Parametri de performanţă (1)

- H - rata de succes (*hit ratio*)
  - procentajul cazurilor în care locaţia căutată a fost găsită în cache
- M - rata de insucces (*miss ratio*)
  - procentajul cazurilor în care locaţia căutată nu a fost găsită în cache

$$0 \leq H, M \leq 1$$

$$M = 1 - H$$

Parametri de performanţă (2)

- T_c - timpul de acces la cache
- T_m - timpul de acces la memorie în cazul unei ratări în cache
- T - timpul mediu de acces la memorie (cu cache)
- T_p - timpul de acces la memoria principală (în absenţa cache-ului)

Performanţa memoriei cache (1)

$T = T_c \cdot H + T_m \cdot M$

- dacă $T < T_p \rightarrow$ spor de viteză
- T - mărime statistică
- cazuri extreme
  - H=100% (M=0): $T = T_c \rightarrow$ ideal
  - H=0 (M=100%): $T = T_m \rightarrow$ pierdere de viteză ($T_m > T_p$)

Performanţa memoriei cache (2)

Situaţia reală - exemplu

- T_c = 2 ns
- T_p = 10 ns
- T_m = 11 ns
- H = 95%
- T = 2.45 ns = 0.245 T_p $\rightarrow$ viteza de acces creşte de peste 4 ori

Adresare

- adresa din cache nu corespunde cu adresa din memoria principală
- căutarea se face după adresa din memoria principală
- deci cache-ul trebuie să reţină şi adresele locaţiilor în memoria principală

Linii de cache

- cache-ul se foloseşte de localizarea temporală
- cum se poate exploata şi localizarea spaţială?
- când se aduce o locaţie în cache, se aduc şi locaţiile vecine
  - acestea formează o *linie de cache*

Politica de înlocuire

- cache mic - se umple repede
- noi linii aduse în cache - trebuie eliminate altele mai vechi
  - eliminare - scriere în memoria principală
- care linii trebuie eliminate?
  - scopul - creşterea vitezei
- cele care nu vor fi accesate în viitorul apropiat !!!

Îmbunătăţirea performanţei

- depinde de două mărimi
  - timpul de acces la cache (T_c)
  - rata de succes (H)
- nu pot fi optimizate simultan
- influenţate de
  - tehnologie
  - politica de înlocuire

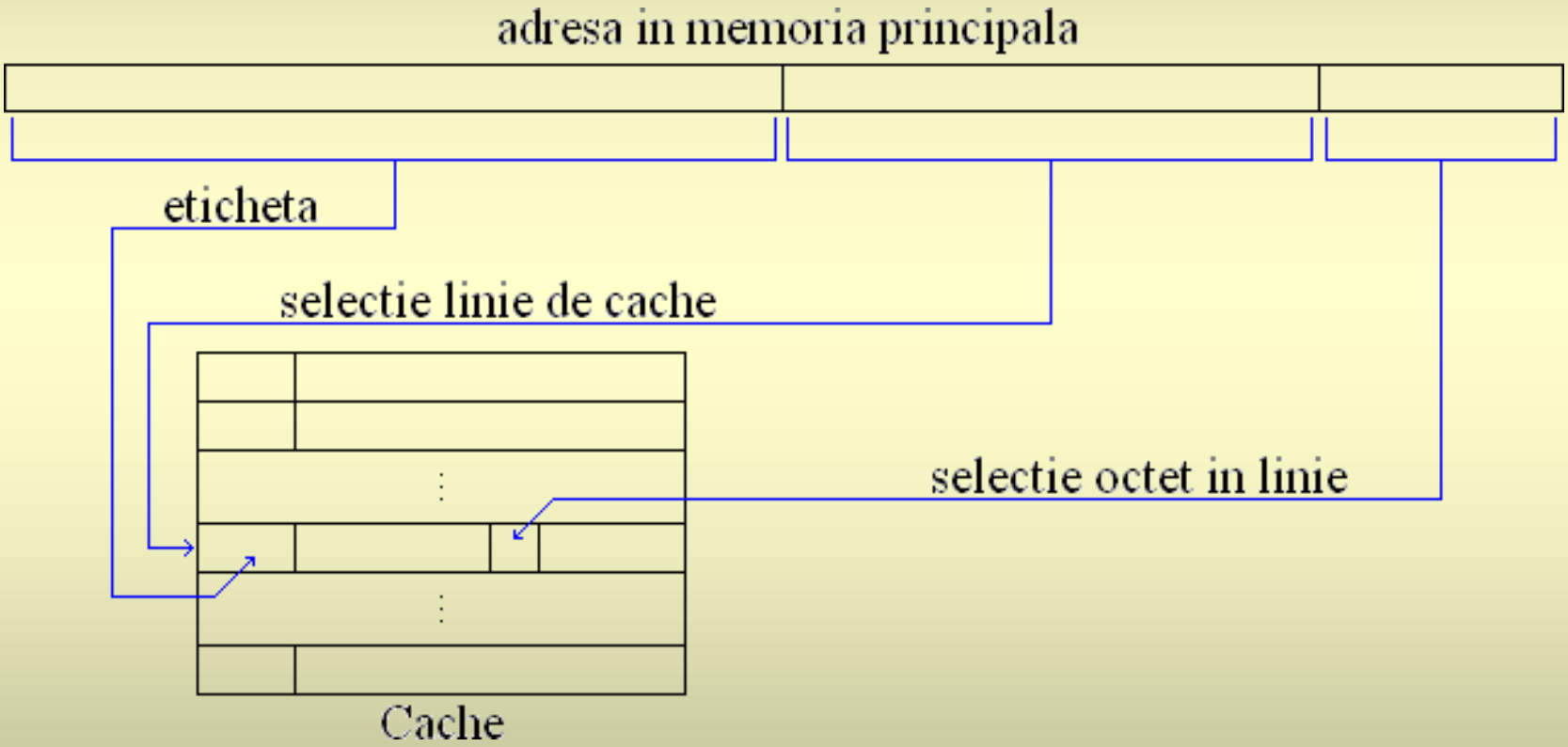
Tipuri constructive de cache

- cu adresare directă (*direct mapped cache*)
- total asociativ (*fully associative cache*)
- parţial asociativ (*set associative cache*)

Cache cu adresare directă (1)

- plasarea unei locaţii în cache
  - linia de cache este întotdeauna aceeaşi
  - depinde de adresa din memoria principală
- adresa din memoria principală - 3 părţi
  - eticheta - se memorează în cache
  - selectorul liniei de cache
  - selectorul octetului în cadrul liniei

Cache cu adresare directă (2)



Cache cu adresare directă (3)

Exemplu

- adresa în memoria principală - 32 biţi
- dimensiune cache: 2^{11} linii × 2^5 octeţi/linie
- adresa în memoria principală se împarte în
  - selectorul liniei de cache - 11 biţi
  - selectorul octetului în cadrul liniei - 5 biţi
  - eticheta - 16 biţi (= 32 – 11 – 5)

Cache cu adresare directă (4)

Exemplu (continuare)

- adresa în memoria principală: $45097373_{(10)}$

$00000010101100000010000110011101_{(2)}$

- eticheta: $0000001010110000_{(2)}=688_{(10)}$
- linia de cache: $00100001100_{(2)}=268_{(10)}$
- octetul în cadrul liniei: $11101_{(2)}=29_{(10)}$

Cache cu adresare directă (5)

Exemplu (continuare)

- Ce adrese din memoria principală sunt aduse în linia de cache?

$00000010101100000010000 1100xxxxx_{(2)}$

$00000010101100000010000 1100 00000 \div$

$00000010101100000010000 1100 11111_{(2)}$

$45097344 \div 45097375_{(10)}$

Cache cu adresare directă (6)

Conţinutul unei linii de cache

- un bit care indică dacă linia conţine date valide
  - iniţial, toate liniile sunt goale, deci invalide
- câmpul etichetă
- datele propriu-zise, aduse din memoria principală

Cache cu adresare directă (7)

Avantaje

- implementare simplă
- timp de acces (T_c) redus

Dezavantaje

- lipsă de flexibilitate
- politică de înlocuire neperformantă - rată de succes (H) scăzută

Cache cu adresare directă (8)

Exemplu

```
for(i=0;i<1000;i++) a=a+i;
```

- adrese: `i` → 3806240, `a` → 1756566572
- ambele sunt memorate în cache în linia 161
- accese alternative → înlocuiri dese în cache → număr mare de ratări

Cache total asociativ (1)

- realizat cu memorii asociative
  - memoria obişnuită - acces la o locaţie pe baza adresei sale
  - memoria asociativă - permite şi regăsirea locaţiei pe baza conţinutului său
  - implementare - valoarea căutată este comparată în paralel cu toate locaţiile
    - de ce în paralel?

Cache total asociativ (2)

Avantaje

- plasarea datelor din memoria principală - în orice linie de cache
- se pot alege convenabil adresele aduse în linia de cache
- se pot implementa politici de înlocuire eficiente - rată de succes (H) ridicată

Cache total asociativ (3)

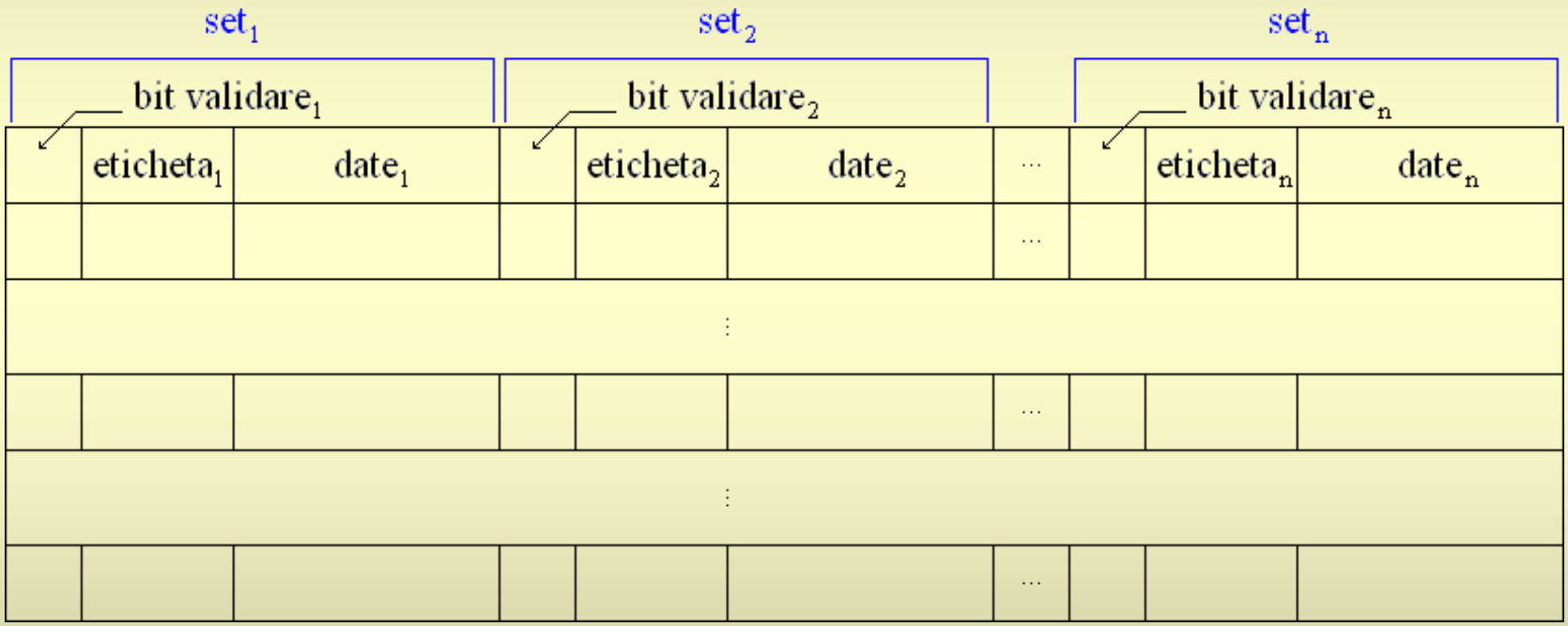
Dezavantaje

- timp de acces (T_c) mare
  - memoriile asociative - lente
  - algoritmi complecşi de înlocuire - timp suplimentar consumat
- hardware complicat pentru memoriile asociative şi algoritmii de înlocuire

Cache parţial asociativ (1)

- numit şi cache asociativ pe seturi
- derivat din cache-ul cu adresare directă
- fiecare linie de cache conţine mai multe seturi de date (4, 8, 16, ...)
- structura unui set
  - bit de validare
  - etichetă
  - date din memoria principală

Cache parţial asociativ (2)



Cache parţial asociativ (3)

Timpul de acces (T_c)

- puţin mai mare decât la cache-ul cu adresare directă
  - trebuie verificate toate cele *n* seturi

Rata de succes (H)

- ridicată
  - elimină problema suprapunerilor

Scrierea în cache (1)

- scriere într-o locaţie care nu se află în cache
- unde se face scrierea?
- variante
  - doar în memoria principală - nu se poate
    - de ce?
  - doar în cache (*write-back*)
  - atât în cache, cât şi în memoria principală (*write-through*)

Scrierea în cache (2)

Cache de tip *write-back*

- scrierea se face doar în cache
- datele ajung în memoria principală doar la evacuarea din cache
- viteză mare
- probleme în sistemele multiprocesor

Scrierea în cache (3)

Cache de tip *write-through*

- scrierea se face atât în cache, cât şi în memoria principală
- mai lent
  - datorită accesului la memoria principală
- ambele tipuri de cache sunt larg folosite

Conceptul de cache - extindere

- aplicabil nu doar la procesoare
- tipul de problemă: comunicarea cu o entitate lentă, de mare capacitate
- soluţia: interpunerea unei entităţi cu capacitate mai mică şi viteză mai mare
  - reţine ultimele date vehiculate

Unde mai putem folosi ideea?

Aplicabilitate

- oriunde funcţionează legile localizării
- hardware
- software

Exemple (1)

Cache-uri de disc

- 2 direcţii de aplicare
  - hardware - circuit de memorie integrat în controller
  - software - o zonă din memoria sistem
- entitatea mai mare şi lentă - discul
- entitatea mai mică şi rapidă - memoria

Exemple (2)

Browserul web

- ultimele pagini accesate sunt reţinute pe disc
  - numai localizare temporală - de ce?
- entitatea mai mare şi lentă - reţeaua (Internet)
- entitatea mai mică şi rapidă - discul