Structura internă de bază a procesoarelor Intel

Structura de bază este cea din figura 1. Microprocesorul include două componente majore: unitatea de execuție şi unitatea de interfață cu magistrala. Structura de bază este considerată ca fiind cea a modelului x86.

Unitatea de interfață cu magistrala execută toate ciclurile de magistrală (read, write, întrerupere), fie la cererea unității de execuție, fie pentru umplerea cozii de instrucțiuni. Coada de instrucțiuni este un registru FIFO alcătuit din 6 cuvinte, instrucțiunile aşteptând aici intrarea în execuție. Ciclurile de încărcare ale instrucțiunilor sunt executate în intervalele de timp în care unitatea de execuție nu solicită magistrala. Dacă unitatea de execuție nu solicită magistrala şi coada de instrucțiuni este plină, atunci au loc cicluri inactive pe magistrală. Execuția instrucțiunilor de salt duce la resetarea cozii, deoarece trebuie extrase instrucțiuni din altă zonă de memorie.

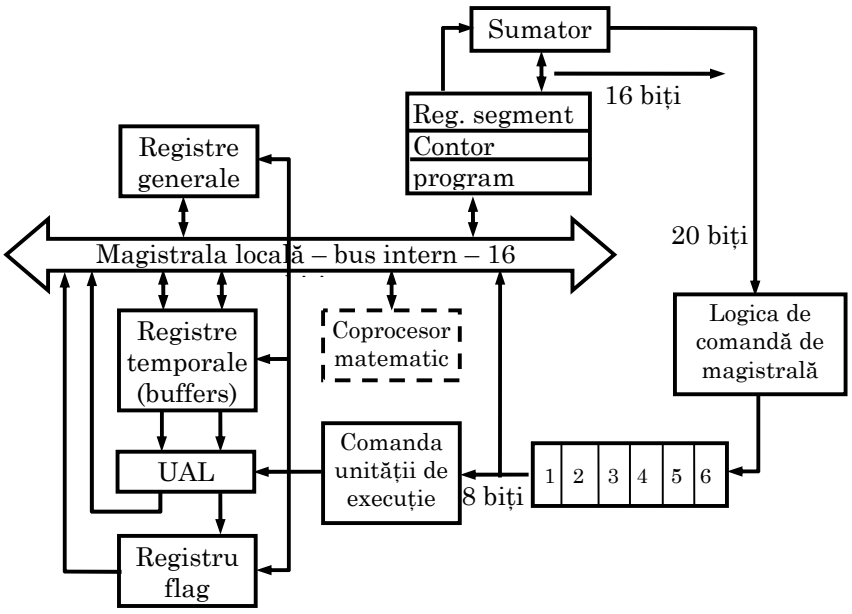

Figura 1. Structura internă de bază a microprocesoarelor Intel

Unitatea de interfață cu magistrala execută toate ciclurile de magistrală (read, write, întrerupere), fie la cererea unității de execuție, fie pentru umplerea cozii de instrucțiuni. Coada de instrucțiuni este un registru FIFO alcătuit din 6 cuvinte, instrucțiunile aşteptând aici intrarea în execuție. Ciclurile de încărcare ale instrucțiunilor sunt executate în intervalele de timp în care unitatea de execuție nu solicită magistrala. Dacă unitatea de execuție nu solicită magistrala şi coada de instrucțiuni este plină, atunci au loc cicluri inactive pe magistrală. Execuția instrucțiunilor de salt duce la resetarea cozii, deoarece trebuie extrase instrucțiuni din altă zonă de memorie.

Unitatea de execuție obține instrucțiuni de la unitatea de interfață cu magistrala, iar în cazul în care coada este vidă, aşteaptă, şi apoi le execută lucrând cu adrese şi date. Registrul de flags este actualizat în funcție de rezultatul fiecărei operații executate. După execuția instrucțiunii, furnizează date şi adrese către unitatea de interfață cu magistrala. De asemenea, unitatea de execuție calculează adresele efective ale operanzilor, conform modului de adresare, prin relocarea adresei efective.

Fiecare ciclu de magistrală constă în intervale de timp date de semnalul de tact, suplimentate cu un timp de aşteptare, dacă semnalul nu este Ready. Pe

durata ciclurilor inactive, se realizează demultiplexarea prin generarea stării anterioare în latch-uri (CBB-RS) externe, pe frontul descrescător al impulsului de tact.

Registrele generale

Registrele generale cuprind:

1) registre de utilizare generală. AX, BX, CX, DX adresabile direct pe 16 biți, fiecare putând servi ca destinație a datelor (acumulator); registrele pot fi adresate şi pe bytes, prin specificarea byte-ului inferior (AL-ALow) sau superior (AH-AHigh) corespunzătoare biților 0-7 (L), respectiv 8-15 (H). Începând cu microprocesoarele I80386 şi până la Pentium, acestea se pot adresa şi pe dublu cuvânt (32 de biți) prin specificarea registrului precedat de E (Extended): EAX, BOX, BCX, EOX. Registrele de ulilizare generală au fost proiectate să aibă o destinație specifică, în concordanță cu operațiile pe care le execută:

(E - Extended - pentru microprocesoarele I80386 şi ulterioare, care lucrează pe 32 de biți.)

AX - este utilizat pentru operațiile de înmulțire şi împărțire, pe 16 biți, respectiv pentru operații de intrare/ieşire pe 16 biți;

AL - este utilizat, pentru aceleaşi operații ca şi AX dar pe 8 biți; în plus se utilizează pentru operații BCO (binary coded decimal) şi conversii de cod;

AH - este folosit pentru înmulțire şi împărțire pe 8 biți;

BX - se utilizează în conversii de cod şi ca registru de bază la adresare;

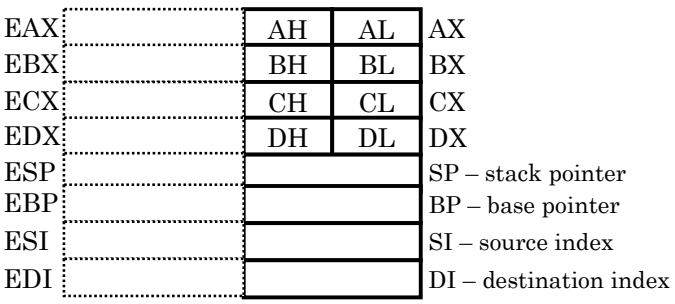

Figura 2. Registrele generale

Registrele de segment

Registrele de segment permit microprocesorului să adreseze direct memoria:

CS (Code Segment) pentru segmentul de program;

DS (Data Segment) pentru segmentul de date curent;

SS (Stack Segment) pentru segmentul de stivă;

ES (Extra Segment) pentru segmentul de date auxiliar;

FS, GS - segmente de date la microprocesoare ulterioare I 80386.

Aceste registre pot defini la un moment dat patru segmente de memorie de câte 64 K fiecare (64 K = 2^6*2^{10}) în mod direct.

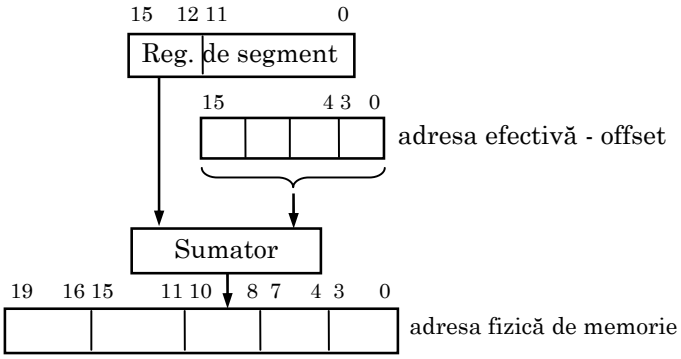

Fig. 3. Tehnica de segmentare

Adresarea în modul real permite microprocesorului să acceseze un spațiu de adrese fizice de până la 1 M (2^{20}) prin tehnica de segmentare, tehnică ce nu adresează direct locațiile de memorie, ci printr-un procedeu care se desfăşoară în două etape:

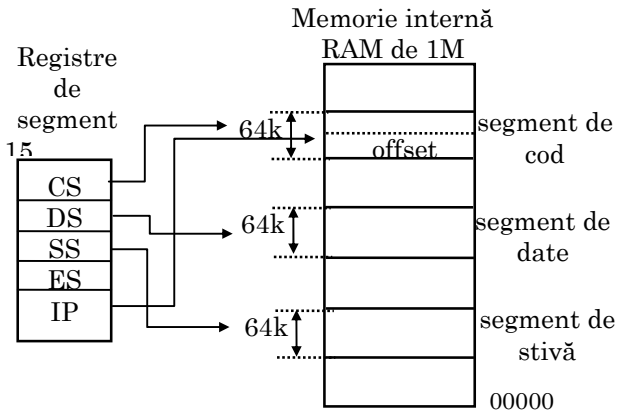

Fig. 4. Adresarea în modul real

- în prima etapă se încarcă un registru de segment cu adresa unui bloc de memorie de 64 K, constituind adresa de bază a segmentului;

- la adresa fizică de bază a segmentului se adună adresa de offset, care constituie deplasarea relativă (offset-ul) față de adresa de bază în interiorul segmentului, rezultând adresa fizică de memorie.

Dezavantajul major constă în aceea că programatorii trebuie să aibă grijă deoarece atunci când se depăşesc limitele unui segment este necesară o nouă reîncărcare a registrelor de segment.

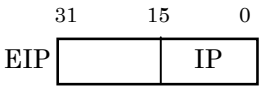

Fig. 5. Registrul contor de program

Adresarea memoriei în mod protejat

Segmentarea

Mecanismul de generare a adreselor în mod protejat este ilustrat în figura 6.

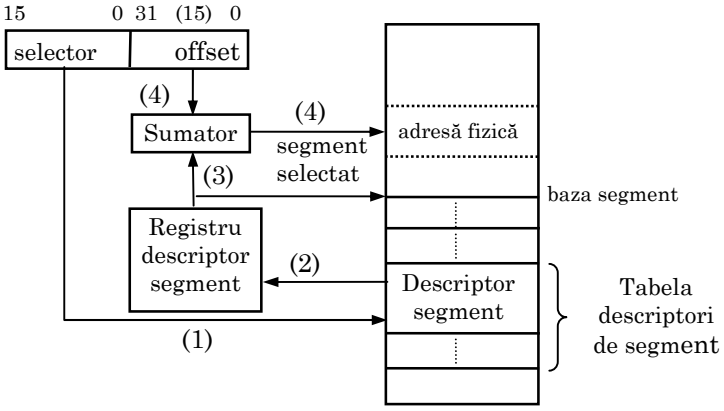

Fig. 6. Adresarea în mod protejat

(1) Registrele de segment în acest mod de adresare nu conțin adresa fizică de bază a segmentului, ci au rol de selectoare cu structura:

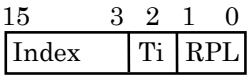

Fig. 7. Registrul selector

RPL (Register Privilege) indică nivelul privilegiat al registrului selector; acesta dispune de patru niveluri de protecție în funcție de softul care solicită adresarea memoriei ca în figura 8.

Protecția constă în capacitatea sistemului de a face ca erorile de software ale unui program să nu afecteze alte programe. În acest scop, microprocesorul dispune de un mecanism de protecție privilegiat de tip inelar crescător, pe patru niveluri, în vederea izolării aplicațiilor software de eventuale erori ale software-ului de bază.

Controlul comunicării între programe şi sistemul de operare este implementat prin separarea spațiului de adresă şi mecanismul privilegiat. Controlul spațiului de adresă separă programele de aplicație unele față de altele, în limp ce mecanismul privilegiat izolează software-ul de bază de cel de aplicație.

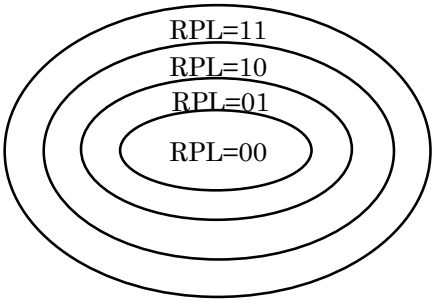

Fig. 8. Nivelurile de protecție

Protecția se bazează pe "ierarhia de prioritati", organizată inelar pe cele 4 niveluri din figura 8:

- nivelul 0 - cea mai mare prioritate,
- nivelul 3 - nivelul cu cea mai mică prioritate.

De remarcat că nivelul privilegiat este un atribut de protecție afectat în toate segmentele din software-ul de bază; acesta determină care proceduri pot accesa segmentul. Drepturile de acces şi limitele sunt realizate prin hardware.

Codul sistemului de operare şi segmentele de date plasate pe cel mai privilegiat nivel (nivelul 0) nu pot fi accesate direct prin programe situate pe alt nivel privilegiat; fiecare program poate accesa date de pe un nivel mai mare sau egal cu nivelul privilegiat care îi este asociat.

TI - indicatoruI tabelului de descriptori de segment.

Tabelele descriptorilor de segment definesc toate segmentele utilizate, existând trei tipuri de tabele ale căror adrese sunt păstrate în registre dedicate:

- tabela descriptorilor globali, GOT (Global Description Table), conține descriptorii disponibili tuturor task-urilor din sistem (TI= 0) şi anume:
  - descriptorii segmentelor de date şi cod folosite de sistemul de operare;
  - descriptorii segmentelor de stare a taskurilor;
  - descriptorii pentru tabelele de descriptori locali ale sistemului.

  Adresa GOT este păstrată în registrul GOTR.
- tabelul deacriptorilor locali, LDT (Local Description Table) - conține descriptorii de segmente ale unui task dat: descriptorul segmentului de cod, date, stivă şi al segmentului ce conține tabela vectorilor de întrerupere (TI = 1). Adresa LDT este reținută în registrul LDTR.

- tabelul vectorilor de întrerupere, IDT (Interrupt Description Table) conține descriptorii ce localizează rutinele de tratare a întreruperii, a căror adresă este păstrată în IDTR.

Indexul este un pointer de intrare în tabelul corespunzător de descriptori.

Un descriptor de segment dintr-un tabel accesat prin index, are următoarea structură descrisă în figura 9:

A (Accessed) = 0 - segmentul nu poate fl accesat
1 - selectorul de segment va fi încărcat în registrul segment

S (segment descriptor) = 1 - descriptor segment de cod sau date
0 - descriptor segment de sistem sau întrerupere

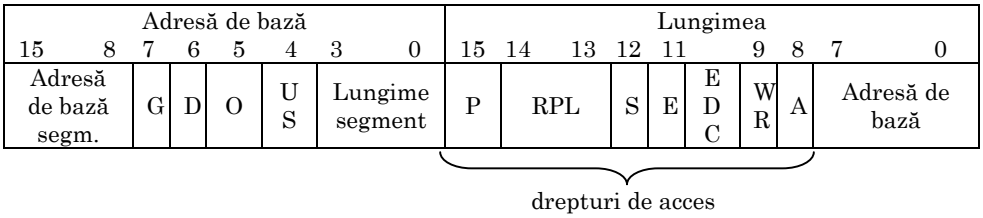

Fig. 9. Descriptor de segment

E (Executable) = 1 - segment de cod
0 - alt segment decât cod

Dacă S = 1, E = 0, atunci este un descriptor de segment de date.

ED (expansion direction)= 0 - extindere în sus, offset ≤ limita
1 - extindere în jos, offset > limita

W (writeable) = 0 – scriere neautorizată
1 – scriere autorizată

Dacă S = 1, E = 1, atunci este un descriptor de segment de cod:

C (conforming) = 1 - se execută dacă are o prioritate ≤ RPL
0 - nu se execută deoarece prioritatea > RPL

R (readable)= 0 - nu autorizează citirea
1 - autorizează citirea

P (present) = 1 - segmentul se află în memorie
0 - segmentul nu se află în memorie

{U S} segment folosit de utilizator sau de sistemul de operare
O - pentru compatibilitatea cu procesoarele ulterioare
D (dimension) = 0 - segment pe 16 biți
1 - segment pe 32 biți
G (granularity) = 0- lungimea segmentului este specificată în bytes
1 - lungime exprimată în pagini (1 pag. = 4 K).

(2) Din tabelul de descriptori segmente, se încarcă automat registrul de descriptor segment (fig. 10), având structura:

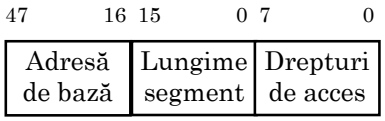

Fig. 10. Registru descriptor de segment

(3) Adresa de bază a segmentului, identifică segmentul din memoria internă,

(4) Adresa fizică se obține relativ la adresa de bază prin intermediul sumatorului, la care se adună offsetul comunicat prin magistrala de adrese de 24 biți ((I80286, I80386, SX $\Rightarrow 2^{24} = 2^{20} \cdot 2^4$ = 16 M RAM) sau 32 biți (ulterioare I80386 DX => $2^{32} = 2^{30}.2^2 = 4$ GRAM).

O altă modalitate de gestiune a memoriei pentru microprocesoarele pe 32 de biți, utilizată pentru sistemele de operare multitasking o constituie paginarea. Spre deosebire de segmentare, care modularizează programele şi datele în segmente de lungime variabilă, ținând seama de logica prelucrării datelor şi a modulelor ce alcătuiesc codul, paginarea divide datele şi programele în zone de dimensiune fixă, numite pagini.

Tehnica paginării, ilustrată în figura 11, preia adresa liniară pe 32 de biți furnizată de magistrala de adrese, conținând:

1) indexul unei intrări în directorul ce conține adresele tabelelor de pagini; pentru a identifica o adresă a unui tabel de pagini, acesta se va aduna cu adresa fizică de bază a directorului de pagini (numărul de intrări este 2^{10} = 1024);

(2) adresa relativă a unui tabel de pagini (2^{10} - 1024 tabele), care se adună

la adresa de bază a tabelului determinat anterior pentru a obține adresa paginii;

(3) offset-ul relativ la adresa paginii selectate anterior, care adunat la această adresă, conduce la obținerea adresei fizice pe 32 de biți.

Paginarea

O altă posibilitate de gestionare a memoriei la microprocesoarele pe 32 de biți, utilizată de către sistemele de operare multitasking, o constituie tehnica paginării. Dacă prin segmentare se modularizează programele şi datele în segmente de lungime variabilă în funcție de logica prelucrării datelor şi a modulelor ce alcătuiesc codul, paginarea împarte datele şi programele în zone de dimensiune fixă numite pagini.

Tehnica paginării este reprezentată în figura 11. Adresa liniară preluată de pe magistrala de adrese, formată din 32 de biți, conține în acest caz trei câmpuri: offset, adresa relativă de tabel şi un index.

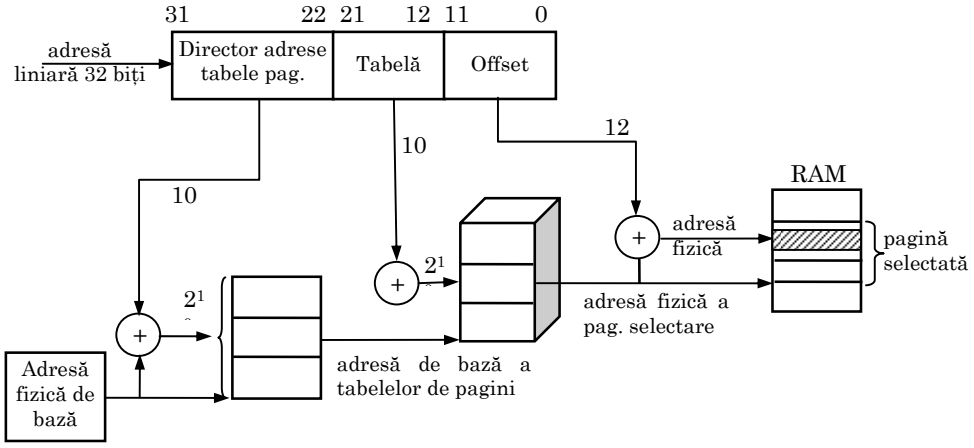

Fig. 11. Tehnica paginării (procesoarele Intel)

Semnificația celor trei câmpuri este următoarea:

1) indexul unei intrări în directorul ce conține adresele tabelelor de pagini; pentru a identifica o adresă a unui tabel de pagini, acesta se va aduna cu adresa fizică de bază a directorului dc pagini (numărul de intrări este 2^{10} = 1024);

(2) adresa relativă a unui tabel de pagini (2^{10} - 1024 tabele), care se adună

la adresa de bază a tabelului determinat anterior pentru a obține adresa paginii;

(3) offset-ul relativ la adresa paginii selectate anterior, care adunat la această adresă, conduce la obținerea adresei fizice pe 32 de biți.

Adresarea memoriei interne pe 32 de biți

La microprocesoarele pe 32 de biți, memoria internă nu mai este conectată la magistrala de 16 biți a sistemului, ci este ataşată direct la magistrala locală a microprocesorului. Aceast fapt a extins modurile de adresare a memoriei interne la 2^{32} = 4G RAM prin adresarea complet liniară (fig. 13).

De asemenea, este folosită tehnica paginării pentru a implementa mecanismul de memorie virtuală prin care un bloc de memorie poate fi încărcat din memoria externă în mod automat, dacă la momentul referirii nu se află în memoria internă. În acest caz, un registru de segment este alcătuit din două câmpuri conform figurii 12.

Fig. 12 Structura registrului de segment

Cu 14 biți ai câmpului Selector se pot selecta $2^{14} = 2^{10} \cdot 2^4$ = 16K adrese a câte 2^{32} off-seturi, deci spațiul de adrese virtuale va fi $2^{14} \cdot 2^{32} = 2^{46}$ = 64 T.

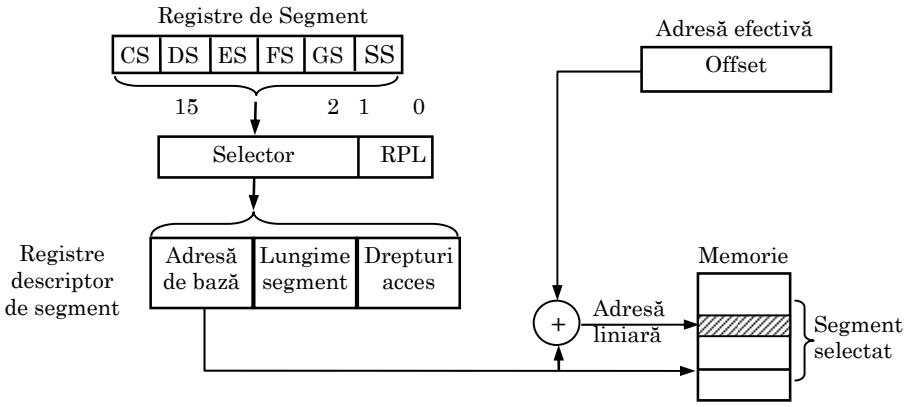

Fig. 13. Adresarea complet liniară

La microprocesoarele pe 32 de biți apare adresarea în mod real virtual, prin care microprocesorul poate simula comportamentul mai multor procesoare care lucrează în mod real. Urmarea este că fiecare utilizator sau task poate să lucreze ca şi când ar avea la dispoziție un întreg mediu real de 1 M.

Modalitățile de adresare a memoriei la microprocesoarele pe 32 de biți sunt redate succint în figura 14.

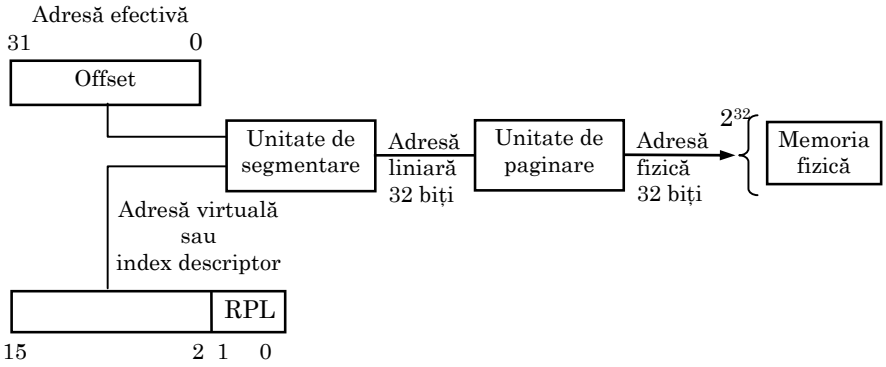

Fig. 14. Adresarea memoriei interne la microprocesoarele pe 32 de biți

Bibliografie

1. Anderson, D., Shanley, T., „Pentium Pro and Pentium II System Architecture", Addison-Wesley, Reading, MA, 1998.
2. Brey, B., „The Intel Microprocessors", Fifth Edition, Prentice Hall, New Jersey, 2000.
3. Bryant, R., O'Hallaron, D., „Computer Systems – A Programmer's Perspective", , Prentice Hall, New Jersey, 2003.
4. Mărşanu, R., „Calculatoare persoanale-elemente arhitecturale", Editura BIC ALL, Bucureşti 2001.
6. Murdocca, M.J., Heuring, V.P., „Principles of Computer Arhitecture", Prentice Hall, 1999.
7. Norton, P., „Sectrete PC", Editura Teora, Bucureşti 1998 (traducere după SAMS Publisluig, USA, 1995).
8. Patterson, D., Hennessy, J., „Organizarea şi proiectarea calculatoarelor: interfața hardwarw/software", Editura ALL, Bucureşti, 2002.
9. Stallings, W., „Computer Organization and Architecture", Fifth Edition, Prentice Hall, New Jersey, 1999.
10. Tamenbaum, A. S., „Organizarea structurată a calculatoarelor", Ediția a IVa, Editura Agora, Tg. Mureş, 1999.
11. Wilkinson, B., „Computer Architecture", Second Edition, Prentice Hall Europe, 1996.