *Pits*, placa are cu totul altă construcţie în ce priveşte crearea bulei. Deasupra unui strat reflectant, se găseşte un strat opac. Acest strat este ars de către laser, în locurile corespunzătoare, la citire. Deoarece procesul de scriere, la unităţi optice se face întotdeauna prin procedee optice, pericolele cunoscute până acum la sistemele magnetice, cum ar fi prăbuşirea capului de scriere-citire, pe suprafaţa discului (aşa-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 şi arhivele-microfişă ar trebui din când în când înlocuite cu unităţi WORM, dată fiind capacitatea de memorare enormă a acestora, şi timpii de acces substanţial mai reduşi. Deoarece datele de pe WORM nu pot fi şterse, domeniul principal de aplicaţii al acestor unităţi, este arhivarea.

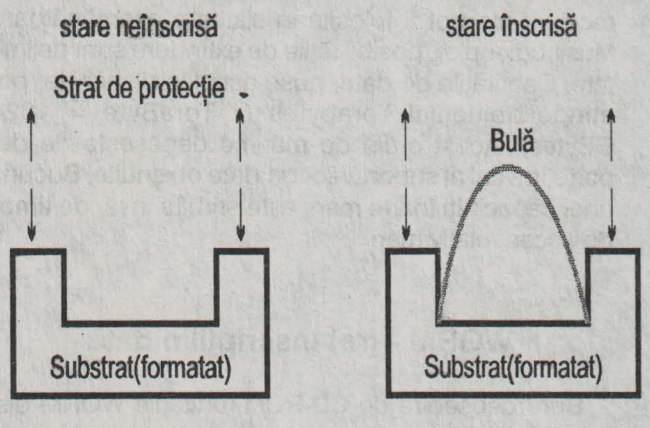


Fig 3 Procedeul de producere a bulei este des folosit la CD-ROM-uri. Aici, o rază laser, se ocupă cu producerea unui Pit.

Alături de mediile reinscriptibile, WORM îşi vor păstra dreptul de a exista, mai ales dacă ne gândim la faptul că, pe piaţa arhivărilor, se înregistrează rate de creştere mari. WORM sunt mai puţin sensibile la factorii de mediu decât CD-R-urile. Deoarece pot fi înscrise numai o singură dată şi numai o dată şterse, ele eşuează 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 combinaţie a tehnicii magnetice cu cea optică. Un procedeu de înregistrare care lucrează după pricipiul digital trebuie să producă două stări diferite într-un punct al mediului de înregistrare, şi să le poată apoi deosebi la citire. Acest lucru este obţinut cu raze laser, diferite ca putere, folosind şi un electromagnet.

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

Aici magnetizarea este perpendiculară pe suprafaţa plăcii. Însă, la puteri mari ale laser-ului, apare pericolul ca zonele de graniţă să fie şi încălzite şi influenţate de câmpul magnetic. De aceea raza laser trebuie să fie focalizată precis. Puterea laser-ului folosit, se găseşte în domeniul miliwaţilor, iar temperatura Curie la câteva sute de grade Celsius.

Ca şi pe suporturile de date de până acum, pe discul magneto-optic, apar două stări magnetice diferite: pozitiv şi negativ. Cele două stări sunt deosebite, prin polarizarea creată de reflexia razei laser, mai săracă în energie. Raza reflectată este polarizată, funcţie de poziţia magneţilor (se vorbeşte de aşa-numitul efect-Kerr). Conţinutul de informaţii 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 ştergere, abia apoi începând procesul propriu-zis de scriere. Acest lucru explică şi de ce un proces de scriere pe un disc magneto-optic este mai mare consumator de timp, faţă de suporturile de date de până acum.

Alternativa: memorii magneto-optice

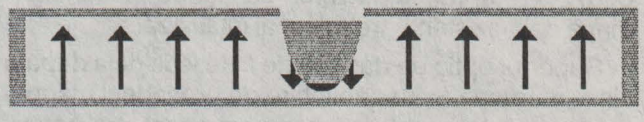
Tehnologia magneto-optică reclamă o tehnică optică complicată pentru capul de scriere-citire. Acest lucru afectează negativ mărimea şi masa capului. Un cap mare şi greu nu se lasă manipulat rapid, influenţând astfel viteza de acces. Acest fapt explică de ce unităţile magneto-optice au performanţe mai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

Unităţilor magneto-optice li se poate asocia denumirea de "hibrid" ,între sistemele pur magnetice şi cele pur optice. Avantajele discurilor magento-optice sunt la îndemână: în formatul de 5,25", pot înregistra date între 600 MBytes şi 1 GByte. Discurile mai mici, tot de acest tip, cu dimensiunea de 3,5" , cuprind şi 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 pote fi modificată decât dacă locul respectiv este încălzit la temperetura necesară. De aici, garantarea securităţii datelor pe perioadă mai lungă. Dar şi alte influenţe exterioare, cum ar fi umiditatea, căldura sau manipulareaneglijentă, nu deteriorează acest tip de discuri. O serie de discuri descriptibile, sunt concepute pentru o siguranţă a datelor între 10 şi 100 de ani.

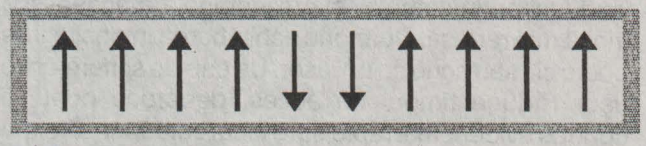
Domeniile de implementare a unităţilor magneto-optice rezultă din avantajele lor, cantităţile mari de date memorate nefiind o problemă. Cine şi-a încercat deja mâna, pe scanner-e color cu profunzimea culorii de 24 de biţi, şi-a dorit desigur, cu pasiune, şi o unitate de memorie corespunzătoare. Aici,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

1) Stare de bază

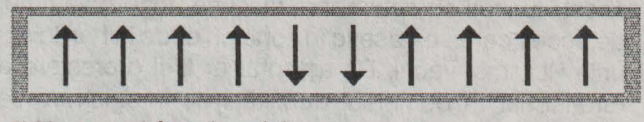
2) Încălzitor laser



3) Orientare (crearea unui câmp magnetic)



4) Răcire (în câmpul creat)



5) Memorare (câmp decuplat)

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

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 unităţile WORM profesionale. Mai nou îşi g.seşte utilizare şi la plăcile optice reinscriptibile. La această tehnologie este vorba de un procedeu de înregistrare pur optic, care se descurcă fără electromagneţi. Datele vor fi scrise şi citite prin 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 şi 20 de miliwaţi. Aceştia sunt suficienţi pentru a încălzi stratul corespunzător până la punctul de topire. Locurile prin care "calcă" laser-ul de scriere îşi modifică reflexia, ceea ce se manifestă la explorarea cu laser-ul de citire.La procedeul cu schimbare a fazei, diferenţierea între cele două intensităţi se face prin fotocelule, spre deosebire de tehnologia magneto-otică unde este determinată de polarizarea luminii. Dacă după reflexie intensitatea a scăzut foarte mult, atunci se recunoaşte starea 1. Dacă raza laser este reflectată fără pierderea intensităţii, aceasta descrie starea 0. La

scriere, raza puternică modifică structura atomică prin încălzire şi răcire rapidă, modificând atfel capacitatea de reflexie a materialului. Aici se poate utiliza faptul că unele materiale pot fi pendulate între starea amorfă şicea cristalină cu o rază laser. În stare amorfă, aceste materiale dovedesc un grad de reflexie evident redus, starea aceasta (starea 1) fiind obţinută prin încălzire până la punctul de topire. Pentru a readuce particulele în starea cristalină, este suficientă folosire unei raze laser mai slabe. Prin reducerea căldurii, este redusă libertatea de mişcare a particulelor, ele reaşezându-se în structura cristalină (pentru noi starea 0). Diferenţele între intensităţile de reflexie sunt aicifundamental mai mari decât diferenţele 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, informaţia deja existentă trebuie mai întâi ştearsă.

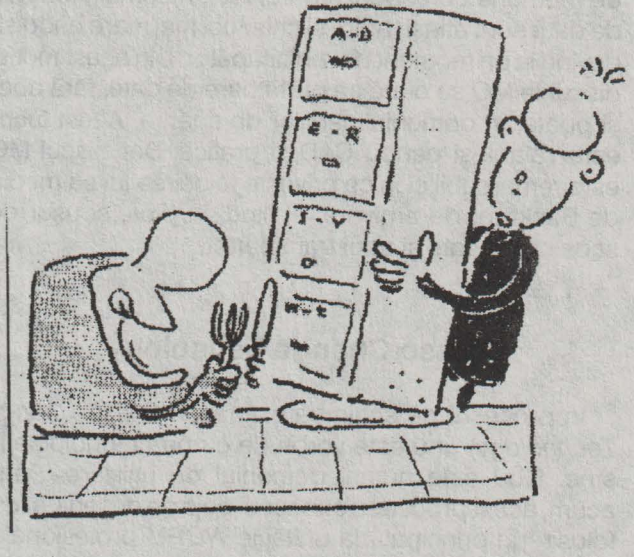
Suportul optic de date poate fi descris deja după o singură rotaţie, spre deosebire de procedeul magneto-optic unde sunt necesare permanent, trei rotaţii. Această (din urmă) unitate optică este mai redusă ca preţ decât cea magneto-optică, capul de scriere-citire fiind mai puţin complex şi prezentând, de aceea, şi o masă mai redusă. Doar prin schimbarea intensităţii se poate citi sau scrie cu un laser. Un cap de scriere-citire uşor reduce timpul de acces, deoarece poate fi condus evident mai rapid spre locul corespunzător pe placă. În ce priveşte timpul de acces, este clar că posibilităţile tehnice nu au fost epuizate, chiar dacă la aceste unităţi cu schimbare fazei se obţin deja timpi de acces,care se găsesc în zona celor de la harddisk-urile RLL mai vechi. De aşteptat ar fi şi o creştere a turaţiei, iar în domeniul densităţii de înregistrare se obţin succese importante. Abia atunci când timpii de acces vor deveni comparabili cu cei ai memoriilor mágnetice, procedeele nemagnetice vor intra în competiţie directă cu acestea.

În ce priveşte capacitatea, la mediile optice, se deosebesc două procedee de înscriere. Pe de-o parte îşi găseşte utilizare procedeul CAV (*Constant Angular Velocity*), aici turaţia plăcii rămânând constantă, şi de aceea densitatea datelor este mai mare în domeniile interioare decât la periferie. Pe de altă parte, este procedeul CLV (*Constant LinearVelocity*), cu o folosire mai eficientă a suprafeţei plăcii. Aici densitatea informaţiilor este aceeaşi, atât pe urmele interioare, cât şi pe cele periferice. Capacitatea de memorare, în general mai mare, este echilibrată - din păcate - de timpul de acces crescut, deoarece pe parcurs turaţia trebuie schimbată şi se cere şi o organizare mai complexă a datelor.

Oricum sensul dezvoltării este, fără îndoială, spre unităţi optice reinsciptibile. Viitorul aparţine unităţilor combinate, care să poată citi atât WORM-uri cât şi CD-uri, cât şi Phase-Change-uri; acestea vor trebui, în plus, să poată şi înscrie diferite tipuri de suporţi. Rate de creştere înregistrează şi *Jukebox* -urile, care pot gestiona mecanic mai multe discuri. Oricum, în privinţa *Jukebox*-urilor, în faţă stau CD-urile. Şi în ceea ce priveşte procedeele de producţie, cercetare va înainta, aşteptându-se preţuri scăzute şi capacităţi 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 şi oferă capacitate mai mare, la gabarit în scădere. Tot aşa, în domeniul arhivării, dezvoltarea unităţilor de bandă magnetică progresează. Însăşi mult-apreciata 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-şi cântărească alegerea, funcţie de domeniul de implementare. Se profilează o reunire de tehnologii, nicidecum o concurenţă a lor.

Matthias Günter

Traducere:
ing. Mihai Beer



Reuşită în a cincea dimensiune

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

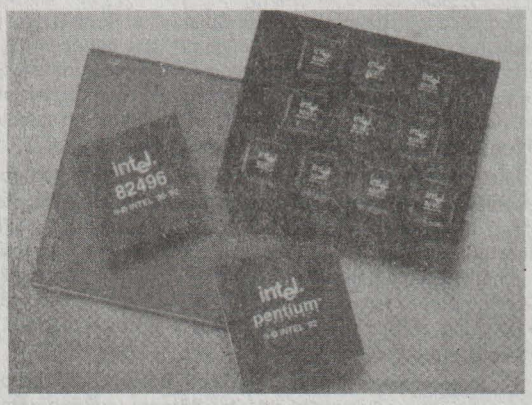
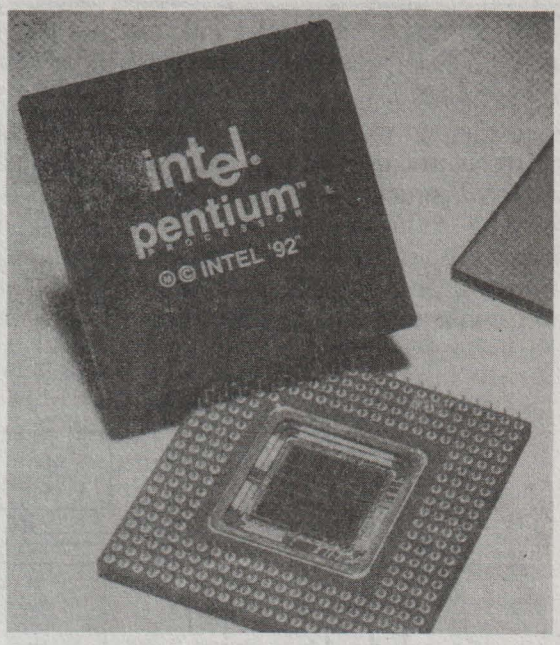


Fig. 1 - Aproape incredibil arată noul procesor de la Intel. Alături noua serie de chipuri pentru Pentium.

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

Superscalar - din doi facem unul

De mai multă vreme este cunoscut faptul că noul zar aruncat de Intel - cum era şi de aşteptat - nu va purta numele de 586. Această denumire s-ar fi potrivit în ordinea 8086, 80286, 80386 şi 486; şi totuşi, după certuri cu concurenţa asupra drepturilor, a devenit clar că Intel nu va putea să-şi protejeze numele sub forma unei combinaţii 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



obişnuit de alegere a numelui şi au botezat noul procesor cu numele, protejat prin drept de autor, de *Pentium*.

Dar nu numai ca nume se deosebeşte *Pentium* de premergătorii săi. Şi 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 unităţilor de integrare, urmând prelucrarea în paralel.

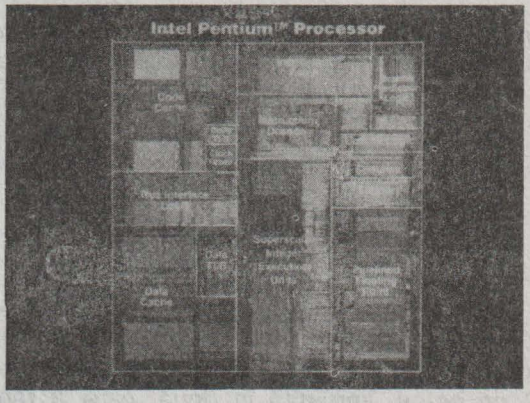


Fig 2 Pe 300 milimetri pătraţi de siliciu sunt înghesuite cele 3,1 milioane de tranzistoare

Dar acest lucru nu este suficient. La o instrucţiune de salt condiţionat sau necondiţionat, *Pentium* încearcă să aibă pregătită instrucţiunea următoare

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

Ea conţine în memorie ramificări ale programului, pentru a putea prevedea deja instrucţiunea corectă la repetarea instrucţiunii de salt. Dacă predicţia este corectă, a doua pipeline poate să îşi dea drumul neîntârziat. Acest truc rafinat îi aduce lui *Pentium* cel mai mare câştig de capacitate faţă de 486. Figura explică construcţia internă a procesorului *Pentium*.

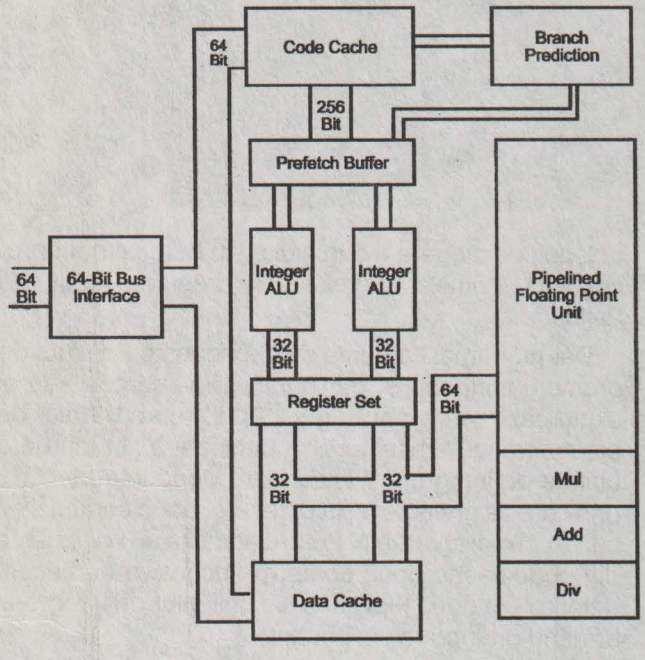


Fig 3 Desenul arată, în mare, construcţia internă a procesorului. Semnificative sunt cele două unităţi de integrare caracteristice arhitecturii superscalare.

BPL prezice

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

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

Şi la cod recursiv (care se schimbă prin el însuşi) consistenţa datelor se păstrează. De aceasta se îngrijeşte o unitate proprie, care verifică permanent dacă conţinutul unui cache mai este actual sau dacă se găseşte deja în forma actualizată în celălat cache.

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

Concurenţa RISC

La începutul anului 1991 producătorii de vârf din domeniul hardware şi software, s-au aşezat la o masă pentru a stabili un numitor comun în ce priveşte posibilitatea folosirii, pe căi simple, a softurilor existente şi pe alte platforme hardware. Acestei iniţiative numită *ACE* (Advanced Computimg Environment) i-au aparţinut, între alţii, Microsoft, Intel, MIPS, Sun şi DEC. Chiar dacă această reuniune a eşuat datorită unor frecuşuri interne, a avut cel puţin un efect: generaţia 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 şi o versiune corespunzătoare ACE. Aceasta este funcţională nu doar pe procesoare MIPS - R3000, R4000 şi R4400, ci rulează şi pe procesoarele Alpha ale firmei DEC precum şi 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 concurenţii lui Intel cu cele ale lui *Pentium* - noul născut - situaţia nici nu este chiar atât de roză.

Tocmai a fost etalat procesorul Sparc al lui Sun. El este cel mai uşor comparabil cu *Pentium* în ce priveşte capacitatea. La procesorul Sparc este vorba despre un CPU eminamente RISC, care este inima 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 şi DEC. Ambele lucrează cu o frecvenţă de tact de 150 MHz. Aceasta are ca rezultat teoretic, datorat arhitecturii superscalare, o capacitate de 300 mips. Dar şi la Specmark - Benchmark-uri mai realiste - este bătut *Pentium* de cele două procesoare de 64 de biţi. Atât Alpha de la DEC cât şi R4400 ajung, cu peste 100 Specmarks, la aproape de două ori valoarea pentru *Pentium*.

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

60 MHz şi într-una de 66 MHz. Astfel, în această privinţă nu este depăşit 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 frecvenţei

Aproape toate instrucţiunile 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".

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

Pentium - minunea calculului

După părerea celor de la Intel, tehnica procesoarelor pe 32 de biţi 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 şi el destul de puternic prin bus-ul local. Absoluta necesitate a unui procesor de 64 de biţi nu se întrevede momentan la Intel.

Cea mai mare creştere a capacităţii a obţinut coprocesorul integrat. El lucrează acum ca pipeline cu 8 nivele cu o unitate de adiţie, multiplicare şi diviziune implementată hardware în mod corespunzător. Un bus de 64 de biţi asigură un transfer de date rapid. Spre deosebire de 486 rezultă un câştig în performanţă de cinci până la zece ori mai mare. Chiar şi în cazul mediilor de aplicaţii 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 demonstraţie, a arătat într-un mod impresionant creşterea de capacitate a noului CPU. Un demo care "vrăjea" o grafică 3D în timp real, pe ecran, a mers cu o viteză obişnuită doar pe workstations. Şi Windows a profitat foarte mult de pe urma acestui procesor de mare capacitate.

În calculatorul pregătit pentru demeonstraţie lucra, acoperit de un radiator gigantic, un *Pentium* pe 66 MHz, pe o placă-procesor separată (figura 4). Avea la dispoziţie 32 MBytes de memorie internă. Comunicarea între CPU şi cartela grafică ATI-Match-32 se făcea prin bus-ul local PCI, specificat de Intel.

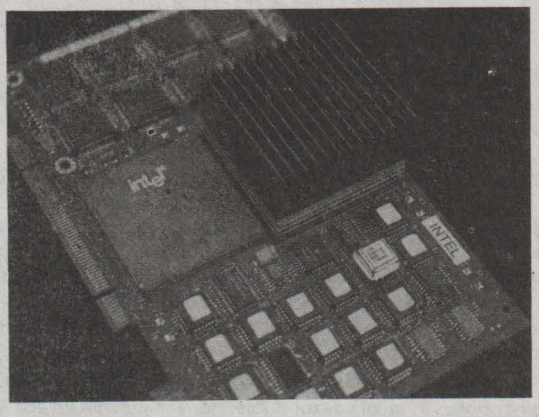


Fig 4 Figura arată placa-procesor a calculatorului pentru demonstraţie. Pentium este ascuns sub radiatorul uriaş.

Pentru a exploata la maxim potenţialul lui *Pentium*, trebuie ca şi software-ul să fie adaptat arhitecturii superscalare. De aceea Intel a dat mare importanţă 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 şi soft-ul vechi, de pe procesoarele 80x86, codat cu acestea, nu-şi pierde din capcitate.

O atenţie deosebită a acordat Intel temei *Multiprocessing*. Primul pas, fără creştere de capacitate, a fost funcţionarea a două procesoare *Pemtium* într-un mod de observare. Aici unul din procesoare preia calculul propriu-zis iar celălalt verifică corectitudinea lucrului. Dacă apar neconcordanţe este trimis un mesaj corespunzător. Această funcţie de siguranţă este gândită, înainte de toate, pentru file-server-e.

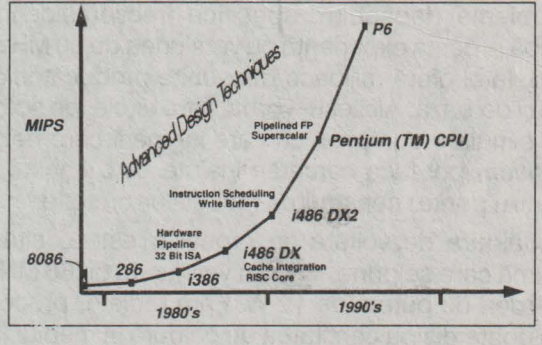


Fig. 5 - Capacitatea lui Pentium comparată cu cea a predecesorilor săi. De pe acum să ne bucurăm de următorul (P6) .

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 neobositconcurenţa pentru a rupe monopolul originalului. Dacă se iau ca bază cifrele actuale de vânzare, Intel nu poate renunţa în nici un caz la această piaţă. În ultimul an au fost mai mulţi 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 creştere a capacităţii lui 486. Ceea ce este bun pentru *Pentium* este pentru 486 doar ieftin - un Cache intern mărit. Astfel şi 486 va obţine un Cache de memorie de 16 KByte, pentru îmbunătăţirea performanţelor sale.

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

Concurenţei îi este greu în momentul de faţă să apară pe piaţă cu un 486 cinstit. Oricum AMD are un 486 funcţional 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 lipseşte de asemenea coprocesorul matematic.

Mai aproape de model se află noua creaţie 486S2-50 al lui Cyrix. Şi acesta, ca şi 486DX/2, se serveşte 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 slăbuţ în comparaţie cu cel de 8 KB al originalului. Totuşi, algoritmi de Cache rafinaţi pot suplini această lipsă. Ca şi lui 486SX şi lui Cyrix 486S2-50 îi lipseşte coprocesorul matematic. Dar încă înaintea sfârşitului lui 1993 va urma un preţios 486 DX.

Şi IBM dezvoltă o versiune proprie de 486 DX. Cu o frecvenţă de tact de 100 MHz şi 16 KB Cache intern va intra în competiţie "super-486".

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

Şi pentru modul multiprocesor real are *Pentium* pregătite câteva "features" . Ele afectează, în primul rând, chestiuni ale consistenţei datelor. Aici sunt implementate hardware câteva funcţiuni 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 economiseşte efort de programare, dar creşte şi performanţa.

Griji şi-au făcut cei de la Intel, în legătură cu frecvenţa de tact. Oricum pe bus-ul de procesor există 60 MHz respectiv 66 MHz - care determină apariţia unor probleme deosebite, specifice frecvenţelor înalte. După proasta experienţă 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 pentru construirea modelelor-prototip.

Căldura dezvoltată de procesor este o altă probelmă 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-şi construiască propria variantă de disipare. Procesorul este livrat fără radiator.

Cât costă *Pentium*, Intel nu vrea să desconspire. În schimb dă asigurări că preţul 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 preţul cu cinci cifre.

Rezumat

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

Pentru aplicaţii grafice şi domeniul CAD probabil că acest salt de capacitate a fost dorit de mult, însă cea mai mare a utilizatorilor nu folosesc această performanţă. Pentru ei este mai importantă o placă grafică mai rapidă sau un hard-disc, decât o creştere a capacităţii CPU-ului. De această realitate sunt convinşi şi 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 producţie - o cifră aproape jenantă în comparaţie cu cele 30 de milioane de 486 produse în acelaş timp. Astfel în anii 1993 şi 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 şi cu hardware rapid el se chinuie. Utilizatorii frenetici de NT vor forma piaţa mare de *Pentium*. Ei vor fi urmaţi, pe baza experienţei de mai târziu, de cei care vor dori software eficient nu unul care să facă doar abuz de resurse.

Traducere:
ing. Mihai Beer

Lanţuri aruncate în aer

Programarea lui *Extended Memory*

Acum 13 ani, când PC-urile apăreau pe piaţă, cei 640 de KBytes ai memoriei de lucru cu care erau echipate păreau aproape gigantici. Pe măsura dezvoltării în continuare a software-ului, a reieşit faptul că, pentru multe aplicaţii, 640 de KBytes sunt pur şi simplu prea puţin. Acest articol arată, cum puteţi "arunca în aer lanţurile" limitării memoriei şi cum puteţi folosi aşa-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 rezervaţi pentru memoria-ecran a cartelelor grafice, BIOS şi extensiile BIOS, MS-DOS are la dispoziţie numai 640 KBytes.

Pentru a sparge graniţa de 640 KBytes, specifică MS-DOS-ului, există două procedee. Pe de-o parte aşa-numita *Expanded Memory*, după standardul LIM-EMS, elaborat împreună de Lotus, Intel şi Microsoft, iar pe de alta, *Extended Memory*, care rezultă inevitabil din construcţia unui PC-AT sau 386. *Expanded Memory* extinde memoria aflată la dispoziţia MS-DOS-ului, utilizând porţiunile nefolosite ale domeniului de memorie rezervată (cei 384 KBytes), despre care vorbeam mai sus. Procedeul EMS (*Expanded Memory System*) intercalează în poziţii neocupate din interiorul memoriei, aşa-numite "pagini de memorie" , având mărimea de câte 16 KBytes. Aici se cer, oricum, şi plăci EMS speciale, care lucrează cu o logică de comandă deosebită, pentru a putea intercala şi apoi scoate paginile de memorie în şi din spaţiul de memorie dorit. Administrarea paginilor de memorie ale lui *Expanded Memory* este preluată de un driver EMM (*Expanded Memory Manager*), care pune la dispoziţie mai multe funcţii prin intermediul întreruperii 67 hex. *Expanded Memory* poate fi însă şi 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 graniţa de 1 MByte şi se numeşte *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 şi sub MS-DOS, domeniul de memorie de deasupra limitei de 1 MByte. Limitări, deoarece accesul controlat, cu simţ de răspundere, la *Extended Memory*, presupune o programare destul de stufoasă. Motivul este, din nou (cum ar putea fi altfel !?!) concepţia 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 înţeles, deoarece el nu poate adresa mai multă memorie cu circuitele sale. Dar 80286 şi 80386, pot de fapt mai mult. Aici MS-DOSpoate fi manevrat numai în *Real Mode*, care emulează cu mare precizie numai proprietăţile 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 aşa-numitul *Protected Mode*, care a fost adăugat la 80286 şi 80386. Acest mod de funcţionare este de fapt noutatea, în comparaţie cu procesorul 8086/8088. *ProtectedMode* 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 faţă de *Real Mode*. Astfel pot fi adresaţi pînă la 16 MBytes, respectiv 4 GBytes de memorie de lucru. Deci în *Protected Mode*, accesul la *Extended Memory*

nu pune nici un fel de probleme; aici pot fi adresaţi, în plus faţă de domeniul convenţional de memorie, mai mulţi MBytes. Cum pot fi folosite aceste capacităţi, când MS-DOS funcţionează totuşi numai în *Real Mode*? Acest lucru este făcut posibil, prin faptul că sub MS-DOS, puteţi comuta şi în *Protected Mode*, poziţionând un *statusbit* special (flag), în cuvântul de stare al maşinii. Prin aceasta puteţi accesa memoria extinsă după cum doriţi, şi apoi să reveniţi în *Real Mode*. Acest procedeu funcţionează într-adevăr, folosirea sa fiind oricum, mult mai greoaie decât pare la prima vedere. Cea mai simplă sarcină este comutarea în modul protejat. Pentru aceasta, 80286 şi 80386 au la dispoziţie comanda în cod-maşină "lmsw", cu care comutarea se face mai lejer. Greutăţile apar atunci când această comandă nu este cea din urmă de executat. Dacă încercaţi să comutaţi în *Protected Mode* numai la comanda "lmsw", de regulă computerul va "cădea". Aceasta se datoreşte faptului că mulţi regiştri şi multe structuri de date, care nu aveau nici o importanţă în *Real Mode*, pot fi de mare importanţă pentru execuţia programului în *Protected Mode*.

Dacă aţi trecut şi peste acest hop, vă aflaţi în faţa problemei accesului propriu-zis la memoria extinsă. Acest lucru nu mai este dificil când dispuneţi de suficiente cunoştinţe de programare. ale lui 80286 şi 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 comutaţia 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 aşa ceva, cu toate că modul protejat oferă mult mai multe posibilităţi 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. Totuşi 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 porneşte 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. Aşa cum puteţi constata din acestă scurtă prezentare, există o serie întreagă de greutăţi când se încearcă trecerea în mod protejat şi apoi în real. Din fericire nu trebuie să vă ingrijiţi dumneavoastră de toate amănuntele în ce priveşte programarea, deoarece BIOS-ul unui AT sau unui PC-386 conţine câteva funcţii, pentru accesul la memoria extinsă.

Cu aceste funcţii se poate determina mărimea memoriei extinse, poate fi comutat procesorul în modul protejat şi celulele de memorie din *Extended Memory* pot fi înscrise sau citite. Aceste funcţii sunt puse la dispoziţie drept subfuncţii ale întreruperii BIOS 15hex. Figura 1 reuneşte funcţiile şi parametrii corespunzători.

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

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

Prin funcţiile sale, întreruperea 15hex poate face, în principiu orice lucru care are legătură cu modul protejat. De un interes deosebit sunt primele două funcţii ale acestei întreruperi. În acest context, funcţia 89 hex este neinteresantă deoarece, aşa cum am menţionat, de îndată ce este apelată această funcţie, computerul se prăbuşeşte, dacă anterior nu au fost utilizaţi anumiţi regiştri. Funcţia 88 hex este foarte uşor de manipulat. Cu aceasta puteţi determina mărimea memoriei extinse. După apelul acestei funcţii se stabileşte mărimea, în KBytes, a memoriei extinse şi este întoarsă în registrul AX. Cele două programe-exemplu scurte, "exramchk.pas" (listing 1) şi "exramchk.c" (listing 2), demonstrează utilizarea funcţiei 88 hex în Turbo-Pascal şi Turbo-C. Ambele programe constată câtă memorie extinsă este instalată şi întorc un mesaj corespunzător. Ceva mai complicat este lucrul cu funcţia 87 hex. Cu acesată funcţie pot fi transferate blocuri de memorie, atât din memoria de lucru convenţională în *Extended Memory*, cât şi invers. Aşa cum arată figura 1, funcţia 87 hex, aşteaptă adresa de segmentşi de offset, a unei GDT. Pentru a explica semnificaţia acestei GDT, trebuie să revenim puţin. Procesoarele 80286 şi 80386 au fost dezvoltate cu scopul de a produce condiţiile iniţiale hardware pentru un mod de lucru multitasking efectiv. Pe lângă mecanismele de protecţie a domeniului de adrese ale unui program faţă de un altul, aparţin aici şi procese de

salvare pe disc, la nevoie, de programe întregi, părţi de programe sau domenii de date, când memoria de lucru este folosită în alte scopuri. Dacă atunci sunt necesare din nou acele părţi de program sau acele date, ele sunt aduse înapoi în memorie de pe harddisk. Acest procedeu presupune existenţa dinainte a unei adresări a memoriei, cu totul diferite de cea din *Real Mode*. De exemplu, pentru a citi un Byte în *Real Mode*, folosiţi o adresă de segment şi 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 şi spaţiul 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 unităţi de gestiune a memoriei (*Memory Management Unit*, pe scurt MMU), integrate în procesor.

Deci conţinutul unui registru segment nu va fi înţeles în *Protected Mode* direct ca adresă, ci ca "sector". Acesta este un indicator spre o tabelă în care se găsesc, din nou, "descriptori". În sfârşit, un descriptor descrie parametrii fizici ai unui segment de memorie, deci a adresei fizice, a lungimii şi priorităţilor de acces ale acestuia. Tabela în care se găsesc descriptorii este denumită Tabelă de Descriptori. Un program care rulează în modul protejat, foloseşte 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 dispoziţie un marcaj corespunzător, în descriptor. La un acces la acest segment, MMU recunoaşte că domeniul de memorie nu este încă la dispoziţie în memoria de lucru, şi încarcă segmentul corespunzător de pe suportul de memorie. Prin selectoare şi descriptoare 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 convenţională şi 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 şi memoria extinsă). De aceea întreruperea aşteaptă adresa unei tabele globale de descriptori, în poziţia din memorie definită prin combinaţia de regiştri 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 construcţia unui GDT.

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

Figura 2: Construcţia unei GDT (Global Descriptor Table)

Pentru aplicaţia următoare, sunt importante numai câmpurile "adresă-sursă" şi "adresă-destinaţie". Toate celelate câmpuri sunt completate independent, la apelul lor, efectuat de către BIOS. Ambele câmpuri conţin descriptori. Prin aceştia trebuie dată adresa fizică a segmentelor de memorie care să servească drept sursă, respectiv dreptdestinaţie, a transferului. Fiecare descriptor are lungimea totală de 8 Bytes. Figura 3 redă construcţia 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: Construcţia unui descriptor, în interiorul unei GDT

trebuie să indicaţi adresa fizică a segmentului de memorie de transferat. Pentru această adresă sunt prevăzuţi 24 de biţi, 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 şi PC-urilor 386 nu poate face deosebirea între diferitele programe care fac acces la *Extended Memory*. În măsura în care trans-

miteţi corect parametrii, BIOS-ul execută tramsferul 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 acelaşi domeniu din memoria extinsă, pentru că "nu ştiu unul de altul". Începând cu versiunea 3.3 a lui MS-DOS, lucrurile merg şi mai rău: aici se poate găzdui chiar RAM-discul în *Exteded Memory*, iar începând cu versiunea 4.0, pot fi găsite aici fişiere-tampon sau zone-tampon. Dar cine garantează că aceste programe nu folosesec exact domeniul din memoria extinsă pe care l-aţi ales pentru memorarea datelor dumneavoastră ? Primul adaos la soluţie constă în împiedicarea unui program de a utiliza memoria ocupată de o altă aplicaţie. Pentru aajunge la o concluzie, trebuie să vă întrebaţi de unde va şti un program care parte de memorie o poate folosi şi care nu. Pentru aceasta, programul cere mărimea memoriei extinse prin funcţia 88 hex a întreruperii 15 hex. Mărimea întoarsă de această funcţie, va fi apoi folosită integral sau parţial.

Deci un program nu foloseşte memorie neautorizată de funcţia 88 hex, deoarece porneşte de la ideea că, peste valoarea determinată, nu mai există spaţiu disponibil în memoria extinsă. Atunci, pentru a proteja un domeniu de memorie de accesul unui al doilea program, trebuie să vă îngrijiţi 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 redirectaţi vectorul întreruperii 15 hex (semnificând adresa rutinei de tratare a întreruperii aparţinătoare), spre o rutină proprie.

Fie cazul unui program pe care l-aţi scris dumneavoastră, care este foarte "avid" de spaţiu 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, funcţia 88 hex întoarce, în mod corespunzător, valoarea 1024. Deci, programul dumneavoastră trebuie să redirecteze întreruperea 15 hex spre o rutină proprie, care să intercepteze funcţia 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 funcţia 88 hex, el nici măcăr neîncercând să acceseze domeniul de memorie de deasupra (în acest caz, cei 640 KBytes rezervaţi de dumneavoastră), acesta fiind deci protejat la acces. Avantajul acestui procedeu este că poate funcţiona de mai multe succesiv deoarece al doilea program care utilizează memoria extinsă poate, la rândul lui, să redirecteze vectorul întreruperii pentru a "amăgi" un program care urmează, cu o mărime a memoriei extinse şi mai reduse.

O problemă apare totuşi, 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 obţinut redirectând vectorul întreruperii 15 hex, din nou pe rutina iniţială din BIOS.

Ce se întâmplă însă dacă, între timp, s-au băgat în vector alte programe? Rutinele-service de întrerupere sunt şi ele "scoase", iar problema iniţială reapare: un program proaspăt încărcat, porneşte iar de la premiza că are la dispoziţie 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 numeşte *vdisk.sys* sau *ramdrive.sys*, şi 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 verificaţi, dacă nu este deja instalat un RAM-disk. Dacă este cazul, atunci trebuie să determinaţi mărimea RAM-disk-ului. După aceeea, puteţi utiliza, pentru aplicaţiile dumneavoastră, memoria aflată deasupra acestui domeniu. Pentru a stabili dacă este instalat un RAM-disk, trebuie ştiut cum funcţionează, de fapt, acesta: un RAM-disk "schimbă direcţia" 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" întreruperi 19 hex. Aceasta înseamnă că funcţionarea 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 şi 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 şi începutul rutinei. În cazul de faţă însă, startul rutinei de întrerupere nu este la începutul segmentului. Mai degrabă, acolo sunt depuşi nişte octeţi de recunoaştere, prin care se poate constata dacă este, sau nu instalat un RAM-disk. Abia după aceea vine prima, şi în acelaşi 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 aşezaţi nişte octeţi de recunoaştere, care indică un RAM-disk instalat. Figura 4 arată, ca exemplu, primii 32 de Bytes ai întreruprii 19

	Conţinut hexazecimal	Conţinut 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	.!=._O.

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

	Conţinut hexazecimal	Conţinut zecimal
00hex	00 00 E7 19 00 08 A9 00	..t...È.
08hex	D4 00 01 00 00 00 00 00	š.....
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 obţineţi, prin următoarea linie, introdusă în *config.sys*:

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

Aşa cum bine se poate vedea în cele două figuri, prezenţa unui RAM-disk este foarte uşor de stabilit. Pe când la o întrerupere 19 hex, cu RAM-disk-ul instalat, octeţii 12 până la 1C hex conţin textul "VDISKV3.3", în aceeaşi octeţi, cu RAM-disk-ul neinstalat, sunt doar semne disparate, în locul unui text inteligibil.

Chiar dacă poziţia exactă în care se găseşte textul în rutina de întrerupere diferă de la MS-DOS la MS-DOS, puteţi să fiţi siguri că un RAM-disk este instalat, deîndată ce textul "VDISK" se găseşte undeva între cei 32 de Bytes citiţi 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 asemnea, din primii 32 de Bytes ai rutinei de tratare a întreruperii 19 hex. Dacă în figura 5 urmăriţi octeţii 2C pînă la 2E, veţi descoperi valorile 00, 00 şi 01. Aceşti 3 Bytes reprezintă o adresă de 24 de biţi, deci trebuie citiţi 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, consultaţi primii Bytes ai RAM-disk-ului, adică octeţii începând cu celula de memorie 010000hex, găsiţi acolo Boot-Block-ul RAM-disk-ului. Figura 6 arată conţinutul primilor 32 de Bytes ai RAM-disk-ului.

În primul rând recunoaşteţi, cu ajutorul octeţilor de la 03 la 0A, că domeniul de memorie este efectiv ocupat de RAM-disk, pentru că aici se găseşte, din nou, şirul de caractere"VDISK3.3".

În octeţii 0B pînă la 1D, urmează blocul de parametri BIOS. În acesta sunt informaţii despre mărimea sectorului, mărimea cluster-ului, numărul de FAT-uri ş.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, foloseşte un sector pe cluster, conţine o tabelă FAT, permite maximum 64 de intrări în directorul principal şi deţine o capacitate totală de 1024 de sectoare, adică 128 Kbytes.

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

Pe scurt, puteţi 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 stabiliţi mai întâi dacă este instalat un RAM-disk, căutând în primii 32 de Bytes ai rutinei întreruperii 19 hex, şirul de caratere "VDISK". Dacă găsiţi acest şir de caractere, atunci, în următorul pas, căutaţi dacă RAM-disk-ul începe în *Extended Memory*.

	Conţinut hexazecimal	Conţinut 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 04	......Ç.

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

Dacă este aşa, citiţi mărimea din Byte-ul 1E şi din 1Fhex, însumaţi domeniul pe care îl doriţi rezervat pentru programul dumneavoastră, şi salvaţi noua valoare, din nou în Boot-Block-ul RAM-disk-ului. În sfârşit puteţi utiliza, din memoria extinsă de deasupra RAM-disk-ului, domeniul de memorie dorit, pentru rutinele şi programele proprii.

După cum vedeţi, 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 puţină 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 - şi anume disponibilă deasupra RAM-disk-ului. Acest domeniu de memorie puteţi să-l folosiţi, după aceea, fără probleme. Oricum, înainte de aceasta, mai trebuie să vă îngrijiţi ca memoria să nu fie folosită de alte programe. Pentru aceasta, scrieţi 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 folosiţi dumneavoastră, când este apelată funcţia 88 hex.

La fel, trebuie să adaptaţi şi 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 ocupaţi un domeniu din memoria extinsă care să simuleze Boot-Block-ul unui RAM-disk adevărat. Astfel sunteţi sigur că un program următor, care vrea de asemenea să utilizeze memoria extinsă şi care pentru stabilirea mărimii memoriei foloseşte numai metoda RAM-disk-ului, nu se va suprapune din greşală 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
Limbaj: 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 functia 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 valorii*/
}


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

"exramchk.c" foloseşte funcţia 87 hex pentru a determina mărimea memoriei extinse.

```
Nume exramchk.pas
Limbaj QuickPascal si TurboPascal incepind 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 functia 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 intreruperii*)
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 memoria extinsa,');
writeln;
end
else
write ('sistemul dvs dispune de ',
cpu.ax, '  KBytes Extended Memory.');
end
end
```

"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 octeţi (500 cuvinte) din memoria convenţională în memoria extinsă.

Module fundamentale şi module standard în dBASE şi Foxpro

Vă prezentăm o aplicaţie standard, care utilizează limbajul celor două medii care dau titlul articolului. Aplicaţia aceasta nu face uz de meniurile puse la dispoziţie de dBASE IV sau Foxpro2, fiind utilă în cazurile în care separaţia dintre utilizator şi 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 aplicaţii. Aceste module vor forma apoi infrastructura noii aplicaţii.

Reflecţii 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? ... Metodiştii studiază problema în cele mai mici amănunte - ceea ce este foarte important la lucrul în echipă, dar cere disciplină şi profesionalism. Practicienii, în schimb, lucrează direct pe instrucţiuni - reinventează roata - ca apoi să constate că totul trebuie reluat de la început. Metoda prezentată aici porneşte 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 paşi:

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

Al 2-lea pas conduce spre definirea punctelor meniului-linie principal. Pasul al 4-lea îşi dă măsura importanţei când se fac modificări, şi mai ales când şi altcineva trebuie să înţeleagă programul. Aceasta deoarece, după o pauză determinată de diverse motive, trece mult timp până înţelegi din nou programul, iar programele nedocumentate ale altora sunt aproape imposibil de înţeles. Dat fiind că testarea cere mult timp, ea nefăcându-se numai separat pe module ci şi global, cu valori minime şi maxime, cu valori eronate şi apoi cu date reale, metoda îşi arată eficienţa mai ales la al acest al 5-lea pas.

Structura programului

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

În partea de iniţializări se face declararea variabilelor, deschiderea fişierului cu proceduri (detalii despre acesta vor apărea la momentul potrivit), şi poziţiona-

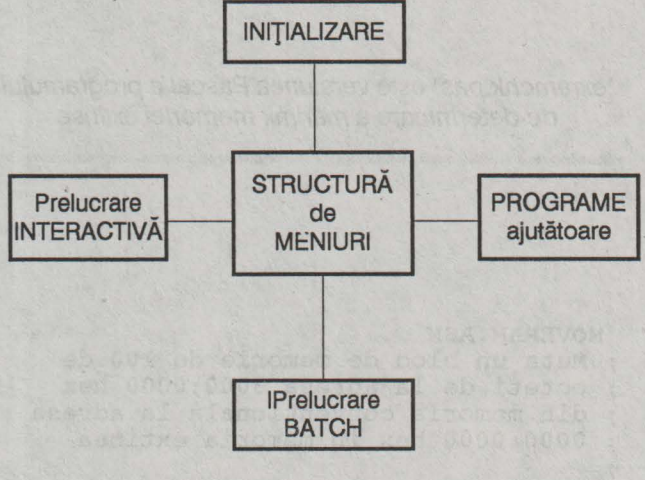


Figura 1

rea "comutatoarelor" dBASE(comenzile SET ...). Modulele : "iniţializări" şi "structura de meniuri" sunt aşa-numitele module fundamentale, iar modulele "prelucrare interactivă", "prelucrare batch" şi "programe auxiliare", sunt aşa-numitele module standard. De ce fundamentale şi de ce standard? ... Fundamentale pentru că sunt baza aplicaţiei - dacă vreţi şi în ce priveşte înţelegerea superficială a ei, şi standard deoarece creează şi servesc un adevărat standard; ceea ce este particular fiecărei aplicaţii se completează apoi, ca într-un checklist. Aţi recunoscut desigur în acest stil de lucru, pe cel al generatoarelor de programe, acestea din urmă fiind totuşi mai rigide. Avantajele conceptului prezentat aici sunt: timpi de dezvoltare şi testare reduşi, posibilităţi de eroare reduse şi o mai mare transparenţă. Două observaţii 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 înţeles, iar a doua este apropierea metodei de faţă de conceptele din domeniul CASE (*Computer Aided Software Engineering*), adică multă analiză şi 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 doreşte sau nu efectuarea unei anumite acţiuni (decizie interactivă). Este preferabil ca aici să se scrie o procedură sau o funcţie, decât să apară acelaşi modul de decizie, altfel scris în fiecare parte de program, "umflând" nejustificat codul. Apoi procedura sau funcţia este găzduită într-un fişier numit "fişier cu proceduri", care în aplicaţia 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 iniţializări - în care apar variabilele de declarat PUBLIC, se poziţionează comenzile SET şi 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 şi MAIN.PRG, se adugă modulul fundamental MASCA.PRG, care reafişează ecranul principal - la revenirea în meniu din fiecare modul aflat în legătură cu punctul de meniu respectiv. În plus, modulul MASCA.PRG poate conţine şi comenzi utile în acest caz, cum ar fi cea de închidere a tuturor fişierelor-bază de date.

Modulele standard sunt MAINxy.PRG - care conţine prelucrări în dialog cu utilizatorul şi o structură de meniuri de nivel 2, şi 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 funcţia
- două litere pentru nivelul în sistemul de meniuri.

Vom intra în câteva amănunte în ce priveşte 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 menţionat mai sus, ş.a.m.d.. A şaptea 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 şi ştergerea - cu instrucţiunea PACK de altfel mare consumatoare de timp - a articolelor marcate pentru ştergere, şi MAIN62.PRG care reface fişierele INDEX la căderi de tensiune, când sortarea în cadrul fişierelor este afectată. Aceste două module formează punctul 6 de meniu principal. Punctele de meniu principal 1-5 sunt rezervate pentru aplicaţii, iar punctul 8 este rezervat pentru gestiunea culorilor, a parametrilor şi 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 aplicaţie nouă la acest punct, apar modificări şi în MAINVARI.PRG. Variabilele locale, denumite aici parametri, vor fi gestionate în MAINPARA.PRG. Acesta are o construcţie similară cu a lui MAINVARI, dar nu se modifică la un modul proaspăt integrat.

Din cauza limitărilor concepţiei lui dBASE, respectiv a lui Foxpro, valorile pentru culori, parametri şi variabile, trebuie salvate în fişiere DBF - adică în CULOARE.DBF, PARAM.DBF şi VARIABLE.DBF, care trebuie neapărat să existe pe suportul pe care sunt celelalte module, pentru a putea lansa în execuţie întreaga aplicaţie asupra căreia ne concentrăm. VARIABLE.DBF îşi capătă structura numai într-o aplicaţie concretă.

Cu aceasta, părţile de program sunt: fişier cu proceduri, program de iniţializare, program principal, mască principală, modul interactiv, modul batch, program de ştergere a articolelor vechi, program de reindexare, gestiunea culorilor, gestiunea parametrilor şi gestiunea variabilelor. Schema completată apare în figura 2.

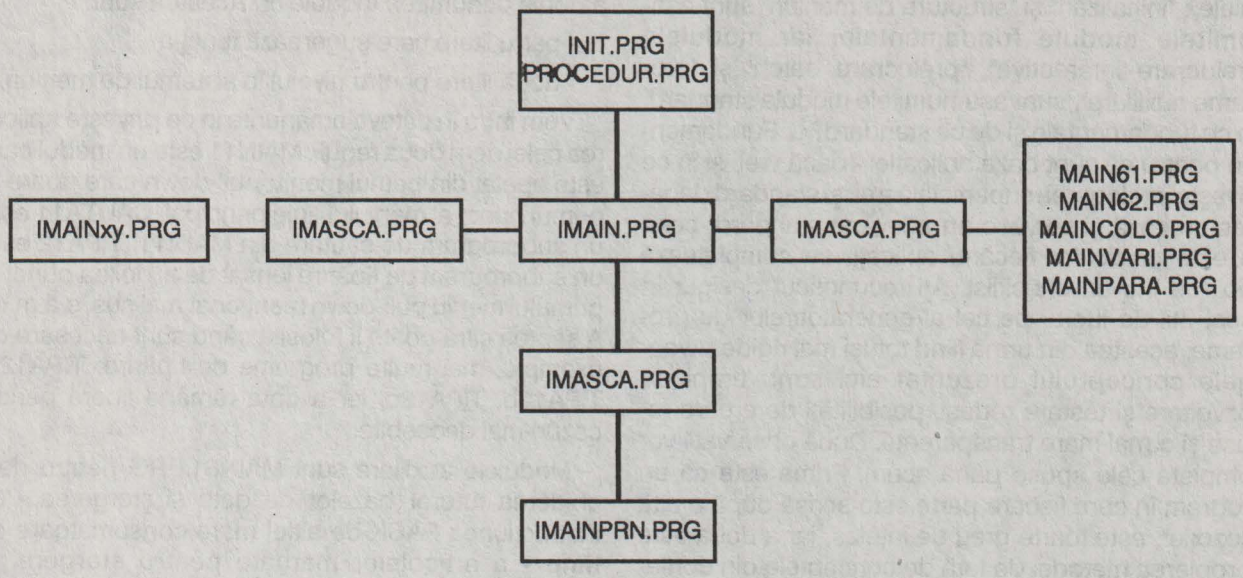


Figura 2

```
*
*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
*** Calculul valorilor coloanelor
VColoana1=0
VColoana2=VColoa-
na2+len(trim(VM_Text1))+VColoana1
VColoana3=VColoa-
na3+len(trim(VM_Text2))+VColoana2
VColoana4=VColoa-
na4+len(trim(VM_Text3))+VColoana3
VColoana5=VColoa-
na5+len(trim(VM_Text4))+VColoana4
VColoana6=VColoa-
na6+len(trim(VM_Text5))+VColoana5
VColoana7=VColoa-
na7+len(trim(VM_Text6))+VColoana6
VColoana8=VColoa-
na8+len(trim(VM_Text7))+VColoana7
VColoana9=VColoa-
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
```

```
* 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 ""
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 parametrii"
* 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.
```

```
  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 eins-
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
```

```
* 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 VNume           && Numele de cau-
tat
public VNrclie         && 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 ţin, evident, de economisirea spaţiului tipografic, nu prezentăm sursele tuturor modulelor. Cei care doresc le pot obţine de la readacţie.

Lumea culorilor VGA

(modurile în 256 culori)

Până acum am arătat cum se programează cartela VGA pentru a obţine diferite moduri în 256 culori. Am programat astfel modurile 320x240 şi 360x480. În programele care foloseau aceste moduri am folosit doar câteva culori, şi de aceea în acest articol prezentăm un program care demonstrează unele posibilităţi de a obţine 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 obţinut din modul 360x480 prin reprogramarea câtorva regiştri ai controlorului VGA, şi anume: rezoluţia pe verticală (360 linii în loc de 480), şi centrarea pe verticală a imaginii în spaţiul de afişare. Ce putem face cu acest mod în 256 culori?

Să observăm diferenţele faţă de modul 360x480:

- raportul dintre densitatea pixelilor pe orizontală şi cea pe verticală este de aproximativ 0.77 (modul 360x360) faţă 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 spaţiu de memorie nefolosit de 368 octeţi în fiecare din cele patru plane de memorie (în total 1472 octeţi).

Faţă de modul 320x240 observăm că rezoluţia este ceva mai mare, în schimb densitatea pixelilor nu este aceeaşi pe orizontală şi pe verticală. Ambele moduri pot fi folosite cu succes pentru a realiza programe de animaţie după modelul prezentat în articolul din nr. 4/1992. Modelul va fi îmbunătăţit într-un articol viitor, folosind câţiva regiştri cu funcţii speciale ai controlorului VGA.

Prezentarea unui algoritm FloodFill

Au fost prezentate până acum următoarele moduri în 256 culori: 320x240, 360x360 şi 360x480. Toate aceste moduri sunt nestandard, ele nefiind cunoscute de driver-ele uzuale (ex: EGAVGA.BGI, mouse etc). De altfel, EGAVGA.BGI nu cunoaşte nici măcar modul 320x200, singurul mod standard în 256 culori. Nu putem avea deci acces la funcţiile din biblioteca grafică lucrând în aceste moduri, şi de aceea aceste funcţii trebuie să le construim cu forţe 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 şi cum limbajul nu ar cunoaşte recursivitatea, folosind în acest scop memoria liberă aflată la dispoziţie;
- în cazul de faţă, pentru o rezoluţie de 360x360 avem nevoie de maximum 360x360x2x2 (=518400) octeţi pentru memorarea coordonatelor tuturor punctelor parcurse, în cazul în care se memorează coordonata fiecărui punct pe 4 octeţi;
- se poate folosi o metodă destul de simplă de comprimare, folosindu-se pentru memorarea coordonatelor unui punct 3 octeţi; în acest caz avem nevoie de maximum 360x360x3 (=388800) octeţi.

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;
```

```
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);
              nsetco-
lor(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;

     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;
}
```

Afişarea programelor din memorie

Sistemul de operare poate încărca în memorie, în vederea execuţiei, două tipuri de programe: din fişiere cu extensia COM şi EXE. Pentru ambele tipuri, încărcătorul de programe va alege o porţiune din memorie cu mărimea de 10 (hex) paragrafe (1 paragraf = 10 hex octeţi), numită *Prefixul Segmentului Program* - PSP, care va constitui baza memoriei alocate programului. Putem recunoaşte un PSP după instrucţiunea 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 aloca fiecărui program încărcat o porţiune de memorie numită *context*, în mărime de pînă la 32K. Adresa contextului va fi scrisă în PSP, la offset-ul 2C hex.

Ne propunem să afişăm toate programele EXE şi COM existente la un moment dat în memorie, precum şi spaţiul ocupat de fiecare.

În acest scop folosim funcţia 52H a întreruperii 21H, care ne furnizează (printre altele) şi 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).

Deţinătorul blocului poate fi un alt bloc (de exemplu deţinătorul unui bloc de tip PSP este blocul de cod al programului respectiv). Dacă blocul nu are deţinător, în această poziţie apare 0.

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

Spaţiul rezervat conţine, în cazul blocurilor fără deţinător, chiar numele programului aflat în memorie. O altă metodă pentru aflarea numelui programului este de a căuta în blocul context deţinut de acel program, dar nu dă întotdeauna rezultate.

Propun în continuare un program bazat pe informaţiile de mai sus. În plus, se foloseşte întreruperea BIOS 12H pentru aflarea memoriei convenţionale totale. Pentru o funcţionare 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 interfaţa VGA există multe programe şi jocuri care expun pe ecran o adevărată demonstraţie de culori. Însă dacă aveţi 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 nuanţe de gri. Părţile roşii şi albastre ale imaginii rămân invizibile.

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

AH = 10 hex

AL = 1B hex.

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

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

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

debug < vga2gri.deb > vga2gri.lst

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

În program veţi găsi, după instrucţiunea de salt la partea de instalare, şase instrucţiuni NOP. Ele servesc la rezervarea spaţiului în care se vor salva vectorii vechilor întreruperi.

Helmut Vogler

Traducere şi adaptare:

ing. Mihai Beer

```
' Program: vga2gri.deb
' Functi: modificarea paletei 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
```

Noutăţi

Efecte speciale

Aplicaţii

Fundamente

Hardware

Memoria extinsă

Baze de date

Laborator

micro ATCI S.R.L.

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

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

Windows 3.1

Cum este alcătuită cartea

Introducere

Partea introductivă oferă informaţii de bază despre Windows: o privire de ansamblu asupra funcţiilor, a hardware-ului necesar, precum şi informaţii de culise asupra memoriei.

Esenţialul în Windows

Această parte conţine funcţiile cele mai importante şi interesante, prezentate prin exemple şi numeroase ilustraţii. Veţi învăţa operarea de bază şi funcţiile avansate ale programului.

Teme care ne conduc mai departe

Acest capitol se concentrează asupra unei colecţii de sfaturi practice şi trucuri privind mediul Windows.