Lab 13

Ierarhia de memorii.

Memoria cache

In PC: elementul central = unitatea centrala de prelucrare CPU a informațiilor (localizată în procesor)

- înconjurat de **o serie de circuite cu rol de memorare** a informațiilor
- organizate pe **mai multe nivele**, într-o **structură ierarhică** (în funcție de distanța față de CPU)

Pe măsură ce **se indepărtează de CPU**, nivelele de memorie* au **capacitate mai mare**, dar **viteză mai mică**

Nivelul regiştrilor procesorului - au cel mai mic timp de acces, se afla in acelaşi circuit cu CPU.

Se doresc registrii cât mai multi posibil pt a mări performanța, dar practic: nr lor este redus

=>e imposibil ca o aplicație decenta să se poată executa exclusiv cu ajutorul regiştrilor (fără a face deloc apel la nivelele următoare de memorie).

Nivelul memoriei cache («*imediată*»)

= singurul nivel care poate lipsi, fără ca aceasta să implice o schimbare în programele care rulează

-> **nu se poate gestiona prin software (prin programe)**

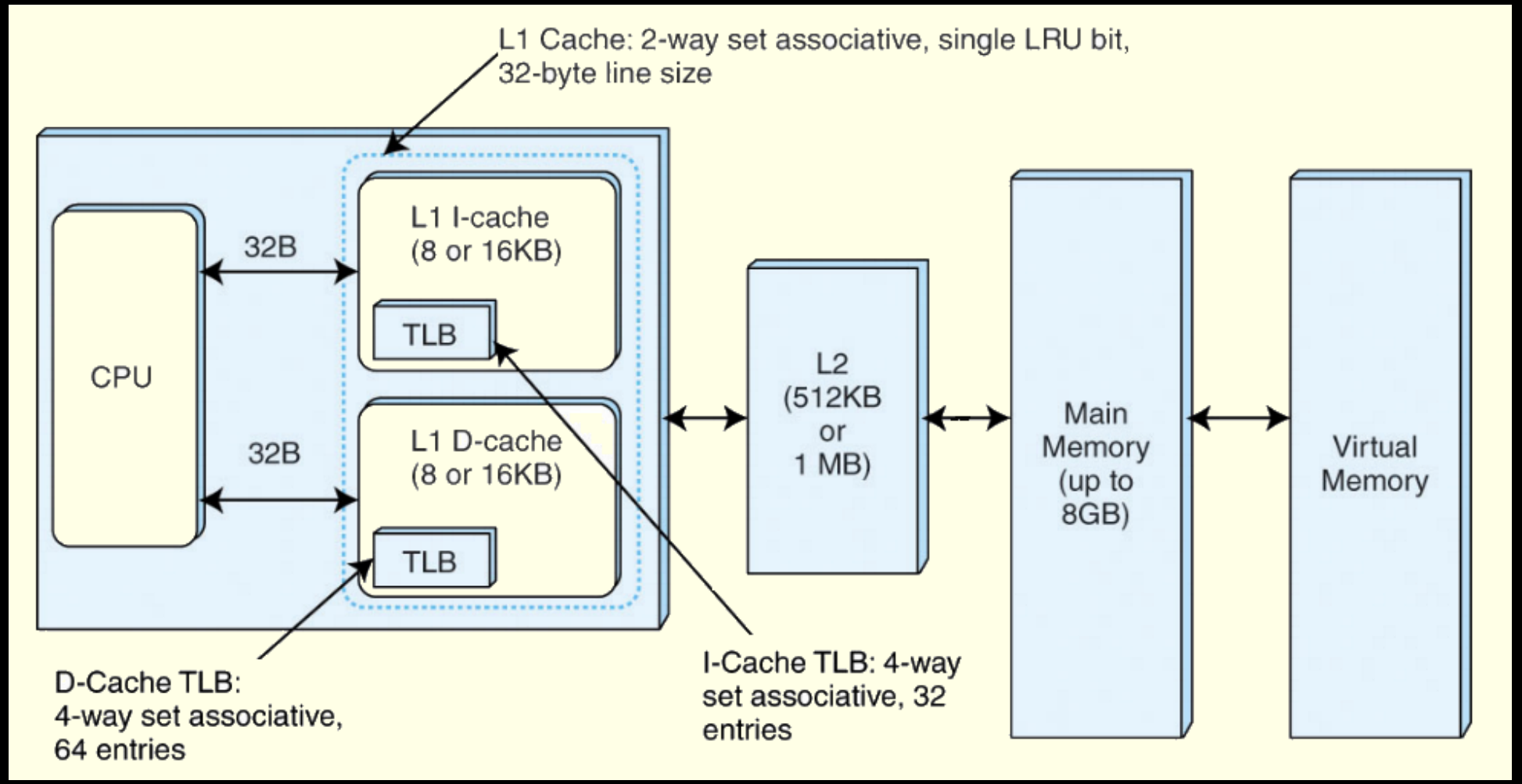
- memoria cache poate fi împărțită pe nivele: poate exista un modul cache în interiorul procesorului (numit cache L1), de capacitate mica şi care funcționează practic la viteza procesorului,
- un alt modul pe placa de bază (cache L2), care este mai mare decât cache-ul L1 şi puțin mai lent.
- unele implementări pot lucra chiar cu 3 nivele de cache, dintre care 2 fiind integrate în procesor.

Nivelul memoriei principale –«*RAM*» (prelucrarea informației înseamnă implicit modificarea acesteia);

Nivelul memoriei secundare - stocare *persistentă*. Spre deosebire de nivelele anterioare, care sunt volatile, la acest nivel informațiile se păstrează şi după întreruperea alimentării calculatorului. Tot aici se considera si memoria *virtuală*. Formele de implementare a memoriei secundare sunt: discul dur (cel mai folosit), discheta, mediile optice (CD, DVD, BD), banda magnetică etc.

*memoria ROM este considerata caz special, nu se incadreaza aici -> vezi BIOS

Un exemplu real de sistem cu memorie



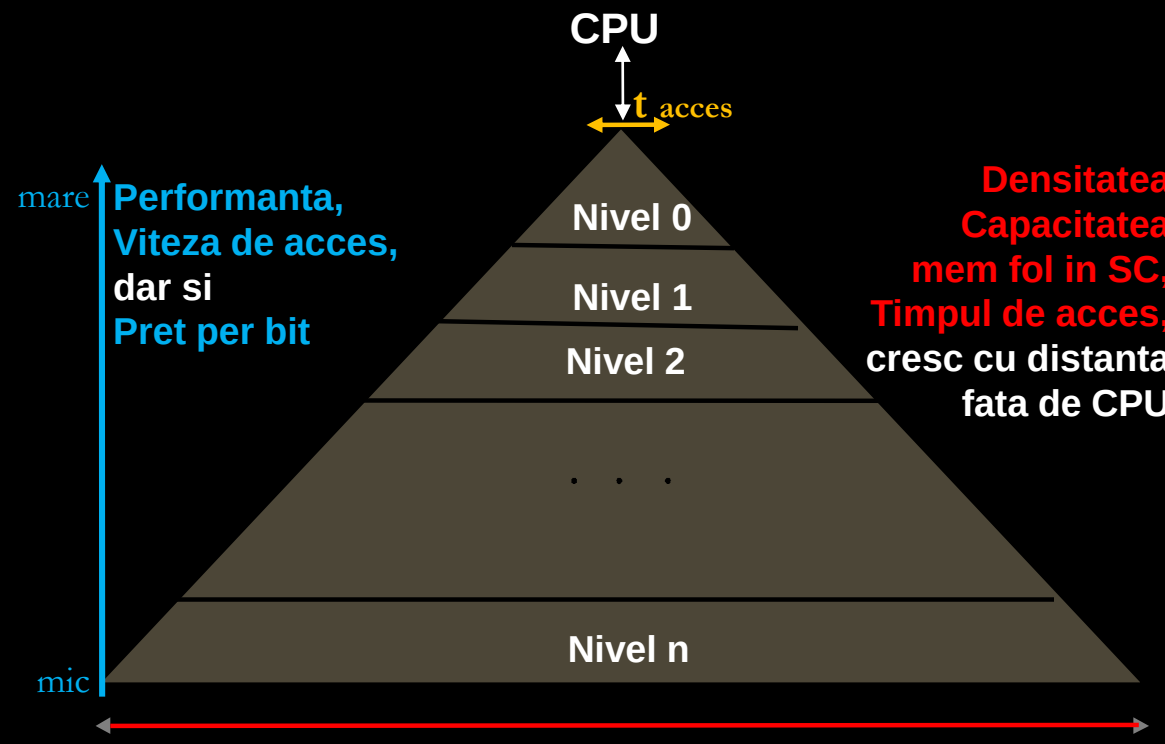
[Null]

In Sistemele de Calcul (SC) – COMPROMIS intre *densitate si viteza:*

ideal ar fi memorie cat mai mica (dpdv al dimens fizice) dar cat mai mare (dpdv al capacitatii),
de viteza cat mai crescuta -> nu prea e posibil fara costuri ridicate

=>* sistem cu *performante bune* la *pret bun* : *o combinatie de memorii

Organizarea ierarhica a memoriei = nivele diferite de mem cu timp de acces si capacitate diferite



Un posibil exemplu:

Nivel 0 – registrele CPU
Nivel 1 = cache nivel 1 - SRAM
(inclus in procesor)
Nivel 2 = cache nivel 2 - SRAM
(extern sau inclus)
Nivel 3 = *mem. Principala*- DRAM
(module pe placa de baza)
Nivel 4 = unitate disc magnetic - HDD
(in cutie, on-line)=*mem. Secundara*
Nivel 5 = mem. optica - CD-ROM
(near-line)
Nivel 6 = caseta/banda magnetica
(off-line, pentru back-up)
Nivel 7 = biblioteci - casete/CD/DVD
(shelf=raft automatizat)

memoria = componenta funcțională destinată păstrării informației

- clasificare in functie de distanta fata de procesor

[masurata in nr cicluri masina necesare accesarii]:

1. Registrele CPU : 1-2

2. cache

- de nivel 1 (interna) : 3-10

- de nivel 2 (interna sau externa) : 25-50

(de nivel 3 - externa)

4. memoria principala : de aprox 10x mai multi decat cache L1

5. memorie secundara (HDD) : de aprox 1000x mai multi decat cache L1

- cu cat e mai aproape de CPU, cu atat e mai rapida
- cu cat e mai rapida, cu atat tehnol de implem e mai evoluata => Costuri per bit mai ridicate

Densitate mai ridicata (dist mai mici intre tranzistoare)

In sist de calcul, compromis:

cele scumpe –> capacitate mai mica (nu ne permitem prea multa de acest tip)

cele ieftine –> capacitate mai mare => se obtin **sisteme cu performante bune la costuri *acceptabile***

Ierarhia de memorii – bazata pe principiile localitatii

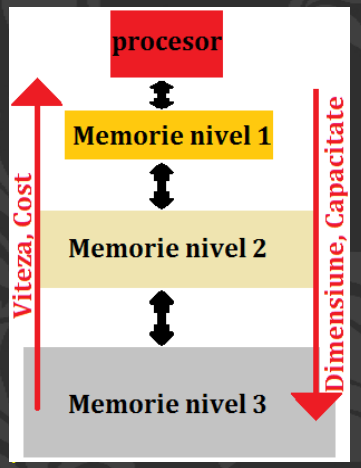
Pt a accesa o anumita informatie (instructiune sau data) , UCP trimite o cerere catre memoria cea mai apropiata (in gen, mem cache); Daca informatia resp nu se afla in cache, atunci se cauta mai departe, pe urmatorul nivel al ierarhiei (in gen, memoria principala, abia apoi pe disc- in mem secundara, etc),

Odata ce informatia vizata este localizata:
Informatia + un numar de elemente apropiate ei (un bloc) sunt aduse (FETCH) in memoria cache

Ce spune *principiului localitatii* : Într-un interval de timp dat, referințele la memorie tind să se restrângă în zone locale ale memoriei (odata ce un octet a fost accesat, este foarte posibil ca un element din apropiere sa fie accesat curand – exemplu: primul element din cadrul unui sir ce se doreste a fi folosit intr-o problema)

3 tipuri de localitate:
Localitatea spațială: acesarile tind sa fie grupate in spatiul de memorie
Un program utilizează date şi instrucțiuni ale căror adrese sunt apropiate unele de altele în spațiul de adrese
Exemple: Referințele la elementele unui tablou, Citirea secvențială a instrucțiunilor din memorie
Localitatea temporală: cele recente (dpdv al accesului) tind sa fie accesate din nou
Datele sau instrucțiunile referite recent au o probabilitate ridicată de a fi referite în viitorul apropiat
Exemplu: buclă iterativă
Localitatea secvențială : Majoritatea instrucțiunilor sunt executate într-o ordine secvențială
Excepții: salturi sau apeluri de proceduri

Accesarea datelor din memorie



Transferurile de date au loc în *blocuri* (pagini sau linii) cu dimensiuni fixe

Blocurile sunt transferate numai între 2 nivele adiacente la un moment dat

- Definitii generalizate privind conceptul de memorie :
  - un succes = cand datele cautate (la un anumit nivel de memorie) sunt gasite
  - un esec = cand datele cautate (la un anumit nivel de memorie) nu sunt gasite

  La procesarea unui succes - e implicat timpul necesar accesarii datelor la un nivel dat de memorie

  La procesarea unui esec–e implicat timpul pt inlocuirea blocului de memorie + timpul necesar livrarii datelor la CPU

Dacă datele cerute de procesor se află într-un bloc al nivelului superior => succes("hit")

Dacă datele nu se află în nivelul superior: => eşec ("miss")

Rata de succes =procentul de cautari incheiate cu un succes

= raportul între nr accesarilor la memorie pt care se obţine un succes şi nr total de accesari

Rata de eşec = procentul de cautari incheiate cu un esec $R_e = 1 - R_s$

= raportul între nr accesarilor la memorie pt care se obţine un eşec şi nr total de accesari

- **daca există** un bloc în memoria M_i

-> există si o copie a blocului în fiecare din nivelele inferioare M_{i+1}, …, M_n

- dacă **un bloc nu este găsit** în memoria M_i:

-> se transmite o cerere pt acest bloc la nivelele inferioare

- dacă **nu mai există spaţiu** în memoria M_i :

-> se înlocuieşte un bloc din M_i fol o **strategie de înlocuire** predefinită

Avantajul unui **sistem de memorie ierarhic**: în maj timpului informaţia e preluată din nivelul cel mai rapid M_1

Timpul de acces mediu al memoriei: apropiat de timpul de acces al nivelului superior

Costul mediu unitar al sistemului de memorie: apropiat de costul nivelului inferior

SRAM (Static RAM) - CACHE versus DRAM (Dynamic RAM sau "RAM")
Avantaj1: nu necesita refresh
Functionarea SRAM: un grup de 6 tranzistoare pt fiecare bit de stocare;
nu foloseste condensatori -> nu se pierde sarcina stocata
- atat timp cat exista alimentare cu energie electrica, SRAM "tine minte" ce a stocat
Avantaj2: e mult mai rapida, fiind capabila sa tina pasul cu procesoarele din sistemele actuale

Dezavantaje:
- Datorita proiectarii au densitate mai mica, ocupand un spatiu mai mare pt aceeasi capacitate
- mult mai scumpe ca pret per bit

Ex: un cip* DRAM stocheaza de aprox 60 ori mai multa informatie in aceeasi unitate vs un cip SRAM
- un cip DRAM ce stocheaza **4 Gb** (512 MB) are corespondent dpdv al dimensiunii fizice (ocupata) un cip SRAM ce poate stoca doar **72Mb** (9MB)

- proiectantii de sisteme au hotarat sa investeasca in memorie SRAM de dimensiune mai mica in sisteme, cu rol de **buffer** sau **tampon de mare viteza** intre UCP si memoria principala,
acest buffer putand lucra la viteze apropiate de cele ale UCP
avand rolul de a anticipa informatia dorita de UCP si
de a o stoca in avans cererii ei de catre UCP

p

- * chip

Istoric SRAM (Static RAM)

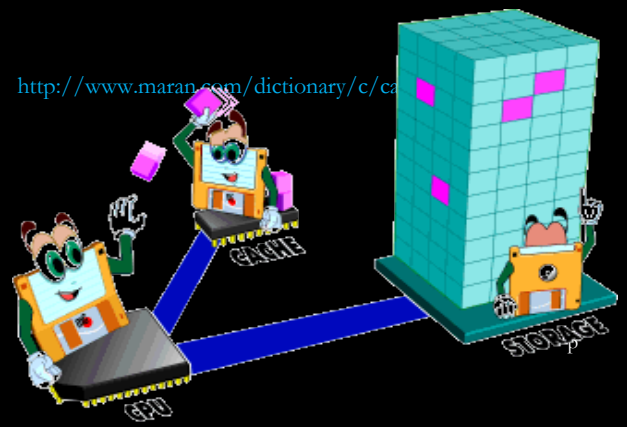
- pana in 1990 DRAM era limitata la aprox. 60 ns (adica 16 MHz = 1/60ns) durata unui ciclu
- procesoarele din acea vreme rulau la frecvente in jurul valorii 16MHz sau mai putin
  => memoria DRAM disponibila atunci tinea pasul cu procesorul si PB
  (deci nu era nevoie de un alt fel de memorie, ex. cache)

- din momentul cand procesoarele au inceput sa aiba viteze > 16MHz
  adica perioada 1986 – 1987 -> in SC : procesoare 386 la 16 MHz ÷ 20 MHz sau ↑
  => memoria DRAM nu a mai facut fata
  => a inceput sa apara memoria SRAM in sisteme ca "**memorie cache**" pe PB

=> ***memoria cache*** = **un buffer** format din mem SRAM ca "zona tampon"
intre UCP si memoria DRAM (care era mai lenta)

- ***controlerul de memorie cache*** anticipeaza ce inf din mem principala doreste CPU a fi accesate si le *preincarca in memoria cache*

- astfel, cand CPU face acces la ac inf, ele sunt livrate din memoria cache la viteze mai ridicate decat ar fi fost livrate din mem principala



Nivele SRAM (Static RAM)

- **Cache de nivel 1** (***cache L1***) in gen e integrata in capsula procesorului
  => cache L1 va rula la viteza miezului procesorului = cea mai rapida din sistem
  - a aparut de la procesoarele 486 ↑ (ca fiind integrata)

(chiar si procesoare 386 -386SL- au avut controller de cache integrat, insa memoria trebuia furnizata ca fiind inafara cipului)

- **Cache de nivel 2** (***cache L2***) la inceput era externa cipului procesorului
  - la procesoarele 386, 486 si prima generatie de Pentium
  -> Cache L2 rula la viteza busului extern procesorului – era instalata pe PB
- Procesoarele care au urmat lui Pentium Pro au inclus cache L2 ca parte a procesorului
  - Procesoarele aparute inainte de 1999 (si unele modele mai tarzii)
  au incorporat cache L2 in capsula procesorului, ca si cache L1,
  ->separate si externe miezului procesorului (dar in aceeasi capsula)

- **Cache de nivel 3** (***cache L3***) – a aparut la Pentium 4 Extreme Edition – 2003- cu 2MB cache L3 pe cip
  - totusi, variantele ce au urmat de Pentium 4 EE si Pentium EE
  au renuntat la cache L3 si au crescut capacitatea cache L2
- Cache L3 a revenit in procesoare in 2007 la AMD Phenom si in 2008 la Intel Core i7
  (ambele cu 4 miezuri (core) pe capsula)
  - cache L3 isi dovedeste utilitatea in acest tip de procesoare cu mai multe core,
  -> fiind partajata de toate core-urile

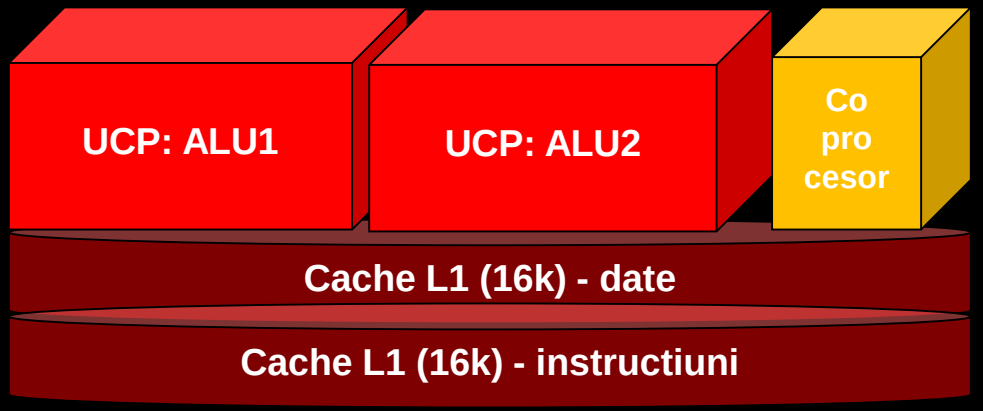
p

80486 DX

La inceput: de dimensiuni de 8Kocteti pt procesorul original 486DX

Azi: uzual 128Kocteti sau mai mult

Pentium



- Contine 3 unitati de executie a instructiunilor : 2 intregi (U-pipe,V-pipe) si una VM (FPU)

Arhitectura Intel Pentium

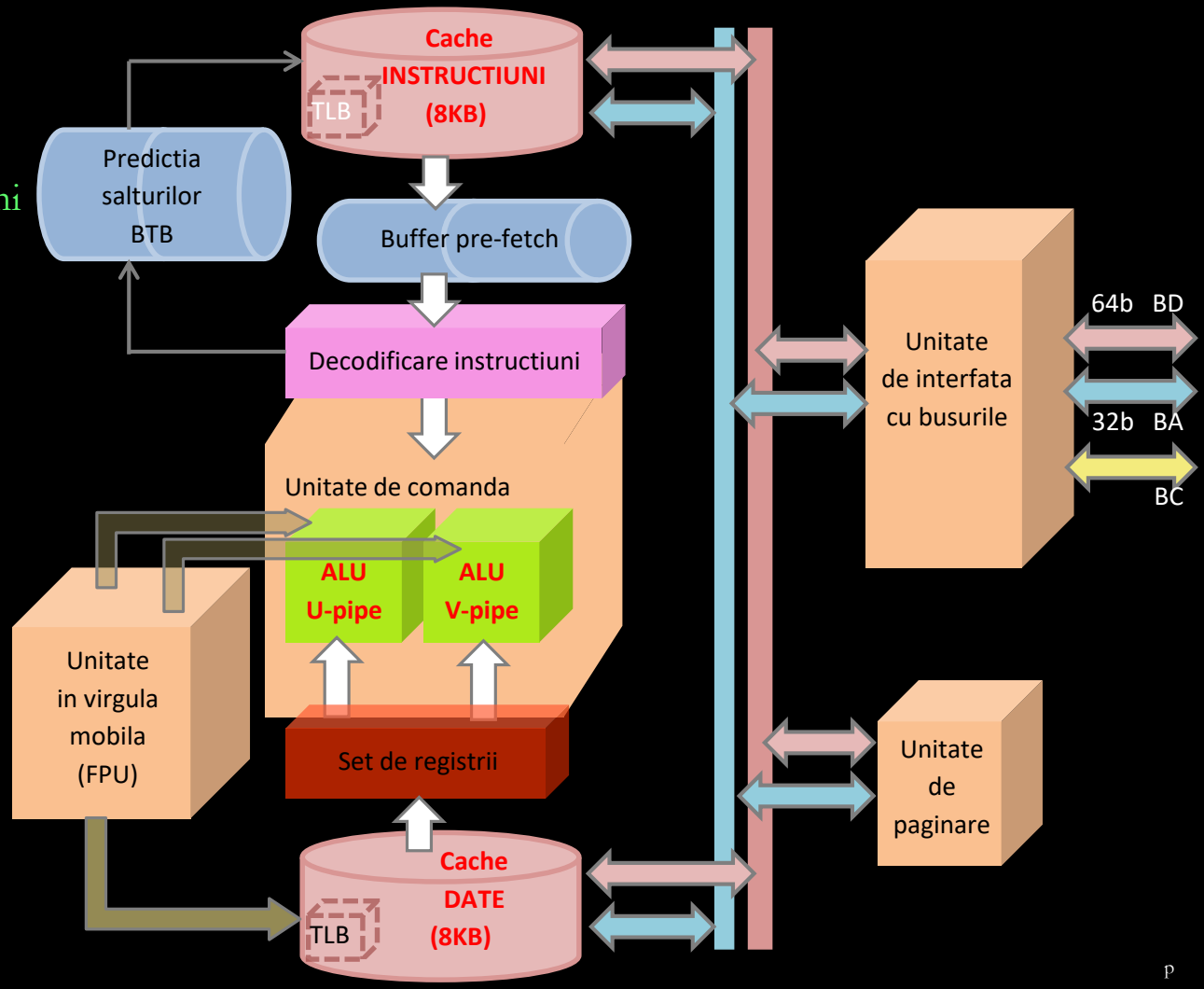
cache L1 intern: 8KB date
8KB instructiuni

cate un modul TLB
(Translation Lookaside Buffer)

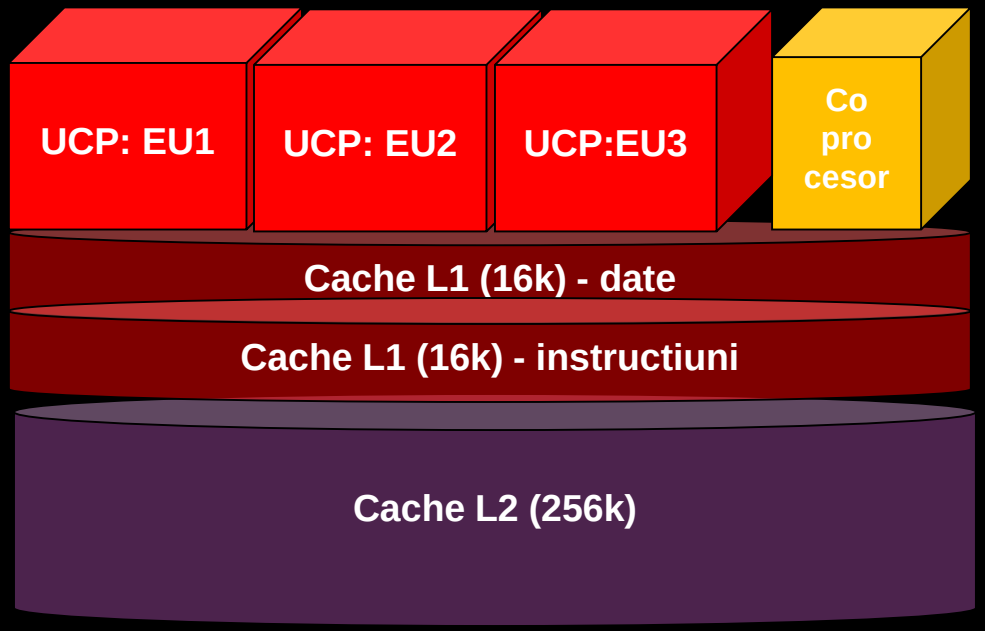
arhitectura superscalara:
pipeline pe 2 cai
U-pipe – cale primara
V-pipe – cale secundara

- blocul BTB
  (Branch Target Buffer)

- coprocesor matem intern FPU



Pentium Pro



Initial, cache L2: 256 kB, in prezent -> 6MB

Poate decodifica 3 instructiuni simultan

Exista un bloc RS (reservation station)

=> se pot executa simultan 4 instructiuni

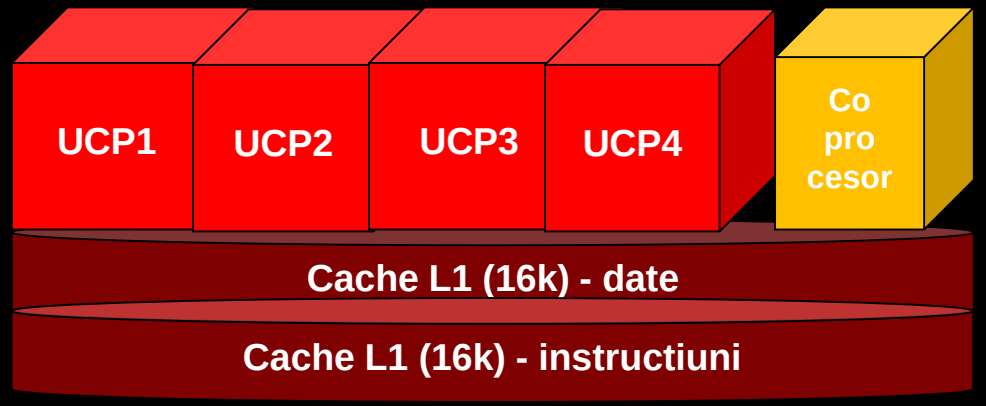
MEMORII RAM -> SRAM (6)

Pentium II,

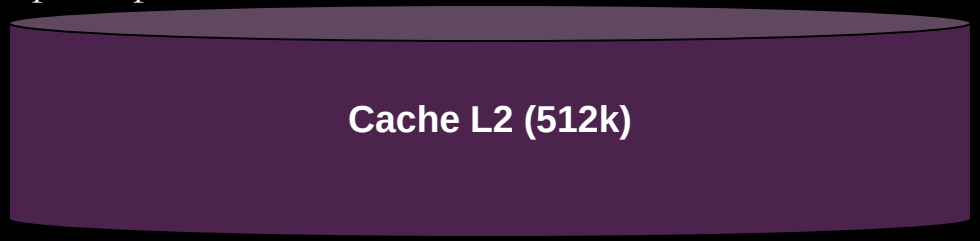
Pentium III,

(unele) Pentium 4,

(unele) Core 2



- Modificarea majora: cache L2 a fost scoasa din capsula procesorului



MEMORIA IN PC

MEMORII RAM -> SRAM (7)

ANCA APATEAN - AC - UTCN

Ordonate cronologic:

Procesor	Nr. miezuri Procesor	ceas	Dimens Reg	BD	Memorie maxima	Cache L1	Cache L2	Cache L3	Viteza cache L2/L3
8088	1	1x	16b	8b	1MB	-	-	-	-
8086	1	1x	16b	16b	1MB	-	-	-	-
286	1	1x	16b	16b	16MB	-	-	-	-
386 (SX,SL)	1	1x	32b	16b	16MB	0KB	-	-	Bus
386 (DX)	1	1x	32b	32b	4GB	-	-	-	
486	1	1x/2x	32b	32b	4GB	8KB	-	-	Bus
Pentium	1	1x/1,5x	32b	64b	4GB	2x8KB sau 2x16KB	-	-	Bus
Pentium Pro	1	2x	32b	64b	64GB	2x8KB	256KB, 512KB, 1MB	-	Miez
Pentium II	1	>2x	32b	64b	64GB	2x16KB	512KB	-	½ Miez
						2x16KB	256KB	-	Miez
Celeron	1	>2x	32b	64b	64GB	2x16KB	0KB	-	Miez
Celeron III	1	>2x	32b	64b	64GB	2x16KB	128KB, 256KB	-	Miez
Pentium III	1	>2x	32b	64b	64GB	2x16KB	128KB, 256KB	-	½ Miez, Miez
Celeron 4	1	>2x	32b	64b	64GB	2x16KB	128KB	-	Miez
Pentium 4,4A,4E	1	>2x	32b	64b	64GB	>16KB	256KB, 512KB, 1MB	-	Miez
Pentium 4 EE	1	>2x	32b	64b	64GB	>16KB	512KB	2MB	Miez
Celeron D	1	>2x	64b	64b	1TB	>16KB	256KB	-	Miez
Pentium D	2	>2x	64b	64b	1TB	>32KB	1MB sau 2MB pe miez	-	Miez
Pentium M	1	>2x	32b	64b	64GB	>32KB	1MB sau 2MB	-	Miez

MEMORII RAM -> SRAM (8)

ANCA APATEAN - AC - UTCN

Ordonate cronologic (continuare):

Procesor	Nr. miezuri Procesor	ceas	Dimens Reg	BD	Memorie maxima	Cache L1	Cache L2	Cache L3	Viteza cache L2/L3
Core Duo	2	>2x	32b	64b	64GB	>64KB	1MB pe miez	-	Miez
Core 2 Duo	2	>2x	64b	64b	1TB	>64KB	2-4MB sau 3-6MB pe miez	-	Miez
Core 2 Quad	4	>2x	64b	64b	1TB	>64KB	4MBsau 2-6 MB pe miez	-	Miez
Core i7	4	>2x	64b	64b	24GB	>64KB	256kB pe miez	8MB	Miez
Core /i5/i7	4	>2x	64b	64b	16GB	>64KB	256kB pe miez	8MB	Miez
Core i3/i5	2	>2x	64b	64b	16GB	>64KB	256kB pe miez	4MB	Miez
Core i7	6	>2x	64b	64b	24GB	>64KB	256kB pe miez	12MB	Miez
Core i7	4	>2x	64b	64b	8-32GB	>64KB	256kB pe miez	8MB	Miez
Core i5	4	>2x	64b	64b	32GB	>64KB	256kB pe miez	6MB	Miez
Core i3/i5	2	>2x	64b	64b	8-32GB	>64KB	256kB pe miez	3MB	Miez

Memoria cache

Memoria cache

Ce este: o memorie SRAM de viteza ridicata

- **scop:** cresterea vitezei de accesare a informatiei
- **cum:** stocheaza cele mai recent/frecvent utilizate info cat mai aproape de CPU

(intr-o memorie cu timp de acces t_{acces} scazut) IN LOC SA O STOCHEZE IN MEM PRINCIPALA

Desi memoria cache are *capacitate mai mica* decat mem principala, dar are *timp de acces mai mic* (o fractiune din timpul meme principale)

Mem principala e accesata prin adresa, in timp ce mem cache e *accesata prin continut* (in mod tipic) => "*memorie adresabila prin continut*"

=> nu e de dorit sa existe o singura memorie cache de capacitate mare=greu de cautat informatia in ea (ca si cum s-ar transforma intr-un "mini-RAM")

Mai multe blocuri ale memoriei cache pot fi mapate in memoria cache,

ele fiind diferentiate prin campul "tag" (marcaj)

Proiectarea unei memorii cache trebuie să răspundă la întrebări de tipul:

1. Care este dimensiunea optimă a unei linii de cache?
2. Cum se regăseşte informaţia conţinută în cache?
3. Care linie se înlocuieşte în cazul în care la un transfer memoria cache este plină?

Memoria cache

-Informatia este pastrata intr-o mem de date sub forma de pagini de dimens reduse = blocuri sau linii

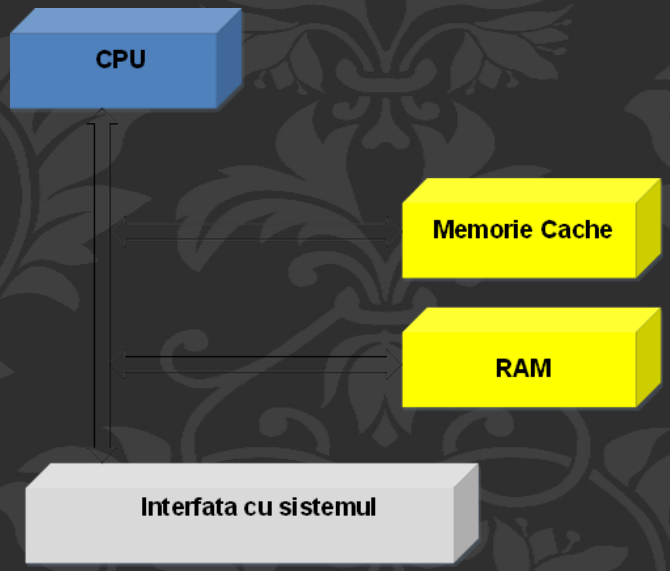
-Cont mem de date=copia unui set de blocuri ale mem princ

-Fiecare "bloc" al memoriei cache e marcat (cu un tag) *cu adresa de bloc*

-**Cu ajutorul tag-ului: memoria cache poate identifica partea spatiului de mem princ. careia ii apartine blocul din slotul resp. din cache**

-colectia marcajelor – se pastreaza intr-o mem separate, numita "de marcaje" (director)

-=> pt imbunatatirea performantei prin fol memoriei cache, trebuie ca: timpul necesar testarii marcajelor+timpul de acces la memoria cache sa fie < timpul de acces la memoria principala



Strategii de înlocuire

Dacă nu mai există spaţiu în memoria cache,
trebuie să se înlocuiască un bloc dintr-unul din sloturile cache-ului cu un **bloc nou**

Înlocuire aleatoare
Alege un bloc în mod aleator şi înlocuieşte blocul cu data nou accesată
Avantaj: implementare simplă
Dezavantaj: blocurile cu probabilitatea maximă de a fi utilizate din nou au aceeaşi şansă de a fi eliminate ca şi blocurile care nu vor fi utilizate din nou

Cel mai puţin frecvent utilizat (LFU – *Least Frequently Used*)
Pentru fiecare bloc, se păstrează un contor al numărului total de utilizări
Avantaj: un bloc utilizat frecvent are o probabilitate mai mare de a rămâne în memoria cache
Dezavantaj: blocurile care au fost încărcate recent au o valoare mică a contorului

Cel mai puţin recent utilizat (LRU – *Least Recently Used*)
Un bloc care nu a fost utilizat pt o perioadă lungă de timp are o şansă mai redusă de a fi fol în viitorul apropiat
Se păstrează evidența acelor blocuri care au fost accesate cel mai recent → contor
Are performanța cea mai bună raportată la cost comparativ cu celelalte metode

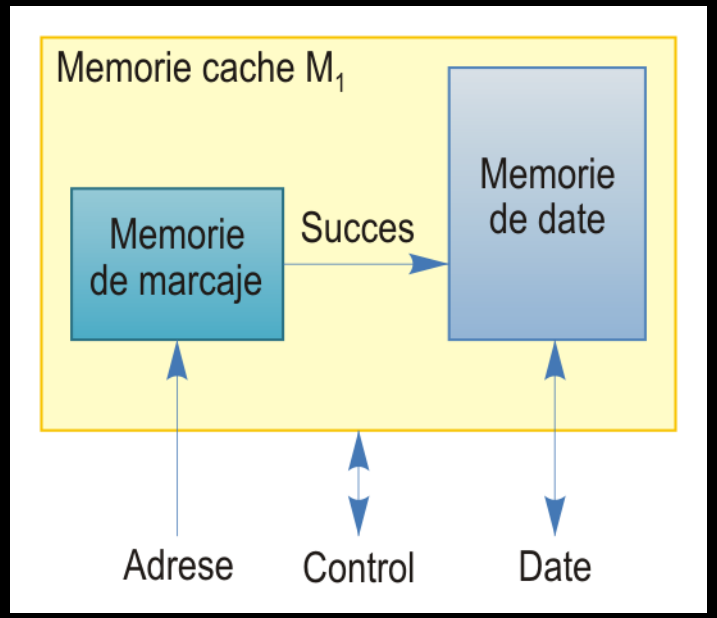
Organizarea memoriei cache

Cuvintele de memorie: păstrate într-o *memorie de date*

Sunt grupate în pagini → blocuri sau linii

Fiecare bloc al memoriei cache este marcat cu adresa sa de bloc → marcaj ("*tag*")

Colecția adreselor de marcaj asignate momentan memoriei cache: păstrată într-o *memorie de marcaje*



Structura de bază a unei memorii cache [Baruch]

Funcționarea memoriei cache

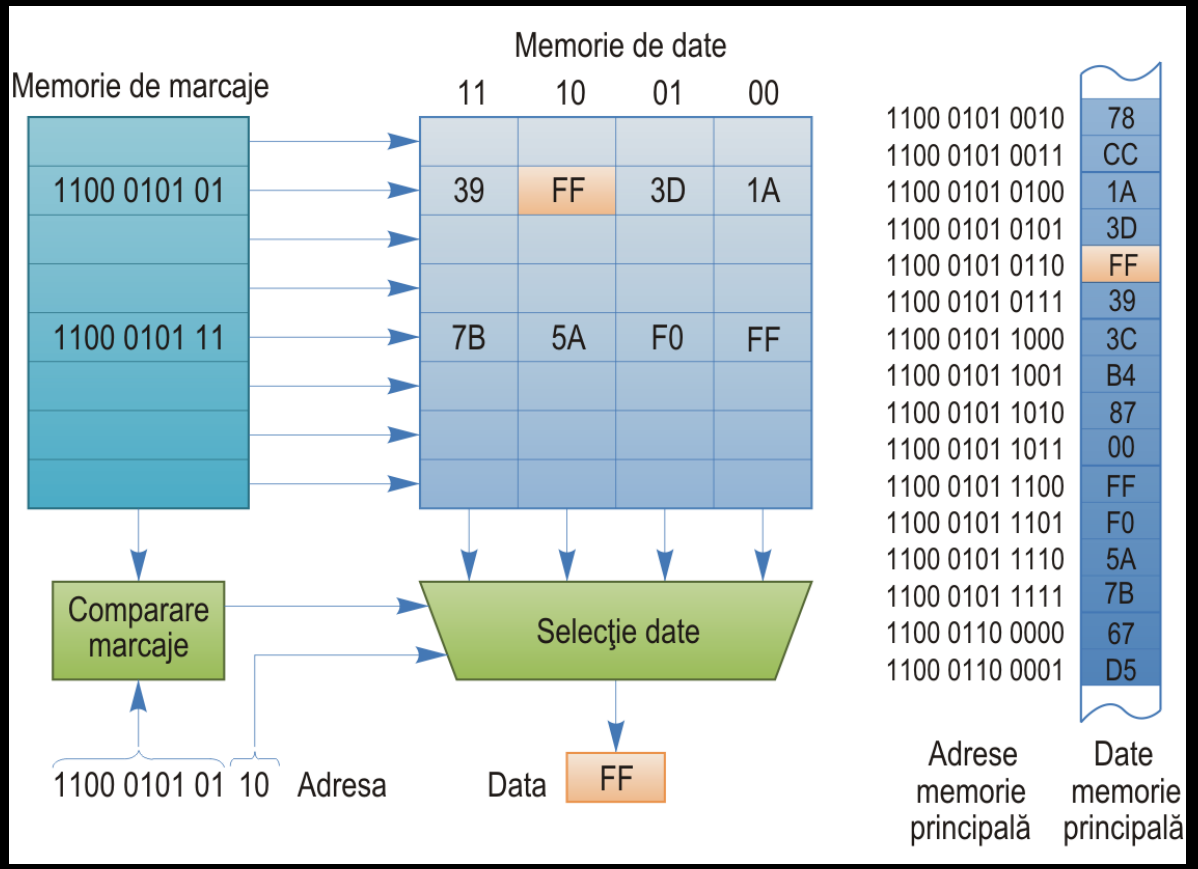
Atunci când UCP generează

o cerere de citire:

Cererea este trimisă la memoria cache

Dacă cuvântul **nu este găsit** în memoria cache: cuvântul solicitat este *furnizat de memoria principală*

Dacă cuvântul **este găsit** în memoria cache: *nu este necesar accesul la memoria principală*



Execuția unei operații de citire [Baruch]

FUNCȚIONAREA MEMORIEI CACHE (2)

Atunci când UCP generează o cerere de scriere:

Cererea este trimisă la memoria cache

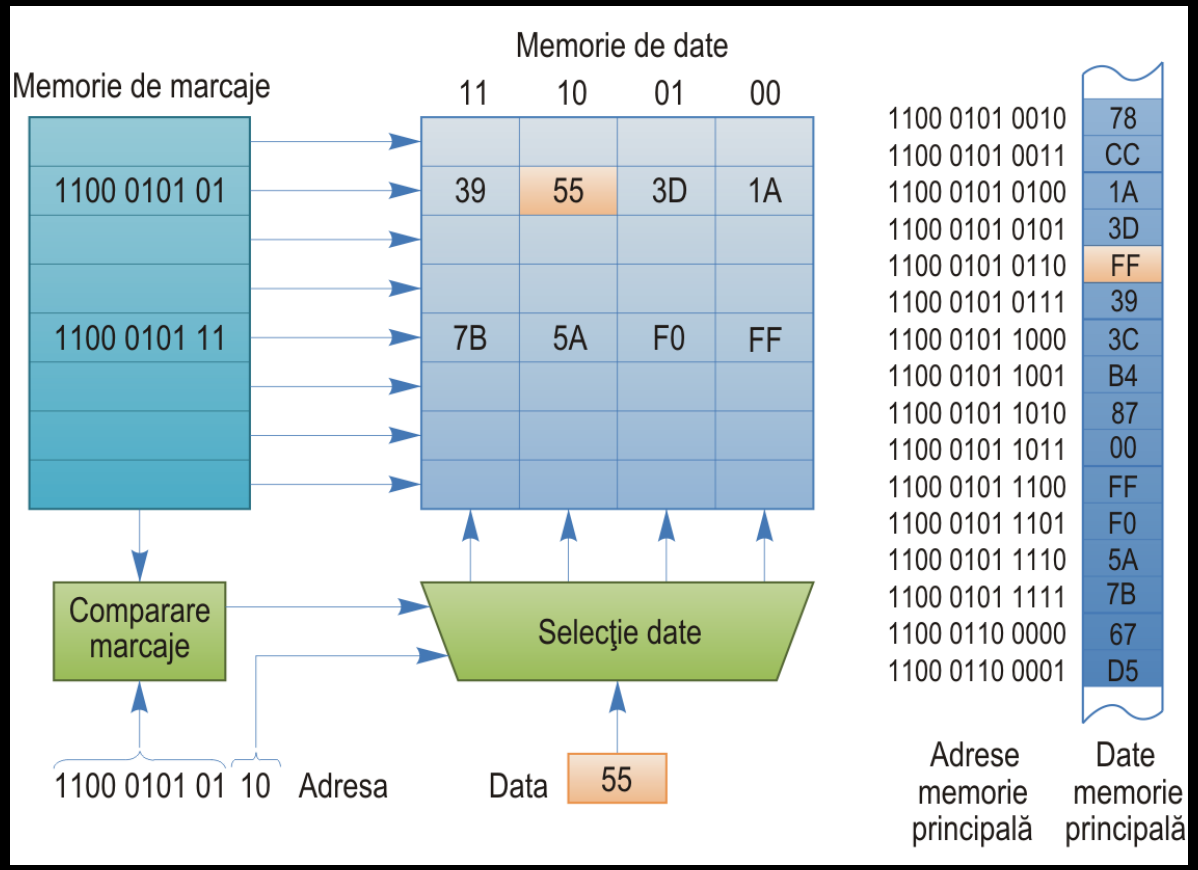
Dacă cuvântul **nu este găsit** în memoria cache:

(1) se încarcă o copie a blocului în care se află cuvântul din memoria principală în memoria cache

(2) Se execută o operație de scriere

Dacă cuvântul **este găsit** în memoria cache:

se execută o operație de scriere



Execuția unei operații de scriere [Baruch]

TEMA

Aratati cum se respecta principiul ierarhiei de memorie dpdv al capacitatii folosite pe propriul vostru PC

Indicatie: trebuie sa gasiti capacitatea fiecarui tip de memorie: HDD, “RAM”, cache L3, cache L2, cache L1

Principiul memoriei cache

Ce este: memoria cache = memorie rapidă, utilizată *temporar* pentru

păstrarea **unei porţiuni** a datelor şi instrucţiunilor în vederea utilizării lor imediate

Unde se afla: între memoria principală şi UCP

Ce conţine: copii ale unor blocuri din memoria principală

dacă un cuvânt solicitat de UCP se află în memoria cache,

nu va fi necesar accesul la memoria principală (care e mai lentă)

⇒ Îmbunătăţirea performanţei SC

⇒ Cresterea performantelor si prin

amplasarea memoriei cache în acelaşi circuit integrat ca şi procesorul

Proprietati ale memoriei cache:

dezavantaj - este mai costisitoare

→ capacitatea ei într-un sistem de calcul este limitată

avantaj – poseda proprietatea de localitate a referinţelor :

o mare parte din cererile de acces vor fi satisfăcute de memoria cache

Istoric: Nivele SRAM (Static RAM) – in timp au aparut mai multe nivele de mem cache in SC

- **Cache de nivel 1** (***cache L1***) in gen - integrata in capsula procesorului
  => cache L1 va rula la viteza miezului procesorului = cea mai rapida din sistem
  - a aparut de la procesoarele 486 ↑ (ca fiind integrata)

(chiar si procesoare 386 -386SL- au avut controller de cache integrat, insa memoria era inafara chipului)

- **Cache de nivel 2** (***cache L2***) la inceput era externa cipului procesorului
  - la procesoarele 386, 486 si prima generatie de Pentium
  -> Cache L2 rula la viteza busului extern procesorului – era instalata pe PB
- Procesoarele care au urmat lui Pentium Pro au inclus cache L2 ca parte a procesorului
  - Procesoarele aparute inainte de 1999 (si unele modele mai tarzii)
  au incorporat cache L2 in capsula procesorului, ca si cache L1,
  ->separate si externe miezului procesorului (dar in aceeasi capsula)

- **Cache de nivel 3** (***cache L3***) – a aparut la Pentium 4 Extreme Edition – 2003- cu 2MB cache L3 pe cip
  - totusi, variantele ce au urmat de Pentium 4 EE si Pentium EE
  au renuntat la cache L3 si au crescut capacitatea cache L2
- Cache L3 a revenit in procesoare in 2007 la AMD Phenom si in 2008 la Intel Core i7
  (ambele cu 4 miezuri (core) pe capsula)
  - cache L3 – utila - la procesoare cu mai multe core,
  -> fiind partajata de toate core-urile

Eficienta SRAM (Static RAM)

- Se exprima in "**rata de succes**" = nr de accesari soldate cu **succes** raportat la nr total de accesari
- ***Cache hit*** – controlerul de cache a reusit sa anticipeze corect informatia dorita de UCP
- ***Cache miss*** - controlerul de cache NU a reusit sa anticipeze corect informatia dorita de UCP

De fiecare data cand are loc un cache miss, procesorul trebuie sa astepte mai mult dupa date
(ciclurile memoriei principale au durata mai lunga decat cele ale procesorului)

Exemplu:
Un procesor la 3,6 GHz avand bus de memorie la 1,333 GHz si memorie cache integrata pe cip:
- UCP si memoria cache vor avea ciclu de 0,28 ns (=1/3,6GHz) in timp ce
memoria principala va avea ciclu de 0,75 ns (=1/1,333GHz), deci de aprox 3 ori mai lenta

⇒ UCP va trebui sa introduca **stari de asteptare** in care nu va face nimic = **timp pierdut**
-> doar va astepta sa-i fie livrata informatia din memorie

⇒ in timp, s-au propus **diverse metode de crestere a performantei memoriei cache**,
tocmai pentru a transforma cat mai multe cache miss in cache hit

⇒ Au aparut **diferite nivele (1,2,3) de memorie cache**

Eficienta SRAM (Static RAM) (cont)

Exemplu: Un sistem tipic Pentium la 233 MHZ cu un procesor la 233MHz (cu ciclu de 4,3ns), cu o PB la 66MHz (cu ciclu de 15ns) si memoria principala la 16MHz (cu ciclu de 60ns):

- procesor ce ruleaza la 233MHz -> durata pe ciclu UCP de 1/233MHz=4,3ns
- Procesorul e pe o PB la 66MHz -> durata pe ciclu a PB = 1/66MHz=15ns
- iar SC are memorie principala la 16MHz -> durata pe ciclu a mem =1/16MHz= 60ns

Cache L1 e integrata in UCP => va rula la viteza procesorului (a miezului) adica la cea mai ridicata viteza

(Procesoarele moderne au **multiplicator de ceas** -> ruleaza la o viteza multiplu de viteza PB in care sunt incluse)

- daca informatia dorita de procesor se afla in cache L1-> procesorul nu va trebui sa astepte deloc (cache hit)

–> procesorul lucreaza cu viteza de 233MHz

- daca informatia dorita de procesor NU se afla in cache L1 (cache miss), o va cauta in cache L2 (daca exista)
  - daca o va gasi -> procesorul va lucra cu viteza de 66MHz

Daca nici in cache L2 nu este, o va lua din memoria principala (sau cache L3 daca exista)

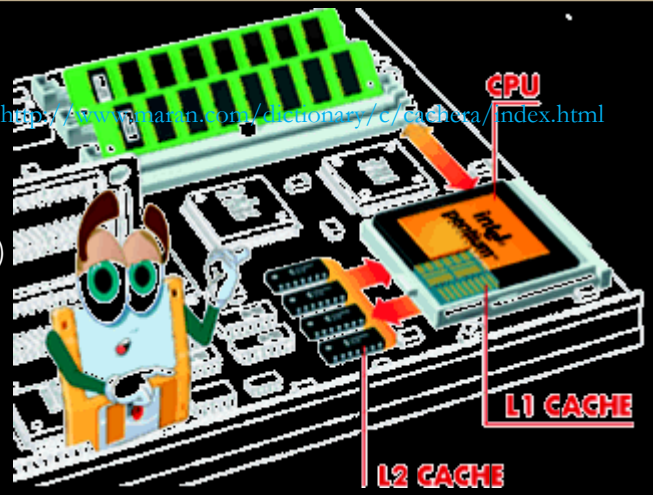
-> procesorul va incetini la viteza de 16MHz

Procesoarele moderne au integrat si cache L2 in procesor => la aceeasi viteza ca si procesorul si cache L1

Vizualizati exemplul – film RESTAURANT

Nivele SRAM (Static RAM) (cont)

- cache L1 de azi intotd se afla in capsula UCP
  => ruleaza la viteza miezului
  => 0 **stari de asteptare** (are aceeasi viteza cu cea a procesorului)



- ca si cache L1, cache L2 are **rata de succes** (**hit ratio) de ordinul 90%**:
  =>sistemul functioneaza in 90% din timp la viteza maxima (233MHz) luand datele din cache L1;
  10% din timp e incetinit pt a lua datele din cache L2
  – si de aici in 90% din timp gaseste informatia cautata in cache L2
  => in doar cele 10% cazuri ramase ajunge sa consulte memoria principala

=> sist va rula la viteza - procesorului in 90% din cazuri – 233Mhz
- placii de baza in 9% cazuri – 66MHz
- memoriei RAM in 1% cazuri – 16MHz

Daca se doreste imbunatatirea sistemului, dar se poate alege: ori cache L2 ori mem principala, ce ati alege ?

Nivele SRAM (Static RAM) (cont)

- la clasa Pentium – cache L2 – pe PB => ruleaza la viteza PB (66MHz->15nsec)
  - deci in loc sa incetineasca de la 233MHz pana la 16 MHZ (60nsec mem princip)
    poate incetini pana la 66 MHz (cache L2->15nsec)

- procesoarele moderne au integrat cache L2 in UCP => la aceeasi viteza cu a UCP

- Cache L3 – obisnuita in cele cu multicore (partajata de toate miezurile): Core i7, AMD, Phenom II, Fx

- Cache L1: apropiat de procesor, sau chiar in aceeasi capsula cu procesorul
- Cache L2: mai indepartata de procesor
- Cache L3: langa memoria principala, cea mai indepartata de procesor

- in aceasta analogie: procesorul era de 15 ori mai rapid ca mem principala;
- in prezent viteza memoriei a crescut de la 16MHz (60nsec) la 333MHz (3nsec);

desi viteza procesoarelor a crescut si ea, raportul viteza UCP/viteza mem principale este de max 7,5-10 ori

[Mueller]

p

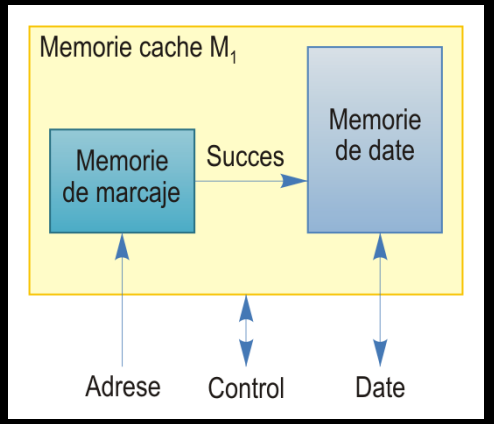
Organizarea memoriei cache

Cuvintele de memorie: păstrate într-o *memorie de date*

- grupate în pagini → *blocuri* sau *linii*

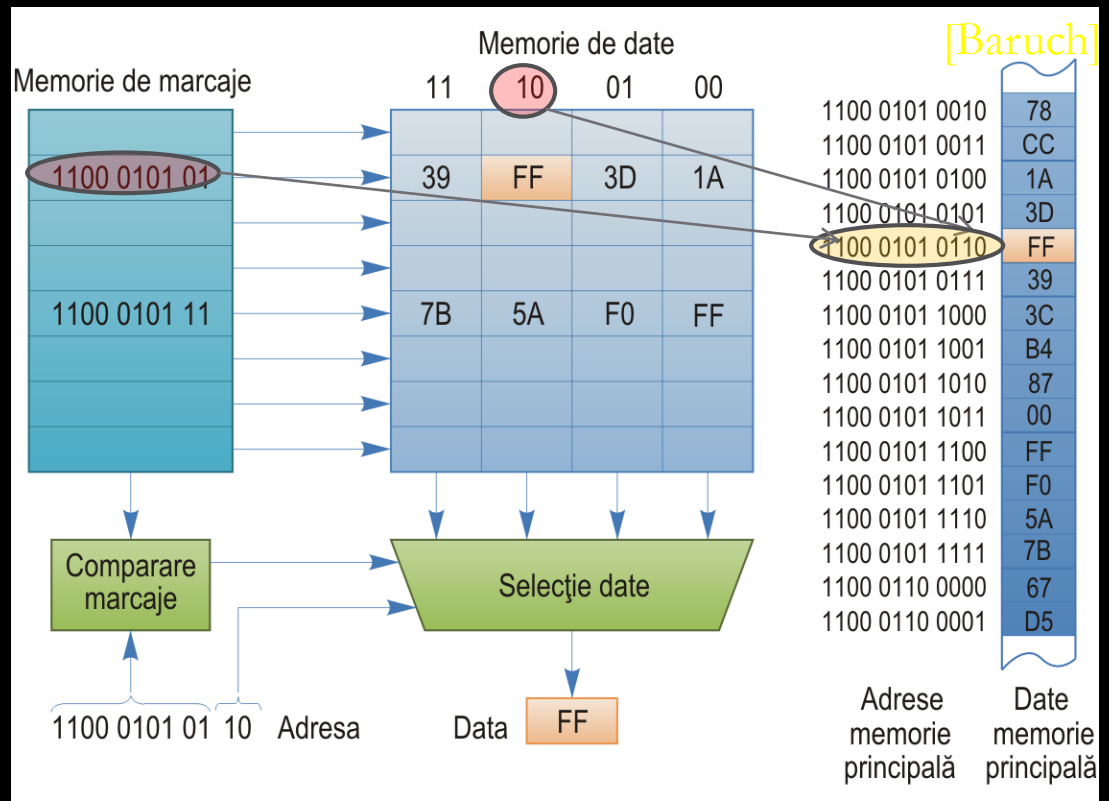
Fiecare bloc din memoria cache este marcat cu o adresa de bloc = marcaj (“*tag*”)

- toate marcajele asignate momentan memoriei cache: într-o *memorie de marcaje*



[Baruch]

Structura de bază a unei memorii cache



UCP generează o cerere de citire:
Cererea este trimisă la memoria cache
Dacă cuvântul nu este găsit în memoria cache:
cuvântul solicitat este furnizat de memoria principală
Dacă cuvântul este găsit în memoria cache:
nu este necesar accesul la memoria principală

Memorii SRAM -> memoria cache (6)

ANCA APATEAN - AC - UTCN

Maparea in memoria cache

Schema aleasa pentru a "aranja" blocurile luate din memoria principala in memoria cache afecteaza performanta si costul sistemului

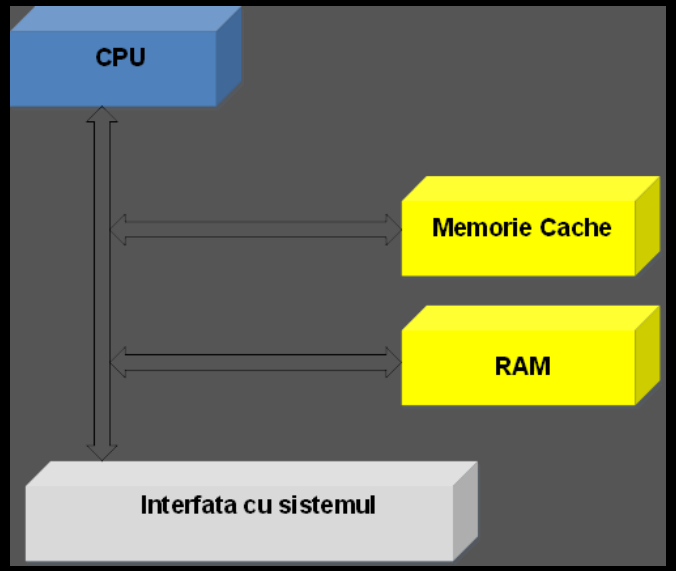
-mem principala nu e *fizic* patitionata astfel (asa o vede cache-ul)

- > toate tag-urile - pastrate intr-o memorie de marcaje

1. Mapare asociativa (associative mapping)

2. Memoria cache cu mapare directa (direct mapping)

3. Memoria cache set asociativa (set direct mapping)



Probleme:

- Cand memoria cache se umple, cum se decide care slot ar trebui eliberat ? (pt a aduce un bloc nou in mem cache)

Politici de inlocuire des utilizate:

least recently used (LRU)
first in first out (FIFO)
least frequently used (LFU)
random (aleator)
Des implemntata: LRU

Cand se incarca pt prima data un program in mem princip, mem cache e golita (curatata)

- exista un bit valid ce indica daca slotul pastreaza un bloc ce apartine programului ce se executa
- exista un bit ("dirty bit") ce pastreaza "urma" daca un bloc s-a modifcat in timp ce era in cache

Un slot ce e modificat tr scris inapoi in mem princip inainte ca slotul sa fie eliberat pt un alt bloc

- daca inf ceruta de procesor se afla in cache => "cache hit" (succes) , altfel "cache miss" (esec)

Cand se incarca prima data un program in memorie, bitii valizi sunt toti pusi pe 0

(1) I instr din program ce se executa va cauza astfel un esec
(nu exista instruct ale progr in cache in acel moment)

(2) blocul ce a cauzat aparitia unui esec, va fi localizat in mem princip si se va incarca in cache

1. Mapare asociativa:

Exemplu: un spatiu de 2^{32} cuvinte din memoria principala

va fi "divizat" in 2^{27} blocuri ("linii") de cate $2^5 = 32$ cuvinte per bloc (27+5=32)

memoria cache contine: 2^{14} sloturi in care se pot plasa blocuri ale mem principale

- la maparea asociativa: *un bloc se poate plasa in orice slot*

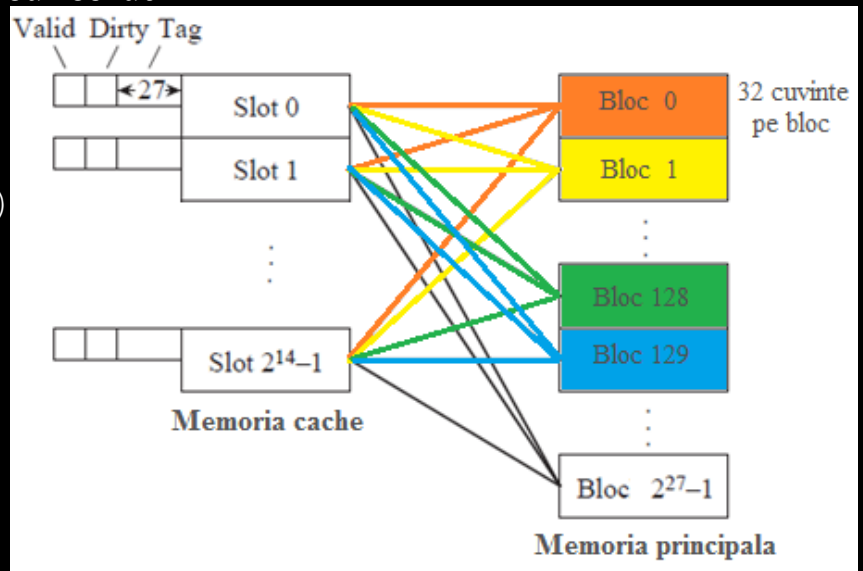
pt "urmarire" -> campul tag (pe 27 biti) va fi adaugat fiecarui slot

- partitionarea adresei:

Tag (marcaj)	Word (cuvant)
27 biti	5 biti

- cand se face referire la o adresa din mem princ, hdw-ul cache-ului intercepteaza referinta si cauta in memoria de marcaje cache sa verifice daca exista in cache blocul cerut

- pt fiec slot, daca bitul valid=1, atunci
campul marcaj al adresei referinta e comparat
cu ***campul marcaj*** al slotului (se cauta in paralel toate marcajele)
- daca se potriveste, atunci cuvantul e luat din pozitia din slot
(specificata de campul "word")
- daca nu e niciunul gasit, atunci
blocul din memoria principala ce contine cuvantul
e adus in cache si cuv de referinta e luat din cache
- se actualizeaza campurile marcaj, bitul valid si bitul dirty



2. Memoria cache cu mapare directa:

se restrictioneaza locul din cache unde poate fi plasat un bloc din mem principala

Exemplu: un spatiu de 2^{32} cuvinte din memoria principala
va fi "divizat" in 2^{27} blocuri ("linii") de cate $2^5 = 32$ cuvinte per bloc (27+5=32)
memoria cache contine: 2^{14} sloturi in care se pot plasa blocuri ale mem principale

- exista mai multe blocuri de mem principala $2^{27}/2^{14} = 2^{13}$ blocuri de mem princip pot fi mapate in fiecare slot
- pt a pastra urma (care din cele 2^{13} blocuri posibile este in slot) -campul marcaj in dom [0; 2^{13} -1] pt fiec slot

"cu mapare directa" pt ca fiecare slot de mem cache coresp la un set explicit de blocuri din mem principala

- ***fiecare bloc din mem principala poate fi mapat intr-un singur slot,***
dar fiecare slot poate fi ocupat de mai mult de 1 singur bloc (in timp)

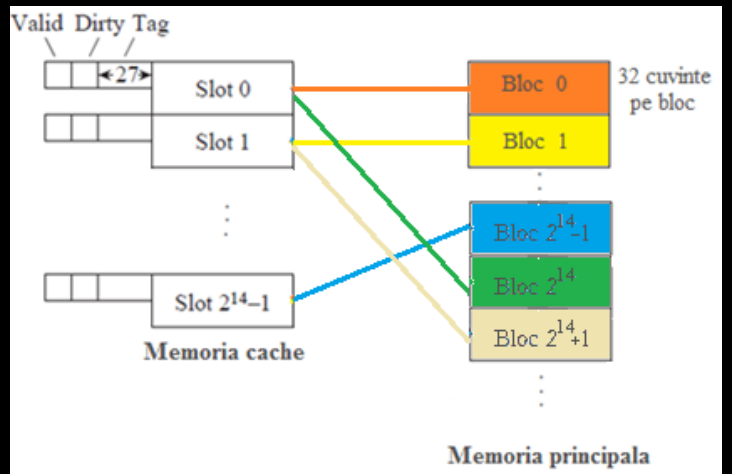
- partitionarea adresei:

Tag (marcaj)	Slot	Word (cuvant)
13 biti	14 biti	5 biti

Cand se face referire la o adresa din mem princ, campul slot identifica in care din cele 2^{14} sloturi se afla blocul(daca e in cache)

- daca bitul valid=1, atunci campul tag al adresei de referinta
e comparat cu campul tag al slotului;
daca sunt identice atunci
cuvantul e luat din pozitia specif de campul word din slot;
daca nu sunt identice, atunci
slotul e scris inapoi in mem principala dc bitul dirty e 1,
iar blocul coresp din mem princip e apoi citit in slot

Pt un program care abia intra in rulare, bitul valid=0
=> blocul va fi scris in slot; bitul valid al blocului va fi apoi pus in 1



3. Memoria cache set asociativa (set direct mapping): - se fol cel mai des

- combina primele 2: simplitatea metodei cu mapare directa cu flexibilitatea maparii asociative
- partea asociativa e limitata la cateva sloturi ce formeaza un set

Pt ex anterior, 2 blocuri formeaza un set => "two-way" set asociative cache - cache cu 2 cai set asociativa

Daca sunt 4 blocuri pe set => "four-way" set asociative cache - cache cu 4 cai set asociativa

Sunt 2^{14} sloturi in cache => $2^{14}/2 = 2^{13}$ seturi

Cand se mapeaza adresa unui set:

Tag 14 biti	set 13 biti	Word 5biti

Avantaje si dezavantaje ale schemelor de mapare pentru memoria cache

1. cu mapare asociativa:

- avantaje: orice bloc din memoria principala poate fi plasat in orice slot al cache-ului
- dezav: memoria de marcaje tr cautata in paralel: 27x 2^{14} biti are mem de marcaje => "mem asociativa" sau "adresabila prin continut" CAM (content addressable memory)
- cand se cauta in cache, campul marcaj al adresei referentiate trebuie comparat cu toate cele 2^{14} campuri tag din cache

2. cu mapare directa:

- e o schema relativ simplu de implem, mem de marcaje e mai mica (=13 x 2^{14}) decat in cazul 2.
- are dezav la accesul datelor de tip matrice (cuv distantate cu mai mult de 2^{19} cuvinte)

3. cu mapare set asociativa:

memoria de marcaje – usor mai mare ca la 2. :

$13x2^{14}$ si doar 2 marcaje trebuie cautate pt fiecare referinta de memorie

Problema 1. Gasiti unde in memoria cache cu schema de mapare asociativa se va mapa informatia din memoria principala de la locatia A135F016h .

Solutie: Va fi mapat:

Tag (marcaj)	Word (cuvant)
1010 0001 0011 0101 1111 0000 000	1 0110

=> 509AF80h si 16h

- daca cuvantul adresat e in cache, va fi gasit in cuvantul 16h al unui slot ce are marcajul 509AF80h

Problema 2. Gasiti unde in memoria cache cu mapare directa se va mapa informatia din memoria principala de la locatia A135F016h

Solutie: Un acces la locatia de memorie A135F016h e mapat in cache:

Tag (marcaj)	Slot	Word (cuvant)
1010 0001 0011 0	101 1111 0000 000	1 0110

- daca cuvantul adresat de in cache, acesta va fi in cuvantul 16h al slotului 2F80h cu marcajul 1426h

Problema 3. Gasiti unde in memoria cache cu mapare set asociativa se va mapa informatia din memoria principala de la locatia A135F016h.

Solutie: Un acces la locatia de memorie A135F016h e mapat in cache:

Tag (marcaj)	Set	Word (cuvant)
1010 0001 0011 010	1 1111 0000 000	1 0110

A135F016h => Tag=309Ah, Set=0F80h, Word=16h

Instrucțiunea CPUID

returnează și informații despre mărimea și caracteristicile memoriei cache interne:

când registrul EAX este initializat cu 2, instrucțiunea CPUID încarcă regiștrii EAX, EBX, ECX și EDX cu ***descriptori*** ce arată caracteristicile cache-ului procesorului.

Cei 8 biți "low" din registrul EAX (adica registrul AL) conțin o valoare care identifică de câte ori trebuie executată instrucțiunea CPUID pentru a obține o **imagine completă a cache-ului procesorului**.

Exemplu: pentru un procesor Pentium Pro se obtine valoarea 1 în registrul AL al lui EAX (c.m.p.s. 8 biti)

=> instrucț CPUID trebuie executată o singură dată (cu EAX=2) pt a obține o imagine completă a cache-ului procesorului

Restul registrului EAX și regiștrii EBX, ECX și EDX conțin pe cate 8 biți **descriptori** valizi (daca bitul 31=0).

Restul registrului EAX şi regiştrii EBX, ECX şi EDX conţin pe cate 8 biţi ***descriptori valizi*** (daca bitul 31=0).

Valoarea descriptor	Descriere cache
00h	Nul
01h	Instrucţiune TLB, pagini de 4k, asociativ 4 căi, 32 intrări
02h	Instrucţiune TLB, pagini de 4M, full asociativ, 2 intrări
03h	Date TLB, pagini de 4k, asociativ 4 căi, 64 intrări
04h	Date TLB, pagini de 4M, asociativ 4 căi, 8 intrări
06h	Cache Instructiuni, 8k, asociativ 4 căi, linii de lungime 32 octeţi
08h	Cache Instructiuni,16k, asociativ 4 căi, linii de lungime 32 octeţi
0Ah	Cache date, 8k, ă asociativ pe 2 căi, linii de lungime 32 octeţi
0Ch	Cache date, 16k, ă asociativ pe 4 căi, linii de lungime 32 octeţi
40h	Nu are cache de nivel L2 (la familia P6) sau L3 (la P4)
41h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 4 căi, 128k
42h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 4 căi, 256k
43h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 4 căi, 512k
44h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 4 căi, 1M
45h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 4 căi, 2M

Valoarea descriptor	Descriere cache
50h	TLB Instructiuni, pagini de 4Ko/2Mo/4Mo , total asoc, 64 intrari
51h	TLB Instructiuni, pagini de 4Ko/2Mo/4Mo, total asoc, 128 intrari
52h	TLB Instructiuni, pagini de 4Ko/2Mo/4Mo, total asoc, 256 intrari
5bh	TLB date, pagini de 4Ko/4MB, total asociativ, 64 intrari
5ch	TLB date, pagini de 4Ko/4MB, total asociativ, 128 intrari
5dh	TLB date, pagini de 4Ko/4MB, total asociativ, 256 intrari
66h	Cache date 8ko, sectorizat, asociativ pe 4 căi, linii de lungime 64o
70h	Cache Trace de instrucţiuni 12ko uOps, asociativ pe 4 căi
79h	Cache L2 128k, asociativ pe 8 căi, linii de lungime 64 octeţi
7ah	Cache L2 256k, asociativ pe 8 căi, linii de lungime 64 octeţi
7bh	Cache L2 512k, asociativ pe 8 căi, linii de lungime 64 octeţi
7ch	Cache L2 1M, asociativ pe 8 căi, linii de lungime 64 octeţi
82h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 8 căi, 512k
84h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 8 căi, 1M
85h	Cache unificat, linii de lungime 32 octeţi, asociativ pe 8 căi, 2M

Exemplu:

Pentru un procesor Pentium 4, dupa executia instructiunii cpuid se returneaza urmatoarele valori in registrii EAX, EBX, ECX, EDX:

EAX= 665B5001h -> b31=0 => descriptori valizi , AL=01 -> trebuie executata o sg data

EBX=00000000h

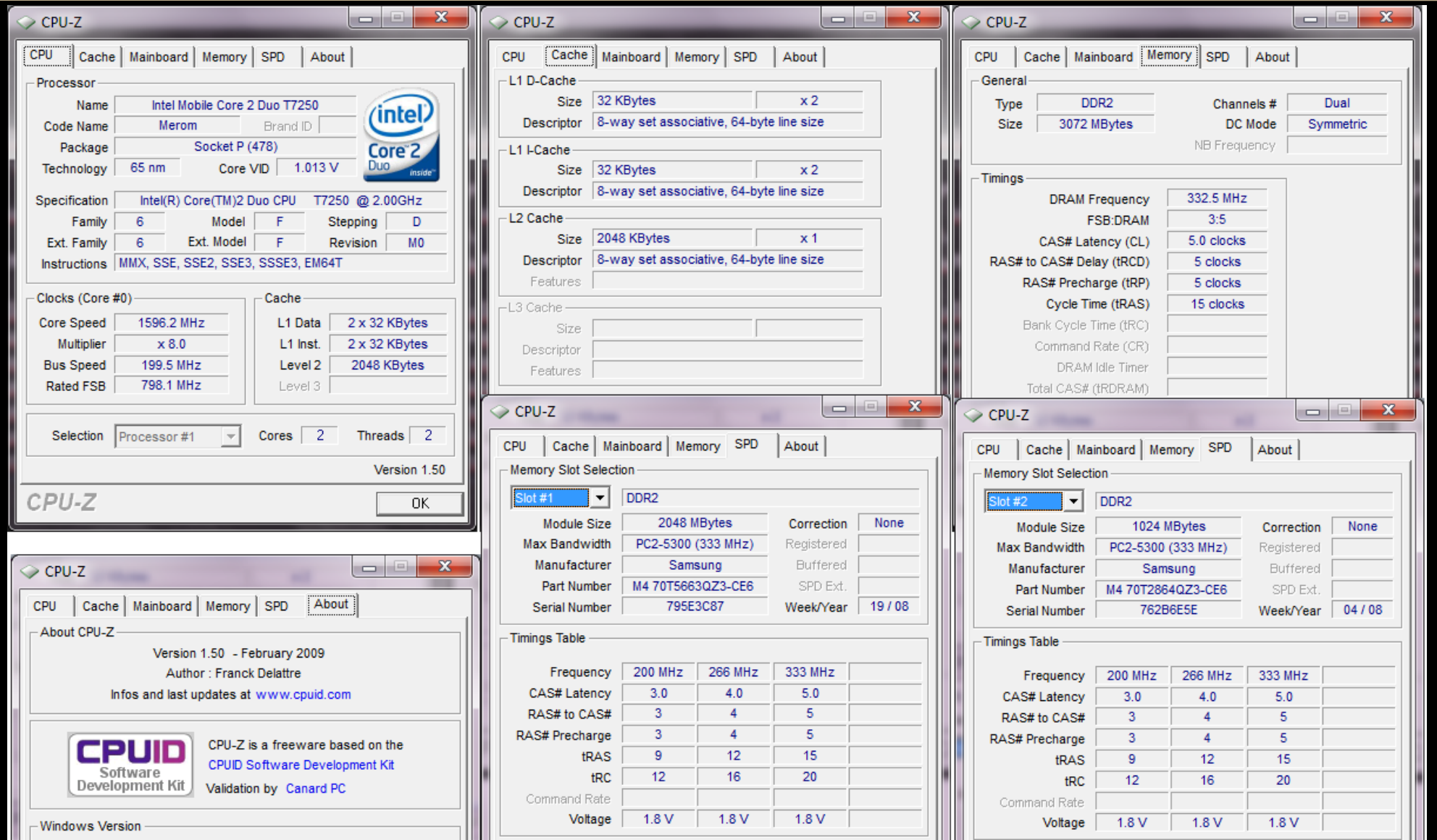
ECX=00000000h

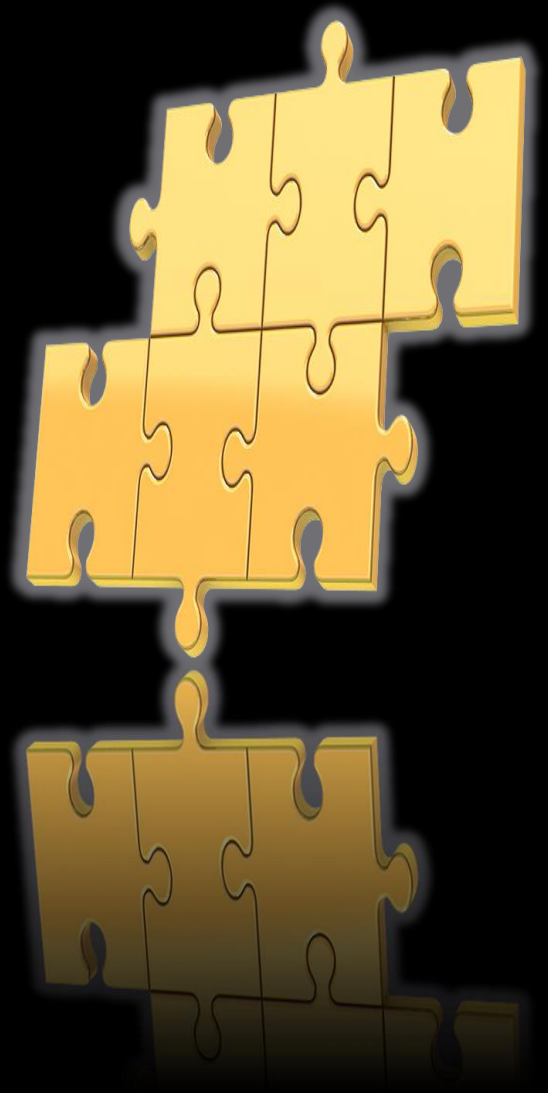
EDX=007A7040h

Memoria si interfata ei in PC

Identificarea caracteristicilor *memoriei* cu ajutorul aplicatiei *CPU-z*

ANCA APATEAN - AC - UTCN





TEME

Memoria si interfata ei in PC

TEMA

ANCA APATEAN - UTCN

Tema 1: urmariti problemele rezolvate de pe slide-urile cu creta

Tema2 : urmariti slide-ul cu CPU-z si identificati informatiile despre:

- memoria cache: ce nivele exista L1,L2,L3? De ce tip (date/instruct)? Cata capacitate? Ce caracteristici?

- memoria principala: Ce tip , este dual-triple-cvasi channel? realizeaza corectie? Alte caract? Cate sloturi de memorie? Ce fel +cata cantitate de mem in fiecare slot? Cat este banda?

Ce inseamna SPD? Care sunt timpii (in nr de cicluri de ceas) specificati si ce exprima acestia?

R: 2x32kB cache L1 date + 2x32kB cache L1 cache instruct + 2048k=2MB cache L2 , organizare 8-cai, set asociativa, dimens liniei: 64 byte

DDR2, dual-channel, slot#1: 2048MB, slot#2: 1024MB, de tip PC2-5300MBps, la 333MHz, fara corectie, nebuferata, ne-registered

Tema 3 : *Calculati rata de succes (hit ratio) si timpul de acces efectiv pentru un computer ce are o memorie cache cu mapare directa cu 4 sloturi de 16-cuvinte. Organizarea si legatura cache-ului cu memoria principala este ilustrata in figura. Timpul de acces al cache-ului este de 80 ns, iar timpul pentru transferarea unui bloc de date din memoria principala in memoria cache este de 2500 ns. Se presupune ca initial cache-ul este gol. Un program de test executa instructiunile ce se afla in memoria principala la locatiile 48-95, apoi face bucla de 10 ori intre locatiile 15-31. S-au inregistrat evenimentele pe masura executiei programului si s-au depus intr-un tabel.*