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

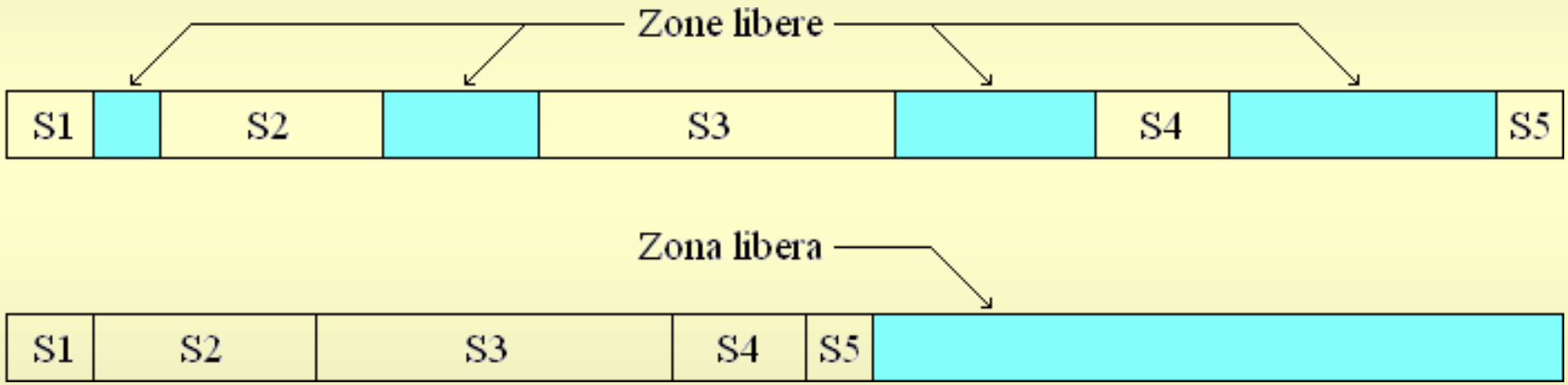
- multe zone libere prea mici pentru a fi utilizate
- apare după un număr mare de alocări şi eliberări de segmente
- indiferent de algoritmul folosit
- plasarea unui segment poate eşua, chiar dacă spaţiul liber total ar fi suficient

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

V.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 KB

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:

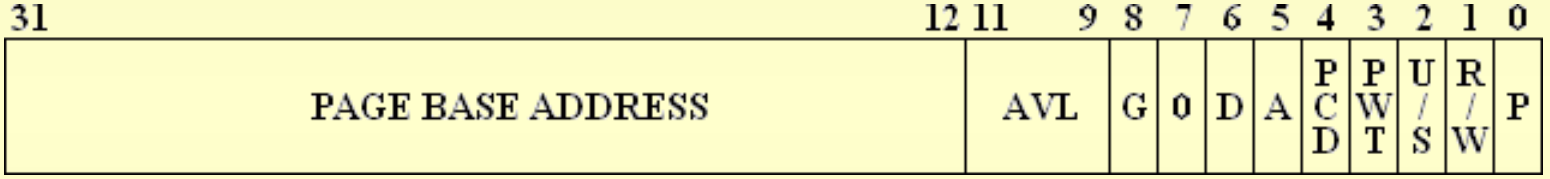
- dimensiunea paginii: 1000
- adresa virtuală: 8039
  - pagina: [8039/1000]=8 → cadrul de pagină 3
  - deplasament: 8039%1000=39 (în cadrul paginii)
- adresa fizică: 3·1000+39=3039

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 alt proces
  - 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 KB

Chiar atât de simplu?

Procesor pe 32 biţi

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

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

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

- la procesoarele pe 64 biţi - şi mai rău

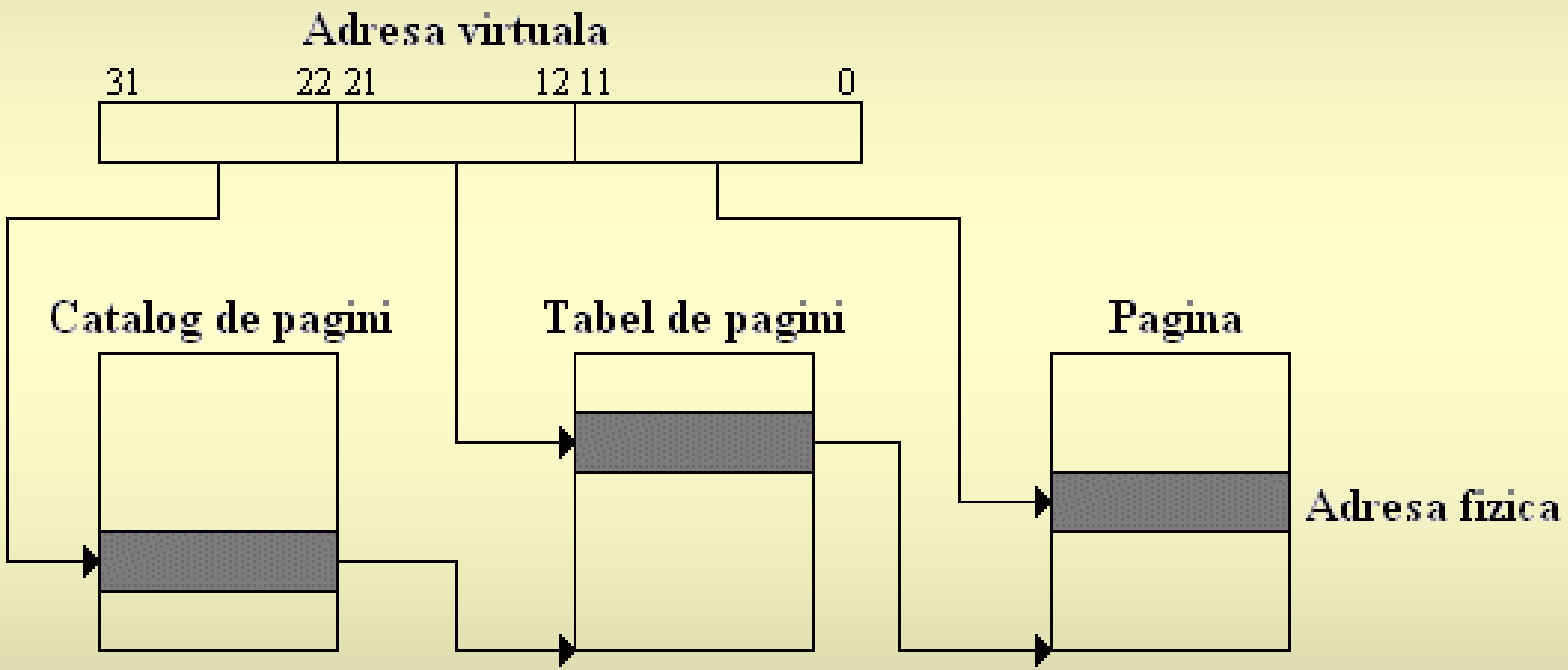
Soluţia 1

- 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

V.5.3. Memoria virtuală

Ideea de pornire

Problema

- aplicaţiile - consum mare de memorie
- memoria disponibilă - insuficientă

Cum se poate rezolva?

- capacitatea discului hard - foarte mare
- nu toate zonele de memorie ocupate sunt accesate la un moment dat

Memoria virtuală

Soluţia - memoria virtuală (*swap*)

- unele zone de memorie - evacuate pe disc
- când este nevoie de ele, sunt aduse înapoi în memorie

Cine gestionează memoria virtuală?

- sunt necesare informaţii globale
- sistemul de operare

Fişierul de paginare

- conţine zonele de memorie evacuate pe disc
- informaţii pentru regăsirea unei zone stocate
  - adresele din memorie
  - programul căruia îi aparţine
  - dimensiunea
  - etc.

Politica de înlocuire (1)

- problema - aceeaşi ca la memoria cache
- aducerea unei zone de memorie din fişierul de paginare implică evacuarea alteia
  - care?
- scop - minimizarea acceselor la disc
- politică ineficientă → număr mare de accese la disc → scăderea vitezei

Politica de înlocuire (2)

- set de lucru (*working set*) - zonele de memorie necesare programului la un moment dat
- uzual mult mai mic decât totalitatea zonelor folosite de program
- dacă încape în memorie - puţine accese la disc

Politica de înlocuire (3)

- se va selecta pentru evacuare zona care nu va fi necesară în viitorul apropiat
- nu se poate şti cu certitudine - estimare
  - pe baza comportării în trecutul apropiat
- paginare la cerere (*demand paging*) - evacuare pe disc numai dacă este strict necesar

Implementare

- prin intermediul mecanismelor de gestiune a memoriei, deja discutate
  - dacă un program încearcă să acceseze o locație aflată temporar pe disc, este necesar același tip de detecție
  - memoria virtuală poate fi folosită împreună atât cu segmentarea, cât şi cu paginarea
- rolul sistemului de întreruperi - sporit

Accesul la memorie (1)

Cazul paginării

1. programul precizează adresa virtuală
2. se determină pagina din care face parte
3. se caută pagina în tabelul de paginare
4. dacă pagina este găsită - salt la pasul 9
5. generare excepţie
6. rutina de tratare caută pagina în fişierul de paginare

Accesul la memorie (2)

Cazul paginării (cont.)

7. dacă pagina nu este în fişierul de paginare - programul este terminat
8. se aduce pagina în memoria fizică
9. se determină cadrul de pagină corespunzător
10. calcul adresă fizică
11. acces la adresa calculată

Reducerea acceselor la disc (1)

- duce la creşterea performanţei
- o pagină este salvată pe disc şi readusă în memorie de mai multe ori
- readucerea în memorie - copia de pe disc nu este ştearsă
- pagina şi copia sa de pe disc sunt identice până la modificarea paginii din memorie

Reducerea acceselor la disc (2)

- evacuarea unei pagini din memorie
  - dacă nu a fost modificată de când se află în memorie - nu mai trebuie salvată
  - util mai ales pentru paginile de cod
- este necesar sprijin hardware pentru detectarea acestei situaţii
  - este suficient să fie detectate operaţiile de scriere

Reducerea acceselor la disc (3)

- tabelul de paginare - structură extinsă
  - fiecare pagină are un bit suplimentar (*dirty bit*)
  - indică dacă pagina a fost modificată de când a fost adusă în memorie
  - resetat la aducerea paginii în memorie
- instrucţiune de scriere în memorie
  - procesorul setează bitul paginii care conţine locaţia modificată

V.5.4. Comunicarea între procese

Comunicare (1)

- pentru a putea coopera, procesele trebuie să-și poată transmite date
  - uneori volume mari
- implementare fizică - zone de memorie comune
  - variabile partajate
  - structuri de date mai complexe, prevăzute cu metode specifice de acces

Comunicare (2)

- este necesar ca două sau mai multe procese să acceseze aceeași zonă de memorie
  - același segment să apară simultan în tabelele de descriptori ale mai multor procese
  - același cadru de pagină să apară simultan în tabelele de paginare ale mai multor procese
- în oricare caz, sistemul de operare controlează zonele comune
  - iar procesele sunt conștiente de caracterul partajat al acestora

Excludere mutuală (1)

- accesul la o resursă comună poate dura
  - și poate consta în mai multe operaţii
- apare pericolul interferenţelor
- exemplu
  - un proces începe accesul la o variabilă comună
  - înainte de a termina, alt proces începe să o acceseze
  - variabila poate fi modificată în mod incorect

Excludere mutuală (2)

- accesul la o resursă comună - doar în anumite condiţii
- excludere mutuală
  - la un moment dat, un singur proces poate accesa o anumită resursă
- mecanisme de control
  - semafor - cel mai simplu
  - structuri partajate ale căror metode de acces asigură excluderea mutuală (ex. monitor)

Implementare

- accesul la o resursă poate fi controlat (și blocat) doar de către sistemul de operare
- deci orice formă de accesare a unei resurse partajate implică un apel sistem
- dacă se lucrează la nivel jos (variabile partajate + semafoare), este sarcina programatorului să se asigure că apelul este realizat corect

Fire de execuție - comunicare

- în cazul firelor de execuție ale aceluiași proces, variabilele globale sunt automat partajate
  - viteză mai mare
  - crește riscul erorilor de programare
- necesitatea excluderii mutuale este prezentă și aici

V.5.5. Utilizarea MMU

Hardware

Cazul Intel

- segmentarea
  - nu poate fi dezactivată
  - dar poate fi "evitată" prin software
- paginarea
  - poate fi activată/dezactivată

Sistemul de operare

Cazurile Windows, Linux

- segmentarea
  - nu este utilizată în practică
  - toate segmentele sunt dimensionate asfel încât să acopere singure întreaga memorie
- paginarea
  - pagini de 4 KB
  - Windows poate folosi şi pagini de 4 MB

Utilitatea MMU (1)

Avantaje

- protecţie la erori
- o aplicaţie nu poate perturba funcţionarea alteia
- verificările se fac în hardware
  - mecanism sigur
  - viteză mai mare

Utilitatea MMU (2)

Dezavantaje

- gestiune complicată
- memorie ocupată cu structurile de date proprii
  - tabelul de descriptori
  - tabelul de paginare
- viteză redusă - dublează numărul acceselor la memorie (sau mai mult)

Utilitatea MMU (3)

Concluzii

- scăderea de performanţă poate fi compensată folosind cache-uri
- procesoarele de azi oferă suficientă viteză
- sisteme multitasking - risc mare de interferenţe
- mecanismele MMU trebuie folosite