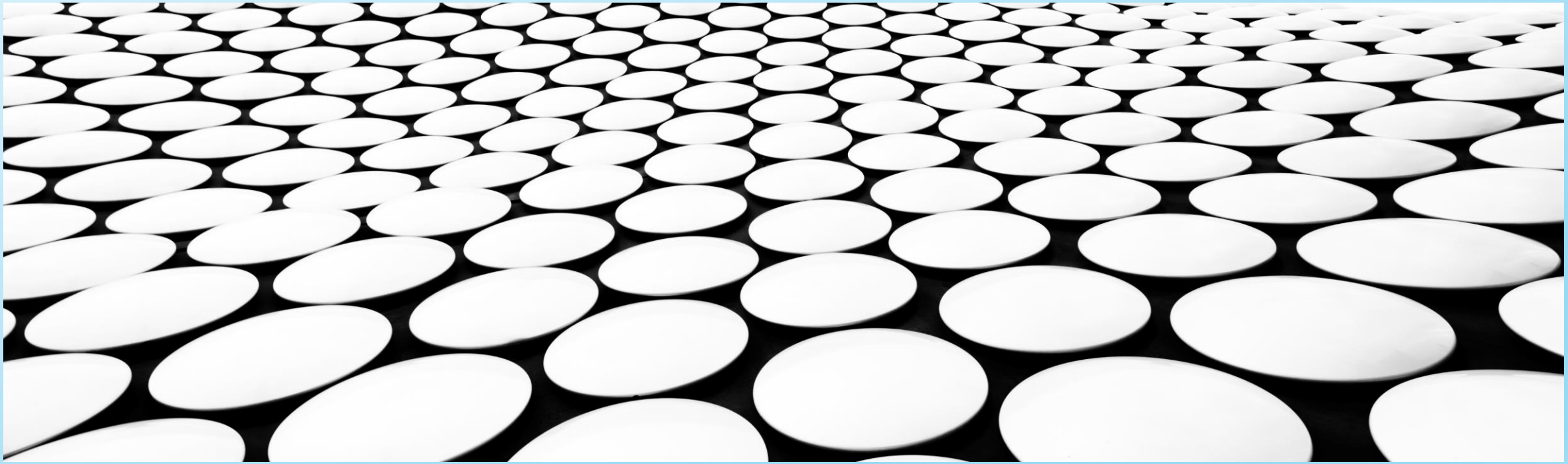
ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



MEMORII CACHE

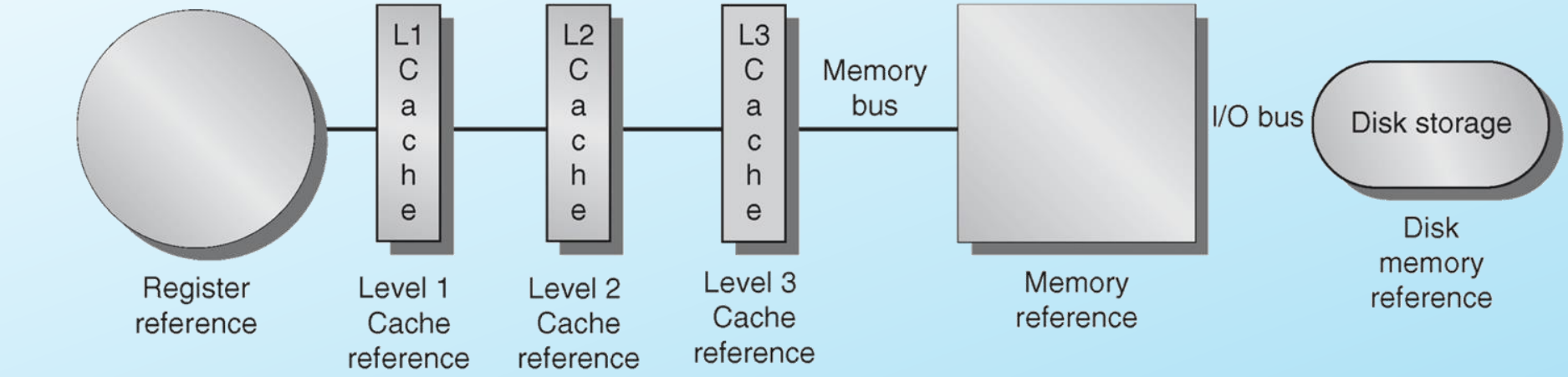
Ce este memoria cache?

Este o memorie, rapida, intermediara, temporara, care stocheaza informatiile mai frecvent utilizate.

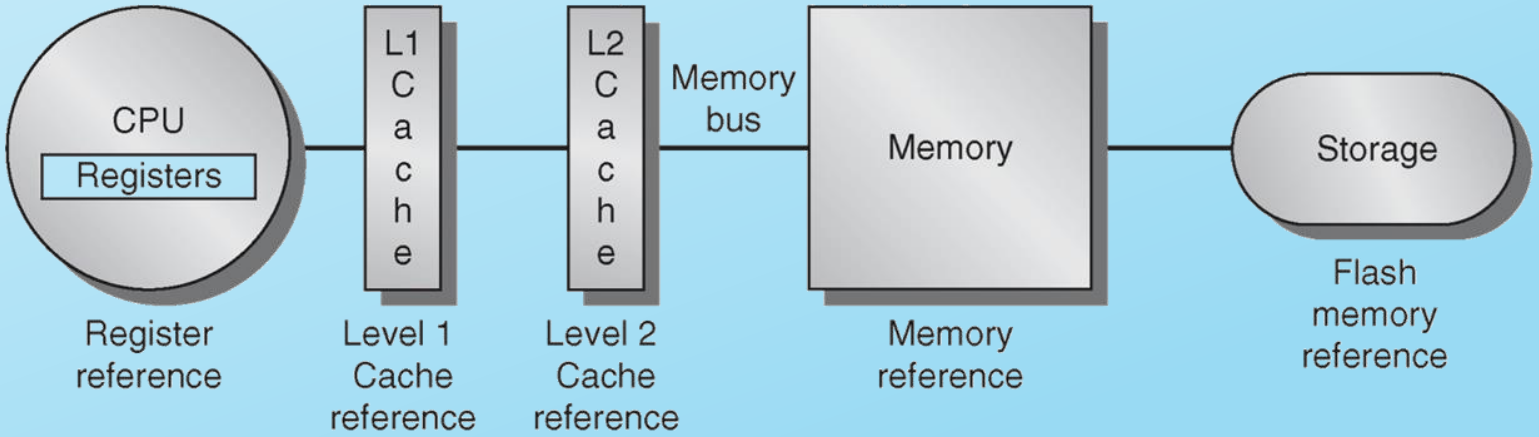
Informatia necesara UC este cautata mai intai in memoria cache si apoi in memoria principala.

Daca exista mai multe memorii chache (ierarhizate) atunci informatia este cautata, mai inatai in memoriile cache, in ordine ierarhica, si apoi daca nu este gasita, in memoria principala

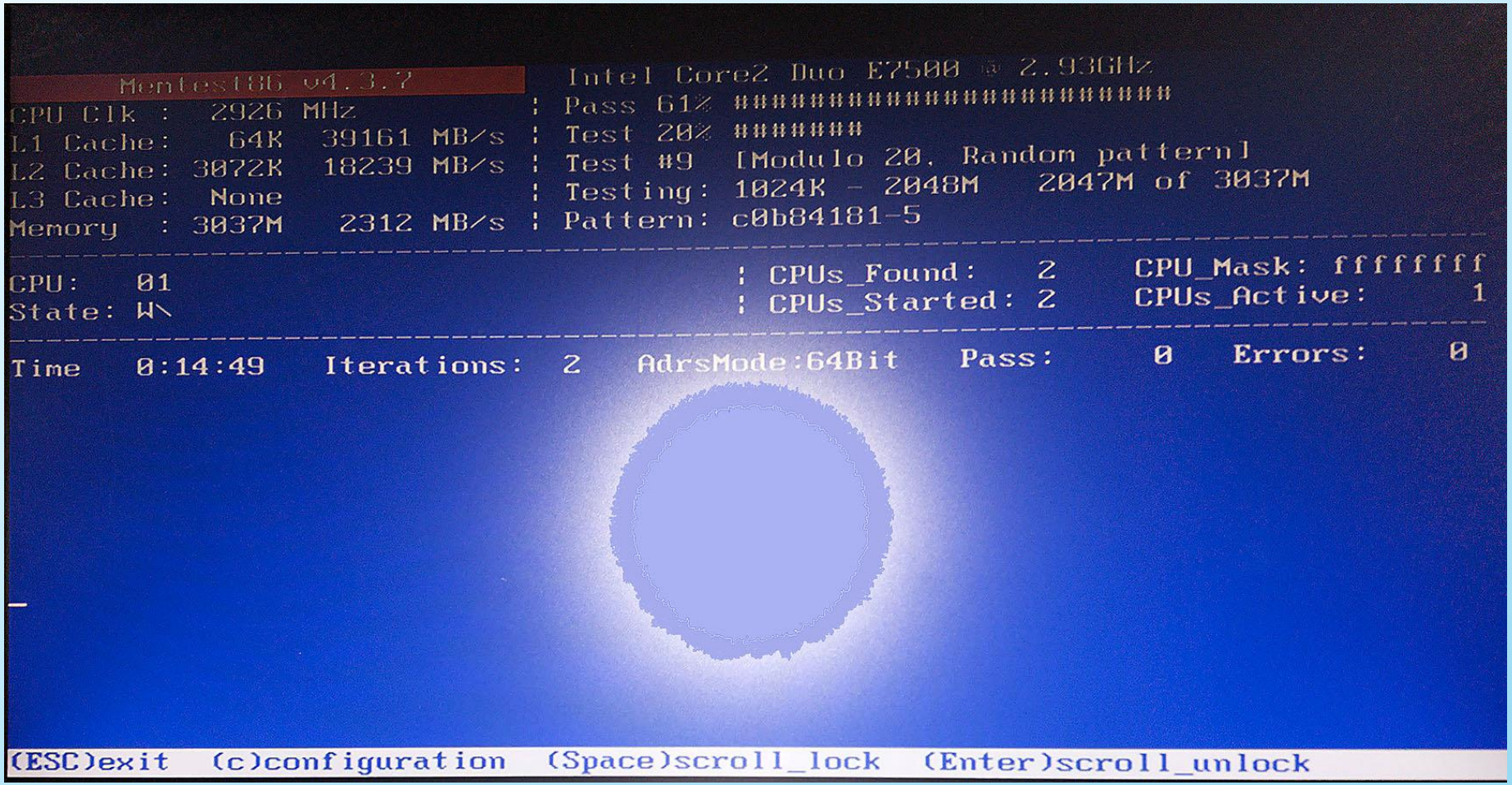
Arhitecturi cu memorii cache



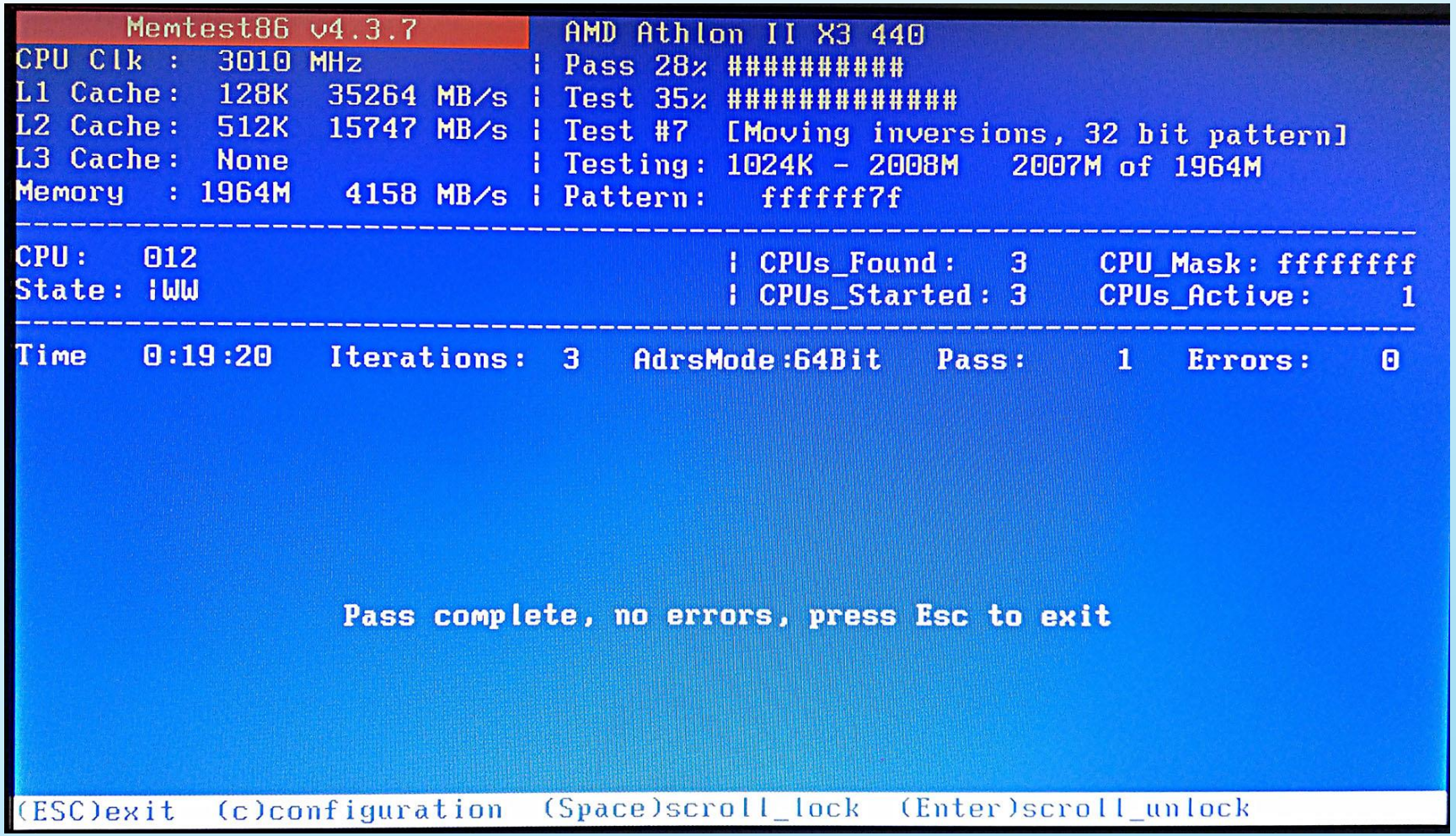
(a) Memory hierarchy for server



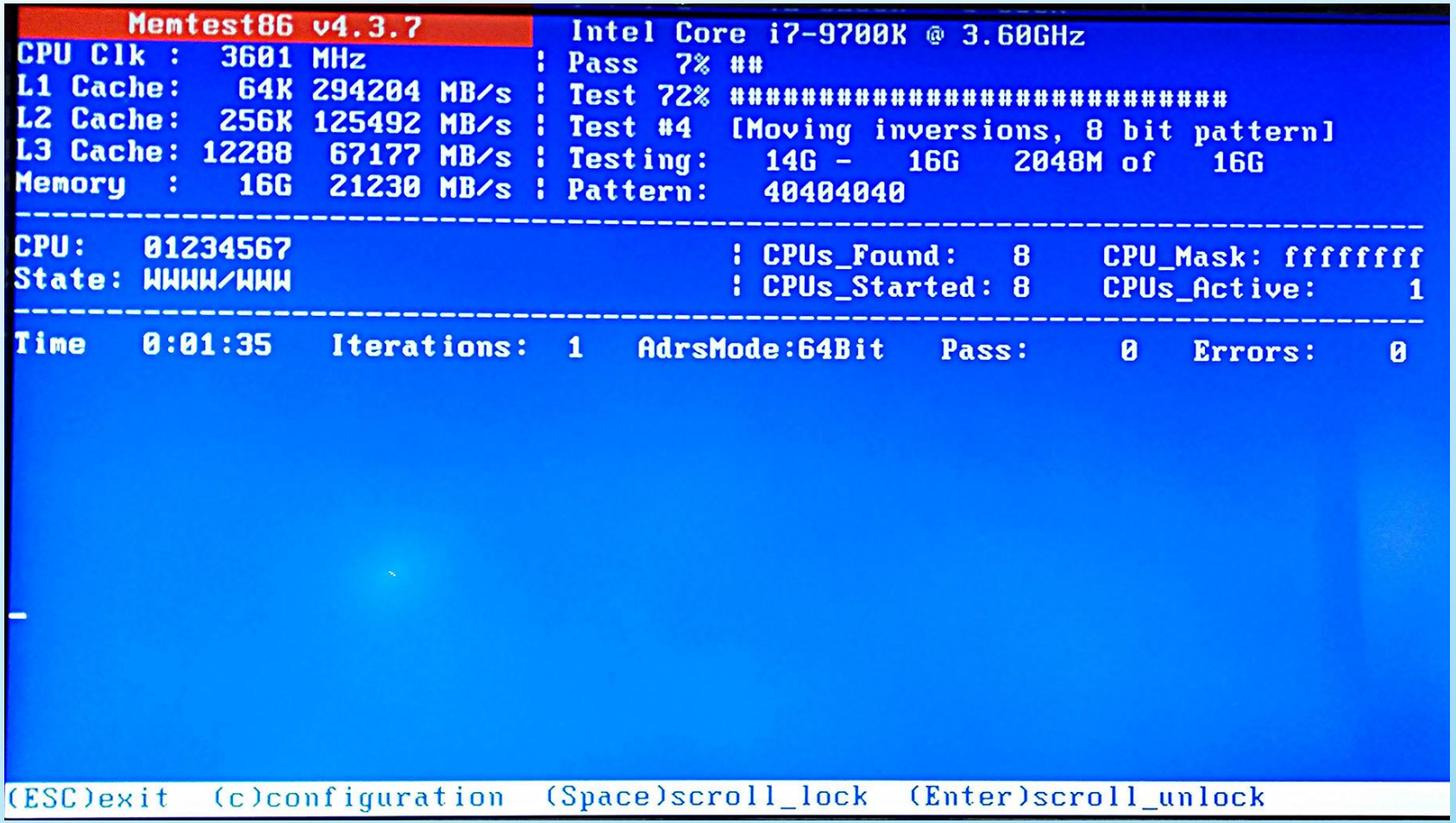
(b) Memory hierarchy for a personal mobile device


Memtest86 v4.3.7
Intel Core2 Duo E7500 @ 2.93GHz
CPU Clk : 2926 MHz
L1 Cache: 64K 39161 MB/s
L2 Cache: 3072K 18239 MB/s
L3 Cache: None
Memory : 3037M 2312 MB/s
Pass 61% ##############################
Test 20% ########
Test #9 [Modulo 20, Random pattern]
Testing: 1024K - 2048M 2047M of 3037M
Pattern: c0b84181-5
CPU: 01
State: W\
CPUs_Found: 2
CPUs_Started: 2
CPU_Mask: ffffffff
CPUs_Active: 1
Time 0:14:49 Iterations: 2 AdrsMode:64Bit Pass: 0 Errors: 0
(ESC)exit (c)configuration (Space)scroll_lock (Enter)scroll_unlock

2007


Memtest86 v4.3.7
AMD Athlon II X3 440
CPU Clk : 3010 MHz
L1 Cache: 128K 35264 MB/s
L2 Cache: 512K 15747 MB/s
L3 Cache: None
Memory : 1964M 4158 MB/s
Pass 28% ##########
Test 35% ##########
Test #7 [Moving inversions, 32 bit pattern]
Testing: 1024K - 2008M 2007M of 1964M
Pattern: ffffff7f
CPU: 012
State: WW
CPUs_Found: 3
CPUs_Started: 3
CPU_Mask: ffffffff
CPUs_Active: 1
Time 0:19:20
Iterations: 3
AdrsMode:64Bit
Pass: 1
Errors: 0
Pass complete, no errors, press Esc to exit
(ESC)exit (c)configuration (Space)scroll_lock (Enter)scroll_unlock

2011


Memtest86 v4.3.7
Intel Core i7-9700K @ 3.60GHz
CPU Clk : 3601 MHz
L1 Cache: 64K 294204 MB/s
L2 Cache: 256K 125492 MB/s
L3 Cache: 12288 67177 MB/s
Memory : 16G 21230 MB/s
Pass 7% ##
Test 72% ##############################
Test #4 [Moving inversions, 8 bit pattern]
Testing: 14G - 16G 2048M of 16G
Pattern: 40404040
CPU: 01234567
State: WWWW/WWW
CPUs_Found: 8
CPUs_Started: 8
CPU_Mask: ffffffff
CPUs_Active: 1
Time 0:01:35
Iterations: 1
AdrsMode:64Bit
Pass: 0
Errors: 0
(ESC)exit (c)configuration (Space)scroll_lock (Enter)scroll_unlock

2019

Cateva definiţii

- cache hit:
  - eveniment de identificare a informației cautate in memoria cache (**succes**)
- cache miss:
  - eveniment cauzat de lipsa informației cautate in memoria cache (**insucces**)
- hit rate:
  - Procentul de accesari **cu** succes a memoriei cache, din totalul accesarilor
- miss rate:
  - Procentul de accesari **fara** succes a memoriei cache, din totalul accesarilor

- Rata tipica de succes (hit rate): >=95%

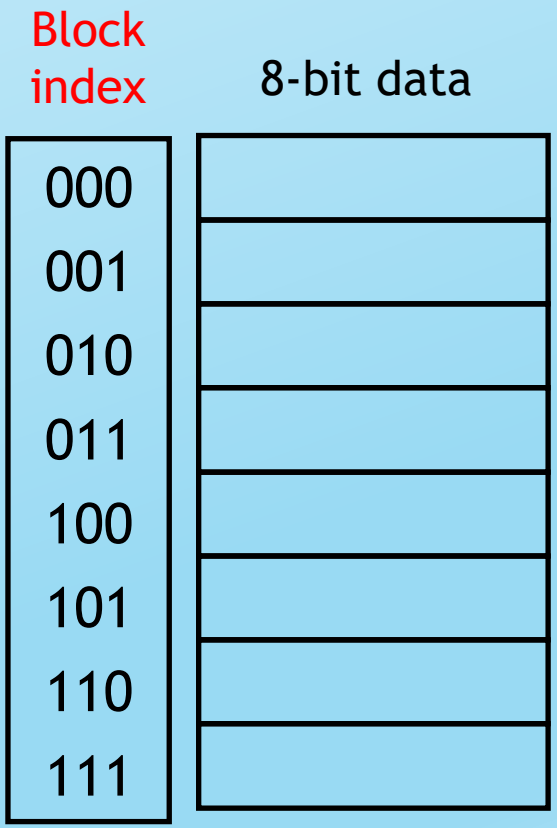
Design simplu de memorie cache

Memoria cache este divizata in blocuri, de dimensiuni egale.

- Numarul de blocuri in memoria cache este, uzual, o putere a lui 2.

- Exemplu simplu
  - fiecare bloc conține un byte.
  - Index cu dimensiunea de 3 biti
  - 8 blocuri

$2^3=8$



Fiecare memorie cache are atasat un controler de memorie.
Ce face controlerul de memorie?

4 intrebari importante:

1. Cand este copiat un bloc de date, din memoria principala in cache, unde este plasat el?

2. Cum se stabileste daca un cuvant (bloc) este deja in cache, sau trebuie adus mai intai din memoria principala?

3. Cache-ul se poate umple.

   Pentru a incarca un nou bloc, trebuie sa inlocuim unul existent!

   Care va fi acela?

4. Cum se desfasoara o operație de scriere in memorie?

TIPURI DE MEMORIE CACHE

Memorie cache cu:

- Mapare directa
- Mapare complet asociativa
- Mapare directa in seturi asociative

Memoria cache cu mapare directa (cea mai simpla structura)

Fiecarei adrese din memoria principala ii corespunde o poziție precisa in memoria cache.

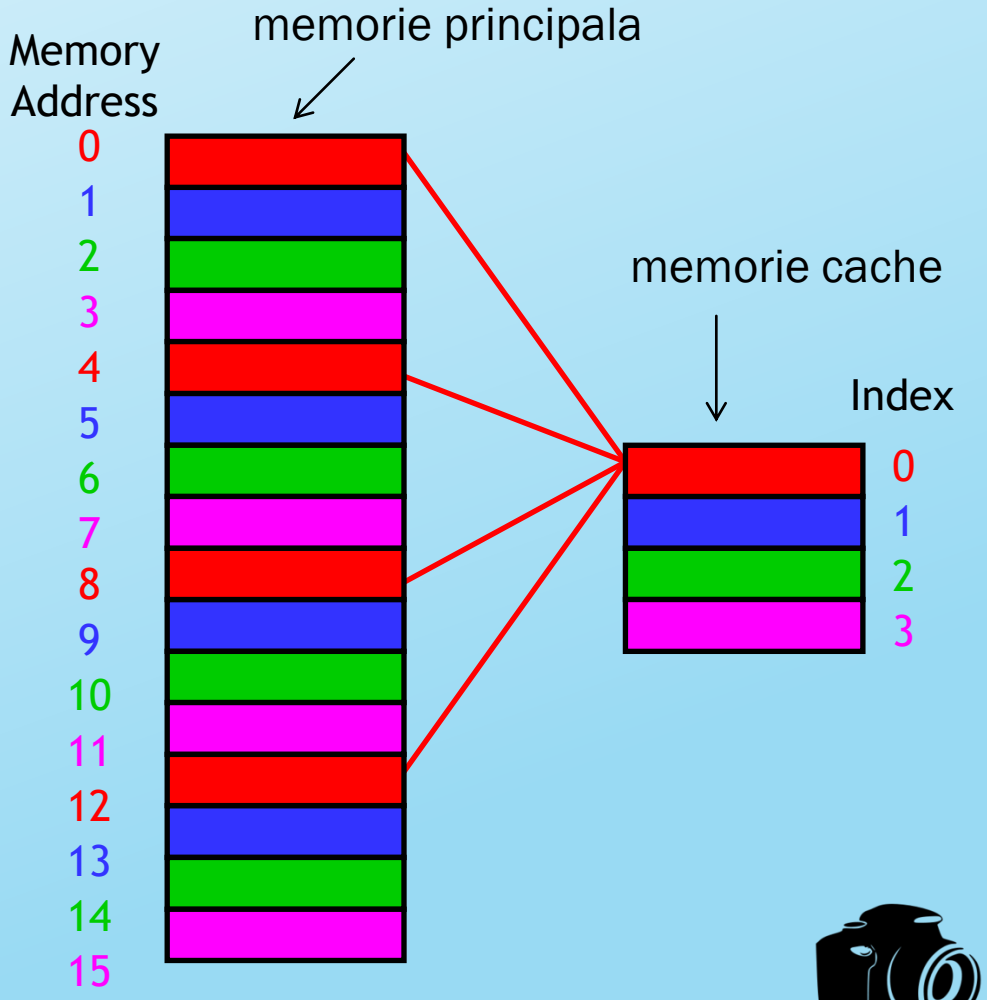
Exemplu:

Memorie principala de 16 byte
Memorie cache de 4 byte

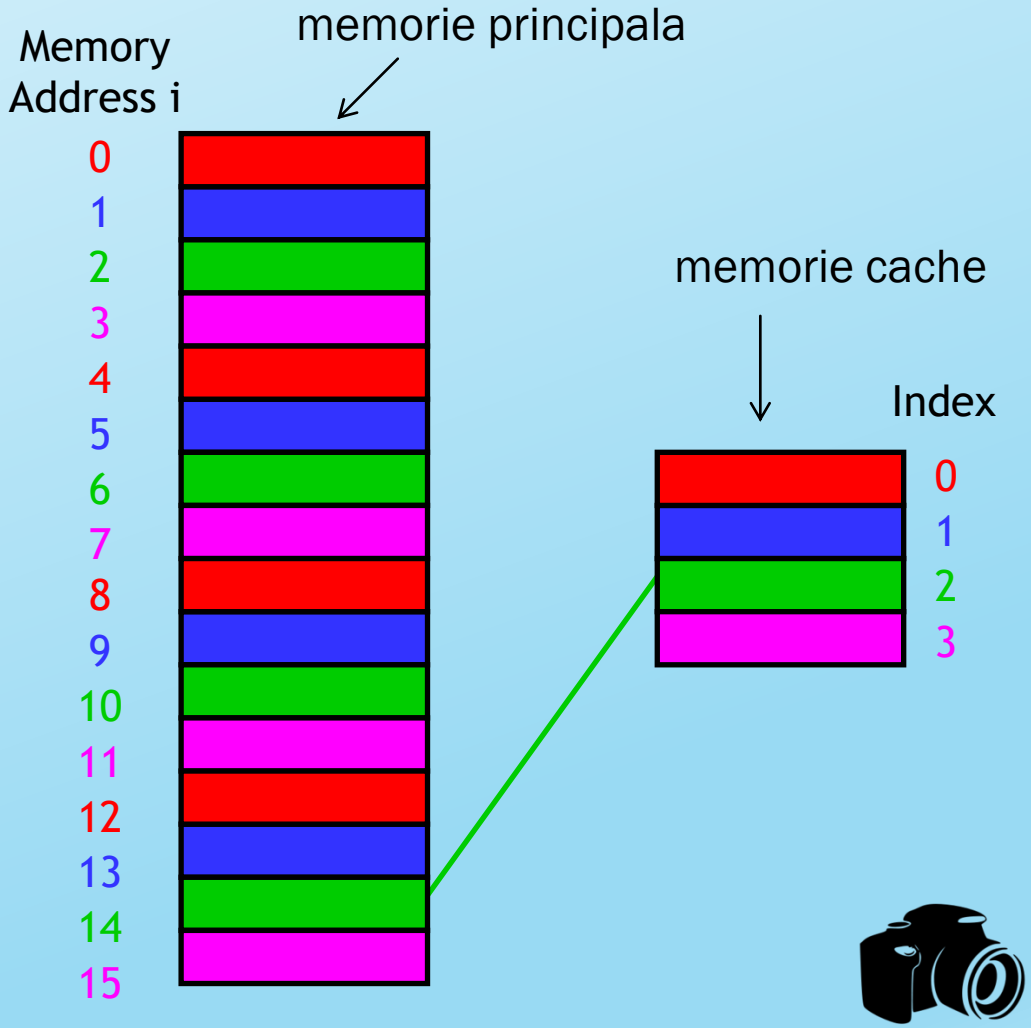
Locațiile de memorie 0, 4, 8 si 12
vor fi mapate in blocul 0

Locațiile de memorie 1, 5, 9 si 13
vor fi mapate in blocul 1

Cum se calculeaza?



- **Se foloseste operatorul de calculare a restului!**
- Daca memoria cache contine 2^k blocuri
  atunci cuvantul aflat in memoria principala la adresa ***i*** va fi copiat in cache in locatia indicata de *index*:
  - *index* = $i \bmod 2^k$
  - $(i \% 2^k)$
- In exemplul nostru
- adresa 14 va fi mapata
- in blocul 2.
- 14 mod 4 = 2



O cale echivalenta: identificarea celor mai putin semnificativi biți din adresa

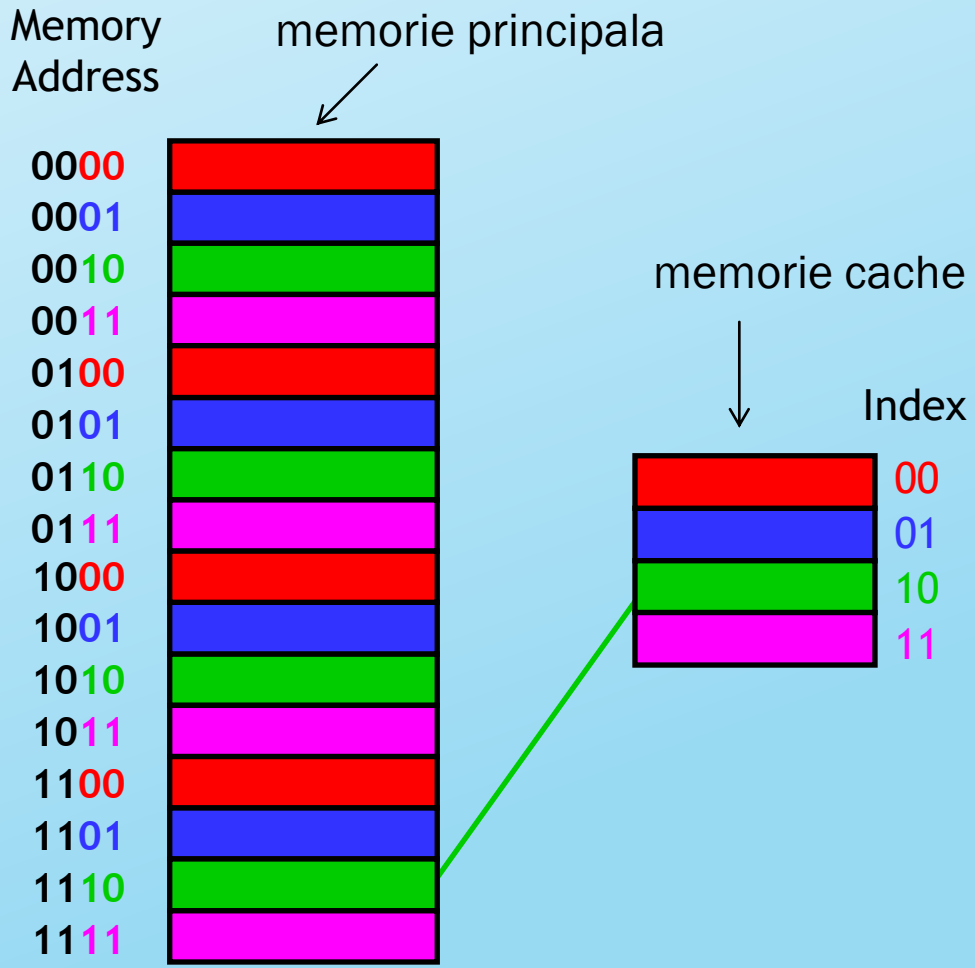
In cazul nostru: ultimii 2 biti

Se poate observa ca

adresa 14 (1110 in binar)

este mapata in
blocul 2 (10 in binar).

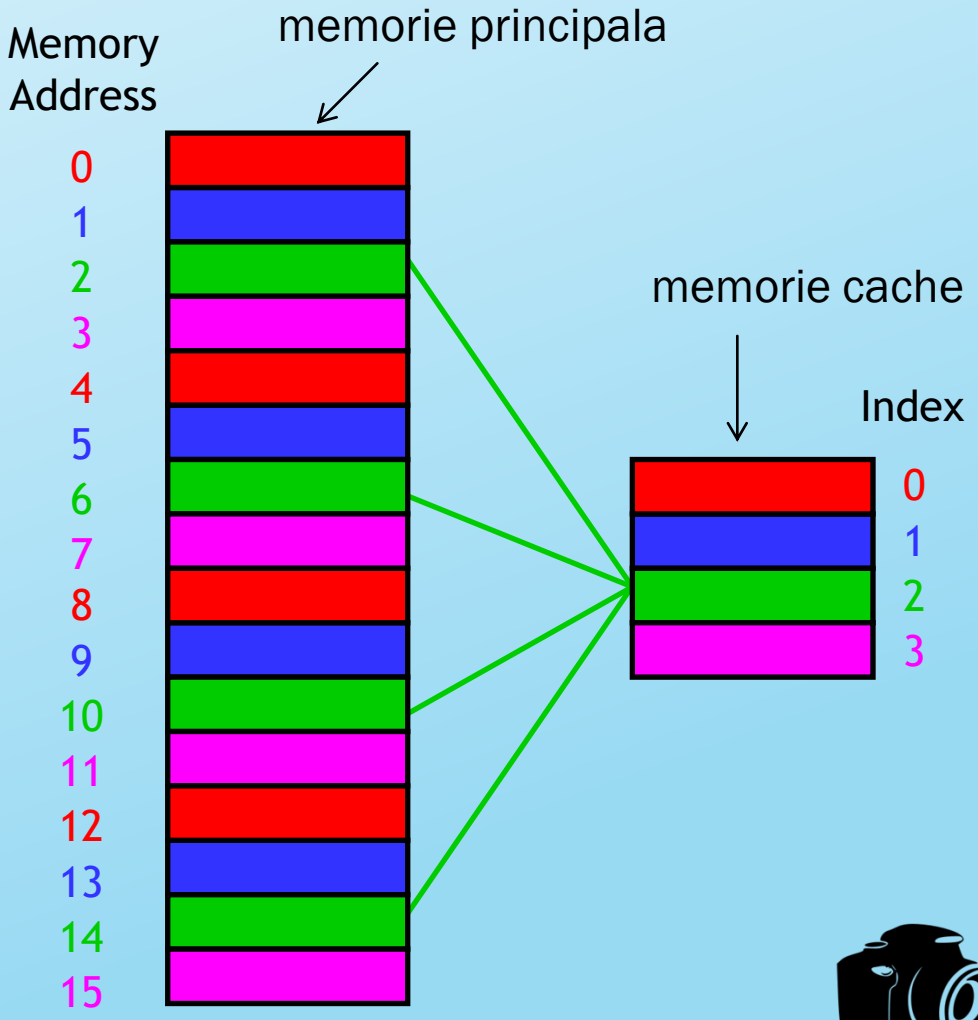
A lua cei mai putin semnificativi *k* biți dintr-o valoare binara insemna a calcula mod 2^k.



Cum se gasesc datele in cache?

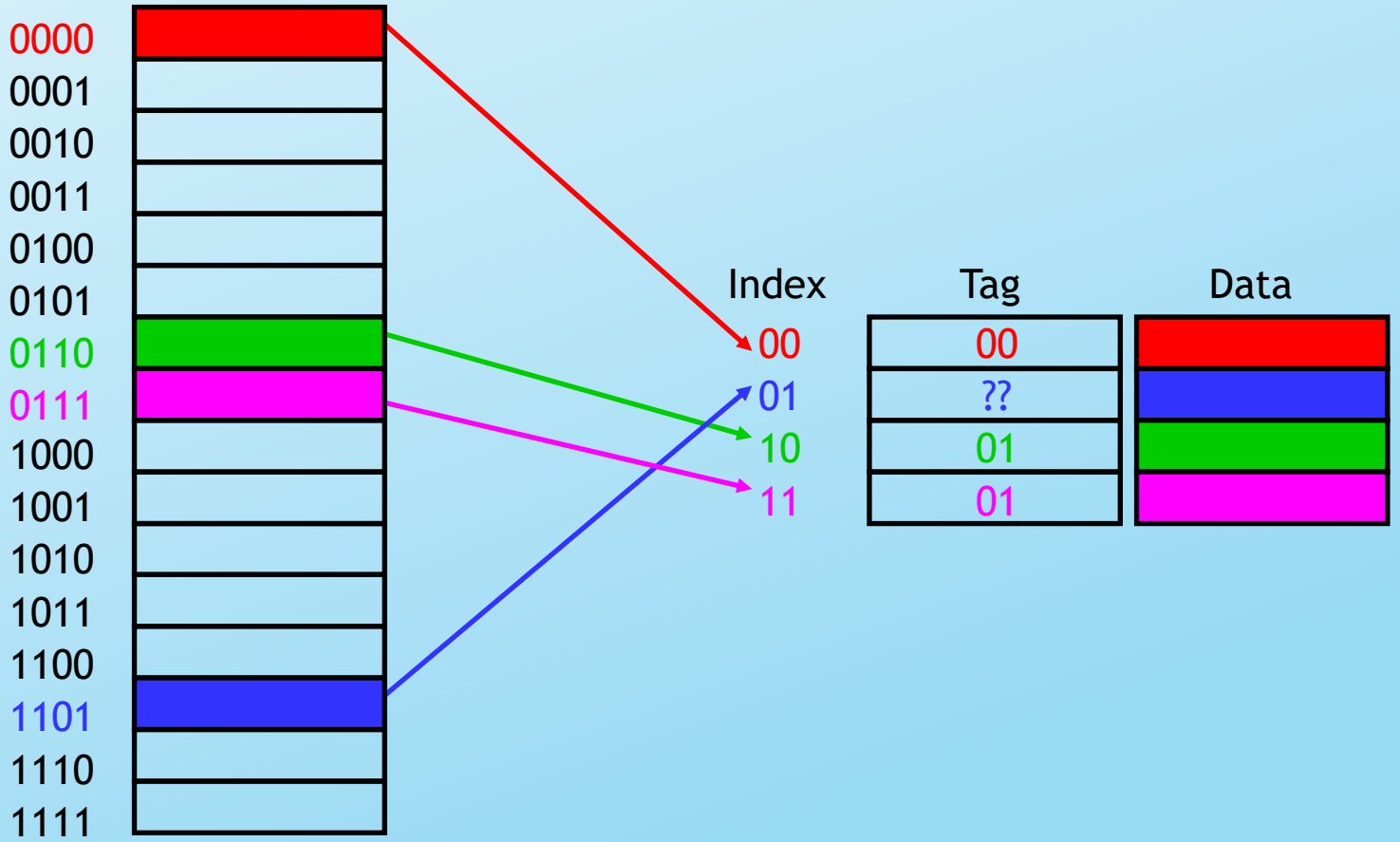
Prin folosirea metodei inverse: soluția este nedeterminata (nu este unica!)

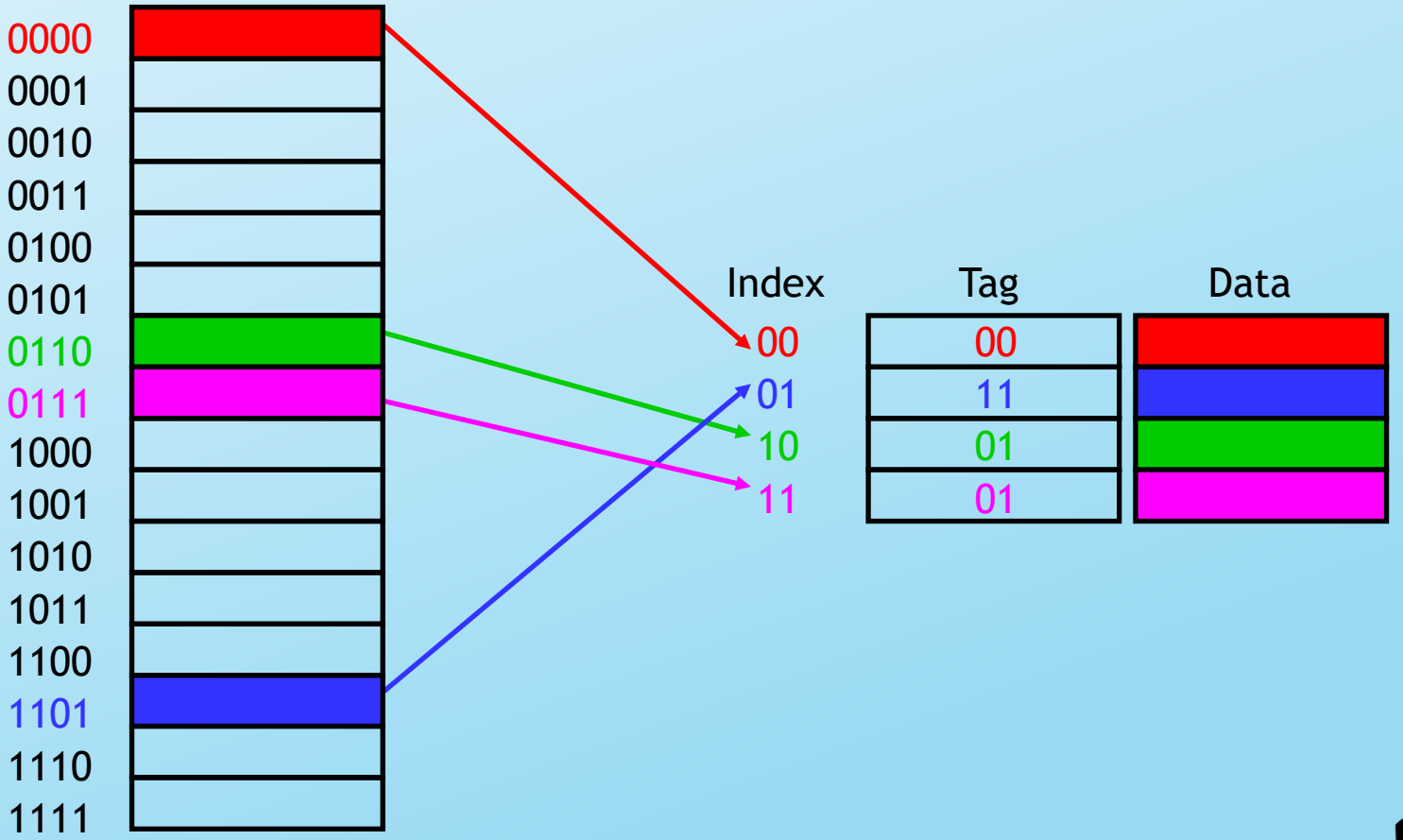
De exemplu: blocul 2
din cache poate conține date de
la adresele 2, 6, 10 *sau* 14.

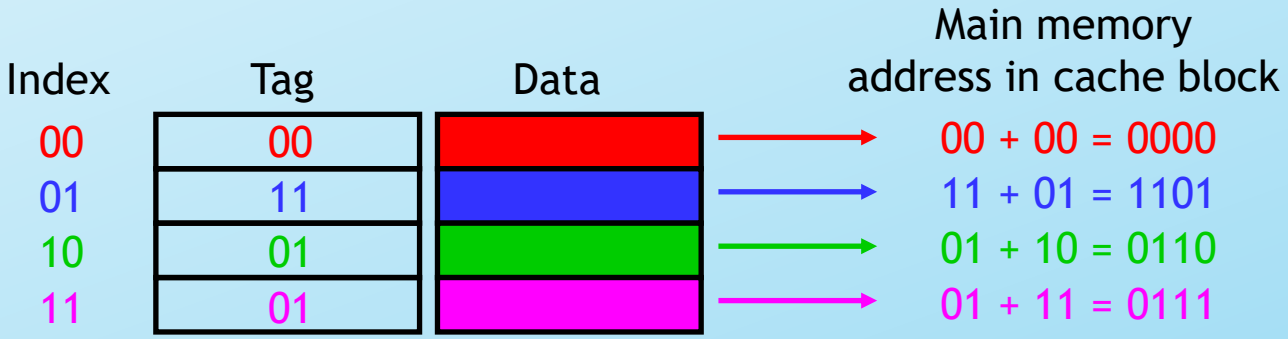


Trebuie introdus in cache o informație suplimentara: tag-ul.

- Suplineste lipsa informatiilor (restul de adresa)
- Distinge locatia de memorie din care provine blocul.




0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Index
00
01
10
11
Tag
00
11
01
01
Data

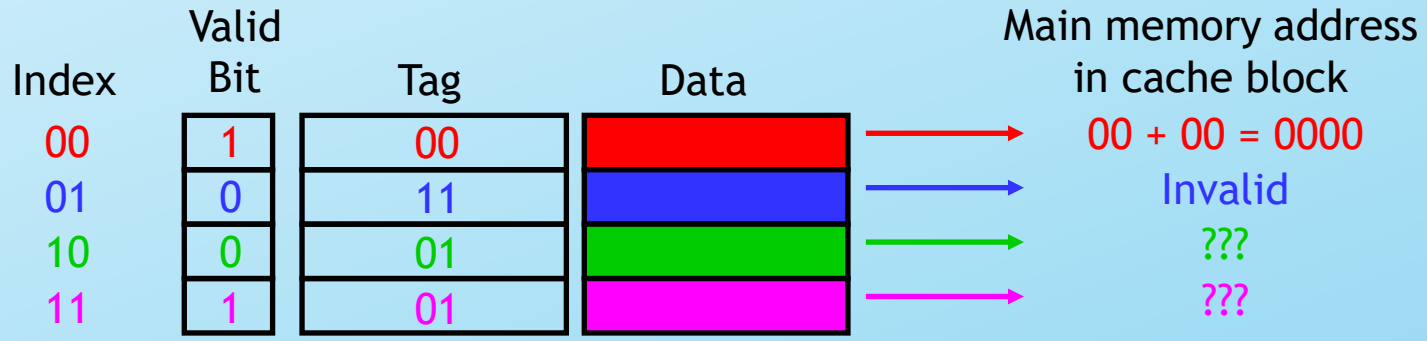


Tag+Index=adresa

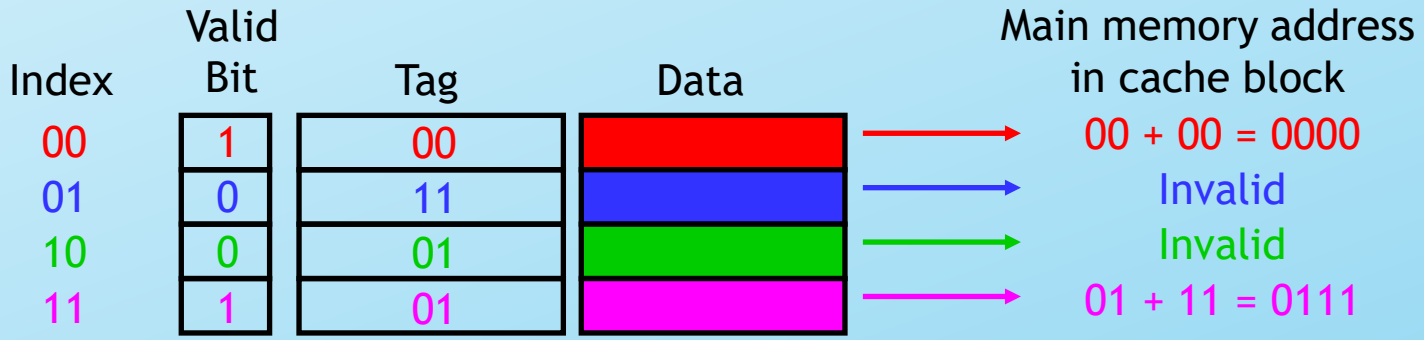
O informaţie suplimentara bitul de prezenţa (**valid bit**)

“lupta impotriva gunoiului”

In cache pot ramane informaţii din operaţiuni anterioare care nu mai sunt valide (de exemplu fac parte din alt program, cu execuţie finalizata, sau care au fost inlocuite in memoria principala)

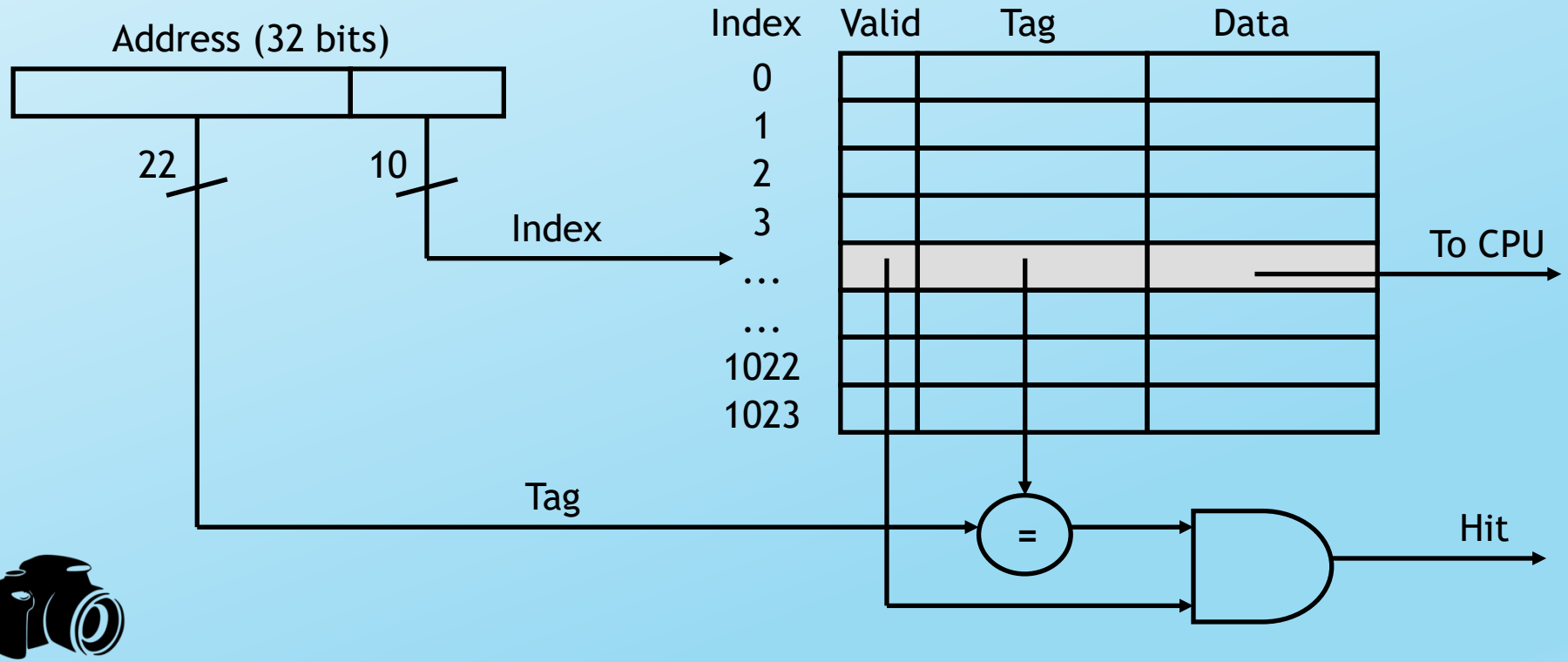


Bitul de prezența are valoarea 1 pentru date valide!


Index
Valid Bit
Tag
Data
Main memory address in cache block
00
01
10
11
1
0
0
1
00
11
01
01
00 + 00 = 0000
Invalid
Invalid
01 + 11 = 0111

Ce se intampla in cazul unui cache hit

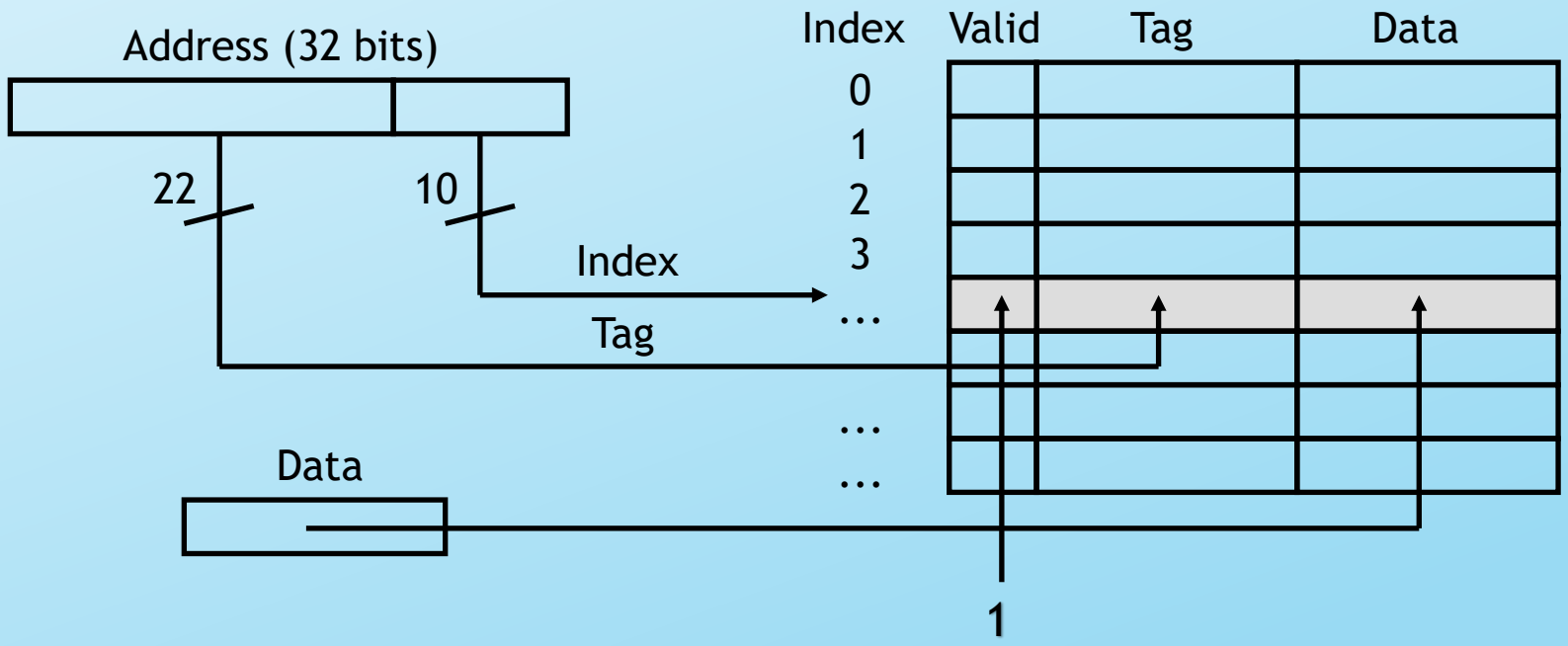
- Cand CPU inițiaza citirea din memorie, adresa (*m* biți) este trimisa unui controler de cache.
  - Cei mai puțin semnificativi k biți sunt transformați in index (cache) si se deschide accesul catre blocul corespunzator.
  - Daca blocul este valid si tag-ul se potriveste cu cei mai semnificativi (m-k) biti informația va fi trimisa catre CPU (impreuna cu un semnal de hit).



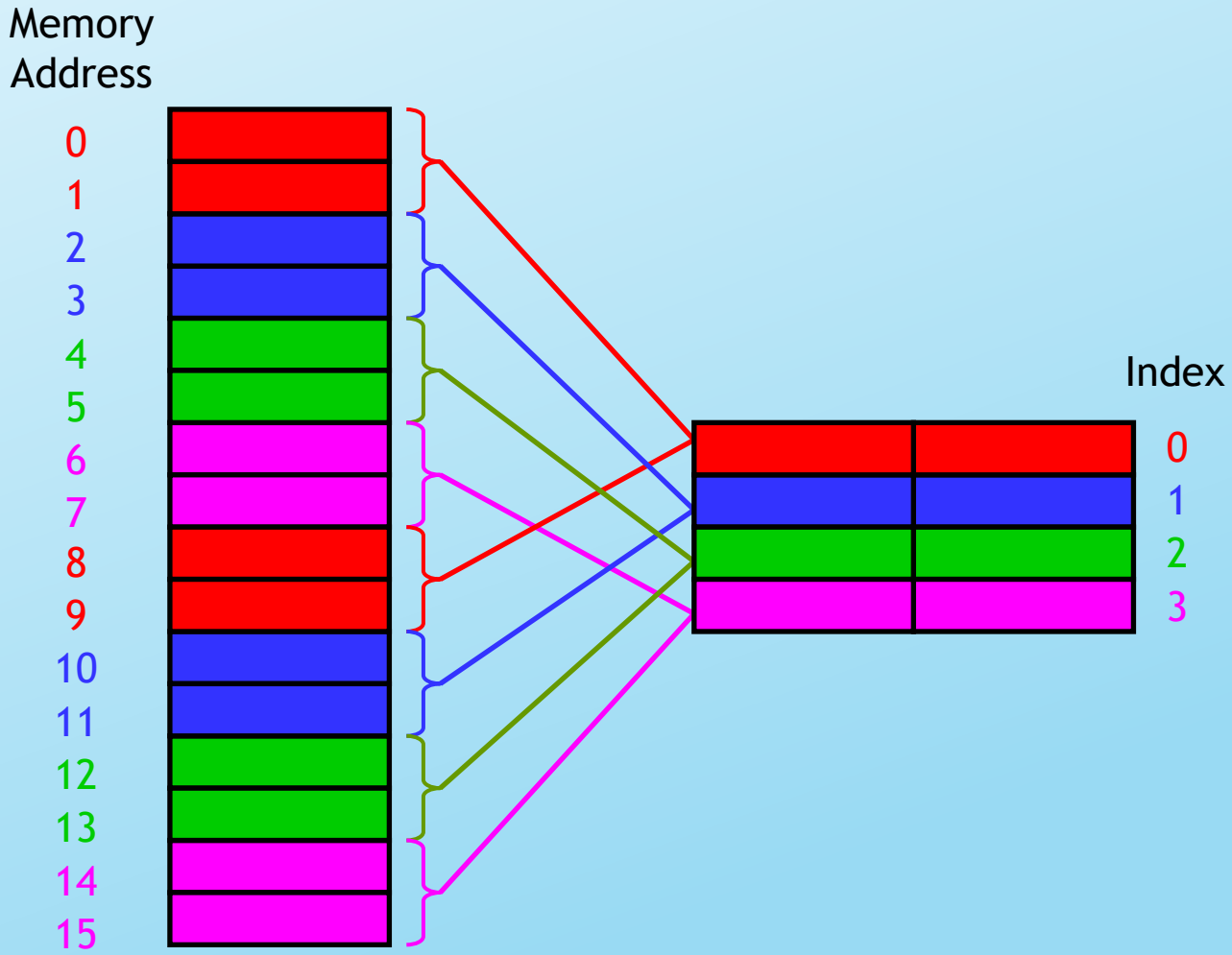
Ce se intampla in caz de cache miss

- Daca CPU nu primeste semnalul hit, (adica primeste 0 si nu 1) initiaza operatiunea de citire din memoria de rang imediat inferior. (in cazul sistemului cu un singur cache, citeste direct din memoria principala)

Incarcarea unui bloc in cache

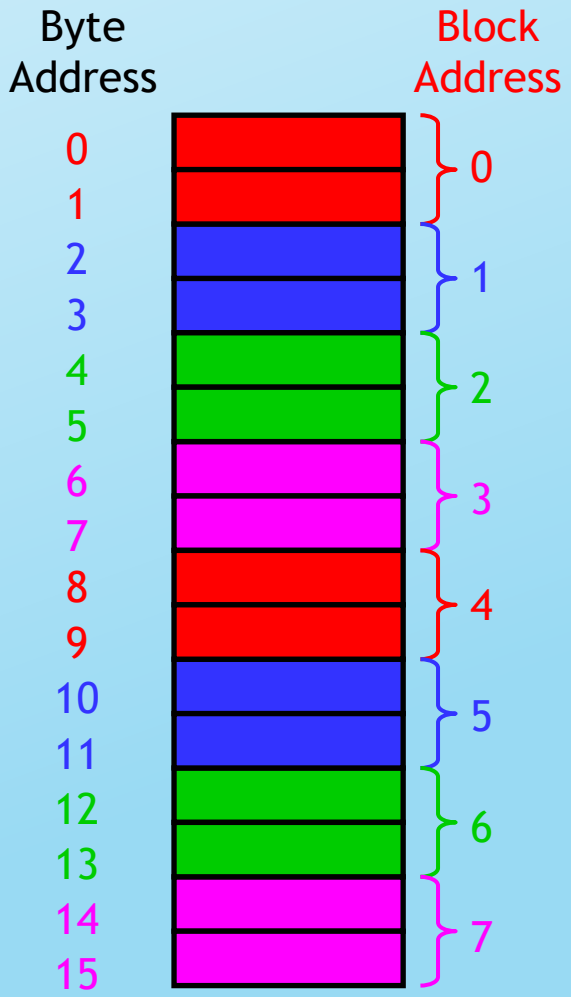


Memorii cache cu blocuri mai mari de 1 byte

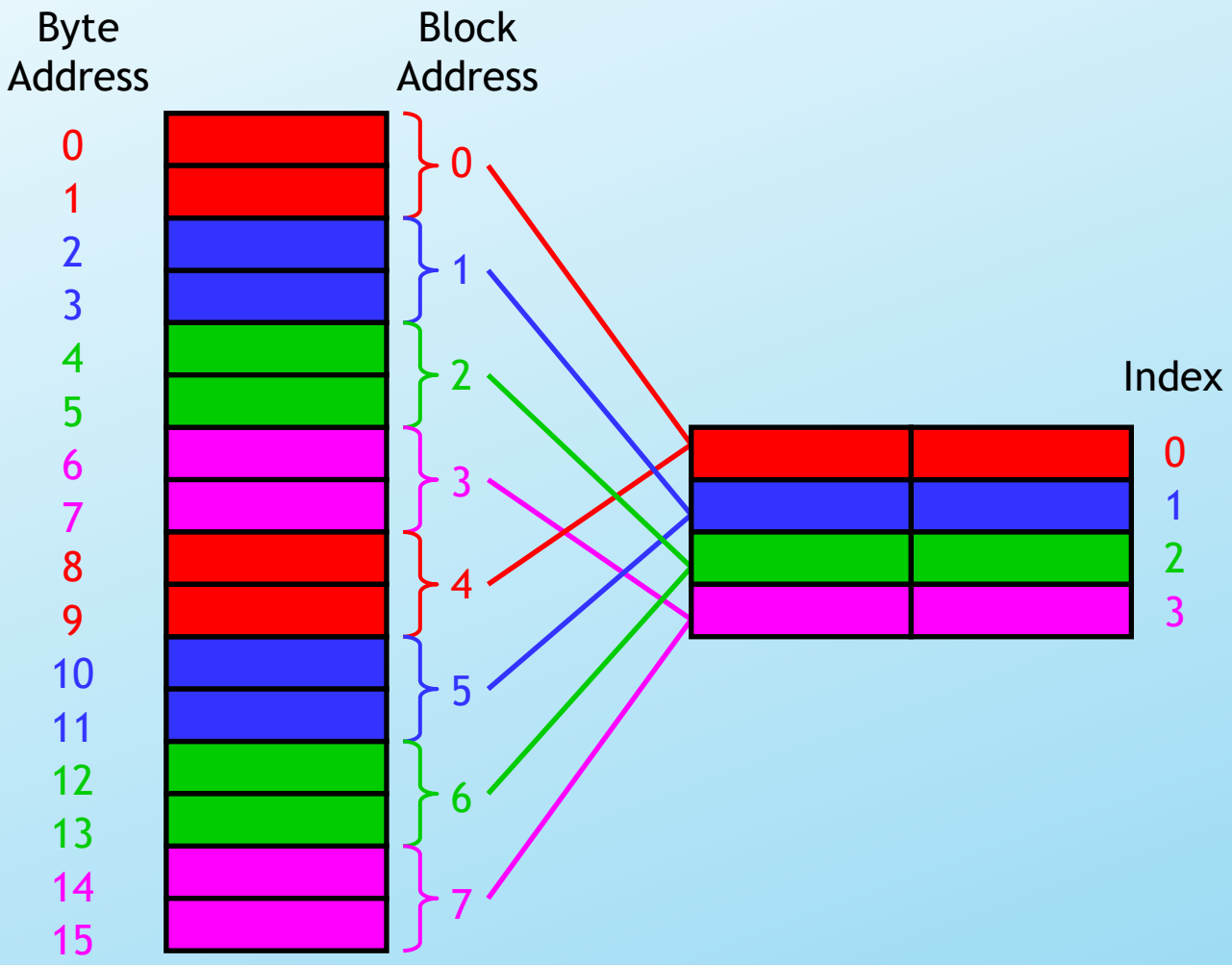


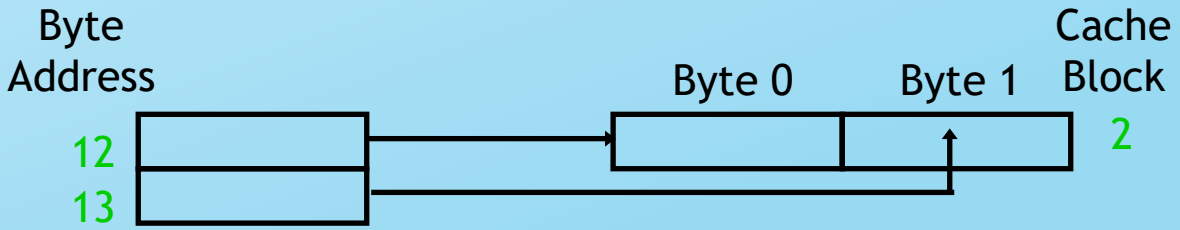
Adrese de bloc

- Daca un bloc din cache are 2^n byte, putem sa divizam memoria principala in fragmente de 2^n byte.
- Pentru a determina adresa (indexul) blocului din cache, adresa din memoria principala *i*, se imparte la 2^n (impartire intreaga)
- $i / 2^n$
- In exemplul nostru:
- Putem sa gandim o memorie principala de 16 byte ca o memorie cu 8 blocuri .



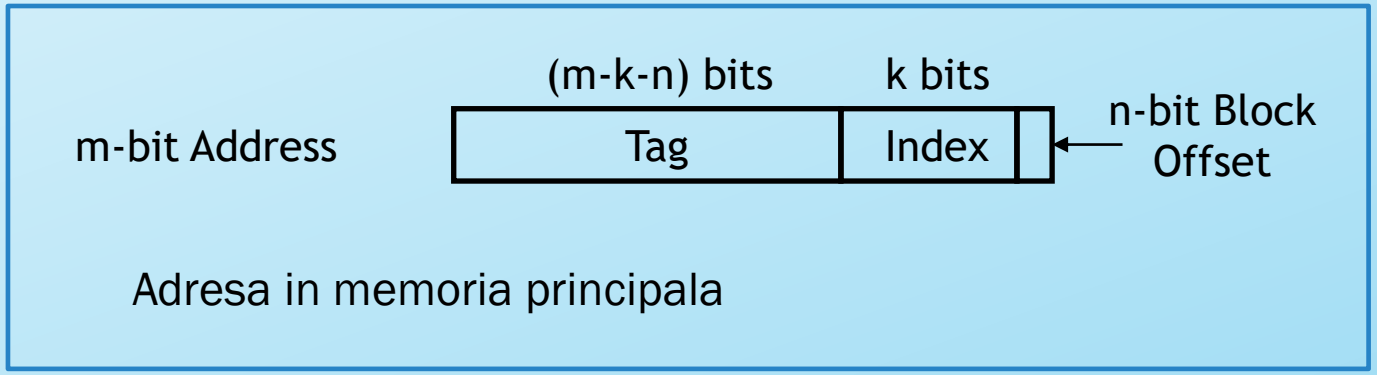
- De exemplu, adresele 12 si 13 corespund adresei de bloc 6, deoarece
- 12 / 2 = 6 and 13 / 2 = 6.


Byte Address
Block Address
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
2
3
4
5
6
7
Index
0
1
2
3

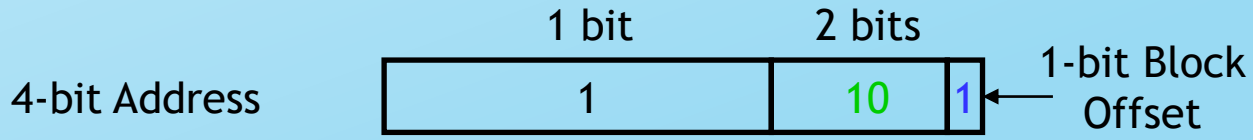

Byte Address
12
13
Byte 0
Byte 1
Cache Block
2

Localizarea datelor in cache

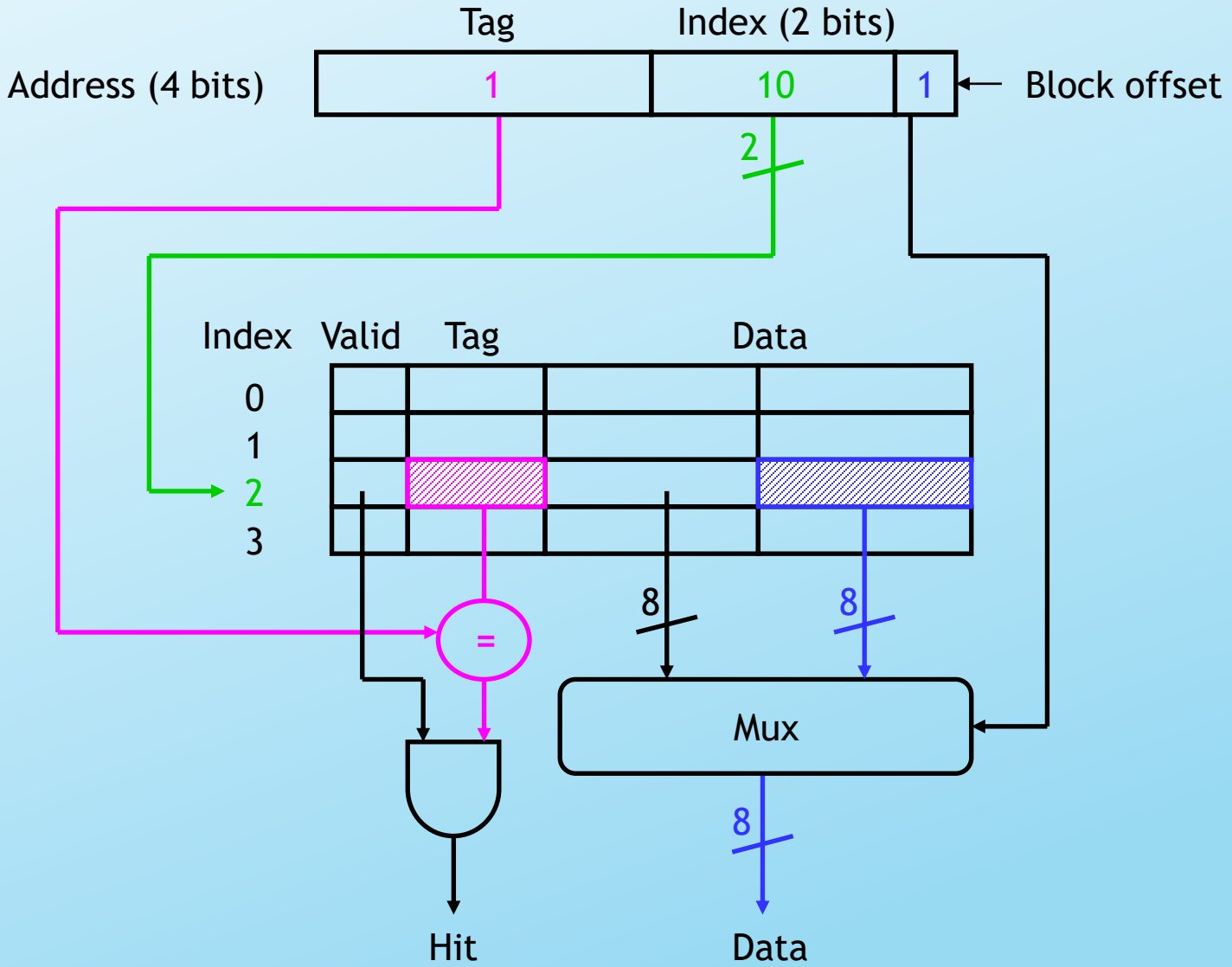
- Sa presupunem ca avem un cache cu 2^k blocuri continand 2^n byte.
- Putem determina pozitia in cache a unui octet plecand de la adresa sa din memoria principala.
  - Cei mai putin semnificativi n biti vor determina offset-ul care decide pozitia octetului in bloc.
  - Cei **k** biti vor identifica unul din cele 2^k blocuri din cache (index)



In exemplul nostru folosim un cache cu 2^2 blocuri si 2^1 byte per bloc. Astfel conținutul adresei 13 (1101) va fi stocata in locatia 1 din blocul 2.



dinspre CPU

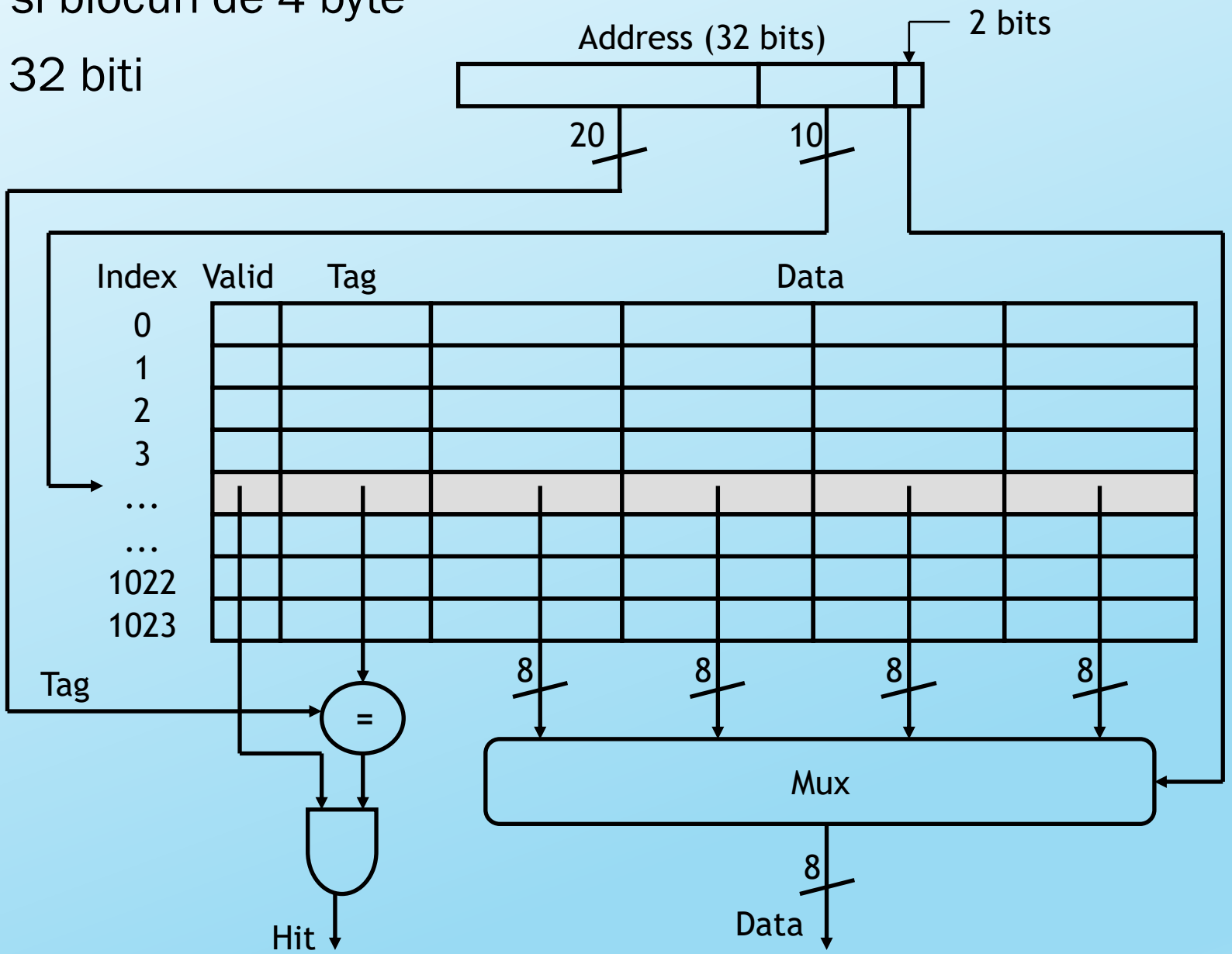


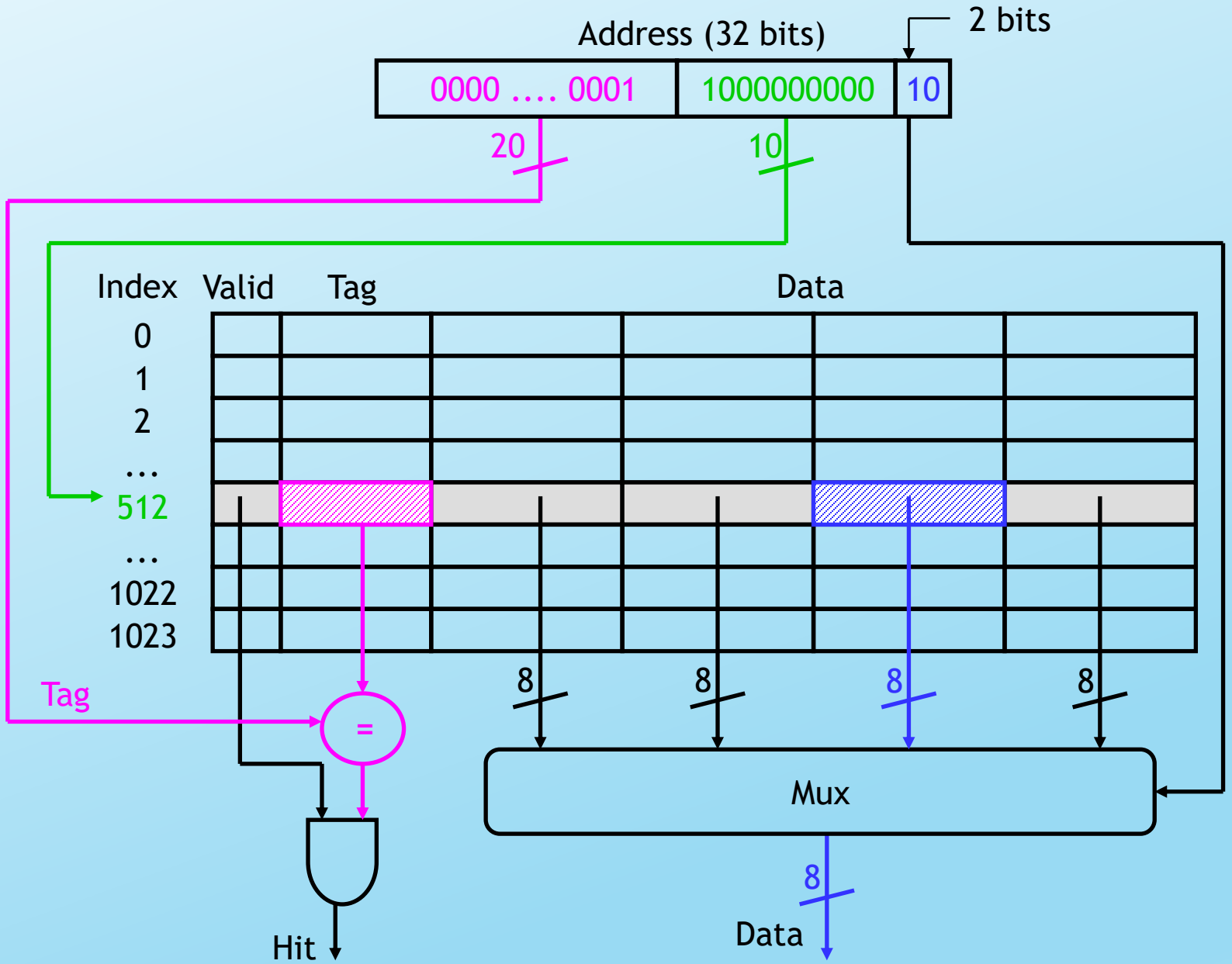
spre CPU

Un alt exemplu

Cache cu 1024 blocuri si blocuri de 4 byte

Memorie cu adrese de 32 biti



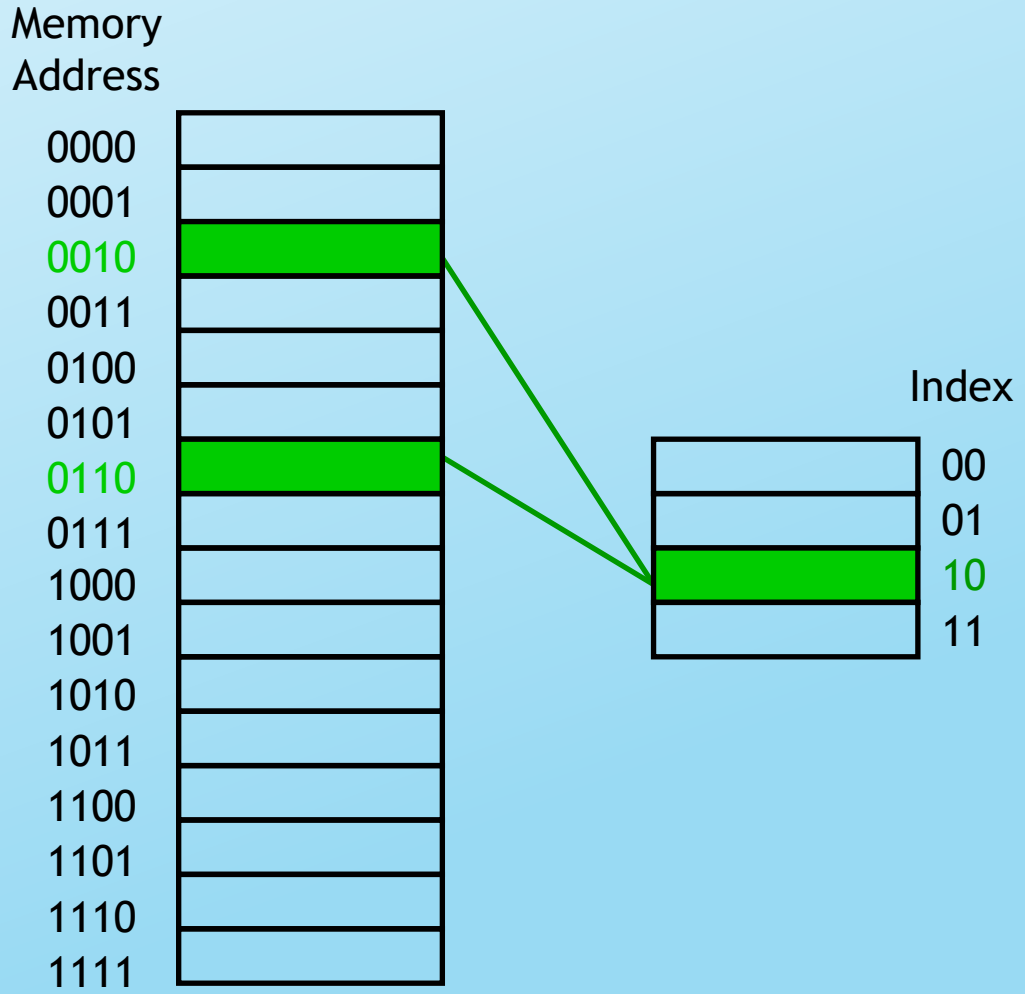

Address (32 bits)
2 bits
0000 0001
1000000000
10
20
10
Index
Valid
Tag
Data
0
1
2
...
512
...
1022
1023
Tag
=
8
8
8
8
Mux
8
Data
Hit

Dezavantajele maparii directe

Daca programul utilizeaza succesiv adresele 2, 6, 2, 6, 2, ...,

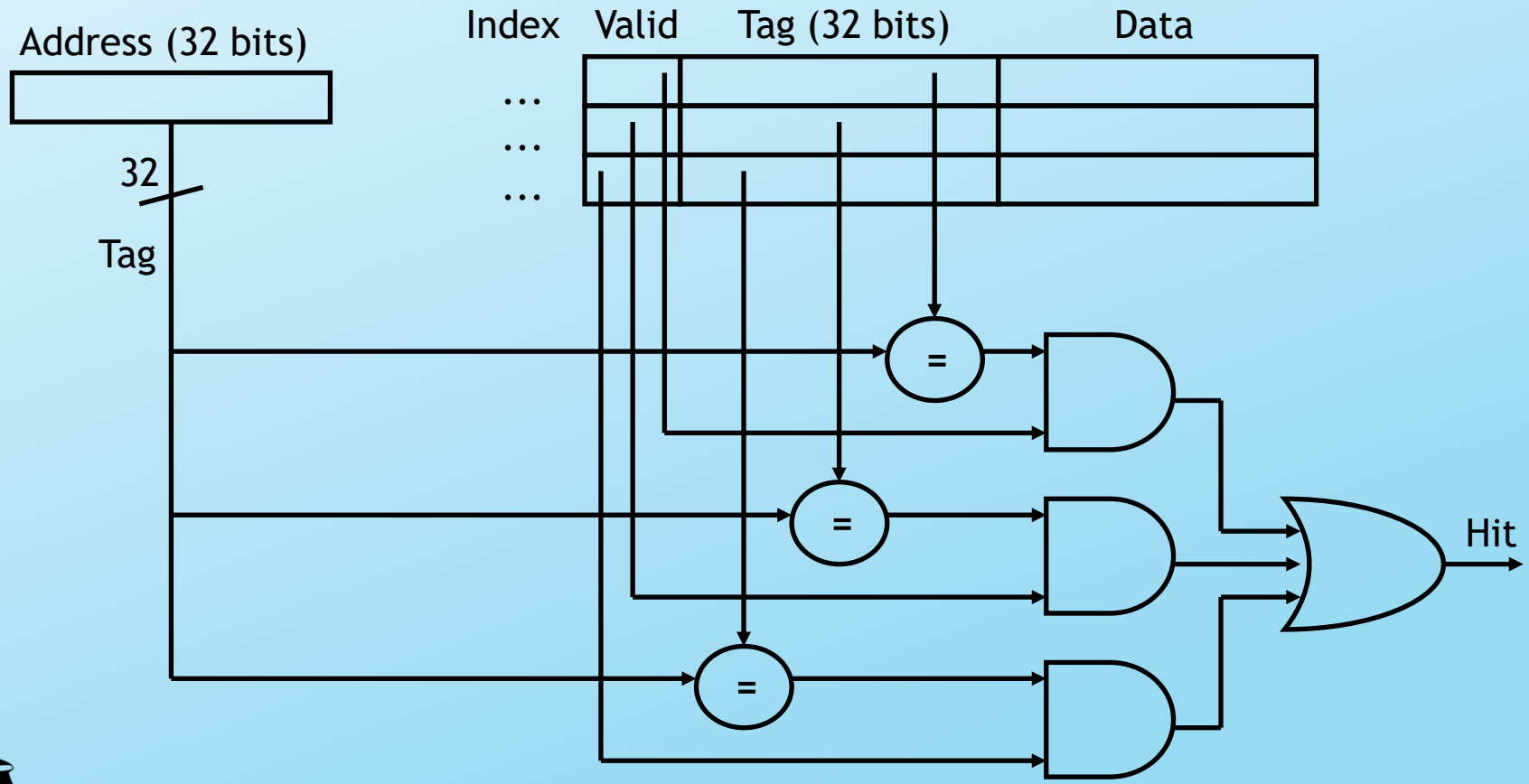
Rezulta cache miss de fiecare data

Ingreuneaza accesul la memorie



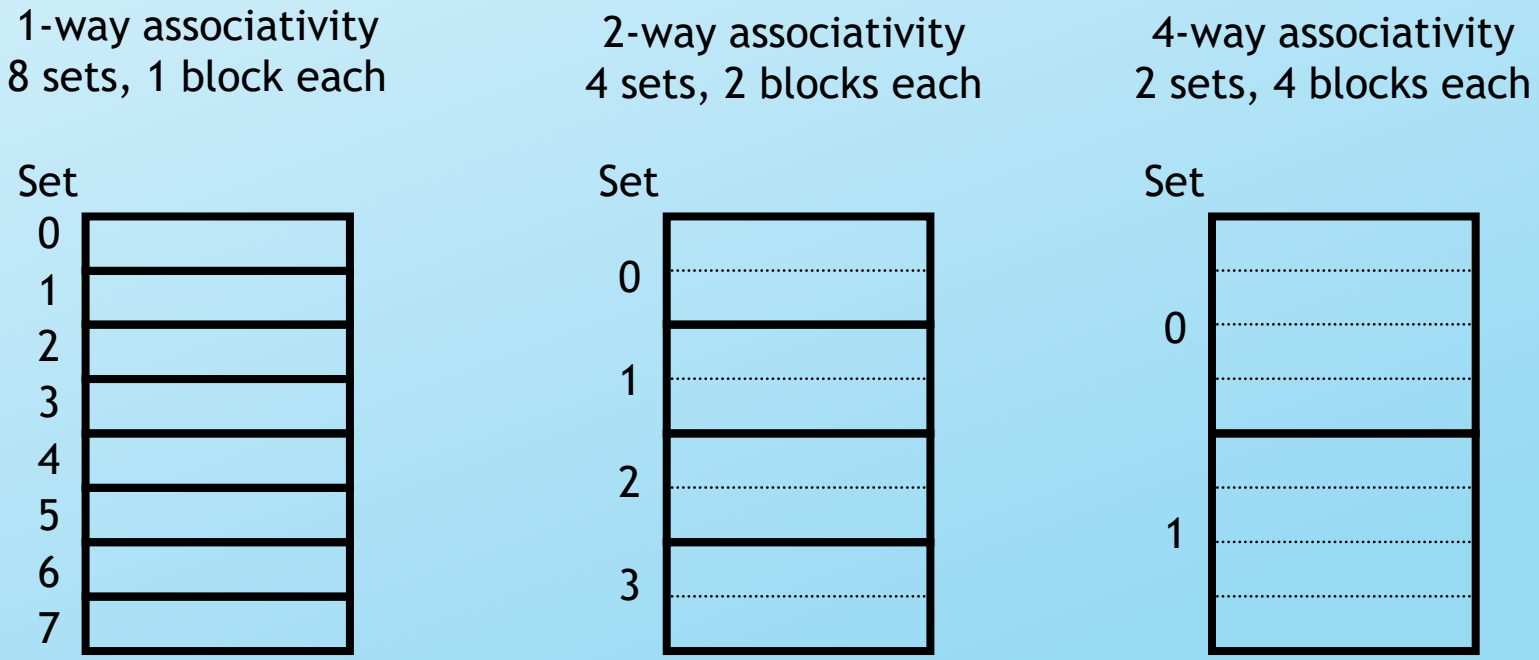
Maparea complet asociativa

- Datele pot fi amplasate in orice bloc
  - tag = adresa completa

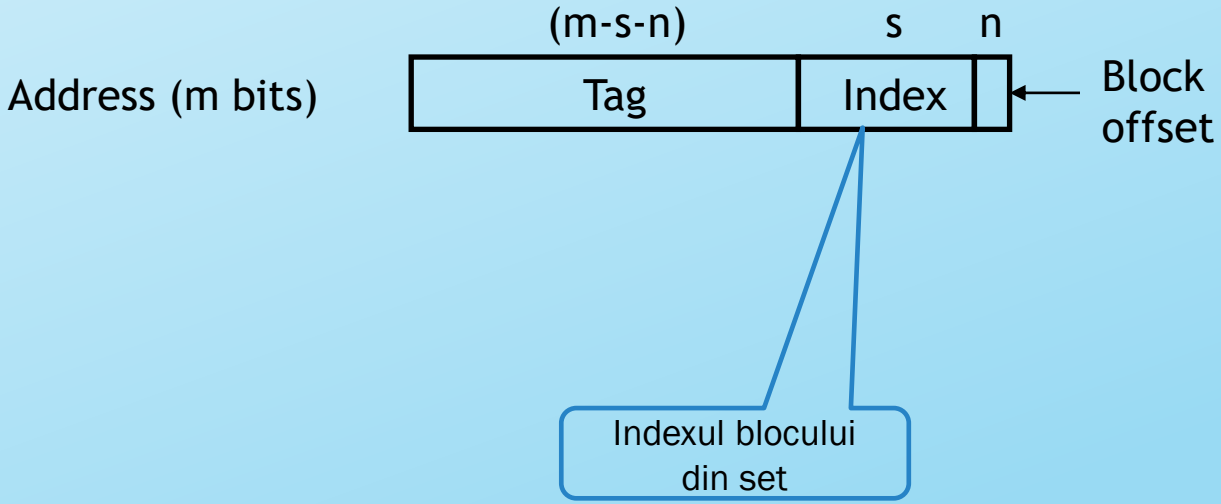


Maparea directa in seturi asociative

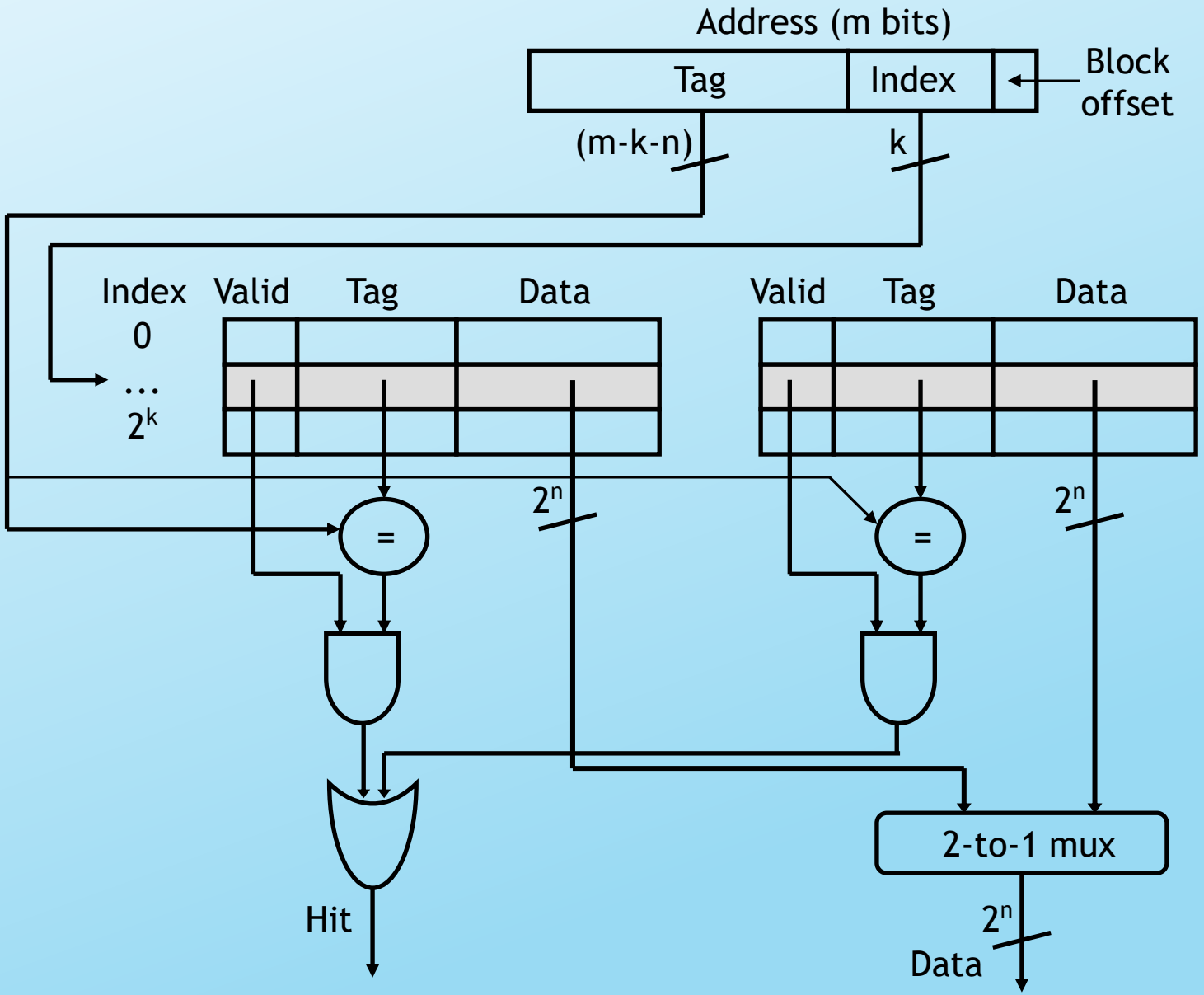
Blocurile sunt grupate in seturi. Datele sint inmagazinate orice bloc dintr-un set, dar intr-un anumit set.

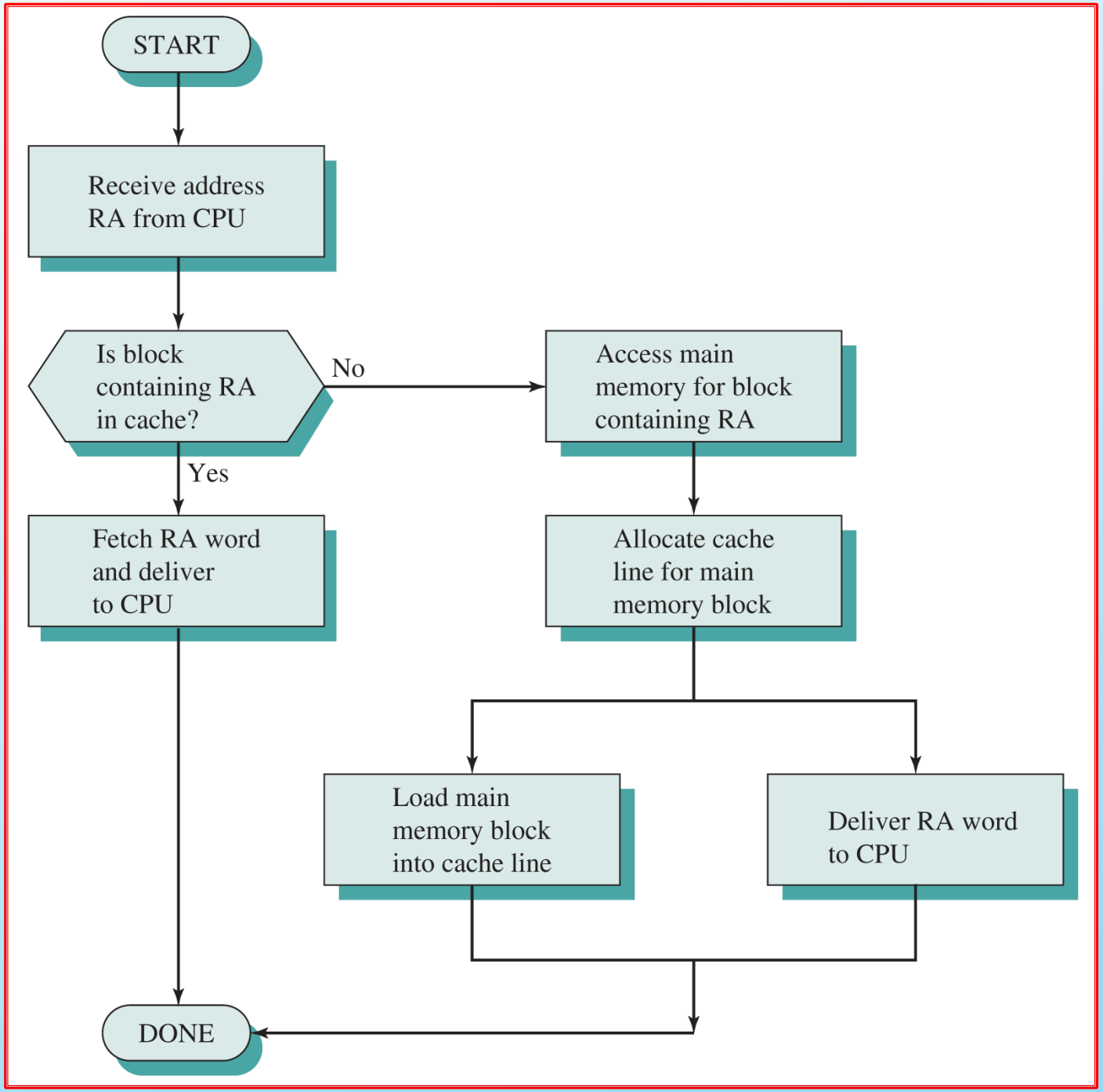


- Cache cu 2^s seturi ,
- Fiecare set cu 2^x blocuri (2^x-way associative cache)
- Fiecare bloc cu 2^n byte

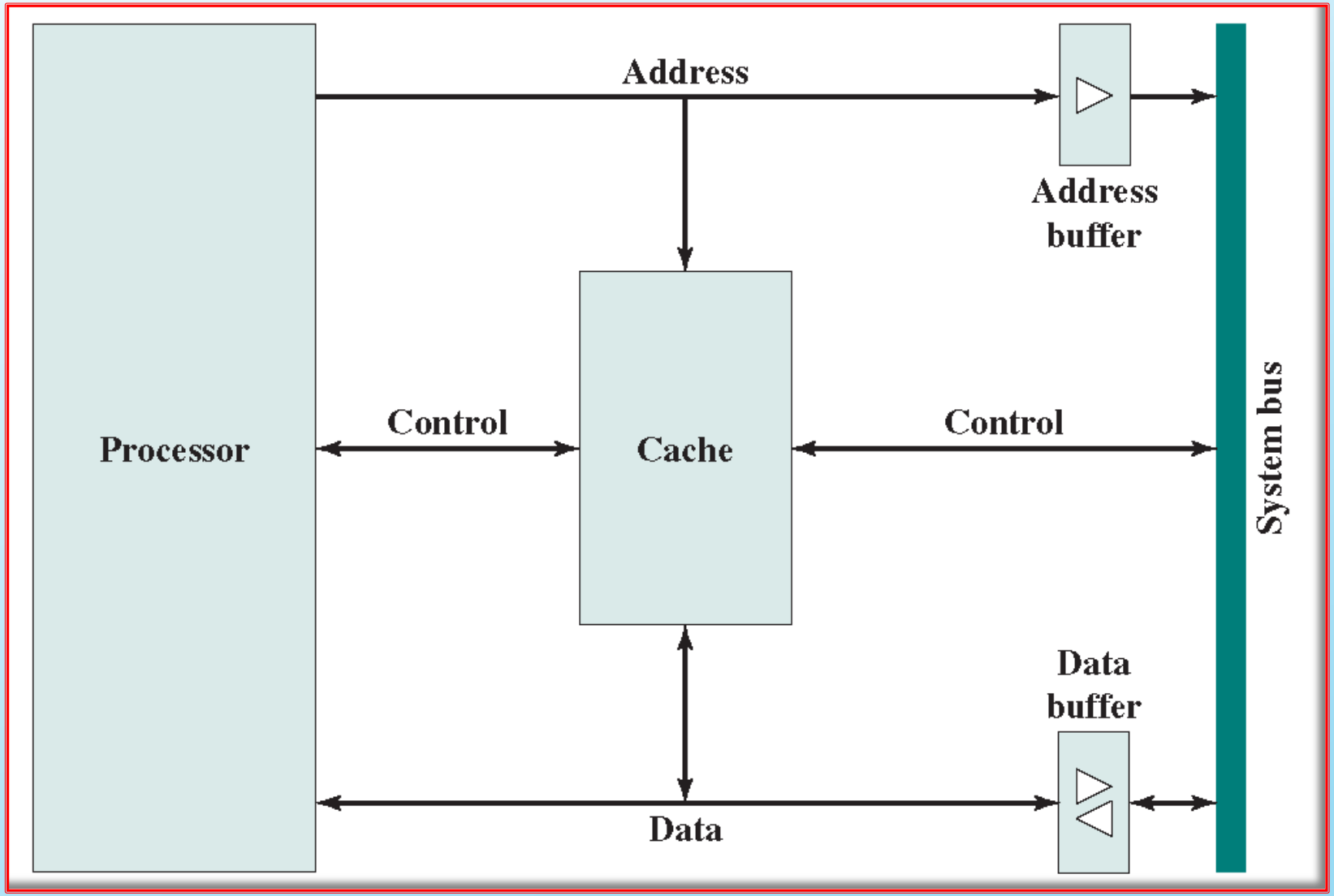


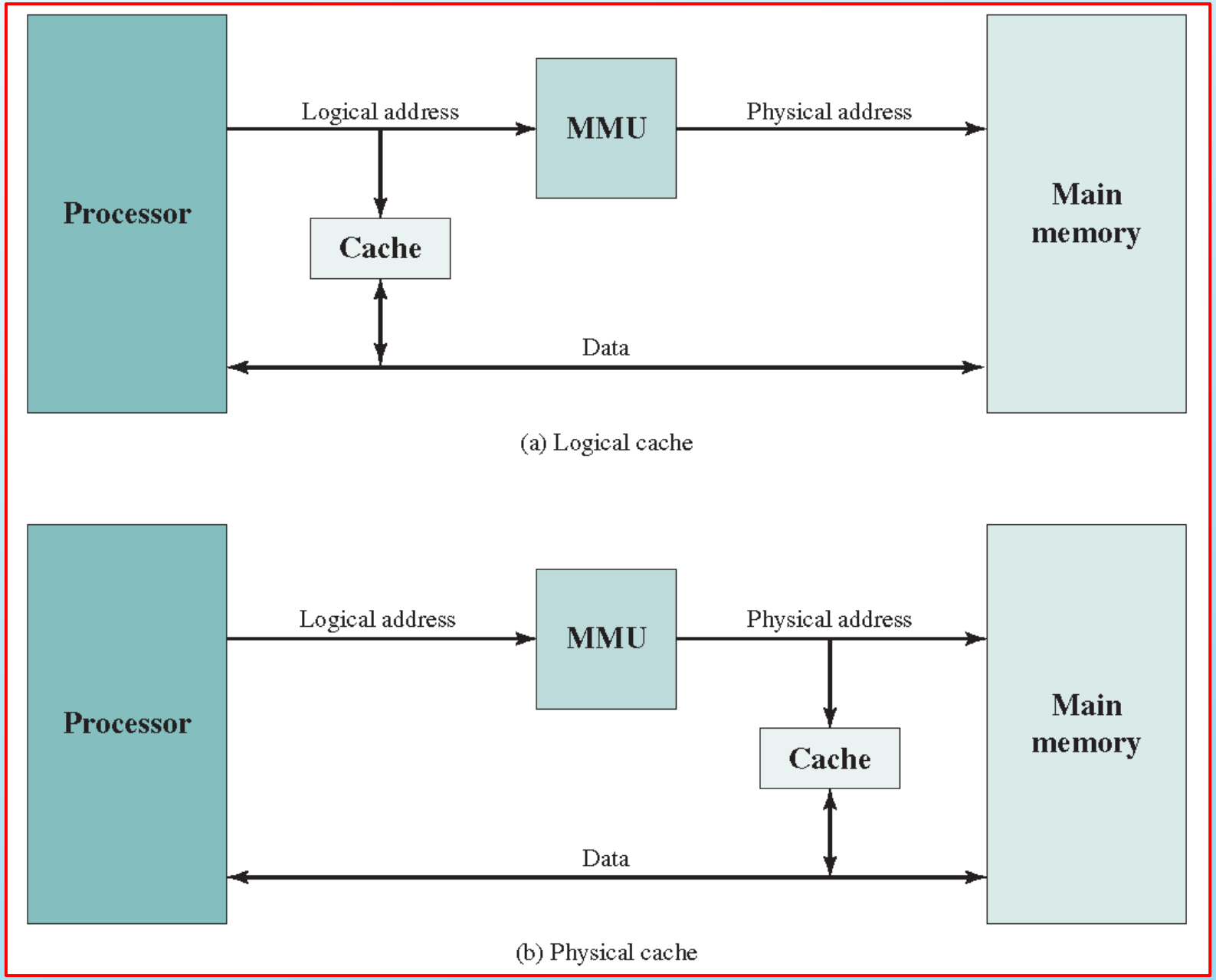
2-way set associative cache





RA= read address

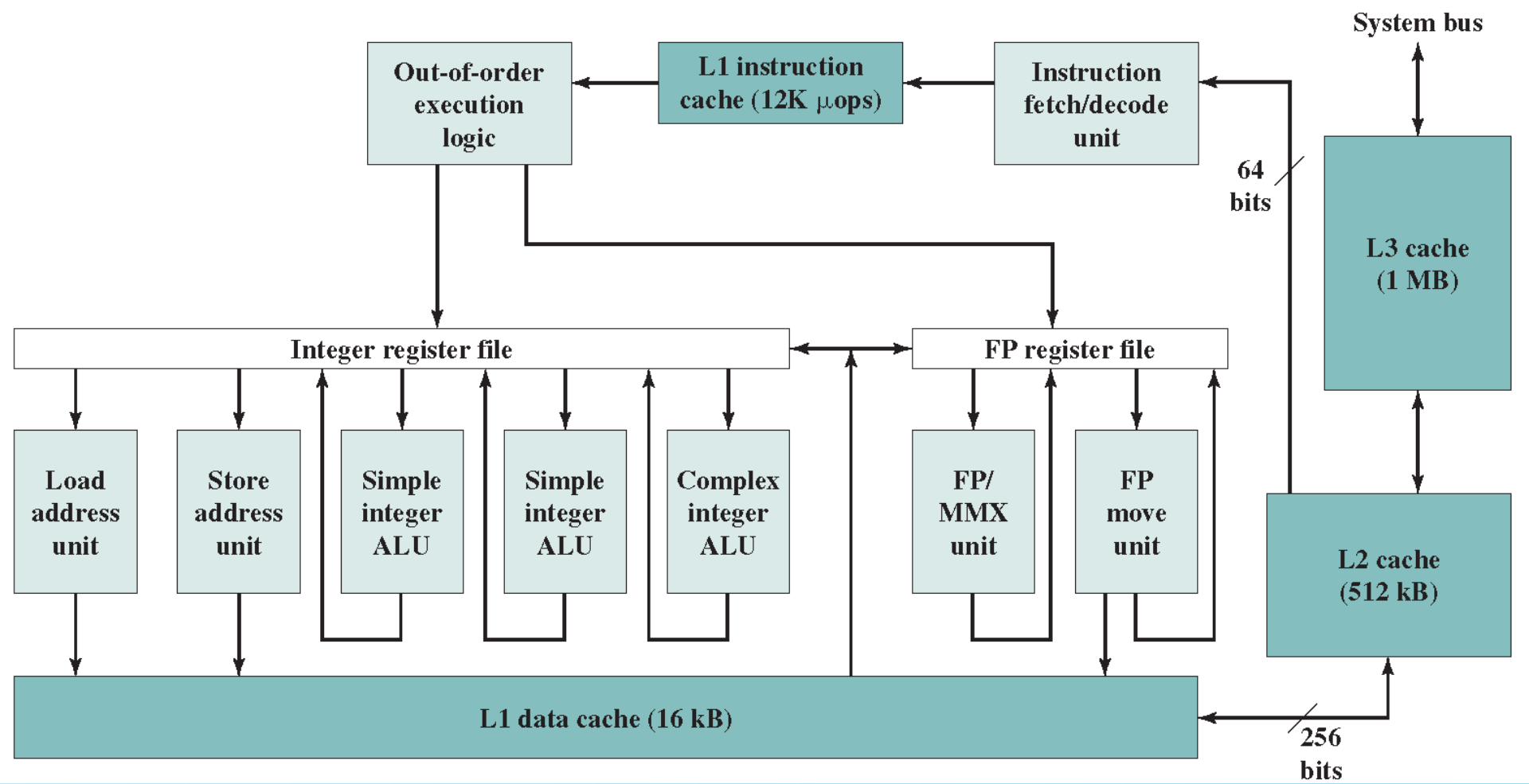

Address
Address buffer
Processor
Control
Cache
Control
System bus
Data buffer
Data



MMU="MEMORY MANAGEMENT UNIT"

TRANSLATEAZA ADRESELE LOGICE IN ADRESE FIZICE

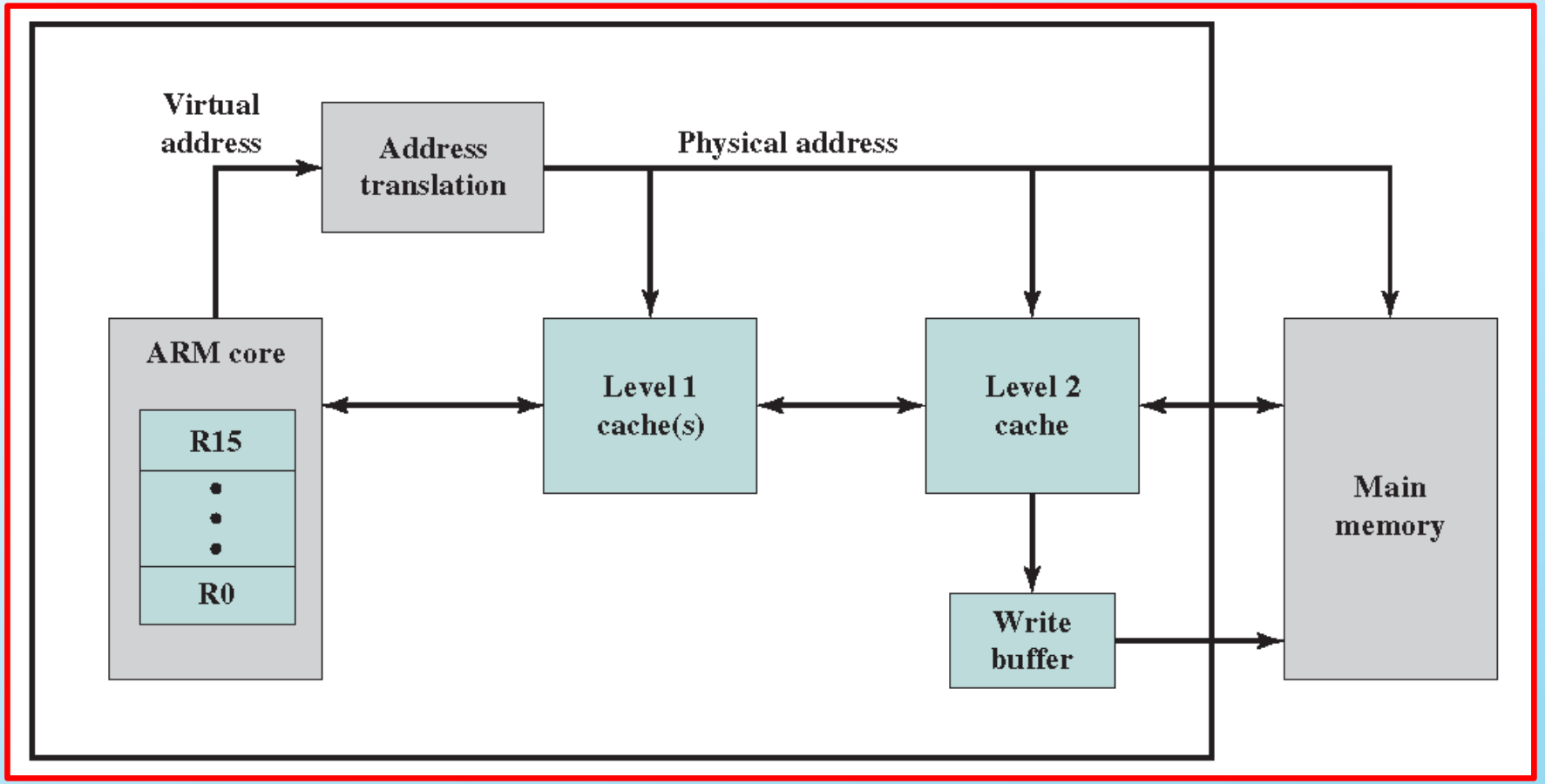
Processor	Type	Year of Introduction	L1 Cache[a]	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16–32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128–256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256–512 kB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 kB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 kB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 kB	4 MB
Itanium 2	PC/server	2002	32 kB	256 kB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1 MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24–48 MB
Intel Core i7 EE 990	Workstation/ server	2011	6 × 32 kB/ 32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4



CACHE UTILIZAT DE MICROPROCESORUL PENTIUM 4

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

ARM


Virtual address
Address translation
Physical address
ARM core
R15
R0
Level 1 cache(s)
Level 2 cache
Write buffer
Main memory

ARM