**Cache**

* memorie interna, conţine informaţiile cele mai recent utilizate de către CPU.
* capacitate relativ mică, dar cu timp de acces foarte rapid.

Proiectare directa:

+ simplitate

- fiecare bloc are o poziţie fixă în memoria cache => rata mare de coleziuni

Proiectare asociativa:

+plasează un bloc de memorie într-un slot oarecare liber => reduce coleziuni

- politica de înlocuire => complexitate mare

Proiectare set-asociativa:

+ reduc riscul de coleziuni (cache-trashing)

- problema de planificare, complexitate mare (necesita logica suplimentare pt gestionarea mia multor blocuri per set)

* pentru o cerere de memorie de la adresa A: I seturi J sloturi, numarul K al setului unde intra blocul: K = A mod I

**Calculul de adresa**

* **OP** notăm spaţiul de nume al obiectelor din programul sursă: nume de constante, de variabile, de etichete, de proceduri etc.
* **AM** este adresa relativă din cadrul unui modul compilat.
* **AR** este adresa relocabilă – adresă relativă din cadrul unui segment dintr-un fişier executabil.
* **AF** notăm mulţimea adreselor fizice din memoria operativă la care face referire programul în timpul execuţiei.

Calculul de adresă este modalitatea prin care se ajunge de la un obiect sursă din OP la adresa ui fizică din AF. **OP-c->AM-l->AR-t->AF**

**Tehnici de alocare**

* **alocare reală:**

o la SO monoutilizator – este disponibil aproape intre spatiul de memorie (gestionat de utilizator – foloseste tehnici de overlay)

o la SO multiutilizator:

* cu partiţii fixe (statică): decuparea memoriei în zone de lungime fixă numite partiţii(O partiţieeste alocată unui proces pe toată durata execuţiei lui, indiferent dacă o ocupă complet sau nu)

o absolută;

* + Adresa Fizica˘=Baza Partiției+Adresa Relativa˘ 
  + **Baza Partiției** este adresa de început a partiției fixe alocate procesului.
  + **Adresa Relativă** este adresa specificată în instrucțiunea sau datele procesului, relativă la începutul partiției.

+ Fiecare proces este încărcat într-o zonă de memorie prestabilită.

- Fragmentare externă și internă, deoarece memoria este împărțită în partiții fixe. Spațiile de memorie pot rămâne neutilizate dacă procesele nu se potrivesc exact în ele

o relocabilă;

+ Alocarea memoriei se face cu bază și deplasament, permițând plasarea proceselor în orice partiție disponibilă)

- Dacă un proces este plasat într-o partiție insuficientă, acesta poate fi eliminat fără a fi executat **sau** Partitii de obicei de lungimi diferite si este greu de calculate dimensiunea

* cu partiţii variabile (dinamică);

+ o exploatare mai suplă şi mai economică a memoriei SC. In funcţie de solicitările la sistem şi de capacitatea de memorie încă disponibilă la un moment dat, numărul şi dimensiunea partiţiilor se modifică automat.

-fragmentare interna a memoriei

**- alocare virtuală: -** capacitatea de a adresa un spaţiu de memorie mai mare decât este cel disponibil la memoria operativă a SC concret

* paginată; (tabela de pagini)
  + M[0..m] notăm memoria operativă, prin k puterea lui 2 (numărul de biţi) care dă lungimea unei pagini, prin TP adresa de start a tabelei de pagini, atunci algoritmul calculul funcţiei t este:

**t (p, d) = M[TP + p] \* 2 ^ k + d**

+ foloseşte mai eficient memoria operativă, fiecare program ocupând numai memoria strict necesară la un moment dat.

- dimensiunile paginilor sunt fixe, este posibil ca procesele să nu utilizeze complet ultima pagină alocată, ducând la risipirea memoriei în cadrul paginilor (fragmentare internă)

* segmentată; (tabela de segmente)
  + la adresa TS se află începutul tabelei de segmente. Cu notaţiile obişnuite, funcţia t de translatare a adresei se calculează astfel: **t(s,d) = M[TS+s]+d**

+ Se pot crea segmente care pot fi folosite în comun de către mai multe procese **sau** se poate realiza o foarte bună protecţie a memoriei. Fiecare segment în parte poate primi alte drepturi de acces, drepturi trecute în tabela de segmente.

- adresa fizică este una oarecare. Este deci posibil să apară fenomenul de fragmentare

* segmentată şi paginată. (tabela de segmente + tabela de pagini):
  + Fie k constanta ce dă dimensiunea unei pagini (2^k), TS adresa de început a tabelei de segmente a unui proces şi presupunem că primul câmp al fiecărei intrări din tabela de segmente este pointerul spre tabela lui de pagini, atunci funcţia t de translatare se calculează astfel:

**t(s,p,d) = M[M[TS+s]+p]\*2k+d**

* + adresa virtuala (s, p, d) s-nr segment, p-nr pag virtuala din segment, d-deplasamentul in cadrul paginii
  + adresa fizica (f, d) f-nr pag fizica, d-deplasamentul

+ alocarea spaţiului pentru fiecare segment să se facă paginat.

**Politici de inlocuire**

* înlocuirea unei pagini care nu a fost recent utilizată (**NRU** - Not Recently Used);

R. M (biutul referit si bitul modificat)

clasa 0: 0 0 <- pagina “victima”

clasa 1:0 1

clasa 2:1 0

clasa 3:1 1

* înlocuirea în ordinea încărcării paginilor (**FIFO** - First In First Out);
* înlocuirea paginii nesolicitate cel mai mult timp (**LRU** - Least Recently Used).

met1: numerator de accese cu un contor <- pagina cu cel mai mic contor “victima”

met2: matricea de referinte<- pagina “victima” = a k-a linie cu cele mai putine cifre de 1

**Politici de incarcare**

* + incarcarea la inceput a tuturor paginilor

+incaracrea o data dupa merge rapid, acces rapid la orice pagina ca sunt deja in memorie

-incarcam pagini de care se poate sa nu avem nevoie, pornire lenta

* + incarcarea paginilor la nevoie

+incarcam doar pagini de care avem nevoie, pornire rapida

-merge lent, acces lent la paginile reincarcate

* prinicpiul vecinatatii: daca un process refera o pagina, e probabil sa refere curand paginile invecinate -> daca ni se cere o pagina aducem in memorie vecinatatea

**Politici de plasare**

* Metoda primei potriviri (**First-fit) -** partiţia solicitată este alocată în prima zonă liberă, în careîncape

+ simplitatea căutării de spaţiu liber

-fragmentare

* Metoda celei mai bune potriviri (**Best-fit**)- căutarea acelei zone libere, care lasă după alocare cel mai puţin spaţiu liber

+ economiseşte zonele de memorie mai mari

- timpul suplimentar de căutare + fragmentarea internă excesivă

- Metoda celei mai rele potriviri (**Worst-fit**) - căutarea acelei zone libere care lasă după alocare cel mai mult spaţiu liber.

+ după alocare rămâne un spaţiu liber mare

-fragmentare interna, timpul de căutare este mai mare ca la first-fit

* Metoda alocării prin camarazi (**Buddy-system**).

**ADRESARE INDIRECTA-B block size A address size**

* simplu bloc indirect B/A
* dublu bloc indirect (B/A)^2
* tripul bloc indirect (B/A)^3
* **i-noduri avand S link-uri directe astfel : 1 legare simplu-indirecta, 1 legare dublu-indirecta si 1 legare triplu-indirecta:** S \* B + A \* B + A^2 \* B + A^3 \* B