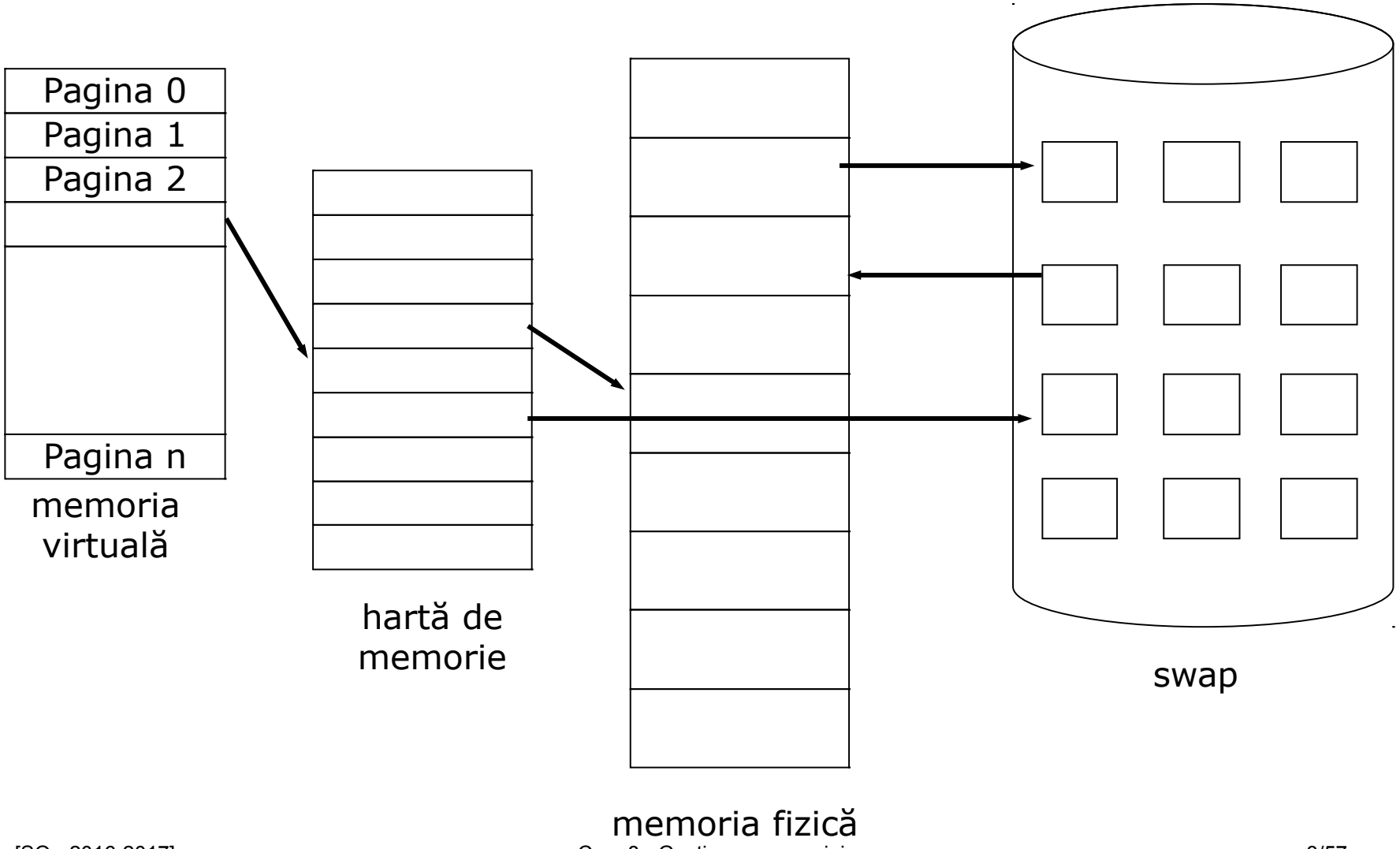
Sisteme de Operare

- Gestiunea memoriei
  - Memoria virtuală
  - Planificarea schimburilor cu memoria (înlocuirea paginii)
  - Algoritmi de înlocuire a paginii
  - Alocarea cadrelor
  - Studii de caz: Linux, Unix, Windows

Memoria virtuală

- Memoria virtuală este o tehnică ce permite execuţia proceselor, chiar dacă acestea nu se află integral în memorie.
- Un avantaj direct este acela al rulării unor programe cu dimensiuni mai mari decât cele ale memoriei fizice

Memoria virtuală



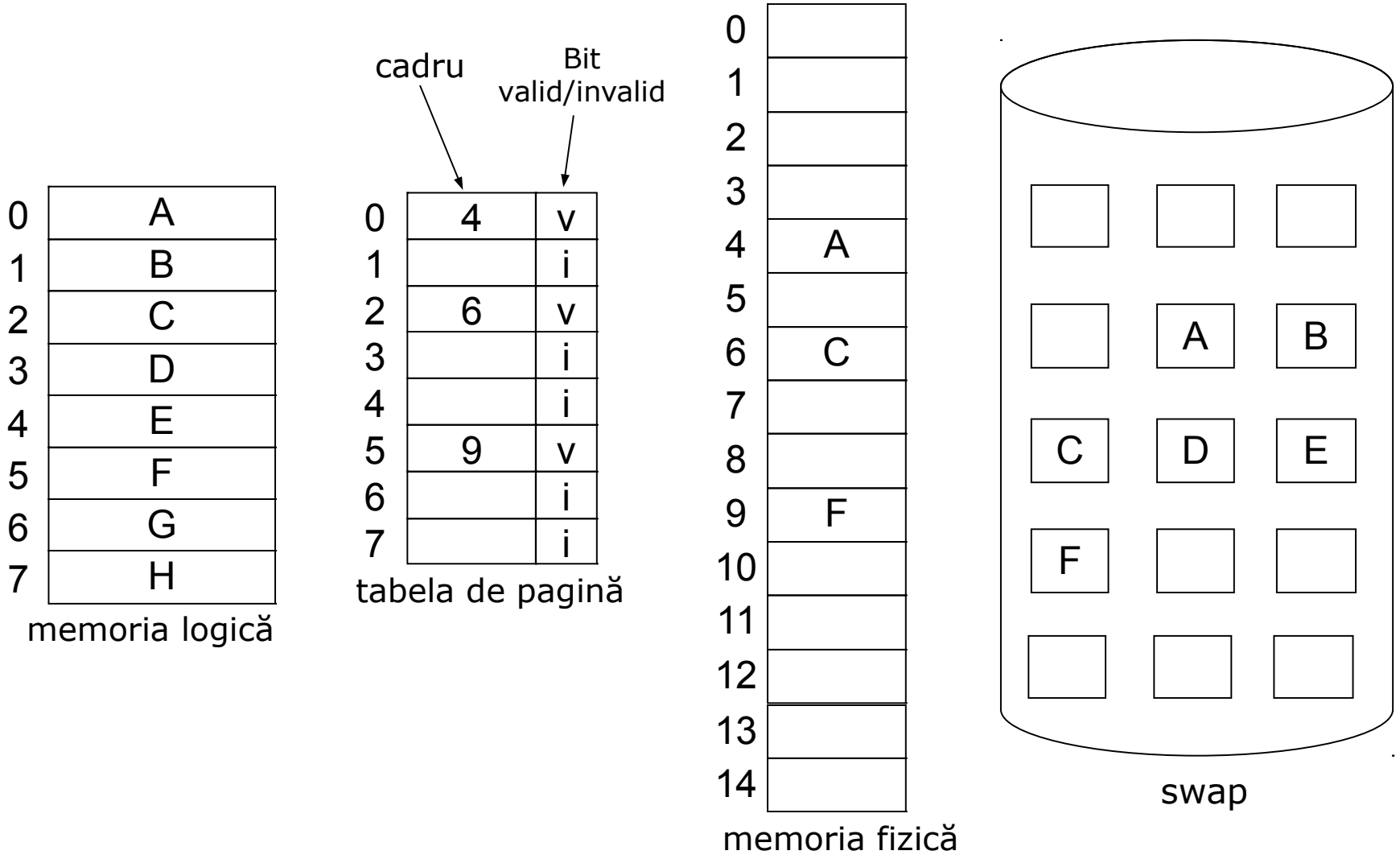
Implementarea memoriei virtuale

- în sistemele cu paginare prin metoda numită paginare la cerere, metodă ce poate fi folosită şi la sistemele cu segmentare sau cu paginarea segmentelor.
- O altă metodă este segmentarea la cerere, mult mai rar folosită, datorită faptului că algoritmii sunt mai complicaţi (din cauza mărimii variabile a segmentelor).

Paginarea la cerere

- se introduce în memorie întregul program, ci numai câteva pagini, atunci când sunt necesare
- Tabela de pagină corespunzătoare unui proces va conţine acelaşi tip de informaţii ca şi în cazul paginării obişnuite, având în plus un bit care va avea rolul de a semnala prezenţa sau absenţa din memorie a paginii la care se referă.

Tabela de pagină în cazul în care anumite pagini nu se află în memoria principală



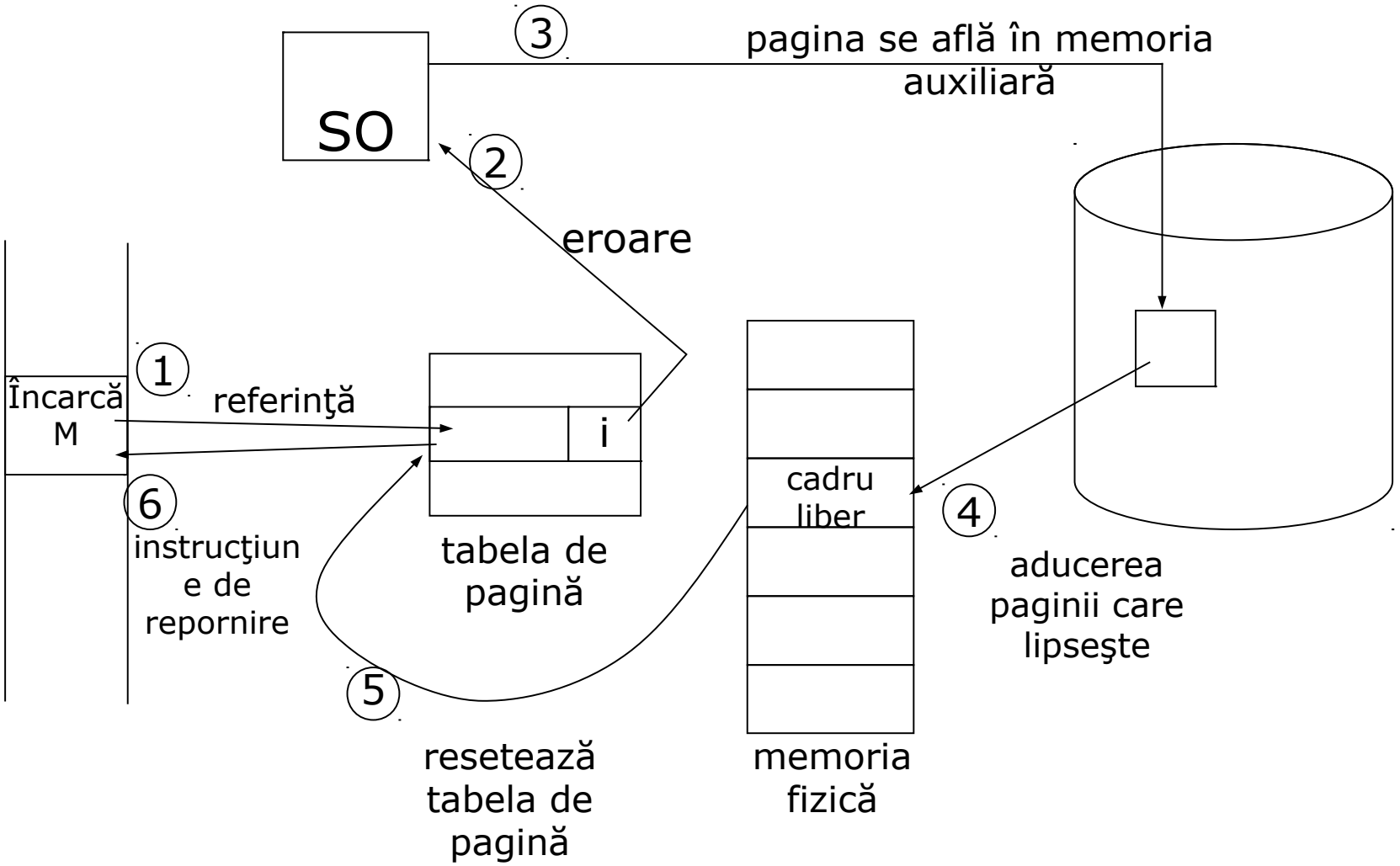
Eroarea de tip pagină lipsă

- Cât timp se folosesc paginile aflate în memorie, execuţia se desfăşoară normal.
- Dacă se încearcă folosirea unei pagini, care nu este încă în memorie, se va genera o **eroare de tip pagină lipsă**.
- Dacă se încearcă o folosire nepermisă a unei adrese de memorie (ex: indice incorect de vector) se generează o eroare ce duce la încheierea forţată a programului.

Etapele de rezolvare ale unei greşeli de pagină

- se verifică o tabelă internă a procesului (păstrată în blocul de control) pentru a stabili dacă încercarea de accesare este permisă sau nu (aparţine spaţiului de adrese logice asociate procesului); dacă este nepermisă programul se termină.
- dacă adresa este corectă şi pagina respectivă nu a fost adusă în memorie, trebuie încărcată din memoria auxiliară;
- se caută un cadru liber în memoria fizică
- se planifică discul pentru citirea paginii dorite în cadrul astfel alocat;
- odată cu încheierea operaţiei de citire se actualizează atât tabela internă asociată procesului, cât şi tabela de pagină, astfel încât să semnaleze apariţia în memorie a noii pagini.
- se reia execuţia instrucţiunii întrerupte de apariţia erorii de adresare.

Etapele de rezolvare ale unei greşeli de pagină



Tratarea paginii lipsă

- generarea erorii către sistemul de operare;
- se salvează regiştrii utilizator şi starea programului;
- se verifică dacă întreruperea este de tip pagină lipsă;
- se verifică dacă este corectă referirea paginii şi se determină poziţia ei pe disc;
- se iniţiază citirea de pe disc într-un cadru disponibil al memoriei interne;
- se aşteaptă în "coada" asociată discului până în momentul în care este satisfăcută cererea de citire;
- se aşteaptă datorită timpului de căutare şi/sau latenţă a discului;
- se începe transferul paginii în cadrul disponibil;
- în timpul aşteptării, UC poate fi alocată unui alt utilizator (planificarea UC);
- se salvează registrele şi starea programului celuilalt utilizator;
- se verifică dacă întreruperea a fost generată de către disc;
- se actualizează tabela de pagină şi celelalte tabele astfel încât să reflecte existenţa în memorie a noii pagini;
- se aşteaptă ca UC să fie alocată din nou procesului;
- se reface conţinutul registrelor utilizator, starea programului şi noua tabelă de pagină, după care se reia execuţia instrucţiunii întrerupte.

Planificarea schimburilor cu memoria (înlocuirea paginii)

- Datorită gradului ridicat de multiprogramare apare "supra-alocarea memoriei":
  - în timpul rulării unui program apare o eroare de tip "pagină lipsă"
  - sistemul de operare verifică tabelele interne pentru stabilirea cauzei (pagină lipsă sau încercare de accesare nepermisă a memoriei),
  - încearcă localizarea paginii pe disc, dar în urma consultării listei de cadre disponibile constată că toată memoria este ocupată (nu există nici un cadru liber).

Planificarea schimburilor cu memoria (înlocuirea paginii)

- Se pot lua următoarele decizii:
  - încheierea execuţiei programului
  - evacuarea unui program pe disc (cu eliberarea cadrelor alocate – se reduce gradul de multiprogramare)
  - folosirea metodei de înlocuire a paginii.

Metoda înlocuirii paginii

- dacă nu este nici un cadru liber, se caută şi se eliberează un cadru care nu este utilizat în acel moment prin memorarea pe disc a paginii conţinute şi se face modificarea tabelelor corespunzătoare pentru a indica faptul că pagina nu se mai află în memorie.
- cadrul eliberat este folosit pentru aducerea paginii care a fost solicitată în momentul apariţiei erorii de tip pagină lipsă.

Rutina de deservire a inlocuirii paginii

- localizarea paginii dorite pe discul magnetic;
- găsirea unui cadru liber;
- dacă există un cadru liber, el va fi folosit;
- altfel, cu ajutorul unui algoritm de înlocuire a paginii, se alege un cadru, se transferă pe disc pagina pe care o conţine şi se modifică tabelele de pagină şi cadru;
- se aduce în cadrul astfel eliberat pagina dorită şi se actualizează tabelele de pagină şi cadru;
- se reia execuţia programului utilizator.
- dacă nu există cadre libere sunt necesare două transferuri de pagină, ceea ce duce la dublarea timpului de deservire a erorii de tip pagină lipsă şi la mărirea timpului efectiv de acces

Rutina de deservire a inlocuirii paginii

- dacă nu există cadre libere?
  - sunt necesare două transferuri de pagină, ceea ce duce la dublarea timpului de deservire a erorii de tip pagină lipsă şi la mărirea timpului efectiv de acces
  - Pentru evitarea acestor efecte se asociază prin hardware fiecărei pagini un bit suplimentar, numit bit de modificare, care este modificat ori de câte ori este modificat conţinutul paginii (prin scrierea unui cuvânt sau a unui octet).
  - Când o pagină este aleasă pentru a fi înlocuită, se verifică mai întâi bitul de modificare asociat, şi dacă este setat înseamnă că în pagină a apărut cel puţin o modificare faţă de momentul în care a fost citită de pe disc. În acest caz, pagina trebuie salvată de pe disc cu noul ei conţinut.
  - Dacă bitul de modificare nu este setat (pagina nu a fost modificată) şi copia de pe disc nu a fost suprascrisă, se poate renunţa la salvarea ei. Astfel timpul de I/O se poate reduce la jumătate şi se diminuează durata de tratare a erorii de tip pagină lipsă.

Algoritmi de înlocuire a paginii

- Evaluarea unui algoritm se face prin executarea sa asupra unei anumite secvenţe de referiri la memorie şi prin calcularea numărului de erori de tip pagină lipsă apărute.
- Şirul de referiri la memorie se numeşte *şir de referinţă* şi poate fi generat (cu un generator de numere aleatoare) sau poate fi înregistrat prin urmărirea unui anumit sistem.

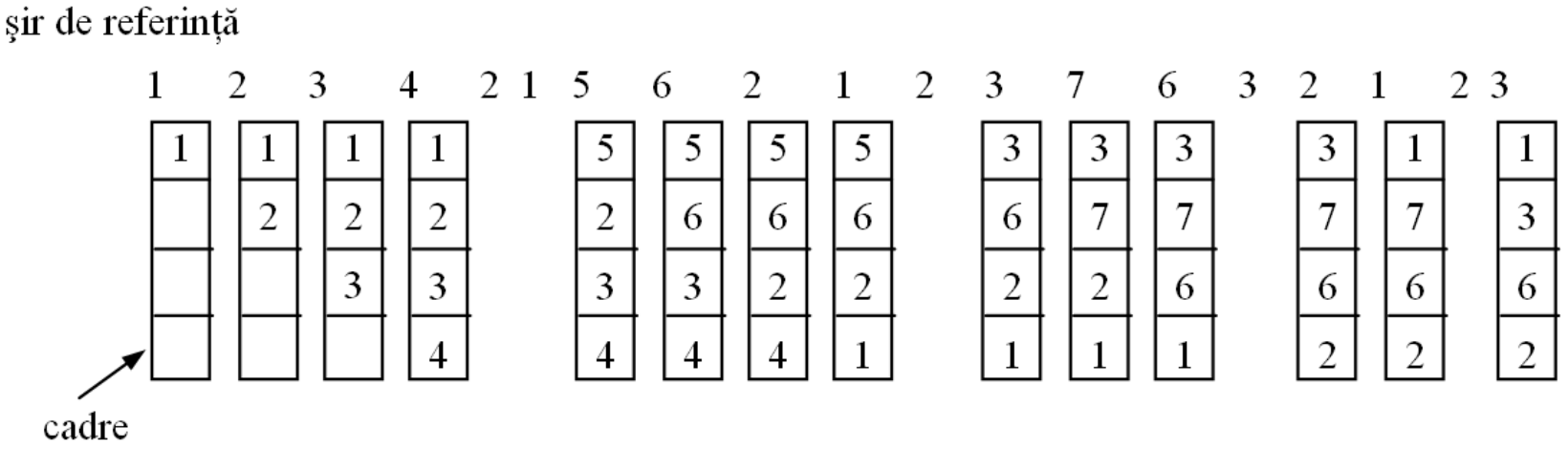
Algoritmi de înlocuire a paginii (2)

- Numărul de erori apărute în timpul execuţiei unui anumit algoritm de înlocuire a paginii asupra unui şir de referinţă depinde de numărul de cadre disponibile în memoria internă (dacă numărul cadrelor creşte scade numărul erorilor).
- Pentru algoritmii care vor fi studiaţi în continuare, se consideră că avem o memorie cu 4 cadre şi ca şirul de referinţă al adreselor este:
- 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

Algoritmul FIFO

- fiecărei pagini i se asociază momentul de timp la care a fost adusă în memorie şi atunci când este necesară o înlocuire se alege pentru inlocuire cea mai “veche” pagină.
- O variantă des utilizată este crearea unei cozi FIFO în care se păstrează toate paginile aduse în memorie, iar pagina inlocuita va fi tot timpul la începutul cozii (noile pagini vor fi plasate la sfârşitul cozii).

Algoritmul FIFO

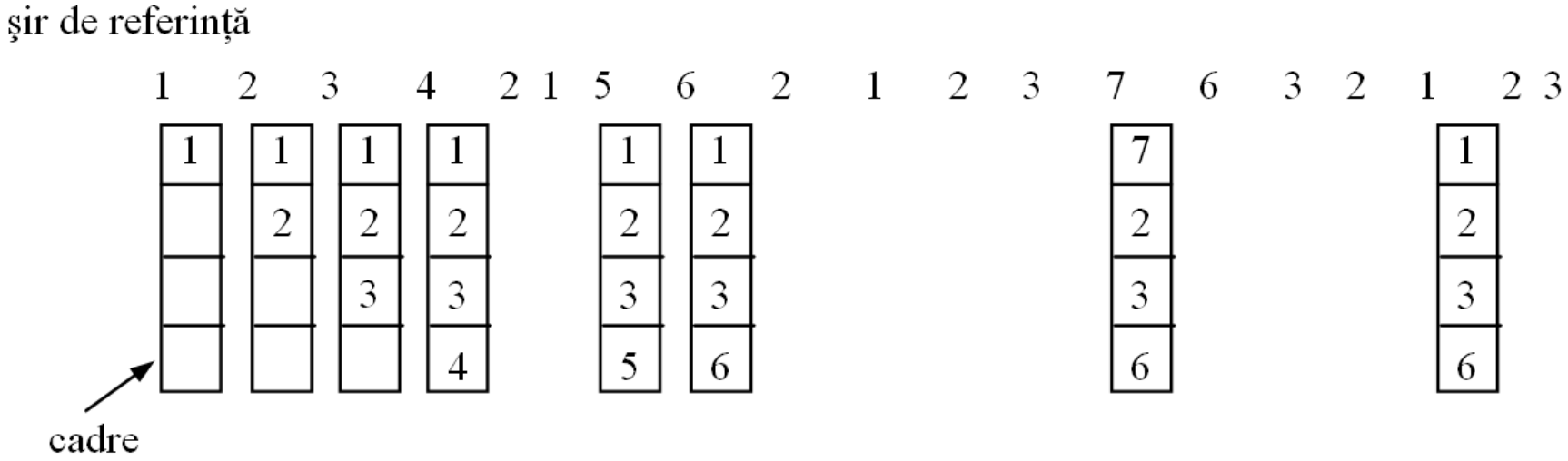


- FIFO este afectat de "**anomalia lui Belady**":
  - este posibil ca r**ata de apariţie a erorilor să crească odată cu creşterea numărului de cadre alocate şi nu să scadă cum ar fi fost normal**.
- În functie de sirul de referinţă este posibil ca numărul de erori de tip pagină lipsă pentru 4 cadre disponibile sa fie mai mare decât numărul corespunzător pentru 3 cadre.

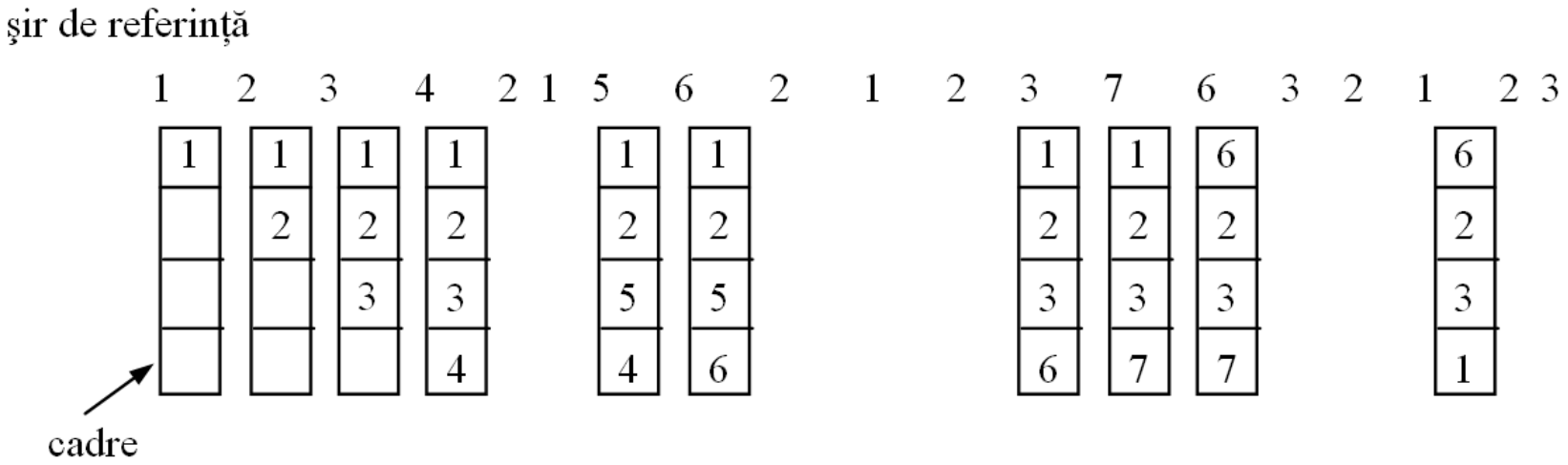
Algoritm de înlocuire optimală

- se alege pentru înlocuire pagina care a stat cel mai mult timp neutilizată (foloseşte momentul de timp la care este utilizată pagina, în timp ce FIFO foloseşte momentul la care a fost adusă pagina)
- Implementarea acestui tip de algoritm este dificilă deoarece trebuie cunoscut şirul de referinţă şi, din acest motiv, este folosit numai pentru realizarea unor studii comparative.

Algoritm de înlocuire optimală



Algoritmul LRU (Last Recently Used)



- asociază fiecărei pagini momentul de timp al ultimei utilizări.
- algoritmul alege pentru inlocuire pagina cu cea mai lungă durată de neutilizare.

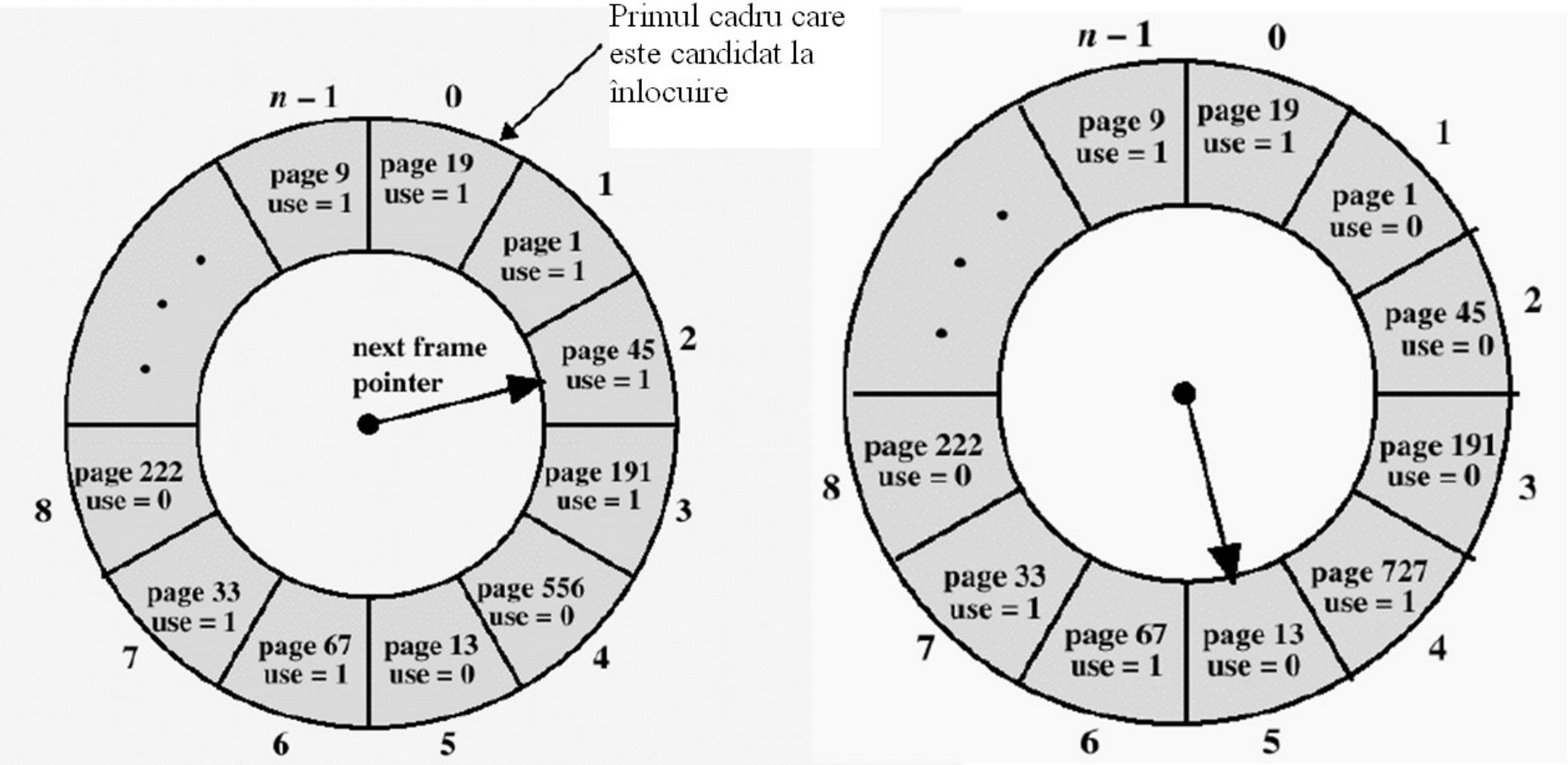
Algoritmul LRU (Last Recently Used)

- Deoarece este un algoritm de înlocuire optimală a paginii care examinează trecutul şi nu viitorul, este posibil ca o pagină să fie înlocuită chiar dacă ea urmează a fi referită din nou la momentul următor.
- Cu toate acestea, datorită numărului redus de erori de tip pagină lipsă, algoritmul LRU este mult mai performant decât algoritmul FIFO.
- Algoritmul nu este afectat de anomalia lui Belady, deoarece paginile deja încărcate în memorie reprezintă cele mai recent utilizate n pagini şi îşi păstrează caracteristica chiar şi în cazul creşterii numărului de cadre utilizate.

Algoritmul “a doua şansă” (Clock Algorithm)

- Algoritmul de bază este FIFO.
- În momentul verificării stării unei pagini, se inspectează bitul de referire asociat:
  - Dacă este 0, pagina va fi înlocuită;
  - dacă are valoarea 1, i se dă paginii o a doua şansă (bitul de referire asociat primeşte valoarea 0, iar timpul de sosire în memorie primeşte valoarea momentului curent), trecându-se (în ordinea FIFO) la verificarea stării următoarei pagini.
  - O pagină căreia i s-a acordat a doua şansă rămâne în memorie până în momentul în care toate celelalte pagini au fost înlocuite (sau au primit o a doua şansă).
- Dacă o pagină este folosită destul de des pentru ca valoarea bitului său să rămână 1, ea nu va fi înlocuită niciodată.
- Dacă toţi biţii au valoarea 1, algoritmul se transformă în unul de tip FIFO.

Algoritmul “a doua şansă” (Clock Algorithm)



Algoritmul LFU (Least Frequently Use – cea mai puțin frecvent utilizată):

- Acest algoritm înregistrează numărul de referiri corespunzătoare fiecărei pagini.
- Algoritmul va selecta pentru înlocuire pagina care are asociat cel mai mic număr de referiri.
- Principalul dezavantaj al acestei metode se observă în cazul în care la începutul programului, o pagină este intens folosită şi apoi nu mai este utilizată deloc, ea rămâne în memorie, deoarece numărul de referiri este foarte mare.

Algoritmul MFU (Most Frequently Use – cea mai frecvent utilizată):

- Acest algoritm consideră că pagina care are asociat cel mai mic număr de referiri este probabil cea mai recent introdusă în memorie şi este posibil ca programul să o folosească în continuare.
- Implementarea algoritmilor LFU şi MFU este foarte costisitoare şi din acest motiv sunt mai puţin folosiţi.

Clasificarea paginilor

- În funcţie de valorile pe care biţii de referire şi de modificare le pot lua, paginile se pot clasifica în:
  - (0,0) nefolosite şi nemodificate;
  - (0,1) nefolosite (recent), dar modificate;
  - (1,0) folosite, dar nemodificate;
  - (1,1) folosite şi modificate.

Algoritmi de înlocuire a paginii - alte tehnici

- Algoritmii de înlocuire a paginii pot fi utilizaţi împreună cu alte proceduri:
- sistemul păstrează o rezervă de cadre libere:
  - Când apare o eroare de tip pagină lipsă, se alege un cadru , dar, înainte ca acesta să fie evacuat din memorie, se citeşte pagina dorită într-un cadru liber.
  - Procedura permite programului să-şi reînceapă execuţia cât mai repede, fără a mai aştepta ca pagina aleasă să fie scrisă pe disc (după realizarea acestei operaţii, cadrul asociat paginii alese pentru înlocuire este ataşat rezervei de cadre)

Algoritmi de înlocuire a paginii - alte tehnici

- folosirea unei liste a paginilor modificate:
  - ori de câte ori dispozitivul de paginare este inactiv, se alege una dintre paginile modificate pentru a fi scrisă pe disc, după care i se setează bitul de modificare asociat.
  - Această "evacuare preventivă" face inutilă evacuarea paginii în momentul în care ea va fi aleasă pentru a fi înlocuită, crescând performanţele sistemului.

Algoritmi de înlocuire a paginii - alte tehnici

- folosirea unei rezerve de cadre libere şi a unor informaţii care să precizeze ce pagină corespundea fiecărui cadru înainte ca acesta să fie inclus în rezervă.
  - Deoarece conţinutul cadrului nu se modifică în urma scrierii pe disc, vechea pagină poate fi reutilizată direct din rezerva de cadre libere (dacă este necesar), fără folosirea altor operaţii de I/O (acest lucru este posibil doar dacă pagina respectivă nu a fost utilizată pentru memorarea altei pagini).

Alocarea cadrelor

- Indiferent de varianta de alocare a memoriei, pentru un sistem monoutilizator, unui program i se alocă oricare dintre cadrele disponibile.
- În cazul multiprogramării combinate cu paginarea la cerere, problemele de alocare trebuie să ia în considerare următoarele constrângeri:
  - nu pot fi alocate mai multe cadre decât există (cu excepţia cazului paginilor folosite în comun);
  - dacă numărul cadrelor alocate fiecărui proces este prea mic, rata de apariţie a erorii de tip pagină lipsă creşte;
  - arhitectura sistemului de calcul poate impune asigurarea unui număr minim de cadre alocate.

Metode de alocare

- Metoda primei potriviri (First-fit)
  - memoria solicitată este alocată în prima zonă în care încape; principalul avantaj este simplitatea căutării de spaţiu liber.
- Metoda celei mai bune potriviri (Best-fit)
  - se caută acea zonă liberă care după alocare lasă cel mai puţin spaţiu liber.
  - Avantaj: economiseşte zonele de memorie.
  - Dezavantaj:
    - timp suplimentar de căutare şi proliferarea blocurilor libere de lungime mică (fragmentare internă excesivă)
    - poate fi eliminat parţial dacă lista de spaţii libere este păstrată nu în ordinea crescătoare a adreselor, ci în ordinea crescătoare a lungimii spaţiilor libere.

Metode de alocare

- Metoda celei mai rele potriviri (Worst-fit)
  - se caută zonele libere care după alocare lasă cel mai mult spaţiu liber.
  - Deşi fragmentarea internă nu evoluează foarte rapid, timpul de căutare este mai mare decât cel de la metoda primei potriviri.
  - Şi aici este posibil ca lista spaţiilor libere să nu fie păstrate în ordinea crescătoare a adreselor, ci în ordinea crescătoare a lungimii spaţiilor libere.

Metode de alocare

- Algoritmul Buddy-system
  - această metodă exploatează reprezentarea binară a adreselor şi faptul că din raţiuni tehnologice, dimensiunea memoriei interne este un multiplu al unei puteri a lui doi.
  - Notăm cu **n** cea mai mare putere a lui 2 prin care se poate exprima dimensiunea memoriei interne şi cu **m** puterea lui 2 care defineşte unitatea de alocare a memoriei.
  - Dimensiunea spaţiilor ocupate şi a celor libere sunt de forma $\mathbf{2^k}$, unde $\mathbf{m \leq k \leq n}$.
  - Ideea principală este de a păstra liste separate de spaţii libere pentru fiecare dimensiune $\mathbf{2^k}$.
  - Astfel, vom avea **n-m+1** liste de spaţii disponibile. Astfel, fiecare spaţiu liber sau ocupat de dimensiune $\mathbf{2^k}$ are adresa de început un multiplu de $\mathbf{2^k}$.

Metode de alocare
Algoritmul Buddy-system

- Definiţie:
  - două spaţii libere de ordinul **k** se numesc camarazi (Buddy) de ordin **k**, dacă adresele lor A1 şi A2 verifică relaţiile:
    - **A1 < A2, A2 = A1 + 2^k** şi **A1 mod 2^{k+1}= 0**,

      sau
    - **A2 < A1, A1 = A2 + 2^k** şi **A2 mod 2^{k+1}= 0**.
  - dacă într-o listă de ordin **k** apar doi camarazi, sistemul îi concatenează într-un spaţiu de dimensiune **2^{k+1}**.

Metode de alocare
Algoritmul Buddy-system

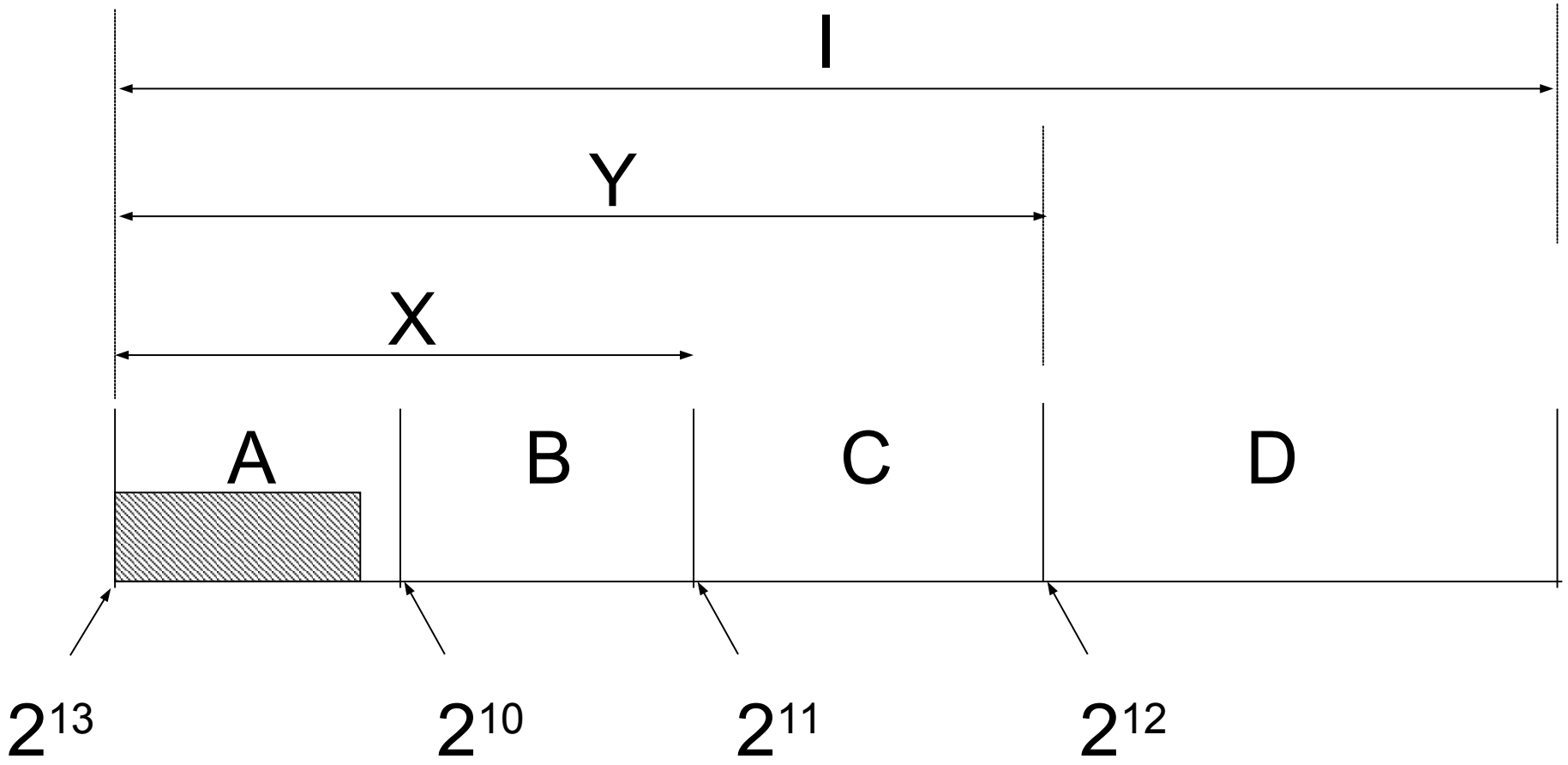
- Algoritmul de alocare este următorul:
  - se determină cel mai mic număr **p**, **m≤p≤n** pentru care numărul de octeţi solicitaţi verifică relaţia **o≤2^p** .
  - se caută, în această ordine, în listele de ordin **p**, **p+1**, **p+2**, **… n** o zonă liberă, de dimensiune cel puţin **o**.
  - dacă se găseşte o zonă de ordin **p**, atunci aceasta este alocată şi se şterge din lista de ordinul **p**.
  - dacă se găseşte o zonă de ordin **k>p**, atunci se alocă primii **2^p** octeţi, se şterge zona din lista de ordin **k** şi se creează, în schimb, alte **k-p** zone libere cu dimensiunile: **2^p, 2^{p+1}, …, 2^{k-1}**.

Metode de alocare
Algoritmul Buddy-system

- Se doreşte alocarea a 1000 octeţi (p = 10)
- Nu s-au găsit zone libere nici de dimensiune $\mathbf{2^{10}}$, nici $\mathbf{2^{11}}$ şi nici $\mathbf{2^{12}}$.
- Prima zonă liberă este de dimensiune $\mathbf{2^{13}}$ şi o notăm cu **I**.
- Ca rezultat al alocării a fost ocupată zona **A** de dimensiune $\mathbf{2^{10}}$ şi au fost create încă trei zone libere:
  - **B** de dimensiune $\mathbf{2^{10}}$
  - **C** de dimensiune $\mathbf{2^{11}}$
  - **D** de dimensiune $\mathbf{2^{12}}$
- Zonele **B**, **C** şi **D** vor trece în listele de ordine **10**, **11** şi **12**, iar zona **I** va fi ştearsă din lista de ordin **13**.

Metode de alocare
Algoritmul Buddy-system

- Alocarea şi eliberarea memoriei



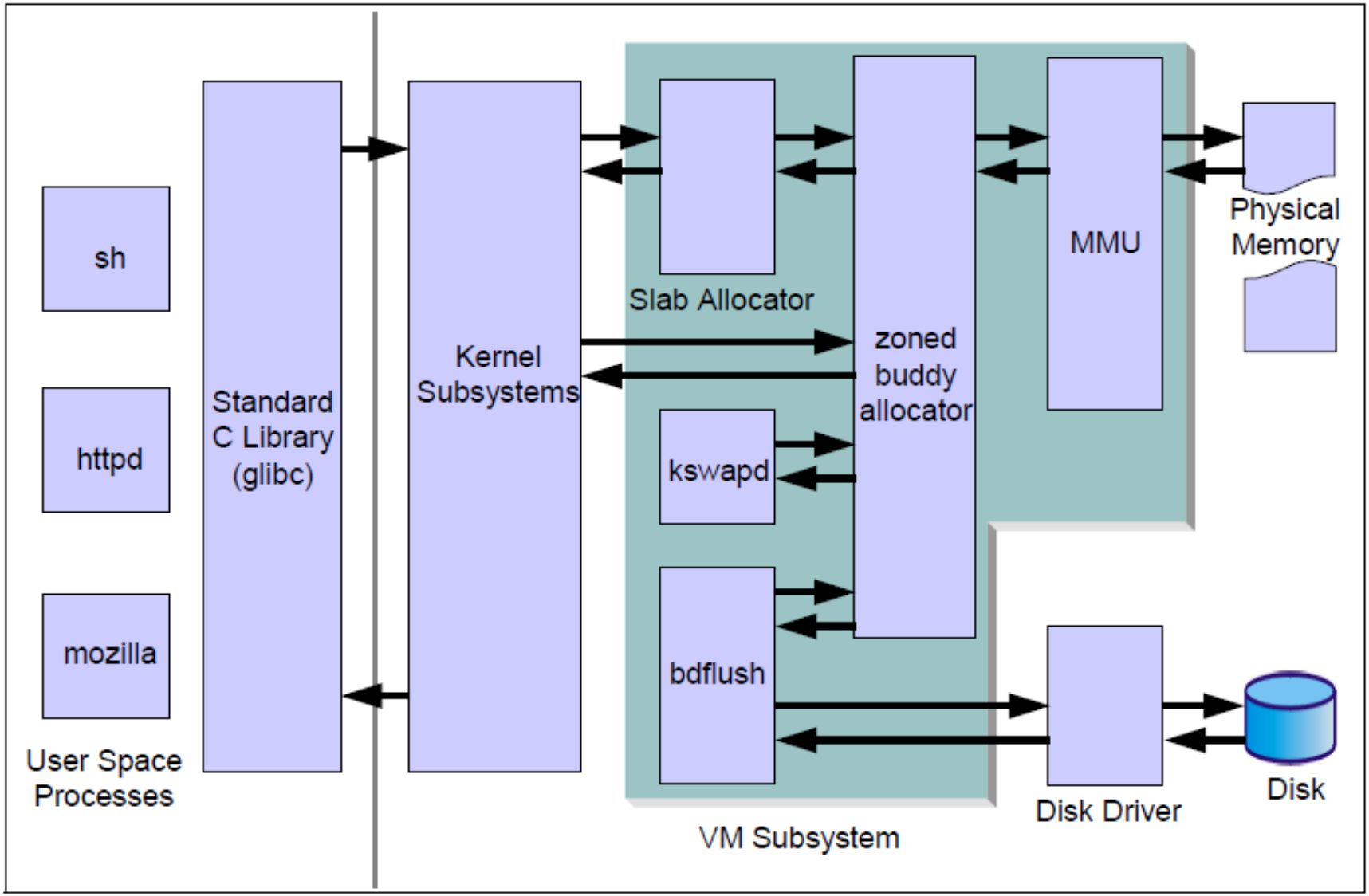
Metode de alocare
Algoritmul Buddy-system

- Algoritmul de eliberare a memoriei:
  1. se introduce zona respectivă în lista de ordin **p**.
  2. se verifică dacă zona eliberată are un camarad de ordin **p**.
     - Dacă da, atunci zona este comasată cu acest camarad şi formează împreună o zonă liberă de dimensiune $\mathbf{2^{p+1}}$.
     - Atât zona eliberată, cât şi camaradul ei se şterg din lista de ordin **p**, iar zona nou apărută va trece în lista de ordin **p+1**.
  3. Se execută pasul 2 în mod repetat, mărind de fiecare dată **p** cu o unitate, până când nu se mai pot face comasări.

Metode de alocare
Algoritmul Buddy-system

- Exemplu:
  - Presupunem că sunt libere zonele A, C şi D, iar zona B este ocupată.
  - Se eliberează zona B şi se execută paşii:
    - se trece zona B în lista de ordin 10;
    - se depistează că zonele A şi B sunt camarazi. Ca urmare, cele două zone vor fi comasate şi se va forma o zonă X care va fi trecută în lista de ordin 11, iar zonele A şi B se şterg din lista de ordin 10.
    - se depistează că zonele X şi C sunt camarazi. Ca urmare ele vor fi comasate, formând zona Y, care va fi trecută în lista de ordin 12. X şi C vor fi şterse din lista de ordin 11.
    - se găseşte că Y şi C sunt camarazi şi vor fi comasate, formând zona I, ce va fi trecută în lista de ordin 13, iar Y şi C vor fi şterse din lista de ordin 12.

Studiu de caz
Linux - Memoria virtuală



Studiu de caz
Linux - Memoria virtuală

- Adresarea memoriei virtuale:
- Sistemul de operare Linux foloseşte o structură de tabelă cu trei niveluri ce constau în (fiecare tabela are dimensiunea unei pagini):
  - **Page directory**: un proces activ are o singură pagină director care este de dimensiunea unei pagini. Fiecare intrare în pagina director pointează către o pagină din page middle directory. Pagina director trebuie să fie în memoria principală pentru un proces activ.
  - **Page middle directory**: fiecare intrare pointează către o pagină din tabela de pagini.
  - **Page table**: fiecare intrare pointează către o pagină virtuală a procesului.

Studiu de caz
Linux - Memoria virtuală

- o adresă virtuală în Linux este văzută ca având patru câmpuri:
  - cel mai semnificativ (din stânga) este utilizat ca index în page directory;
  - următorul este index în page middle directory;
  - al treilea câmp este index în page table
  - ultimul este offset-ul în pagina de memorie selectată.

Studiu de caz
Linux - Memoria virtuală

- Această structură este independentă de platformă şi a fost proiectată pentru procesoarele Alpha de 64 biţi care oferă suport pentru 3 niveluri de paginare.
- Pentru procesoarele Pentium pe 32 biţi şi x86 care au implementat un mecanism hardware de paginare cu două niveluri, Linux defineşte dimensiunea **page middle directory** ca fiind 1.

Studiu de caz
Linux - Alocarea paginilor

- este folosit algoritmul Buddy.
- Kernelul menţine o listă care conţine un grup de pagini contigue de dimensiune fixă iar fiecare grup poate consta în 1, 2, 4,8, 16 sau 32 pagini.
- Pe măsură ce paginile sunt alocate şi eliberate din memorie, grupurile disponibile sunt împărţite şi regrupate folosind algoritmul buddy.

Studiu de caz
Linux - Înlocuirea paginii

- Algoritmul folosit de Linux pentru înlocuirea paginii este bazat pe algoritmul "a doua şansă" (clock algorithm).
- În cazul Linux, bitul suplimentar de utilizare este înlocuit cu o variabilă de 8 biţi.
- De fiecare dată când o pagină este accesată, această variabilă este incrementată.
- În paralel, Linux parcurge lista paginilor şi decrementează variabilele pentru fiecare pagină.
- Dacă o pagină are variabila egală cu 0, atunci este considerat ca fiind cel mai bun candidat pentru înlocuire.
- O valoare mare a variabilei indică faptul că pagina este folosită frecvent şi este mai puţin luată în considerare la o posibilă înlocuire. De aceea, algoritmul folosit de Linux este considerat ca fiind o formă de LFU (Least Frequently Used).

Studiu de caz
Linux - Alocarea memoriei pentru kernel

- Alocarea memoriei pentru kernel este bazată pe mecanismele de alocare a memoriei virtuale utilizator.
- Este folosit algoritmul Buddy pentru a aloca sau elibera unităţi de una sau mai multe pagini.
- Deoarece cantitatea de memorie minimă alocată în acest mod este de o pagină, algoritmul va fi ineficient deoarece kernelul are nevoie de zone de memorie de dimensiune impară pentru un timp foarte scurt.
- Pentru rezolvarea acestei probleme, Linux foloseste *slab allocation* (alocare pe bucăţi (plăci) sau pe porţiuni) în cadrul unei pagini alocate.
- Pentru arhitectura Pentium/x86, pagina este de 4 kbytes iar bucăţile alocate pot fi de 32, 64, 128, 252, 508, 2040 sau 4080 bytes. În esenţă, Linux păstrează un set de liste, câte una pentru fiecare bucată memorie. Zonele alocate pot fi gestionate după modelul algoritmului Buddy.

Studiu de caz
Unix / Solaris

- Deoarece este un sistem de operare independent de platformă, managementul memoriei variază de la un sistem la altul.
- Primele versiuni de UNIX nu foloseau memorie virtuală.
- Implementările curente, inclusiv SVR4 şi Solaris 2.x, folosesc memorie virtuală paginată.
- De fapt, în SVR4 şi Solaris 2.x, sunt folosite două scheme de management a memoriei.

Studiu de caz
Unix / Solaris

- Sistemul cu paginare oferă posibilitatea memoriei virtuale de a aloca pagini atât în memoria principală pentru procese, cât şi alocarea paginilor pentru bufferele blocurilor de pe disc.
- Deşi este o schemă eficientă de alocare atât pentru procesele utilizator, cât şi pentru operaţiile de I/O, este mai puţin potrivită pentru alocarea memoriei kernelului.

Studiu de caz
Unix / Solaris - Alocarea paginilor

- **Disk block descriptor**: asociată fiecărei pagini a procesului o intrare această tabelă descrie copia de pe disc a memoriei virtuale.

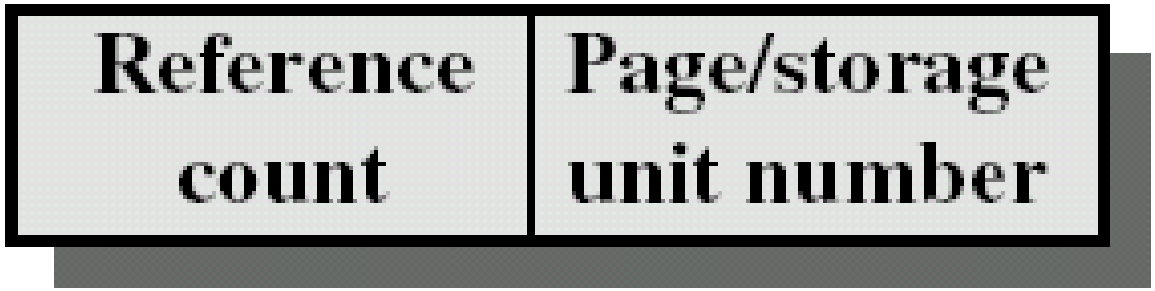
Swap device number	Device block number	Type of storage

- **Page frame data table**: Descrie fiecare zonă din memoria principală şi este indexată după numărul zonei.

Page state	Reference count	Logical device	Block number	Pfdata pointer

Studiu de caz
Unix / Solaris - Alocarea paginilor

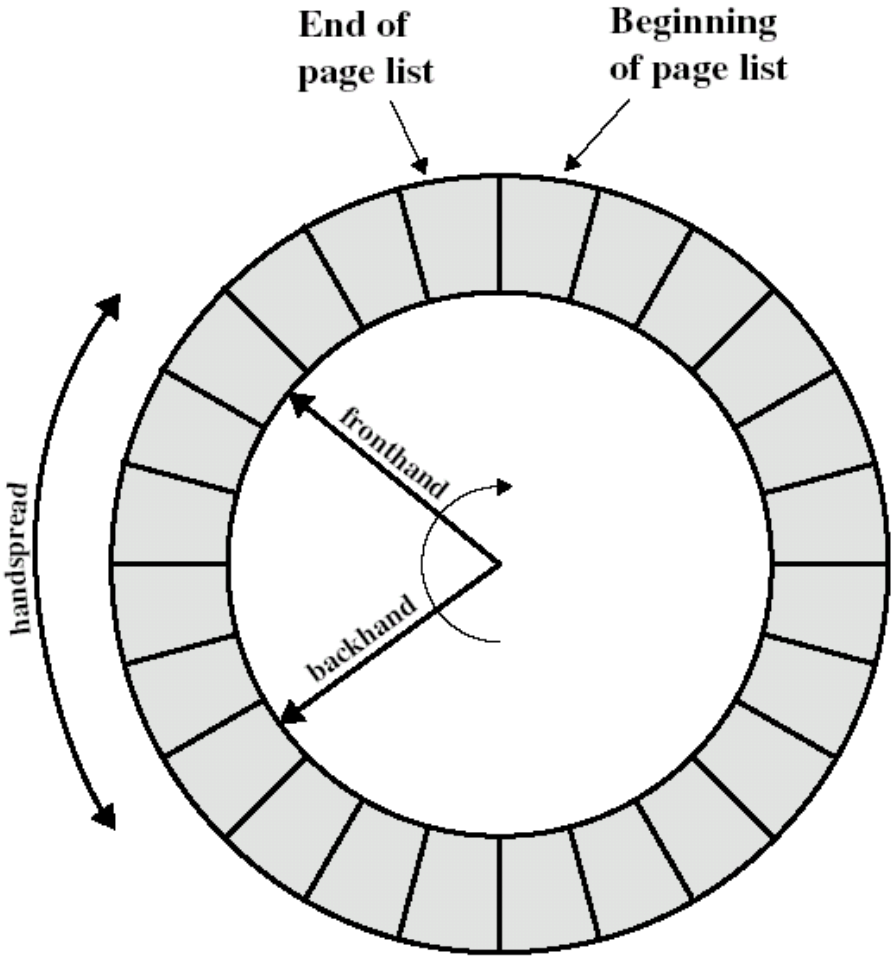
- **Swap-use table**: Este folosită câte o tabelă de acest tip pentru fiecare dispozitiv de păstrare a swap-ului, cu câte o intrare pentru fiecare pagină.



Studiu de caz
Unix / Solaris - Înlocuirea paginilor

- Structura page frame data table este folosită pentru înlocuirea paginilor.
- Algoritmul folosit în cazul SVR4 este o variantă a algoritmului "a doua şansă".
- Algoritmul foloseşte bitul de referinţă pentru fiecare pagină care poate fi trecută în swap.
  - Bitul este setat pe 0 când este adusă în memorie prima dată şi este setat pe 1 când pagina este folosită pentru citire sau scriere.
- Pe de altă parte, are loc o parcurgere în sens invers pentru verificarea bitului de referinţă.
  - Dacă este 1, atunci pagina respectivă este ignorată, iar dacă este 0 atunci înseamnă că pagina nu a fost referită între cele două parcurgeri şi este posibil să fie înlocuită.

Studiu de caz
Unix / Solaris - Înlocuirea paginilor



- Doi parametri caracterizează funcţionarea acestui algoritm:
  - Scanrate: numărul de treceri peste lista de pagini în cele două sensuri, măsurat în pagini pe secundă.
  - Handspread: intervalul dintre cele două parcurgeri.
- Aceşti doi parametri au valori diferite la bootare în funcţie de memoria disponibilă.
- Împreună determină durata de utilizare a unei pagini înainte de a fi trecută în swap datorită faptului că nu este utilizată.

Studiu de caz
Unix / Solaris - Alocarea memoriei pentru kernel

- Kernel-ul alocă şi eliberează tabele mici şi buffer-ele în timpul execuţiei în următoarele cazuri:
  - **pathname translation**: poate aloca un buffer pentru a copia path-ul din spaţiul utilizator;
  - funcţia **allocb()** alocă buffer-ele de dimensiune variabilă;
  - multe implementări de UNIX alocă aşa numitele **structuri zombie** pentru a păstra informaţiile de ieşire ale unui proces terminat;
  - în SVR4 şi Solaris, kernelul alocă diverse obiecte (structuri **proc**, **vnodes** şi **file block descriptor**) dinamic atunci când are nevoie;
- este folosit un **algoritm Buddy modificat**: costul alocării unui bloc liber de memorie trebuie sa fie mai mic decât costul alocării folosind algoritmii **first-fit** sau **best-fit**

Studiu de caz Windows 2000

- La Windows 2000, pentru folosirea memoriei virtuale, se poate adresa un spaţiu de adrese pe 32 de biţi.
- fiecare proces poate accesa 2GB pentru el şi 2GB din spaţiul partajat cu celelalte procese.
- Paginarea la Windows 2000:
  - o pagină poate fi în 3 stări:
    - available: nu este folosită curent de proces;
    - reserved: rezervată, dar nu este luată în calcul pentru aflarea memoriei ocupate de proces;
    - committed: paginile pentru care managerul memoriei virtuale a rezervat spaţiu în tabela de pagini.

Studiu de caz
Windows 2000

- Dacă apare o eroare de tip pagină lipsă, este selectată o pagină din setul de pagini rezervate pentru alocare.
- Dacă memoria este suficientă, este permisă creşterea zonelor rezervate pe măsură ce paginile sunt încărcate în memorie.
- Dacă este puţină memorie disponibilă, sunt evacuate din setul paginilor rezervate paginile cel mai puţin folosite.