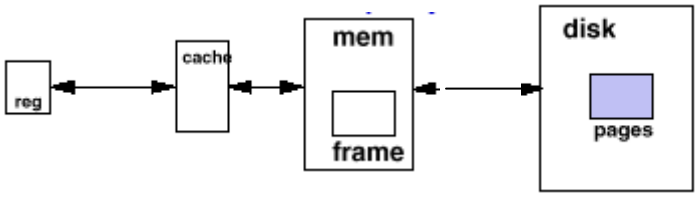
Cursul 8_4- Continuare

Aspectele ale Proiectarii Sistemului de Memorie Virtuala:

- Dimensiunea blocurilor de informatii, care se transfera de la memoria secundara la memoria principala M ?→ **dimensiunea blocului;**
- Blocul de informatii trebuie adus in M, iar M este plin, atunci o regiune din M trebuie eliberata pentu a face loc noului bloc → **politica de inlocuire;**
- Care regiune din M trebuie sa fie ocupata de catre noul bloc? →**politica de plasare**;
- Elementul care lipseste este extras din memoria secundara numai la aparitia unei erori de pagina (page fault) → **politica de cereri de incarcare**;



Organizarea pe pagini

Spatiul de adrese virtuale si fizice este partitionat in blocuri de dimensiuni egale

pagini cadre

Maparea Adreselor

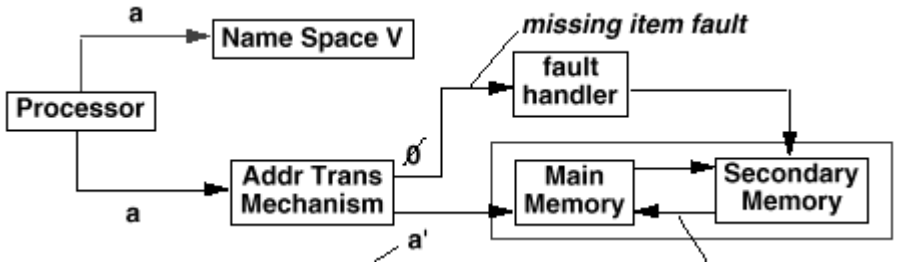
V = {0,1,….,n-1} - spatiul adreselor virtuale;

M = {0,1,….,m-1} - spatiul adreselor fizice;

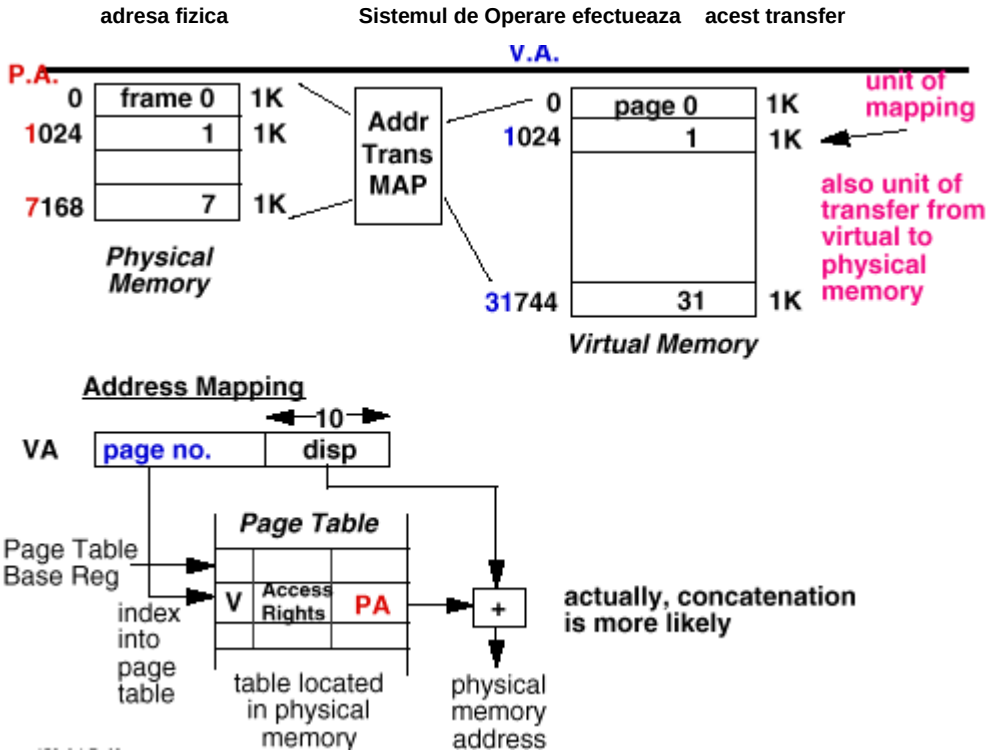
MAP : V → M ∪ {0} - functia de mapare a adreselor (n > m)

MAP(a) = **a'** daca **data** de la adresa virtuala **a** este prezenta in cea fizica **a'** si **a'** in M

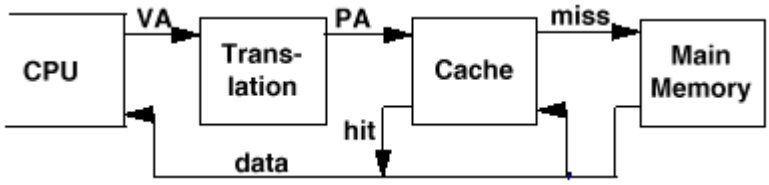
= **0** daca **data** de la adresa virtuala a nu este prezenta in **M**



Organizarea pe pagini



Adrese Virtuale si Memoria Intermediara



Este necesar un acces suplimentar la memorie pentru a translata AV in AF

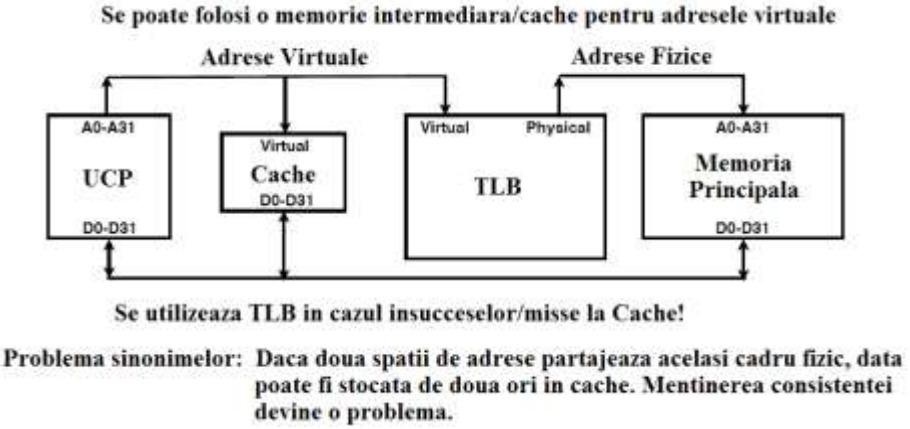
Aceasta face accesul la memoria intermediara extrem de costisitor, iar aceasta bucla interna-primara trebuie sa fie executata cat mai repede;

LATERAL (aside): De ce trebuie accesata memoria intermediara cu AF (Adrese Fizice)?

Memoriile intermediare pentru AV au probleme:

- *Sinonime/*problema *alias*: doua adrese virtuale diferite se mapeaza in aceeasi adresa fizica => doua intrari diferite “cache” pastreaza data pentru aceeasi adresa fizica!

- Pentru actualizare: trebuie aduse la zi toate intrarile ”cache” cu aceeasi adresa fizica sau memoria devine inconsistenta;
- Aceasta decizie necesita un hardware substantial, in esenta o cautare asociativa pe etichetele adreselor fizice pentru a vedea daca exista succese (hits) multiple sau limita sinonima fortata hardware: acelasi **lsb** pentru AV si AF > dimensiunea “cache”.



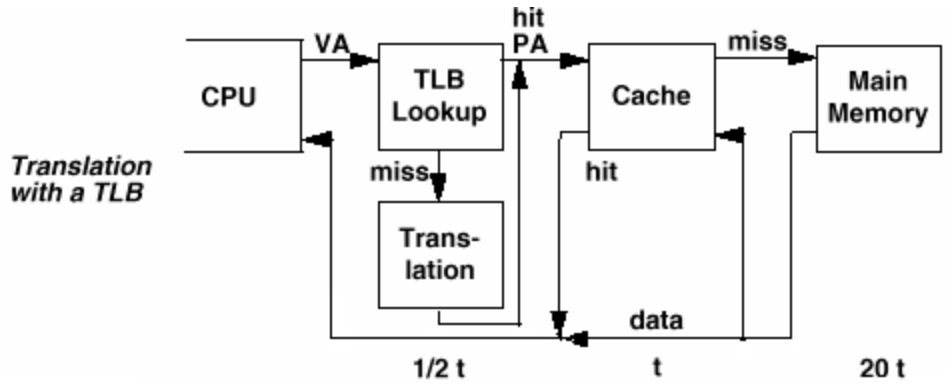
TLB Translation Look-Aside Buffer

O modalitate de accelerare a traducerii consta in utilizarea unui “cache” special pentru intrarile in tabela de pagini cele mai recent utilizate, care are numeroase nume, cel mai frecvent fiind: **TLB Translation Look-aside Buffer**

Virtual Address	Physical Address	Dirty	Ref	Valid	Access

Timpul de acces la TLB este comparabil cu timpul de acces la memoria intermediara (mult mai mic decat timpul de acces la memoria principala)

Ca si oricare memorie “cache”, TLB poate fi organizat complet asociativ sau prin mapare directa. TLB sunt, de regula, de dimensiuni mici cu 128-256 intrari/locatii chiar pe sistemele mari. Aceasta permite organizarea complet asociativa. Sistemele de capacitate medie folosesc organizarea asociativa pe seturi cu n cai



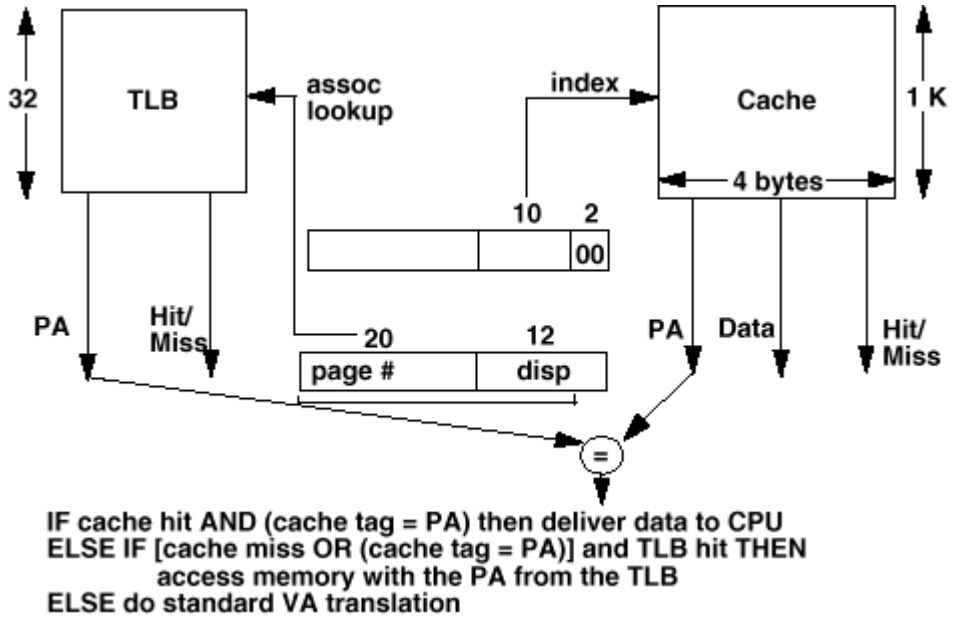
Reducerea timpului de traducere/translatare

Sistemele cu TLB merg mai departe cu un pas in ceea ce priveste reducerea numarului de cicluri/accese la “cache”.

Ele suprapun accesele la “cache” cu accesele la TLB.

Aceasta functioneaza deoarece bitii superiori ai AV sunt utilizati pentru cautare in TLB, in timp ce bitii inferiori sunt folositi ca index in “cache”

Accesele suprapuse la “Cache” si TLB

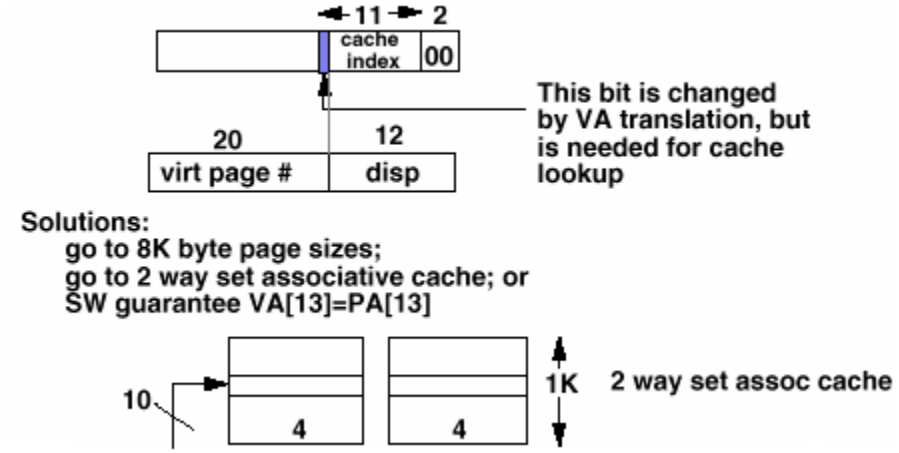


Probleme privind accesul suprapus la TLB

Accesul suprapus functioneaza cat timp bitii de adresa utilizati pentru indexarea in “cache” nu se schimba ca rezultat al traducerii/translatarii AV

Aceasta limiteaza operarea la memorii “cache” de dimensiuni mici, pagini mari sau memorii “cache” mari asociative pe seturi cu n cai, daca se doreste o memorie “cache” de mari dimensiuni.

Exemplu: considera ca nimic nu se schimba cu exceptia capacitatii memoriei “cache”: de la 4 Kbytes la 8 Kbytes



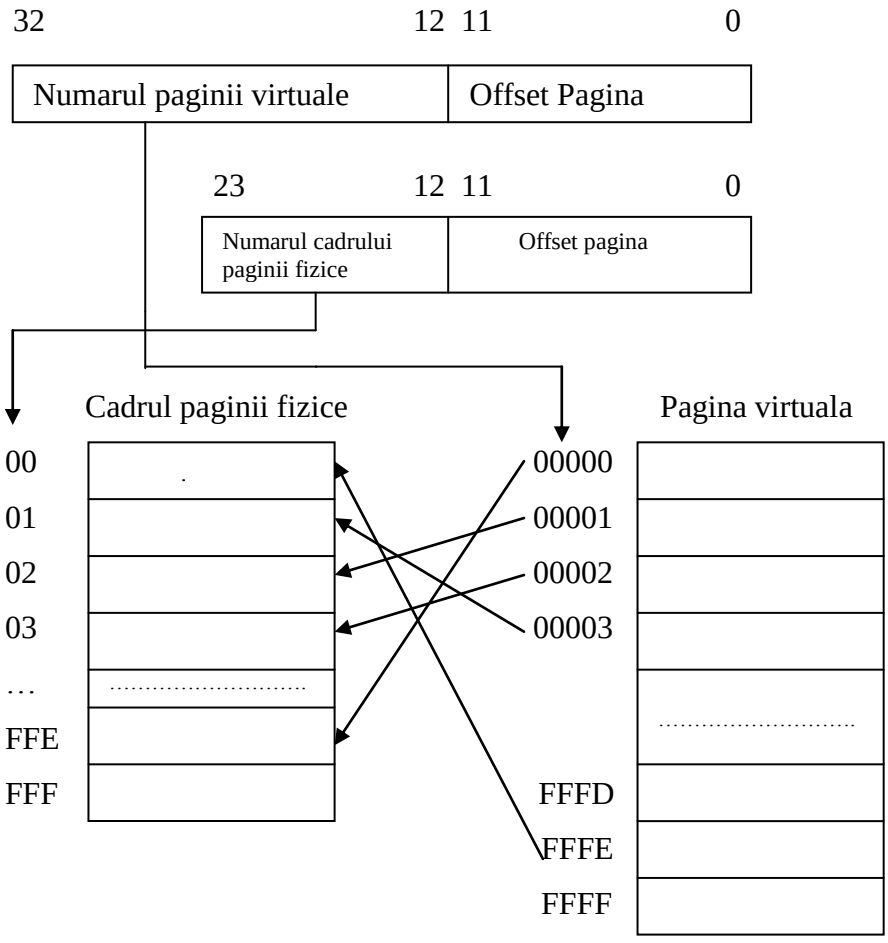
TLB, Memoria Virtuala

- Memoriile intermediare, TLBurile, Memoria Virtuala sunt intelese prin examinarea urmatoarelor patru aspecte:
  1. Unde poate fi plasat un bloc?
  2. Cum poate fi gasit un bloc?
  3. Care bloc este inlocuit in caz de insucces?
  4. Cum sunt rezolvate scrierile?
- Tabelele de pagini mapeaza AV in AF.
- TLB sunt importante pentru traducerea/translatarea rapida
- Insuccesele TLB sunt importante pentru performanta procesorului (cele mai multe sisteme nu pot accesa memoriile intermediare de pe nivelul 2, fara insuccese TLB)
- Memoria Virtuala a fost controversata cu ani in urma: se pot rezolva prin software cerintele diferitelor programe in conditiile unei memorii de 64KB?
  Cresterea capacitatii DRAM de 1000 de ori a rezolvat problema

- Actualmente MV permit mai multor procese sa partajeze o singura memorie fara necesitatea unui “swap” pentru toate procesele pe disc; protectia MV este mai importanta decat ierarhia de memorii
- Timpul UCP este o functie de numarul de operatii/s si numarul insuccesselor la “cache” si nu numai de numarul de operatii/s

Memoria Virtuala (continuare).

Se considera o memorie virtuala cu o capacitate de 2^{32} octeti, organizata in 2^{20} pagini virtuale de cate 4Kocteti (1Kcuvant: 32 de biti/cuvant). Memoria fizica are o capacitate de 16 Mocteti, care se structureaza in 2^{12} cadre de pagini. Campurile de adrese virtuale si fizice, cat si maparea lor sunt date mai jos:



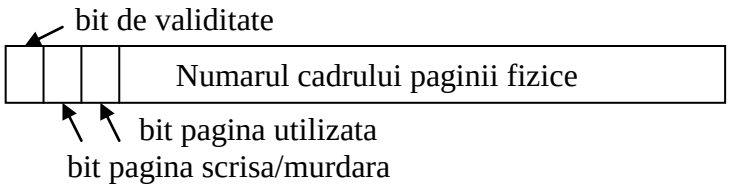
Tabelele de pagini

In general, exista un numar mare de pagini virtuale, fiecare dintre acestea fiind mapata, fie in memoria principala, fie in memoria secundara.

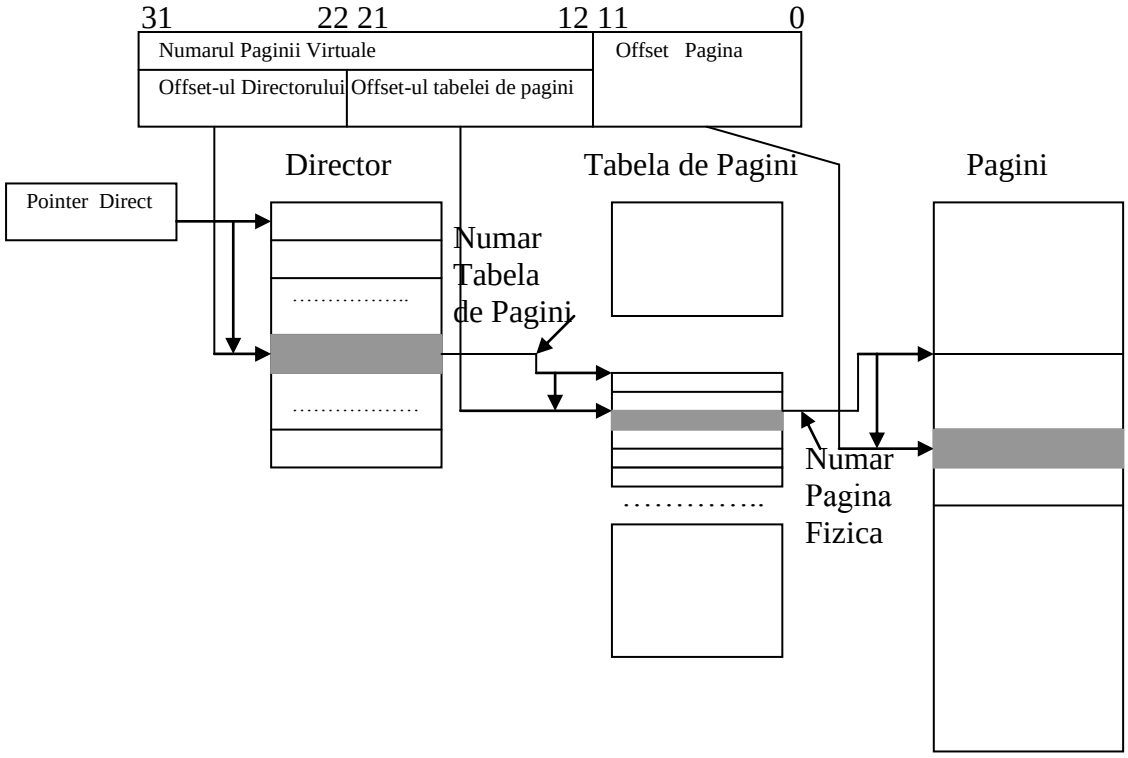
Maparea este stocata intr-o structura de date numita tabela de pagini. Tabelele de pagini pot fi structurate si accesate in diverse moduri.

Se considera ca tabelele de pagini sunt stocate tot in pagini si ca reprezentarea unei mapari necesita un cuvant, ceea ce inseamna ca 1K mapari se pot stoca intr-o pagina. (4Kocteti).

Pentru fiecare program exista o tabela numita pagina director, in care sunt pastrate maparile de pagini pentru programul dat. Continutul unei locatii din tabela de pagini de pagini este dat mai jos:



Exemplu de structura de Tabela de Pagini



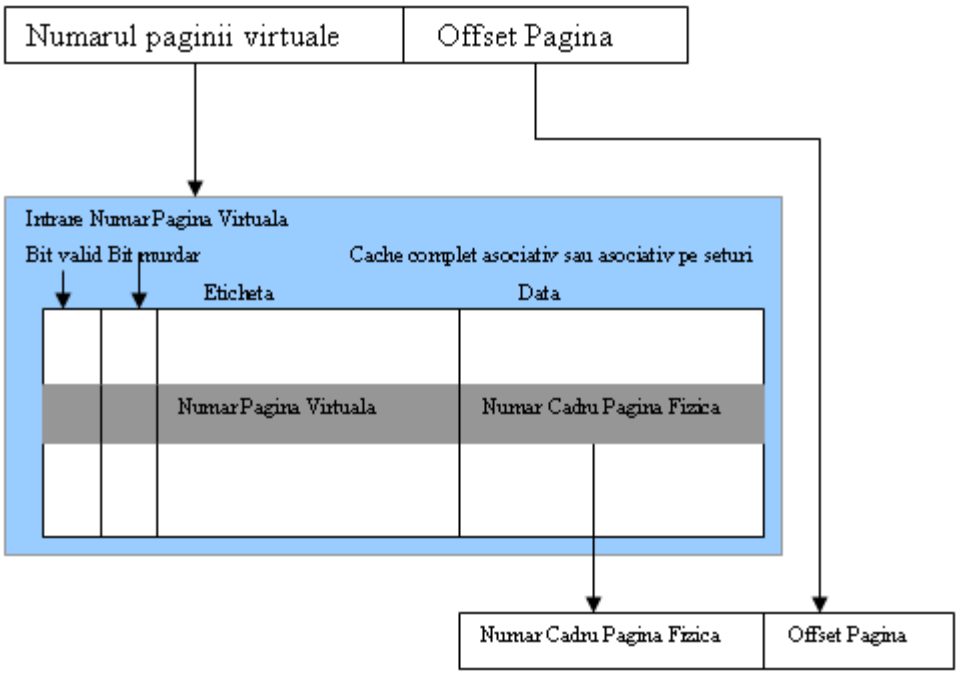
Se observa ca memoria virtuala prezinta penalizari de timp importante, chiar in cazul in care directorul, tabela de pagini si pagina, care trebuie accesata, se afla in memoria principala.

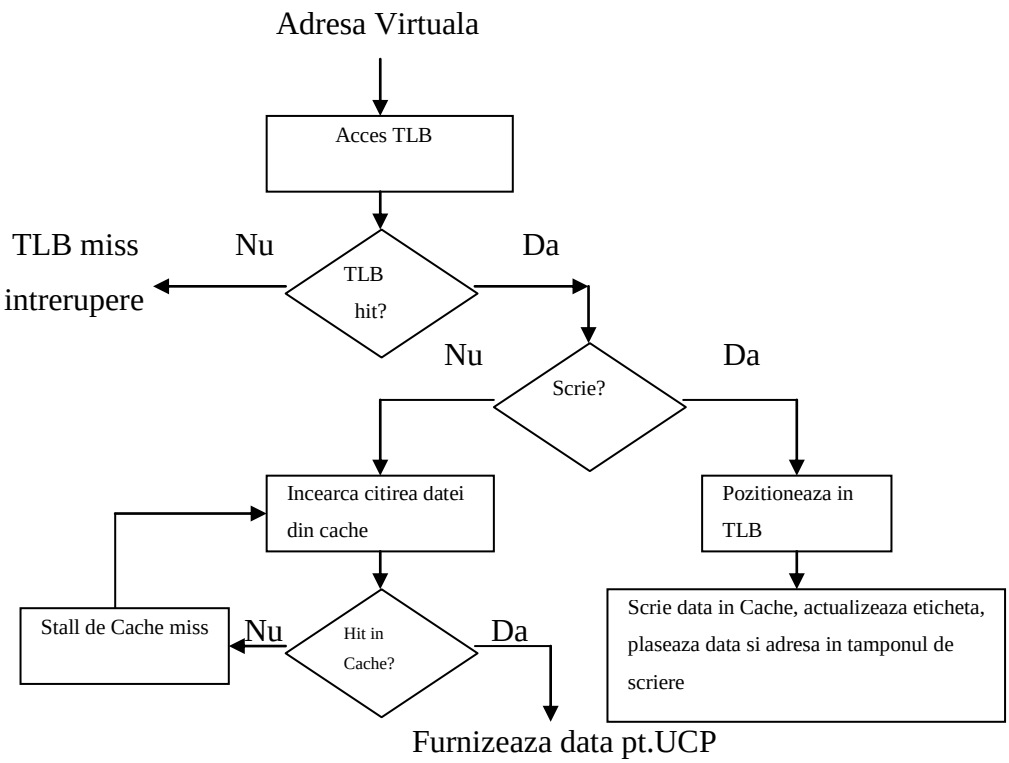
Pentru a accesa un operand sau o instructiune sunt necesare trei accese la memorie:

- pentru locatia in director;
- pentru locatia in tabela de pagini;
- pentru operand sau instructiune.

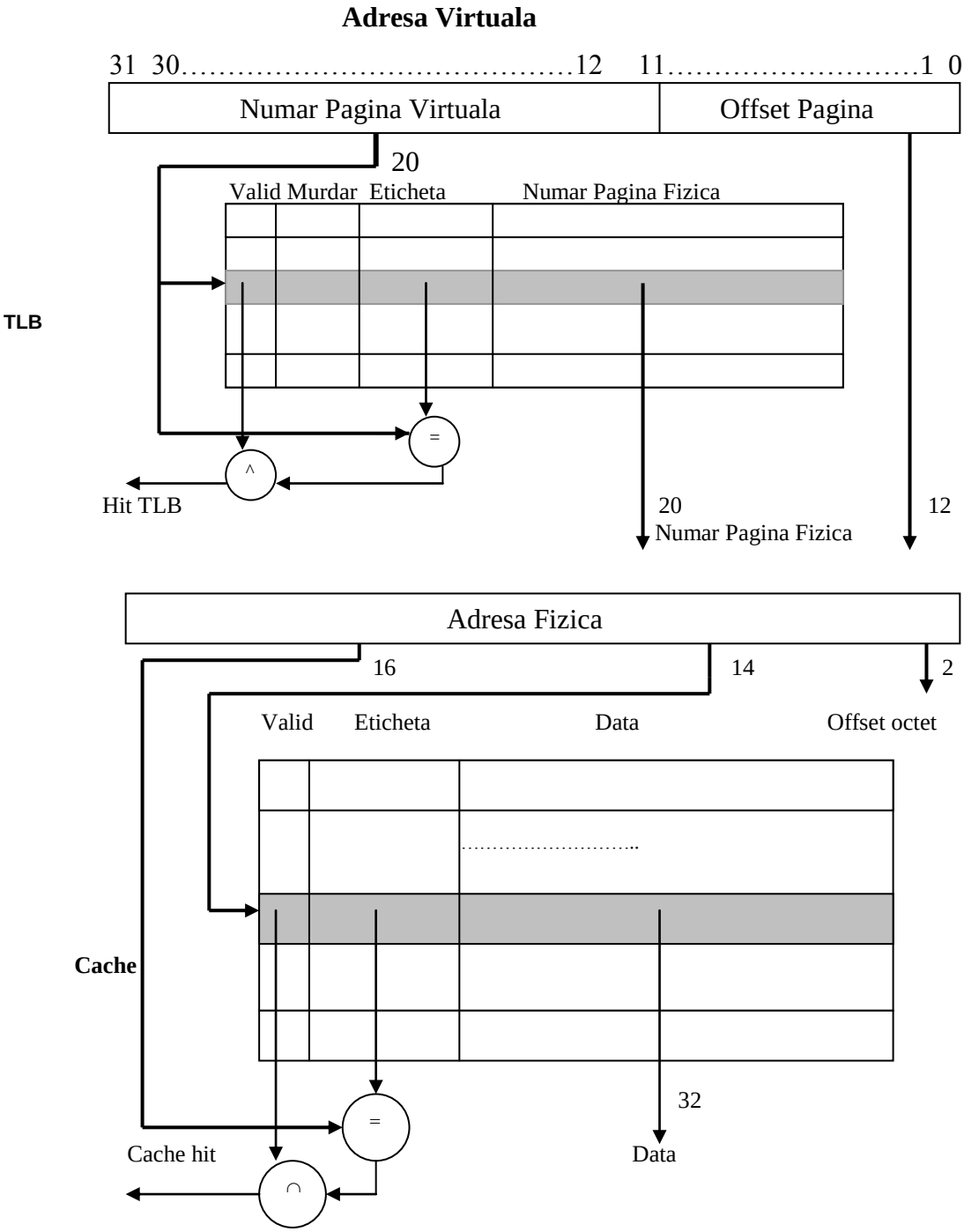
Acesta sunt realizate automat prin hardware de catre unitatea de management a memoriei. Daca aceste informatii se afla in cache trebuie sa se execute trei accese la cache.

Pentru rezolvarea acestei probleme se foloseste o alta memorie cache care traduce adresa virtuala in adresa fizica. Este vorba de Translation Lookaside Buffer (TLB). TLB stocheaza locatiile recent utilizate.





Prelucrarea unei citiri sau scrieri in TLB-ul si Cache-ul lui DECStation 3100



DECStation 3000: TLB si Cache implementeaza procesul de tranzitie de la adresa virtuala la data.

- **A TLB acts as a fast cache for recent address translations.**
- **Operating systems manage the page table and (often) the TLB**