5 Modelul program ale microprocesoarelor de ultima generaţie x86

Fiecare program la execuţie obţine anumite resurse ale microprocesorului. Aceste resurse sunt necesare pentru executarea şi păstrarea în memorie a instrucţiunilor şi datelor programului, a informaţiei despre starea curentă a programului şi a microprocesorului. Totalitatea acestor resurse reprezintă *modelul program al microprocesorului*. La început vom prezenta modelul program al microprocesoarelor Intel pe 32 biţi, apoi microprocesoarele cu extensii pe 64 biţi (AMDx86-64 şi Intel EM64T). Arhitectura IA-64 (procesoarele Itanium) nu este folosită în calculatoarele personale şi nu se studiază în cazul dat.

Schema, prezentată în figura 5.1, reprezintă modelul program al microprocesoarelor Intel pe 32 biţi (Intel Architecture (IA)-32).

Pentru fiecare grup de registre, în paranteze, se indică începând de la care model a apărut grupul dat de registre în modelul program al microprocesoarelor Intel. Dacă nu este indicat, grupul dat de registre a fost prezent în microprocesoarele i386 şi i486.

În modelul program al microprocesorului Intel intră:

- Harta memoriei fizice (Pentium IV – 2^{36} – 1 B);
- Setul de registre generali;
- Setul de registre de segment;
- Setul de registre de stare şi control;
- Setul de registre ai unităţii de calcul în virgulă mobilă (coprocesorului);
- Setul de registre destinate extensiilor MMX, reflectaţi pe registrii coprocesorului (Pentium MMX);
- Setul de registre destinate extensiilor MMX în virgulă mobilă (Pentium III);
- Stiva. Structură de memorie, accesul la care se petrece cu instrucţiuni speciale (PUSH, POP).

Setul de registre

Modelul program include câteva grupe de registre folosite de programe:

- Registre generali eax/ax/ah/al, ebx/bx/bh/bl, edx/dx/dh/dl, ecx/cx/ch/cl, ebp/bp, esi/si, edi/di, esp/sp. Aceste registre conţin, de regulă, operanzi şi/sau rezultate aferente instrucţiunilor de transfer şi prelucrare a datelor;
- Registre segment cs, ds, ss, es, fs, gs. Aceste registre sunt folosite pentru crearea diviziunilor logice în memorie numite segmente;
- Registre coprocesorului st(0), st(l), st(2), st(3), st(4), st(5), st(6), st(7). Sunt folosiţi în calcule în virgulă mobilă;
- Registre extensiilor MMX (date împachetate, întregi) mmx0, mmx1, mmx2, mmx3, mmx4, mmx5, mmx6, mmx7;
- Registre extensiilor XMM (date împachetate, în virgulă mobilă) xmm0, xmm1, xmm2, xmm3, xmm4, xmm5, xmm6, xmm7;
- Registre de stare şi control – conţin informaţia despre starea microprocesorului, programului în execuţie;
- Registrul de fanioane eflags/flags;
- Registrul indicator de instrucţiuni;
- Registre de sistem – dedicate controlului modurilor de funcţionare microprocesorului.

Conponenţa setului registrelor de sistem diferă la diferite microprocesoare şi din această cauză nu sunt concretizaţi în figura 5.1.

Registrii notaţi în figură, de exemplu al, ah, ax , indică registre pe 8 şi 16 biţi ale registrului extins eax pe 32 biţi (prefix „e” (Extended)).

Grupele de registre de uz general, segment, registrul indicator de instrucţiuni, registrul de fanioane ale Pentium IV au destinaţii identice cu registrele respective ale i8086, diferând numai prin lungimea lor.

Grupul de registre segment a fost extins cu două registre de date „gs” şi „fs”.

În registrul de fanioane au fost introduşi indicatori noi (tabelul 5.1).

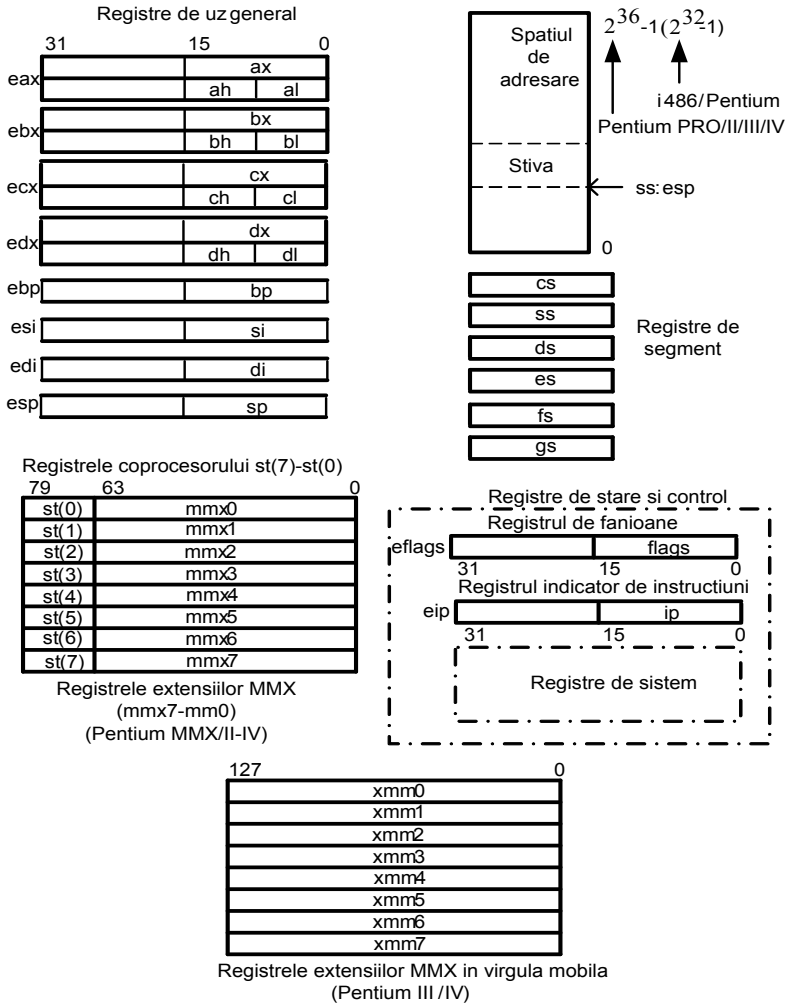

Figura 5.1 Modelul program IA-32

Tabelul 5.1

RF	Fanionul reluare (Resume Flag)	16	Se foloseşte la prelucrarea întreruperilor provenite de la registrul de reglare
VM	Fanionul modului virtual 8086 (Virtual 8086 Mode)	17	Indicatorul funcţionării microprocesorului în mod virtual 8086: 1 —microprocesorul funcţionează în modul virtual 8086 0 — microprocesorul funcţionează în modul real
AC	Fanionul alinierii (Alignment Check)	18	Destinat controlului alinierii la adresarea memoriei
VIF	Fanionul Întreruperii Virtuale (Virtual Interrupt Flag)	19	În modul virtual este analogic fanionului if. Fanionul vif se foloseşte împreună cu fanionul vip. A fost introdus în Pentium.
VIP	Fanionul Întreruperii Virtuale amânate (Virtual Interrupt Pending flag)	20	Se setează în 1 pentru indicarea întreruperii amânate. Se foloseşte în modul V împreună cu fanionul vif. Introdus la Pentium.

ID	Fanionul identificării (IDentification flag)	21	Indică susţinerea instrucţiunii cpuid.

Modelul program ale microprocesoarelor x86-64

Extensiile x86-64 şi EM64T sunt destinate pentru majorarea spaţiului de memorie adresabilă: registrele pe 64 biţi permit adresarea până la 2^{64}=18,4 x 10^{18} Byte. Acest spaţiu este limita de adresare a microprocesoarelor pe 64 biţi, dar practic sunt folosiţi numai 48 de biţi, cei mai puţin semnificativi.

Microprocesoarele oferă un sistem de privilegii pe 4 nivele destinat protecţiei memoriei, spaţiului de intrare/ieşire şi întreruperilor, şi un mecanism destinat comutării sarcinilor pentru sistemele de operare *multitask.* Setul de instrucţiuni conţine toate instrucţiunile microprocesoarelor precedente x86 şi este extins cu noi instrucţiuni. Cu extensia setului de instrucţiuni se extinde şi setul de registre arhitecturali (MMX, XMM, noi registre de uz general pe 64 biţi).

Microprocesoarele funcţionează în diferite moduri, ce determină mecanismele de protecţie şi de adresare a memoriei: modul real 8086 (pe 16 biţi), modul virtual 8086 (V8086), modul protejat pe 32 biţi (inclusiv protejat pe 16 biţi). Modul de funcţionare a microprocesorului este impus de sistemul de operare în conformitate cu modul definit de aplicaţii (task-uri). În microprocesoarele pe 64 biţi au fost introduse noi moduri de funcţionare, între care sunt moduri de compatibilitate cu sistemele operaţionale şi aplicaţii pe 32 biţi. Modurile noi introduse sunt folosite numai de sistemele de operare pe 64 biţi.

Modurile de funcţionare a microprocesoarelor

Microprocesoarele pe 32 biţi pot funcţiona în unul din următoarele moduri:

- Modul de adresare reală (real address mode), sau modul real (real mode), compatibil cu modul 8086. În acest mod este posibilă adresarea până la 1 MByte de memorie fizică.
- Modul virtual protejat de adresare (protected virtual address mode), sau modul protejat (protected mode). În acest mod microprocesorul conectează mecanismele de segmentare şi paginare. Mecanismul segmentării permite accesarea la un spaţiu de memorie virtuală până la 64 TByte. Practic este folosit numai mecanismul paginării care asigură accesul fiecărei sarcini (task) la 4 GByte memorie virtuală. Implicit, adresele şi operanzii sunt pe 32 biţi. În modul protejat microprocesorul poate executa instrucţiuni, inaccesibile în modul real, un şir de instrucţiuni de transfer a controlului, prelucrare a întreruperilor sunt executate în mod diferit de modul real.
- Modul virtual 8086 (Virtual 8086 Mode, V86) este un mod deosebit al modului protejat, în care microprocesorul funcţionează ca 8086 (adresele şi datele pe 16 biţi). Pe un microprocesor, în acest mod, pot fi executate în paralel câteva sarcini cu resurse izolate. Totodată adresarea spaţiului fizic de memorie este gestionat de mecanismele de segmentare şi paginare. Încercările de a executa instrucţiuni nepermise, ieşirea după limitele spaţiului de memorie şi a spaţiului de intrare/ieşire permis sunt controlate de sistemul de protecţie. Mai efectiv este modul virtual extins 8086 (Enhanced Virtual 8086 Mode, EV86) în care virtualizarea întreruperilor este optimizată.
- În modul de gestiune a sistemului (System Management Mode, SMM) microprocesorul accesează un spaţiu de memorie izolat de alte moduri. Acest mod este folosit în scopuri de control şi reglare. De exemplu, invizibil sunt executate funcţiile de control asupra alimentării, se emulează adresările la dispozitive inexistente (emularea tastaturii şi mouse-lui PS/2 pentru USB).

Pentru microprocesoarele x86-64 modurile enunţate sunt grupate şi grupa are denumire *legacy mode*, dar a mai fost introdus un nou mod *long mode* cu două submoduri:

- Modul pe 64 biţi (64-bit mode) – acest mod susţine adresarea virtuală pe 64 biţi şi extensiile registrelor pe 64 biţi. În acest mod este folosit numai *modelul plat de memorie* (un segment comun pentru cod, date şi stivă). Implicit adresa este pe 64 biţi, a operanzilor – 32 biţi, dar cu prefixul (REX) pot fi definiţi operanzi pe 64 biţi. A fost introdus un nou mod de adresare – relativ la indicatorul de instrucţiuni. Acest mod este folosit de sistemele de operare (SO) pe 64 biţi la lansarea aplicaţiilor pe 64 biţi – este setat de SO pentru segmentul de cod a unei sarcini;
- Modul de compatibilitate (compatibility mode) permite SO să execute aplicaţii pe 32 şi 16 biţi.

Pentru aplicaţii microprocesorul reprezintă un microprocesor pe 32 biţi cu toate atributele modului protejat, cu mecanismele de segmentare şi paginare. Proprietăţile pe 64 biţi sunt folosite numai de SO ce se reflectează în procedurile de translatare a adreselor, de prelucrare a excepţiilor şi întreruperilor. Modul este setat de SO pentru segmentul de cod ale unei sarcini concrete.

SO pe 32 biţi folosesc microprocesoarele x86-64 numai în modul *legacy mode* (ca un microprocesor IA-32).

Registre de arhitectură şi tipuri de date

Operanzii folosiţi de microprocesoare operează cu date de diferite tipuri:

- Numere întregi (cu sau fără semn) mărime de Byte, cuvânt (16 biţi), cuvânt dublu (DWord, 32 biţi), cuvânt din 4 Byte (QWord, 64 biţi), cuvânt din10 Byte (TB, 80 biţi) şi cuvânt din 16 Byte (DQWord, 128 biţi);
- Biţi, câmpuri de biţi, şiruri de biţi;
- Numere în virgulă mobilă (FP) mărime 32, 64, 80 biţi.

Posibilitatea de a folosi operanzi lungi (64 şi 128 biţi) a apărut în microprocesoarele cu extensiile MMX şi XMM. Maximal de rapid microprocesorul procesează datele ce se află în registrele sale interne. Componenţa registrelor la care au acces aplicaţiile este prezentată în figura 5.2.

Notaţiile registrelor pe 64 biţi încep cu litera R.

Microprocesoarele includ un set de registre de sistem ce nu sunt folosite de aplicaţii (în figură nu sunt prezentaţi). La ei se referă registrele de adrese de sistem, registrele de control, registrele pentru reglări şi testări. Un şir din aceste registre sunt specifice (Model-Specific Registers, MSR), ce sunt destinate controlului reglărilor, monitorizării productivităţii, controlului efectuat de microprocesor, gestionarea memoriei *cache* ş.a. Componenţa lor diferă la diferite microprocesoare, accesul este privilegiat.

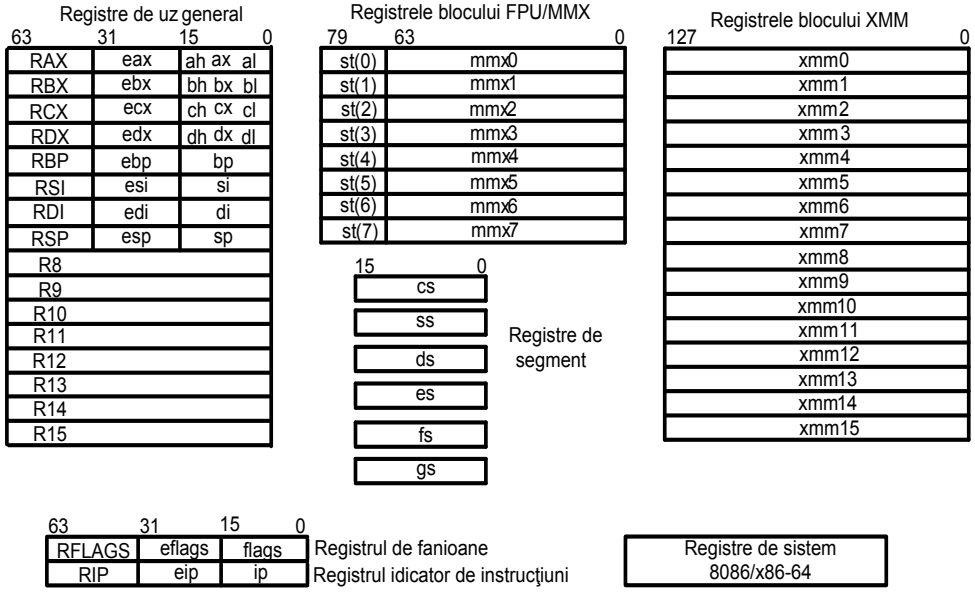

Figura 5.2 Registrele microprocesoarelor x86-64

Registrele de uz general

Structura şi destinaţia registrelor pe 32 biţi este analogică IA-32. Au fost adăugate 8 registre de uz general (R8...R15), adresarea a fost unificată: adresarea la fiecare din 16 registre se petrece ca la un registru pe 64-, 32-, 16- sau 8 biţi (se folosesc numai biţii inferiori).

RIP, în varianta pe 64 biţi, poate fi utilizat la adresarea relativă a datelor.

Registre FPU

Registrele blocului FPU sunt destinaţi executării operaţiilor aritmetice, calculului principalelor funcţii matematice (trigonometrice, exponente, logaritmi) ş.a. În diferite generaţii de microprocesoare acest bloc era numit FPU (Floating Point Unit- unitatea de prelucrare în virgulă mobilă) , NPX (Numeric Processor extension – extensie numerică a procesorului) sau coprocesor. Coprocesorul prelucrează 7 tipuri de date: numere întregi pe 16, 32, 64 biţi; numere în virgulă mobilă pe 32, 64, 80 biţi (format FP) şi 18 numere în cod binar-zecimal (BCD). Aplicarea coprocesorului accelerează

considerabil viteza de calcul.

Blocul MMX şi extensiile 3DNow!

Tehnologia MMX este orientată aplicaţiilor multimedia, grafica 2D/3D şi comunicaţiilor. Avantajul principal al MMX constă în prelucrarea concomitentă a unui şir de date cu o instrucţiune – conform tehnologiei SIMD (Single Instruction — Multiple Data). Tehnologia MMX prelucrează noi tipuri de date împachetate în registre pe 64 biţi: 8 Byte, 4 cuvinte, 2 cuvinte duble, cuvânt pe 64 biţi. Aceste date sunt prelucrate în registrele MMX0-MMX7, care reprezintă 64 biţi inferiori ai registrelor FPU pe 80 biţi.

Tehnologia 3DNow!

Tehnologia 3DNow! (21 instrucţiuni) a fost implementată de firma AMD în microprocesoarele sale K6-2, pentru extinderea capacităţilor blocului MMX. Această tehnologie prelucrează date în format FP – 2 cuvinte pe 32 biţi cu o precizie unitară şi date împachetate (8 Byte, 4 cuvinte, 2 cuvinte duble, cuvânt pe 64 biţi). În microprocesoarele Athlon setul de instrucţiuni a fost extins cu 24 instrucţiuni. Setul extins este notat - *3DNow!E.* Noile instrucţiuni sunt destinate procesoarelor DSP (12 instrucţiuni), ce prelucrează date în format FP, sa extins setul de instrucţiuni MMX (12 instrucţiuni numere întregi), extinsă gestiunea memoriei cache (7 instrucţiuni destinate accelerării transferului de date). Unele instrucţiuni coincid cu instrucţiunile din setul SSE. Setul *3DNow! Professional* (72 instrucţiuni), introdus în microprocesoarele Sempron, este compatibil cu setul SSE şi este notat *3DNow!P*.
În microprocesoarele Intel extensiile 3DNow! nu sunt utilizate.

Blocul XMM şi extensiile SSE

Extensiile SSE (Streaming SIMD Extensions) sunt destinate accelerării prelucrării fluxurilor mari de date în virgulă mobilă (FP) şi numere întregi. Extensiile fluxurilor de date sunt realizate pe baza registrelor blocului XMM (8 registre pe 128 biţi la microprocesoarele pe 32 biţi, 16 - la microprocesoarele x86-64).

Registrele XMM prelucrează următoarele tipuri de date:

- 4 numere în virgulă mobilă cu precizie unitară (32 biţi);
- 2 numere în virgulă mobilă cu precizie dublă (64 biţi);
- Numere întregi: 16 Byte, 8 cuvinte, 4 cuvinte duble sau 2 cuvinte pe 64biţi (numai în SSE2)

În procesoarele Pentium III este aplicat setul SSE, în Pentium 4 – SSE2 destinat pentru prelucrarea datelor în grafica 3D, codificarea/decodificarea video, cifrarea datelor. Apoi a fost aplicată şi extensia SSE3, iar în procesaorele moderne şi SSE4.2. Blocul XMM şi seturile SSE1, SSE2, SSE3 sunt aplicate şi în microprocesoarele AMD.

Setul de instrucţiuni

Setul de instrucţiuni ale microprocesoarelor x86 moderne, conţine toate instrucţiunile ale microprocesoarelor din generaţiile precedente. Instrucţiunile pot fi divizate în instrucţiuni de uz general, ce sunt folosite de diferite aplicaţii şi de sistem, ce sunt folosite de sistemul operaţional pentru crearea mediului de funcţionare a aplicaţiilor. În figura 5.3 sunt prezentate grupele de instrucţiuni ale microprocesoarelor x86 în ordinea apariţiei lor în microprocesoarele din diferite generaţii şi modele.

Instrucţiunile de uz general - instrucţiuni x86 pentru prelucrarea numerelor întregi şi sunt aplicate practic de toate programele. *Instrucţiunile modurilor pe 64 biţi* (long mode instructions) au apărut în microprocesoarele pe 64 biţi şi sunt accesibile numai în modurile respective.
Instrucţiunile în virgulă mobilă x86 funcţionează cu blocul FPU şi prelucrează date în virgulă mobilă: 80 biţi – precizie extinsă, 64 – precizie dublă, 32 – precizie unitară. Suplimentar susţin formatul BCD.

Media instrucţiuni pe 64 biţi prelucrează datele ce se află în registrele MMX pe 64 biţi. Ele operează cu date întregi şi în virgulă mobilă, sunt destinate media aplicaţiilor ce prelucrează blocuri de date. Au apărut cu implementarea registrelor MMX, setul a fost extins cu instrucţiuni AMD *Extension toMMX*

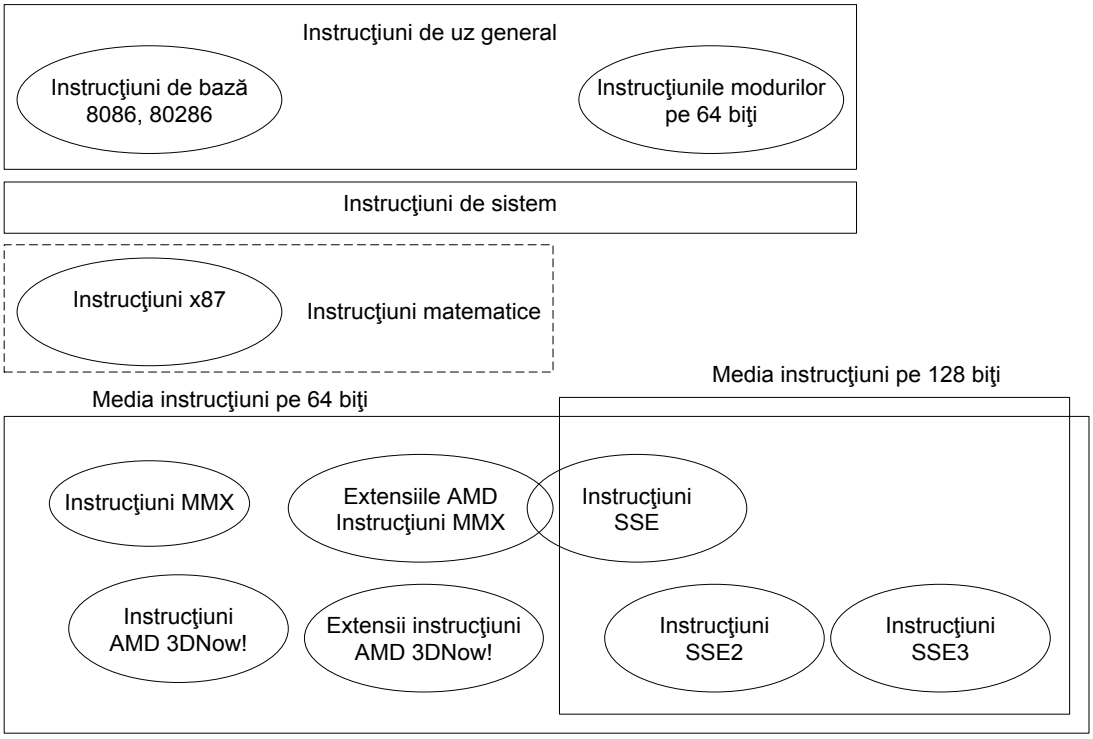

Figura 5.3 Setul de instrucţiuni ale microprocesoarelor x86

şi parţial cu instrucţiuni SSE, introduse de Intel. Instrucţiuni în virgulă mobilă au apărut în 3DNow!, setul a fost extins cu instrucţiuni *AMD Extensions to 3DNow! Instructions.*

Media instrucţiuni pe 128 biţi prelucrează datele ce se află în registrele XMM pe 128 biţi. Aceste instrucţiuni operează cu date întregi şi în virgulă mobilă, sunt destinate media aplicaţiilor ce prelucrează mari blocuri de date. În setul SSE sunt definite instrucţiuni în virgulă mobilă cu precizie unitară (32 biţi), în SSE2 au fost introduse instrucţiuni în virgulă mobilă cu precizie dublă (64 biţi), în SSE3 introduse 13 instrucţiuni (la setul din 144 instrucţiuni din SSE şi SSE2). Un şir de instrucţiuni reprezintă punţi între grupele de instrucţiuni susnumite. Ele asigură transferuri de date între blocurile MMX, XMM şi registre de uz general. Un şir de instrucţiuni pot fi executate şi în blocul MMX (şi FPU) şi în blocul XMM.

6 Gestionarea memoriei

6.1 Memoria virtuală

Vom trece pe scurt în revistă câteva dintre principiile mai importante care caracterizează evoluţia microprocesoarelor.

Un prim principiu porneşte de la segmentarea memoriei microcalculatorului în subdiviziuni logice, aşa cum procedează Intel 8086.

Există deci două modalităţi principial diferite de a trata memoria:

- *Adresarea liniară,* care presupune accesul în memorie în mod continuu de la adresa 0 la adresa 2^{n-1}, unde "n" este numărul de biţi ai unei adrese (sau, fizic, numărul de linii ai magistralei de adrese).
- *Adresarea segmentată,* în care memoria este divizată logic în porţiuni numite segmente, în interiorul cărora adresarea este liniară. Orice localizare în memorie se face relativ la baza segmentului iar fixarea fizică a bazei unui segment este transparentă utilizatorului. Am prezentat deja, pentru 8086, mecanismul segmentelor de dimesiune fixă, noţiunea de *adresă logică* precum şi translatarea ei în adresă fizică.

O generalizare interesantă este aceea de a forma segmente de dimensiuni variabile. Aceasta permite o mai bună adaptare a subdiviziunilor logice ale memoriei la dimensiunile programelor sau ale structurilor de date.

Un alt concept este acela al formării unei memorii virtuale.

Conceptul de memorie virtuală se bazează pe ideea separării memoriei logice a utilizatorului de cea fizică şi extinderea memoriei logice prin stocare imaginii sale pe hard disc (fişierul pagefile.sys). Fiecare program la execuţie, nu este obligatoriu să fie stocat întreg în memoria RAM, ci doar o secvenţă de cod şi date, executate la un moment dat. Apare, în mod firesc, ideea de a extinde spaţiul de memorie logică, care poate fi realizat cu mult mai mare decât memoria fizică. Acesta este principiul de bază al organizării memoriei virtuale.

Legat de conceptul memoriei virtuale apare mecanismul de gestionare a memoriei. El constă în translatarea adreselor virtuale (folosite de programator) în adrese fizice identificabile în structura fizică, concretă, a microcalculatorului.

Un alt concept este acela de multiprocesare (concurenţă, multiprogramare, "multitasking") care se referă la capacitatea calculatorului de a executa mai multe procese ("task-uri") simultan aceasta se realizează prin comutarea, secvenţială, de la o secvenţă de instrucţiuni a unui proces (task) la altă secvenţă aparţinând altui proces şi aşa mai departe, apoi se comută din nou la prima secvenţă etc. Tot acest mecanism este transparent utilizatorului care are impresia desfaşurării *simultane* a mai multor procese (aceasta poate însemna, eventual, mai mulţi utilizatori, sau un singur utilizator care are nevoie de mai multe procese simultan). Ideea nu trebuie să surprindă: este firesc ca într-un sistem cu o memorie virtuală de mari dimensiuni (la *Pentium* ajunge la 64TB), organizată logic într-o multitudine de segmente, să fie loc pentru mai multe procese.

Precizăm că noţiunea de proces (task) folosită aici se referă la o secvenţă de acţiuni coerente (eventual organizate ca programe de sine stătătoare) care duc la îndeplinirea unui scop, folosind resursele calculatorului: memorie, timp de procesor, memorie externă etc.

O consecinţă importantă a gestionării memoriei este mecanismul protecţiei. Legătura este atât de firească încât modul de lucru "virtual" al microprocesoarelor Intel se mai numeşte şi "protejat" (subliniem că, din motive de compatibilitate, toţi membrii familiei Intel au şi un mod "real" de funcţionare similar cu funcţionarea lui 8086). Protecţia are, pentru toate microprocesoarele evoluate, trei aspecte de bază:

- Controlul informaţiei (coduri sau diverse tipuri de date);
- Izolarea utilizatorilor unul faţă de altul (protecţia "inter-task");
- Izolarea software-ului de sistem de cel de aplicaţii (protecţia "intra-task").

Se impun aici câteva observaţii strict necesare:

- Componentele esenţiale ale mecanismului gestionării memoriei şi implicit ale protecţiei sunt cele de segment şi proces (task), în accepţiunile definiţiilor date anterior. De altfel din punctul de vedere al mecanismului protecţiei, segmentul se redefineşte ca fiind cea mai mică regiune (logică) de memorie cu atribute de protecţie precizate.

- Cele trei aspecte de bază ale protecţiei nu pot fi asigurate satisfacător numai prin mecanismul gestionării memoriei. De aceea multe familii de microprocesoare evoluate adaugă un mecanism suplimentar: privilegiile multi-nivel. Acesta asigură izolarea software-ului de sistem de cel de aplicaţii

prin ierarhizarea pe mai multe nivele de privilegii (patru pentru familia Intel) a programelor (coduri şi date) şi a proceselor.

Segmentarea memoriei şi multiprocesarea încurajează un procedeu de a rula procese ce au nevoie de resurse importante de memorie, în memorii fizice cu mult mai reduse. Acest deziderat se realizează prin procedeul denumit interschimbarea locului proceselor ("swapping tasks"), care constă în trimiterea provizorie în memorie a proceselor (sau a unor părţi componente dintr-un proces) din memoria internă în cea externă, dacă acel proces (sau acea parte din el) nu este în execuţie la un moment dat. Schimbarea locului între memoria internă şi cea externă se poate face de câte ori este necesar pentru a menţine continuu în memoria internă doar strictul necesar bunei desfaşurări a unei acţiuni.

Un mecanism folosit practic de toate calculatoarele actuale este paginarea, ceea ce semnifică implementarea memoriei virtuale bazată pe blocuri de mărime fixă numite pagini. Multiprocesarea precum şi procedeul de interschimbarea locului proceselor se poate realiza cu pagini în loc de segmente. Familia Intel combină segmentarea memoriei cu paginarea.

Localizarea unei informaţii în memorie devine un mecanism complicat: între adresa virtuală şi adresa fizică se interpune un nou tip de adrese: adrese liniare; acestea, la rândul lor, sunt translatate prin mecanismul paginării în adrese fizice.

Microprocesoarele evoluate care folosesc mecanisme complicate de localizare a informaţiei în memorie (ca în cazul descris anterior: adresă virtuală→adresă liniară→adresă fizică) au nevoie şi de procedee prin care să minimizeze timpul necesar unei referiri în memorie. De regulă, metoda folosită este aceea a elementelor de structură ascunse ("cache"). Astfel Intel foloseşte registre cache care prelungesc, transparent pentru utilizator, informaţia din registrele segment, astfel încât să aibă disponibile iriformaţiile necesare formării segmentelor în memorie şi atributele lor de protecţie.

Microprocesorul PentiumIV foloseşte în plus o memorie asociativă cache TLB ("translation lookaside buffer") care conţine adresele celor mai folosite pagini şi diminuează considerabil timpul afectat mecanismului paginării.

Modul protejat Protected Mode, mai precis Protected Virtual Address Mode (modul protejat de adresare a memoriei virtuale), este modul de funcţionare de bază a microprocesoarelor pe 32 biţi. În acest mod microprocesoarele pot adresa până la 64 Tbyte memorie virtuală, care cu ajutorul mecanismului paginării poate fi translatată în 64 GByte (Pentium IV) memorie fizică. Cu apariţia microprocesoarelor pe 64 biţi volumul memoriei virtuale a fost extins până la 2^{64} Byte. Practic în primele microprocesoare pe 64 biţi lungimea adresei virtuale a fost limitată până la 48 biţi, iar adresa fizică – 52biţi.

Modul virtual 8086 — Virtual 8086 Mode sau V86 — este o stare a modului protejat, în care microprocesorul funcţionează ca 8086 având posibilitatea de a folosi adrese şi operanzi pe 32 biţi.

Modul protejat este destinat să asigure executarea a mai multe procese (sarcini) simultan, ce presupune protecţia resurselor unei sarcini de la acţiunea alteia (ca procese presupunem aplicaţiile, sarcinile sistemului operaţional).

Principala resursă protejată este memoria, în care se păstrează codurile, datele şi diferite tabeluri de sistem (de exemplu, tabelul întreruperilor), accesul la care se petrece printr-o varietate de moduri de adresare (24 moduri). E necesar de protejat şi dispozitivele utilizate în comun, accesarea la care se petrece cu ajutorul instrucţiunilor de intrare/ieşire şi întreruperi.

Memoria logic poate fi tratată ca un segment sau o mulţime de segmente de dimensiuni variabile (în mod real - dimensiuni fixe). Exceptând segmentarea, în modul protejat este posibilă (folosind mecanismul paginării) divizarea logică a memoriei pe pagini de dimensiuni fixe de 4Kbyte (2MB, 4MB), fiind posibilă poziţionarea lor în orice zonă a memoriei fizice.

Referitor la memorie deosebim 3 spaţii de adresare: logic, linear şi fizic. Combinând segmentare şi paginarea deosebim două modele de memorie:

- În modelul segmentat al memoriei, aplicaţia foloseşte câteva segmente de memorie (cod, date, stivă). În acest model aplicaţia operează cu adrese logice.
- În modelul plat al memoriei, aplicaţiei i se atribuie un singur segment. În acest model aplicaţia operează cu adrese lineare. Modelul plat de memorie este mai simplu şi mai practic în utilizare şi este folosit de sistemele de operare moderne.

Segmentarea este un mod logic de organizare a memoriei la nivel practic. Paginarea este folosită la nivel de sistem pentru gestionarea memoriei fizice. Segmentele şi paginile pot fi îndepărtate din

memoria operativă (RAM) în memoria externă (hard disk), şi la necesitate, să fie încărcate din nou în memoria operativă. În aşa mod se realizează memoria virtuală.

6.2 Memoria virtuală pentru microprocesoarele pe 32 biţi

Vom detalia aici principiul gestionării memoriei, folosind pentru exemplificare componenta familiei Intel - microprocesorul PentiumIV, ce combină segmentarea memoriei cu paginarea.

Adresa virtuală, ca şi adresa logică, are pentru utilizator două părţi componente:

- *adresa efectivă* (sau offsetul ădresei virtuale);
- *selectorul,* care înlocuieşte adresa segment.

Adresa efectivă este adresa în interiorul segmentului şi utilizatorul trebuie să o conceapă ca fiind relativă la baza unui segment predefinit. Adresa efectivă este precizată de modul de adresare aferent instrucţiunii care o utilizează.

Selectorul se află, ca şi în cazul adreselor logice, tot într-un registru segment. Noutatea constă în faptul că definirea segmentului asociat unui selector nu se mai face direct (ca în cazul microprocesorului Intel 8086) ci prin *mecanismul adresării indirecte.*

Microprocesorul are următoarele particularităţi:

- Adresa virtuală este o adresă pe 48 de biţi:

$$AV \equiv adr48$$

- Adresa efectivă este o adresă pe 32 biţi:

$$AE \equiv adr32$$

Aceasta impune dimensiunea maximă a unui segment la 2^{32} B = 4 GB.

Se foloseşte un concept nou: cuanta de informaţie într-un segment poate fi octetul (ca la predecesorii familiei) sau "pagina" definită ca fiind o subdiviziune logică de dimensiuni fixe (4kB) a memoriei virtuale. Mecanismul paginării va fi detaliat într-un subcapitol ulterior.

Selectorul are 16 biţi şi este compus din trei entităţi informaţionale:

- un câmp numit INDEX pe 13 biţi;
- un "indicator de tabelă" (TI) care ocupă 1 bit;
- un câmp de 2 biţi denumit "nivelul de privilegiu cerut" (RPL). Deci:

$$SELECTOR = INDEX \uparrow TI \uparrow RPL$$

Figura 6.1 prezintă: a) configuraţia adresei virtuale şi b) configuraţia selectorului pentru microprocesorul Intel PentiumIV.

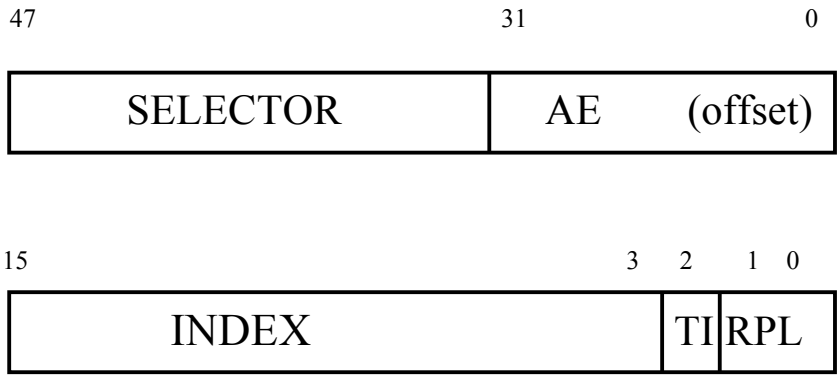

Figura 6.1

Dintre cele trei câmpuri enumerate mai sus, numai primele două folosesc pentru definirea unui segment în memorie (RPL este utilizat de mecanismul privilegiilor multi-nivel şi nu va fi explicat în acest capitol). Cei 14 biţi pe care îi ocupă INDEX şi TI conduc la posibilitatea de a forma în memorie un maximum de 2^{14} segmente = 16k segmente.

Rezultă o primă concluzie importantă: dimensiunea memoriei virtuale pe care o poate adresa Intel PentiumIV este dată de numărul maxim de segmente, fiecare presupus la dimensiunea maximă:

$$2^{14} \text{ segmente} * 2^{32} \text{ B/segment} = 2^{46} \text{ B} = 64 \text{ TB}$$

Subliniem că pentru Intel PentiumIV o adresă fizică are 36 biţi:

$$AF=adr36,$$

ceea ce înseamna că harta memoriei fizice are 2^{36} B = 64 GB. Afirmaţia făcută anterior cu privire la faptul că memoria virtuală întrece cu mult memoria fizică (internă) îşi găseşte, deci, o primă justificare.

Memoria virtuală este divizată logic în două jumătăţi:

- Pentru TI = 0 se defineşte "spaţiul adreselor globale" folosit de întregul sistem şi "împârţit" de toate procesele pentru a nu fi nevoie să se multiplice procedurile generale pentru fiecare proces în parte. Aici se află sistemul de operare, librăriile de proceduri, suportul pentru compilatoare etc.
- Pentru TI = 1 se defineşte "spaţiul adreselor locale" care cuprinde segmentele de program şi de date pentru fiecare proces în parte.

În fiecare dintre cele două zone adresabile de către un utilizator se pot forma 2^{13} segmente = 8k segmente.

Câmpul INDEX foloseşte pentru identificarea propriu-zisă a unui segment. El acţionează ca *un deplasament* într-o tabelă localizată în memorie numită tabelă de descriptori. Fiecare descriptor conţine, între altele, *adresa fizică a bazei unui segment precum şi dimensiunea sa.* Se observă deci că se utilizează adresarea indirectă cu memoria pentru identificarea unui segfment.

6.3 Generarea adresei fizice

Translatarea adresei logice în fizică pentru procesoarele pe 32 biţi este ilustrată în figura 6.2. Blocul segmentării translatează spaţiul logic de adresare în spaţiu linear de adrese pe 32 biţi. Adresa lineară se formează prin adunarea adresei de bază a segmentului şi adresei efective. Adresa de bază a segmentului în mod real se obţine prin deplasarea la stângă cu patru poziţii binare a valorii din registrul de segment (ca în 8086). În mod protejat adresa de bază se încarcă din descriptor, ce se află într-un tabel de descriptori, selectat de selectorul ce se află în registrul de segment utilizat.

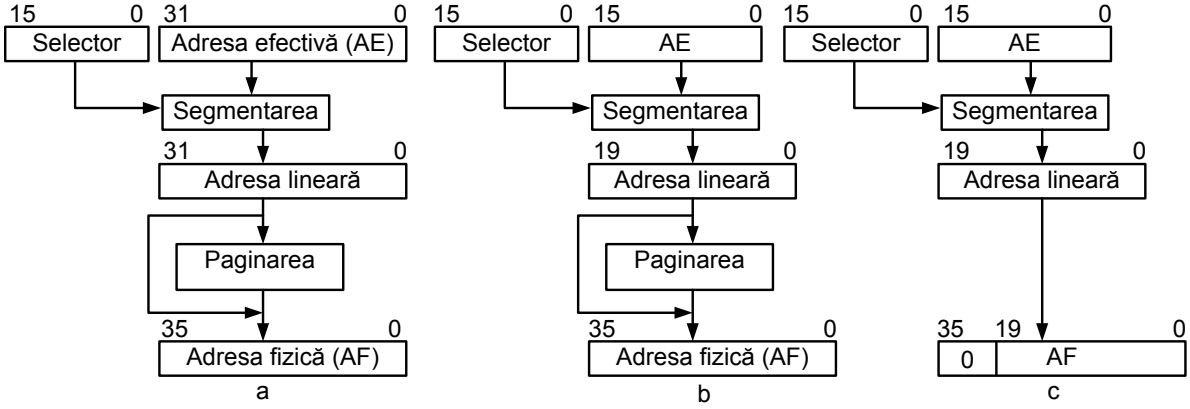

Figura 6.2 Generarea adresei fizice în microprocesoarele pe 32 biţi: a — în mod protejat, b — în mod V86, c — în mod real

Adresa fizică este formată după translatarea adresei lineare cu ajutorul unităţii de paginare. Adresa se plasează pe magistrala externă a procesorului. Când unitatea de paginare este deconectată adresa fizică coincide cu cea lineară. Unitatea de paginare activată translatează adresa lineară în cea fizică ce localizează pagini de 4 KByte (2 sau 4 MB). Unitate de paginare poate fi activată numai în modul protejat.

În modul pe 64 biţi segmentarea nu se foloseşte (figura 6.3, a): aplicaţiile operează cu adrese virtuale lineare. În microprocesoare cu extensii pe 64 biţi mecanismul segmentării este utilizat numai în modul de compatibilitate (figura 6.3, b). Părţile superioare ale acestor adrese trebuie să fie egale cu zero, în caz contrar mecanismul protecţiei va fixa excepţie. Din registrele de segment procesorul utilizează numai registrele CS, FS şi GS. În descriptorul indicat de CS sunt utilizate numai atributele: indicatorul modului pe 64 biţi, implicit mărimea operandului şi nivelul de privilegii.

Registrele FS şi GS sunt utilizaţi în noul mod de adresare: adresa de bază din descriptorul de segment poate fi utilizat ca deplasament la calculul adresei (efective, virtuale şi lineare – acum noţiuni identice). Unitatea de paginare permite utilizarea adresei fizice cu o lungime diferită de lungimea adresei lineare.

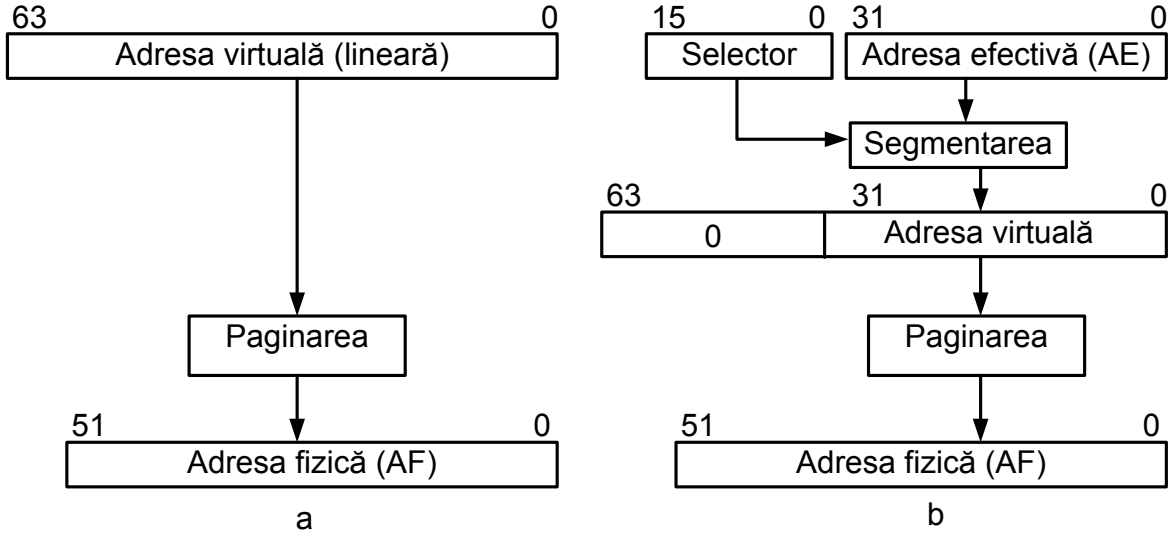

Figura 6.3 Generarea adresei fizice în microprocesoarele cu extensii pe 64 biţi:
a — în mod pe 64 biţi, b— în mod de compatibilitate

În microprocesoarele cu extensii pe 64 biţi, adresa lineară este limitată la 48 biţi, iar adresa fizică până la 52 biţi.

6.4 Descriptori şi tabele de descriptori

Descriptorul constituie elementul de bază în definirea infomiaţiilor legate de segmentarea şi gestionarea memoriei virtuale precum şi de mecanismul protecţiilor proceselor. Descriptorii sunt folosiţi pentru o coerenţă maximă în descrierea celor mai diferite entităţi informaţionale. Astfel, există următoarcle tipuri de descriptori:

a) Descriptorii segmentelor care permit identificarea segmentelor uzuale de lucru (de programe şi/sau de date).

b) Descriptorii speciali de control care sunt, la rândul lor, de mai multe feluri:

1) *Descriptorii segmentelor de sistem,* segmente care sunt folosite de microprocesor în mecanismul gestionării memoriei.

2) *Descriptorii "segmentelor de stare a proceselor"* prin care se poate implementa mecanismul multiprocesării.

3) *Descriptorii "porţilor"* (Gate) folosiţi în mai multe tipuri de acţiuni:
protecţia multi-nivel, multiprocesarea, răspunsul la cererile de întreruperi.

Pentru familia de microprocesoare Intel, forma generală a unui descriptor este reprezintată în Figura 6.4 Indiferent de tip, un descriptor are 8B (numerotaţi în figură de la 0 la 7), fiecare octet având semnificaţii precizate.

Cei 8B ai descriptorului au următoarele semnificaţii:

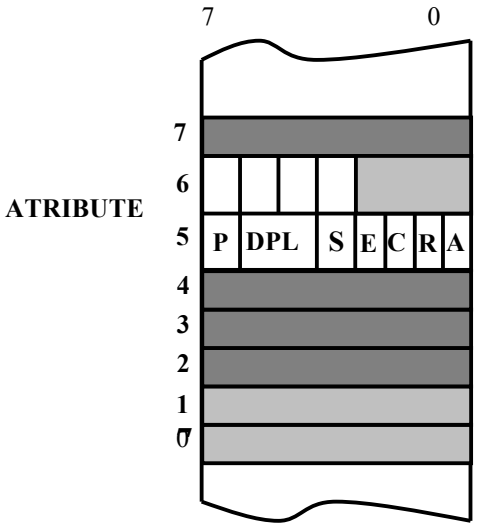

Figura 6.4

- Octeţii 0, 1 şi *nibble-ul* inferior ai octetului 6, conţin 20 biţi din LIMITA segmentului selectat.
- Octeţii 2, 3, 4 şi 7 conţin 32 biţi din BAZA (adresa fizică a bazei) segmentului selectat.
- Octetul 5 şi *nibble-ul* superior ai octetului 6, este câmpul ATRIBUTE, cu diverse informaţii despre segment pe care le vom detalia în continuare.

În ceea ce priveşte câmpul ATRIBUTE, notăm următoarele semnificaţii (începând cu octetul 5);

- Bitul 7, notat cu P, are rolul de a indica dacă segmentul este "prezent" în memoria fizică (P = 1) sau nu (P = 0).
- Biţii 6 şi 5 formează "Nivelul de privilegiu al descriptorului" (DPL) şi este folosit în mecanismul protecţiei multi-nivel.
- Bitul 4, notat cu S, este folosit, aşa cum arătam deja, pentru a distinge între segmentele normale de lucru (S=1) sau segmentele speciale de sistem (S=0).

În acest ultim caz, configuraţia *nibble-ului* inferior este 0010 (informaţie prezentată succint cu codificarea tip = 2) care indică faptul că descriptorul defineşte în spaţiul adreselor globale o LDT.
Pentru segmentele obişnuite, semnificaţia biţilor inferiori este:

- Bitul 3, notat cu E, indică un segment executabil (E=1), adică un segment de program, sau un segment pentru date (E = 0).

În funcţie de această ultimă clasificare, următorii trei biţi au semnificaţii diferite:

a) Pentru segmentele de program:

- Bitul 2, notat cu **C**, este folosit de mecanismul protecţiei multi-nivel pentru a introduce o excepţie în modul de acces al acestui tip de segment.
- Bitul 1, notat cu **R**, indică, pentru R = 1 că segmentul poate fi şi citit nu numai executat.
- Bitul 0, notat cu **A**, semnifică "accesat". Ne vom opri puţin mai în detaliu asupra acestui bit, deoarece microprocesoarele Intel îl folosesc în cadrul unui procedeu cu o aplicabilitate mult mai generală.

Procedeul poartă numele de Strategia "LRU- Least Recently Used" " ("cel mai puţin utilizat") şi ajută sistemul de operare să decidă ce segment este mai puţin utilizat pentru a fi trimis în memoria externă a calculatorului, în aşteptare. Astfel **A** este setat ori de câte ori se face referire la descriptorul care îl conţine; periodic **A** este resetat de sistemul de operare, orice resetare efectivă incrementând un contor. Segmentul al cărui contor asociat este minim va fi ales ca "victimă" pentru a aştepta în memoria externă.

b) Pentru segmentele de date:

- Bitul 2, notat acum cu ED (prescurtarea de la "expanded down"), indică dacă segmentul de date este folosit ca stivă.
  - pentru ED= 0 - segment de date;
  - pentru ED=1 - stivă.
- Bitul 1, notat acum cu W, indică, pentru W = 1, dacă segmentul de date poate fi şi "înscris" nu numai citit.
- Bitul 0, notat cu A, are aceeaşi semnificaţie ca şi pentru segmentele de program, fiind folosit în cadrul "Strategiei LRU".

Se impun câteva observaţii:

- Dimensiunea adresei fizice a bazei segmentului indică o locaţie în harta memoriei fizice de 4 GB.
- Câmpul LIMITA indică o dimensiune maximă de 2^{20} = 1 M cuante de informaţie. Numai că această cuantă poate fi octetul sau pagina (în dimensiune fixă de 4 kB).

Acest tip de organizare a memoriei este indicat de bitul 7 al octetului 6, bit notat cu G ("granularitate"). Pentru G = 1, Pentium foloseşte mecanismul paginării. Se observă că în cazul granularităţii pe octet segmentele au maximum 1 MB, iar în cazul granularităţii pe pagină, segmentele ating dimensiunea maximă de 4 GB.

Ceilalţi trei biţi din octetul 6 au următoarele semnificaţii:

- Bitul 6, notat cu D/B (D pentru date şi B pentru coduri), indică pentru D/B = 1 că se lucrează cu operanzi pe 32 de biţi, cu toate modurile de adresare care îi sunt caracteristice. Pentru D/B = 0, se lucrează cu operanzi pe 16 biţi şi cu segmente de maximum 64 kB.
- Bitul 5 este 0 pentru compatibilitate cu descendenţii familiei Intel.
- Bitul 4, notat cu AVL este la dispoziţia utilizatorului ("disponibil").

Se degajă câteva concluzii generale despre descriptorii de segment, indiferent de tipul microprocesorului:

- Segmentele de sistem de tip LDT nu pot fi în mod explicit nici citite, nici înscrise, nici executate. Ele sunt utilizate numai în mecanismul translatării în spaţiul adreselor locale. Pentru a crea o LDT, sistemul de operare creează în GDT un descriptor pentru un segment de date, înscrie informaţiile în segmentul definit cu acest descriptor, apoi schimbă tipul descriptorului prin modificarea biţilor corespunzători din ATRIBUTE.
- Informaţia conţinută în descriptor este deosebit de bogată. Astfel:
  1. Se precizează adresa fizică a bazei segmentului precum şi dimensiunea sa. Bitul ED ajută la interpretarea acestei dimensiuni.
  2. Biţii P şi A ajută sistemul de operare să decidă care segmente vor fi menţinute în memoria internă.
  3. Se pot distinge segmentele de program, de date şi speciale.
  4. Biţii R şi W definesc accesul în segmente, asigurând o primă protecţie.
  5. Câmpul DPL şi bitul C asigură privilegii suplimentare.
- Orice încălcare a unui tip de protecţie este semnalată prin verificare "hardware" cu o cerere de întrerupere dedicată.

Tabelele de descriptori guvernează interpretarea adreselor virtuale. Orice translatare a adreselor virtuale în adrese fizice face apel implicit la aceste tabele. Există trei categorii de tabele de descriptori:

1. "Tabela descriptorilor globali" (GDT) destinată să conţină informaţiile pentru o descriere completă a spaţiului adreselor globale. Să reţinem că există *o singură* tabelă de acest fel.
2. "Tabelele de descriptori locali" (LDT) conţin informaţiile din spaţiul adreselor locale folosite de unul sau mai multe procese.
3. „Tabela de descriptori de întrerupere” (IDT), folosită în modul protejat, poate conţine până la 256 întreruperi.

În mod normal, fiecare proces trebuie să aibă acces la trei tabele de descriptori care să-i definească spaţiul adreselor virtuale ce îi este destinat. În acest fel se asigură toate protecţiile necesare pentru a nu depăşi acest spaţiu dar nici pentru a nu fi afectat de alte procese. Cele trei tabele sunt: Tabela descriptorilor globali (folosită de toate procesele), o Tabelă de descriptori locali particulară (privată sau "împărţită" cu un grup de procese înrudite) şi tabela de descriptori de întreruperi (accesarea întreruperilor cu instrucţiuni INT, întreruperi Hard, excepţii ale microprocesorului).

Pentru sistemele cele mai simple, în care protecţia proceselor nu este necesară, poate fi definit doar spaţiul adreselor globale şi deci GDT este suficientă.

O tabelă poate avea 8k descriptori necesari pentru a defini un maximum de 8k segmente potenţiale. O tabelă poate avea un număr predefinit de descriptori chiar dacă la un moment dat nu sunt definite tot atâtea segmente. Acest lucru este asigurat prin rezervarea spaţiului din tabelă unor "descriptori nuli": descriptori pentru care câmpul *atribute* este 0H, ceea ce semnifică faptul că aceşti descriptori nu identifică nici-un segment în memoria virtuală.

Fiecărui tabel îi corespunde un registru al procesorului (GDTR, LDTR şi IDTR). Instrucţiunile ce încarcă registrele LGDT, LLDT şi LIDT sunt privilegiate (nivelul 0).

Sistemul de privilegii

Sistemul de privilegii pe 4 nivele este destinat gestionării instrucţiunilor privilegiate şi accesului la descriptori. Nivelele sunt numerotate 0-3, nivelul zero corespunde accesului nelimitat şi este destinat nucleului sistemului operaţional (SO). Nivelul 3, acces minimal şi practic este destinat aplicaţiilor. Serviciile oferite proceselor, sarcinilor (tasks) pot avea diferite niveluri de protecţie. Transferul controlului între sarcini este gestionat de porţi (Gate), ce controlează regulile de utilizare a nivelelor de privilegii. Utilizând porţile, sarcinile pot primi acces la serviciile segmentelor din diferite nivele.

Nivelele de privilegii sunt destinate descriptorilor, selectorilor şi sarcinilor. În registrul de fanioane sunt indicatori de privilegii de intrare/ieşire, cu ajutorul cărora este asigurat controlul accesului la instrucţiunile de intrare/ieşire şi controlul fanionului de întreruperi.

Descriptorii şi privilegiile sunt baza sistemului de protecţie: descriptorii determină structura elementelor de program, iar privilegiile determină posibilitatea de accesare la descriptori şi executarea instrucţiunilor privilegiate. Orice violare a protecţiei provoacă apariţia excepţiilor speciale care sunt prelucrate de nucleul SO.

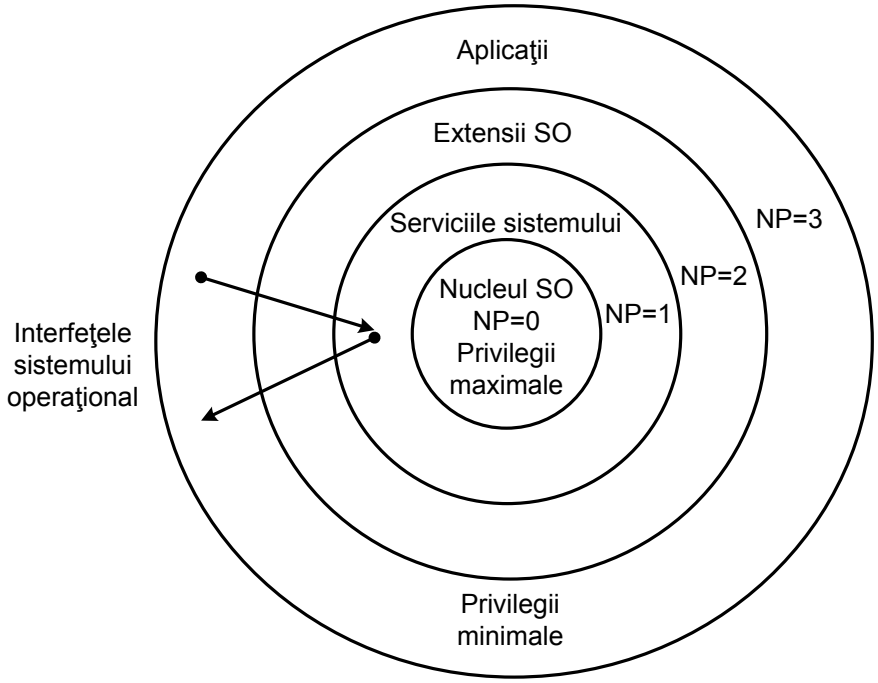

Figura 6.5 Nivelele de privilegii

Mecanismul memoriei virtuale permite fiecărei sarcini utilizarea spaţiului logic de adresare cu dimensiunea până la 64 TByte (16 K segmente de 4 GByte). Prezenţa fiecărui segment în memoria operativă, la un moment dat, este indicat de un bit în descriptorul segmentului. Segmentul neutilizat poate fi descărcat din memoria operativă în cea externă (de exemplu, harddisk), fapt ce este notificat în descriptor. Spaţiul eliberat poate fi utilizat pentru restabilirea componentelor altui segment (procesul swapping) şi în descriptor se notifică prezenţa în memorie. Când un proces adresează segmentul ce nu este prezent în memorie, procesorul formează excepţie respectivă şi segmentul (pagina) se încarcă în memoria operativă.

6.5 Translatarea adresei virtuale

Mecanismul de translatare porneşte de la cele două entităţi informaţionale din adresa virtuală (selector și adresa efectivă):

- Selectorul furnizează, la rândul său, următoarele informaţii:
  1) Prin intermediul lui TI se identifică tipul de tabelă.
  2) Câmpul INDEX este multiplicat cu 8 (un descriptor are 8B) şi se obţine poziţia relativă faţă de baza tabelei, identificându-se astfel un descriptor din tabelă.

Din descriptor se extrage:

a) Adresa fizică a bazei segmentului ("BAZA") pe 4 octeţi.

b) Dimensiunea segmentului vizat ("LIMITA") pe 20 de biţi.

- Adresa efectivă se adună la BAZA obţinută indirect cu ajutorul selectorului (aşa cum arătam mai sus) obţinându-se astfel adresa fizică (AF) a informaţiei din segmentul vizat. Concomitent se verifică dacă dimensiunea prestabilită a segmentului nu este depăşită:

$$AE \leq LIMITA$$

Mecanismul translatării adresei virtuale (TI=1) este dat în Figura 6.6.

Stabilirea bazei tabelei de descriptori depinde de tipul de tabelă: în spaţiul adreselor globale sau locale.

În spaţiul adreselor globale, GDT are o modalitate mai simplă de definire: fiind unică, baza şi lungimea sa sunt stocate într-un registru intern al microprocesorului: "Registrul tabelei de descriptori globali" (GDTR). Accesul la acest registru este asigurat de instrucţiuni privilegiate, astfel încât un utilizator obişnuit să nu poată avea acces la redefinirea acestei tabele (reamintim că GDT este folosită de toate procesele, inclusiv de sistemul de operare).

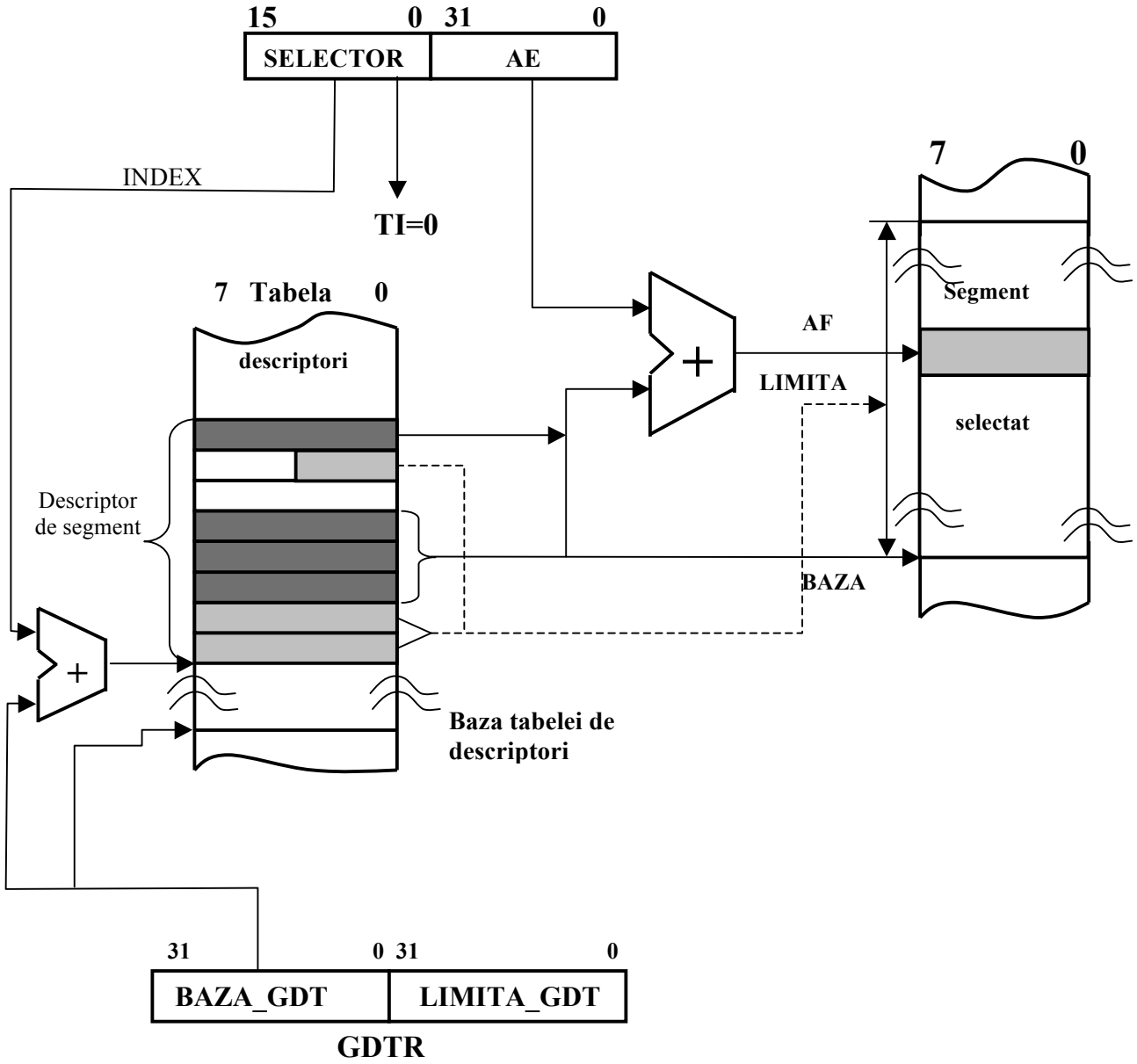

Figura 6.6

Putem nota următoarele observaţii:

- Orice LDT este privită ca un segment, dar nu unul obişnuit, ci un *segment de sistem.* Fiecare astfel de segment trebuie să se afle în spaţiul adreselor globale, deci *descriptorii pentru LDT-uri se vor afla în* GDT.
- GDT nu are nevoie de un descriptor special adresa fizică a bazei sale precum şi dimensiunea sa sunt stocate într-un registru dedicat.

6.6 Registre "cache"

Examinând modul de translatare a adreselor virtuale se poate observa că referinţele în memorie, mai ales în spaţiul adreselor locale, implică multe operaţii şi necesită un timp lung.

Microprocesorul PentiumIV aplică în acest caz o metodă de a minimiza timpul de acces în memorie.

Presupunând că referinţele în memorie se fac mai des decât schimbarea informaţiei din registrele segment şi decât schimbarea procesului activ (deci schimbarea selectorului din LDTR) s-a recurs la folosirea unor registre ascunse ("cache").

Figura 6.7 prezintă registrele cache ale microprocesorului.

Se observă că registrele segmentelor, precum şi LDTR au "o prelungire" invizibilă utilizatorului. Aici se află toate informaţiile necesare din descriptorul asociat selectorului din registrul respectiv.

Ori cate ori este încărcat un registru segment "vizibil" cu un selector, se încarcă şi cei opteţi utili

din descriptorul asociat în registru cache respectiv. Aici se află adresa fizică a bazei segmentului vizat, dimensiunea sa precum şi drepturile de acces. Deci toate informaţiile necesare unei referinţe în memorie sunt stocate în aceste registre care reprezintă o "memorie" pe cip, foarte rapid adresabilă. Adresa fizică se calculează direct prin suma dintre baza segmentului, aflată în registrul cache, şi adresa efectivă din adresa virtuală specificată conform modului de adresare. Concomitent se pot face toate verificările necesare (dacă informaţia este în interiorul dimensiunii predefinite a segmentului, dacă segmentul e prezent, ce tip de segment este vizat etc).

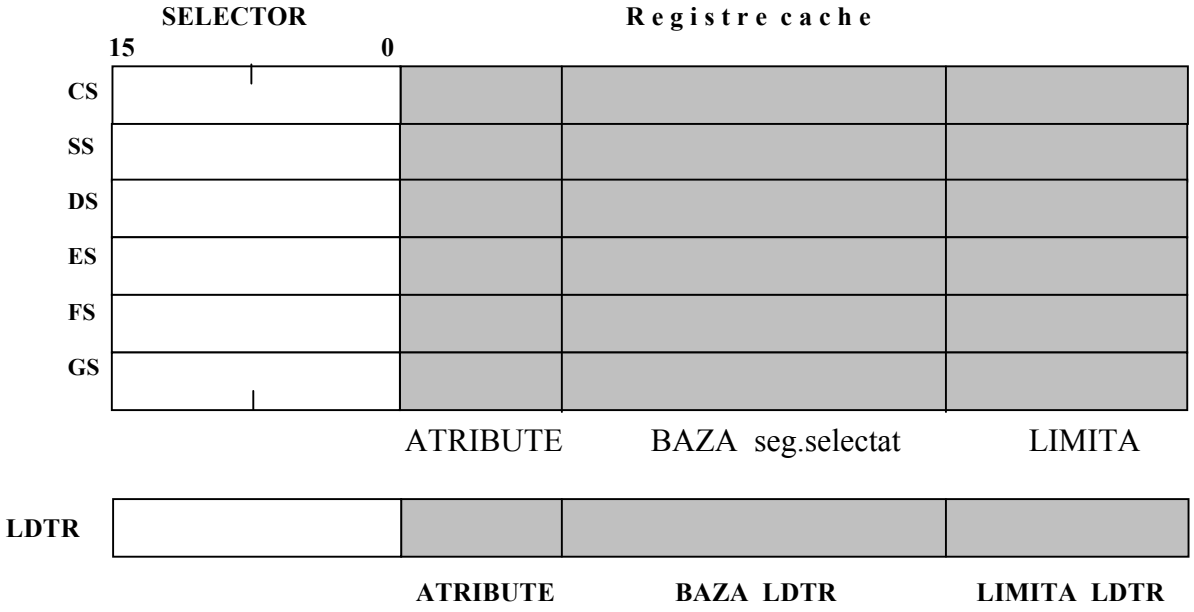

Figura 6.7

Registrele segment pot încărca un selector de 16 biţi. Partea invizibilă cuprinde 32 de biţi pentru adresa bazei segmentului selectat, 32 de biţi pentru dimensionarea segmentului (din care se folosesc numai 20 de biţi) şi un câmp pentru diversele atribute ale segmentului (prezent, nivelul de privilegiu, accesat, granularitate etc).

Din motive uşor de înţeles, GDTR nu are porţiune ascunsă. El este în întregime vizibil şi încârcarea sa se face tot cu o instrucţiune privilegiată: LGDT

Pentru LDTR situaţia este puţin diferită. Aici partea "invizibilă" conţine numai adresa bazei şi dimensiunea tabelei de descriptori locali care este vizată în memorie. În partea sa "vizibilă" există un selector care are obligatoriu TI = 0. Încărcarea acestui registru se face cu o instrucţiune privilegiată: LLDT.

Mai subliniem că nu am considerat decât registrele care iau parte la mecanismul gestionării memoriei, deşi principiul registrelor cache are o aplicabilitate mai largă.

6.7 Paginarea

Paginarea este procedeul de realizare în memorie a unor blocuri fixe numite pagini, care pot fi utilizate în mecanismul interschimbării proceselor (transferul blocurilor între memoria internă şi cea externă).

Tehnica paginării este larg folosită şi este aplicată de unitatea de gestionare a memoriei (Memory Management Unit, MMU). Paginarea utilizează directorul şi tabele de descriptori a paginilor – structuri de date în memoria fizică (operativă). Unitatea MMU divizează adresa lineară în pagini virtuale de o mărime fixă (4KB, 4MB, 2MB). Pe pagini similare este divizat şi spaţiul adreselor fizice.

Avantajele paginării pot fi rezumate astfel:

- Un obiect în memorie nu trebuie să fie continuu: pagina poate constitui o nouă *"cuantă" de* informaţie.
- Mecanismul paginării şi, implicit, întreaga tehnică de a schimba continuu blocuri fixe de informaţii întră memoria internă şi cea externă nu sunt vizibile utilizatorului.

Paginarea introduce şi o nouă noţiune legată de adrese: adresa liniară. Acest tip de adresă rezultă în urma translatării adresei virtuale şi urmează, la rândul ei, să fie translatată în adresă fizică.

Notăm următoarele observaţii preliminare:

- mecanismul paginării se aplică numai în funcţie de bitul de granularitate (G) din descriptorii de segmente.
- adresa virtuală, alcătuită din selector şi adresa efectivă, este translatată în adresa liniară şi apoi în cea fizică. Ca şi adresa fizică, adresa liniară (AL) este de 32 de biţi.
- Dimensiunea maximă a unui segment în memoria virtuală se obţine numai pentru G = 1, în acest caz un segment poate avea:

$$2^{20} \text{ pagini} * 2^{12} \text{ B/pagină} = 2^{32} \text{ B} = 4 \text{ GB}.$$

Mecanismul paginării (pagini pe 4K) are următoarele elemente esenţiale:

- Se utilizează două nivele de adresare indirectă cu memoria, astfel încât, până să se ajungă la pagina propriu-zisă, se face o referinţă într-un "director" din care se selectează o "tabelă a paginilor".
- Toate obiectele din memorie implicate (directorul şi tabelele) sunt tratate la rândul lor ca nişte *pagini speciale;* deci au toate mărimea fixă de 4 kB. Se păstrează astfel *principiul coerenţei informaţiei în memorie* (ca şi în cazul segmentelor).
- Atât directorul cât şi tabelele de pagini au structură uniformă fiind formate din 1 k "elemente" de câte 4 octeţi. Senmificaţia informaţiei din aceste elemente va fi detaliată în cadrul acestui subcapitol. Ne limităm să menţionăm acum că aici se găseşte adresa bazei tabelei şi respectiv a paginii selectate.
- Orice adresă a bazei unei tabele sau a unei pagini este dată pe 20 de biţi, aceştia constituind partea mai semnificativă a adresei fizice vizate. Toate blocurile în memorie sunt "aliniate" din 4 kB în 4 kB, deci, automat cei mai puţin semnificativi 12 biţi ai adreselor fizice ale bazelor sunt 0.
- Adresa fizică a bazei directorului se află într-unul din registrele de control ale microprocesorului CR3 (Page Directory Physical Base Address) (desigur şi aici cei mai puţin semnificativi 12 biţi ai adresei sunt 0).

Mai menţionăm că registrul de control CR2 este şi el utilizat în paginare, dar pentru verificarea corectitudinii procedeului; el conţine adresa liniară la care se detectează ultima eroare în mecanismul paginării (Page Fault Linear Address). Se generează un cod de eroare care se încarcă în stiva aferentă procesului activ.

Figura 6.8 prezintă modul în care se produce paginarea prin cele două nivele de adresare indirecta cu memoria.

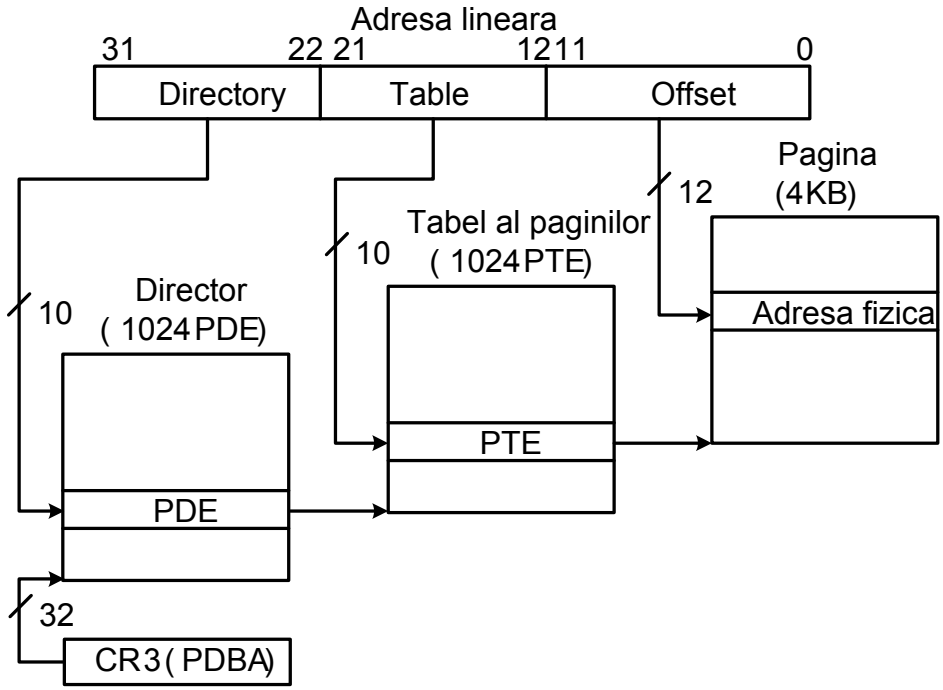

Figura 6.8

Adresa liniară cuprinde trei entităţi informaţionale:

AL=Directory ↑Table ↑Offset

În care:

a) Director (Directory) este un câmp de 10 biţi. Aceştia, concatenaţi cu 00, formează adresa relativă la bază a elementului vizat din director. Concatenarea cu 00 este normală pentru că un element are 4 B, deci informaţia este "aliniată" din 4B în 4B.

b) Tabela (Table) este un câmp de 10 biţi. Tot prin concatenare cu 00 formează adresa relativă la bază a elementului selectat dintr-o tabelă a paginilor.

c) Offset este un câmp de 12 biţi care localizează informaţia în pagina selectată. Evident şi această adresă este relativă la adresa fizică a bazei paginii.

Directorul (4KB) conţine 1024 elemente pe 32 biţi - PDE (Page Directory Entry) (figura 6.9,a). Tabela de pagini conţine tot 1024 elemente pe 32 biţi - PTE (Page Table Entry) (figura 6.9, b) şi aceste elemente conţin adresa fizică de bază (Page Frame Address) şi atributele paginilor.

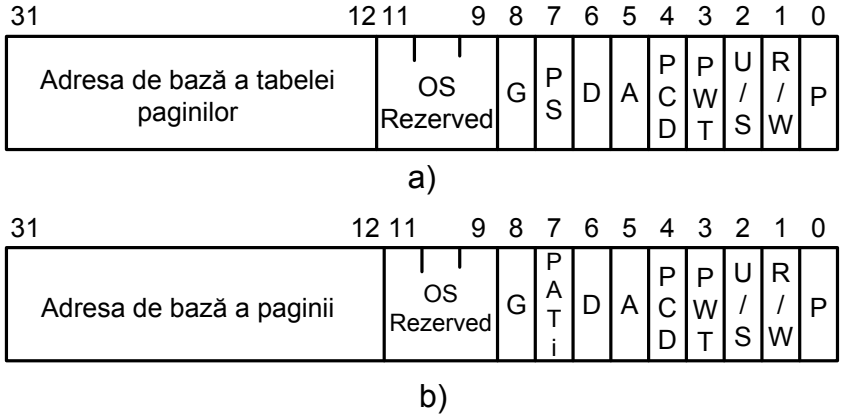

Figura 6.9 Structura elementelor paginării pe 32 biţi: a – elementul din director (PDE), b – elementul tabelei de pagini (PTE)

Detalierea câmpurilor (figura 6.9) dintr-un element (fie din director, fie dintr-o tabelă a paginilor) este următoarea, începând cu cei mai semnificativi biţi:

a) Biţii 12 ÷31 constituie cei mai semnificativi 20 de biţi ai adresei fizice a bazei unei tabele a paginilor şi respectiv a unei pagini propriu-zise.
b) Biţii 9÷11 sunt rezervaţi sistemului de operare. Se pot utiliza, de pildă, în "strategia LRU (Least Recently Used)" pentru a determina timpul cât o pagină rămâne "activă" în memona internă.
c) Biţii 7 şi 8 sunt rezervaţi pentru compatibilitate cu descendenţii familiei Intel. în cazul de faţă ei sunt obligatoriu 0.
d) Bitul 6 notat cu D ("dirty") indică, numai în cazul tabelelor paginilor, pentru D = 1, dacă a avut loc o *scriere* în pagina selectată. Acest bit nu este definit pentru elementele din director.
e) Bitul 5 notat cu A ("accesat") indică, pentru A = 1, un acces de orice fel (scriere/citire) în pagină.
f) Biţii 3 şi 4 sunt folosiţi numai de Intel 486 în adresarea memoriei cache de pe cip (PCD -"page cache disable" şi PWT - "page write through").
g) Biţii 1 şi 2 sunt utilizaţi în mecanismul protecţiei paginilor (R/W, U/S-User/Supervisor).
h) Bitul 0 notat cu P ("prezent") indică:
   1) Pentru P = 1 - pagina este prezentă în memoria internă, deci elementul poate fi folosit pentru translatarea adresei liniare în adresă fizică.
   2) Pentru P = 0 - pagina este în memoria externă; în acest caz tot restul informaţiei din element este irelevant.

Mecanismul paginării permite şi alte două observaţii interesante, care reflectă aplicarea unor principii mai generale:

a) Structura cu două nivele de adresare indirectă cu memoria pare greoaie şi consumă mult timp. Ea este însă absolut necesară. Într-adevăr, să ne imaginăm un singur nivel de adresare indirectă. în acest caz, în memoria fizică încap 1M pagini, de câte 4 kB. Tabela de adresare ar trebui să conţină 1M elemente de câte 4 B = 4 MB. Structura coerentă cu director şi tabele ale paginilor de câte 4 kB fiecare (exact ca şi paginile propriu-zise) rezolvă elegant această problemă.

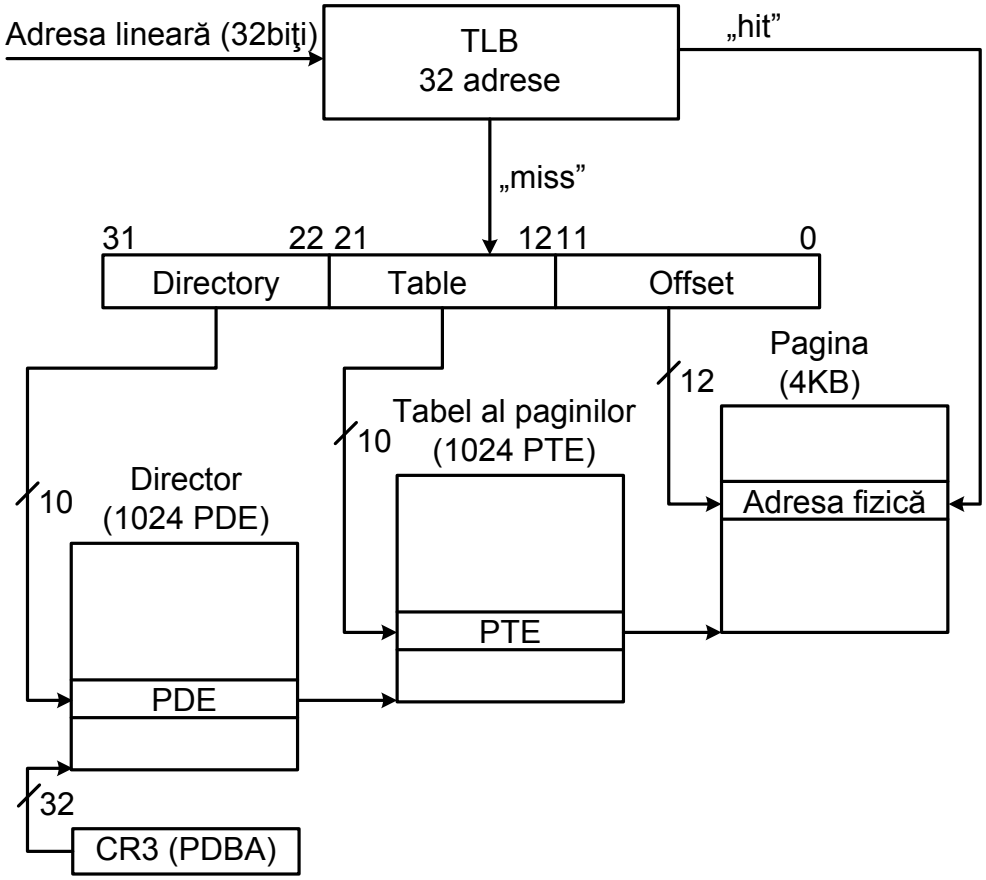

Figura 6.10

b) Problema timpului necesar pentru accesul într-o pagină din memorie se rezolvă folosind principiul general al memoriei asociative cache. Pentru a elimina stările suplimentare impuse de mecanismul paginării, unitatea care se ocupă de aceasta conţine o astfel de memorie, adresabilă prin conţinut, denumită TLB ("translation lookaside buffer"). Principiul este prezentat în figura 6.10.
Memoria conţine cele mai frecvent utilizate 32 de elemente din director şi tabelele paginilor. De câte ori este necesar accesul într-o pagină, se verifică întâi TLB. Dacă aici se găseşte elementul vizat (ceea ce se numeşte "cache hit"), translatarea se face citind adresa corespunzătoare din TLB, fără timpii adiţionali de calcul şi căutare în tabele succesive. Reactualizarea TLB se face ori de câte ori nu se găseşte elementul cerut ("cache miss"). Evident TLB este complet reînnoit când CR3 este încărcat cu o nouă adresă de bază a directotiilui. Intel ne asigură că în 98% dintre referinţele în memorie, mecanismul de translatare este înlocuit cu simpla citire a adresei din TLB.