Memoria virtuală
- continuare -

Putem folosi scheme de memorii cache cu scrieri simultane doar dacă implementăm bufer-e de memorie (diferenţa dintre timpul de acces al memoriei cache şi cel al memoriei principale este de ordinul zecilor de cicluri)

Sistemele de memorie virtuală trebuie să implementeze scrierea la loc (copy back) – se execută scrieri individuale în pagina de memorie şi se copiază această pagină înapoi pe disc, atunci când este înlocuită în memorie.

Va trebui să determinăm dacă o pagină de memorie necesită transferul înapoi atunci când ea este înlocuită în memorie.

Pentru aceasta adăugăm un bit de scriere ‘’dirty bit’’. Acest bit iniţial 0 va deveni 1 în momentul în care se scrie prima dată în această pagină.

Mărirea vitezei de translatare a adresei

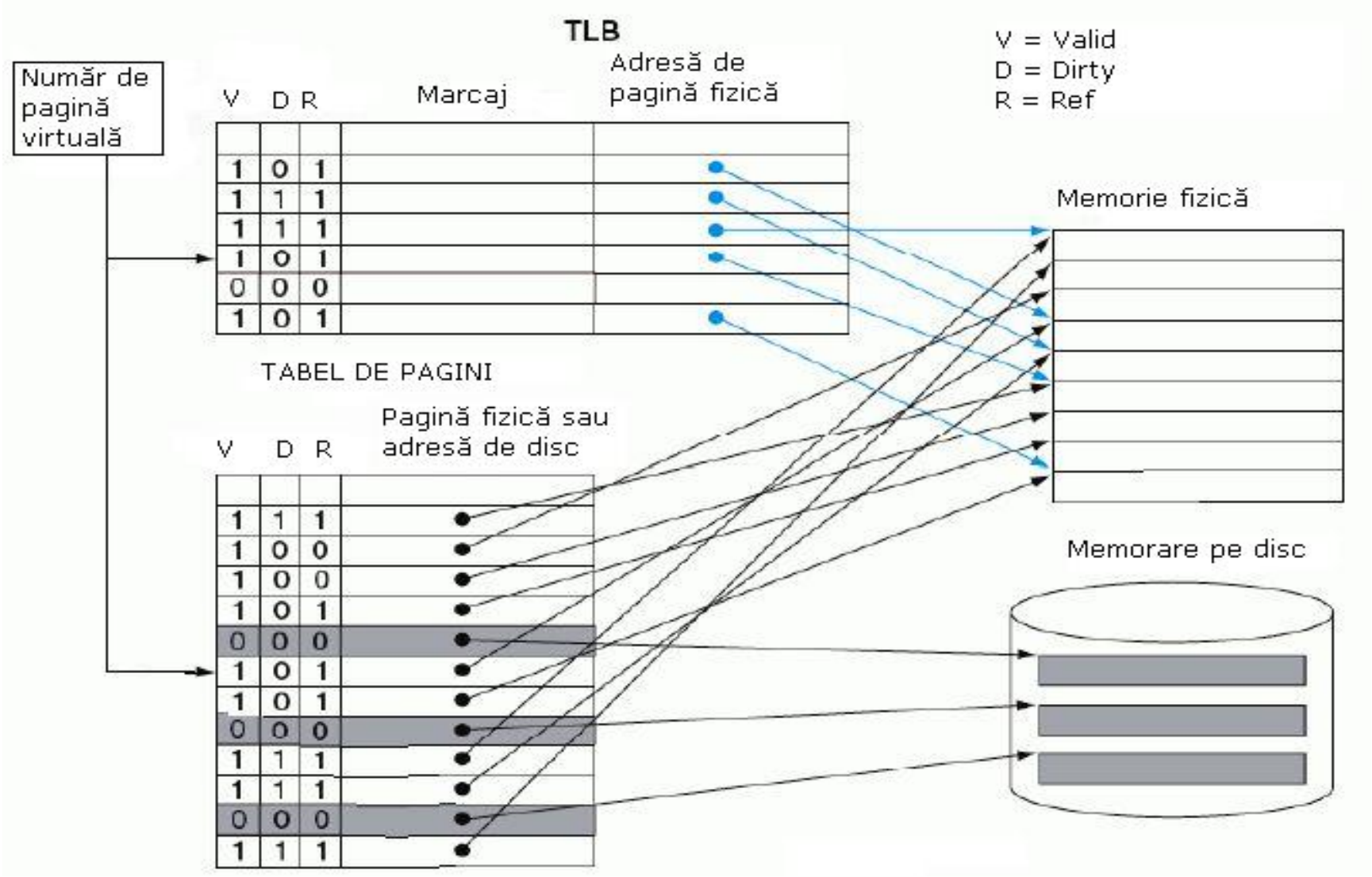
Unde sunt păstrate tabelele de pagini ?

Ce accese la memorie sunt necesare pentru un program ?

Ce principiu putem utiliza pentru mărirea performanţei ?

Când este folosită o translatare pentru un număr de pagină virtuală, aceasta va fi probabil necesară din nou în viitorul apropiat. De ce ?

Din aceste considerente, maşinile moderne includ o memorie cache specială care menţine evidenţa translatărilor recente numită **Translation Lookaside Buffer – memorie tampon de translatare cu căutare laterală - TLB**



TLB acţionează ca şi o memorie cache a tabelului de pagini doar pentru poziţiile acestuia ce au corespondent în paginile fizice

La fiecare referinţă se va căuta numărul de pagină virtuală din TLB.

Daca avem HIT, numărul paginii fizice este utilizat pentru formarea adresei, iar bitul de referinţă corespunzător va fi setat. Bitul de scriere va fi setat şi în cazul în care procesorul execută o scriere

Dacă avem MISS trebuie determinată cauza: greşeală de pagină sau eşec în TLB.

Cazul 1 – eşec în TLB, deci pagina este în memorie => că translatarea lipseşte. UCP-ul va trata eroarea prin încărcarea translatării din tabelul de pagini în TLB şi reluarea referinţei.

Cazul 2 – greşeală de pagină, UCP-ul va invoca sistemul de operare folosind o excepţie.

Eşecurile în TLB vor fi mult mai frecvente decât greşelile de pagină. **De ce ?**

Eşecurile în TLB vor fi tratate software sau hardware. Ambele metode oferă performanţe similare.

Integrarea memoriilor

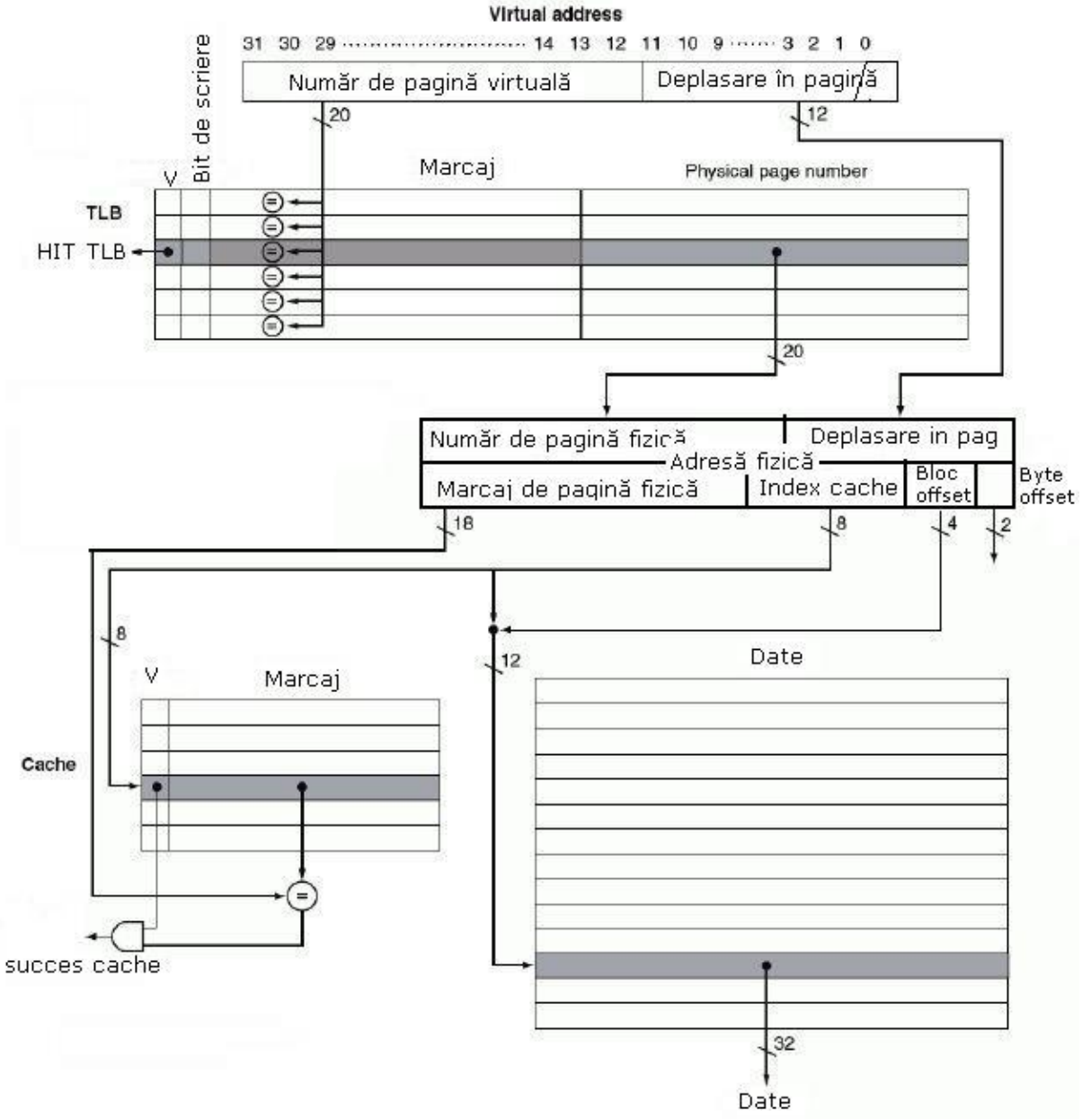
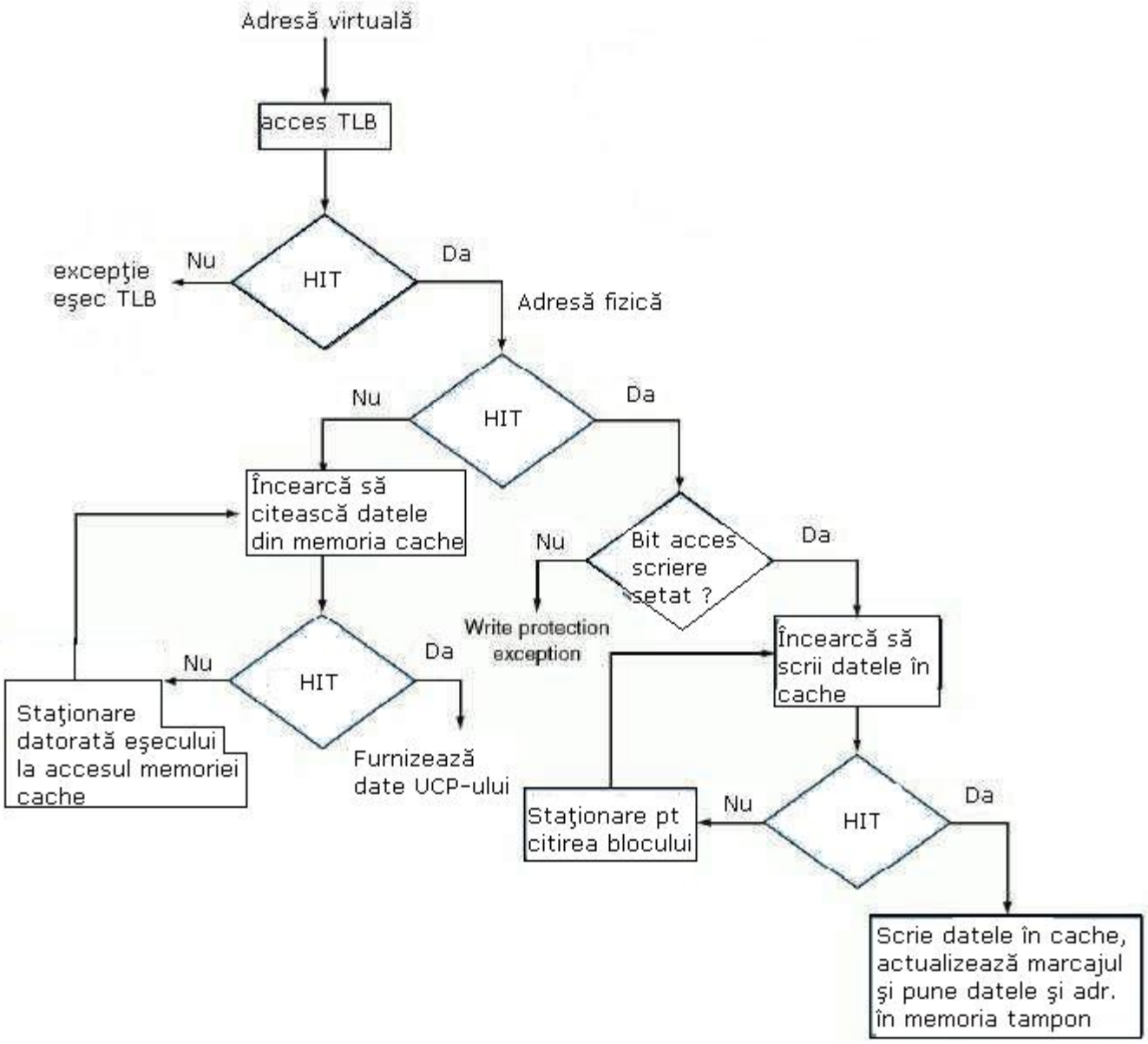


Fig 1.

În timp ce memoria cache are corespondenţă directă, TLB-ul are asociativitate totală. Implementarea asociativităţii totale pentru TLB necesită ca fiecare marcaj din TLB şi fie comparat cu valoarea indexului T deoarece poziţia căutată poate fi oriunde în TLB.

Dacă bitul de validare al poziţiei corecte este setat, accesul la TLB este un succes şi numărul de pagină împreună cu deplasarea acesteia formează indexul ce este folosit pentru accesul la memoria cache.

Citirea sau scrierea cu un TLB şi o memorie cache – DEC Station 3100



Considerăm ierarhia de memorie din figura 1 ce include:

un TLB

o memorie cache

O referinţă la memorie poate avea următoarele 3 tipuri de eşecuri:

un eşec al memoriei cache

un eşec al TLB-ului

greşeală de pagină

Memorie cache	TLB	Memorie virtuală	Eveniment posibil ? În ce condiţii ?
MISS	HIT	HIT	Posibil, tabelul de pagini nu este niciodată verificat la succes TLB
HIT	MISS	HIT	Eşec TLB, dar poziţia este găsită în tabelul de pagini; după reîncărcare, datele se vor găsi în memoria cache
MISS	MISS	HIT	Eşec TLB, poziţia este găsită în tabelul de pagini; după reîncărcare datele vor lipsi din cache
MISS	MISS	MISS	Eşec TLB urmat de o greşeală de pagină; după reîncărcare datele trebuie să lipsească din memoria cache
MISS	HIT	MISS	Imposibil: TLB-ul nu poate conţine o translatare dacă pagina nu se găseşte în memorie
HIT	HIT	MISS	Imposibil:
HIT	MISS	MISS	Imposibil

Exemplul prezentat presupune că adresele de memorie sunt translatate în adrese fizice înainte ca memoria cache să fie accesată => memoria cache este indexată şi marcată fizic.

Timpul de acces la memorie (în cazul HIT) = timpul de acces TLB + timp acces memoria cache – accesele pot fi executate în pipe.

O alternativă la această soluţie este aceea ca UCP-ul să indexeze memoria cache cu o adresă parţial/complet virtuală – **MEMORIE CACHE ADRESATĂ VIRTUAL.**

În astfel de implementări, memoria cache este indexată şi marcată virtual.

Hardware-ul pentru translatarea adresei (exp. TLB-ul) nu este folosit în timpul accesului normal la memoria cache De ce ?

În cazul unor astfel de implementări, când paginile sunt folosite în comun de către programe, există posibilitatea de suprapunere - **ALIASING**

Protecţia folosind memoria virtuală

Funcţia cea mai importantă a unei memorii virtuale – ***folosirea de către mai multe procese în comun a memoriei principale***.

Mecanismul de protecţie trebuie să asigure:
• un proces nu poate să scrie în spaţiul de adrese al altui proces sau în sistemul de operare
• nu trebuie să fie posibilă situaţia în care un proces citeşte datele altui proces

Pentru asigurarea acestor cerinţe hardware-ul trebuie să ofere:

• trebuie să suporte cel puţin două moduri ce indică dacă procesul executat este un proces al utilizatorului sau un proces al sistemului de operare (kernel process sau supervisor process)

• trebuie să furnizeze o porţiune a stării UCP-ului pe care să o poată citi, dar nu şi scrie un proces al utilizatorului – sistemul de operare foloseşte instrucţiuni speciale oferite doar în modul supervizor

• trebuie să ofere mecanisme prin care UCP poate trece din modul utilizator în cel supervizor şi invers

Tratarea greşelilor de pagină şi a eşecurilor în TLB

Un eşec TLB apare atunci când nici o poziţie în TLB nu corespunde unei anumite adrese virtuale. Putem avea următoarele posibilităţi:

1. Pagina este prezentă în memorie şi trebuie creată poziţia lipsă din TLB
2. Pagina nu este prezentă în memorie şi trebuie ca SO-ul să să prelucreze această situaţie

   Cum stabilim care situaţie este situaţia curentă ?

Un eşec TLB poate fi tratat software sau hardware – va trebui o secvenţă scurtă de operaţiuni pentru a copia o poziţie validă din tabelul de pagini în TLB.

Excepţia cauzată de greşeala de pagină trebuie declarată până la sfârşitul aceluiaşi ciclu de ceas în care se face accesul la memorie.

CONCLUZII

Memoria virtuală reprezintă numele unui nivel din ierarhia de memorie ce administrează caching-ul dintre memoria principală şi unitatea de disc.

Permite unui singur program să îşi extindă spaţiul de adrese în afara limitelor memoriei principale.

Memoria virtuală permite folosirea în comun a memoriei principale de către mai multe procese simultan active oferind mecanisme de protecţie a memoriei.

Tehnicile existente pentru administrarea ierarhiei de memorie între memoria principală şi unitatea de disc sunt:

1. Folosirea de blocuri de date de dimensiuni mari (pagini) pentru folosirea principiului localizării spaţiale
2. Corespondenţa dintre adresele virtuale şi cele fizice – implementată printr-un tabel de pagini – este total asociativă => o pagină virtuală poate fi plasată oriunde în memoria principală.
3. SO-ul foloseşte LRU sau bitul de referinţă pentru alegerea paginii care trebuie înlocuită

Mecanismul memoriei virtuale oferă translatarea din adrese virtuale în adrese fizice. Acestă translatare permite folosirea în comun – în mod partajat – a memoriei principale.

TLB-ul acţionează ca o memorie cache pentru translatările din tabelul de pagini. Fiecare adresă este translatată dintr-una virtuală în una fizică folosind translatările din TLB.

UNDE POATE FI AMPLASAT UN BLOC DE DATE ?

Amplasarea blocului la nivelul superior al ierarhiei. Avantajul creşterii gradului de asociativitate este acela al scăderii ratei de eşec.

Performanţa creşte prea puţin în cazul măririi dimensiunii memoriei cache deoarece rata de eşec globală a unei memorii cache de dimensiuni mari este mai redusă.

Dezavantajul schemelor cu asociativitate este dat de costurile crescute şi de timpul de acces mai mare.

CUM SE POATE REGĂSI UN BLOC DE DATE ?

Implementarea unui grad înalt de asociativitate în memoriile cache nu este o soluţie optimă datorită costurilor comparatoarelor care cresc în timp ce rata îmbunătăţirile ratei de eşec sunt mici.

În sistemele de memorie virtuală se păstrează un tabel separat de corespondenţă pentru indexarea memoriei. Alegerea asociativităţii totale şi a acestui tabel sunt justificate de 4 factori. Care sunt aceeia ?

Schemele de amplasare cu asociativitate parţială sunt folosite pentru memoriile cache şi TLB-uri, unde accesul combină indexarea şi căutarea într-un set redus de locaţii.

CE BLOC DE DATE TREBUIE ÎNLOCUIT ÎN CAZ DE MISS PENTRU MEMORIA CACHE ?

În cazul în care memoria este cu asociativitate totală, toate blocurile pot fi înlocuite. Dacă memoria cache este parţial asociativă, trebuie ales între blocurile din set. Dacă avem memorie cu corespondenţă directă, atunci avem o singură posibilitate de înlocuire.

Strategiile fundamentale de înlocuire sunt: aleatorie şi LRU. LRU nu se implementează în ierarhii cu mai mult de 2-4 grade de asociativitate. Pentru restul de ierarhii LRU este doar aproximat.

În memoriile cache algoritmul de înlocuire este codificat în hardware. Odată cu creşterea dimensiunii memoriei cache rata de eşec pentru ambele startegii de înlocuire scade şi diferenţa devine foarte mică.

CE SE ÎNTÂMPLĂ ÎN CAZUL SCRIERILOR ?

Există 2 metode de bază:

Scrierea simultană – informaţia este scrisă atât în blocul de date din memoria cache cât şi în blocul de date din nivelul inferior al ierarhiei.

Scrierea la loc – informaţia este scrisă doar în blocul din memoria cache. Blocul modificat este scris la nivelul ierarhic inferior doar atunci când este înlocuit.

AVANTAJE pentru scrierea la loc:

1. Cuvintele individuale pot fi scrise de procesor la viteza cu care sunt acceptate de către memoria cache şi nu de către memoria principală.

2. Scrierile multiple în interiorul blocului necesită o singură scriere în nivelul inferior al ierarhiei de memorie

3. Pentru scrierea blocurilor la nivelul inferior se foloseşte un transfer de bandă mare.

AVANTAJE pentru scrierea simultană:

1. **Eşecurile sunt mai simple şi au un cost mai scăzut, deoarece nu necesită scrierea unui bloc la loc în nivelul inferior de memorie**

2. Este un sistem mai uşor de implementat în hardware