CURS NR. 6, 7

## MEMORIA CACHE

Memoria Cache este numită şi buffer de mare viteză sau zonă tampon, fiind o parte din memoria internă folosită pentru a executa prelucrarea diverselor operaţii din cadrul unui sistem de calcul cu viteză sporită. Poate fi folosită ca buffer de lucru ataşat unităţii centrale de prelucrare pentru păstrarea temporară a datelor şi instrucţiunilor.

Locul ei este între memoria internă şi procesor. În cazul în care datele şi programele sunt plasate în memoria cache nu mai este necesar accesul la o memorie auxiliară, acest lucru conducând la mărirea vitezei de lucru.

În memoria cache sunt reţinute datele şi programele cu frecvenţa cea mai mare de utilizare. Este o memorie de capacitate mică dare de viteză foarte mare.

O dată stocate date sau programe în cadrul ei atunci în mod automat, prin transparenţă, acestea sunt stocate şi în memoria internă principală.

Memoria de tip SRAM este utilizată pentru fabricarea memoriei cache ce se implementează pe plăcile de bază sub denumirea de cache level 2 (L2), la calculatoarele moderne acest cache level 2 se află integrat în procesor, asemenea memoriei cache level 1 (L1) ce este integrată tot în structura procesoarelor.

Memoria cache L1 funcţionează la aceeaşi frecvenţă cu cea a procesorului în timp ce pentru memoria cache L2 frecvenţa de lucru este jumătate din frecvenţa procesorului. Memoria cache a fost introdusă ca un artificiu tehnologic, care trebuie să suplinească diferenţa de frecvenţă dintre procesor şi memorie.

Memoria cache L3 este o memorie specializată care cooperează cu memoria cache L1 si cache L2 pentru a îmbunătăţii performantele sistemului. Are dimensiunea mai mare decât memoriile L1 şi L2, fiind de ordinul MB. Deşi latenţa introdusă este mai mare decât la L1 şi L2 accesarea ei este mai rapidă decât a memoriei DRAM.

## MEMORIA VIRTUALĂ

Memoria virtuală reprezintă o tehnică de organizare a memoriei, prin intermediul căreia programatorul “vede” (dispune de) un spaţiu virtual de adresare de capacitate foarte mare şi care, fără ca programatorul să “simtă” în mod explicit, este mapat dinamic în memoria fizică disponibilă (memoria principală). Uzual, spaţiul virtual de adrese corespunde suportului disc magnetic (memoria secundară), programatorul având impresia, prin mecanismele de memorie virtuală (MV), că are la dispoziție o memorie unică, de capacitatea hard-discului şi nu de

capacitatea, mult mai redusă, a memoriei principale, implementată preponderent sub forma de memorie DRAM, în calculatoarele de uz general (limitată de la 2 la 8Go la ora actuală).

De asemenea, MV oferă și funcții de protecție a accesului la memorie, prin implementarea unor drepturi de acces. În urma comparării acestor drepturi cu nivelul de privilegiu al programelor aflate în rulare, acestea primesc, sau nu accesul la o anumită informație stocată în memorie.

În cazul MV, memoria principală este oarecum analoagă memoriei cache dintre CPU şi memoria principală, numai că de această dată ea se situează, din punct de vedere logic, între CPU şi discul magnetic. Deci memoria principală (MP) se comportă oarecum analog cu un "cache", situat logic între CPU şi discul hard. Prin mecanismele de MV se măreşte probabilitatea ca informaţia (instrucțiuni și date) ce se doreşte a fi accesată de către CPU din spaţiul virtual (disc), să se afle în MP, reducânduse astfel dramatic timpul de acces de la 8 la 15ms cât este uzual timpul de acces la disc, la doar 25 la50 ns (timpul de acces la DRAM).

ARHITECTURA MEMORIEI (Organizare şi adresare)

Caracteristica principală a arhitecturii unui calculator este organizarea memoriei şi modul în care informaţia din memorie este accesată.

Memoria principală este organizată ca un set de locaţii de memorare, numerotate consecutiv, începând de la 0. Numerele asociate locaţiilor fizice reprezintă ***adresa fizică***, iar mulţimea totală a adreselor fizice constituie spaţiul adreselor fizice.

O ***adresă logică*** este o adresă utilizată într-o instrucţiune de către programator. Adresele care pot fi utilizate de un program constituie spaţiul de adrese logice.

a) ***Memoria liniară*** – este cea mai obişnuită organizare a spaţiului de adrese logice şi cea mai obişnuită arhitectură pentru memorie: ***un spaţiu liniar, continuu de adrese***.

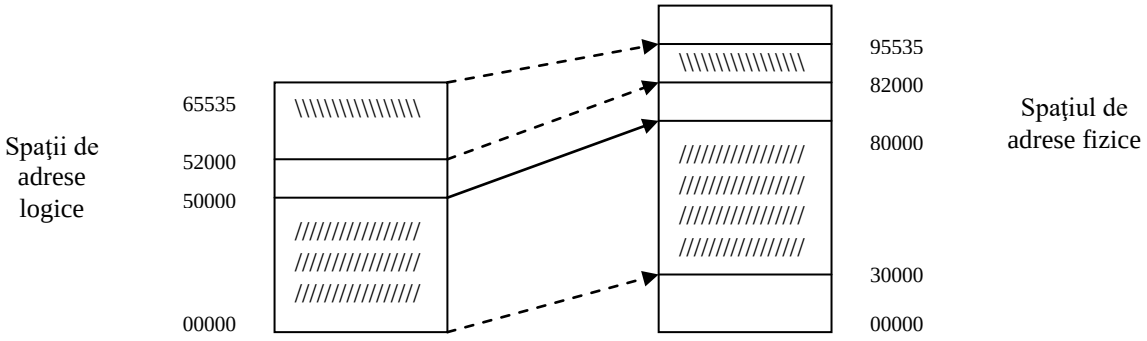
Adresele pornesc de la 0 şi continuă în mod liniar, fără goluri sau întreruperi, până la limita superioară, mpusă de numărul total de biţi dintr-o adresă logică

Ex.: Un program, adesea constând din mai multe proceduri şi datele sale sunt toate plasate în acest spaţiu de adresă unic. Spaţiul de adrese logice al unei memorii liniare are aceeaşi organizare de bază ca memoria fizică.

Pentru o memorie cu 16 linii de adresă pot fi generate 65535 de adrese distincte. O astfel de adresă generată de un program este utilizată de partea hardware a memoriei pentru a plasa data.

***Maparea memoriei liniare***

*Maparea* reprezintă procesul de translatarea adreselor logice în adrese fizice, deci o adresă logică poate fi asignată la o adresă fizică arbitrară

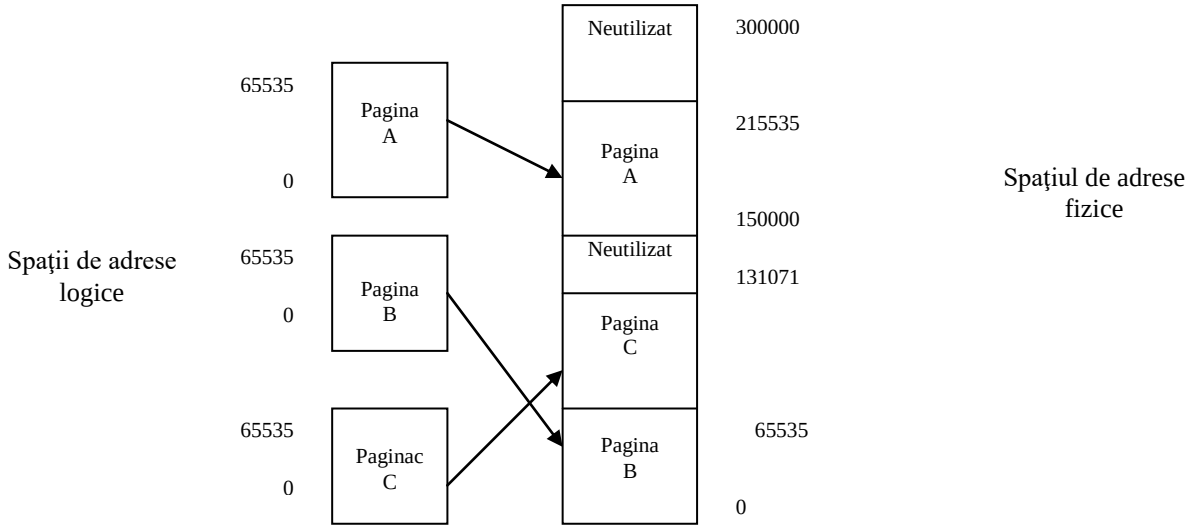



Spaţiul de adrese logice al unui program (0-65535) este mapat (suprapus) peste locaţiile fizice 30000 – 95535. O referinţă a unui program la locaţia 50000 va lua data efectiv de la locaţia fizică 80000 (30000 +50000).

Această operaţie este foarte utilă în sistemele multiprogram pentru care a fost dezvoltată maparea. Fiecare program îşi are propriul spaţiu de adrese. Unitatea de mapare suprapune tot spaţiul adreselor logice într-o memorie fizică.

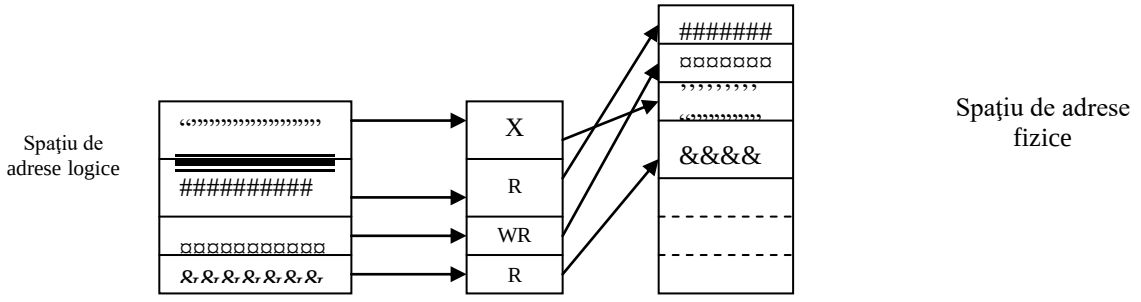
***Maparea bazată pe pagini***

În locul mapării întregului spaţiu logic de adrese ca o unitate, mecanisme mult mai avansate de translaţie a adresei, mapează pagini de dimensiune fixe, mai mici, ale spaţiului de adrese logice, în pagini de memorie fizică. Astfel, un program mare nu trebuie să fie realocat într-o zonă continuă de memorie, care poate fi greu de găsit într-un cadru cu multiprograme, ci în mai multe secţiuni de memorie, mai mici, ce sunt disponibile.




Fiecare pagină poate avea asociate atribute (drepturi de acces) care indică modul de accesare a paginii. Aceste atribute pot permite:

- numai citire;
- citire/scriere;
- pot împiedica orice acces.




b) ***Memoria virtuală***

Spaţiul de adrese logice este mult mai mare decât memoria fizică. Memoria virtuală este un mecanism pentru a extinde limitele memoriei fizice. Într-un sistem cu memorie virtuală, aceasta apare utilizatorului ca şi cum întregul spaţiu logic de adrese este disponibil pentru

memorare. De fapt, în orice moment doar câte pagini din spaţiul logic de adrese sunt mapate peste spaţiul fizic. Alte pagini nu sunt prezente în memoria principală; în schimb, informaţia din aceste pagini este memorată într-o memorie secundară, cum ar fi discul, al cărui cost /bit este mult mai scăzut.

Ori de câte ori este accesată o pagină care lipseşte, sistemul de operare încarcă pagina respectivă de pe disc şi memorează pe disc o altă pagină, care nu a fost recent referită.

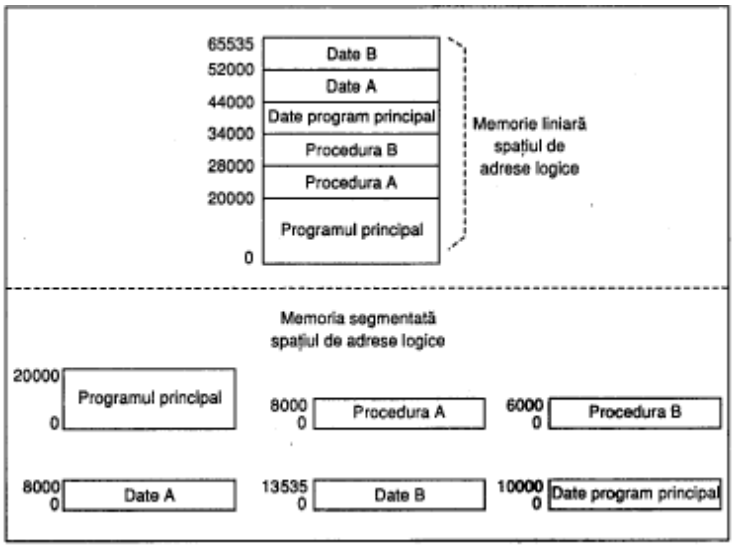
MEMORIA SEGMENTATĂ

O altă formă de organizare a memoriei logice este memoria segmentată. Motivaţia acesteia o reprezintă faptul că programele nu sunt scrise ca o secvenţă liniară de instrucţiuni şi date, ci mai degrabă ca bucăţi (secvenţe) de cod şi bucăţi de date. De exemplu, pot exista o secţiune principală de cod şi mai multe proceduri separate. Aceste module de cod si date pot fi de diferite dimensiuni. Spaţiul logic de adrese este despărţit în mai multe spaţii liniare de adrese, fiecare având o anumită dimensiune. Fiecare dintre aceste spaţii de adrese liniare este denumit segment. Fiecare element dintr-un segment este accesat printr-o adresă cu două componente:

- selectorul segmentului, care specifică adresa de început a segmentului;
- deplasamentul, care specifică adresa relativă, faţă de baza segmentului, a elementului selectat.

Fiecare segment poate fi asociat unui modul de date sau de program. Programul poate avea procedura principală într-un segment, fiecare altă procedură în propriul ei segment fiecare structură importantă de date în segmentul propriu. Astfel, structura adreselor logice reflectă organizarea logică a programului.

Mecanismele de protecţie pentru memoria liniară sunt bazate, de obicei, pe pagini cu lungime fixă, a căror dimensiune este determinată pe criterii hardware fără vreo relaţie cu structura logică a programului.



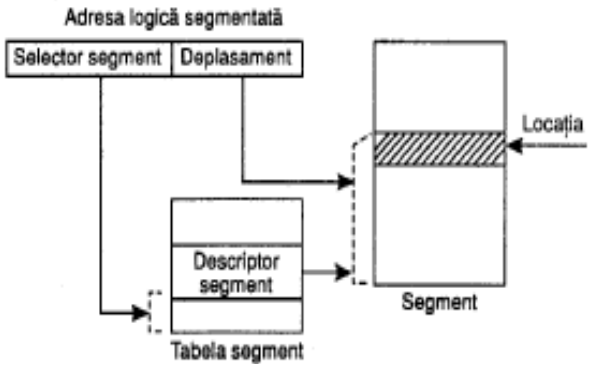

Compararea memoriei liniare cu cea segmentată

Problema descompunerii spaţiului de adrese logice în pagini este că mecanismul cu protecţie nu poate proteja exact modulul de program; acesta protejează fie prea puţin, fie prea mult. Dimensiunea paginii este determinată din raţiuni de hardware, deci nu are legătură cu structura logică a programelor. În schimb, întrucât fiecare segment are o anumită lungime, este uşor să-l protejăm de alte programe. Mecanismele de memorie virtuală pot fi implementate şi pentru arhitecturi segmentate. În acest caz, segmentul este unitatea de memorie ce se interschimbă (swapping) cu memoria externă.

***Maparea memoriei segmentate***

Maparea, în acest caz, este implementată printr-o tabelă de segment, care păstrează descriptor de segment pentru fiecare segment.

*Descriptorul de segment* conţine adresa de început a segmentului şi lungimea acestuia. Componenta selectorului de segment a unei adrese logice este utilizată ca un index pentru a selecta descriptorul de segment din tabela (descriptorilor) de segment. Apoi deplasamentul este adunat la adresa de start a segmentului, furnizată de descriptor, pentru a calcula adresa fizică a operandului referit. Deplasamentul este verificat de hardware, pentru a se asigura că referinţa nu depăşeşte lungimea segmentului.




TIPURI DE SEGMENTE ŞI DREPTURI DE ACCES

Descriptorul de segment mai conţine, pe lângă adresa de bază a segmentului şi limita sa, şi atribute referitoare la tipul segmentului. Drepturile de acces sunt, deci, asociate, în particular, fiecărui segment, în ciuda faptului că modulele programului referă acele segmente. Dacă un segment este de tip read-only (numai pentru citire), de exemplu, îşi menţine tipul pentru toate modulele care fac referire la acesta. Această asociere a drepturilor de acces cu segmentele este un dezavantaj, deoarece putem dori să dăm diferitelor module acces la acelaşi segment, dar cu diferite drepturi de acces.

### CONTROLUL ACCESULUI

Un alt dezavantaj al mecanismului de mapare a segmentelor este dificultatea de a limita accesul unui program la segmentele altui program. Deoarece tabela de segment conţine toţi descriptorii de segment, orice program poate accesa orice segment prin simpla indexare în tabela de segmente. Soluţia este ca fiecare program să aibă propriile sale tabele. Dar aceasta înseamnă că, ori de câte ori un segment este realocat în memoria fizică, toate programele partajează segmentul vor trebui să actualizeze descriptorii lor de segment. Pentru a reduce numărul operaţiilor de acest gen, fiecare task vede spaţiul de memorie divizat în două: un spaţiu global, care conţine serviciile de sistem, sistemul de operare şi un spaţiu local asociat fiecărui task.

### MEMORIA VIRTUALĂ ŞI ALOCAREA DINAMICĂ

Fiecare modul de program are propriul său cadru de acces şi în acest cadru (de exemplu, 32 de linii de adresă) se pot genera $2^{32}$ adrese diferite. Deci spaţiul total de adrese, cel virtual, este egal cu dimensiunea maximă a unui segment înmulţită cu numărul total de segmente.

Exemplu: la 386/486 şi la Pentium cu o magistrală de adrese de 32 de biţi şi un selector de 16 biţi, dintre care 14 sunt utilizaţi pentru adresare, rezultă un spaţiu total de adresare virtuală de 64 To. Fiecare descriptor de segment trebuie să conţină un câmp de biţi, actualizaţi hardware, utilizaţi de sistemul de operare pentru a implementa memoria virtuală:

- ✓ valid (sau prezent), identifică dacă segmentul este prezent în acel moment în memorie;
- ✓ memorie alocată, indică dacă a fost asociată memorie acestui descriptor;
- ✓ accesat, indică dacă segmentul a fost accesat de un program;
- ✓ modificat, indică dacă informaţia din segment a fost modificată sau nu de un task.

Sistemul de operare poate utiliza aceşti biţi:

- ✓ valid/memorie alocată, pentru a detecta când un segment fizic nu este prezent memorie;
- ✓ accesat/modificat, pentru a decide care dintre segmentele curente prezente ar trebui interschimbat sau pur si simplu, scrierea segmentului nou peste cel vechi (neutilizat curent) dacă n-au fost făcute modificări.

În plus, câteva câmpuri din descriptor pot fi utilizate de sistemul de operare, pentru a memora alte informaţii folositoare despre segment.

Cantitatea de memorie alocată unui modul nu trebuie să fie fixă la momentul compilării; se poate modifica dinamic. Mecanismul de alocare dinamică a memoriei face memoria virtuală mult mai eficientă prin alocarea doar a segmentelor, când sunt ele necesar împiedicând astfel segmentele nereferite să utilizeze memoria.