Paginarea si segmentarea a memoriei

-Conceptul de memorie virtuala

-Translatarea adreselor

-Alocarea paginata a memoriei

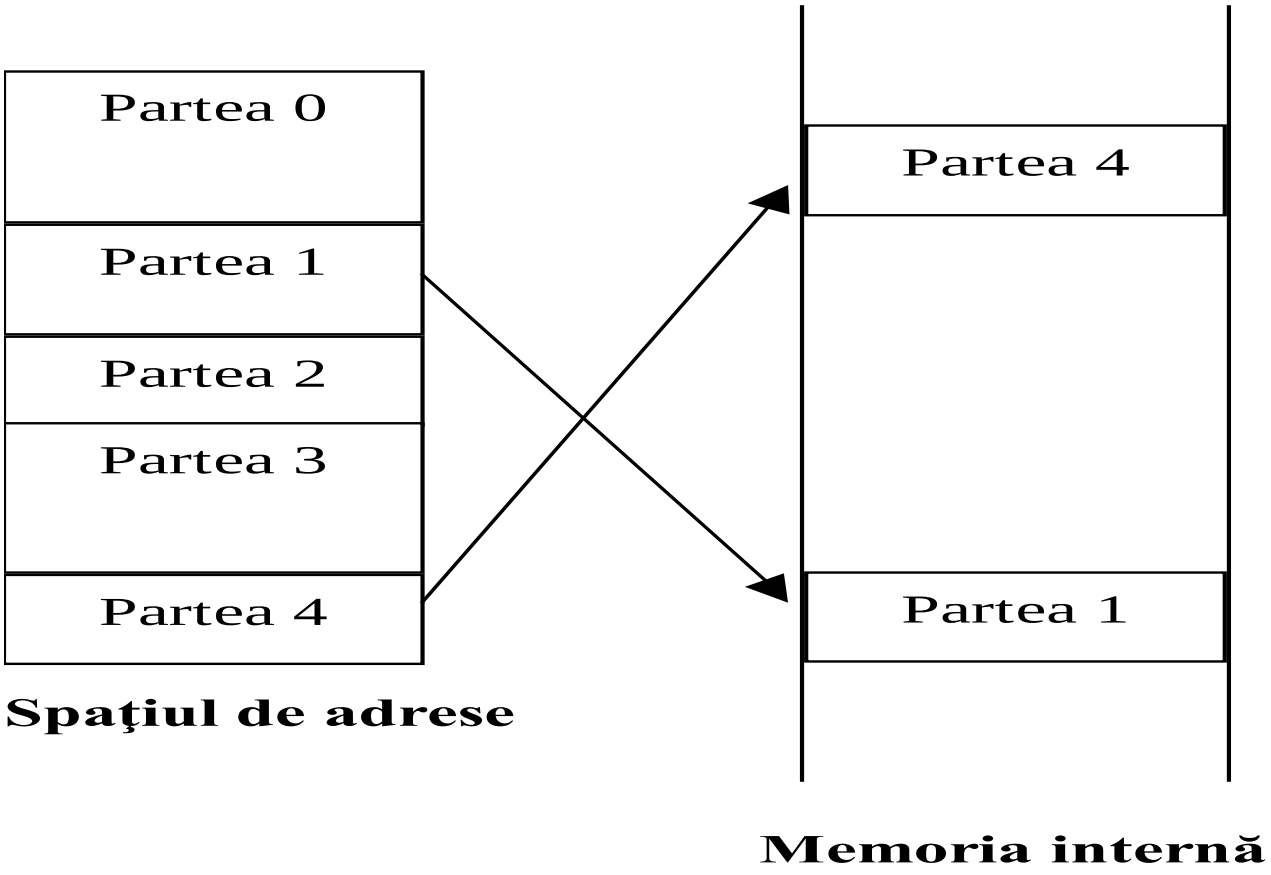
-Alocarea segmentata a memoriei

-Alocarea paginata si segmentata a memoriei

Memoria virtuală

- Dacă presupunem că n este dimensiunea magistralei de adrese, atunci numărul maxim de locaţii adresabile este 2^n . Dimensiunea memoriei interne este mult mai mică decât această valoarea. Acest fapt sugerează prelungirea spaţiului de adrese pe un disc rapid. Mulţimea acestor locaţii de memorie formează memoria virtuală.
- Spaţiul de adrese al procesului este divizat în părţi care pot fi încărcate în memoria internă atunci când execuţia procesului necesită acest lucru şi transferate înapoi în memoria secundară, când nu mai este nevoie de ele.
- Spaţiul de adrese al unui program se împarte în **partea de cod**, cea **de date** şi cea **de stivă**, identificate atât de compilator cât şi de mecanismul hardware utilizat pentru relocare.
- Partea (segmentul) de cod are evident un număr de componente mai mare, fiind determinată de fazele de execuţie (logica) ale programului.
- De **exemplu**, aproape toate programele conţin o fază necesară iniţializării structurilor de date utilizate în program, alta pentru citirea datelor de intrare, una (sau mai multe) pentru efectuarea unor calcule, altele pentru descoperirea erorilor şi una pentru ieşiri. Analog, există partiţii ale segmentului de date. Această caracteristică a programului se numeşte localizare a **referinţelor în spaţiu** şi este foarte importantă în strategiile utilizate de către sistemele de memorie virtuală.

- Când o anumită parte a programului este executată, este utilizată o anumită porţiune din spaţiul său de adrese, adică este realizată o localizare a referinţelor. Când se trece la o altă fază a calculului, corespunzătoare logicii programului, este referenţiată o altă parte a spaţiului de adrese şi, deci se schimbă această localizare a referinţelor. în figura urm., spaţiul de adrese este divizat în 5 părţi.



- Numai părţile 1 şi 4 din spaţiul de adrese corespund unor faze ale programului care se execută la momentul respectiv, deci numai acestea vor fi încărcate în memoria internă.
- Părţi diferite ale programului vor fi încărcate în memoria primară la momente diferite, în funcţie de localizarea în cadrul procesului.
- Sarcina administratorului memoriei este de a deduce localizarea programului şi de a urmări încărcarea în memorie a partiţiilor din spaţiul de adrese corespunzătoare, precum şi de a ţine evidenţa acestora în memoria internă, atâta timp cât sunt utilizate de către proces.
- Administratorul memoriei virtuale alocă porţiuni din memoria internă care au aceeaşi dimensiune cu cele ale partiţiilor din spaţiul de adrese şi, deci încarcă imaginea executabilă a părţii corespondente din spaţiul de adrese într-o zonă din memoria primară.
- Acest lucru are ca efect utilizarea de către proces a unei cantităţi de memorie mult mai reduse.

Translatarea adreselor

- Funcţia de translatare a adreselor virtuale, Ψ_t este o corespondenţă, variabilă în timp a **spaţiului de adrese virtuale** ale unui proces, în **spaţiul de adrese fizice**, adică mulţimea tuturor locaţiilor din memoria internă alocate acelui proces.
- Se cunosc două metode de virtualizare: **alocare paginată şi alocare segmentată.** Deci:

$$\Psi_t :< Spatiul\_de\_adrese\_relative > \rightarrow < Spatiul\_de\_adrese\_fizice > \cup \{\Omega\}$$

- în care `t` este un număr întreg, care reprezintă timpul virtual al procesului iar Ω este un simbol care corespunde adresei nule. Când un element `i`, al spaţiului de adrese virtuale este încărcat în memoria internă, $\Psi_t(i)$ este adresa fzică unde adresa virtuală `i` este încărcată.

- Dacă $\Psi_t(i) = \Omega$, la momentul virtual `t` şi procesul face referinţă la locaţia `i,` atunci sistemul întreprinde următoarele acţiuni:

  - Administratorul memoriei cere oprirea execuţiei procesului.

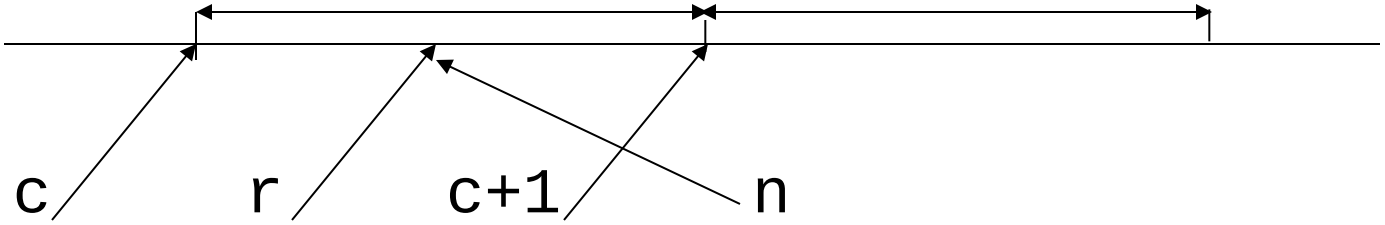
  - Informaţia referenţiată este regăsită în memoria secundară şi încărcată într-o locaţie de memorie k.

  - Administratorul memoriei schimbă valoarea funcţiei .

  - Administratorul memoriei cere reluarea execuţiei programului.
- Observăm că locaţia referenţiată din spaţiul de adrese virtuale nu este încărcată în memoria primară, după ce s-a declanşat execuţia instrucţiunii cod maşină respective. Acest lucru va declanşa **reexecutarea** instrucţiunii după ce locaţia respectivă a fost încărcată din memoria virtuală în memoria primară.
- De asemenea, dimensiunea spaţiului de adrese virtuale ale unui proces, este **mai mare** decât dimensiunea spaţiului de adrese fizice alocat procesului, spre deosebire de metodele când întregul fişier executabil era încărcat în memorie.

Alocarea paginată

- **Alocarea paginată** a apărut la diverse SC pentru a evita fragmentarea memoriei interne, care apare la metodele anterioare de alocare.
- Memoria virtuală este împărţită în zone de lungime fixă numite **pagini virtuale.** Paginile virtuale se păstrează în memoria secundară. Memoria operativă este împărţită în zone de lungime fixă, numite **pagini fizice.**
- Lungimea unei pagini fizice este fixată prin hard. Paginile virtuale şi cele reale(fizice) au aceeaşi lungime, lungime care este o putere a lui 2 şi care este o constantă a sistemului(de exemplu 1Ko, 2Ko etc).
- **Adresa paginata**. Presup. ca $l = 2^k$ este dimensiunea unei pagini; daca n este adresa unei locatii din memoria fizica, atunci conform teoremei de impartire cu rest, $n = c2^k + r$; c este numarul de pagina in care se afla locatia de adresa n iar r este adresa (deplasamentul) din cadrul paginii. Perechea (c, r) se numeste adresa paginata fizica a locatiei de memorie. Analog se defineste notiunea de adresa paginata a unei locatii din memoria virtuala.

- **Translatarea adreselor virtuale**.
- 

- In momentul cind informatia dintr-o locatie virtuala trebuie incarcata in memoria interna, se pune problem transl. adresei virtuale intr-una fizica. Deoarece fiecare pagină are aceeaşi dimensiune `l`, adresa virtuală `i` poate fi convertită într-un număr de pagină şi un deplasament în cadrul paginii, numărul de pagină fiind `i div l`, iar deplasamentul `i mod l.`
- Deci fiecare adresă virtuală, repectiv adresă fizică va fi de forma `(p,d),` respectiv de forma `(f,d),` unde `p` este numărul paginii virtuale, `f` este numărul paginii fizice, iar `d` este adresa (deplasamentul) în cadrul paginii.
-

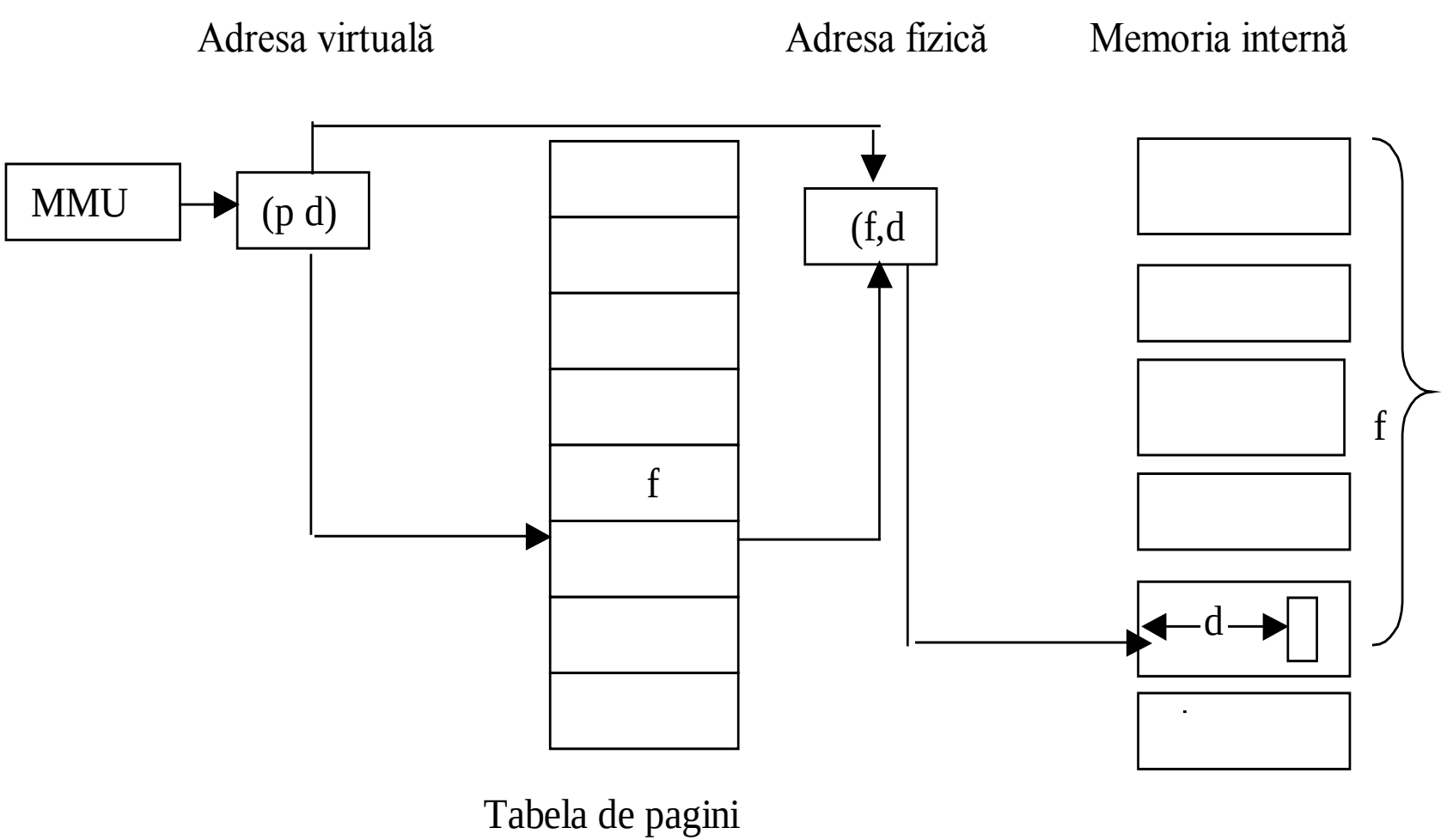
- Orice pagină virtuală din spaţiul de adrese al procesului, poate fi încărcată în oricare din paginile fizice din memoria internă alocate acestuia. Acest lucru este realizat printr-o componentă hardware numită MMU(**M**emory **M**anagement **U**nit), care implem. funcţia Ψ (figura urm).
- Dacă este referenţiată o locaţie dintr-o pagină care nu este încărcată în memoria internă, MMU opreşte activitatea CPU, pentru ca SO să poată executa următorii paşi:

  - Procesul care cere o pagină neîncărcată în memoria internă este suspendat.

  - Administratorul memoriei localizează pagina respectivă în memoria secundară.

  - Pagina este încărcată în memoria internă, eventual în locul altei pagini, dacă în memoria internă nu mai există pagini fizice libere alocate procesului respectiv şi în acest caz, tabela de pagini este modificată.

  - Execuţia procesului se reia din locul în care a fost suspendat.


Adresa virtuală
Adresa fizică
Memoria internă
MMU
(p d)
(f,d
f
f
d
Tabela de pagini

- Fiecare proces are propria lui tabelă de pagini, în care este trecută adresa fizică a paginii virtuale, dacă ea este prezentă în memoria operativă.
- La încărcarea unei noi pagini virtuale, aceasta se depune într-o pagină fizică liberă. Deci, în memoria operativă, paginile fizice sunt distribuite în general necontinuu, între mai multe procese.
- Spunem că are loc o **proiectie a spatiului virtual peste cel real.**
- Acest mecanism are avantajul că foloseşte mai eficient memoria operativă, fiecare program ocupând numai memoria strict necesară la un moment dat.
- Un alt avantaj este posibilitatea folosirii în comun, de către mai multe programe, a instrucţiunilor unor proceduri. O procedură care permite acest lucru se numeste **procedură reentrantă.**

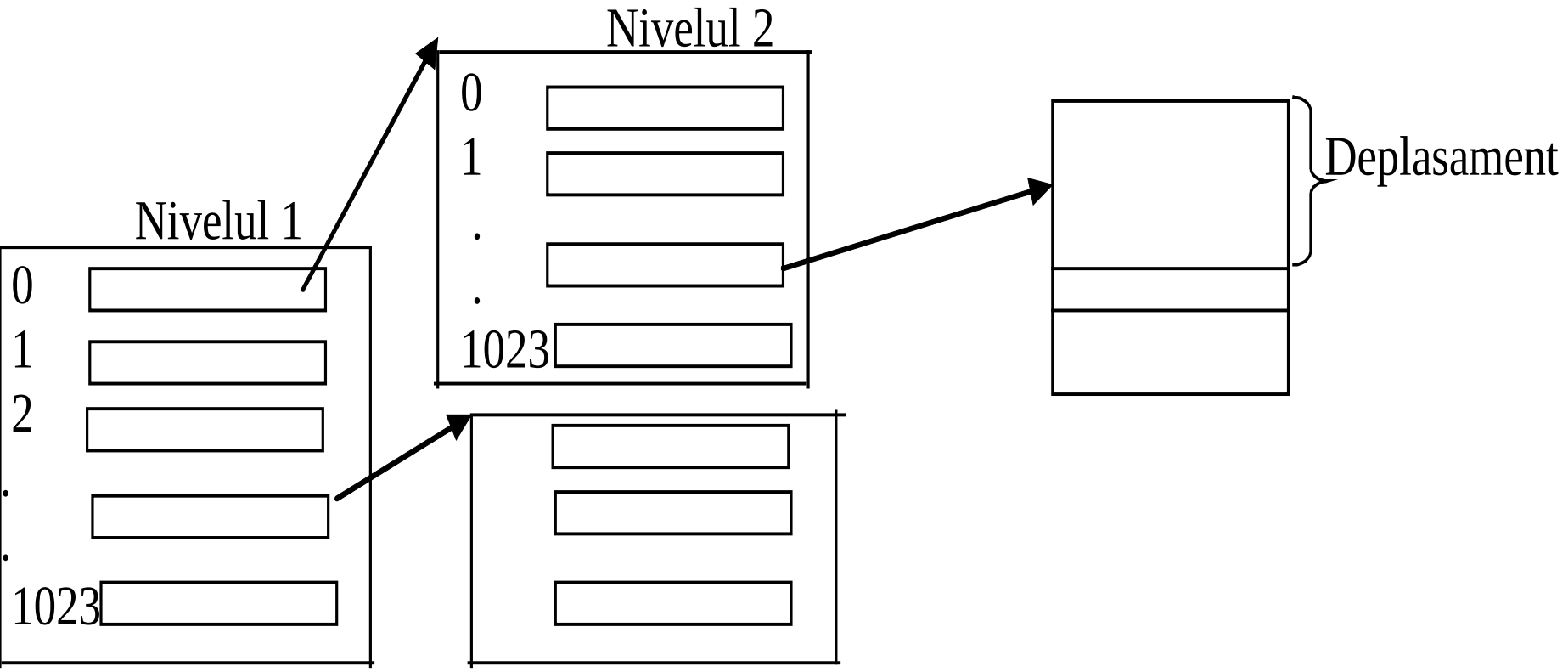
- **Evidenţa paginilor virtuale** încărcate în pagini fizice se poate realiza prin tabela de pagini.
- Tabela de pagini poate fi privită ca o funcţie, care are ca argument numărul de pagină virtuală şi care determină numărul de pagină fizică in care se va incarca.
- Pe baza deplasamentului se determină locaţia din memoria fizică unde se va încărca locaţia virtuală referenţiată.
- Această metodă ridică două mari probleme:

  1. Tabela de pagini poate avea un număr mare de intrări. Calculatoarele moderne folosesc adrese virtuale pe cel puţin 32 de biţi. Dacă, de exemplu o pagină are dimensiunea de 4K, atunci numărul intrărilor în tabelă este mai mare decât un million. În cazul adreselor pe 64 de biţi numărul intrărilor în tabelă va fi, evident mult mai mare.

  2. Corespondenţa dintre locaţia de memorie virtuală şi cea din memoria fizică trebuie să se realizeze cât mai rapid posibil; la un moment dat, o instrucţiune cod maşină poate referenţia una sau chiar două locaţii din memoria virtuală.

  O metodă mult mai eficientă, care înlocuieşte căutarea secvenţială cu cea arborescentă este **organizarea tabelei pe mai multe niveluri.** Astfel, o adresă virtuală pe 32 de biţi de exemplu, este un triplet (Pt1, Pt2, d), primele două câmpuri având o lungime de 10 biţi, iar ultimul de 12 biţi.

- În figura urm. este ilustrată o astfel de tabelă de pagini.



- Observăm că numărul de intrări în tabela de nivel 1 este de $2^{10}=1024$;
- Fiecare intrare în această tabelă conţine un pointer către o tabelă a nivelului 2.
- Dimensiunea unei pagini este de $2^{12}=4x2^{10}=4K$.
- Când o adresă **virtuală** este prezentată MMU, se extrage valoarea câmpului PT1, care este folosit ca index în tabela de la nivelul 1.
- Pe baza acestuia, se găseşte adresa de început a uneia dintre tabelele de la nivelul 2.
- Câmpul PT2 va fi un index în tabela de la nivelul 2 selectată, de unde se va lua numărul de pagină fizică corespunzător numărului de adresă virtuală conţinut în adresa referenţiată.
- Pe baza acestui număr şi a deplasamentului (d), se determină locaţia de memorie fizică în care va fi încărcată respectiva locaţie din memoria virtuală.
- În condiţiile specificate, o tabelă de la nivelul 2 va gestiona o zonă de memorie de capacitate $1024 \times 4K=4M$.

- **Observaţie.** Tabela de pagini se poate organiza şi pe 3 sau 4 niveluri, în funcţie de dimensiunea memoriei virtuale şi a celei fizice. Dacă tabela are o dimensiune mare, ea poate fi păstrată(cel puţin parţial) şi pe un disc rapid.
- **Exemplu.** Fie adresa virtuală 0x00403004 (4202996 în zecimal). -- Dacă se face transformarea în binar se obţine:

  00000000010000000011000000000100

  ← Pt1 → ← Pt2 → ← depl →

  - In reprez. zec., câmpul PT1 are valoarea 1, câmpul PT2 are valoarea 3, iar pentru deplasament *se* obt. valoarea 4.

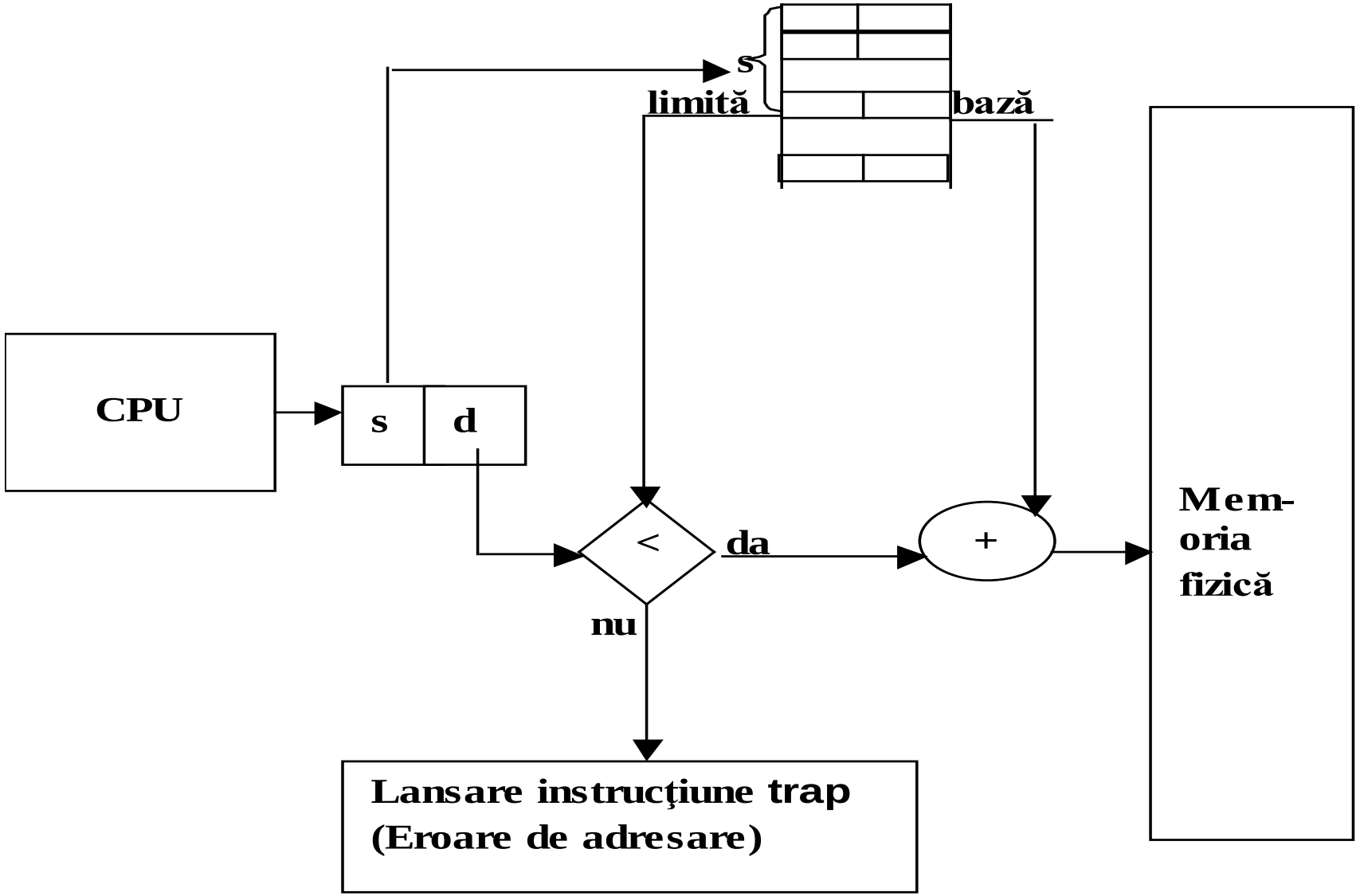
  - Pe baza valorii lui PT1, MMU va găsi că componenta PT2 a adresei se află în a doua tabelă de la nivelul 2, care corespunde la adrese între 4M şi 8M .

  - Folosind PT2 ca index în a doua tabelă de nivel 2, având valoarea 3, se determină că adresa locaţiei este cuprinsă între adresele relative la pagina respectivă de memorie 12292 şi 16383, corespunzătoare adreselor absolute 4 202 592 şi 4 210 687.
  - Se adugă valoarea 4 a deplasamentului şi se obţine adresa locaţiei, adică 4 202 596.
- Pe baza valorii bitului **present/absent** se determină dacă pagina virtuală este (sau nu) adusă în memoria fizică.

Alocare segmentată.

- Din punctual de vedere al utilizatorului**,** o aplicaţie este formată dintr-un program principal şi o mulţime de subprograme(funcţii sau proceduri). Aceste folosesc diferite structuri de date (tablouri, stive etc), precum şi o mulţime de simboluri (variabile locale sau globale). Toate aceste sunt identificate printr-un nume.
- Pornind de la această divizare logică a unui program, s-a ajuns la o metoda de alocării **segmentare** a memoriei. Spre deosebire de metodele de alocare a memoriei bazate pe partiţionare, unde fiecărui proces trebuie să i se asigure un spaţiu contiguu de memorie, mecanismul de alocare segmentată, permite ca un proces să fie plasat în zone de program distincte, fiecare dintre ele conţinând o entitate de program, numit **segment**. Segmentele pot fi definite explicit prin directive ale limbajului de programare sau implicit prin semantica programului.
- De exemplu, un compilator de C poate crea segmente **de cod** pentru fiecare procedură sau funcţie, segmente pentru **variabilele globale,** segmente pentru **variabilele locale,** precum şi **segmente de stivă**. Deosebire esenţială dintre alocarea paginată şi cea segmentată este aceea că segmentele sunt de **lungimi diferite.**

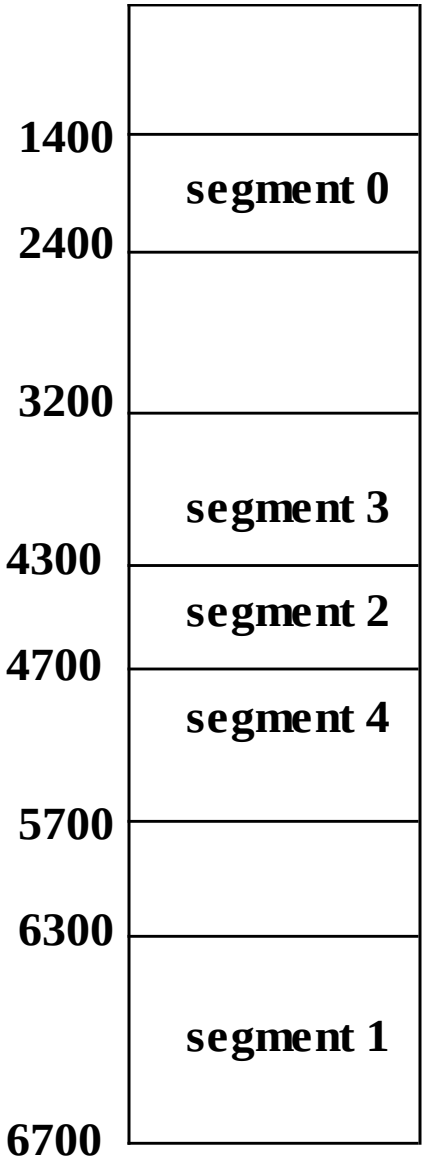
- În mod analog cu alocarea paginată, **o adresă virtuală** este o pereche (s,d), unde s este numărul segmentului iar d este adresa din cadrul segmentului.
- Adresa reală (fizică) este o adresă obisnuită.
- Transformarea unei adrese virtuale într-o adresă fizică, se face pe baza unei **tabele de segmente**. Fiecare intrare în acestă tabelă este compusă dintr-o **adresă de bază**(adresa fizică unde este localizat segmentul în memorie) şi **lungimea segmentului(limită)**.
- În figura urm. este ilustrat modul de utilizare a tabelei de segmente.
- Componenta s a adresei virtuale, este un indice în tabela de segmente.
- Valoarea deplasamentului d trebuie să fie cuprinsă între 0 şi lungimea segmentului respectiv(în caz contrar este lansată o instrucţiune trap, care generează o întrerupere).
- Dacă valoarea d este validă, ea este adăugată adresei de început a segmentului şi astfel se obţine adresa fizică a locaţiei respective.


s
limită
bază
CPU
s
d
<
da
+
nu
Mem-
oria
fizică
Lansare instrucţiune trap
(Eroare de adresare)

- **Exemplu.** Să presupunem că avem un proces care, din punct de vedere logic este format dintr-un program principal, un subprogram şi în care se apelează o procedură de sistem.
- Corespunzător celor trei entităţi vom avea trei segmente, la care se adaugă un segment pentru stivă şi unul pentru tabele de simboluri.
- Figura urm. redă alocarea de spaţiu în memoria internă pentru acest proces.

Număr segment	Limită	Bază
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

Tabela de segmente



Memoria fizică

- Segmentul 2 are o lugime de 400 octeţi şi începe la locaţia 4300. Astfel, unei referinţe la octetul 53 al segmentului 2, îi corespunde locaţia din memoria internă 4500+53=4353.
- Avantajele alocării segmentate, faţă de cea bazată pe partiţii sunt:

  - se pot creea **segmente reentrante**, care pot fi partajate de mai multe procese. Pentru aceasta este suficient ca toate procesele să aibă în tabelele lor aceeaşi adresă pentru segmentul respectiv.

  - se poate realiza o **bună protecţie a memoriei**. Fiecare segment în parte poate primi alte drepturi de acces, drepturi trecute în tabela de segmente.

  - De **exemplu**, un segment care conţine instrucţiuni cod maşină, poate fi declarat “read-only” sau executabil.

  - La orice calcul de adresă, se verifică respectarea modului de acces al locaţiilor din segmental respectiv.

Alocarea segmentată cu paginare

- Cele două metode de alocare a memoriei au avantajele şi dezavantajele lor.
- În cazul segmentării poate să apară fenomenul de fragmentare.
- În cazul paginării, se efectuează o serie de operaţii suplimentare de adresare.
- De asemenea, unele procesoare folosesc alocarea paginată (Motorola 6800), iar altele folosesc segmentarea (Intel 80x86, Pentium).
- Ideea segmentării cu paginare, este aceea că alocarea spaţiului pentru fiecare segment să se facă paginat. Această metodă este utilizată de sistemele de operare actuale.
- Spaţiul de adrese virtuale al fiecărui proces este împărţit în două partiţii:
- - prima partiţie este utilizată numai de către procesul respectiv;
- - cealaltă partiţie, este partajată împreună cu alte procese.
- - informaţiile despre prima partiţie, respective a doua partiţie sunt păstrate în **descriptorul tabelei locale (LDT – L**ocal **D**escriptor **T**able**),** respectiv **descriptorul tabelei globale (GDT – G**lobal **D**escriptor **T**able**).**
- - Fiecare intrare în LDT/GDT este reprezentată pe 8 octeţi şi conţine informaţii despre un anumit segment, printre care adresa de început şi lungimea acelui segment.

- Adresa virtuală este un cvarduplu *(g,s,p,d)*, în care:

  - *g* indică dacă segmentul este local sau global (se selectează LDT sau GDT);

  - *s* este numărul de segment;

  - *p* specifică pagina;

  - *d* valoarea deplasamentului
- La nivelul fiecărui segment, există o tabelă de translatare a adreselor virtuale în adrese fizice, organizată arborescent, pe trei niveluri, descrisă în anterior. Translatarea unei adrese virtuale în adresă fizică se realizează astfel:

  - pe baza valorii *g* se alege tabela de segmente;

  - pe baza valorii *s* se alege intrarea corespunzătoare segmentului, de unde se iau adresa de începuta şi lungimea segmentului;

  - prin utilizarea tabelei de pagini, pe baza valorilor *p,d* se obţine adresa fizică a locaţiei virtuale respective.