ELEMENTE DE GESTIUNE A MEMORIEI

De regulă la microprocesoarele de 8 biți, adresa generată şi depusă pe magistrala externă de adrese este folosita direct în adresarea unei locații de memorie sau a unui dispozitiv de I/O. Începând cu microprocesoarele de 16 biți apar noțiunile de adresă logică şi adresă fizică. Adresele generate de un program sunt considerate *adrese logice* şi totalitatea acestora formează *spațiul adreselor logice*. Totalitatea adreselor ce corespund memoriei (şi/sau dispozitivelor de I/O) formează *spațiul adreselor fizice*. Cele 2 spații, al adreselor logice şi fizice, pot să fie egale sau inegale. Ca urmare, trebuie să existe un mecanism de conversie a adreselor, de translatare din adrese logice în adrese fizice. La microprocesoarele Intel din seria x86 mecanismul de translatare este inclus pe acelaşi substrat de siliciu cu microprocesorul. La unele din microprocesore (ca de exemplu Motorola MC68000) mecanismul de translatare din adresele logice în cele fizice este preluat de un circuit extern microprocesorului denumit "unitate de administrare a memoriei" (MMU - Memory Management Unit).

În general, tehnicile folosite pentru lucrul cu sistemul de memorie într-un calculator, fie că este vorba de selecție, adresare, alocare de spațiu în memoria principală, folosesc un ansamblu de resurse hardware şi software, operațiile executate de acestea fiind incluse în noțiunea generală de *management al memoriei*. Administrarea întregii memorii (de pe toate nivelurile ierarhice) a calculatorului, incluzând şi modalitățile de partiționare logică şi adresare în spațiul adreselor logice (*L*) şi în spațiul de memorare (*M*, adrese fizice) se face cu ajutorul unei unități hardware pe care o notăm în continuare MMU (Figura 1).

Când un program trebuie rulat (lansat în execuție), el este mai întâi adus din memoria auxiliară înmemoria principală (în întregime sau parțial).

Pentru MMU se impun următoarele cerințe de bază:

1. *să realizeze translatarea adreselor şi să susțină alocarea dinamică a memoriei*
   Înainte ca un program să fie executat, este necesar ca acestuia să i se aloce un spațiu în memoria principală (spațiu de adrese în memoria fizic existentă în calculator). Dacă această atribuire de spațiu (adrese) se face doar la încărcarea programului în memoria principală, procedeul este numit *alocare statică* şi prezintă dezavantajul că adresele de încărcare sunt fixe. *Alocarea dinamică* a memoriei se caracterizează prin atribuirea spațiului necesar programelor, sau proceselor, în timpul execuției acestora. Adresa la care se încarcă programul în memoria principală nu mai este fixă, ea putând fi modificată, la momente de timp diferite.

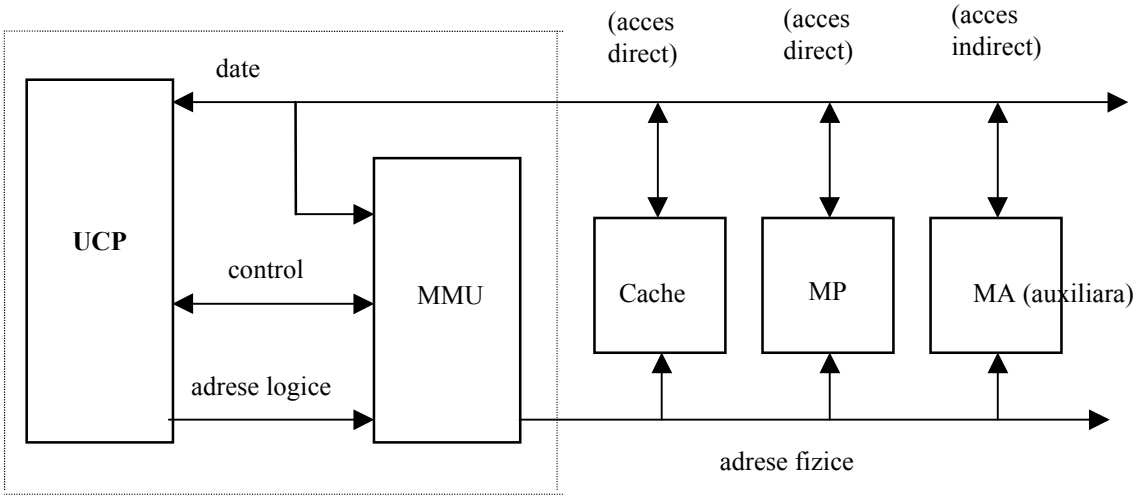

Figura 1. Organizarea ierarhica a memoriei

Se folosesc trei procedee de translatare a adreselor logice în adrese fizice toate bazându-se pe o divizare a spațiului adreselor logice în blocuri de adrese logice continue, după cum urmează:

- *segmente* de lungime arbitrară;
- *pagini* (în spațiul adreselor logice) şi *cadre-pagină / blocuri* (în cadrul adreselor fizice), de lungime fixă;
- combinații de adresare *segmentat-paginată.*

Mecanismul de translatare, indiferent că se folosesc pagini sau segmente, se produce în faza de execuţie a programului şi include printre altele, un *tabel de translatare* prin care se face corespondenţa între adrese logice şi adrese fizice (principiul este prezentat în fig. 2).

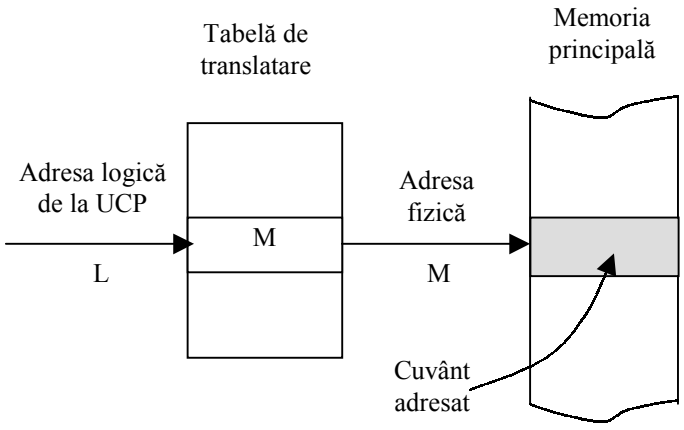

Figura 2. Principiul tehnicilor de translatare a adreselor

De la microprocesor se recepţionează adresa logică L. Dacă elementul adresat nu se găseşte în memoria principală, atunci registrul (de adresă) L va conţine adresa din memoria externă unde se găseşte elementul respectiv şi el va fi adus de rutine ale sistemului de operare în memoria principală, la o adresă fizică a cărei valoare M se înscrie în registrul de adresă L din tabela de translatare. Daca elementul adresat este în memoria principală rezultă că adresa fizică este M. Acest mecanism este totuşi ineficient, căci tabela de translatare are dimensiunile egale cu ale memoriei principale. Pentru micşorarea dimensiunii tabelelor se foloseşte divizarea spaţiului logic în segmente / pagini, iar referirile se fac doar la nivelul de adresă logică de bloc. Trebuie menţionat că dacă sunt lansate două procese în execuţie, fiecare cu spaţiul său de adrese logice, există două tabele de translatare spre memoria principală. Sistemul de operare va reactualiza conţinutul tabelei de translatare ori de câte ori va avea loc o relocare (schimbare de poziţie în spaţiul adreselor fizice) a programelor în memorie.

2. *să susţină mecanismele de implementare ale memoriei virtuale*

   Mecanismul memoriei virtuale permite programelor să fie executate chiar dacă numai o parte din instrucţiunile lor se încarcă în memoria principală, iar restul se găsesc în memoria auxiliară de tip disc magnetic. Adresele logice sunt numite adrese virtuale, în cazul implementării memoriei virtuale, pentru că nu există fizic în memoria principală ci ele se găsesc în memoria auxiliară.

3. *să poată furniza protecţie şi securitate pentru memorie*

   Cerinţa se referă la protecţia informaţiei, la alocarea de drepturi de acces la zone din memoria principală şi la zone din spaţiul adreselor logice, (protecţie la citire, scriere, ştergere, copiere, execuţie). Se protejează memoria disponibilă alocată programelor de sistem şi programelor utilizator, asigurându-se securitatea informaţiei (acces limitat la informaţii), prin controlul accesului la resursele sistemului.

Aşa cum s-a pomenit şi mai sus există două strategii folosite pentru implementarea MMU:

a) MMU se află pe chip-ul procesorului; se aplică la microprocesoarele cu spaţiul segmentat al adreselor logice. De exemplu: Intel x86, Pentium, Zilog Z800, Z8000)
b) MMU este construită separat de UCP, variantă aleasă la microprocesoarele cu spaţiu liniar (organizare liniară a) al adreselor logice. De exemplu: MC 680x0, Z8001).

În cazul în care organizarea logică a memoriei este liniară, adresele încep în mod obişnuit de la 0 şi avansează liniar. Memoria poate fi apoi structurată prin software, la nivelul translaţiei de adrese. În cazul în care organizarea memoriei este segmentată, programele nu sunt scrise ca secvenţe liniare de instrucţiuni şi date, ci ca segmente separate de cod, date, sau stivă. Spaţiul adreselor logice este spart în mai multe spaţii cu adresare liniară, fiecare cu lungime diferită, definită de compilator sau de programator. O adresă logică efectivă este calculată ca o combinaţie între numărul segmentului (care indică adresa de bază a unui bloc de memorie) şi un deplasament (offset) în cadrul segmentului.

În general schemele de adresare liniară sunt mai potrivite pentru manipularea structurilor mari de date, în timp ce segmentarea facilitează programarea, permițând programatorului să structureze programele în module segment. În plus adresarea segmentată simplifică protecția şi relocarea obiectelor în memorie. Segmentarea facilitează gestiunea memoriei în sistemele de calcul multiuser, multitasking.

TRANSLATAREA ADRESELOR

Indiferent de schema de organizare a memoriei (liniară sau segmentată) procesorul trebuie să aibă un mecanism de translatare a adreselor, util în implementarea memoriei virtuale. Acest mecanism este de asemenea util pentru protejarea informației din memorie. Translatarea de adrese este un proces atribuire şi organizare ("mapare") a adreselor logice în adrese fizice de memorie. Mecanismul de translatare împarte memoria principală în *blocuri (cadre pagină)*. Aşa cum am amintit mai sus se folosesc 2 scheme de translatare:

(1) translatare prin paginare
(2) translatare prin segmentare

În *sistemele paginate*, memoria principală este împărțită în blocuri de *lungime fixă* în timp ce în *sistemele segmentate* blocurile sunt de *lungime variabilă*.
Paginile au în general lungimi de ordinul 256 - 4096 cuvinte, în timp ce segmentele cu lungimea definită de compilator sau programator au 64 K-cuvinte, sau mai mult. În sistemele cu multiprogramare şi time-sharing[1], mai mulți utilizatori folosesc aceleaşi programe cum sunt editoare, compilatoare, programe utilitare, biblioteci de programe etc. Atât sistemele cu paginare cât şi cele cu segmentare permit mecanisme de partajare, între procesele utilizator, a paginilor, respectiv a segmentelor. Aceste mecanisme se bazează pe intrări în tabele de mapare de pagină (sau segment) în care intrările diferitelor procese indică către acelaşi bloc din memoria principală. Combinația între segmente şi pagini presupune că un segment conține una sau mai multe pagini virtuale. Mecanismul de segmentare administrează spațiul virtual, împărțind programele în segmente, în timp ce paginarea este destinată administrării memoriei fizice care este împărțită în cadre pagina (blocuri). Vom descrie în continuare mecanismele folosite pentru maparea segmentelor şi paginilor.

Maparea adreselor folosind pagini

În cazul paginării, implementarea tabelului pentru maparea adreselor este simplă pentru că informația din spațiile de adresare şi memorare este divizată în blocuri de dimensiune fixă. Memoria fizică este împărțită logic în blocuri de aceeaşi dimensiune (64-4096 cuvinte fiecare). Termenul *pagină* se refera la blocuri, *de aceeaşi dimensiune*, de adrese din spațiul de adresare. De exemplu, presupunem un calculator care are 20 de biți de adresă şi foloseşte doar 32KB în memoria principală (memorie implementată fizic prin circuite de memorie); dacă o pagină, respectiv un bloc, au dimensiuni egale cu 1 kB, atunci spațiul de adrese e divizat în 1024 pagini, iar spațiul de memorie e divizat în 32 de blocuri.
Se consideră că şi programele sunt împărțite în pagini. Porțiuni din programe sunt mutate din memoria auxiliară în memoria principală în înregistrări egale cu mărimea paginii. În loc de bloc de memorie e folosit uneori termenul de cadru pagină ("page frame").

În cazul mapării prin paginare adresa virtuală este reprezentată printr-un singur cuvânt de adresă împărțit într-un câmp corespunzător numărului paginii (adresa paginii) şi un câmp pentru deplasament. La maparea prin segmentarea un singur cuvânt de adresă nu mai este suficient; dimensiunea variabilă a segmentelor conduce la existența a două cuvinte de adresare, în care primul indică numărul (adresa) segmentului, iar cel de-al doilea deplasamentul în cadrul segmentului.

Într-un calculator cu 2^p cuvinte pe pagină, p biți sunt folosiți pentru a specifica o adresă de linie iar cei mai semnificativi biți rămaşi în adresa virtuală specifică numărul de pagină. Fie de exemplu un sistem simplu cu o adresă virtuală cu dimensiunea de 16 biți şi pagini cu 4096 cuvinte. Pentru că o pagină are 2^{12} cuvinte, cei patru biți mai semnificativi vor specifica una din cele 16 pagini, iar cei 12 biți mai puțin semnificativi indică adresa liniei în cadrul paginii.

[1] time-sharing = partajarea timpului UCP

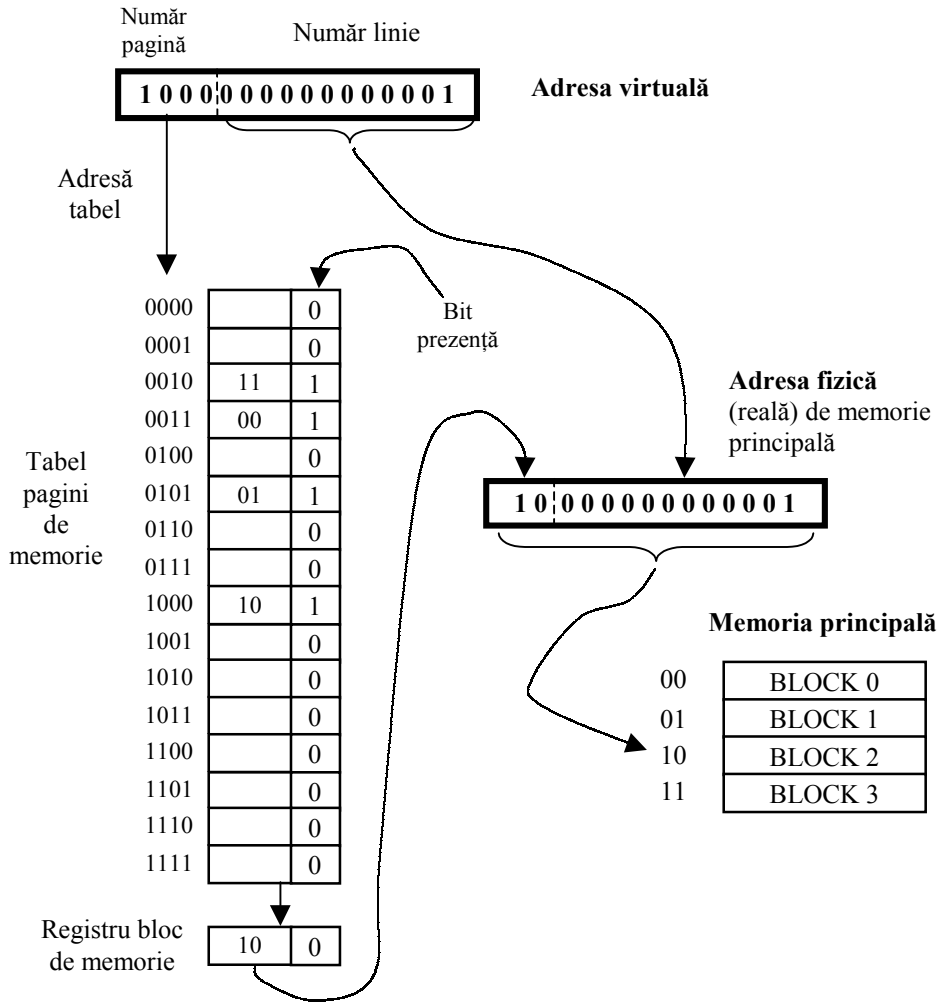

Figura 3. Paginarea memoriei

Ca urmare maparea trebuie făcută doar de la un număr de pagină la un număr de bloc din memoria principală, pentru că adresa liniei e aceeaşi pentru ambele spaţii. Organizarea de principiu a tabelului de mapare a memoriei într-un sistem paginat poate arăta ca în figura 3.

În tabelul paginii de memorie, adresa conduce la numărul paginii, iar conţinutul indică numărul blocului unde pagina este stocată în memoria principală. Se observă că paginile virtuale 2, 3, 5 şi 8 se află în memoria principală, în blocurile 3, 0, 1 şi respectiv 2. Un bit de prezenţă adăugat fiecărei locaţii a tabelului indică dacă pagina respectivă a fost mapată în memoria principală (i s-a alocat spaţiu şi a fost transferată din memoria auxiliară în memoria principală). Valoarea 0 a bitului de prezenţă indică că pagina nu este în memoria principală. În exemplul dat, cei 4 biţi ai adresei virtuale specifică atât numărul paginii cât şi adresa în tabelul paginilor de memorie. Conţinutul tabelului este citit în registrul ce indică blocul de memorie. Dacă bitul de prezenţă este 1, numărul de bloc e transferat în registrul ce păstrează adresa fizică din memoria principală. Dacă bitul de prezenţă este 0 rezultă că adresa virtuală se referă la un articol ce nu se găseşte în memoria principală. În acest caz, se generează o cerere (în forma unei întreruperi software, de tip "eroare de pagină[2]") către sistemul de operare (SO) pentru a aduce pagina cerută din memoria auxiliară în memoria principală, înainte de reluarea programului care a accesat adresa din pagina respectivă.

O astfel de organizare în memoria principală a tabelului de translatare este însă ineficientă, căci tabelul are multe locaţii goale. Dacă presupunem un calculator cu spaţiul de adrese de 4 GB (2^{32}), cu pagini de memorie de 32 KB (2^{15}), iar spaţiul memoriei fizice este 32 MB (2^{25}) rezultă 128 K-pagini virtuale de memorie şi 1024 blocuri (1 K-blocuri). Conform principiului din figura 6.24 tabelul de translatare ar avea

[2] page fault = eroare de pagină

131071 (2^{17} = 128 K) linii şi doar 1024 locaţii ar fi ocupate (cu bit de prezenţă 1), restul fiind goale, neutilizate. O soluţie a acestei probleme o constituie folosirea unei memorii asociative, ca în figura 4, (*tabel asociativ de pagini de memorie*) cu număr de locaţii egal cu numărul de blocuri din memoria principală.

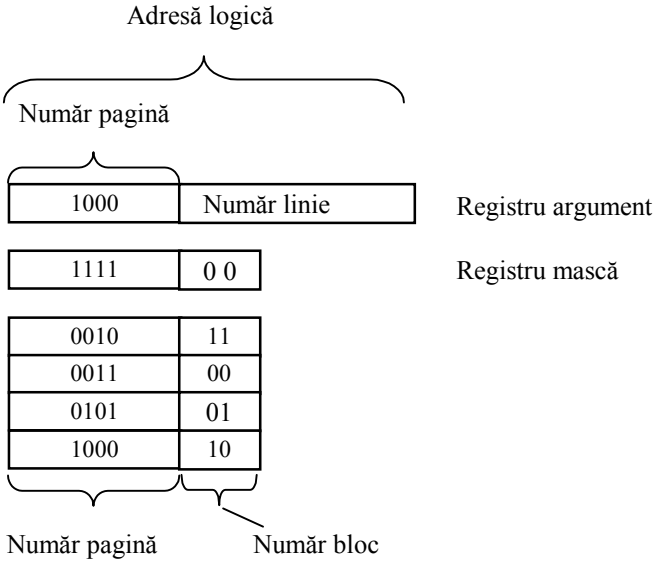

Figura 4. Translatarea adresei de pagină cu memorie adresabila prin continut

În fiecare locaţie de memorie se stochează numărul paginii virtuale şi numărul corespunzător al blocului de memorie principală alocat.

Adresa logică se compară combinaţional cu conţinutul tabelului asociativ, doar pentru poziţiile binare ce corespund la 1 logic în registrul de mascare, deci în figura 4, doar pentru câmpul de adresă al numărului de pagină virtuală. Dacă la apelarea unei pagini virtuale, aceasta nu este mapată în memoria principală, operaţia de adresare este o ratare (miss) şi evenimentul este rezolvat printr-o rutina de servire a sistemului de operare care aduce pagina respectivă din memoria auxiliară. Şi acest tabel asociativ de translatare poate avea dimensiuni mari uneori. De aceea se folosesc tabele asociative de mare viteză la care numărul liniilor este mai mic decât numărul de blocuri (cadre pagină) şi în care se salvează numai o parte din mapările realizate. Acest registru asociativ este numit TLB (Translation lookaside buffer) şi este descris mai jos.

Mapare prin segmentare

La acest tip de mapare spaţiul adreselor logice este împărţit în *segmente*, care spre deosebire de pagini, *nu au o dimensiune fixă*. Segmentele sunt spaţii de adrese continue (liniare), care includ programe sau porţiuni din programe cu atribute comune. Mecanismul de translatare este prezentat în schema de principiu din figura 5.

Spre deosebire de operaţia de paginare, la segmentare nu se mai face concatenarea (alăturarea) adresei de deplasament (din adresa logică) la adresa fizică. Aici adresa de deplasament este adunată cu adresa fizică a segmentului (adresa sa de bază) citită din tabelul de translatare al segmentelor, specific unui anumit proces.

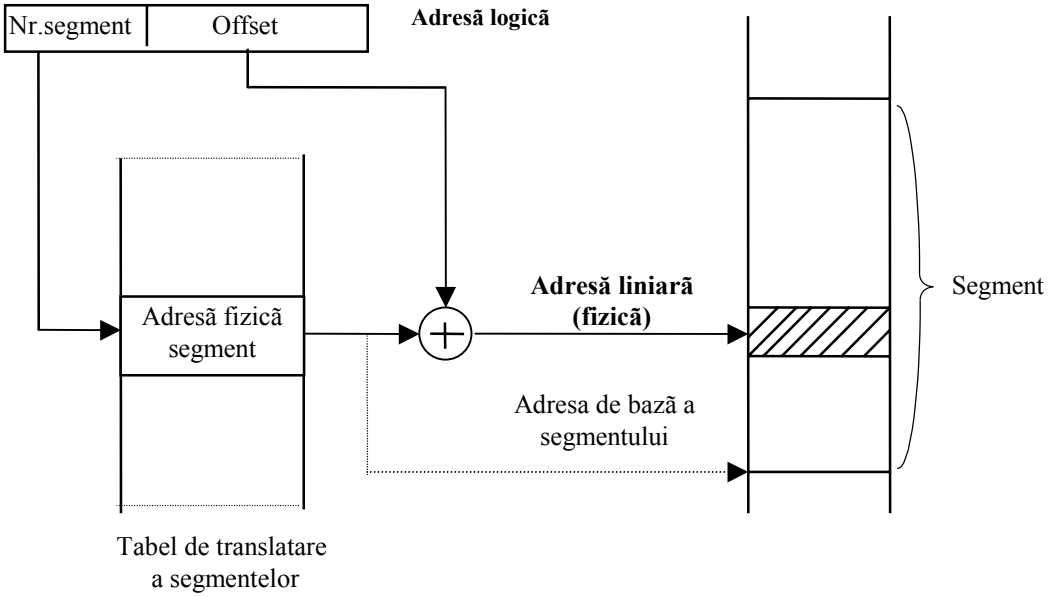

Figura 5: Segmentarea memoriei

Noțiuni privind protecția memoriei

Inventarea multiprogramării, la care un procesor este partajat între mai multe programe aflate în execuție, a condus la noi cerințe privind protecția între programe. Mecanismele de protecție sunt strâns legate de implementarea memoriei virtuale.

Multiprogramarea conduce la conceptul de proces. Un proces este constituit dintr-un program aflat în execuție plus orice stare necesară pentru continuarea rulării sale. Partajarea timpului ("time-sharing") este o variantă a multi-programării care partajează UCP şi memoria între diferiți utilizatori simultani, iar mecanismul de partajare dă iluzia că toți utilizatorii au propriile maşini. Astfel că, în oricare dintre variante, trebuie să fie posibil să se comute de la un proces la altul. Această comutare este numită *comutare de proces* sau *comutare de context*.

Un proces trebuie să se execute corect indiferent dacă se execută continuu de la start până la terminare, sau dacă este întrerupt în mod repetat pentru a se face comutarea către alte procese. Responsabilitatea menținerii comportării corecte a proceselor cade atât în sarcina resurselor hardware, care asigură salvarea şi restaurarea corectă a proceselor ce rulează pe UCP cât şi în sarcina sistemului de operare care trebuie să garanteze că un proces nu interferează cu celelalte şi că un proces nu poate modifica datele altor procese. Aceasta înseamnă asigurarea protecției datelor de către sistemul de operare. În afară de protecție, sistemul de operare permite, în anumite condiții, şi partajarea codului şi a datelor între diferite procese pentru salvare de spațiu de memorie prin reducerea numărului de copii de informații identice.

Cel mai simplu mecanism de protecție este constituit de o pereche de registre care verifică fiecare adresă, astfel încât să nu se permită accesul în afara unor limite alocate în spațiul de adrese. Aceste registre sunt numite registru - bază şi registru limită. O adresă este validă dacă se încadrează între cele două adrese conținute în registre:

$$Bază \leq Adresă \leq Limită$$

La unele sisteme adresa este considerată ca un număr fără semn, care se adună întotdeauna la o adresă de bază, astfel că testul de adresă limită se reduce la:

$$(\text{Bază} + \text{Adresă}) \leq \text{Limită}$$

Dacă proceselor utilizator li s-ar permite să modifice registrele de adrese de bază şi limită, atunci acest mecanism de protecție nu ar putea funcționa. Modificarea respectivelor registre este permisă doar sistemului de operare, pentru asigurarea protecției între procese.

Pentru asigurarea protecției sistemul de operare (SO) are trei responsabilități principale:

1. Să furnizeze cel puțin două moduri de execuție indicând că procesul aflat în rulare este un proces utilizator, sau un proces al sistemului de operare. La diferite SO ultimul tip de proces este numit în diferite feluri: *proces kernel* (nucleu), *proces supervizor*, sau *proces executiv*.
2. Să prevadă o porțiune a stării CPU pe care procesul utilizator o poate doar citi, dar nu o poate scrie. Aceasta include registrele bază / limită, indicatori (biți) pentru moduri utilizator / supervizor şi indicatori pentru validare / invalidare evenimente de tip excepție.

3. Să prevadă un mecanism prin care UCP poate trece din mod utilizator (user) în mod supervizor şi viceversa. Primul sens de trecere este specific apelurilor sistem ("system calls" – apeluri de servicii oferite de SO), implementate ca instrucţiuni speciale care transferă temporar controlul la o locaţie precisă din spaţiul de cod al supervizorului. Registrul contor de program corespunzător locului unde se face un apel sistem este salvat, iar UCP trece în mod supervizor. Revenirea la modul utilizator este similară cu cea produsă la o revenire din procedură.

Adresele de bază şi limită constituie minimul unui sistem de protecţie. Mecanismul memoriei virtuale oferă alternative mai complexe şi mai sigure decât modelul simplu prin bază şi limită. Aşa cum s-a văzut, adresele virtuale sunt translatate în adrese fizice pe baza unor tabele de translatare. Acest mod de mapare, prin tabele, oferă posibilitatea introducerii de informaţii pentru controlul erorilor de program (intenţionate sau nu) care încearcă să treacă peste mecanismele de protecţie. Cea mai simplă cale este introducerea unor indicatori de permisie pentru fiecare pagină sau segment. De exemplu pot exista indicatori care să permită doar citire, doar execuţie, sau care să interzică accesul unui proces utilizator la anumite pagini / segmente. Fiecare proces poate face adresare doar către paginile proprii de memorie, procesul utilizator neavând dreptul să modifice tabelele sale de pagină / segment.

Protecţia poate fi extinsă chiar pe mai mult decât două niveluri (nu doar utilizator şi supervizor), privite ca şi inele concentrice de protecţie, în centru găsindu-se nivelul de protecţie cel mai înalt. În această ierarhie de niveluri de protecţie un program poate accesa doar date de pe nivelul său de protecţie şi de pe nivelurile inferioare în ierarhie. Poate face însă operaţii de apelare (call) a serviciilor sistemului de operare, servicii oferite de rutine ce se află pe niveluri superioare de protecţie. Adesea se face comparaţia, în oarecare măsură nefericită, cu clasificări de tip militar ale sistemelor: top – secret, secret, confidenţial şi neclasificat. Programele utilizator ("civilii" în exemplul militar) au doar dreptul de acces la nivelul de protecţie cel mai de jos: "ne-clasificat. La sistemele de protecţie de tip inele concentrice, deosebirea faţă de exemplul anterior, este că se pot face apelări la rutine situate pe niveluri superioare de protecţie, dacă există "chei" de acces către acele niveluri. Poate exista dreptul de apelare a unor servicii ale sistemului de operare, prin mecanismul de comutare a proceselor.

Aşa cum am pomenit şi mai sus, informaţia de protecţie este setată în registre speciale ataşate fiecărei intrări în tabelele de translatare, registre setate doar de rutine de control al sistemului de operare. Drepturile de acces pot fi de tipul:

a. Atribuirea de privilegii complete de citire şi scriere. Aceste drepturi se atribuie programului atunci când execută propriile instrucţiuni.
b. Read-only (protecţie la scriere). Protecţia la scriere este utilă la operaţiile de partajare a unor rutine de sistem (utilitare, biblioteci, etc.).
c. Execute only (program protection). Protejează programul la copiere. Restricţionează referirea la segment doar în timpul fazei de fetch a instrucţiunii şi nu şi în timpul fazei de execuţie. Asta permite utilizatorului să execute instrucţiunile segmentului de program, dar nu permite citirea instrucţiunilor ca date cu scopul de a copia conţinutul lor.

Sistemul de protecţie a memoriei, în sensul celor spuse mai sus, se construieşte pentru:

(a) memorie (detectează orice eroare de adresare înainte ca aceasta să creeze erori accidentale sau voite);
(b) programe (previne ca programele utilizator (de aplicaţii) să facă modificări ilegale în rutinele SO);
(c) utilizatori (programele utilizatorilor între ele);
(d) securitate informaţie (acces limitat la informaţiile unde utilizatorul nu are drept de acces).