

Exploatarea ierarhiei de memorie

Un exemplu intuitiv

Un student aflat în biblioteca facultății are pe masă o serie de cărți

Subiectul căutat nu se regăsește în cărțile aflate pe masă

Studentul se reîntoarce la rafturi și extrage o nouă carte

Existența unui număr mare de cărți pe masă reduce timpul de căutare

Probabilitatea de căutare nu este aceeași pentru toate cărțile din bibliotecă

Un program nu accesează toată secțiunea sa de cod sau de date cu aceeași probabilitate.

Principiul localizării - stă la baza modului de operare a programelor

stabilește faptul că programele accesează o porțiune relativ redusă a spațiului lor de adrese la orice moment de timp

Localizarea temporală - localizare în timp - dacă se face referire la un anumit obiect, este posibil ca acesta să fie referit din nou cât de curând

Localizarea spațială - localizare în spațiu - dacă se face referire la un anumit obiect din memorie, obiectele ale căror adrese sunt învecinate cu acesta, tind să fie adresate cât de curând.

Principiul localizării temporare este folosit la implementarea memoriei unui calculator sub forma unei ierarhii de memorie.

O ierarhie de memorii poate fi alcătuită din mai multe niveluri, însă datele la un moment dat sunt copiate doar între două niveluri adiacente.

Unitatea minimă de informație care poate fi prezentă sau absentă într-o ierarhie de memorie se numește bloc.

Unitatea minimă de informație care poate fi prezentă sau absentă într-o ierarhie de memorie se numește bloc.

Regăsirea informației necesare procesorului într-un bloc de memorie superior se numește **HIT**

Neregăsirea datelor pe nivelul superior se numește **MISS**

Rata de succes - fracțiunea din accesele la memorie ce au găsit datele în nivelul superior de memorie.

Rata de eșec = 1 - rata de succes fracțiunea din accesele la memorie care nu au găsit datele în nivelul superior de memorie.

Timpul de succes - reprezintă timpul necesar accesului la un nivel superior al ierarhiei de memorie, ce include și timpul necesar determinării tipului de acces.

Penalizarea de eșec - reprezintă timpul necesar înlocuirii blocului din nivelul superior cu blocul corespunzător din nivelul inferior, incluzând și timpul trimiterii acestui bloc către procesor.

Construirea sistemelor de memorie afectează:

1. modul în care SO-ul administrează memoria și perifericele
2. modul în care compilatoarele generează codul
3. modul în care aplicațiile folosesc mașina de calcul

Principiu de bază

Programele prezintă localizare temporală cât și localizare spațială.

Ierarhiile de memorie folosesc avantajul localizării temporale păstrând datele accesate recent cât mai aproape de procesor.

Ierarhiile de memorie folosesc avantajul localizării spațiale prin mutarea blocurilor conținând cuvinte învecinate din memorie în nivelurile superioare ale ierarhiei.

În anii '60 s-a folosit cuvântul **cache** pentru a desemna nivelul ierarhiei de memorie aflat între UCP și memoria principală.

PRINCIPIILE DE BAZĂ ALE MEMORIILOR CACHE

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_3

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_n
X_3

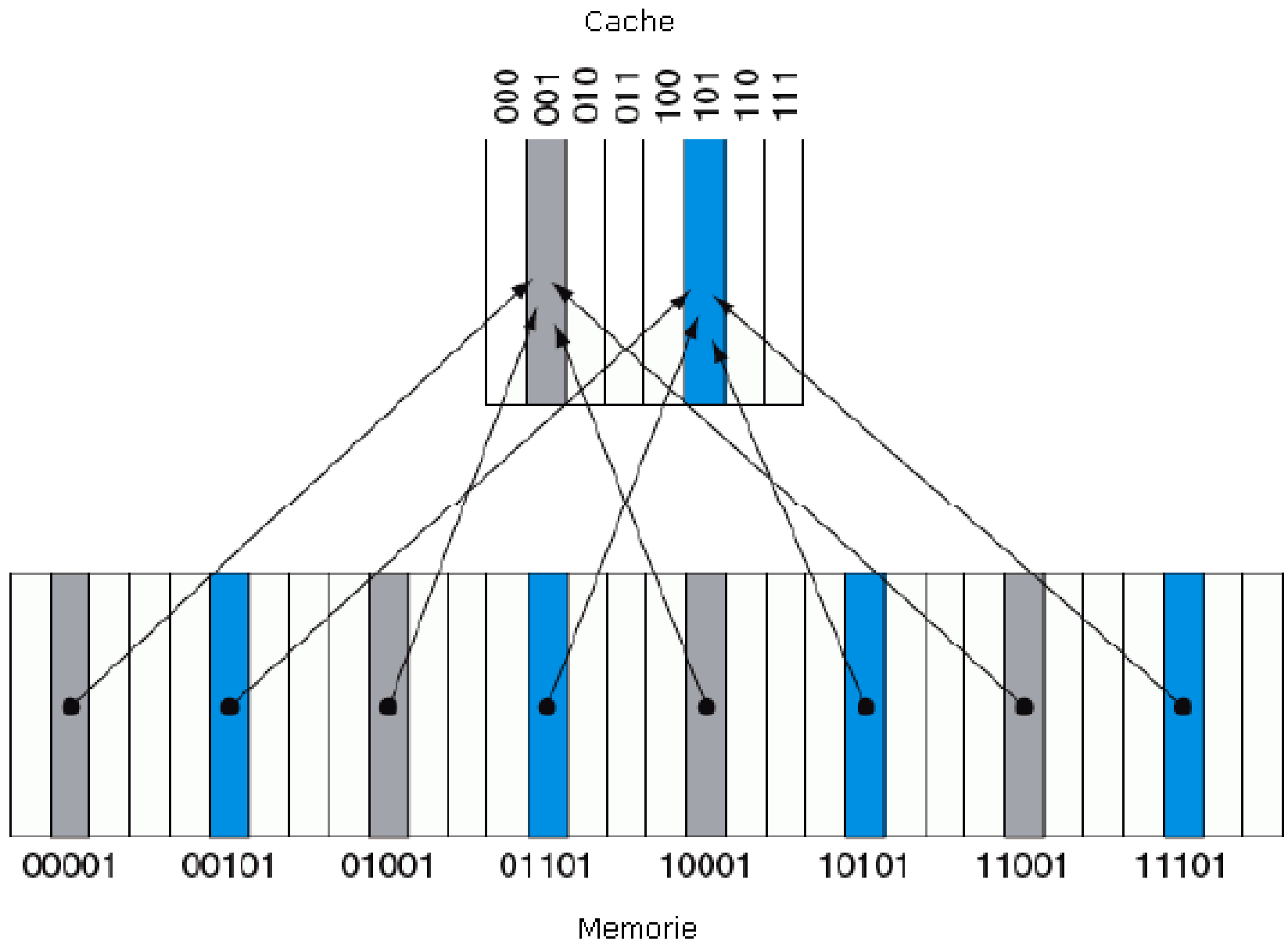
Inițial cuvântul de date X_n
nu se găsește în cache

Cum se poate determina dacă o
dată este în memoria cache ?

Dacă o dată există în memoria
cache cum poate fi ea găsită ?

!!!! Fiecare cuvânt poate fi memorat doar într-o anumită locație a memoriei cache =>
corespondență directă.

Corespondența dintre adrese și locațiile memoriei cache se determină astfel:
(Adresa blocului) modulo (numărul de blocuri din memoria cache)



Cum determinăm dacă o dată este validă sau nu ?

Metoda cea mai des folosită este cea de a aduna un **bit de validare** pentru a indica dacă o locație conține o adresă validă.

Accesarea unei memorii cache cu corespundeță directă

Cerere	Adresa	HIT / MISS	Blocul din cache
22	10110		
26	11010		
22	10110		
26	11010		
16	10000		
3	00011		
16	10000		
18	10010		

Index	V	Marcaj	Date
0	N		
1	N		
10	N		
11	N		
100	N		
101	N		
110	N		
111	N		

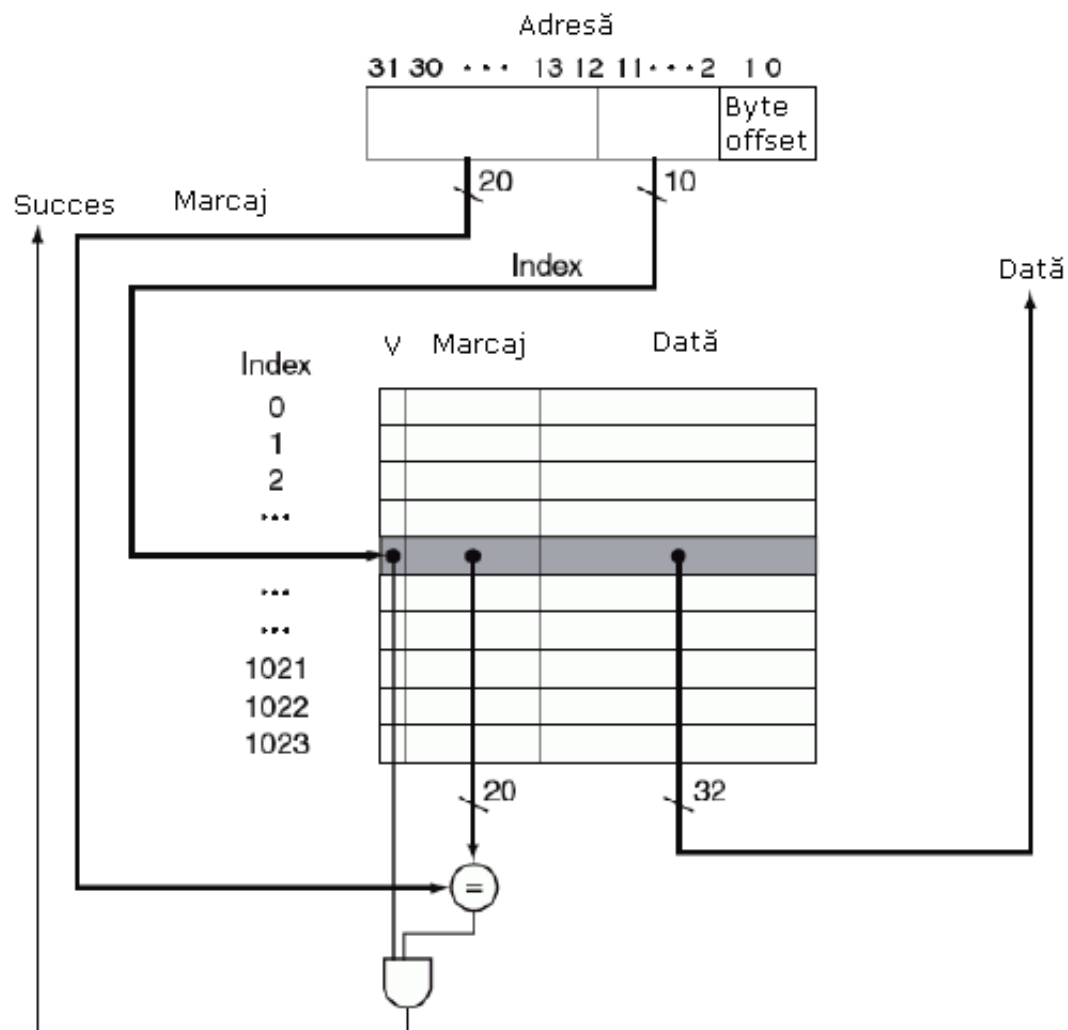
Index	V	Marcaj	Date
0	N		
1	N		
10	N		
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

Index	V	Marcaj	Date
0	N		
1	N		
10	Y	11	mem (11010)
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

Index	V	Marcaj	Date
0	Y	10	mem (10000)
1	N		
10	Y	11	mem (11010)
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

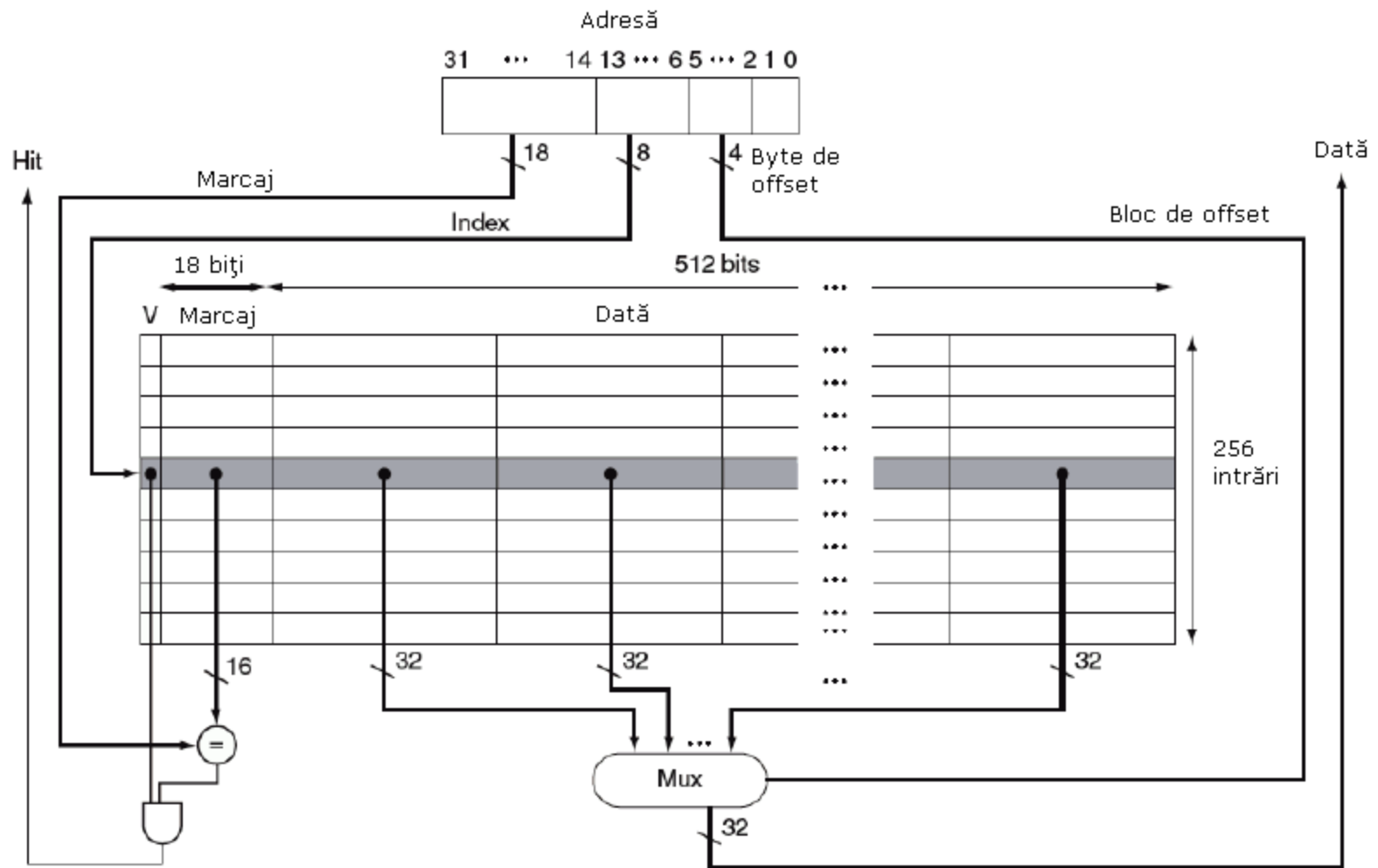
Acest tip de accesare permite folosirea principiului localizării temporale - cuvintele accesate recent le înlocuiesc pe cele accesate mai puțin recent.

Tag = Marcaj



1. indexul memoriei cache, folosit la selectarea blocului de memorie cache
2. câmpul marcajului, folosit la compararea cu valoarea din câmpul marcaj al memoriei cache

FOLOSIREA LOCALIZĂRII SPAȚIALE



Se dorește ca blocul memoriei cache să fie mai mare decât lungimea unui cuvânt

Până acum am folosit doar schema de amplasare a blocurilor din memoria cache denumită **corespondență directă**.

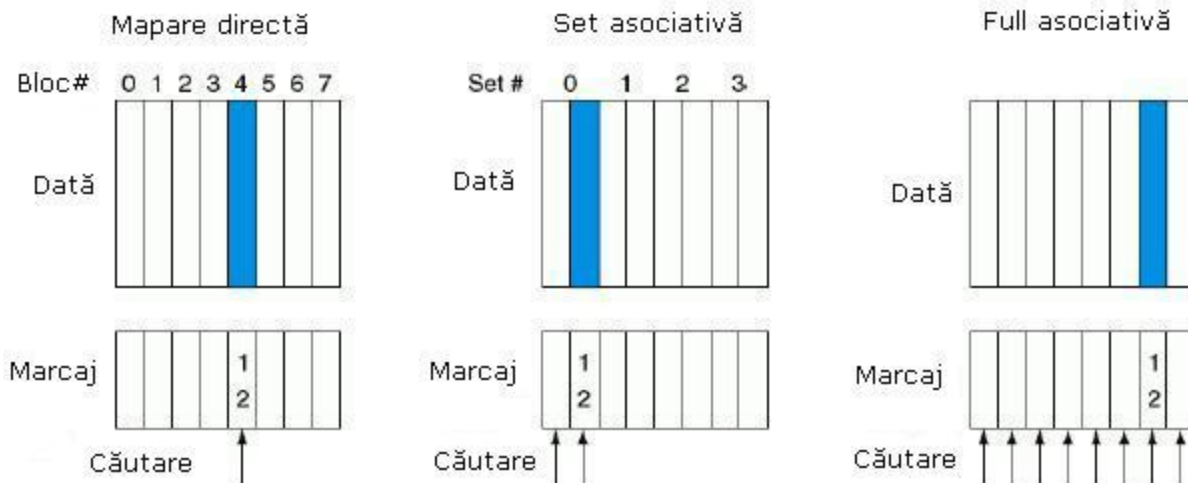
Schema în care un bloc de date poate fi amplasat în orice locație din memoria cache se numește **schemă cu asociativitate totală**.

Regăsirea blocului de date presupune examinarea tuturor locațiilor de memorie cache
=> paralelizarea căutării

O altă schemă care este între cele două se numește **schema cu asociativitate parțială**.

În aceste tipuri de scheme memoria cache are un număr fix de locații în care se poate amplasa fiecare bloc de date. Deci vom avea un număr de seturi fiecare format din n blocuri de date.

Pentru regăsirea blocului este necesar să parcurgem toate blocurile unui set.



Într-o memorie cache cu asociativitate parțială, setul conținând un anumit bloc de memorie este dat de relația:

$(\text{numărul blocului}) \bmod (\text{numărul de seturi din memoria cache})$

OBSERVAȚIE : Creșterea gradului de asociativitate reduce rata de eșec, dar crește timpul de HIT.

EXEMPLU

Avem 3 memorii cache fiecare având 4 blocuri de câte 1 cuvânt. Cele trei memorii cache sunt cu asociativitate totală, asociativitate cu 2 căi și corespondență directă.

Să se găsească numărul de eșecuri pentru fiecare dintre cele 3 scheme de amplasare având următoarea secvență de adrese de bloc: 0,8, 0, 6, 8.

SOLUȚIE

Cazul 1 – memoria cache cu corespondență directă

Detectăm blocul din memoria cache corespunzător adreselor date:

Adresa blocului	Blocul memoriei cache
0	$0 \bmod 4 = 0$
6	$6 \bmod 4 = 2$
8	$8 \bmod 4 = 0$

Conținutul memoriei cache după fiecare referință

Adresa blocului de memorie accesat	HIT sau MISS	Blocul 0	Blocul 1	Blocul 2	Blocul 3
0	Miss	Mem(0)			
8	Miss	Mem(8)			
0	Miss	Mem(0)			
6	Miss	Mem(0)		Mem(6)	
8	Miss	Mem(8)		Mem(6)	

Cazul 2. Memoria cache cu asociativitate parțială cu 2 căi conține 2 seturi (indicii fiind 0 și 1), fiecare având 4 elemente. Vom determina setul corespunzător fiecărei adrese a blocurilor

Adresa blocului	Blocul memoriei cache
0	$0 \text{ modulo } 2 = 0$
6	$6 \text{ modulo } 2 = 0$
8	$8 \text{ modulo } 2 = 0$

Pentru înlocuire vom folosi LRU

Adresa blocului de memorie accesat	HIT sau MISS	Set 0	Set 1	Set 2	Set 3
0	Miss	Mem(0)			
8	Miss	Mem(0)	Mem(8)		
0	HIT	Mem(0)	Mem(8)		
6	Miss	Mem(0)	Mem(6)		
8	Miss	Mem(8)	Mem(6)		

Avem doar 4 eșecuri, deci soluția aceasta este mai bună decât precedenta

Memoria cache cu asociativitate totală

Adresa blocului de memorie accesat	HIT sau MISS	Set 0	Set 1	Set 2	Set 3
0	Miss	Mem(0)			
8	Miss	Mem(0)	Mem(8)		
0	HIT	Mem(0)	Mem(8)		
6	Miss	Mem(0)	Mem(8)	Mem(6)	
8	HIT	Mem(0)	Mem(8)	Mem(6)	

Aceasta este varianta optimă – avem doar 3 eșecuri

Localizarea unui bloc în memoria cache cu asociativitate parțială

