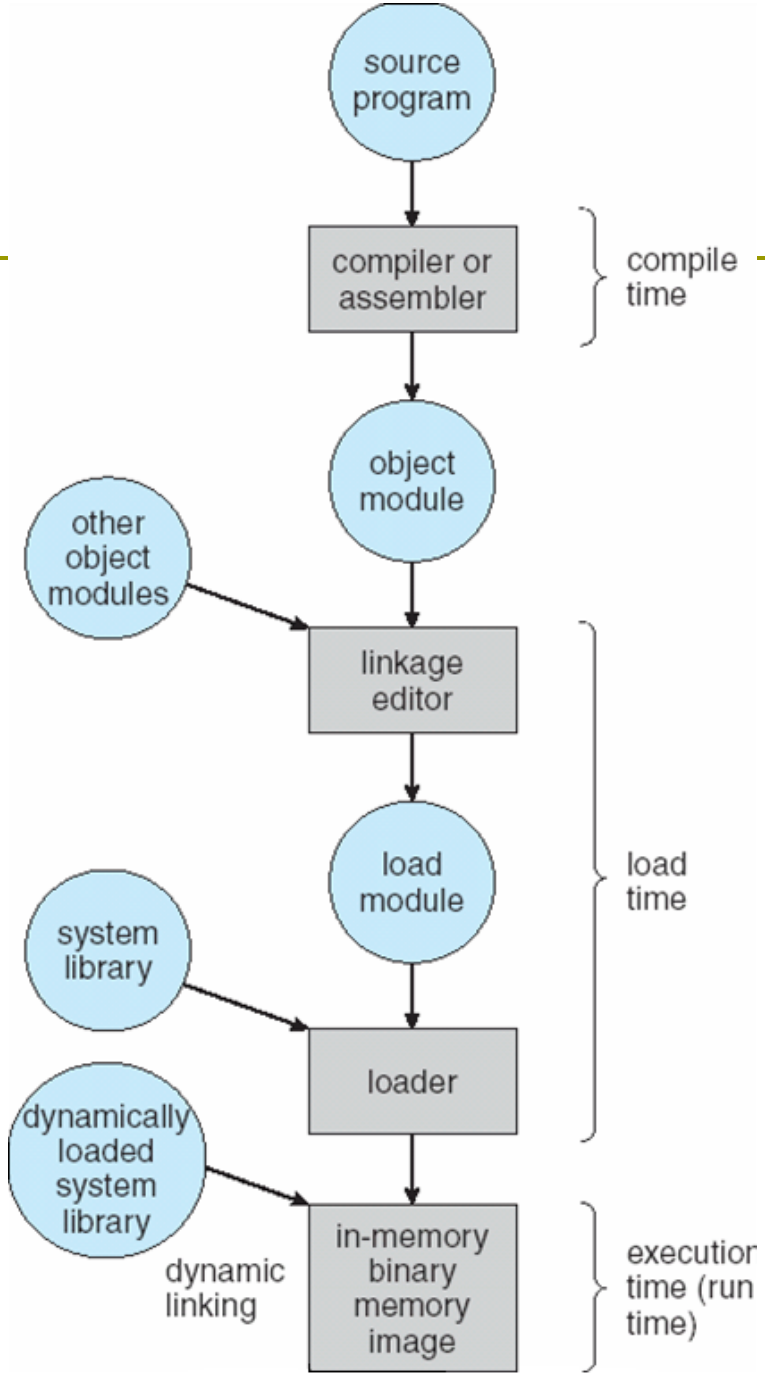
Sisteme de Operare

- Gestiunea memoriei
  - Scheme de alocare a memoriei
  - Memoria virtuală

Gestiunea memoriei

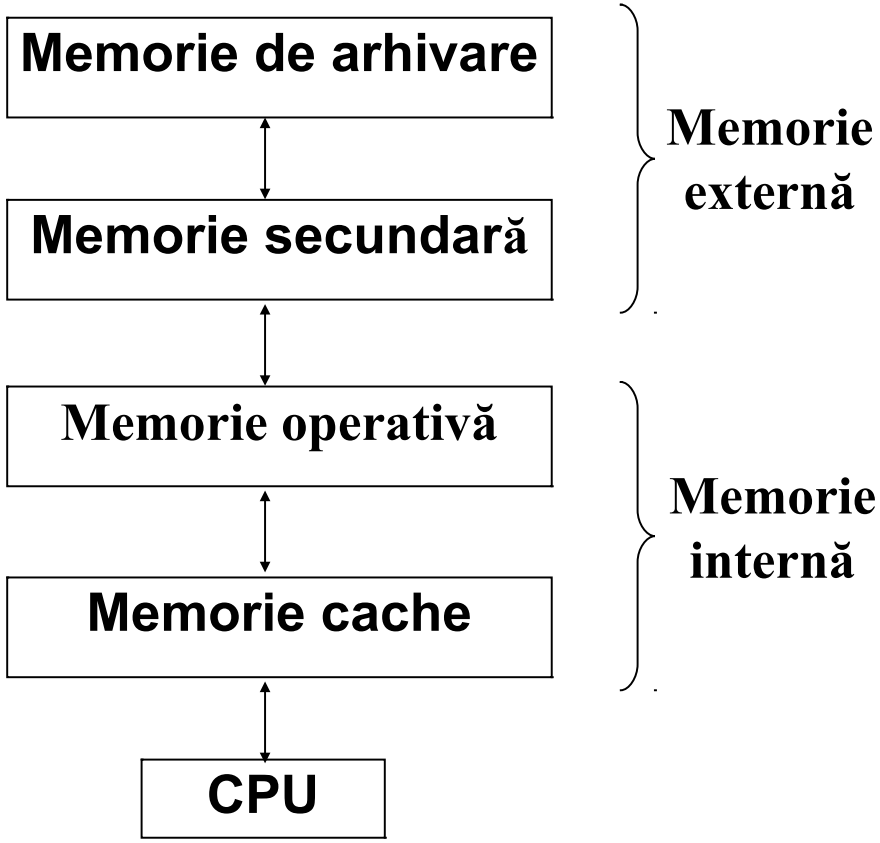
- Pentru a putea fi executat, un proces are nevoie de o anumită cantitate de memorie.
- Dacă sistemul suportă multiprogramare, este necesar ca în memorie să fie prezente mai multe programe, fiecare folosind zonele de memorie alocate independent de eventualele programe active. Pe durata execuţiei unui proces, necesarul de memorie poate varia.
- Spaţiul de memorie principală a unui sistem de calcul (aceea care poate fi accesată direct de către CPU) este fix şi trebuie gestionat cât mai eficient.


source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
executior time (run time)

Gestiunea memoriei

- Obiective:
  - calculul de translatare a adresei (relocare)
  - protecţia memoriei
  - organizarea şi alocarea memoriei operative
  - gestiunea memoriei secundare
  - politici de schimb între proces, memoria operativă şi memoria secundară

Gestiunea memoriei
Structura ierarhică a memoriei



- Memoria cache
  - conţine informaţiile cele mai recent utilizate de CPU, are capacitate mică dar timp de acces foarte rapid.
  - La fiecare acces CPU verifică dacă data se află în memoria cache şi apoi solicită memoria operativă. Dacă este, are loc transferul între CPU şi memoria operativă, iar dacă nu se caută data în nivelurile superioare.

Gestiunea memoriei

- Principiul vecinătăţii:
  - dacă la un moment dat se solicită o informaţie dintr-un anumit loc, atunci solicitarea din momentul următor se va face cu o mare probabilitate la o informaţie din apropierea precedentei.
  - Informaţia cerută de CPU este adusă din nivelul în care se află, dar împreună cu ea sunt aduse şi un număr de locaţii vecine astfel încât să umple memoria cache.

Gestiunea memoriei
Structura ierarhică a memoriei

- Memoria operativă
  - conţine programele (codul) şi datele pentru toate procesele existente în sistem.
  - În momentul în care un proces este terminat şi distrus, spaţiul din memoria operativă pe care l-a ocupat este eliberat şi va fi alocat altui proces.
  - Viteza de acces este foarte mare (memoria SDRAM, DDRAM).

Gestiunea memoriei
Structura ierarhică a memoriei

- Memoria secundară
  - apare la SO care deţin mecanisme de memorie virtuală.
  - Această memorie este privită ca o extensie a memoriei operative.
  - Suportul ei principal este discul magnetic şi din acest motiv este mult mai lentă decât memoria operativă.

Memoria expandată

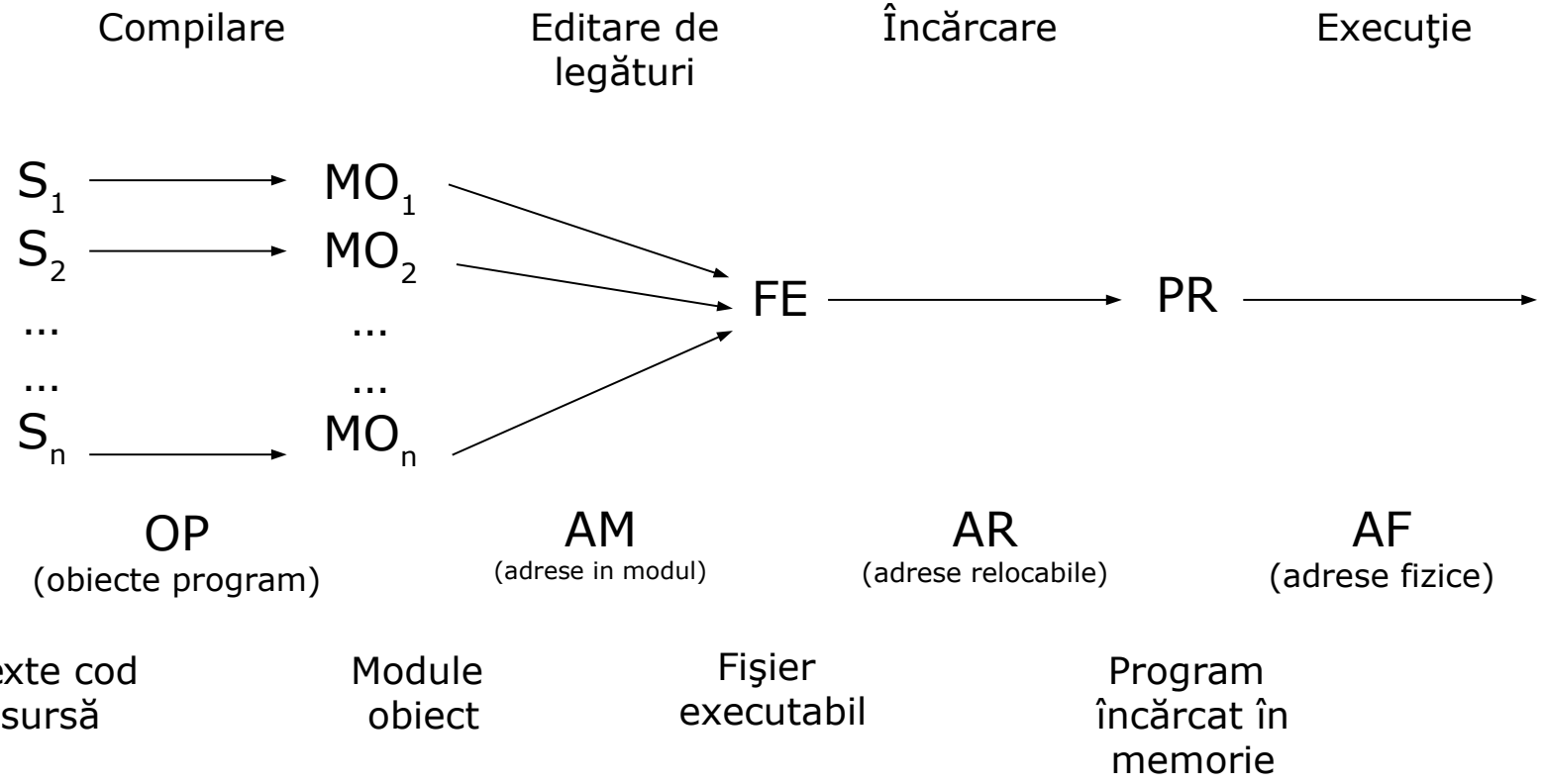
- Este mecanism ce permite ca mai multe chip-uri de memorie operativă să aibă, alternativ, aceeaşi adresă de memorie.
- Apare la primele sisteme IBM PC din seria 8086 şi 8080, la care memoria disponibilă era de 640Ko.
- Este o memorie secundară care are ca suport memoria internă.

Gestiunea memoriei
Structura ierarhică a memoriei

- Memoria de arhivare
  - Este gestionată de utilizator şi constă din fişiere, baze de date etc, rezidente pe diferite suporturi de stocare a informaţiei.
- Memoria cache şi memoria operativă formează memoria internă.
  - Accesul CPU se face în mod direct.
  - Pentru a avea acces la datele din memoria secundară şi de arhivare, acestea trebuie mutate în memoria internă.

Gestiunea memoriei
Mecanisme de translatare a adresei

- Adresarea memoriei constă în realizarea legăturii între un obiect al programului şi adresa corespunzătoare din memoria fizică



Fazele translatării unui program

Gestiunea memoriei
Mecanisme de translatare a adresei

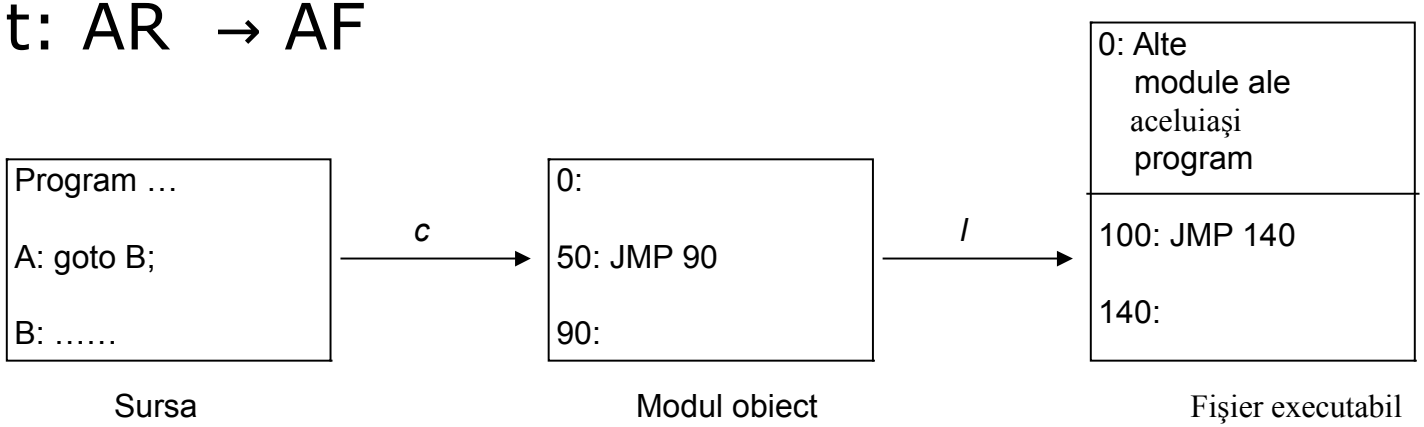
- Calculul de adresă este modalitatea prin care se ajunge de la OP la AF şi necesită trei faze corespunzătoare fazelor programului:
  - **compilare**: textul sursă este transformat în module obiect. Numele de obiectele din program sunt transformate în numere reprezentând **adrese în cadrul modulului (AM)**.
    - Prima funcţie din cadrul calculului de adresă este:
      - c: OP →AM

      şi este executată de compilator. Modul ei de evaluare depinde de limbaj, compilator şi SO.
  - **editare de legături**: sunt grupate mai multe module pentru a forma fişierul executabil. Adresele din cadrul modulului sunt transformate în **adrese relocabile (AR)** de către editorul de legături.
    - Funcţia este:
      - l: AM → AR

Gestiunea memoriei

Funcția de translatare (relocare) a adresei

- transformă adresele relocabile în **adrese fizice (AF)**, este executată de CPU şi depinde de sistemul de calcul şi de existenţa dispozitivelor de management a memoriei:
  - t: AR → AF



Transformarea în fişier executabil

Gestiunea memoriei

- Moduri de adresare uzuale
  - adresarea absolută:
    - are loc atunci când **t(x)=x**, ∀ **x∈ AR** sau altfel spus când **AR=AF**.
  - adresarea bazată:
    - presupune utilizarea unui registru general ca registru de bază:
      - **t(x) = (R_b) + x**, ∀ **x∈ AR**
      - toate adresele relocabile sunt mărite cu ajutorul registrului de bază.
  - adresarea indexată:
    - presupune specificarea, în cadrul instrucţiunii maşină a unui **registru de index R_i**. După ce se obţine o adresă provizorie AF1 (în funcţie de arhitectura procesorului), adresa definitivă se obţine astfel:

$$AF2 = AF1 + (R_i)$$

    - Acest mod de adresare se foloseşte pentru localizarea elementelor în cadrul unui tablou.

Gestiunea memoriei

- Moduri de adresare uzuale
  - adresarea relativă:
    - se foloseşte pentru realizarea de salturi într-un program, precizându-se sensul şi numărul de locaţii peste care trebuie sărit pentru a ajunge la noua adresă:

$$AF2 = AF1 + l$$

  - adresarea indirectă:
    - După ce procesorul obţine o adresă AF1, el interpretează conţinutul de la AF1 nu ca o valoare a unui operand, ci ca o nouă adresă:

$$AF2 = (AF1).$$

    - Acest mod de adresare se aplică în special la invocarea parametrilor actuali din cadrul unui subprogram. Programul apelant transmite subprogramului lista cu adresele parametrilor actuali, după care subprogramul are acces la ei folosind adresarea indirectă.

Gestiunea memoriei

- Protecţia memoriei
  - Fiecare sistem de calcul şi sistem de operare trebuie să aibă mecanisme care să asigure utilizarea corectă a spaţiului de memorie de către toate procesele.
  - Fiecare entitate de memorie alocată conţine o **cheie de protecţie**, iar fiecare entitate de program încărcabilă în memorie la un moment dat are o **cheie de acces**.

Gestiunea memoriei
Protecția memoriei

- Cheia de protecţie - un şir de biţi prin care se specifică posibilităţile zonei respective:
  - R: din zonă se poate citi.
    - Se permite execuţia instrucţiunilor maşină care aduc datele din zonă în regiştri. Dacă este numai read-only, memorarea datelor din regiştri în zona respectivă este interzisă.
    - În aceste zone se trec de obicei elementele constante ale proceselor.
  - W: în zonă se poate scrie.
    - Este permisă memorarea datelor din regiştri în zona respectivă (dacă este interzisă avem situaţia de la read-only).
    - Există două cazuri speciale de scriere, care la unele SO sunt specificate separat: extindere (scrierea la sfârşitul zonei) şi ştergere (pregătirea zonei astfel încât extinderea la momentul următor se va face de la începutul zonei) şi sunt folosite mai mult la fişiere decât la memorie.
  - E: conţinutul zonei poate fi executat
    - există instrucţiuni maşină ce pot fi executate.
    - În aceste zone conţinutul rămâne neschimbat, fiind interzis proceselor să modifice zona respectivă.
    - La unele SO este asimilată cu posibilitatea de read-only.

Gestiunea memoriei
Protecția memoriei

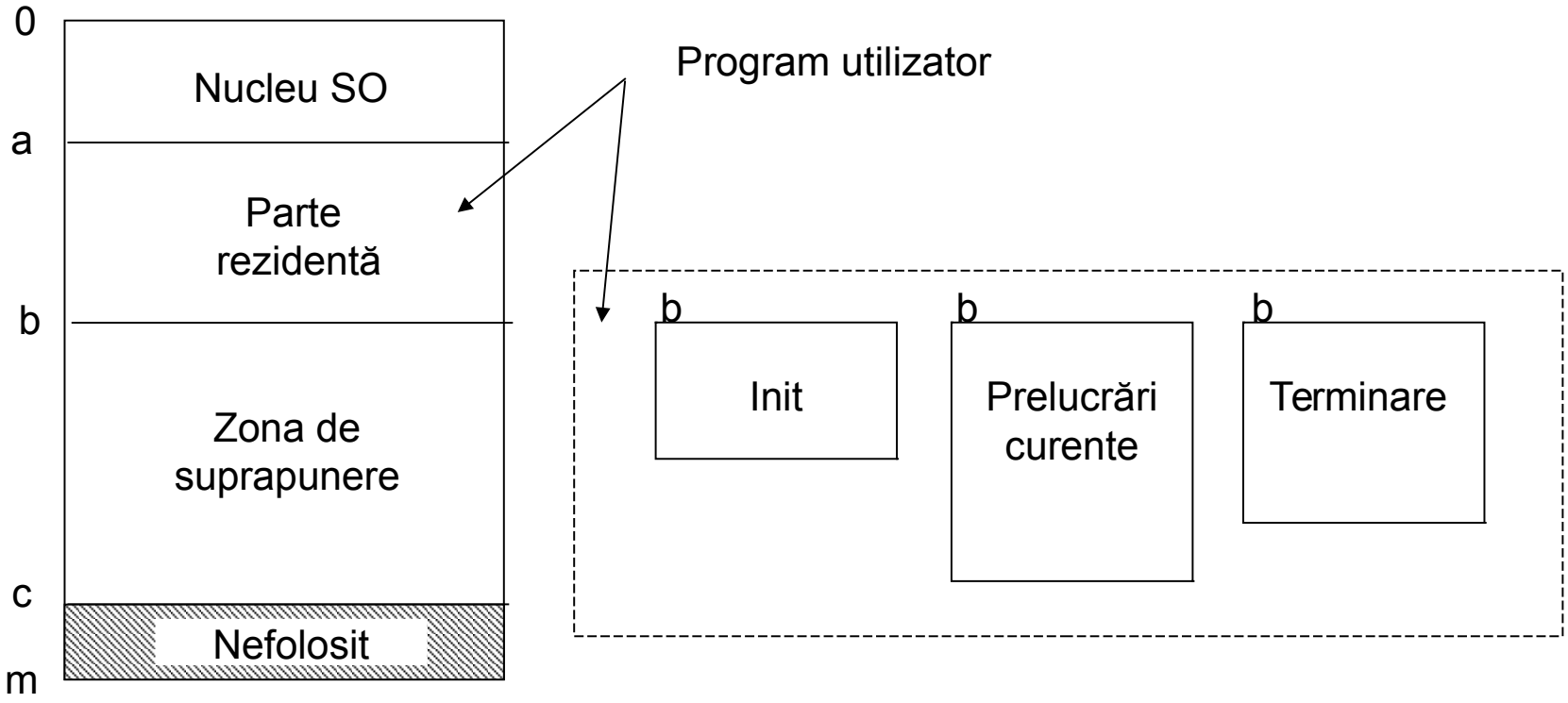
- Cheia de acces
  - un şir de biţi reprezentând drepturi de acces.
  - Este primită de un proces la încărcare
- Protecţia memoriei se asigură executând doi paşi:
  - la fiecare solicitare a unei locaţii de memorie se compară cheia de protecţie cu cheia de acces.
  - În caz de neconcordanţă accesul este interzis şi procesul se termină cu mesaj de eroare.
  - Dacă cheile coincid, se compară posibilităţile zonei solicitate cu drepturile de acces ale procesului şi cu acţiunea cerută de proces.
  - Accesul este permis numai dacă răspunsul la aceste comparaţii este afirmativ.

Scheme de alocare a memoriei

- alocare reală:
  - la sistemele monoutilizator
  - la sistemele multiutilizator:
    - cu partiţii fixe (statică):
      - absolută;
      - relocabilă;
    - cu partiţii variabile (dinamică)
- alocare virtuală:
  - paginată;
  - segmentată;
  - segmentată şi paginată

Alocarea la sistemele monoutilizator

- La sistemele monoutilizator este disponibil aproape întreg spaţiul de memorie.
- Gestiunea spaţiului cade în sarcina utilizatorului, existând tehnici de suprapunere (overlay) pentru a-şi putea rula programele mari:



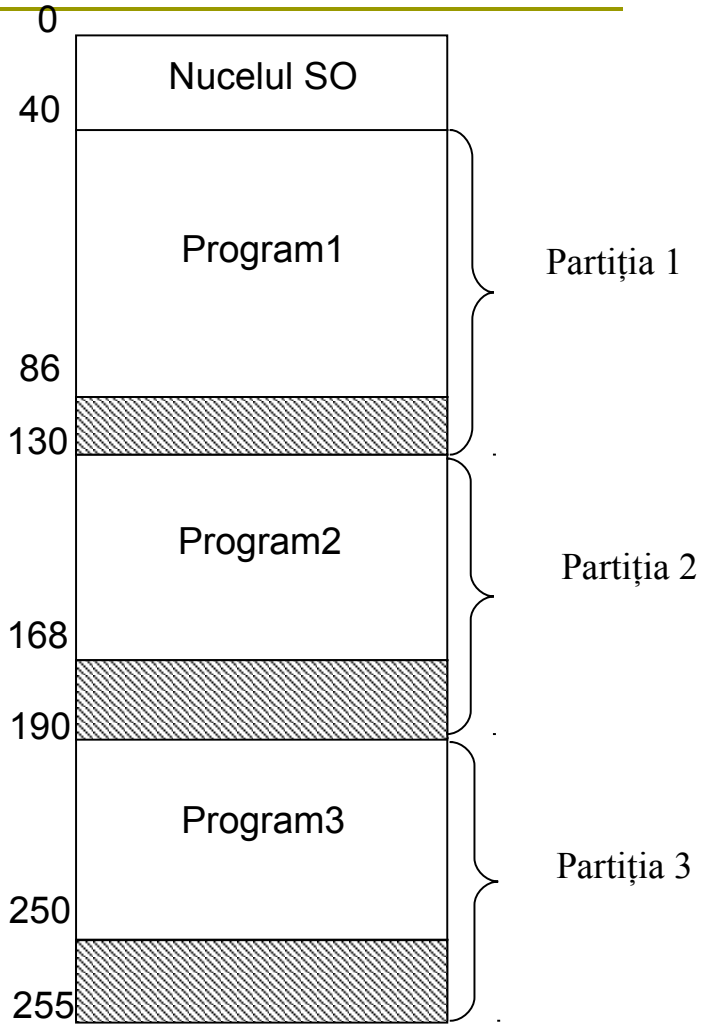
Alocarea memoriei la un sistem monoutilizator folosind suprapunerea

tehnici de suprapunere

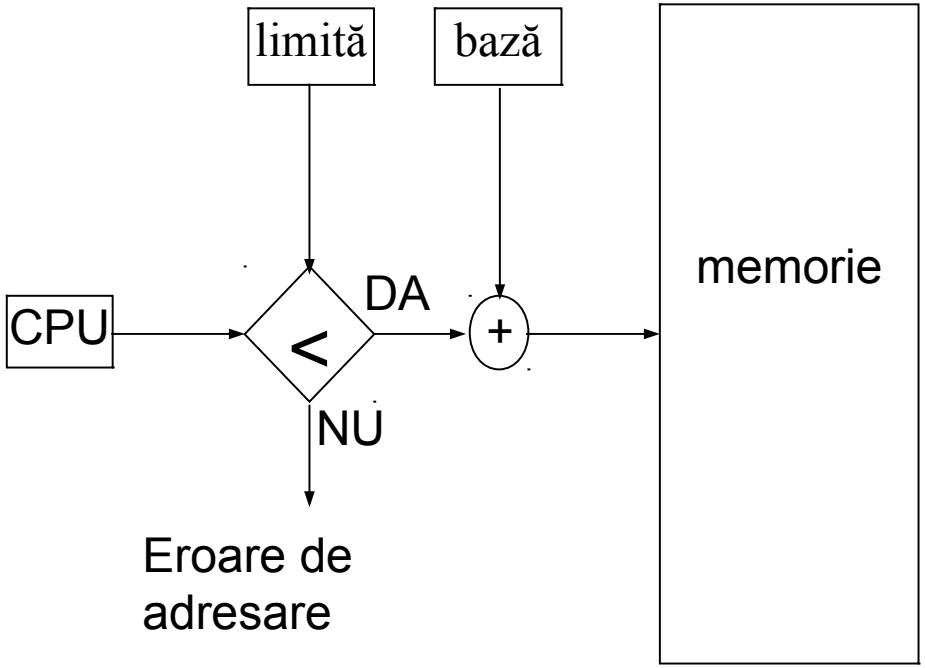
- presupune păstrarea în memorie a instrucţiunilor şi datelor necesare la un moment dat.
- Când este nevoie de alte instrucţiuni acestea sunt încărcate în memorie în locul celor de care nu mai este nevoie:
- zona dintre adresele **0** şi **a-1** este rezervată nucleului sistemului de operare ce rămâne acolo de la încărcare şi până la oprirea sistemului.
- Între **c** şi **m-1** este spaţiul nefolosit de către programul utilizator activ (dacă memoria are m locaţii).
- Adresa **c** variază de la un program utilizator la altul.

Alocarea cu partiții fixe

- Această alocare se mai numeşte şi alocare statică sau alocare **MFT – Memory Fix Task**.
- presupune decuparea memoriei în zone de lungime fixă numite partiţii.
- O partiţie este alocată unui proces pe toată durata execuţiei lui, indiferent dacă o ocupă complet sau nu.
- partiţiile au lungimi diferite



Alocarea cu partiții fixe



Adresarea cu registru bază

- **Alocarea absolută**: se face pentru programe pregătite de editorul de legături pentru a fi rulate într-o zonă de memorie şi numai acolo.
- **Alocarea relocabilă**: adresarea în partiţie se face cu bază şi deplasament: la încărcarea în memorie a programului, în registrul lui de bază se pune adresa de început a partiţiei.

Alocarea cu partiții fixe

- Fixarea dimensiunii partiţiilor
  - Nu se pot prevedea cantităţile de memorie pe care le vor solicita procesele încărcate în aceste partiţii
  - Alegerea unor partiţii de dimensiuni mari scade probabilitatea ca unele procese să nu poată fi executate, dar şi numărul de procese active din sistem.
  - La fiecare partiţie există un şir de procese care aşteaptă să fie executate
    - Modul în care este organizat sistemul de aşteptare poate influenţa performanţele de ansamblu ale sistemului de calcul şi poate atenua efectul unei dimensionări defectuoase a partiţiilor.

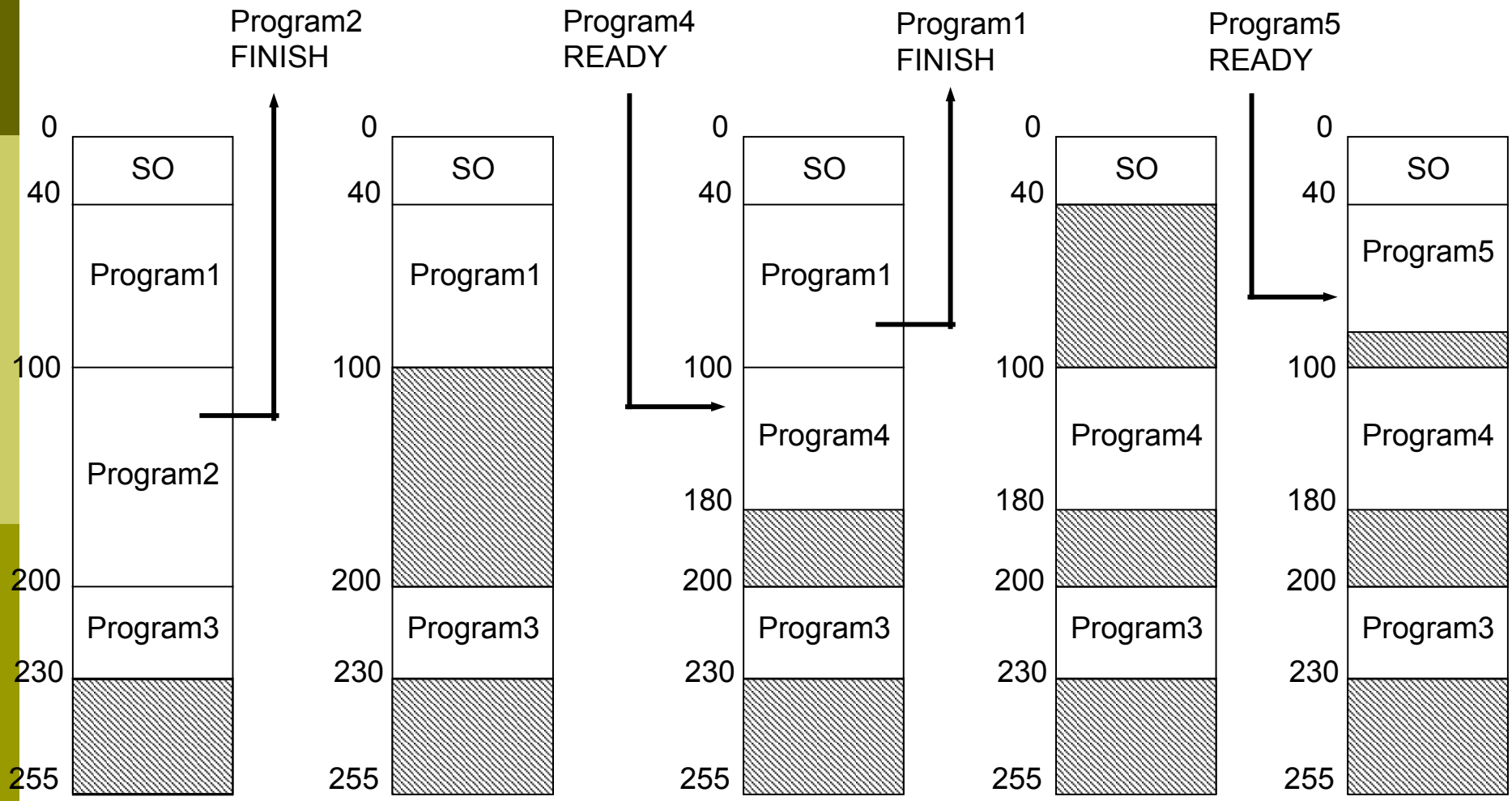
Alocarea cu partiţii fixe

- Legarea proceselor de partiţii
  - fiecare partiţie are o coadă proprie:
    - este o metodă mai simplă din punctul de vedere al sistemului de operare (se întâlneşte la primele sisteme monoutilizator IBM OS/MFT).
  - o singură coadă pentru toate partiţiile:
    - se poate alege partiţia cea mai potrivită pentru plasarea unui proces.

Alocarea cu partiţii variabile

- Această alocare se mai numeşte şi alocare dinamică sau alocare **MVT – Memory Variable Task** (era folosită la PDP11).
- Este o extensie a alocării cu partiţii fixe şi permite o exploatare mai eficientă a memoriei sistemului de calcul:
  - numărul şi dimensiunea partiţiilor se modifică automat în funcţie de:
    - solicitări
    - de capacitatea memoriei rămasă disponibilă la un moment dat

Alocarea memoriei cu partiții variabile



Fragmentarea internă a memoriei

- La intrarea în sistem, procesele sunt plasate în memorie într-un spaţiu în care încape cea mai lungă ramură a sa.
- Spaţiul liber în care a intrat procesul se împarte în două partiţii: una în care este procesul şi una liberă.
- Dacă sistemul funcţionează un timp îndelungat se ajunge la situaţia în care numărul spaţiilor libere va creşte, iar dimensiunea lor va scădea.

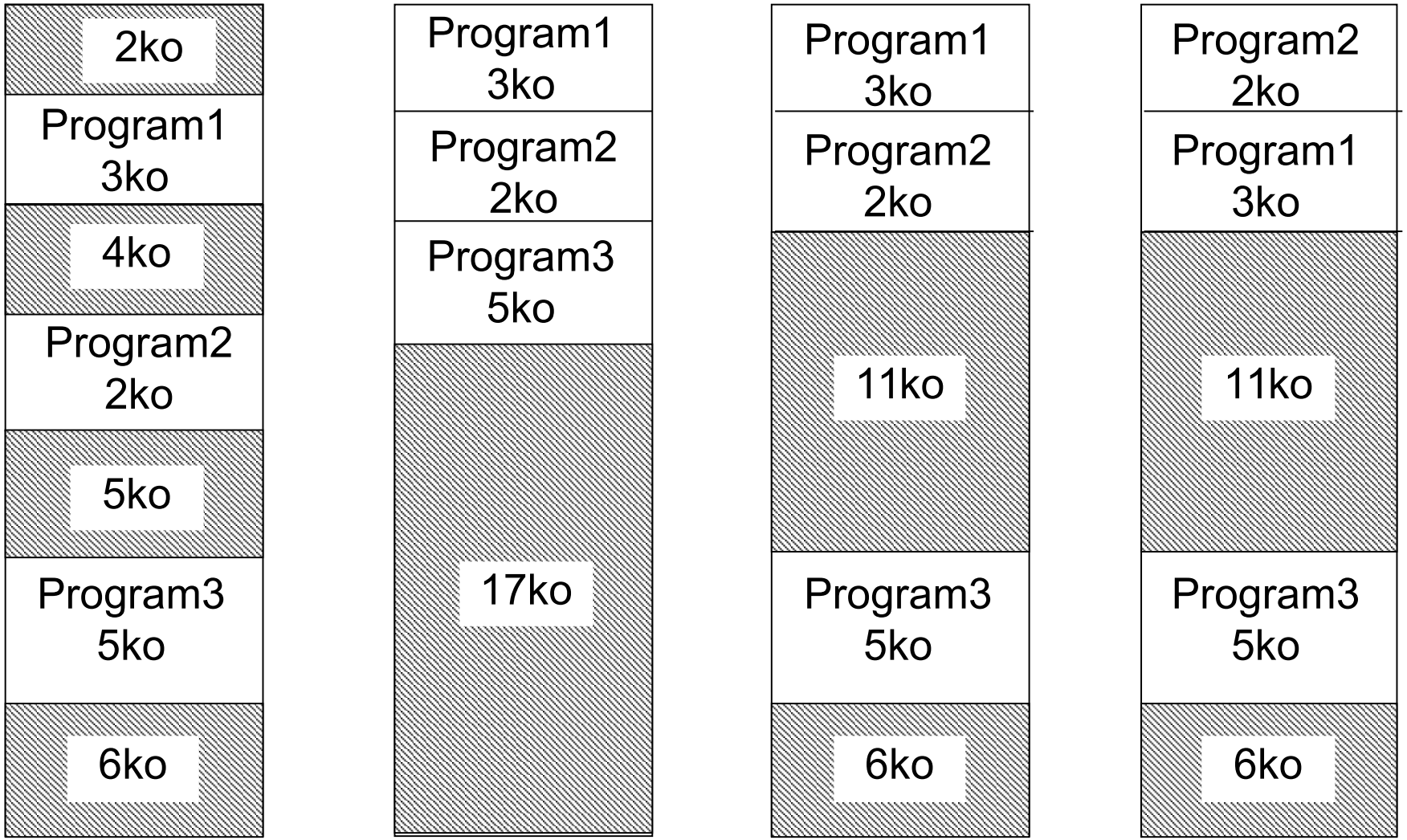
Fragmentarea internă a memoriei

- Dacă un proces nu are spaţiu să se încarce în memorie, sistemul de operare poate lua următoarele decizii:
  - procesul aşteaptă până se eliberează o cantitate suficientă de memorie;
  - sistemul de operare încearcă alipirea unor spaţii de memorie libere vecine, pentru a obţine un spaţiu de memorie liber suficient de mare pentru încărcarea programului;
  - sistemul de operare decide efectuarea unei operaţii de compactare a memoriei (relocare) – deplasarea partiţiilor active către partiţia unde se află partea rezidentă a SO pentru a se absorbi toate "fragmentele" de memorie neutilizate.

Compactarea memoriei

- Compactarea memoriei este, de regulă, o operaţie costisitoare şi în practică se aleg soluţii de compromis:
  - se lansează periodic compactarea, indiferent de starea sistemului. Procesele care nu au loc în memorie aşteaptă compactarea sau terminarea altui proces;
  - se realizează o compactare parţială pentru a asigura loc numai procesului care aşteaptă;
  - se încearcă numai mutarea unora dintre procese şi se colecţionează spaţiile libere rămase.

Compactarea memoriei



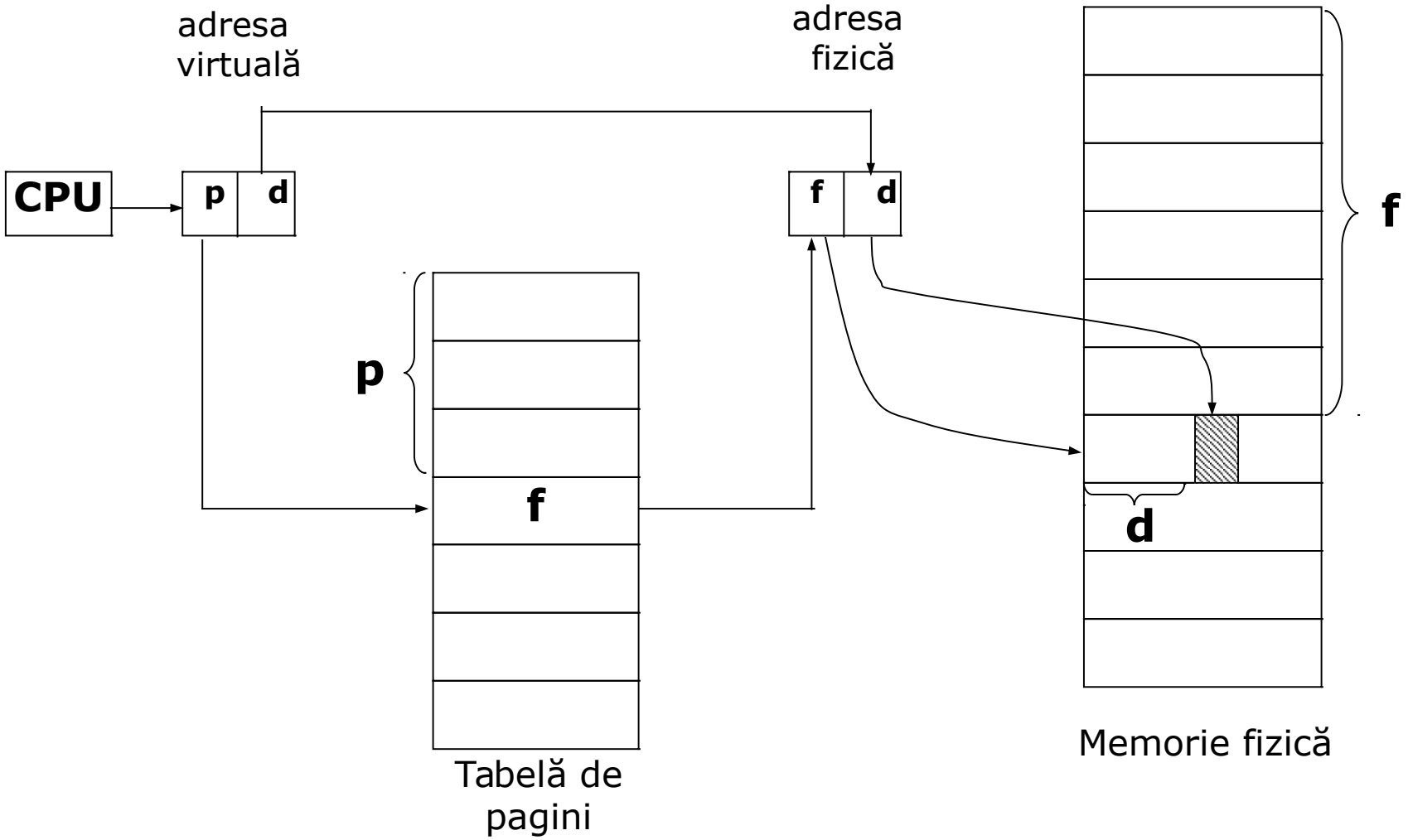
Alocarea paginată a memoriei

- Paginarea este o metodă care rezolvă problema fragmentării:
  - memoria alocată unui program nu este contiguă, adică programul poate fi încărcat în memorie acolo unde există memorie disponibilă.
- Această alocare presupune că:
  - instrucţiunile şi datele fiecărui program sunt împărţite în zone de lungime fixă, numite pagini virtuale. Fiecare adresă relocabilă (AR) aparţine unei pagini virtuale. Paginile virtuale se păstrează în memoria secundară.
  - memoria operativă este împărţită în **zone de lungime fixă**, numite **pagini fizice** sau **cadre**.
    - Lungimea unei pagini fizice este fixată prin hardware.
    - Paginile virtuale şi cele reale au aceeaşi lungime (o putere a lui 2) şi reprezintă o constantă a sistemului (de exemplu: 1Ko, 2Ko etc).

Translatarea unei pagini virtuale într-o pagină fizică

- fiecare **AR (adresă relocabilă)**este o pereche de forma **(p, d)** unde **p** este numărul paginii şi **d** este adresa în cadrul paginii (deplasarea în pagină).
- fiecare **AF (adresă fizică)** este o pereche de forma **(f, d)** unde **f** este numărul paginii fizice şi d este adresa în cadrul paginii.
- calculul funcţiei de translare se face prin hardware.

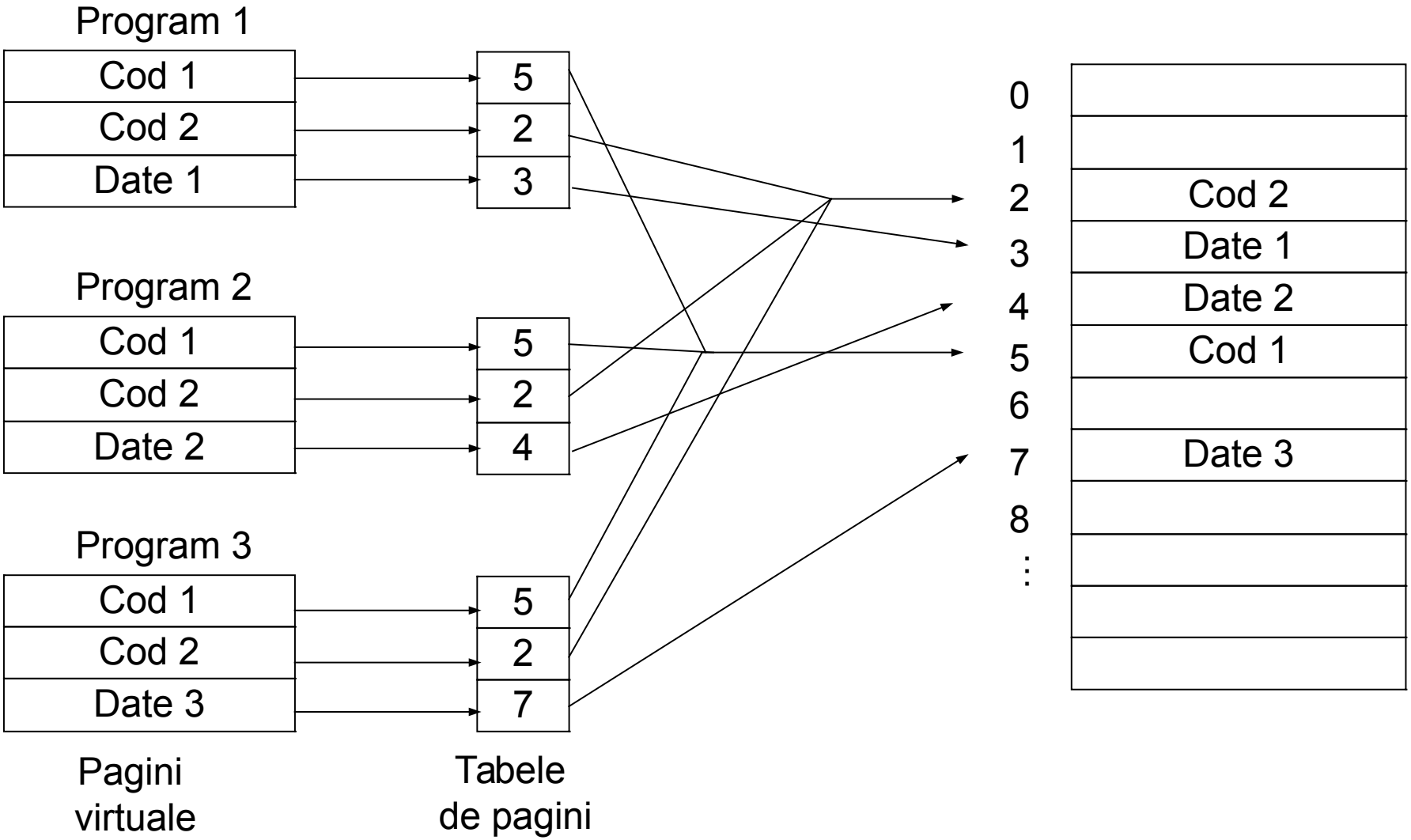
Translatarea unei pagini virtuale într-o pagină fizică



Alocarea paginată a memoriei

- 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 şi în acest fel are loc o proiectare a spaţiului virtual peste cel real
  - se foloseşte mai eficient memoria operativă.
  - folosirea în comun a instrucţiunilor unor proceduri de către mai multe programe. O procedură care permite acest mod de lucru se numeşte ***procedură reentrantă (cod reentrant - instrucţiuni pure fără date)***.

Procedură reentrantă alocată cu paginare



Implementarea tabelei de pagină

- Dacă dimensiunea tabelei de pagină este redusă se poate utiliza un set de registre specializate, foarte rapide, care să asigure o eficienţă ridicată a translării adreselor.
- Instrucţiunile destinate încărcării sau modificării acestor registre trebuie accesate numai de către sistemul de operare.
- Dacă dimensiunea tabelei este mare, este preferabil ca ea să fie păstrată în memoria principală, într-o zonă indicată de valoarea unui registru specializat numit **registrul de bază al tabelei de pagină (RBTP)**

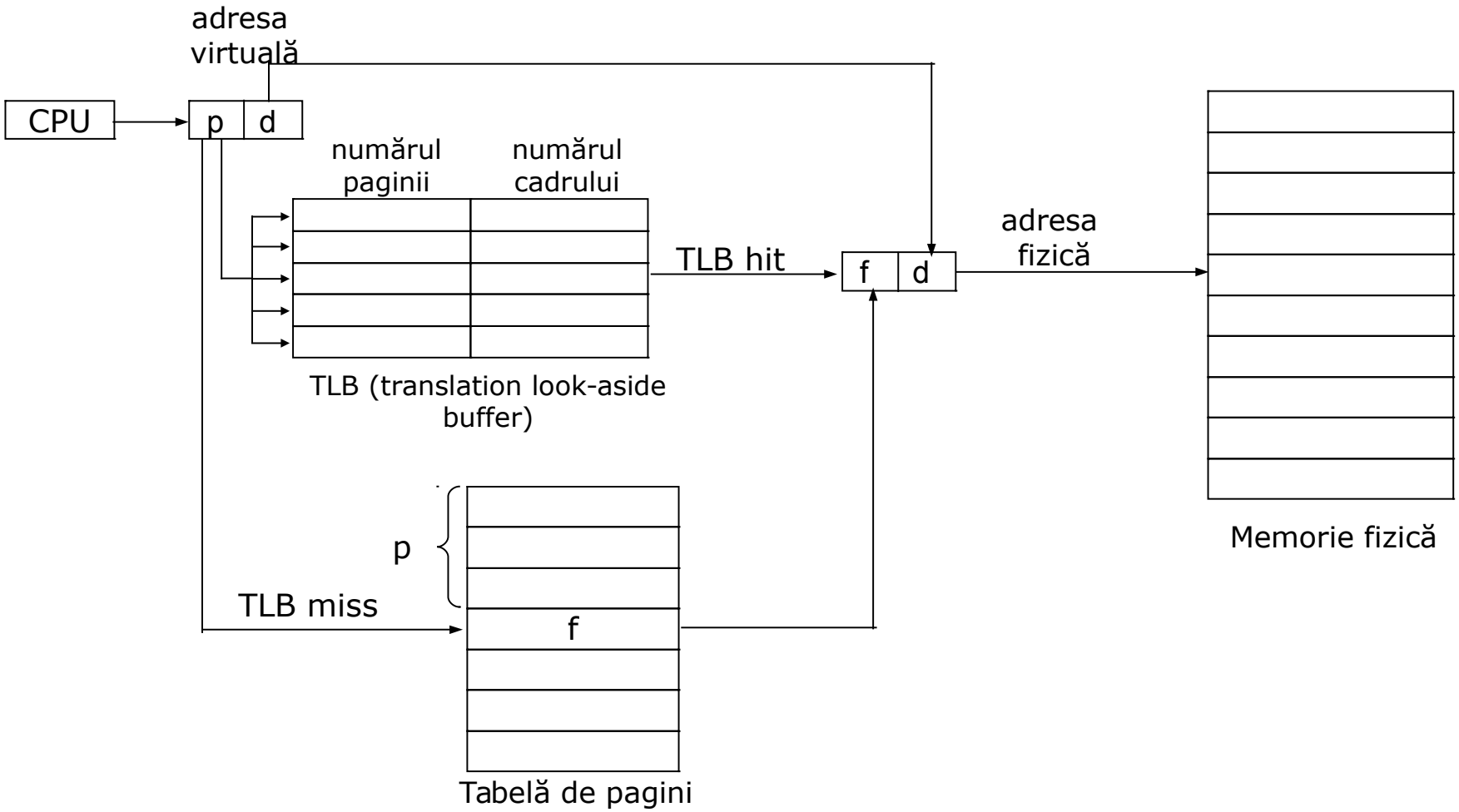
Implementarea tabelei de pagină

- Dacă trebuie să se lucreze cu o altă tabelă de pagină decât cea curentă trebuie reîncărcat registrul cu o altă valoare scăzând astfel timpul de schimbare al contextului.
- O particularitate a acestei soluţii este faptul că pentru a accesa o zonă de memorie utilizator sunt necesare două operaţii de acces la memorie – una pentru tabela de pagină şi alta pentru cuvântul propriu-zis.
- Folosind valoarea din RBTP deplasată cu numărul de pagină - p (aflat în adresa logică), se determină mai întâi numărul de cadru – c asociat paginii, care împreună cu deplasamentul în pagină – d dă adresa reală.

Implementarea tabelei de pagină

- O altă soluţie ar fi folosirea unei memorii hardware speciale (un set de registre asociative sau **translation look-aside buffer - TLB**), de mică dimensiune, cu următoarele caracteristici:
  - fiecare registru are două părţi, cheie (conţine numărul paginii) şi valoare (numărul cadrului).

Translarea paginii folosind TLB



Paginarea multinivel

- Sistemele de calcul suportă un spaţiu logic de adresare foarte mare (2^{32} sau 2^{64}) şi din acest motiv tabela de pagini trebuie să fie foarte mare.
- Pentru un sistem cu spaţiul logic de adresare pe 32 de biţi , dacă avem mărimea paginii de 4K bytes (2^{12}), atunci numărul de intrări în **tabela de pagini** ar trebui să fie de peste 1 milion (2^{32} / 2^{12} = 2^{20}=1048576).
- Deoarece fiecare intrare constă în 4 bytes, fiecare proces poate avea nevoie de 4 Mbytes de spaţiu de adresare pentru tabela de pagini.
- Este aproape imposibil să alocăm tabela de pagini într-o zonă contiguă de memorie.

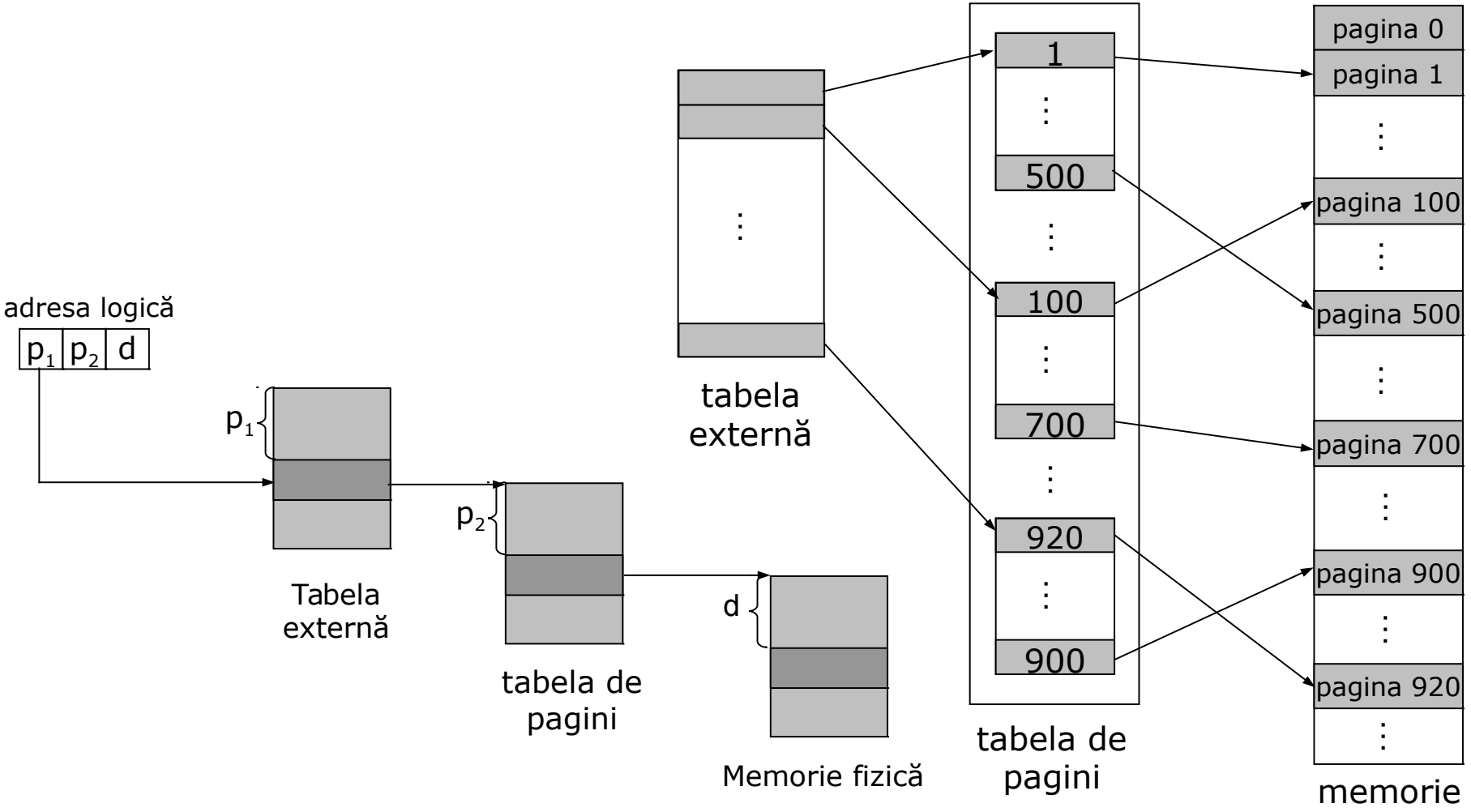
Paginarea multinivel

- Soluţia pentru rezolvarea acestei probleme este paginarea multinivel:
  - adresa logică o putem împărţi în numărul de pagină de 20 biţi şi deplasamentul în pagină de 12 biţi.
- Deoarece vrem să paginăm tabela de pagini, putem împărţi numărul paginii în două părţi:

Numărul paginii		deplasament
p_1	p_2	d
10	10	12

  - unde $\mathbf{p_1}$ este index în tabela de pagini externă, $\mathbf{p_2}$ este deplasamentul în această tabelă şi **d** este deplasamentul în pagina de memorie.

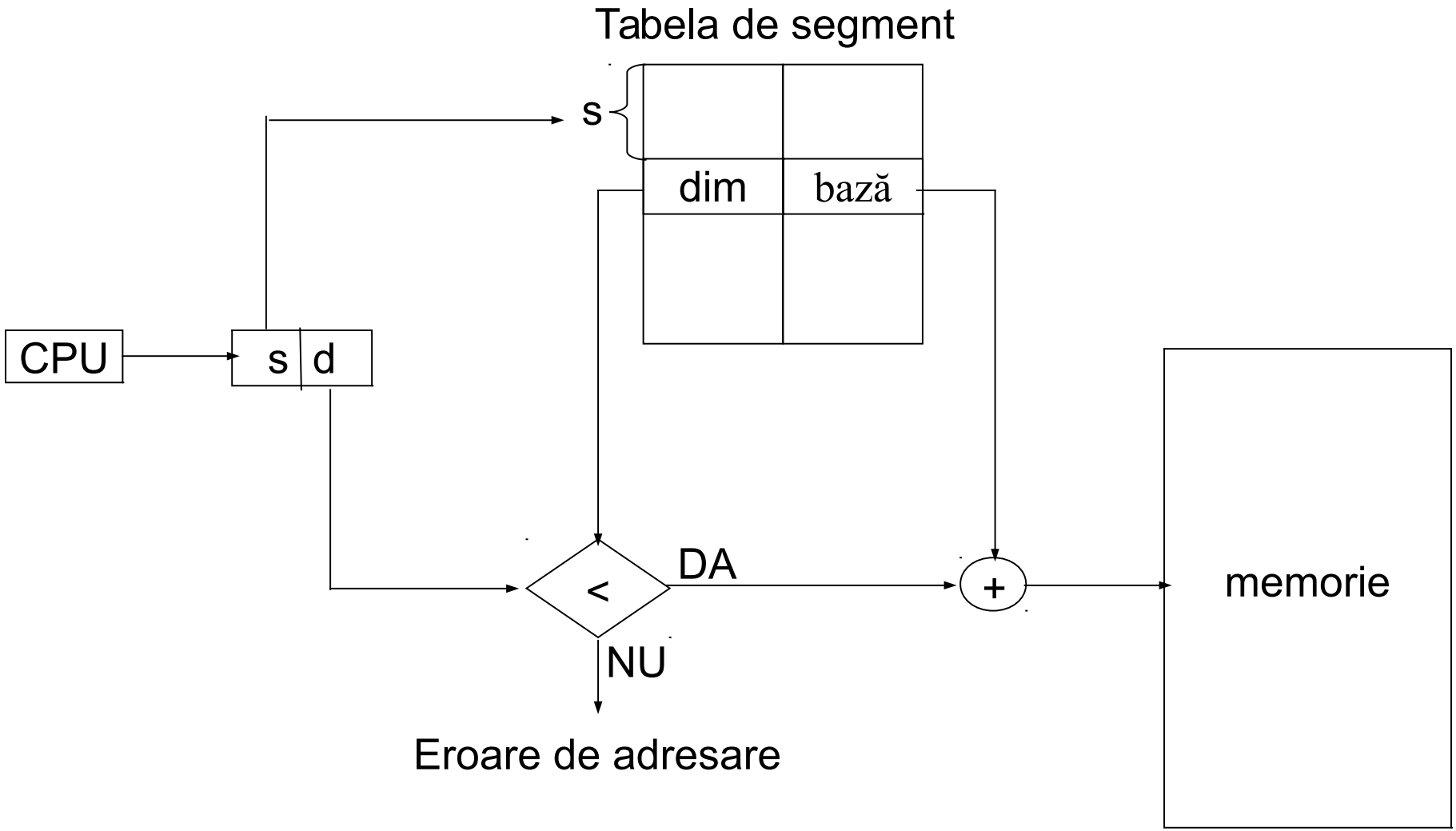
Translarea adresei în cazul paginării cu 2 niveluri



Alocarea segmentată a memoriei

- Mecanismul de alocare segmentată introduce faptul că textul unui program poate fi plasat în zone de memorie distincte, fiecare zonă conţinând o bucată de program numită segment.
- Fiecare segment este caracterizat prin nume şi lungime.
- O adresă virtuală este o pereche **(s, d)**, unde **s** este numărul segmentului şi **d** este deplasamentul în cadrul segmentului.
- Acestei perechi îi corespunde o adresă fizică, iar corespondenţa este realizată prin tabela de segment ce conţine un număr de intrări egal cu numărul de segmente din program.

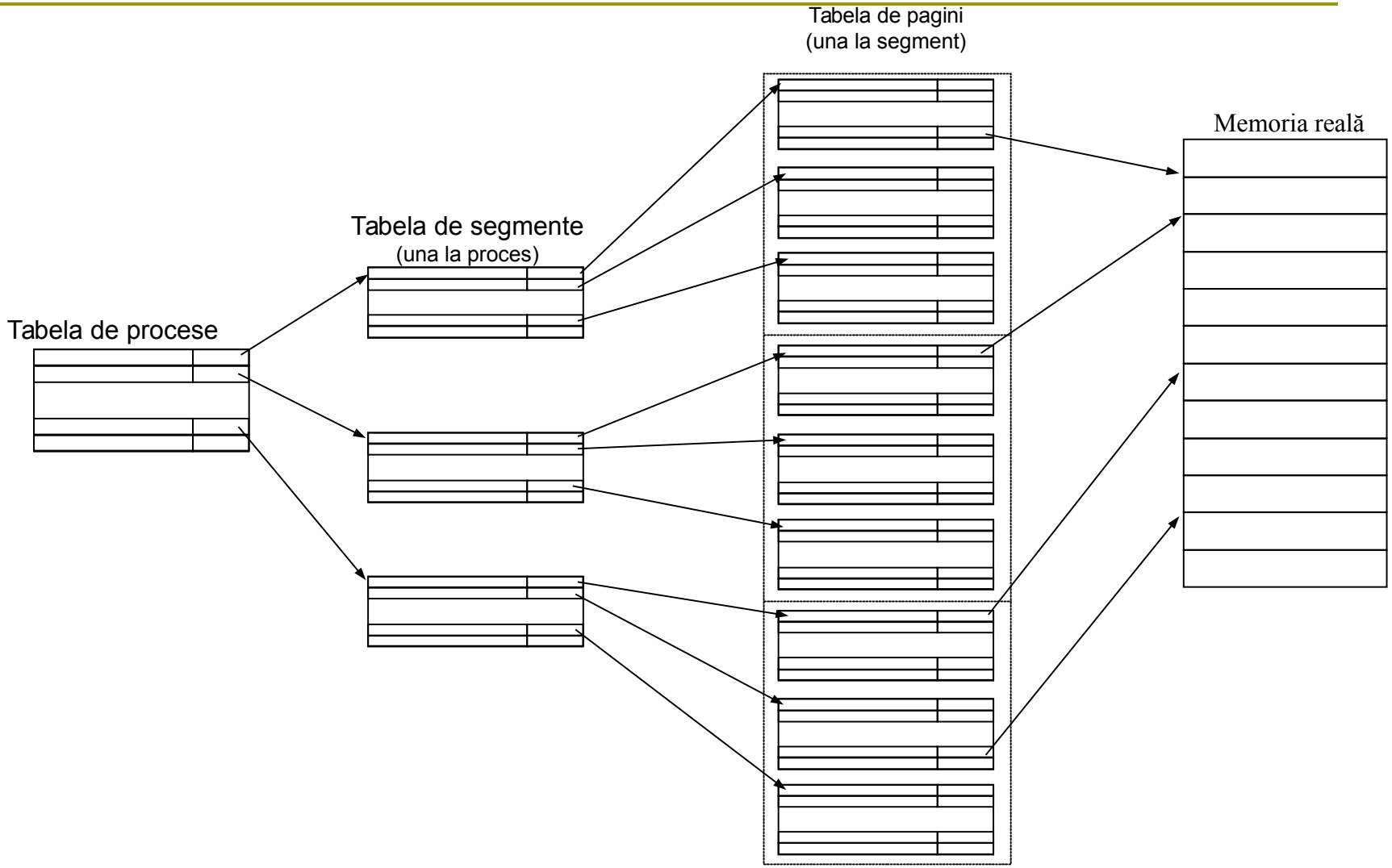
Translatarea adresei virtuale cu ajutorul tabelei de segment



Avantaje față de alocarea pe partiții

- se pot crea **segmente reentrante** ce pot fi folosite de mai multe procese.
- se poate realiza protecţia memoriei prin adăugarea unor drepturi de acces.

Alocarea segmentată şi paginată



Alocarea segmentată şi paginată

- Fiecare proces are propria lui tabelă de segmente.
- Fiecare segment are propria lui tabelă de pagini.
- Fiecare intrare în tabela de segmente are un câmp rezervat adresei de început a tabelei de pagini proprii segmentului.
- Adresa virtuală este de forma **(s, p , d)**, unde **s** este numărul segmentului, **p** este numărul paginii virtuale în cadrul segmentului, iar **d** este deplasamentul în cadrul paginii.
- Adresa fizică este de forma **(f, d)**, unde **f** este numărul paginii fizice, iar **d** este deplasamentul în cadrul paginii.

Paginarea segmentelor