VII.3. Drivere

Ce sunt driverele?

- module de program care gestionează comunicarea cu perifericele
  - specializate - câte un driver pentru fiecare periferic
- parte a sistemului de operare
  - acces direct la hardware
  - se execută în modul nucleu al procesorului

Utilizare

- nu fac parte din nucleu
  - dar se află sub comanda nucleului
- folosite de rutinele de tratare ale întreruperilor hardware
- înlocuire periferic → înlocuire driver
  - organizare modulară
  - nu trebuie reinstalat tot sistemul de operare

VII.4. Gestiunea proceselor

Procese (1)

- se pot lansa în execuţie mai multe programe în acelaşi timp (*multitasking*)
- paralelismul nu este real
  - doar dacă sistemul are mai multe procesoare
  - altfel - concurenţă
- un program se poate împărţi în mai multe secvenţe de instrucţiuni - *procese*
  - se pot executa paralel sau concurent

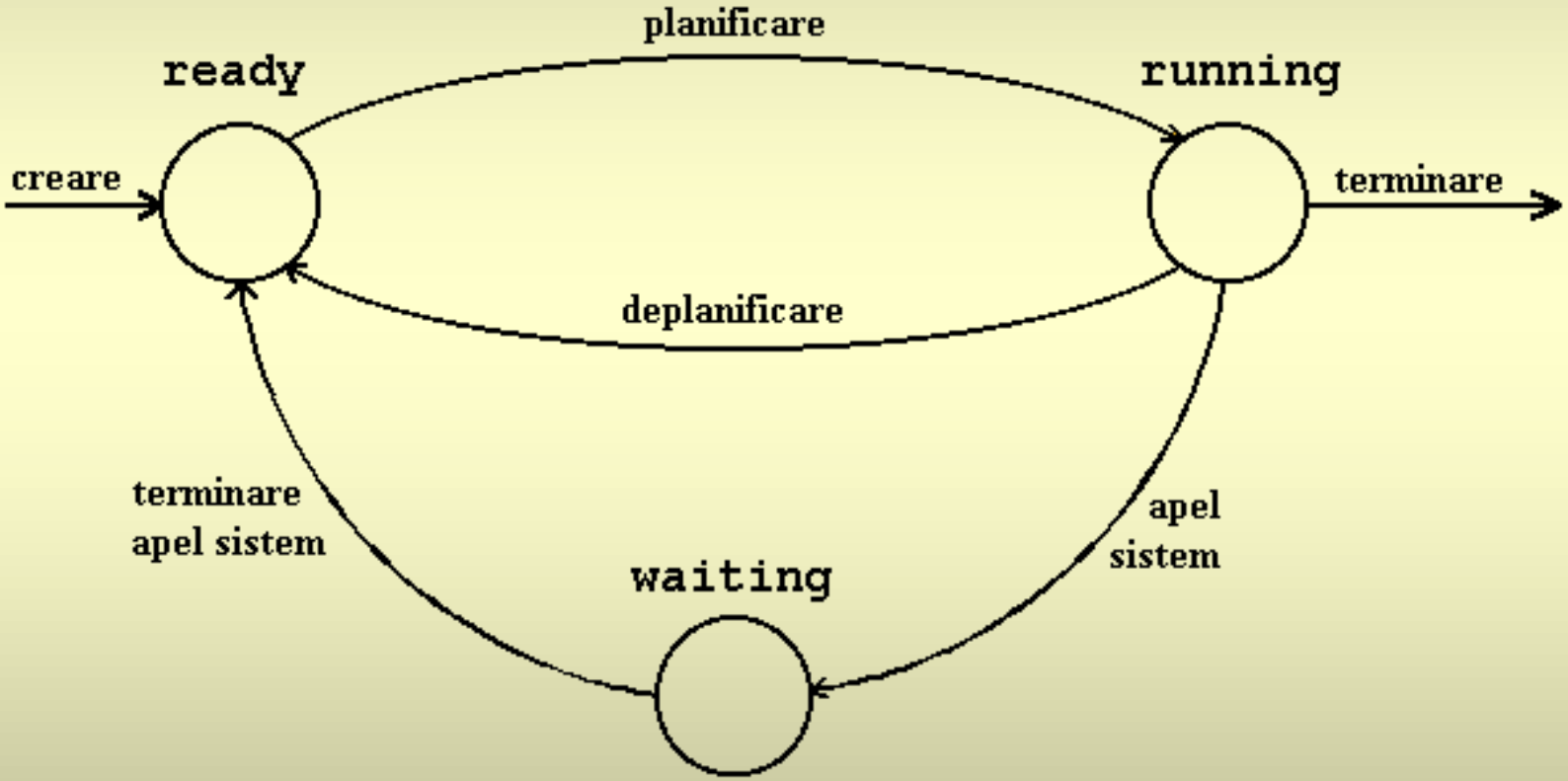
Procese (2)

- sistemul de operare lucrează cu procese
  - nu cu programe
- la lansare, un program constă dintr-un singur proces
  - poate crea alte procese
  - care pot crea alte procese ş.a.m.d.
- un procesor poate executa la un moment dat instrucţiunile unui singur proces

Stările unui proces (1)

- în execuţie (*running*)
  - instrucţiunile sale sunt executate de procesor
- gata de execuţie (*ready*)
  - aşteaptă să fie executat de procesor
- în aşteptare (*waiting*)
  - aşteaptă terminarea unui apel sistem
  - nu concurează momentan pentru planificarea la procesor

Stările unui proces (2)



Stările unui proces (3)

- procesul aflat în execuţie părăseşte această stare
  - la terminarea sa
    - normală sau în urma unei erori
  - la efectuarea unui apel sistem (→ *waiting*)
  - când instrucţiunile sale au fost executate un timp suficient de lung şi este rândul altui proces să fie executat (deplanificare)

Forme de multitasking

- non-preemptiv
  - nu permite deplanificarea unui proces
  - un proces poate fi scos din execuţie doar în celelalte situaţii
  - dezavantaj - erorile de programare pot bloca procesele (ex. buclă infinită)
- preemptiv

Deplanificarea

- cum ştie sistemul de operare cât timp s-a executat un proces?
- este necesară o formă de măsurare a timpului
- ceasul de timp real
  - dispozitiv periferic
  - generează cereri de întrerupere la intervale regulate de timp

VII.5. Gestiunea memoriei

Gestiunea memoriei

Funcţii

- alocarea zonelor de memorie către aplicaţii
- prevenirea interferenţelor între aplicaţii
- detectarea şi oprirea acceselor incorecte

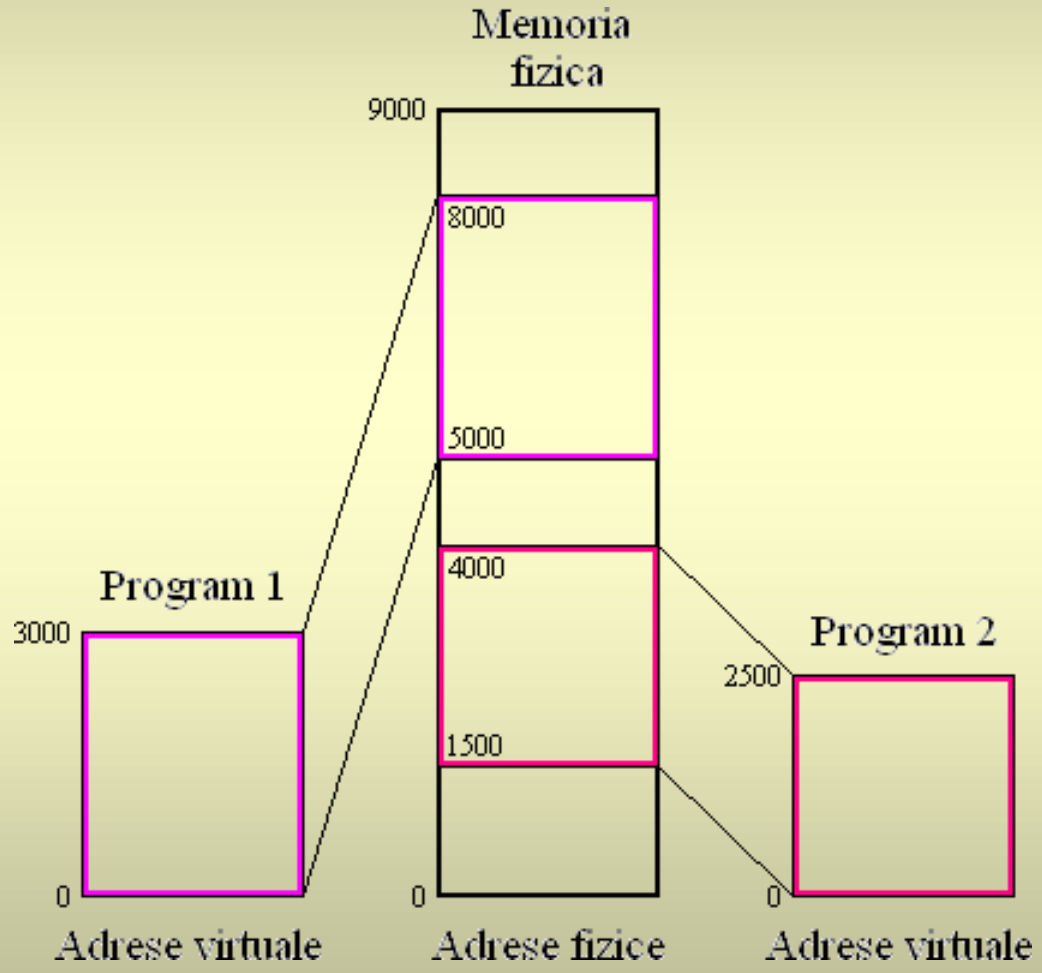
Problema fundamentală

- mai multe aplicaţii → zone de memorie disjuncte
- fiecare aplicaţie → anumite zone de memorie; care sunt aceste zone?
  - depind de ocuparea memoriei la acel moment
  - nu pot fi cunoscute la compilare

Soluţia

- două tipuri de adrese
  - virtuale - aplicaţia crede că le accesează
  - fizice - sunt accesate în realitate
- corespondenţa între adresele virtuale şi cele fizice - gestionată de sistemul de operare

Adrese virtuale şi fizice (1)



Adrese virtuale şi fizice (2)

Gestionarea adreselor fizice şi virtuale

- 2 metode diferite
  - segmentare
  - paginare
- pot fi folosite şi împreună
- componentă dedicată a procesorului - MMU (*Memory Management Unit*)

VII.5.1. Segmentarea memoriei

Principiul de bază (1)

- segment - zonă continuă de memorie
- conţine informaţii de acelaşi tip (cod, date etc.)
- vizibil programatorului
- adresa unei locaţii - formată din 2 părţi
  - adresa de început a segmentului
  - deplasamentul în cadrul segmentului (*offset*)

Principiul de bază (2)

- la rulări diferite ale programului, segmentele încep la adrese diferite
- efectul asupra adreselor locaţiilor
  - adresa de început a segmentului trebuie actualizată
  - deplasamentul - nemodificat
- problema este rezolvată numai parţial
  - dorim ca adresa să nu fie modificată deloc

Descriptori (1)

- descriptor de segment - structură de date pentru gestionarea unui segment
- informaţii reţinute
  - adresa de început
  - dimensiunea
  - drepturi de acces
  - etc.

Descriptori (2)

- descriptorii - plasaţi într-un tabel
- accesul la un segment - pe baza indicelui în tabelul de descriptori (selector)
- adresa virtuală - 2 componente
  - indicele în tabelul de descriptori
  - deplasamentul în cadrul segmentului
- adresa fizică = adresa de început a segmentului + deplasamentul

Descriptori (3)

- la rulări diferite ale programului, segmentele încep la adrese diferite
- efectul asupra adreselor locaţiilor
  - nici unul
  - trebuie modificată doar adresa de început a segmentului în descriptor
  - o singură dată (la încărcarea segmentului în memorie)
  - sarcina sistemului de operare

Accesul la memorie (1)

- programul precizează adresa virtuală
- identificare descriptor segment
- verificare drepturi acces
  - drepturi insuficiente - generare excepţie
- verificare deplasament
  - dacă deplasamentul depăşeşte dimensiunea segmentului - generare excepţie

Accesul la memorie (2)

- dacă s-a produs o eroare la paşii anteriori
  - rutina de tratare a excepţiei termină programul
- dacă nu s-a produs nici o eroare
  - calcul adresă fizică (adresă început segment + deplasament)
  - acces la adresa calculată

Exemplificare (1)

Tabel descriptori (simplificat)

Indice	Adresa început	Dimensiune
0	65000	43000
1	211000	15500
2	20000	30000
3	155000	49000
4	250000	35000

Exemplificare (2)

Exemplu 1:

```
mov byte ptr ds:[eax],25
```

- ds = 3 → adresa început segment = 155000
- eax = 27348 < 49000
  - deplasament valid (nu depăşeşte dimensiunea segmentului)
- adresa fizică: 155000 + 27348 = 182348

Exemplificare (3)

Exemplu 2:

```
add dword ptr ss:[ebp],4
```

- ss = 1 → adresa început segment = 211000
- ebp = 19370 > 15500
  - deplasament invalid (depăşeşte dimensiunea segmentului)
  - eroare → generare excepţie

Cazul Intel (1)

3 tabele de descriptori

- global (GDT - *Global Descriptor Table*)
  - accesibil tuturor proceselor
- local (LDT - *Local Descriptor Table*)
  - specific fiecărui proces
- de întreruperi (IDT - *Interrupt Descriptor Table*)
  - nu este direct accesibil aplicaţiilor

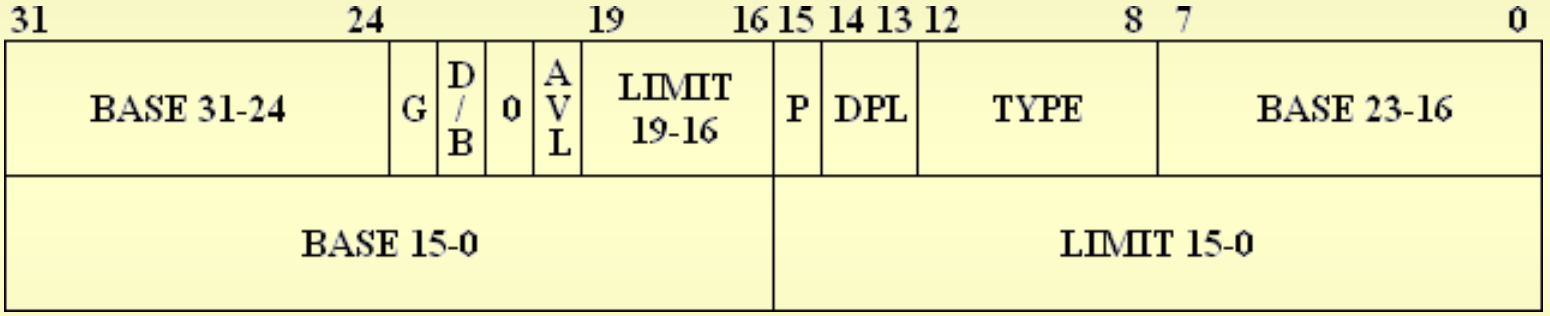
Cazul Intel (2)

Segmentele - accesate cu ajutorul selectorilor

Structura unui selector (16 biţi)

- primii 13 biţi - indicele în tabelul de descriptori
  - maximum 8192 descriptori/tabel
- 1 bit - tabelul folosit (global/local)
- ultimii 2 biţi - nivelul de privilegii
  - 0 - cel mai înalt, 3 - cel mai scăzut

Cazul Intel (3)



Cazul Intel (4)

- intervin nivelele de privilegii a 3 entităţi
  1. CPL (*Current Privilege Level*)
  – al procesului - reţinut de procesor
  2. RPL (*Requested Privilege Level*)
  – cel solicitat - preluat din selector
  3. DPL (*Descriptor Privilege Level*)
  – cel al segmentului accesat - din descriptor

Cazul Intel (5)

- relaţiile dintre aceste nivele de privilegii decid dacă se poate realiza accesul
- condiţia pentru realizarea accesului: CPL<=DPL şi RPL<=DPL (simultan)
- orice altă situaţie indică o încercare de acces la un nivel prea înalt
  - generare excepţie

Zone libere şi ocupate (1)

Problemă

- crearea unui segment nou → plasare în memorie
- este necesară o zonă liberă continuă suficient de mare
- pot exista mai multe asemenea zone - care este aleasă?

Zone libere şi ocupate (2)

Algoritmi de plasare în memorie

- *First Fit* - prima zonă liberă găsită suficient de mare
- *Best Fit* - cea mai mică zonă liberă suficient de mare
- *Worst Fit* - cea mai mare zonă liberă (dacă este suficient de mare)

Zone libere şi ocupate (3)

Fragmentare (1)

Fragmentarea externă a memoriei

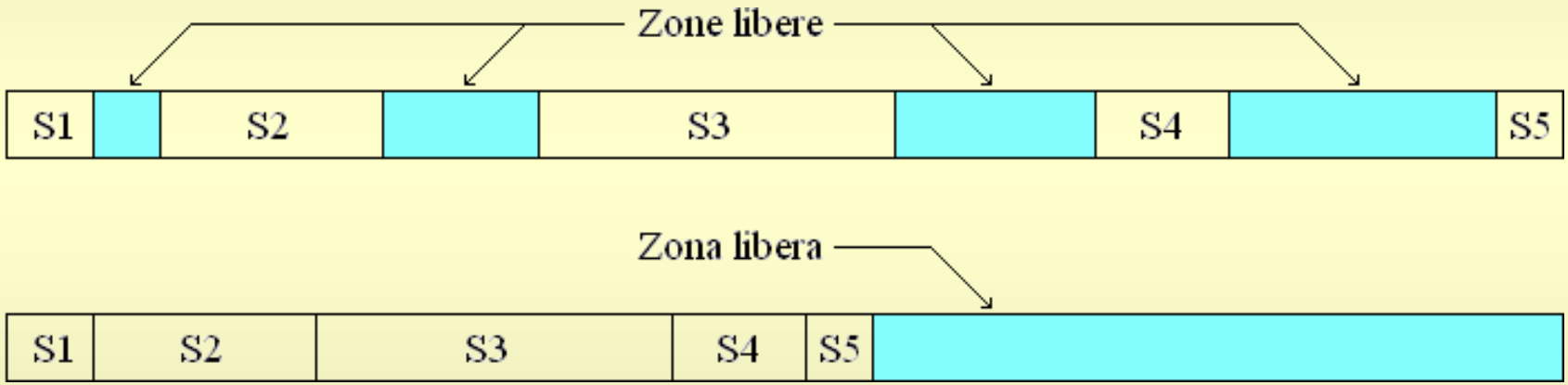
- multe zone libere prea mici pentru a fi utilizate
- apare după un număr mare de alocări şi eliberări de segmente
- indiferent de algoritmul folosit
- plasarea unui segment poate eşua, chiar dacă spaţiul liber total ar fi suficient

Fragmentare (2)

Eliminarea fragmentării externe

- compactarea memoriei
  - deplasarea segmentelor astfel încât să nu mai existe zone libere între ele
  - se crează o singură zonă liberă, de dimensiune maximă
  - realizată de un program specializat, parte a sistemului de operare

Compactare (1)



Compactare (2)

Rularea programului de compactare a memoriei

- consumă mult timp
  - mutarea segmentelor în memorie
  - actualizarea descriptorilor de segment
- nu poate fi rulat foarte des
- numai când este necesar

Compactare (3)

Situaţii în care se poate decide rularea programului de compactare

- când plasarea în memorie a unui segment eşuează din lipsă de spaţiu
- la intervale regulate de timp
- când gradul de fragmentare a memoriei depăşeşte un anumit nivel

Segmentarea - concluzii

Probleme ale mecanismului de segmentare

- gestiune complicată
  - suprapunerea segmentelor - greu de detectat
- fragmentarea externă - de obicei puternică
  - mult spaţiu liber nefolosit
- compactarea consumă timp

VII.5.2. Paginarea memoriei

Principiul de bază

- spaţiul adreselor virtuale - împărţit în pagini (*pages*)
  - zone de dimensiune fixă
- spaţiul adreselor fizice - împărţit în cadre de pagină (*page frames*)
  - aceeaşi dimensiune ca şi paginile
- dimensiune - uzual 4 Ko

Tabele de paginare (1)

- corespondenţa între pagini şi cadre de pagină - sarcina sistemului de operare
- structura de bază - tabelul de paginare
- câte unul pentru fiecare proces care rulează
- permite detectarea acceselor incorecte la memorie

Tabele de paginare (2)

- la rulări diferite ale programului, paginile sunt plasate în cadre diferite
- efectul asupra adreselor locaţiilor
  - nici unul
  - trebuie modificat tabelul de paginare
  - o singură dată (la încărcarea paginii în memorie)
  - sarcina sistemului de operare

Accesul la memorie

- programul precizează adresa virtuală
- se determină pagina din care face parte
- se caută pagina în tabelul de paginare
  - dacă nu este găsită - generare excepţie
- se determină cadrul de pagină corespunzător
- calcul adresă fizică
- acces la adresa calculată

Exemplificare (1)

Tabel de paginare (simplificat)

Pagini	0	1	2	8	9	11	14	15
Cadre de pagină	5	7	4	3	9	2	14	21

Exemplificare (2)

Exemplu 1:

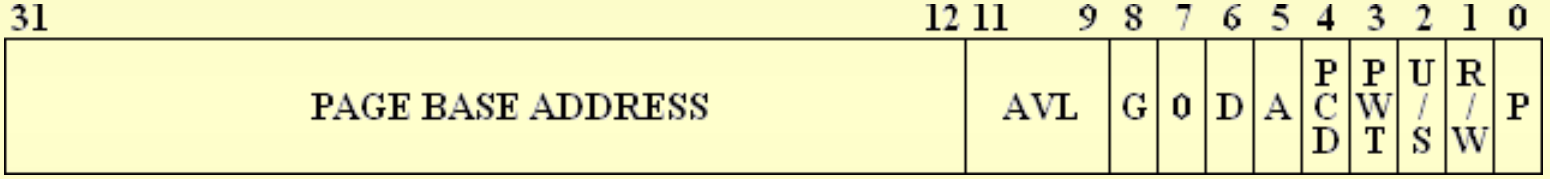
- dimensiunea paginii: 1000
- adresa virtuală: 8039
  - pagina: [8039/1000]=8 → cadrul de pagină 3
  - deplasament: 8039%1000=39 (în cadrul paginii)
- adresa fizică: 3·1000+39=3039

Exemplificare (3)

Exemplu 2:

- dimensiunea paginii: 1000
- adresa virtuală: 5276
  - pagina: [5276/1000]=5
  - nu apare în tabelul de paginare
  - eroare → generare excepţie

Cazul Intel



Restricţii

Construirea tabelelor de paginare

- sarcina sistemului de operare
- trebuie să evite suprapunerile între aplicaţii
- restricţii
  - o pagină virtuală poate să apară pe cel mult o poziţie într-un tabel de paginare
  - un cadru de pagină fizică poate să apară cel mult o dată în toate tabelele de paginare existente la un moment dat

Fragmentare (1)

Fragmentarea internă a memoriei

- între pagini nu există spaţiu → nu apare fragmentare externă
- fragmentarea internă
  - spaţiu liber nefolosit în interiorul unei pagini
  - nu poate fi preluat de altă pagină
  - nu se poate face compactare
- mai puţin severă decât fragmentarea externă

Fragmentare (2)

Alegerea dimensiunii paginilor

- putere a lui 2 (nu rămân resturi de pagină)
- odată aleasă, nu se mai schimbă
- se stabileşte ca un compromis
  - prea mare - fragmentare internă puternică
  - prea mică - mult spaţiu ocupat de tabelele de paginare
  - uzual - 4 Ko

Chiar atât de simplu?

Procesor pe 32 biţi

- spaţiu de adrese: 4 Go ($=2^{32}$)
- dimensiunea paginii: 4 Ko ($=2^{12}$)

$\rightarrow$ tabel cu 2^{20} elemente

  - ar ocupa prea mult loc
  - consumă din memoria disponibilă aplicaţiilor

- la procesoarele pe 64 biţi - şi mai rău

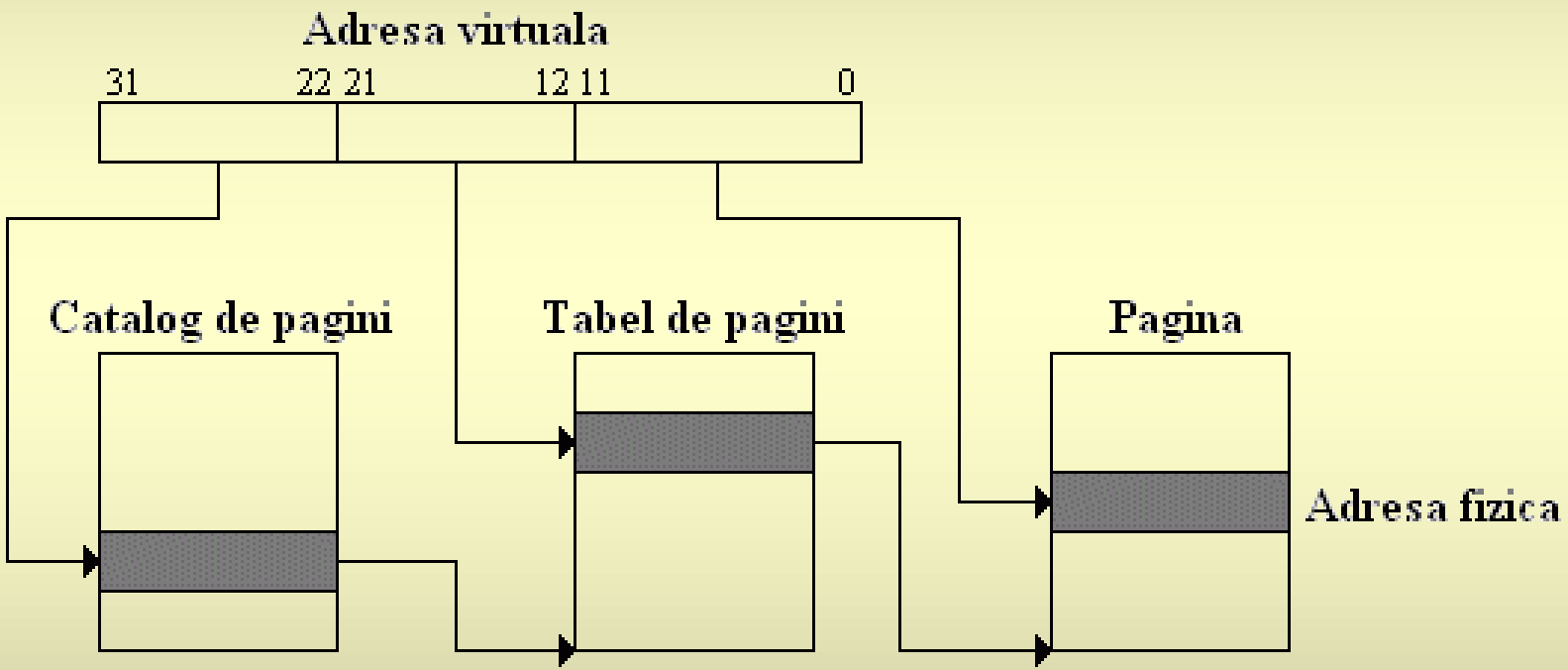
Soluţia 1

- tabele de pagini inversate
- nu se reţin toate cele 2^{20} elemente
  - doar paginile folosite
- plasare/căutare în tabel - funcţie *hash*
- greu de implementat în hardware
  - viteză
  - evitare coliziuni

Soluţia 2

- tabele pe mai multe nivele
- cazul Intel - 2 nivele
  - catalog de pagini (*Page Directory*)
  - tabel de pagini (*Page Table*)
- elementele din catalog - adrese de tabele de pagini
- se alocă doar tabelele de pagini folosite

Structura Intel



Performanţa (1)

- cataloagele şi tabelele de pagini se află în memorie
  - prea mari pentru a fi reţinute în procesor
  - prea multe - specifice fiecărui proces
- efect - performanţă scăzută
  - pentru fiecare acces la memorie solicitat de proces - 2 accese suplimentare
- soluţia - cache dedicat

Performanţa (2)

- TLB (*Translation Lookaside Buffer*)
  - în interiorul procesorului
  - reţine corespondenţe între pagini virtuale şi cadre de pagină fizice
  - ultimele accesate
  - trebuie invalidat atunci când se trece la execuţia altui proces