16 Conceptele CISC şi RISC

În primii 20 de ani de dezvoltare a microprocesoarelor a fost adoptată versiunea CISC (*Complex Instruction Set Computer*), calculator cu set complex de instrucțiuni. După 1990, au apărut direcții noi de dezvoltare cum este structura arhitecturală de tip RISC (*Reduced Instruction Set Computer*), calculator cu set redus de instrucțiuni, care prezintă importante avantaje în ceea ce priveşte raportul dintre performanțe şi preț de cost.

Diferența dintre CISC şi RISC nu constă nicidecum doar în numărul de instrucțiuni executabile ci se manifestă mai ales în structura internă şi în modul de funcționare. Avantajele noii arhitecturi au fost atât de seducătoare încât au fost aplicate concepții de tip RISC şi la microprocesoarele cu arhitectură de bază de tip CISC (i486, Pentium etc.).

Totul indică o convergență în viitor a arhitecturilor CISC şi RISC, fiecare preluând de la cealaltă ceea ce are mai avantajos.

16.1 Instrucțiuni simple şi instrucțiuni complexe

Instrucțiunile sunt codificate în cod binar şi stocate în memorie; pentru a fi executate ele sunt decodate de procesor. Instrucțiunile cele mai frecvente sunt cele aritmetice, logice şi de transfer. O instrucțiune poate fi simplă sau complexă:

- instrucțiunea de incrementare a unui registru intern este simplă; ea constă într-o adunare, care se execută în interiorul procesorului şi este suficientă (în general) o singură perioadă de tact pentru execuție.
- instrucțiunea de incrementare a unei locații de memorie este o instrucțiune complexă; operațiile necesare sunt: selectarea celulei de memorie prin intermediul adresei, transferul conținutului ei în interiorul procesorului într-un registru, incrementarea registrului, transferul conținutului înapoi în celula de memorie, la aceeaşi adresă. Desigur, sunt necesare mai multe perioade de tact pentru execuția unei asemenea instrucțiuni.

O instrucțiune simplă este executată direct de procesor prin intermediul unei unități specializate (ca de exemplu UAL - unitatea aritmetică şi logică). Execuția instrucțiunilor complexe în acelaşi mod, ar necesita un număr mare de unități specializate (pentru fiecare tip de instrucțiune, o unitate specifică).

Există însă o limită fizică şi anume numărul de componente ce pot fi integrate într-un microprocesor, care deşi foarte mare, nu permite decât realizarea unui set redus de unităţi specializate. Ca urmare a fost adoptată altă soluţie, încă de la apariţia primelor microprocesore.

Fiecare instrucţiune complexă este descompusă într-o secvenţă de instrucţiuni simple, numite **microinstrucţiuni**. Pentru execuţia unei instrucţiuni complexe se execută de fapt secvenţa de microinstrucţiuni corespunzătoare. Acestea sunt înregistrate, în procesul de fabricaţie, într-o memorie ROM, pe pastila de siliciu a procesorului sub formă de **microcod**. Memoria ROM pentru microcod este accesată prin intermediul unor circuite logice specializate în extragerea secvenţială a microinstrucţiunilor sub comanda unităţii de instrucţiuni.

Microcodul şi logica de extragere şi asamblare alcătuiesc ceea ce se numeşte **firmware**, cuvânt care nu are corespondent în limba română.

Acest concept, care a fost aplicat procesoarelor CISC prezintă avantaje dar şi unele inconveniente.

Decodorul de instrucţiuni descifrează instrucţiunea curentă şi o aplică blocului de microcodaj care o transformă în operaţii elementare (în figura 1 este dată o reprezentare simplificată a acestor operaţii).

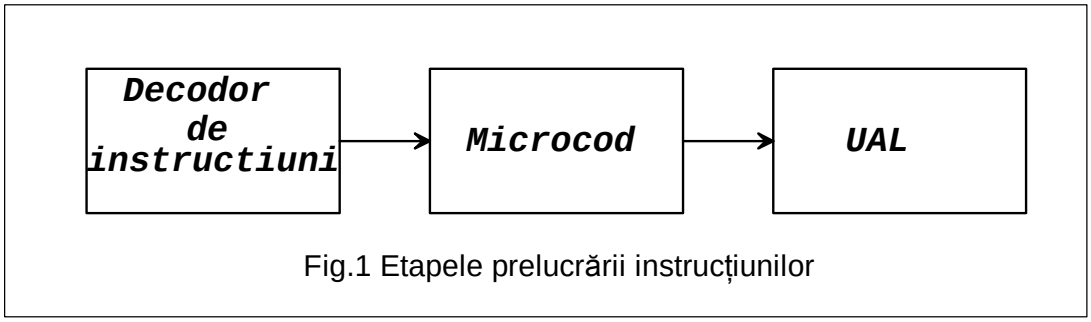

Fig.1 Etapele prelucrării instrucţiunilor

Aşadar, microcodul este înregistrat de fabricant şi este apelat de instrucţiunile executabile după decodarea lor; aceeaşi microinstrucţiune poate fi apelată de mai multe instrucţiuni complexe în procesul de execuţie.

Principalul *avantaj* al acestei concepţii constă în scurtarea programelor de aplicaţii: acestea sunt formate dintr-un număr redus de instrucţiuni complexe, care în procesul de execuţie se multiplică în instrucţiuni simple. Dacă programul de aplicaţie este scurt, el ocupă în memorie un număr redus de locaţii.

Dezavantajele utilizării instrucţiunilor microcodate sunt:

Lungimea instrucţiunilor (în octeţi) şi timpul de execuţie (în perioade de tact) diferă foarte mult de la o instrucţiune la alta, ceea ce

impune procesorului operații logice suplimentare (numărarea octeților pentru separarea unei instrucțiuni, execuția secvențială a unui număr diferit de microinstrucțiuni pe instrucțiune etc.). Viteza de execuție trebuie corelată cu timpul de execuție al celei mai complexe instrucțiuni care este cea mai "cronofagă".

Pe de altă parte, studiile statistice au arătat că în programele de aplicații, instrucțiunile complexe sunt mult mai puțin frecvente decât cele simple (sub 10%), deci instrucțiunile complexe penalizează întregul sistem.

De asemenea, memoria ROM pentru microcod şi circuitele aferente acesteia, ocupă o suprafață importantă pe pastila de siliciu a procesorului, cu implicații directe asupra prețului de cost. Ca urmare, suprimarea microcodului duce la economii substanțiale.

16.2 Suprimarea microcodului şi alte caracteristici ale conceptului RISC

Suprimarea microcodului este concepția de bază a arhitecturilor RISC. Cum vor fi excecutate, în acest caz instrucțiunile complexe ? Există două soluții:

1. O instrucțiune complexă este înlocuită printr-o serie de instrucțiuni simple care realizează aceeaşi funcție în cadrul programului.

Dacă programul este conceput într-un limbaj de nivel înalt, compilatorul este cel care preia sarcina de înlocuire a instrucțiunilor complexe prin serii de instrucțiuni simple. Se pune accentul, aşadar pe realizarea unor compilatoare performante.

2. Dacă totuşi, trebuie păstrate câteva instrucțiuni complexe, acestea nu vor fi microcodate ci vor fi cablate pe pastila de siliciu a procesorului, ceea ce implică adăugarea unor circuite specifice, pentru fiecare instrucțiune complexă, capabile să execute operațiile corespunzătore.

Desigur, aceste circuite ocupă spațiu suplimentar, dar mai puțin decât memoria ROM pentru microcod. De asemenea, execuția unei instrucțiuni cablate este mult mai rapidă decât execuția aceleiaşi instrucțiuni microcodate.

Suprimarea microcodului şi realizarea sub formă cablată a instrucțiunilor complexe oferă încă un avantaj: fiecare instrucțiune poate fi executată într-o singură perioadă de tact, ceea ce constituie unul din obiectivele de maxim interes.

Conceptul RISC (*Reduced Instruction Set Computer*) se bazează pe deviza: "*Să facem totul simplu şi rapid* ". Primele cercetări asupra arhitecturilor de tip RISC au fost inițiate la centrul de cercetări Thomas J. Watson, lângă New York, (IBM) în 1975, sub conducerea lui John Cocke, considerat astăzi autorul conceptului RISC.

John Cocke a analizat funcționarea unui calculator IBM 370, studiind distribuția statistică a utilizării instrucțiunilor. El a constatat că doar un număr mic de tipuri de instrucțiuni sunt folosite foarte frecvent: LOAD, STORE, ADD, SUBSTRACT, BRANCH; celelalte instrucțiuni, în particular cele care execută operații de mare complexitate, intervin foarte rar în programe.

Aceste rezultate au dus la realizarea unui calculator simplu, care avea în setul de instrucțiuni doar pe acelea cu utilizare frecventă, ceea ce a sporit viteza de calcul; pe de altă parte, instrucțiunile complexe au fost înlocuite în program prin serii de instrucțiuni simple, ceea ce a dus la creşterea lungimii programelor. În ansamblu, rezultatele indicau conceptul RISC ca fiind mai eficient.

În anii '70, memoria RAM pentru stocarea programelor în formă executabilă era foarte scumpă, ceea ce justifică în parte de ce i-a fost necesar conceptului RISC un timp îndelungat pentru a se impune.

În 1979, John Cocke realizează primul prototip de calculator bazat pe concepte RISC, IBM 801.

Dezvoltând aceste idei, David Patterson, pe atunci student la universitatea Berkeley, California, apoi profesor, utilizează pentru prima dată termenul RISC. El confirmă că introducerea microcodului constituie un impediment important în creşterea vitezei de calcul a procesoarelor.

Calculatorul IBM 801 a arătat că este posibilă reducerea setului de instrucțiuni şi crearea unor compilatoare eficiente. De asemenea, funcționarea în paralel a mai multor unități interne poate fi realizată şi obiectivul " *o instrucțiune pe o perioadă de tact* " devine realizabil.

La conceptul originar de RISC, se adaugă ulterior idei noi ca:

- Lungimea instrucțiunilor constantă (în octeți).
- Simplificarea modurilor de adresare a memoriei pe baza arhitecturii interne; accesul la memorie se obține exclusiv prin două instrucțiuni: LOAD şi STORE, fără nici o altă complicație.
- Multiplicarea numărului de registre interne ale procesorului, atât a registrelor specializate dar şi a celor de uz general, pentru a limita numărul de transferuri între procesor şi memoria externă.
- Formatul instrucțiunilor pe trei operanzi, ceea ce permite specificarea datei, sursei şi destinației.
- Utilizarea intensivă a memoriei tampon sau "cache", pentru accelerarea operațiilor.

Conceptul RISC nu se reduce aşadar la utilizarea unui set redus de instrucțiuni simple ci include mai multe concepte care simplifică arhitectura internă şi sporesc viteza de calcul. O definiție scurtă şi precisă pentru RISC nu există încă.

Procesoarele bazate pe concepțiile RISC au mai puține tranzistoare în structură, necesită o suprafață de siliciu redusă, sunt mai fiabile şi un preț de cost mai mic. Simplificarea structurii a permis printre altele să crească frecvența de tact.

Stațiile de lucru RS/6000 - IBM, sunt lansate cu succes; bazate pe concepte RISC, sub sistem de operare AIX (versiunea IBM a sistemului UNIX) ele aplică o arhitectură numită POWER, din care derivă şi noua familie de microprocesoare numită PowerPC.

Printre primele microprocesoare RISC comercializate se pot cita:

Clipper C100 (Fairchild), M 88000 (Motorola), ARM (Accorn), Sparc (Sun), AM 29000 (AMD), Transputers (INMOS), PowerPC etc.

Microprocesorul PowerPC se remarcă prin progrese notabile: are o arhitectură pe 64 biți, primele versiuni conțin 2 800 000 tranzistoare pe $11 mm^2$ siliciu în comparație cu primul Pentium care conține 3 100 000 tranzistoare pe $292\ mm^2$ siliciu; are un consum de 9W (PowerPC 601) față de 16 W pentru Pentium.

Tabelul 1 care prezintă caracteristicile de bază pentru câteva procesore RISC de mare performanță a fost publicat în revista americană *PC Magazine* din febr.1994.

Tabelul 1.

Procesor	Alpha21064	Mips R4400	PA-RISC 7100	Pentium	PowerPC 601	Sun Super-Sparc
Concepție	Digital Equipment	MIPS Technology	Hewlett-Packard	INTEL	Apple, IBM, Motorola	Sun Microsystems
Frecvența	200 MHz	150 MHz	99 MHz	66 MHz	80 MHz	50 MHz
Preț unitar	1304 $	450-600 $	(?)	898 $	545 $	860 $
Dim.registre	64 biți	64 biți	32 biți	32 biți	32 biți	32 biți
Alimentare	3,3 V	3,3 V, 5 V	5 V	5 V	3,6 V	5 V
Consum tipic	27 W	10 W - 20 W	23 W	13 W	9 W	9,5 W
Nr.instr./ciclu	2	1	2	2	3	4
Dim. cache	16 k	32 k	(?)	16 k	32 k	32 k

Procesoarele PowerPC reprezintă un bun exemplu de ceea ce propun arhitecturile RISC şi permit evidențierea principalelor puncte de reper:

- Arhitectura generală PowerPC este o *arhitectură superscalară*, în care mai multe instrucțiuni diferite pot fi executate simultan de către unități independente.
- Procesoarele pot deservi sisteme de 32 şi 64 biți.
- Acceptă o gamă variată de memorie: centrală, cache, ierarhizată etc.
- Prin construcție, realizează funcțiile unui sistem multiprocesor, ceea ce garantează eficiența lor.

16.3 RISC - atribute fumdamentale

16.3.1. Multiplicarea unităților specializate

Instrucțiunile sunt distribuite către trei unități de execuție specializate, cu funcționare independentă (PowerPC - fig.2): unitatea de extragere, unitatea de calcul în virgulă fixă şi unitatea de calcul în virgulă flotantă. Instrucțiunile sunt distribuite simultan şi pot fi executate în paralel. Unitățile de execuție dispun de resurse proprii şi sunt sincronizate în mod transparent pentru utilizator. Un spațiu de memorie foarte mare este adresat liniar (calculul adresei se face rapid) iar memoria cache intervine la toate nivelurile sensibile.

16.3.2. Funcționarea "registru la registru"

Este o particularitate a arhitecturii RISC cu tendință de generalizare.

Creşterea numărului de tranzistoare integrate pe o pastilă de siliciu la câteva milioane, pe baza noilor tehnologii, a permis creşterea numărului de registre interne şi implicit apariția unui nou principiu de funcționare: "registru la registru". Conform acestui principiu, datele curente se păstrează în registrele interne evitând transferurile repetate în şi din memoria externă, mari consumatoare de timp. Operațiile aritmetice şi logice se efectuează de asemenea direct între registre: două registre conțin operanzii iar un al treilea va stoca temporar rezultatul operației.

Pentru comparație, microprocesoarele CISC dispun doar de un registru acumulator (care stochează unul din operanzi şi rezultatul operației) şi câteva registre ajutătoare.

Arhitectura RISC adoptă aşadar un mare număr de registre, de două categorii:

1. Registre de uz general, la dispoziția programatorului, în care se poate stoca orice tip de informație;

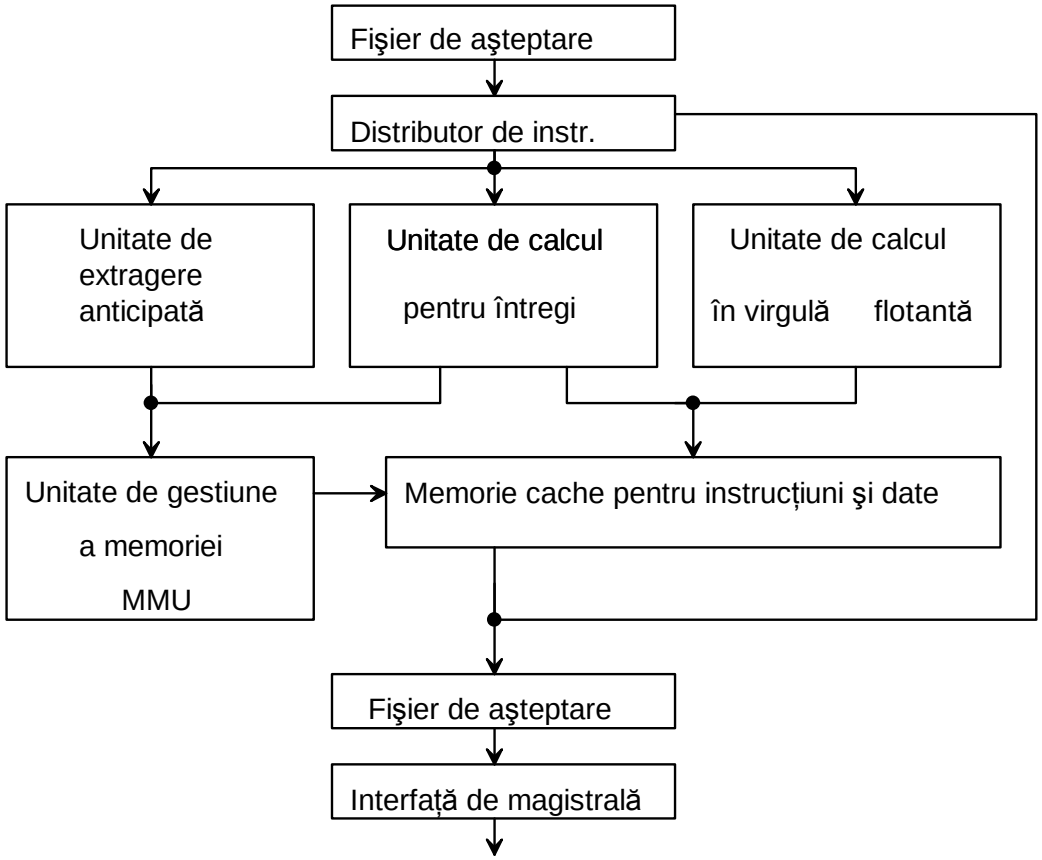

Fig. 2 Arhitectura superscalară la PowerPC.

2. Registre specializate, ale căror funcții sunt predefinite de procesor.

Un procesor de tip RISC, (cum este PowerPC 603) dispune de:

- 32 registre de 32 biți pentru calcule cu întregi;
- 32 registre de 32 biți pentru calcule în virgulă mobilă;
- 32 registre de 32 biți cu funcții predefinite (speciale);
- alte registre speciale pentru funcții de supervizor.

Fiecare instrucțiune poate defini trei registre: două registre sursă şi un registru destinație; când un operand nu se află într-un registru, el este furnizat de instrucțiune (se află în corpul instrucțiunii).

16.3.3. Operații cu memoria

Printre instrucțiunile complexe ale procesoarelor CISC, sunt cele care operează asupra memoriei. De exemplu, instrucțiunea care incrementează conținutul unei celule de memorie, constă în următoarele operații:

- citeşte conținutul celulei de memorie;

- încarcă data citită într-un registru intern al procesorului;
- transferă conținutul registrului unității UAL, care îl adună cu "1";
- stochează rezultatul într-un registru;
- transferă conținutul registrului în memorie la aceeaşi adresă;

Este efectiv complex! În cazul unui procesor de tip RISC, această instrucțiune este suprimată şi este înlocuită cu trei instrucțiuni simple:

- citirea memoriei;
- incrementarea datei citite;
- stocarea în memorie a datei.

În general, la procesoarele RISC se utilizează doar două instrucțiuni pentru operații cu memoia centrală:

- instrucțiunea de citire memorie, LOAD;
- instrucțiunea de stocare în memorie, STORE.

O problemă determinată de concepția de mai sus este aceea de a stoca data la aceeaşi adresă de memorie de unde a fost citită.Problema este rezolvată printr-o *rezervare în memorie*: după transferul datei din memorie în procesor, se rezervă celula din care s-a citit; stocarea condiționată, transferă data în celula rezervată anterior. Între timp, conținutul acestei celule de memorie nu poate şi nu trebuie să fie modificat de un alt procesor sau alt mecanism de operare.

16.3.4. Instrucțiuni de lungime fixă

La procesoarele CISC "tradiționale", lungimea instrucțiunilor este efectiv variabilă: sunt instrucțiuni pe un octet, pe doi, trei sau 4 octeți. De aceea, prima operație realizată de decodorul de instrucțiuni este de a extrage din primul octet al instrucțiunii informația cu privire la lungimea acesteia. În cazul în care toate instrucțiunile au aceeaşi lungime (RISC), această operație devine inutilă şi ca urmare se economiseşte timp; în plus, este suficientă o singură perioadă de tact pentru citirea unei instrucțiuni complete. Utilizarea instrucțiunilor de lungime fixă nu a fost posibilă decât atunci când magistralele de date şi registrele interne au fost extinse la 32 sau 64 biți.

16.3.5. Convergența CISC - RISC

În rezumat, se poate considera că arhitectura RISC este definită prin următoarele caracteristici de bază:

- Majoritatea instrucțiunilor sunt simple;
- Lungimea instrucțiunilor este fixă;

- Operațiile cu memoria centrală se realizează doar prin LOAD şi STORE;
- Arhitectura se bazează pe transferul registru - registru;
- Nu există memorie ROM internă pentru microcod; operațiile sunt cablate;
- O instrucțiune se execută în cel mult o perioadă de tact;
- Formatul instrucțiunilor este cu trei operanzi.

Se constată că majoritatea microprocesoarelor avansate evoluează în aceeaşi direcție şi se adoptă aceleaşi soluții, chiar dacă unele se încadrează în clasa CISC iar altele în clasa RISC.

Diviziunea CISC - RISC nu este exclusivă, ceea ce confirmă încă o dată deviza:

"Cele mai bune soluții rezultă din cele mai bune compromisuri !".

Desigur, evoluția microprocesoarelor nu se va opri la formula RISC.

În prezent, limitele tehnologice impun ca lățimea traseelor conductoare să nu poată fi mai mică de 0,2 microni şi frecvența de tact mai mare de 1000 MHz. De aceea, se caută formule noi, una din soluții fiind utilizarea unor instrucțiuni cu număr mare de octeți (instrucțiuni lungi), **VLIW** - *Very Large Instruction Word*.

17 Arhitecturi evoluate

17.1 Criterii de performanță

Compararea diferitelor arhitecturi este posibilă pe baza unui set de criterii, general acceptate, care au în vedere principalele aspecte hard şi soft.

- Dimensiunea magistralei de date externe. Numărul liniilor de magistrală determină viteza de transfer, exprimată în biți / secundă. O magistrală internă mai largă decât cea externă, contribuie la creşterea performanțelor.
- Frecvența de tact (în MHz). Este determinată de tehnologia de fabricație a microprocesorului şi determină în mod esențial viteza de prelucrare. Se estimează că pe baza tehnologiilor actuale, nu se poate depăşi frecvența limită de 1000 MHz.
- Numărul şi dimensiunea registrelor interne prezintă mai multe aspecte:
  - numărul şi dimensiunea registrelor de uz general;
  - numărul registrelor index;
  - numărul registrelor pointer (de acces la stive);
  - organizarea fizică a registrelor (organizarea sub formă de memorie RAM impune restricții în utilizarea registrelor.
- Memoria adresabilă. Este determinată de dimensiunea magistralei de adrese şi impune luarea în considerație a următoarelor aspecte:
  - numărul locațiilor de memorie şi formatul lor (pe 8, 16, 32, 64 biți)
  - tipul de adresare (liniară, segmentată etc.);
- Timpul de execuție al unei instrucțiuni tipice. Se alege ca referință o instrucțiune aritmetică referitoare la operanzi din registrele interne. Timpul de execuție se exprimă în număr de stări (perioade de tact) necesare.
- Stiva:
  - tipul şi organizarea stivei;
  - existența unor stive *alternative* sau secundare;
  - modul de utilizare a stivei.
- Modurile de adresare a memoriei constituie un criteriu esențial de performanță şi se exprimă prin:
  - numărul şi complexitatea tipurilor de adresare, care determină structurile de date ce se pot construi şi facilitatea utilizării lor;
  - facilități de adresare a porturilor;
  - stabilirea adreselor de salt şi apel la subrutine.
- Setul de instrucțiuni reprezintă unul din principalele atribute de arhitectură. În evaluarea setului de instrucțiuni se ține seama de: operațiile aritmetice şi

logice executabile, tipurile de transfer de date, tipurile de instrucţiuni test, tipurile de instrucţiuni de intrare/ieşire şi tipurile de instrucţiuni de control.

17.2 Microprocesorul Intel 80286. Arhitectură, funcţionare

A apărut în 1981 şi a fost imediat adoptat de IBM pentru realizarea calculatoarelor PC evoluate, de tip AT (**t**ehnologie **a**vansată). Rămâne compatibil cu 8086/8088, dar se extinde modul de adresare a memoriei şi capacitatea acesteia. Intel 80286 se înscrie pe linia evolutivă ce porneşte de la 4004 în 1971:

- 4004 în 1971 (4 biţi);
- 8008 în 1972 (8 biţi);
- 8080 în 1972 (8 biţi);
- 8085 în 1977 (8 biţi);
- 8086 în 1978 (16 biţi);
- 8088 în 1979 (8086 cu magistrala de date de 8 biţi);
- **80286 în 1981 (16 biţi);**
- 80186, 80188 în 1982 - două procesoare asemănătoare cu 8086/8088 dar derivate din 80286 şi conţinând multe circuite specializate în operaţii I/O; aplicaţiile civile au rămas foarte limitate dacă nu chiar confidenţiale !;
- i 386 în 1985 (32 biţi);
- i 486 în 1989 (32 biţi);
- Pentium în 1993 (64 biţi);
- familia P6 în 1996 (Pentium II, III etc.);

Toate procesoarele de mai sus sunt "universale", adică nu sunt proiectate pentru o funcţie particulară. Cele specializate pentru anumite funcţii se numesc coprocesoare şi funcţionează subordonate unui "master".

Cele mai importante clase de coprocesoare sunt:

- **NPX** (*Numeric Processor eXtension*), coprocesor aritmetic;
- **IOP** (*Input-Output Processor*), coprocesor de intrare - ieşire;
- **OSF** (*Operating System Firmware*), pentru de sistem de operare.

Faţă de primele microprocesoare Intel de 16 biţi (8086, 8088) la care a apărut conceptul de segmentare a memoriei, Intel 80286 face încă un pas către mecanismele evoluate de gestionare a memoriei prin utilizarea *memoriei virtuale.*

Aceasta va fi dezvoltată la microprocesoarele următoare şi completată cu alte concepte, a căror înţelegere ar fi mai dificilă fără familiarizarea prealabilă cu soluţiile adoptate la Intel 80286. Deşi acest microprocesor este practic ieşit din uz, descrierea sa prezintă interes

pentru parcurgerea gradată a arhitecturilor evoluate, de complexitate din ce în ce mai mare.

Principalele caracteristici ale microprocesorului Intel 80286 sunt:

- circuit integrat în tehnologia NMOS, de 16 biți, 64 pini, alimentare 5V c.c.;
- păstrează trăsăturile microprocesoarelor 8086/8088, 80186/80188 cu privire la împărțirea memoriei în segmente logice, sistemul de intrare/ieşire şi întreruperi, facilitățile de conectare cu coprocesoare şi sisteme multiprocesor;
- permite utilizarea segmentelor de memorie de dimensiune variabilă (1- 64 kB);
- dispune de mecanisme eficiente pentru modul "*multi-tasking*" (mai multe programe care pot fi rulate succesiv într-o anumită ordine), asigurând o comutare rapidă a task-urilor precum şi protecție pe patru niveluri care blochează tentativele de violare a zonelor de program sau date protejate;
- magistrala de date de 16 biți $(D_0 \div D_{15})$, cea de adrese de 24 biți $(A_0 \div A_{23})$;
- capacitatea de memorie fizică adresabilă 16 MB, memorie virtuală 1GB / task;
- două moduri de lucru: în modul real de adresare este compatibil la nivel de cod maşină cu Intel 8086 iar în modul de adresare protejată este compatibil la nivel de cod sursă cu Intel 8086; în acest mod de lucru, este de circa 6 ori mai rapid decât 8086, la aceeaşi frecvență de tact;
- setul de instrucțiuni este o extensie a setului 8086.
- frecvența de tact: 6 - 20 MHz.

17.2.1. Structura internă

Procesorul dispune de patru unități de execuție care funcționează în modul "*pipeline*": unitatea de adresare a memoriei (AU), unitatea de instrucțiuni (IU), unitatea de execuție (EU) şi unitatea de interfață cu magistralele (BIU).

Unitatea de adresare a memoriei (*Addressing Unit*) este de fapt o unitate de gestiune a memoriei ("*Memory Management Unit"*), care realizează translatarea adreselor logice în adrese fizice şi are funcții de protecție a memoriei; calculul adresei se realizează într-o singură perioadă de tact. Conține registre, sumatoare, şi comparatoare care permit implementarea mecanismelor de adresare, atât în modul real cât şi în modul protejat.

Unitatea de instrucțiuni (*Instruction Unit*) are ca funcție de bază decodificarea instrucțiunilor primite de la BIU; dispune de un fişier de aşteptare, care poate stoca până la 3 instrucțiuni sub formă decodificată şi de un decodificator de instrucțiuni.

Ea funcționează pe baza principiului "pipeline" (conductă, coadă de aşteptare) ilustrat în figurile 3 şi 4. Fişierul de 6 octeți (din BIU), conține codurile a 3 instrucțiuni consecutive extrase din memoria de programe, care după decodificare se încarcă în 3 registre duble (de 16 biți) ce reprezintă coada instrucțiunilor decodificate.

Unitatea de execuție extrage instrucțiunea din primul registru şi ca urmare grupul de instrucțiuni se deplasează spre stânga. Se eliberează astfel două locații în fişierul instrucțiunilor nedecodificate care vor fi încărcate din memorie cu doi octeți de cod, corespunzători instrucțiunii următoare din program, imediat ce magistralele devin libere. Astfel, extragerea unor instrucțiuni se face în paralel cu execuția altora, mărind semnificativ viteza globală de procesare.

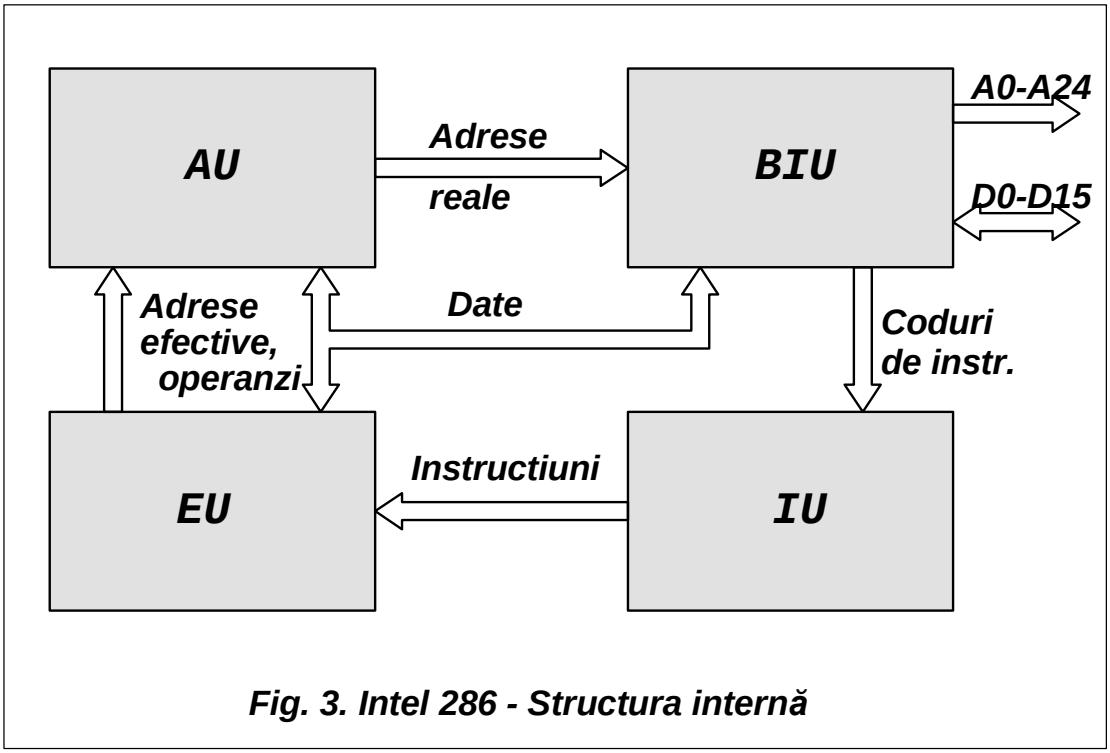

Fig. 3. Intel 286 - Structura internă

Dacă instrucțiunea decodificată la un moment dat este de salt, coada este anulată şi începe reumplerea ei de la noua adresă (de salt).

Unitatea de execuție (*Execution Unit*) are rol de comandă asupra celorlalte unități în procesul de execuție a operațiilor aritmetice şi logice, ce rezultă din instrucțiuni; accesul la magistrale este asigurat cu prioritate de BIU. Unitatea de execuție se compune din:

- unitatea aritmetică şi logică (UAL) destinată efectuării operațiilor aritmetice, logice, deplasări şi rotații;

- registre de uz general, registre speciale şi registre temporare care preiau operanzii de pe magistrala internă de date punându-i la dispoziția UAL;

- unitatea de comandă şi control (UCC), care după interpretarea codului instrucţiunii curente execută următoarele operaţii: comenzi către celelalte blocuri din EU pentru calculul adresei operanzilor sau pentru execuţia unei operaţii aritmetice sau logice; cereri către BIU pentru transferul operanzilor sau rezultatelor (către dispozitive externe sau către procesor); transferul operanzilor de tip imediat din coada de aşteptare; calculul adresei următoarei instrucţiuni. Se poate observa că unitatea de execuţie este separată de exterior, toate sarcinile privind transferul cu exteriorul revenind unităţii de interfaţă cu magistralele (BIU).

Unitatea de interfaţă cu magistralele (*Bus Interface Unit*) controlează accesul la magistrale (de date şi adrese), generează semnalele de comandă pentru transfer de date (inclusiv pentru comunicaţia cu un coprocesor) şi conţine:

- blocul de registre şi drivere de adrese care păstrează adresa stabilă pe magistrală pe durata unui transfer, cu *fan-out* corespunzător;
- amplificatoarele bidirecţionale (*transceivers*) pentru magistrala de date;
- interfaţa cu coprocesorul aritmetic *Intel 80287*;
- blocul de preextragere şi fişierul de 6 octeţi în care se depozitează temporar instrucţiunile extrase în avans din memorie, care formează "coada de aşteptare"; acestea sunt extrase în intervalele de timp în care magistralele sunt libere. În figura 4 este prezentată arhitectura internă dezvoltată.

Microprocesorul Intel 80286 poate funcţiona în două moduri, ceea ce îl deosebeşte de predecesori:

1. **Modul real** - este modul care se instalează automat la iniţializare şi asigură o deplină compatibilitate cu 8086/8088 privind: modul de calcul al adresei efective, setul de instrucţiuni, spaţiul de memorie adresabilă (1MB); este de 2,5 ori mai rapid decât 8086/8088 la aceeaşi frecvenţă de tact.
2. **Modul protejat** - se instalează prin intermediul instrucţiunilor, după instalarea modului real:

LMSW - setează indicatorul PE (*protection enable*) din MSW (*Machine Status Word*);

LIDT - încarcă în memorie tabela descriptorilor de întreruperi.

Sunt utilizate toate resursele hard şi soft: poate adresa 16 MB memorie fizică şi 1GB memorie virtuală (pe 30 biţi), poate funcţiona în modul *multitasking* (execută mai multe programe simultan - de fapt comută execuţia de la un program la altul, cu rapiditate şi utilizarea eficientă a timpului, ceea ce pentru utilizator apare ca execuţie simultană).

În acest mod de lucru este compatibil cu 8086/8088 la nivel de *cod sursă*, adică programele pentru 8086/8088 pot fi executate de 286 după recompilare.

Gestionarea memoriei se face într-un mod flexibil: se păstrează segmentarea, dar lungimea unui segment este variabilă (1- 64 kB) iar adresa de început a unui segment poate fi oricare.

Conținutul unui registru selector de segment nu indică adresa de bază a segmentului ci este un selector pentru extragerea descriptorului de segment dintr-o tabelă de descriptori; descriptorul conține atât adresa de bază a segmentului cât şi alte informații (lungimea segmentului, drepturi de acces etc.).

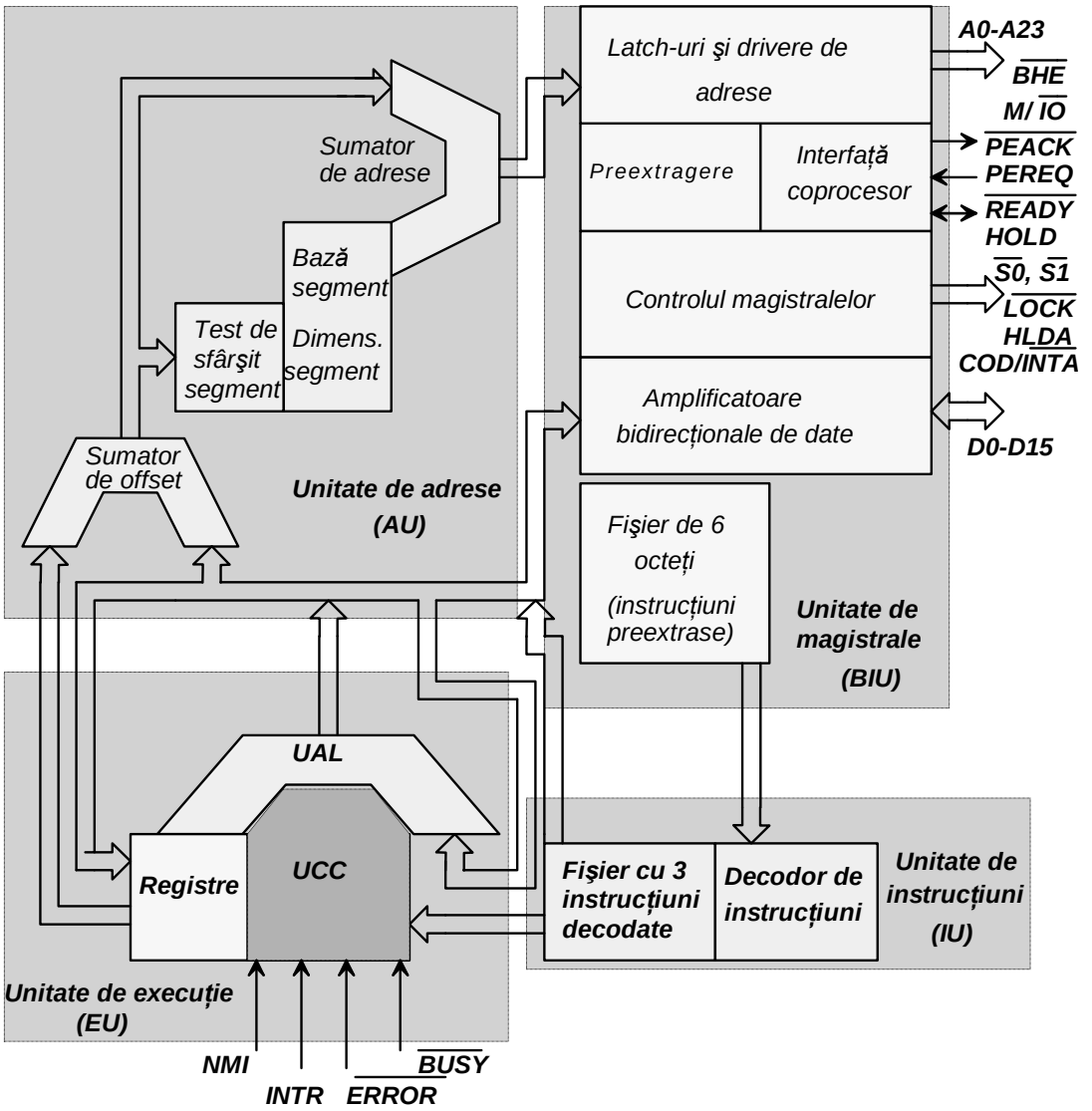

Fig. 4. Arhitectura internă a microprocesorului Intel 286

Observație:

Intel a prevăzut comutarea din modul real în modul protejat, dar a omis comanda inversă. Pentru a reveni din modul protejat în cel real, trebuie salvate toate datele şi reinițializat sistemul.

Proiectanții au considerat că în modul de lucru protejat avantajele sunt suficient de mari încât utilizatorul nu va mai dori să revină în modul real, ceea ce nu s-a confirmat în practică. De aceea, succesorul său, *Intel* 386, dispune de ambele comutări, ceea ce a contribuit la succesul său.

17.2.2. Registrele interne şi cuvântul de stare

Din considerente de compatibilitate, setul de registre este acelaşi cu setul de la Intel 8086/8088. Totuşi, Intel 80286 are un registru în plus,

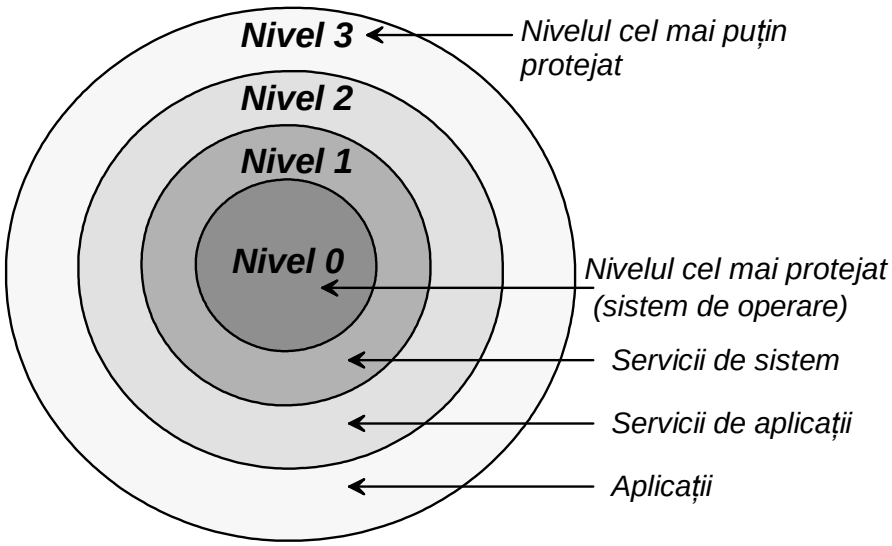

Fig.5. Cele patru niveluri de privilegiu la Intel 286.

destinat memorării cuvântului de stare, MSW ("*Machine Status Word*"), care conține indicatori de sistem. Numărul de registre creşte astfel de la 14 la 15.

Din punct de vedere funcțional, registrele interne pot fi grupate în trei clase: de uz general, de segment, de stare şi control (din care face parte şi IP).

1. **Registrele de uz general** sunt utilizate ca registre de date (AX, BX, CX, DX) sau de adresare a memoriei (SP, BP, SI, DI). Cele de date pot fi utilizate ca registre de 16 biți (AX, BX, CX, DX) sau ca registre de 8 biți, caz în care se păstrează prima literă iar a doua va fi L (Low) pentru

biţii B0 - B7 şi H (High) pentru biţii B8 - B15. Registrele de date au în anumite condiţii funcţii specifice;

- AX (acumulator) poate fi accesat ca AH şi AL;
- BX (bază în adresarea datelor) poate fi accesat ca BH şi BL;
- CX (contor) poate fi accesat ca CH şi CL;
- DX (date) poate fi accesat ca DH şi DL.

Registrele de date sunt utilizate în instrucţiunile aritmetice şi logice.

Anumite instrucţiuni aritmetice utilizează aceste registre cu semnificaţii speciale:

AX - operaţii de intrare/ieşire şi înmulţire/împărţire pe 16 biţi; registrul AL este utilizat în operaţii de intrare/ieşire pe 8 biţi, în operaţii BCD şi în înmuţiri/împărţiri pe 8 biţi; registrul AH este implicat în operaţii de înmulţire/împărţire pe 8 biţi.

BX - operaţii de adresare indirectă a memoriei.

CX - operaţii repetitive cu şiruri şi bucle (contor).

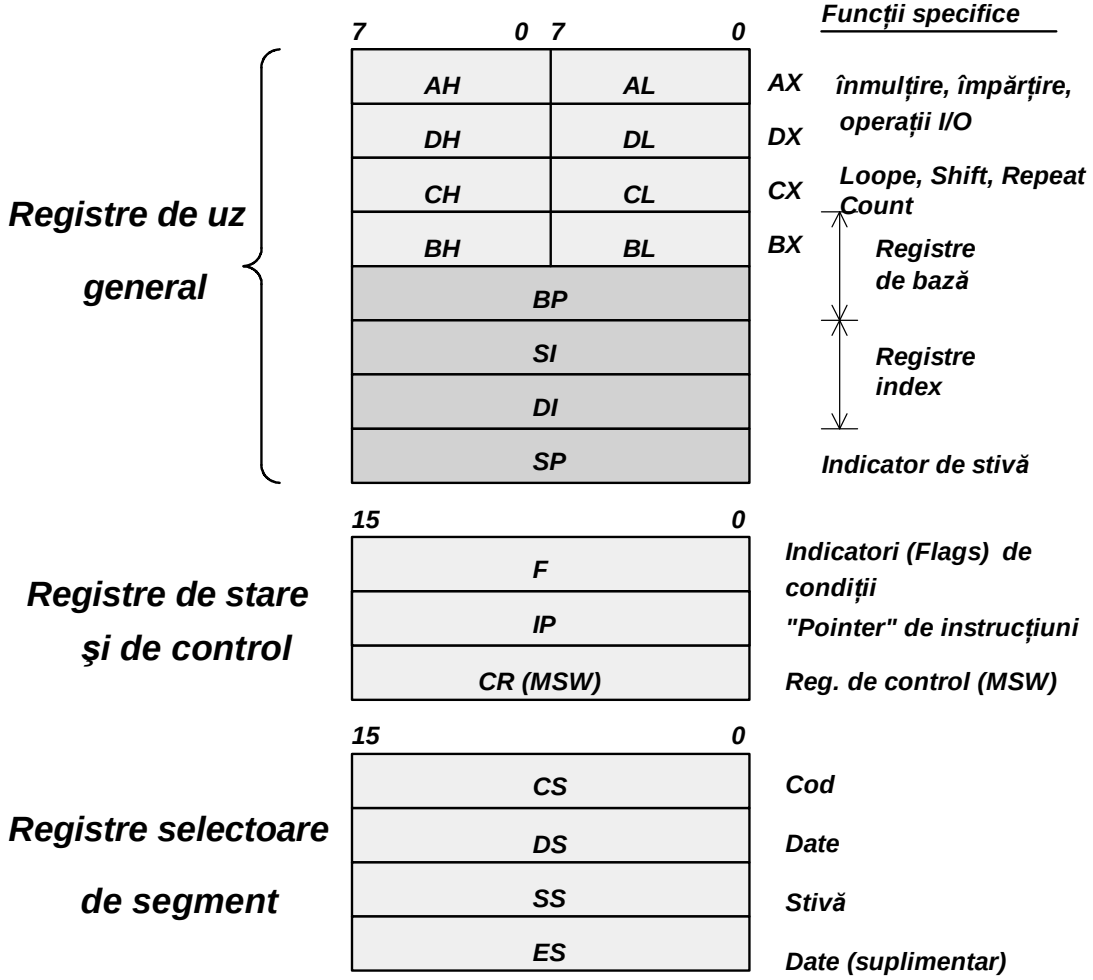

Fig. 6. Registrele interne la Intel 286

DX - operaţii de intrare/ieşire (IN / OUT) cu adresare indirectă şi înmulţire/împărţire pe 16 biţi.

Registrele generale de adresare sunt:

SP - (*Stack Pointer*) conţine adresa locaţiei curente din memoria stivă.

BP - conţine adresa de bază pentru adresarea indirectă a stivei.

SI, DI sunt registre index pentru adresarea indexată a datelor (SI pentru zona sursă, DI pentru zona destinaţie) şi se utilizează în instrucţiunile destinate prelucrării şirurilor de date.

Registrele de segment.

Sunt aceleaşi de la *Intel* 8086/8088, fiind utilizate pentru localizarea segmentelor în memorie. La *Intel* 80286 însă, conţinutul registrelor segment (CS, DS, SS, ES) reprezintă o adresă (selector) care aplicată unei *tabele de descriptori* permite extragerea informaţiilor necesare despre segment (descriptorul de segment):

CS conţine selectorul pentru segmentul de program;

DS conţine selectorul pentru segmentul de date curent;

ES conţine selectorul pentru segmentul de date suplimentar;

SS conţine selectorul pentru segmentul stivă.

Registrele de stare şi control (IP, F, CR)

Numărătorul de instrucţiuni IP (*Instruction Pointer*) conţine deplasamentul adresei pentru instrucţiunea următoare. În mod normal el este incrementat la fiecare extragere a unui octet din zona de cod a memoriei; instrucţiunile de salt modifică conţinutul acestuia prin încărcare paralelă directă sau indirectă.

Registrul de stare F, de 16 biţi, conţine doi indicatori (*flags*) în plus faţă de Intel 8086/8088 (fig. 7.):

IOPL - (*Input/Output Privilege Level*), 2 biţi, utilizaţi în modul protejat, care arată nivelul curent de privilegiu pentru executarea instrucţiunilor I/O fără operaţii suplimentare.

NT - (*Nested Task*) utilizat în modul protejat arată (dacă NT=1) că task-ul curent are în TSS (*Task State Segment*) o legătură validă de întoarcere la TSS-ul task-ului anterior, din care i s-a cedat controlul. Este poziţionat în 1 sau 0 la transferul controlului între task-uri, precum şi de instrucţiunile IRET şi POPF.

Registrul CR (*Control Register*) conţine cuvântul de stare al maşinii (MSW), format din 4 indicatori:

TS (*Task Switched*) este setat automat când se execută o operaţie de schimbare a task - ului.

EM (*Emulate Coprocessor)* când este setat, instrucțiunea ESC generează o excepție de tip "fault" (tip 7) - Coprocesor inexistent.

MP (*Monitor Coprocessor*) când este setat iar TS este de asemenea setat, instrucțiunea WAIT va genera excepția 7, gen "trap".

PE (*Protection Enable*) când este setat este validat modul protejat iar când este resetat se instalează modul real. Atât la Intel 286 cât şi la succesorii lui, PE nu poate fi resetat de instrucțiunea LMSW.

17.2.3. Modul de lucru virtual protejat

Microprocesorul Intel 286 dispune de facilități suplimentare în administrarea memoriei (prin mecanismul adreselor virtuale) şi în protecția task - urilor în lucrul multi-tasking. Aceste facilități sunt operative în modul "evoluat" numit mod virtual protejat sau mai simplu mod protejat, determinat de setarea bitului PE (*Protection Enable*) din MSW cu instrucțiunea LMSW (*Load Machine Status Word*).

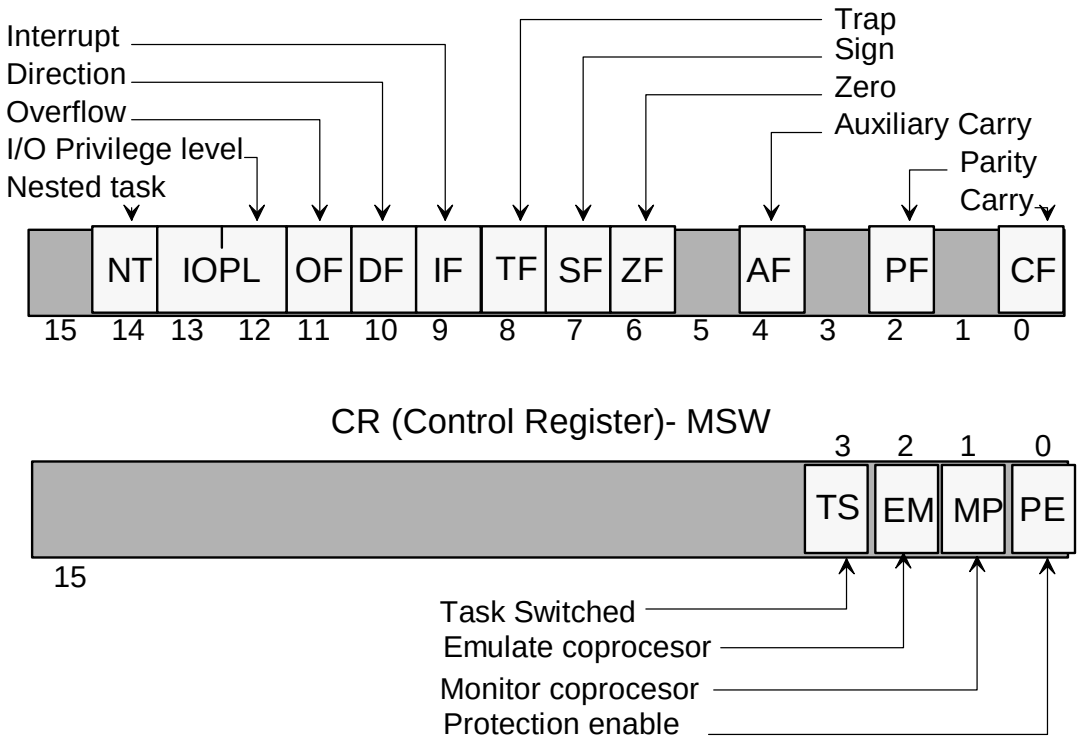

Fig. 7. Registrul de stare (F) şi registrul de control (CR).

În modul protejat, spațiul virtual de memorie este de 1 Gigabyte pentru fiecare din cele 4 task-uri posibile simultan şi un spațiu fizic de memorie de 16 Megabyte determinat de liniile de adresă A0 - A23. Spațiul virtual de memorie este mai mare decât cel fizic deoarece utilizarea unei adrese care nu corespunde unei locații fizice, generează o întrerupere care

comută alte unități de memorie (de exemplu un disc) în spațiul fizic, pentru localizarea operanzilor.

Adresa unei locații este calculată pe baza a două componente (ca în modul real) dar localizarea segmentului nu mai este directă, specificată în instrucțiune sau implicită ci indirectă, prin extragerea din memorie a unui "descriptor de segment". Procedura este impusă de dimensiunea registrelor segment (16 biți) care este insuficientă pentru stocarea informației de adresare şi de protecție a task-urilor. Aşadar, adresa virtuală de 32 de biți se compune dintr-un offset (16 biți) ce reprezintă adresă efectivă şi **selector** (16 biți) aflat într-un registru segment.

Selectorul permite localizarea în memorie a unui grup de octeți care conține atât adresa de bază a segmentului cât şi informații cu privire la drepturile de acces la memorie. Este un mecanism implicit de adresare indirectă a memoriei după cum rezultă din figura 8.

Se observă că adresa fizică se obține prin adunarea offset - ului (adresa efectivă) la adresa de bază a segmentului (24 biți) care la rândul său este obținută din descriptorul de segment, extras dintr-o tabelă de descriptori folosind ca adresă câmpul "selector".

Descriptorii de segment sunt grupuri de 8 octeți (la 80286 se utilizează numai 6) plasați în memorie la adrese succesive, formând "tabela descriptorilor de segment". Tabela este apelată automat de microprocesor la execuția unei instrucțiuni care încarcă un registru de segment.

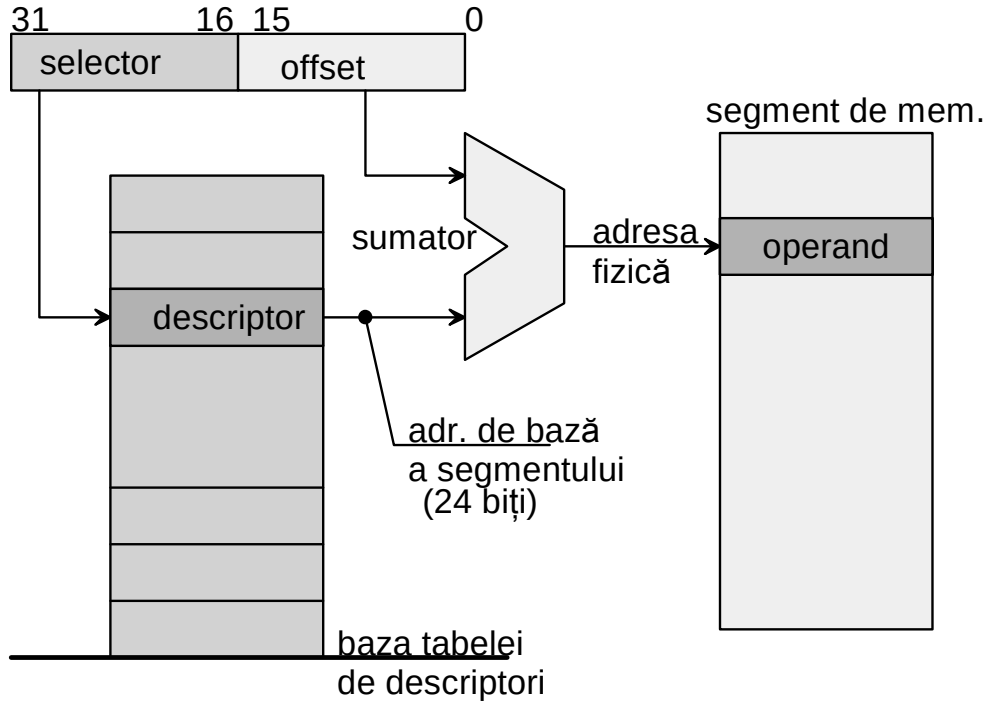

Fig. 8. Adresarea în modul protejat

Microprocesorul va apela automat tabela descriptorilor de segment la execuţia unei instrucţiuni care utilizează un registru segment. Informaţiile de adresare şi acces din descriptor vor fi încărcate în mod transparent, într-un registru ascuns (*caché*) al microprocesorului.

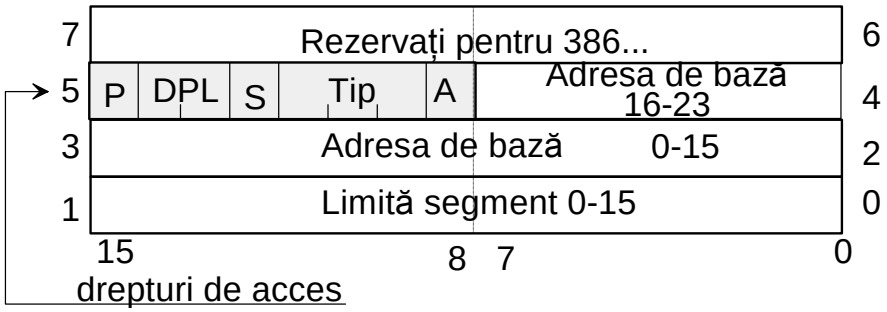

Fig. 9. Structura descriptorului de segment de cod, date şi stivă

Există tipuri speciale de descriptori pentru funcţii de control şi comutare a task-urilor, pe lângă cei de segment de cod, date şi stivă.

Se observă (fig. 9.) că un descriptor de segment conţine informaţii referitoare la:

- adresa de bază (de început) a segmentului de memorie (24 biţi);
- limita segmentului de memorie (16 biţi) - pentru identificarea tentativelor de acces ilegal în exteriorul segmentului;
- drepturile de acces (8 biţi) - specifică atributele segmentului.

Orice utilizare a unui segment care violează atributele indicate de descriptorul de segment, interzice apariţia ciclului maşină de acces la memorie şi produce o întrerupere sau excepţie.

Codurile de instrucţiuni şi datele (inclusiv cele din stivă) sunt memorate în segmentele de cod şi de date, ambele fiind definite de bitul 4 din octetul drepturilor de acces (*S=1*). Segmentele de cod au bitul 3 de valoare logică 1 (*Execution, E=1*).

Segmentele de cod şi de date au trei câmpuri comune în octetul drepturilor de acces: bitul *P (Present*), câmpul *DPL (Descriptor Privilege Level)* şi bitul *A (Accesed)*. Dacă *P=0*, orice încercare de utilizare a segmentului produce o întrerupere de inexistenţă a segmentului (tip 11).

Segmentele de date pot fi parcurse în două sensuri: în sus (*ED=0*, adresa creşte) pentru date propriu-zise şi în jos (*ED=1*, adresa scade) pentru segmentele de tip stivă.

Bitul de conformitate (C) arată că numai anumite programe pot folosi segmentul de cod respectiv (dacă CPL >=DPL).

Semnificațiile câmpurilor octetului privind drepturile de acces rezultă din tabelul 1.

Tabelul 1. Octetul drepturilor de acces

Bitul	Denumire	Funcție
7	Prezent (P)	P=1 ; segmentul are corespondent în memoria fizică P=0 ; nu există corespondent în memoria fizică
6,5	Nivelul de privilegiu al segmentului (DPL)	00 = nivel 0; 01 = nivel 1; 10 = nivel 2; 11 = nivel 3;
4	Tipul segmentului descris (S)	S = 1 ; descriptor de segment de cod, date sau stivă S = 0 ; descriptor de segment tip sistem
3	Executabil (E)	**E = 0** ; descriptor de segment de date
2	Direcție de extindere (ED)	ED = 0 ; extindere în sus (adresa creşte) ED = 1 ; extindere în jos (adresa scade)
1	Drepturi de scriere (W)	W = 0 ; nu se poate scrie în segment W = 1 ; se poate scrie în segment
3	Executabil (E)	**E = 1** ; descriptor de segment de cod
2	Conform (C)	C = 1 ; codul (programul) poate fi executat doar dacă DPL <= CPL (nivelul de privilegiu curent)
1	Drepturi de citire (R)	R = 0 ; nu se poate citi din segment R = 1 ; se poate citi din segment
0	Accesat (A)	A = 0 ; segmentul nu poate fi accesat A = 1 ; segmentul poate fi accesat

Descriptori speciali de control

Modul de lucru protejat defineşte încă două tipuri de descriptori utilizați de sistemul de operare:

1. Descriptorii segmentelor de sistem, care definesc segmente de date speciale (tabele de descriptori) sau stări ale execuției unui task.

2. Descriptorii poartă, folosiți în operații de protecție multi-nivel, multiprocesare şi tratarea cererilor de întrerupere.

Selectorul din adresa virtuală. Tabele de descriptori.

Adresa virtuală este formată dintr-un offset de 16 biți şi dintr-un selector de 16 biți care are rolul de a localiza descriptorul de segment în tabela corespunzătoare. Selectorul are trei câmpuri (Fig. 11) :

- câmpul *INDEX* pe 13 biți utilizat ca offset față de baza tabelei de descriptori:
- un bit (*Table Index - TI*) ca indicator de tabelă:
- câmpul pe doi biți - *RPL* (*Requested Privilege Level*) pentru nivelul de privilegiu cerut.

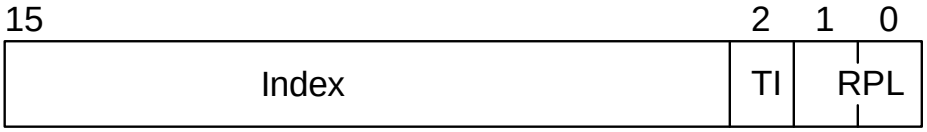

Fig. 11 Structura selectorului

Din cele trei câmpuri, se utilizează pentru localizarea descriptorului numai primele două, adică 14 biți.

Fiecărui descriptor îi corespunde un segment de memorie cu dimensiunea maximă de 64 kB. Se poate calcula dimensiunea maximă a memoriei virtuale:

$$DMV = 2^{14} segmente; \cdot ; 2^{16} B/segm. = 2^{30} B = 1GB \quad ,$$

iar memoria fizică adresabilă este dată de dimensiunea magistralei de adrese (24 biți), adică 16 MB. Din cele 16 x 1024 segmente, există 8 x 1024 segmente cu bitul TI = 0 şi 8 x 1024 segmente cu bitul TI = 1; aşadar, memoria virtuală este divizată logic în două jumătăți:

- pentru *TI = 0* se defineşte *spațiul adreselor globale* folosit de întregul sistem şi de toate procesele (programele), pentru a nu multiplica procedurile generale pentru fiecare proces în parte; în acest spațiu de memorie se află sistemul de operare, bibliotecile de proceduri, suportul pentru compilatoare;
- pentru *TI = 1* se defineşte *spațiul adreselor locale* ce cuprinde segmentele de memorie destinate programelor de aplicații şi datele aferente pentru fiecare proces în parte.

Corespunzător celor două spații de memorie, există două categorii de descriptori, grupați în două categorii de tabele de descriptori de segment:

- Tabela descriptorilor globali (*GDT - Global Descriptor Table*) - unică ;

- Tabelele de descriptori locali (*LDT - Local Descriptor Table*);

Deoarece fiecare descriptor ocupă 8 octeți în memorie, dimensiunea unei tabele de descriptori este de 8 x 8 kB = 64 kB, fiind ea însăşi un segment special (segment de sistem) al cărui descriptor se află în *GDT*.

Modul de lucru protejat defineşte şi utilizează o tabelă specială - Tabela descriptorilor de întreruperi (*Interrupt Descriptor Table - IDT*) care conține descriptori de tip poartă pentru 256 de întreruperi. Referirea la IDT se face prin instrucțiunea INT, vectori de întrerupere externi sau prin excepții.

Registre asociate tabelelor de descriptori

Fiecare tabelă de descriptori se află localizată în memorie la o adresă dată direct sau indirect de câte un registru intern al microprocesorului. Acestea sunt:

- LDTR - registrul tabelelor descriptorilor locali;
- GDTR - registrul tabelei descriptorilor globali;
- IDTR - registrul tabelei descriptorilor de întreruperi.

Aceste registre pot fi accesate în modul protejat prin instrucțiunile LLDT, LGDT, LIDT, SLDT, SGDT, SIDT (primele trei de încărcare, ultimele trei de salvare).