Hardver alapok

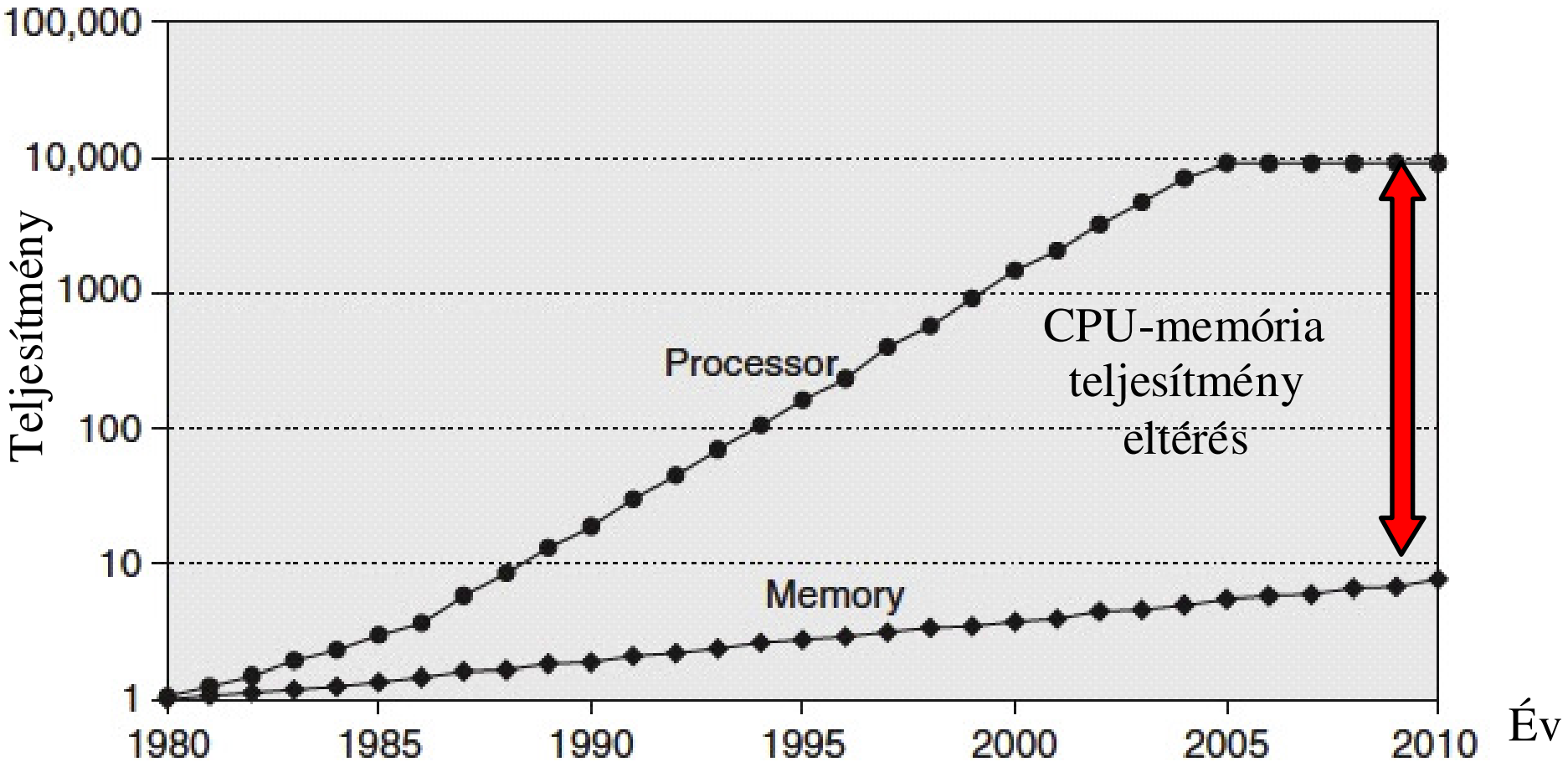
(VIIIBA01)

Memória gyorsító tárak

Pilászy György gpilaszy@iit.bme.hu

2018/19. tanév őszi félév

CPU – Memória teljesítmény



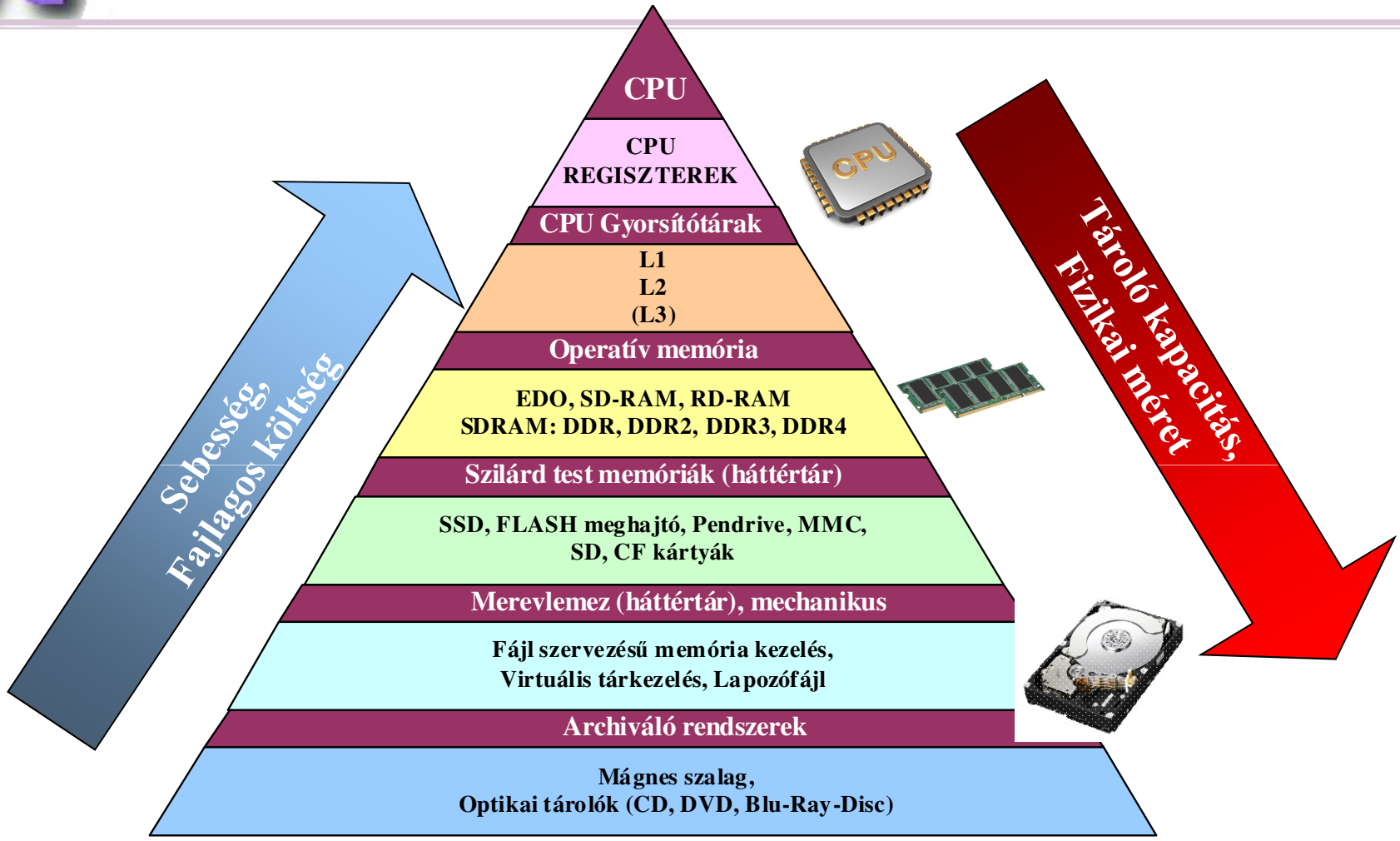
Memória – kapacitás – elérési idő

Elvárások (*követelmények*)

- nagy kapacitás
- nagy sebesség
- alacsony ár
- tartalom megőrzés (hordozhatóság)

Szint	Memória	Tipikus méret	Elérési idő [ns]	Sávszélesség [MB/sec]
1	Regiszter	< 2kB	0.15-0.3	100
2	Gyorsítótár	32kB-8MB	0.5-15	10 000-40 000
3	Operatív memória	< 512GB	30-200	5 000-20 000
4	háttértár	>1TB	5 000 000	50-500

Memória hierarchia



A nagykapacitású, olcsó memória lassú

Hierarchikus memória felépítés

- *A sebesség különbség nő → megoldást kell találni*

❑ Lokalitás elve

- *Térbeli*

  Ha a program egy adott helyen lévő információt használ, akkor valószínűleg a környezetében lévőt is használni fogja

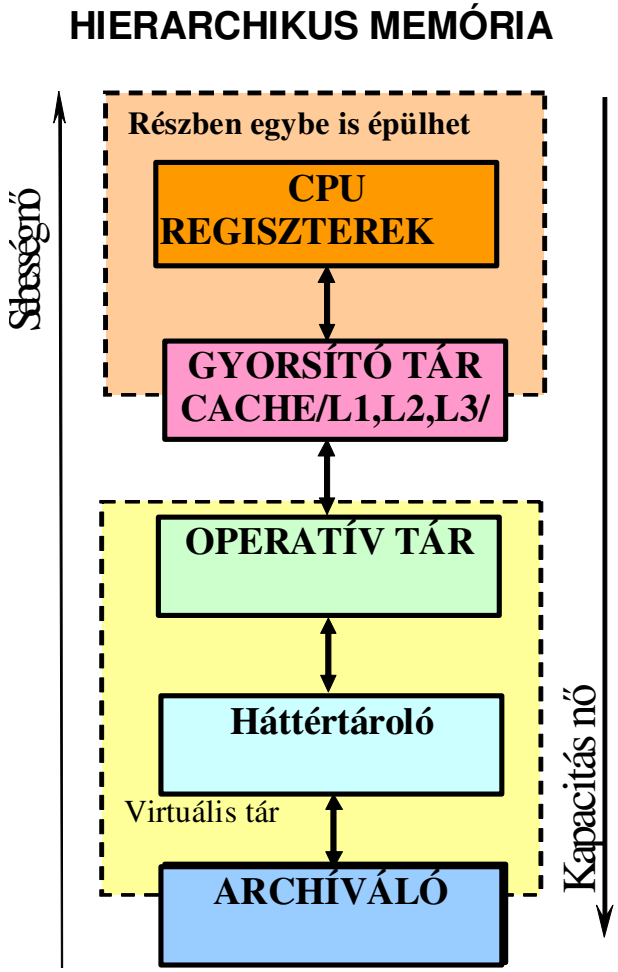
- *Időbeli*

  Ha egy adott információt használ a program, akkor valószínűleg a nem túl távoli időben is használni fogja

❑ Hierarchikus memória felépítés → gazdaságos realizálás,

- A használni kívánt információt és környezetét (*blokkját*) helyezzük át egy kisebb kapacitású, de *gyorsabb* memóriába → *válasz a sebesség igényére* → *Gyorsító tár (gyorstár) CACHE*
- Tekintsük memóriának a háttértárolót is, ahonnan blokkokban tölthetünk be információt az operatív memóriába → *válasz a kapacitás igényére* → *Virtuális Tárkezelés*

Hierarchikus memória felépítés

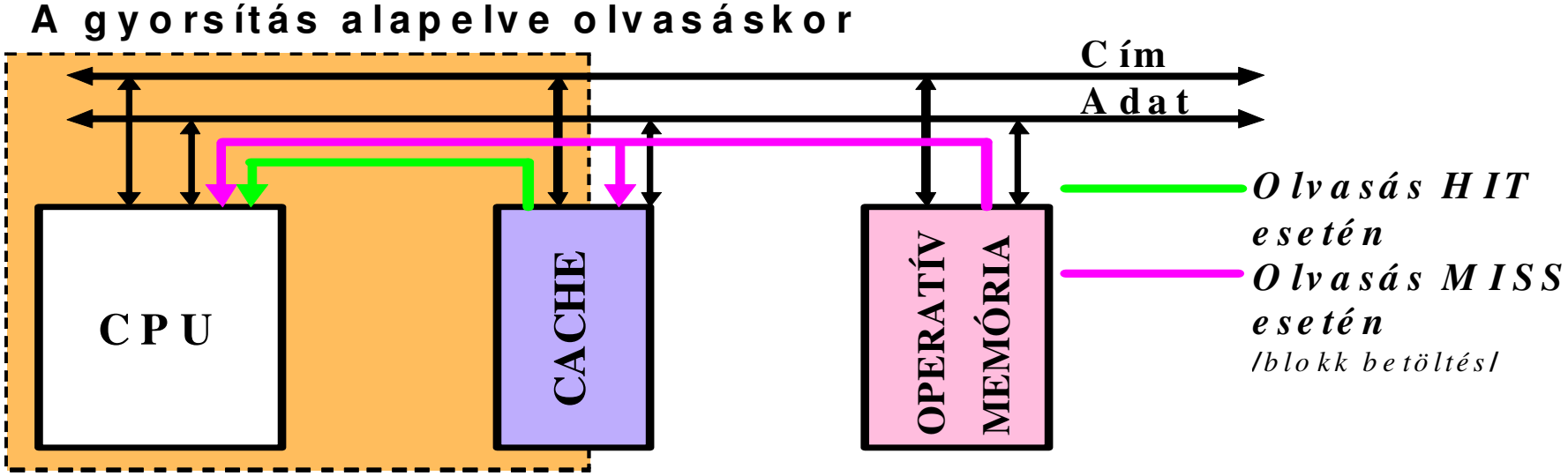


- ***Felsőbb szint kisebb, de gyorsabb***
- ***Tranziens állapottól eltekintve a felső szint részhalmaza az alsónak***
- ***Szintek között az információ csere blokkos***
- ***Hatékonyság a sikeres hozzáférés arányával mérhető***
- ***Sikeres hozzáférés, ha az információ a felső /gyorsabb/ szinten érhető el***
- ***Találati arány*** */hit rate, hit ratio/* ***Hr***

$$Hr = \frac{\text{sikeres hozzáférés}}{\text{összes hozzáférés}}$$

- ***Hibaarány*** */miss rate/* ***Mr=(1-Hr)***

Gyorsítótár - Cache



Nyereség: az effektív memória elérési idő csökkenése
(T_L: *látszólagos memória elérési idő)*

T_C *(Cache elérési idő)* T_{OM} *(operatív memória-OM elérési idő)* T_{MP} *(plusz időráfordítás hiánynál- miss penalty)* esetén, *a blokkáttöltés plusz idejét elhanyagolva*

$$T_L = H_r \cdot T_C + (1 - H_r) \cdot T_{OM} \qquad T_L = T_C + (M_r) \cdot T_{MP}$$

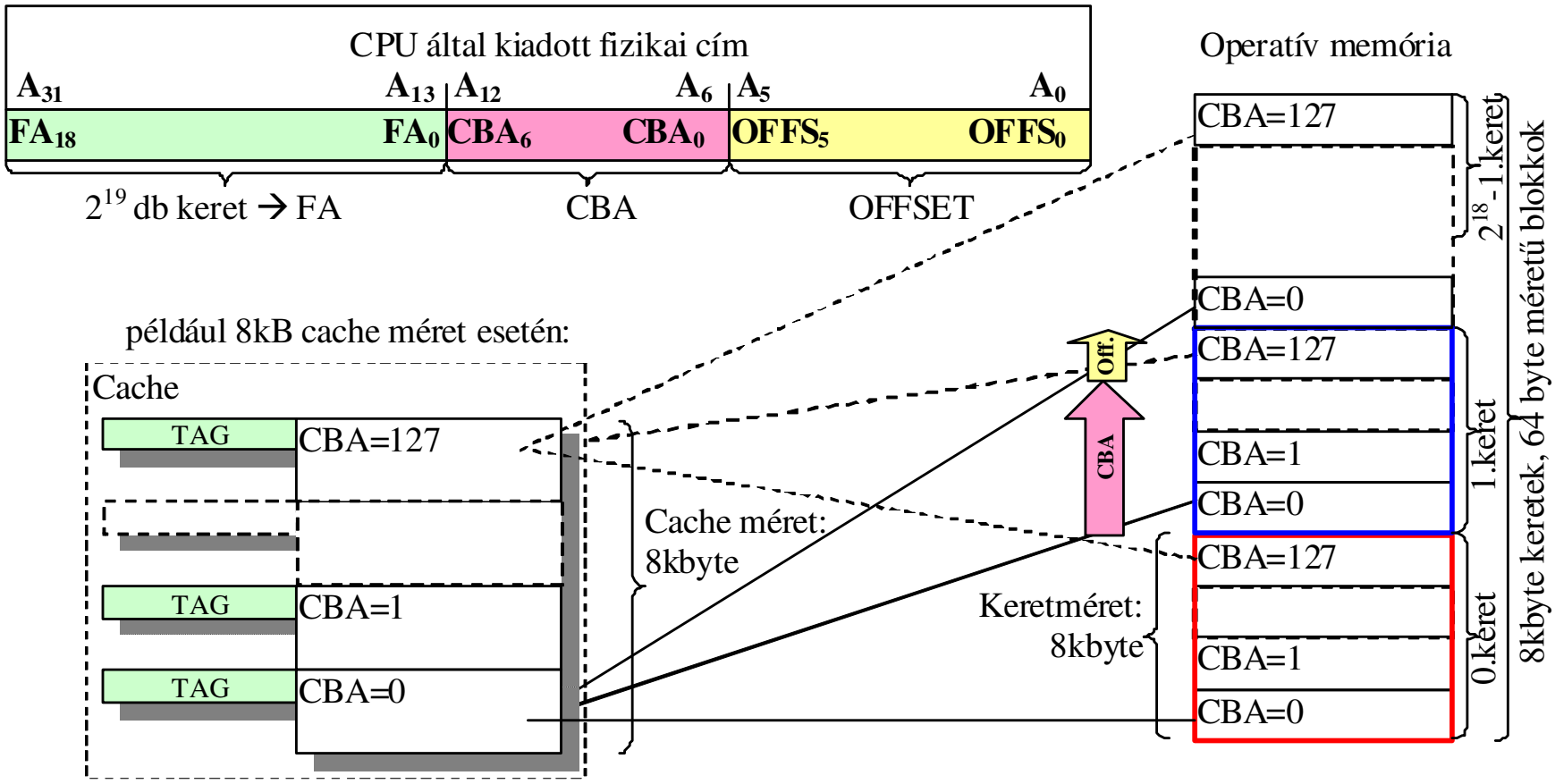
Pl.: Hr=90% Tc=5ns TOM=50ns TL=?

$$T_L = \frac{(90 \cdot 5) + 10 \cdot 50}{100} = 9{,}5ns$$

A CACHE szervezés kérdései

- Leképzési stratégia (*placement policy*)
  Az operatív memória egy adott blokkját a CACHE melyik blokkjába (sorába, line) tölti
  - *Blokk mérete: kicsi ←→ nagy*
  - *Hogyan ismeri fel a betöltött információt*
- Behozatali stratégia (*Fetch policy*)
  Mikor töltünk be egy blokkot
- Írási stratégia (*update policy*)
  Hogyan írjunk a CACHE-be, illetve az OM-be
- Blokkcsere stratégia (*block replacement policy*)
  Új blokk betöltésekor melyik blokk helyére töltünk be
- *Gyors blokkbetöltés megoldása*

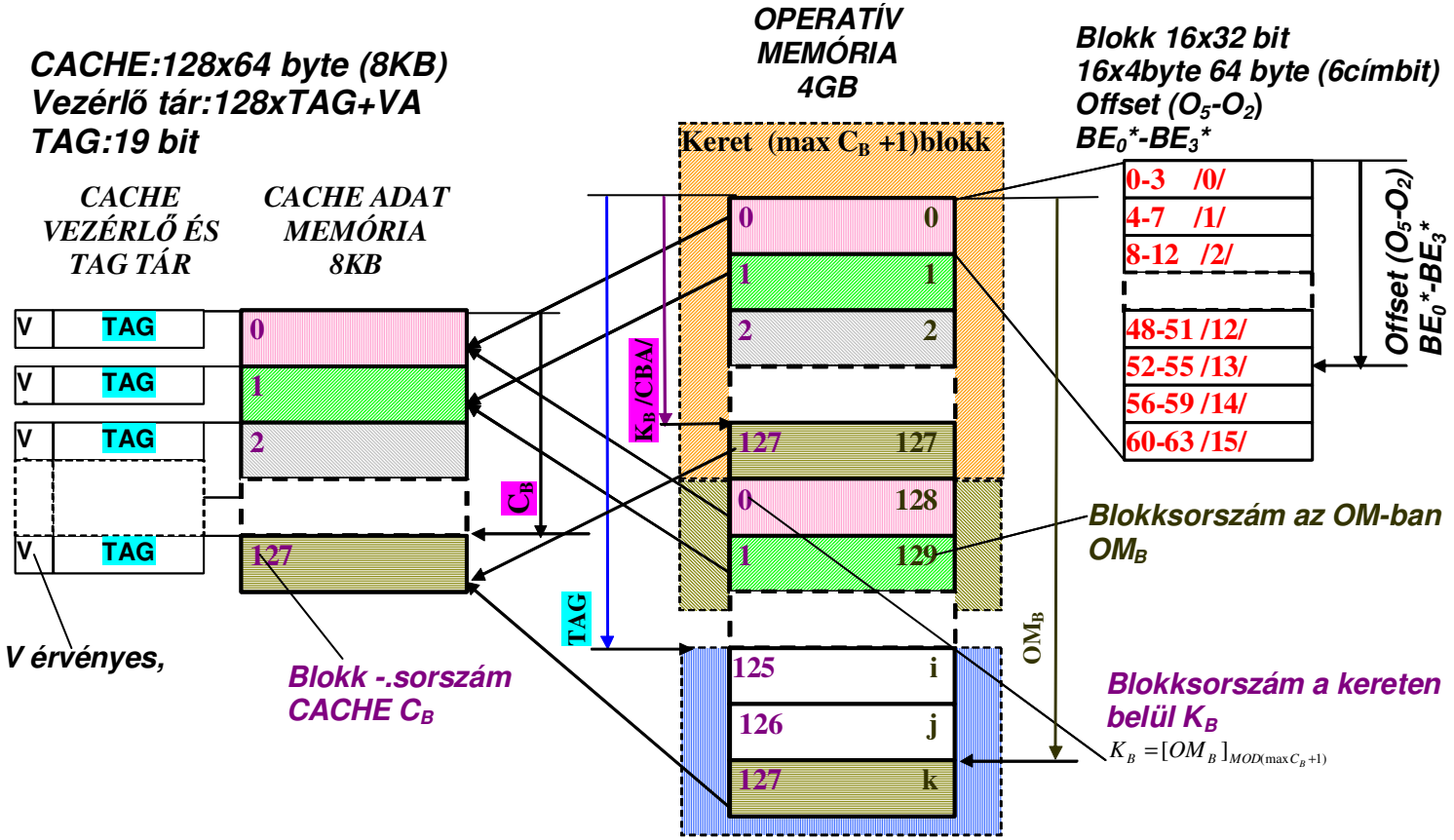
A leképezés elve



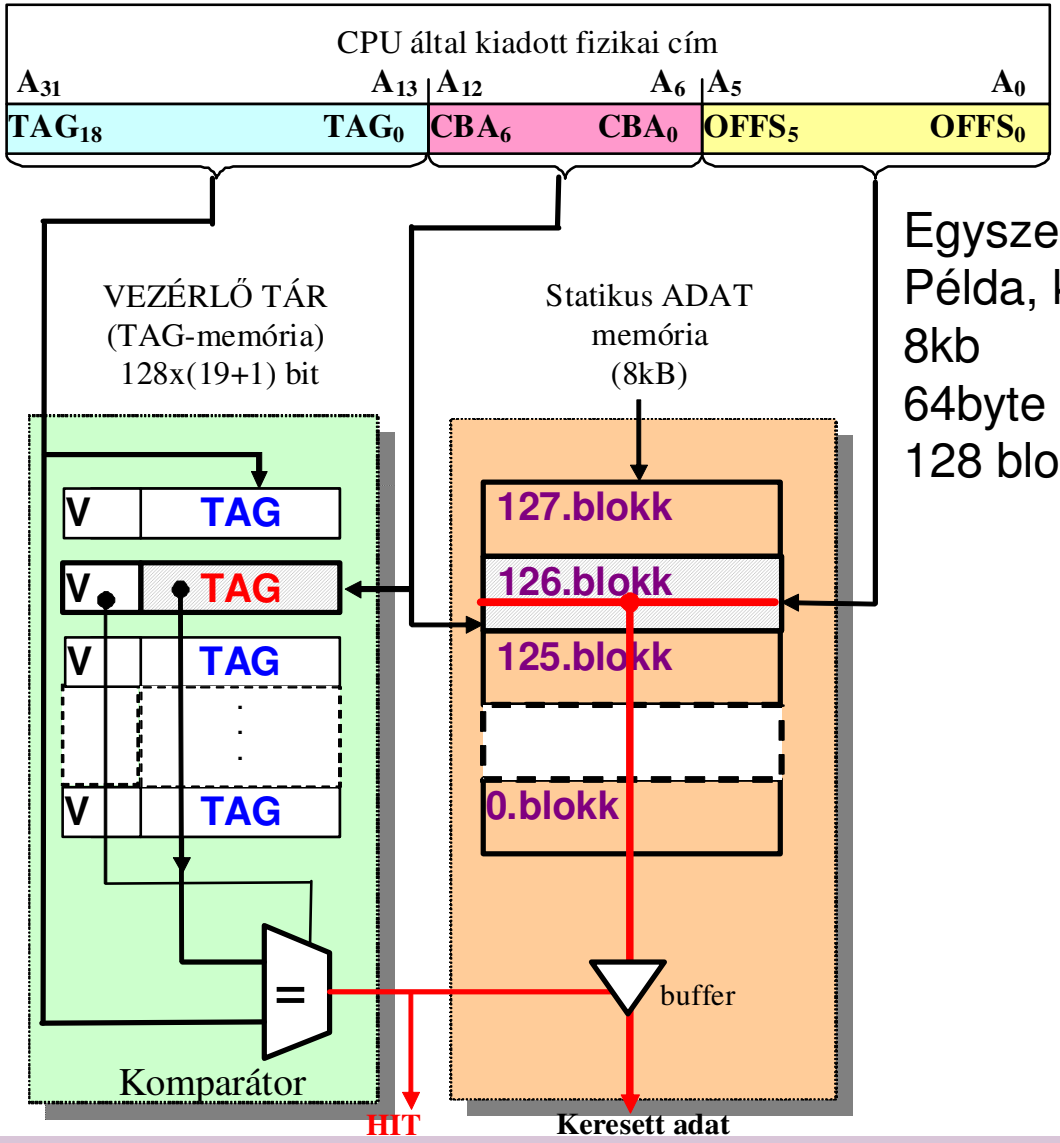
Közvetlen leképzés

Pl.:Operatív memória: 4Gbyte Cache: 128 blokk Blokk:64 byte

	A_{31} OPERATÍV MEMÓRIA A_0			4GB CÍM 32bit
	TAG_{18} TAG_0	CBA_6 CBA_0	$OFFS_5$ $OFFS_0$	128 blokk 7bit
V	A_{31} TAG A_{13}	A_{12} CBA A_6	A_5 OFFSET A_0	64 byte 6bit



Direkt leképezésű Cache blokkvázlata



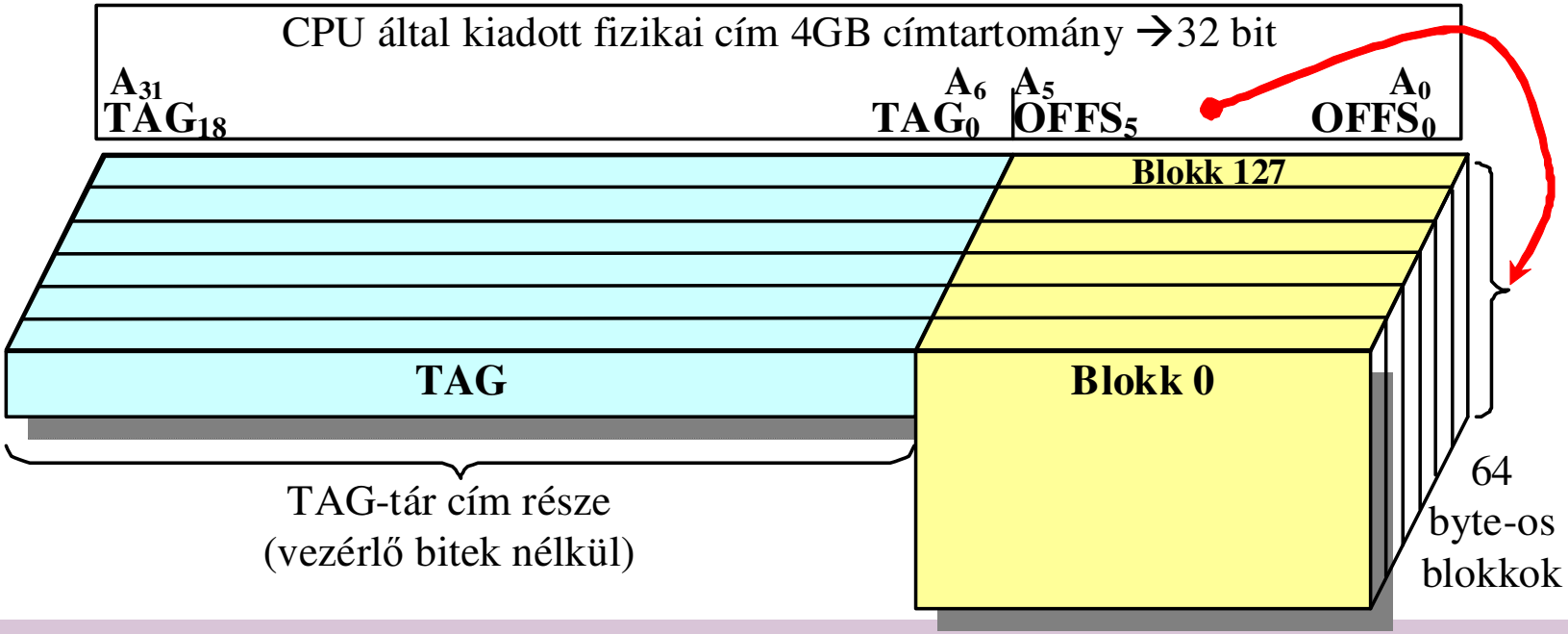
Egyszerűsített blokkvázlat
Példa, konkrét adatokkal:
8kb
64byte blokk méret
128 blokk

Direkt leképzésű cache

- Előnyök
  - Egyszerű hardver
  - Nem kell blokkcsere algoritmus
  - Gyors működés
  - Alacsony ár
- Hátrányok
  - HIT rate alacsony lehet
    (*ha a méret nő, jobb a HIT rate*)
  - Nem teljes körű a felhasználhatóság
- Asszociatív leképzéssel kombinálva
  - Legjobb kompromisszum
  - *leggyakoribb megoldás n-utas direkt leképzés*

Asszociatív leképzés (*associative mapping*)

- *Az adott keretsorszámú (K_B) blokk a CACHE bármely sorszámú (C_B) blokkjába helyezhető*
  - *Az Offset-en kívül minden címbit a TAG-be kerül*
  - *Találat keresésekor az összes TAG-tartalmát vizsgálni kell (lassú, méret)*
  - *A realizálás nehézségei miatt legfeljebb csak kis méretben alkalmazzák*

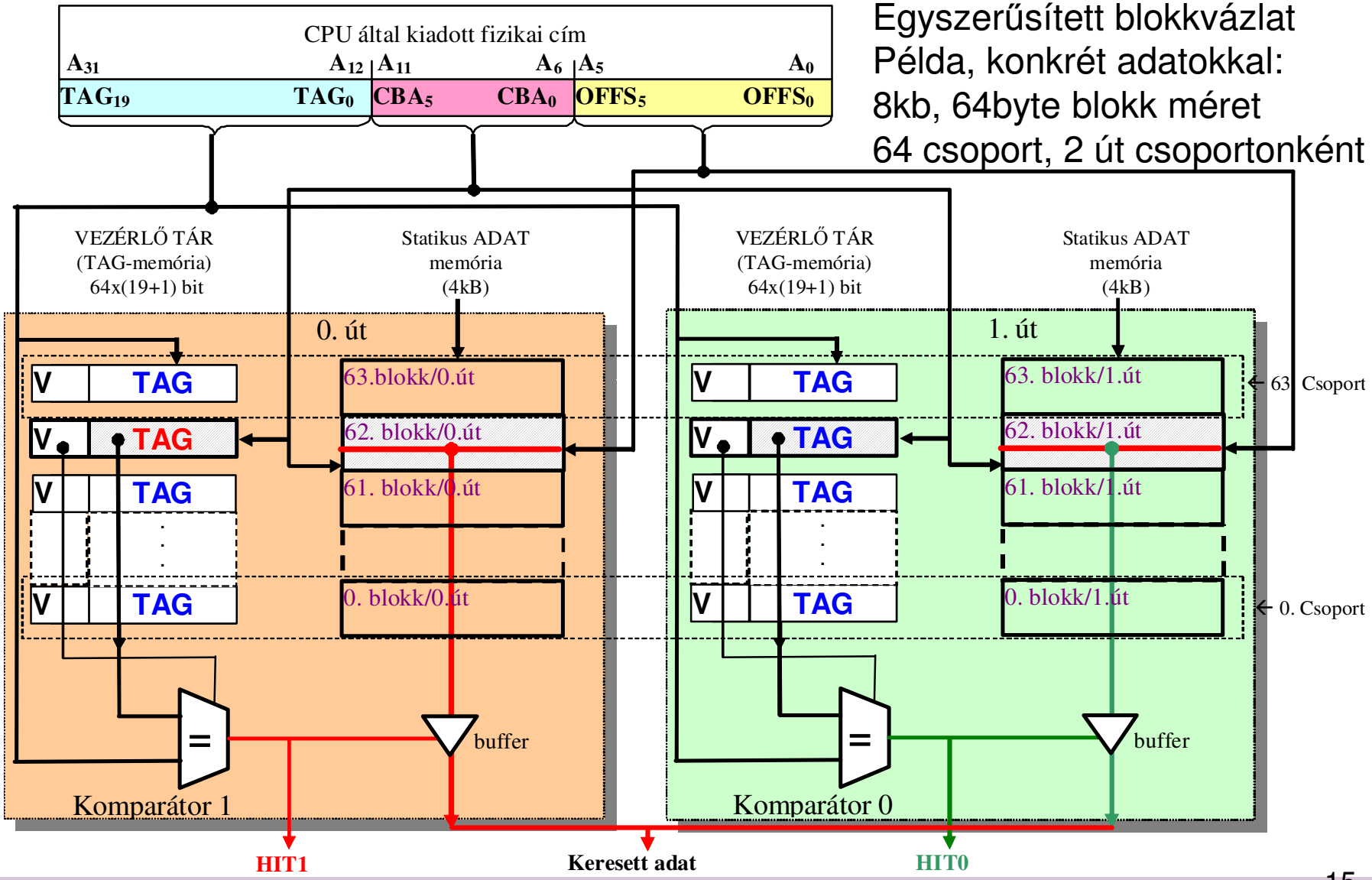


Részben asszociatív leképzés (*set associative mapping*)

- *A CACHE 2,-4,-8,-n azonos méretű részegységre van bontva (n =2^m n utas direkt leképezés), az egyes részegységek (út) önállóan a közvetlen leképzés szabálya szerint működnek.*
  Egy adott keretsorszámú (K_B) blokk a CACHE részegységei azonos sorszámú (C_B) blokkjainak (n db csoport,set) valamelyikébe helyezhető

2 utas set-asszociatív leképezés

Egyszerűsített blokkvázlat
Példa, konkrét adatokkal:
8kb, 64byte blokk méret
64 csoport, 2 út csoportonként



n -utas részben asszociatív leképzés

- Nő a HIT rate, jobb a kihasználtsága */előny/*
- Kell blokkcsere algoritmus */hátrány/*

Operatív memória: 4Gbyte Cache: 128 blokk Blokk:64 byte **Közvetlen leképzés**

	A_{31} OPERATÍV MEMÓRIA A_0			4GB CÍM 32bit
	TAG_{18} TAG_0	CBA_6 CBA_0	$OFFS_5$ $OFFS_0$	128 blokk 7bit
V	A_{31} TAG A_{13}	A_{12} CBA A_6	A_5 OFFSET A_0	64 byte 6bit

Két-utas leképzés (asszociativitás)

	A_{31} OPERATÍV MEMÓRIA A_0			4GB CÍM 32bit
	TAG_{19} TAG_0	CBA_5 CBA_0	$OFFS_5$ $OFFS_0$	2x64 blokk 6bit
V A	A_{31} TAG A_{12}	A_{11} CBA A_6	A_5 OFFSET A_0	64 byte 6bit

Négy-utas leképzés (asszociativitás)

	A_{31} OPERATÍV MEMÓRIA A_0			4GB CÍM 32bit
	TAG_{20} TAG_0	CBA_4 CBA_0	$OFFS_5$ $OFFS_0$	4x32 blokk 5bit
V A	A_{31} TAG A_{11}	A_{10} CBA A_6	A_5 OFFSET A_0	64 byte 6bit

Cache leképzési stratégiák összehasonlítása

leképzés	Találati arány	Keresés sebessége	Bonyolultság
Közvetlen	*jó*	*legjobb*	*legegyszerűbb*
Asszociatív	*legjobb*	*közepes*	*legbonyolultabb*
N-utas direkt /szet asszociatív/	*nagyon jó* *jobb, ha N nő*	*jó* *romlik, ha N nő*	*bonyolult*

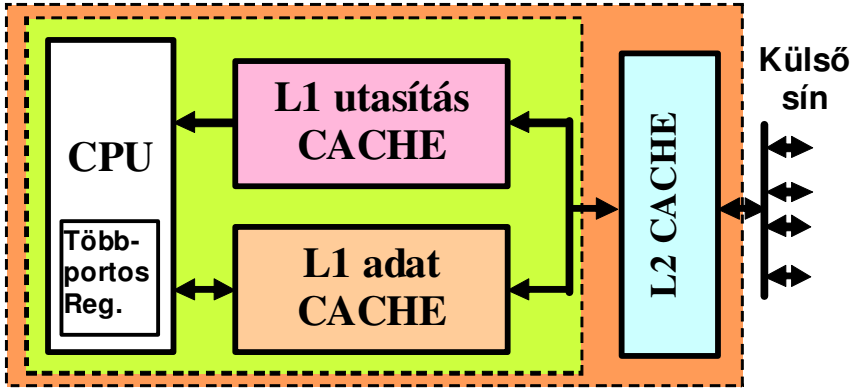
A CACHE szervezés kérdései

- Leképzési stratégia (*placement policy)*
  Az operatív memória egy adott blokkját a CACHE melyik sorába (blokkjába, ***line****) tölti*
  - *Blokk mérete: kicsi ←→ nagy*
  - *Hogyan ismeri fel a betöltött információt*
- Behozatali stratégia (*Fetch policy*)
  Mikor töltünk be egy blokkot
- Írási stratégia (*update policy*)
  Hogyan írjunk a CACHE-be, illetve az OM-be
- Blokkcsere stratégia (*block replacement policy*)
  Új blokk betöltésekor melyik blokk helyére töltünk be
- *Gyors blokkbetöltés megoldása*

Cache blokk behozatali stratégiák

- Mikor hozunk be egy blokkot /***Fetch policy***/?
  - ❑Közvetlen igény esetén /***MISS-nél***/
  - ❑Előrelátással
    - *i. blokk igénye esetén az (i+1) blokkot is*
      - →*Adat koherencia kezelése (**összetartozó adatok?**)*
    - *spekulatív „okos” betöltés →Pentium 4 L2*
  - ❑Szelektíven
    - *cím szerint (bizonyos címtartományokat **nem** töltünk be)*
    - *típus szerint külön utasítás és/vagy adat*

Hierarchikus felépítésű CACHE



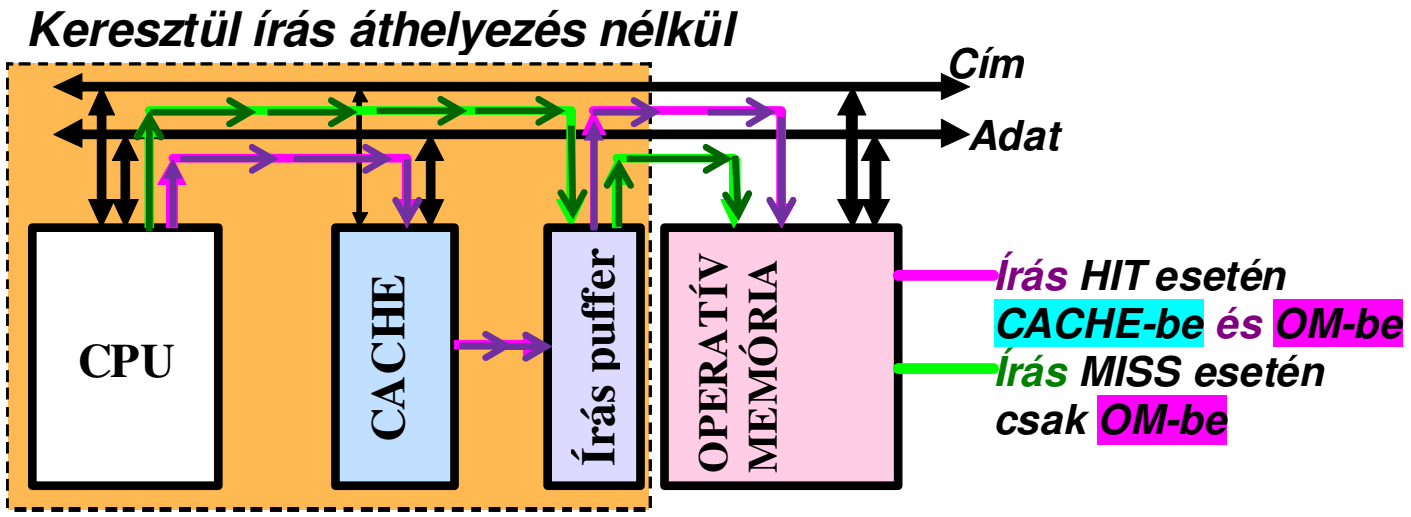
pl.:Pentium 4	***2X (16-32kB)***	***8 utas***	***L1***
	1X (2-4MB)	***8-16 utas***	***L2 cache***

A CACHE szervezés kérdései

- Leképzési stratégia (*placement policy)*
  Az operatív memória egy adott blokkját a CACHE melyik sorába (blokkjába, ***line****) tölti*
  - *Blokk mérete :kicsi ←→ nagy*
  - *Hogyan ismeri fel a betöltött információt*
- Behozatali stratégia (*Fetch policy*)
  Mikor töltünk be egy blokkot
- Írási stratégia (*update policy*)
  Hogyan írjunk a CACHE-be, illetve az OM-be
- Blokkcsere stratégia (*block replacement policy*)
  Új blokk betöltésekor melyik blokk helyére töltünk be
- *Gyors blokkbetöltés megoldása*

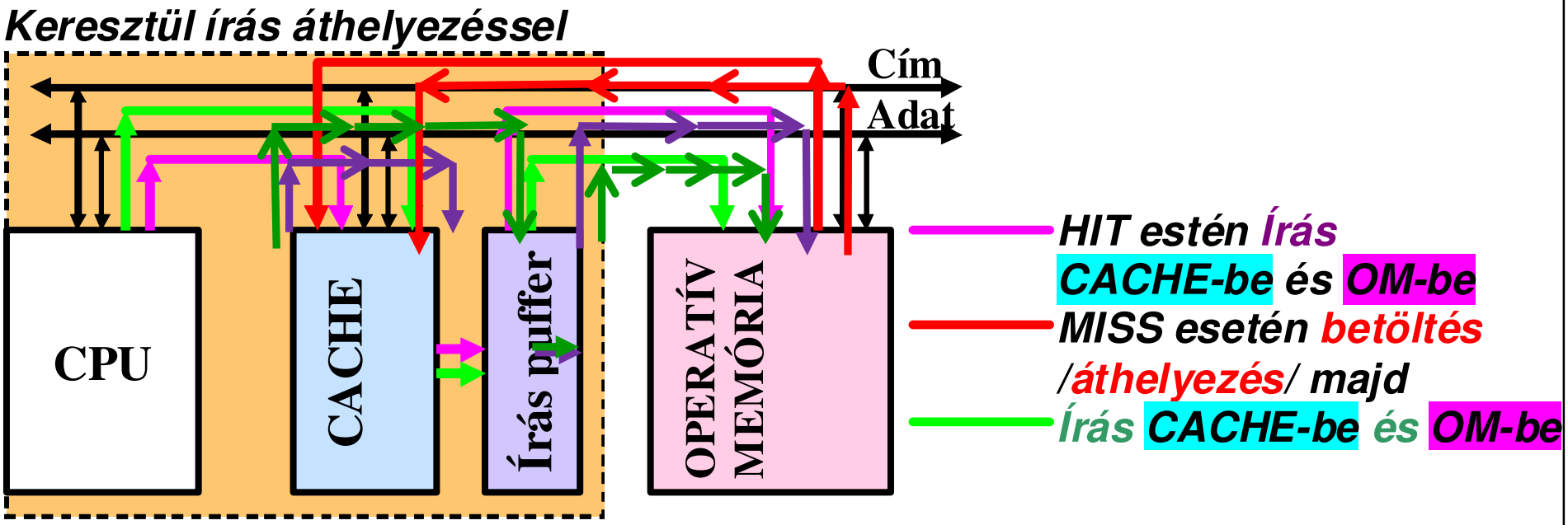
Keresztül írás áthelyezés nélkül

- Keresztülíró stratégia /***write through policy***/
  - Találat esetén /***HIT***/ *ÍR a CACHE-be és az OM-be is*
  - MISS esetén
    - Írás áthelyezés nélkül /***write through with no write allocate policy***/ *csak az OM-be ÍR*



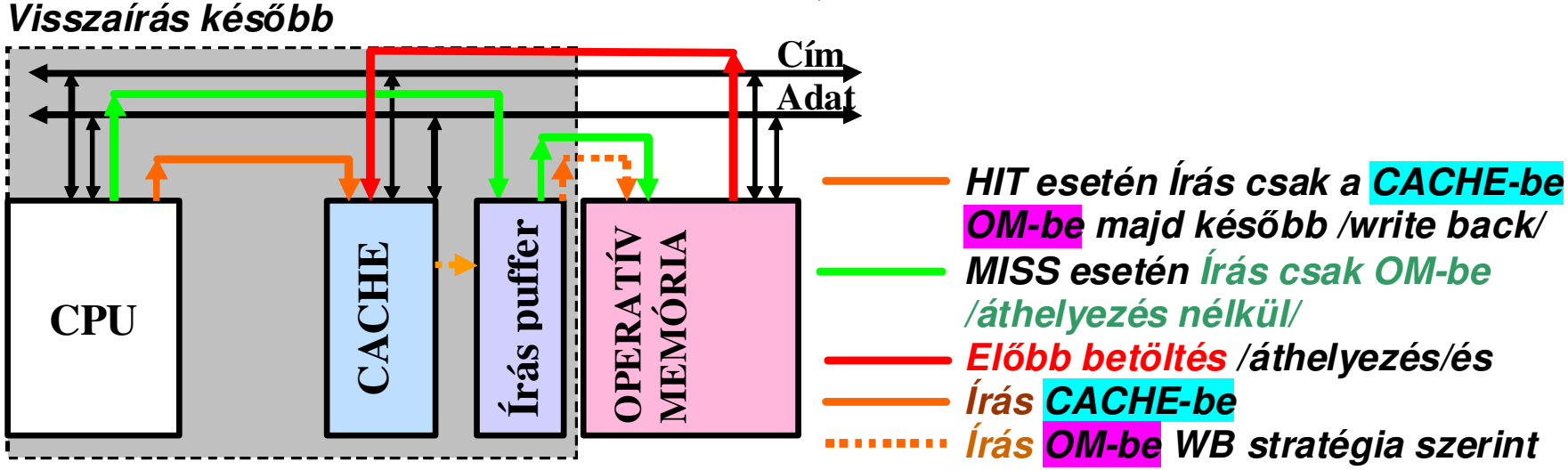
Keresztülírás áthelyezéssel

- **Írás áthelyezéssel** */write through with write allocate policy/*
  betölti, majd keresztülírja



Visszaírás

- Visszaírás (***write back policy***)
  - *Működhet áthelyezés nélkül és áthelyezéssel*
  - *HIT esetén csak a CACHE-ba ír, az OM-be csak később*



Írási stratégiák problémái

- Adat konzisztencia problémák kezelése
  - Adott címtartomány CACHE-be töltésének tiltása
  - CACHE érvényes bejegyzés törlése HW/SW
  - Írás puffer kezelése
    - » bizonyos műveletek felfüggesztése (pl.:I/O) a kiírás befejezéséig
    - » írás - olvasás sorrendjének felcserélése!?
  - Többszintű- többmagos- többprocesszoros rendszernél
    - » MESI protokoll alkalmazása ***/Modified, Exclusive, Shared, Invalid állapot/***
    - ✧ *Állapot nyilvántartás*
    - ✧ *A többi egység figyelése (snoop)*
    - ✧ *Jelzés*

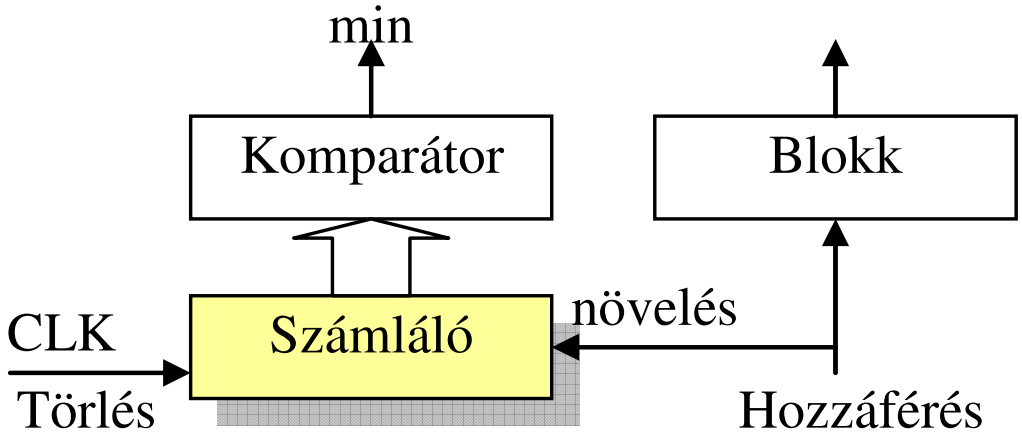
A CACHE szervezés kérdései

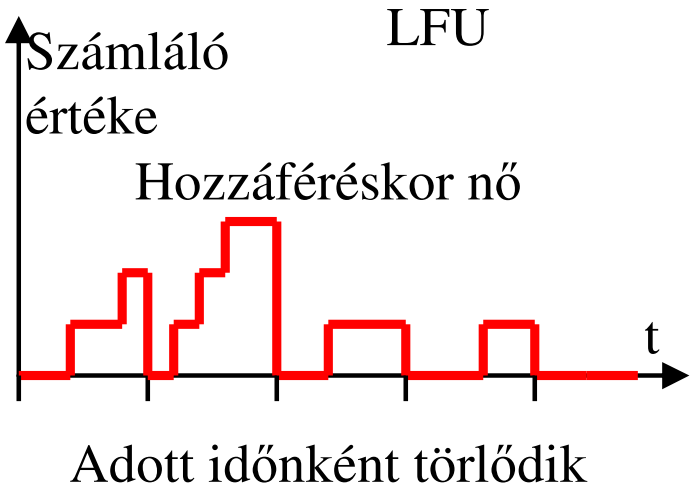
- Leképzési stratégia (*placement policy*)
  Az operatív memória egy adott blokkját a CACHE melyik sorába (blokkjába, ***line****) tölti*
  - *Blokk mérete: kicsi ←→ nagy*
  - *Hogyan ismeri fel a betöltött információt*
- Behozatali stratégia (*Fetch policy*)
  Mikor töltünk be egy blokkot
- Írási stratégia (*update policy*)
  Hogyan írjunk a CACHE-be, illetve az OM-be
- Blokkcsere stratégia (*block replacement policy*)
  Új blokk betöltésekor melyik blokk helyére töltünk be
- *Gyors blokkbetöltés megoldása*

Blokkcsere stratégiák

- Közvetlen leképzésnél nem kell (*csak egy helyre tölthető*)
- Asszociatív- N utas közvetlen (N way set associative) leképzés
  - Ha van üres oda
  - LFU (***least frequently used)*** legritkábban használt helyére
  - LRU (***least recently used)*** legrégebben használt helyére
  - FIFO (***first in- first out)*** legrégebben betöltött helyére
  - RANDOM tetszőleges (véletlenszerű) áldozat választás
  - Az írás-stratégia is módosíthatja az előzőket

LFU blokkcsere megvalósítása





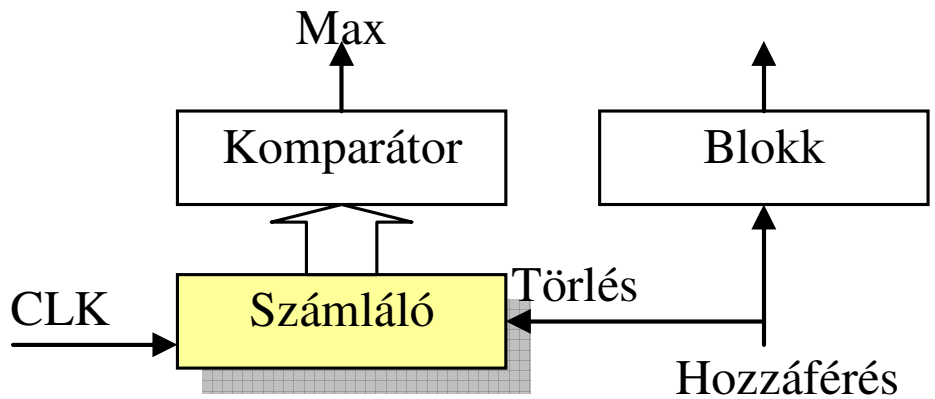
Néhány bites számláló

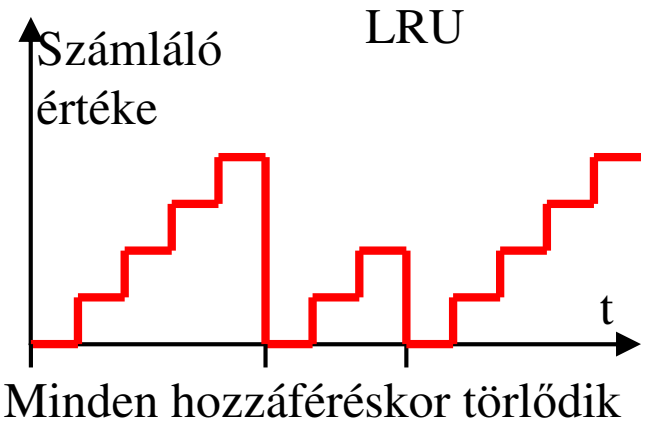
Minden hozzáférés növeli

Adott időnként törlés

A legkisebb a legritkábban használt

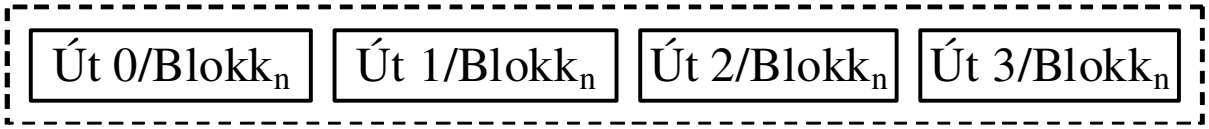
LRU blokkcsere megvalósítása



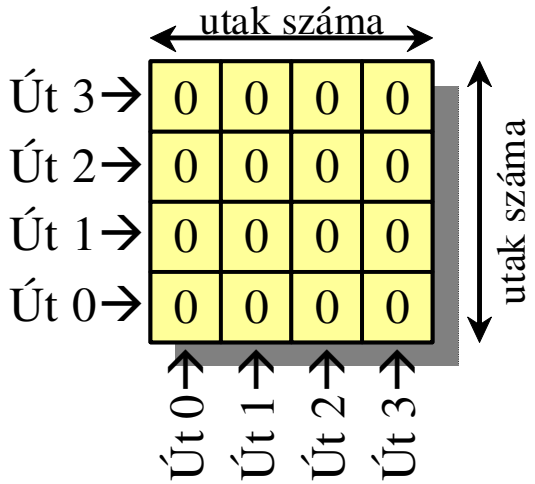


- Néhány bites számláló
- Minden hozzáférés törli
- A legnagyobb értékűt választjuk

Real-LRU Megvalósítás



Csoport $_n$
(asszociatív blokkok)

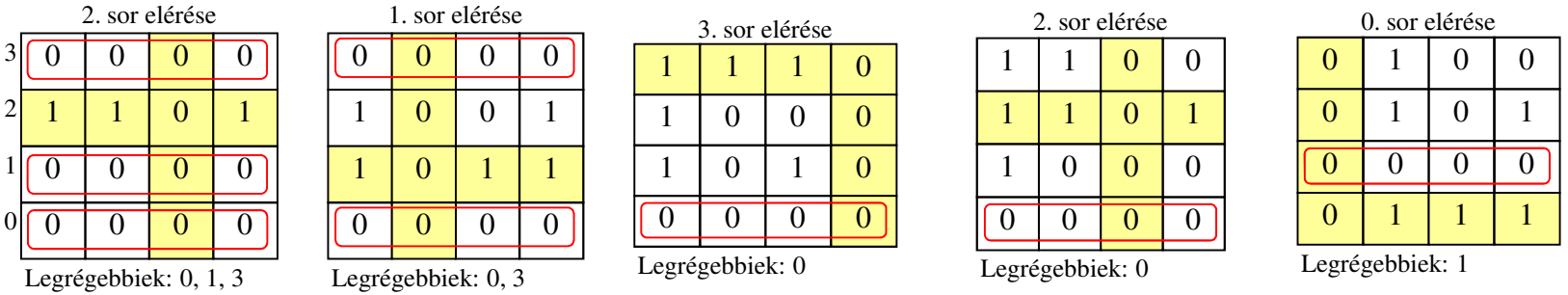


Működése:
A hivatkozott sort 1-be állítjuk
A hivatkozott oszlopon 0-ba állítjuk

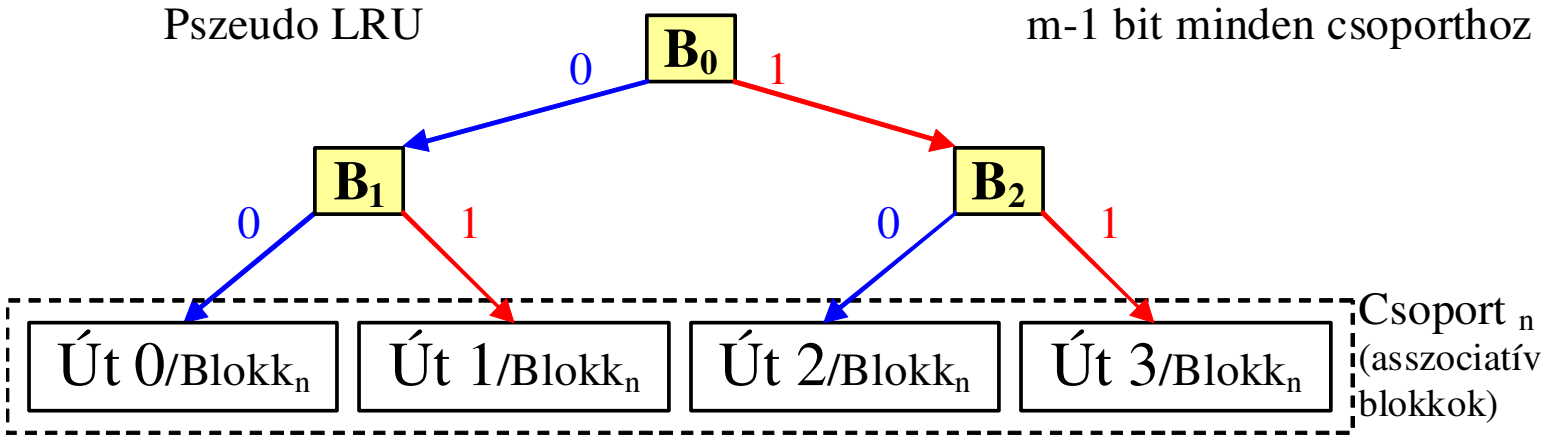
A tiszta 0 sor(ok)at használták a legrégebben

m^2 bit minden csoporthoz

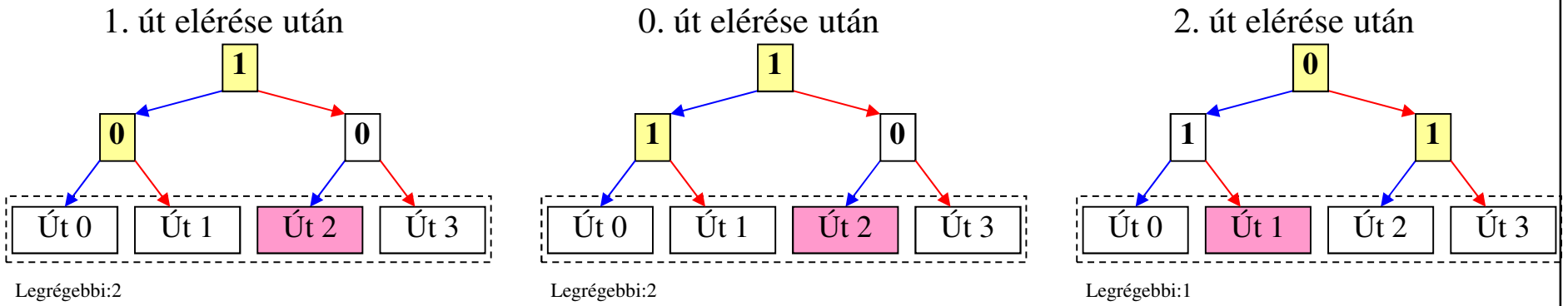
Legyenek a beérkező hivatkozások a következők: 2, 1, 3, 2, 0



Pszeudo LRU



B_0,B_1,B_2=0; Legyenek a beérkező hivatkozások a következők: 1, 0, 2



Cache szervezés összefoglalás

Leképezés:

- Direkt
- Asszociatív
- Részben asszociatív

Írás:

- Keresztül írás
  - Áthelyezéssel
  - Áthelyezés nélkül
- Visszaírás
  - Áthelyezéssel
  - Áthelyezés nélkül

Behozatal:

- Igény szerinti
- Előrelátó
- Szelektív

Blokkcsere:

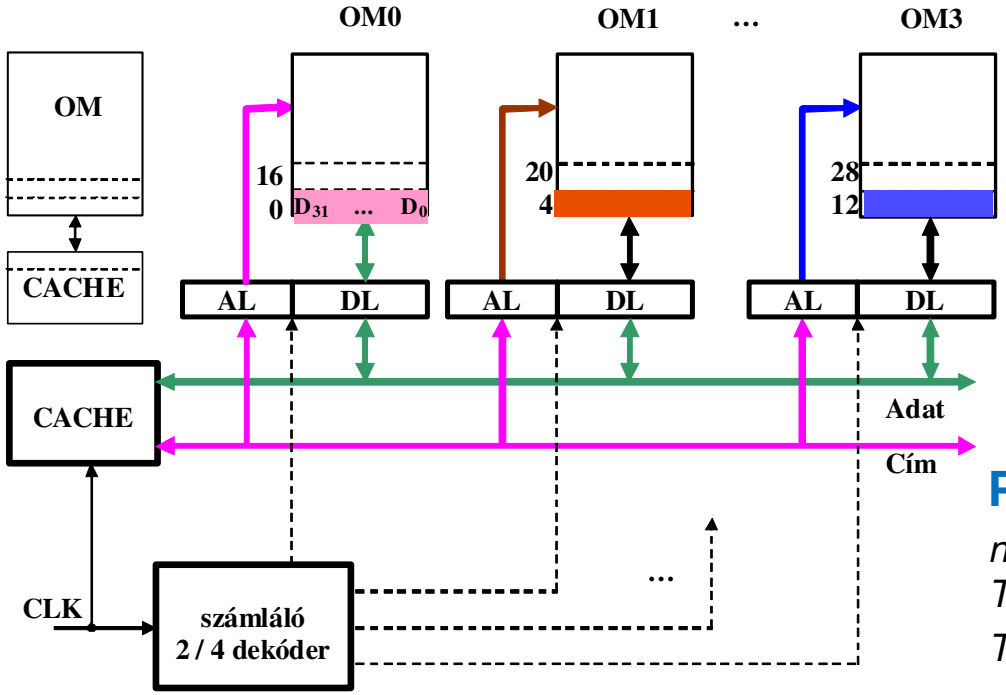
- Legritkábban használt
- Legrégebben használt
- Legrégebben betöltött
- Véletlen

A CACHE szervezés kérdései

- Leképzési stratégia (*placement policy)*
  Az operatív memória egy adott blokkját a CACHE melyik sorába (blokkjába, ***line****) tölti*
  - *Blokk mérete : kicsi ←→ nagy*
  - *Hogyan ismeri fel a betöltött információt*
- Behozatali stratégia (*Fetch policy*)
  Mikor töltünk be egy blokkot
- Írási stratégia (*update policy*)
  Hogyan írjunk a CACHE-be, illetve az OM-be
- Blokkcsere stratégia (*block replacement policy*)
  Új blokk betöltésekor melyik blokk helyére töltünk be
- *Gyors blokkbetöltés megoldása*

A blokkbetöltés gyorsítása

Átlapolt memóriaműködés /*memory interleave*/



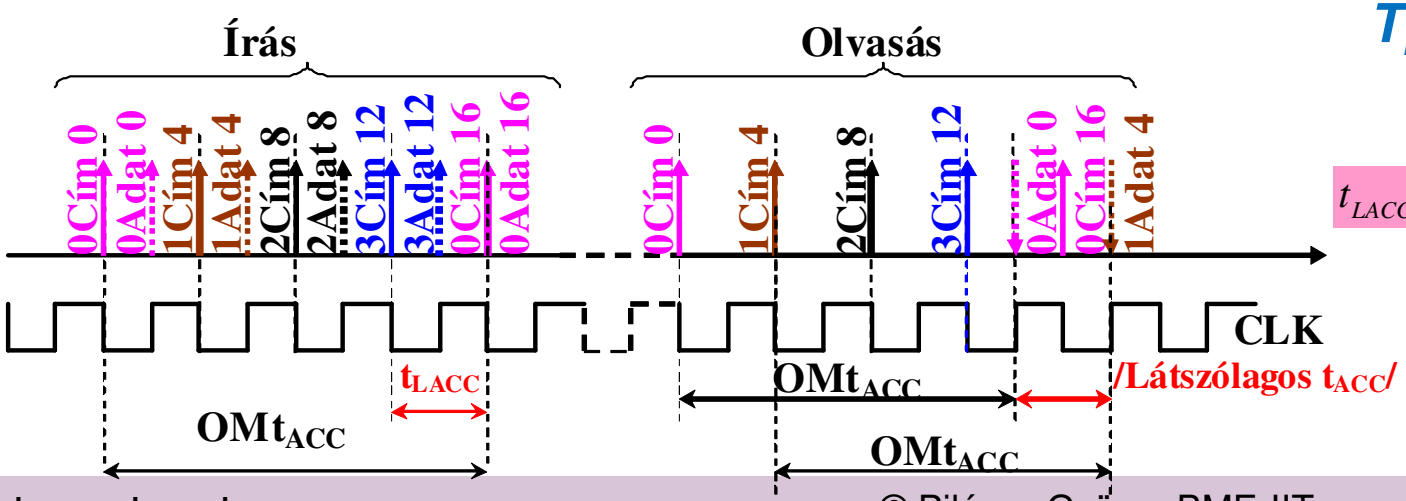
Pl.: T_C=11ns, T_{OM}=40 ns

n=4

T_{LOM}=40/4=10ns

T_{LACC}= max(10ns ,11ns)

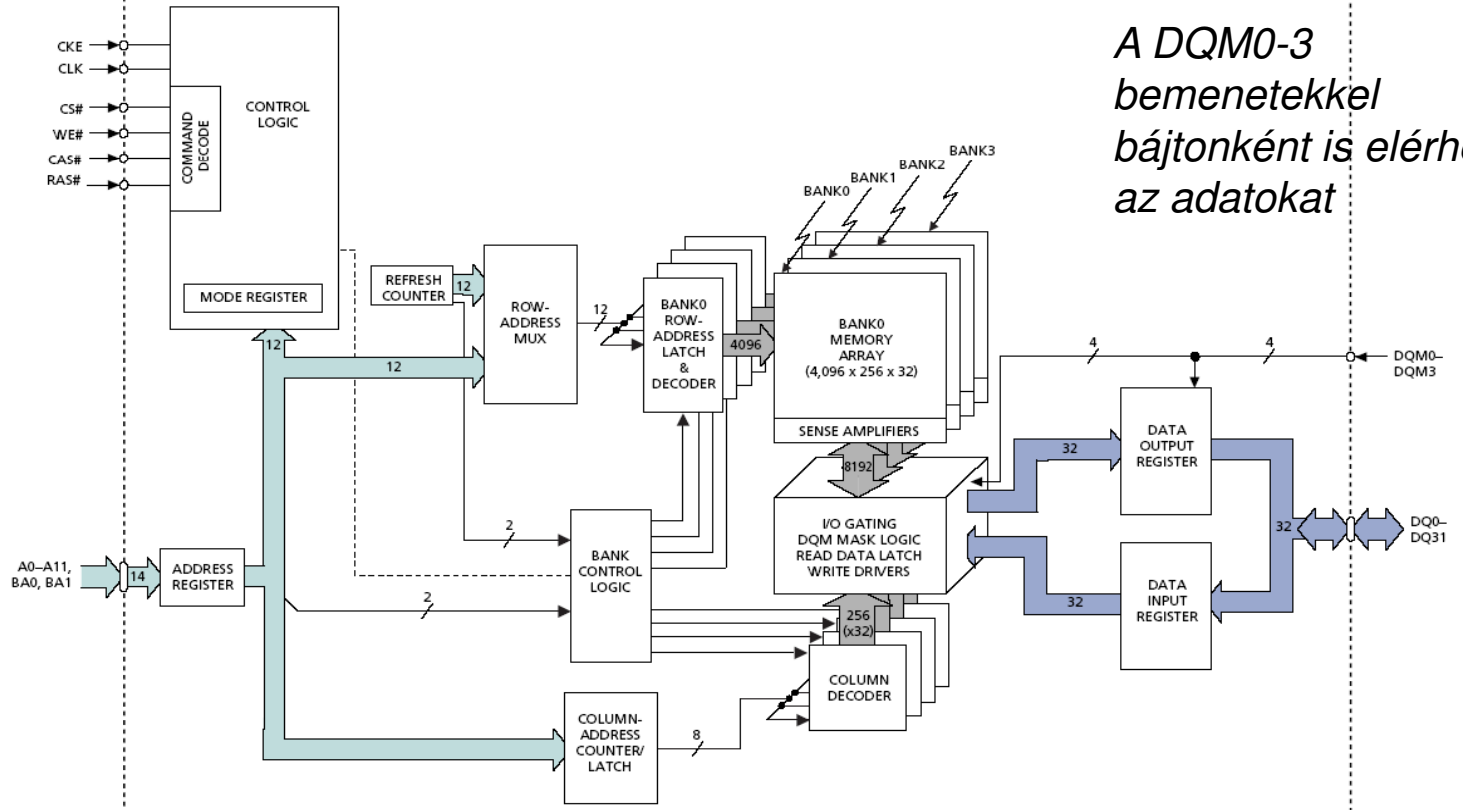
T_{LACC}=11ns



$$t_{LACC} = \max(t_{LOM}, T_C)$$

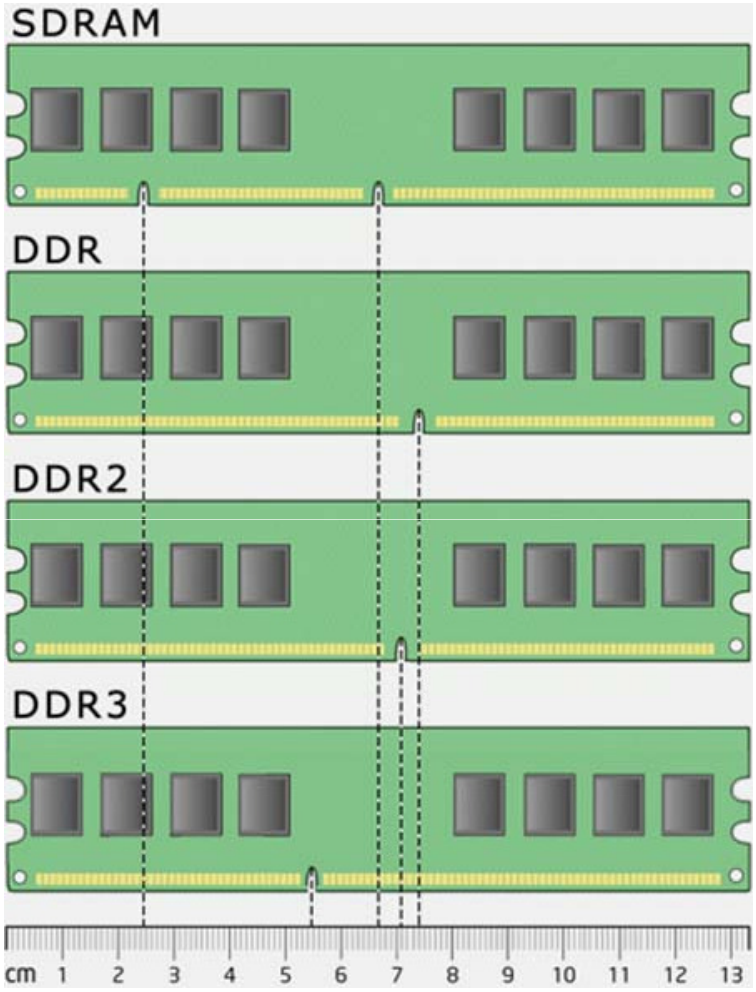
A blokkbetöltés gyorsítása DRAM-ok esetén

- Kihasználják a rendezett adatátvitelben rejlő lehetőséget
- FPM /***Fast page mode***/ ugyanabból a sorból csak oszlop cím kell
- EDO /***Extended Data Out***/ mint előző, de adat latch is van
- BEDO /***Burst Extended Data Out***/ belső oszlop címgenerálás
- SDRAM /***Synchronous DRAM***/ előzőek órajel ütemére

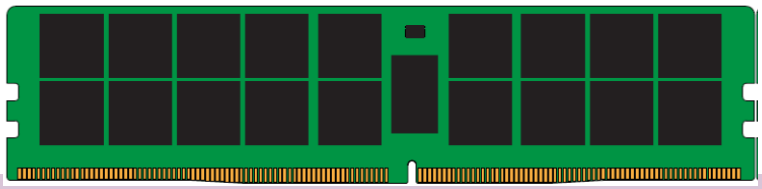


A DQM0-3 bemenetekkel bájtonként is elérhetjük az adatokat

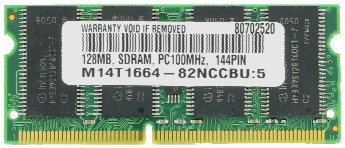
Memória modulok



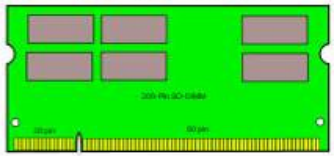
DDR4



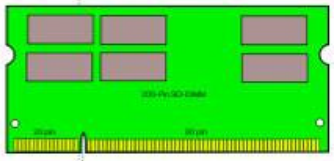
SO-DIMM SDRAM



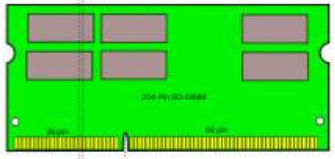
SO-DIMM DDR



SO-DIMM DDR 2



SO-DIMM DDR 3



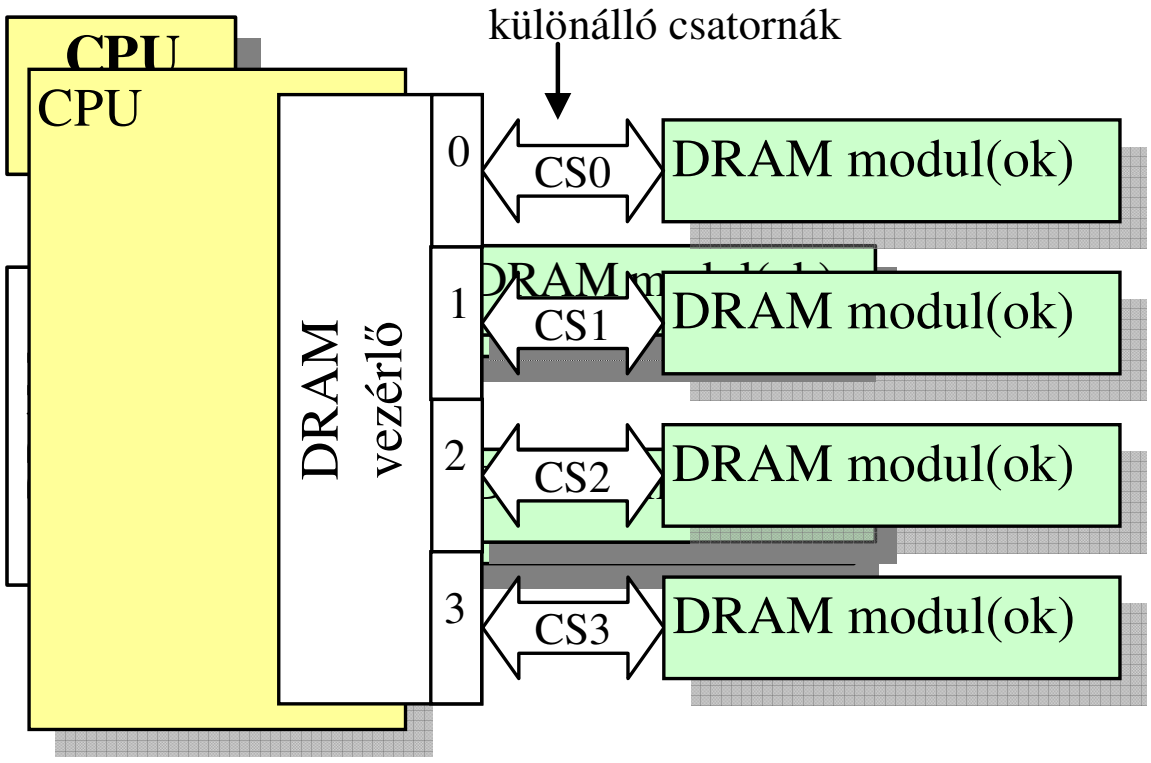
SO-DIMM DDR4

Különböző modulok
Asztali és hordozható számítógépekbe

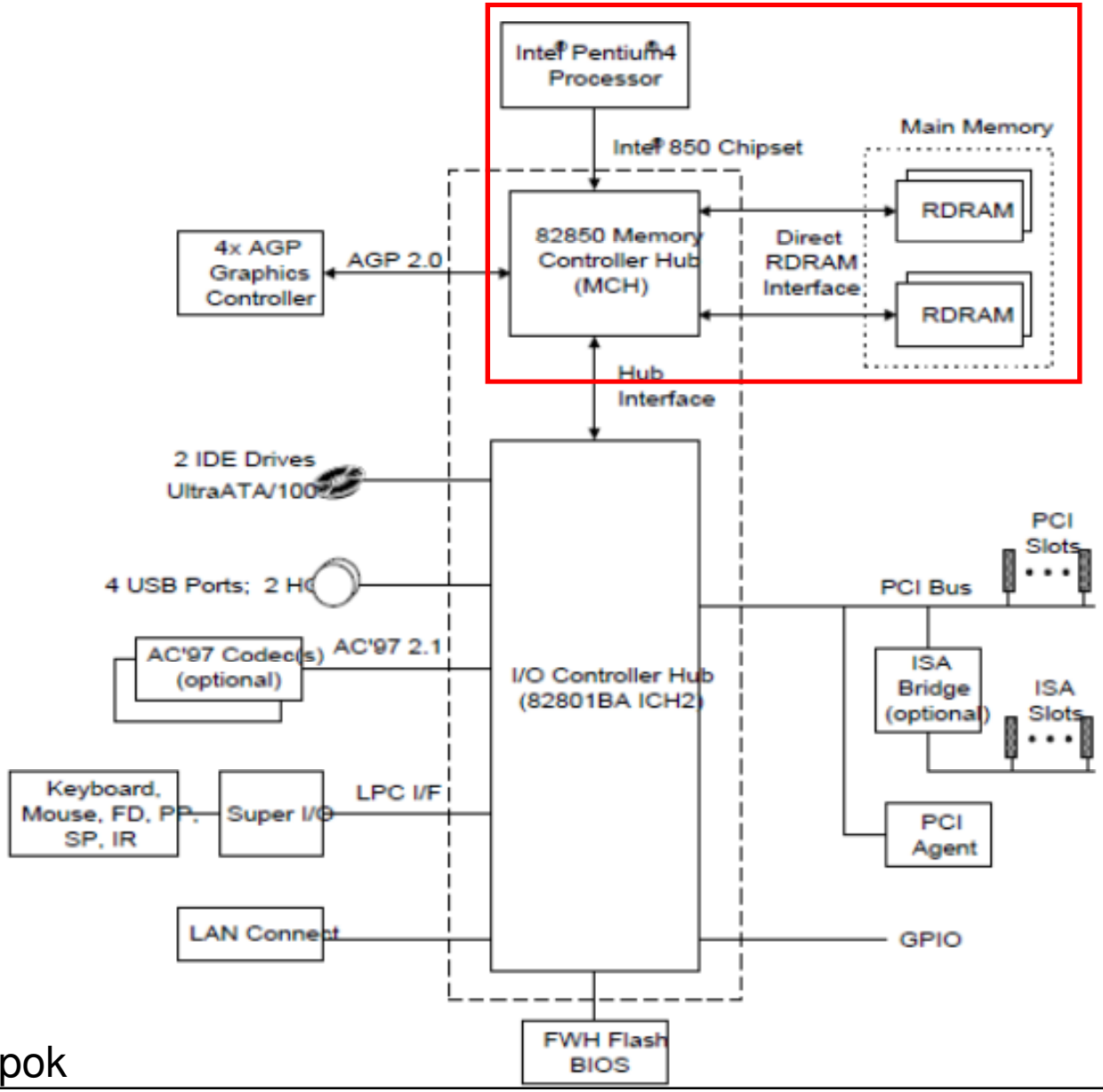
Modulok kapcsolata a CPU-val

DRAM vezérlő

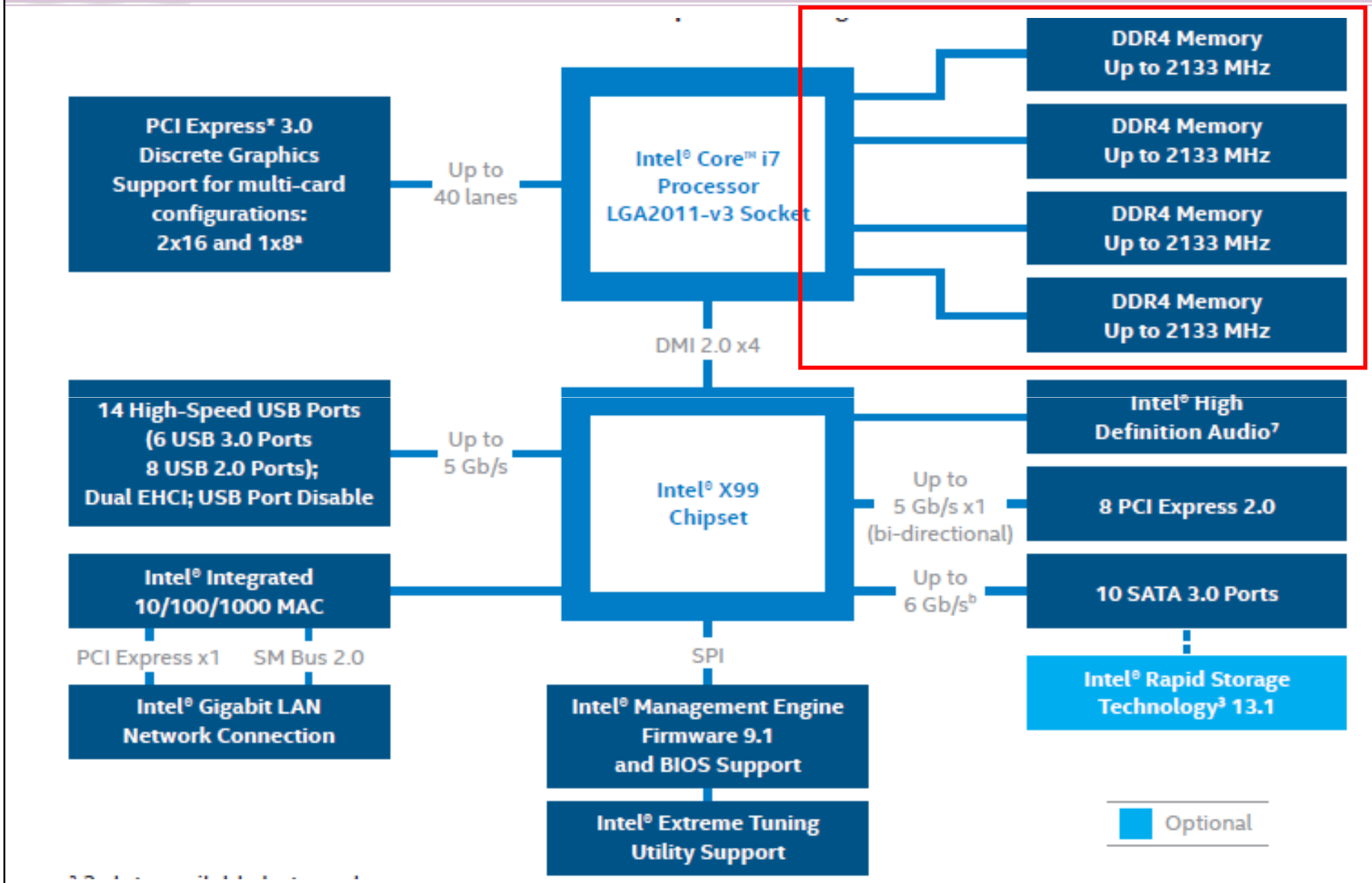
- Korábban különálló egység (chipset része)
- Ma a processzorlapka része



Különálló DRAM vezérlő

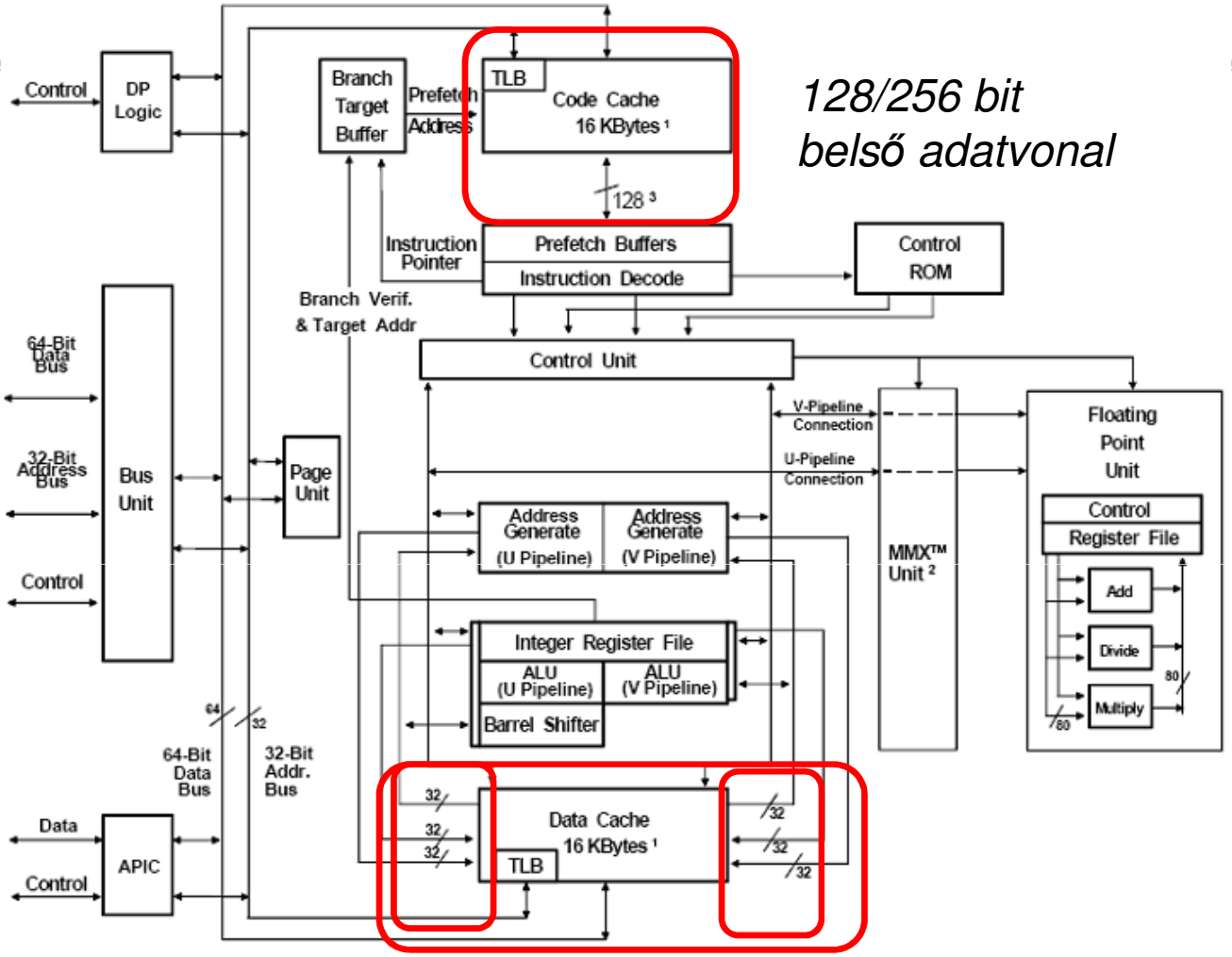


Processzorba integrált DRAM vezérlő



Esettanulmány - *Pentium CACHE megoldások*

- Pentium



128/256 bit belső adatvonal

Duál port adat cache

NOTES:

1. The Code and Data caches are each 8 Kbytes in size on the Pentium® processor (75/90/100/120/133/150/166/200).
2. The MMX Unit is present only on the Pentium processor with MMX™ technology
3. The internal instruction bus is 256 bits wide on the Pentium processor (75/90/100/120/133/150/166/200)

Pentium 4 */Intel technology journál Q1 2001/ / belső adatátviteli sebesség 48GB/s*

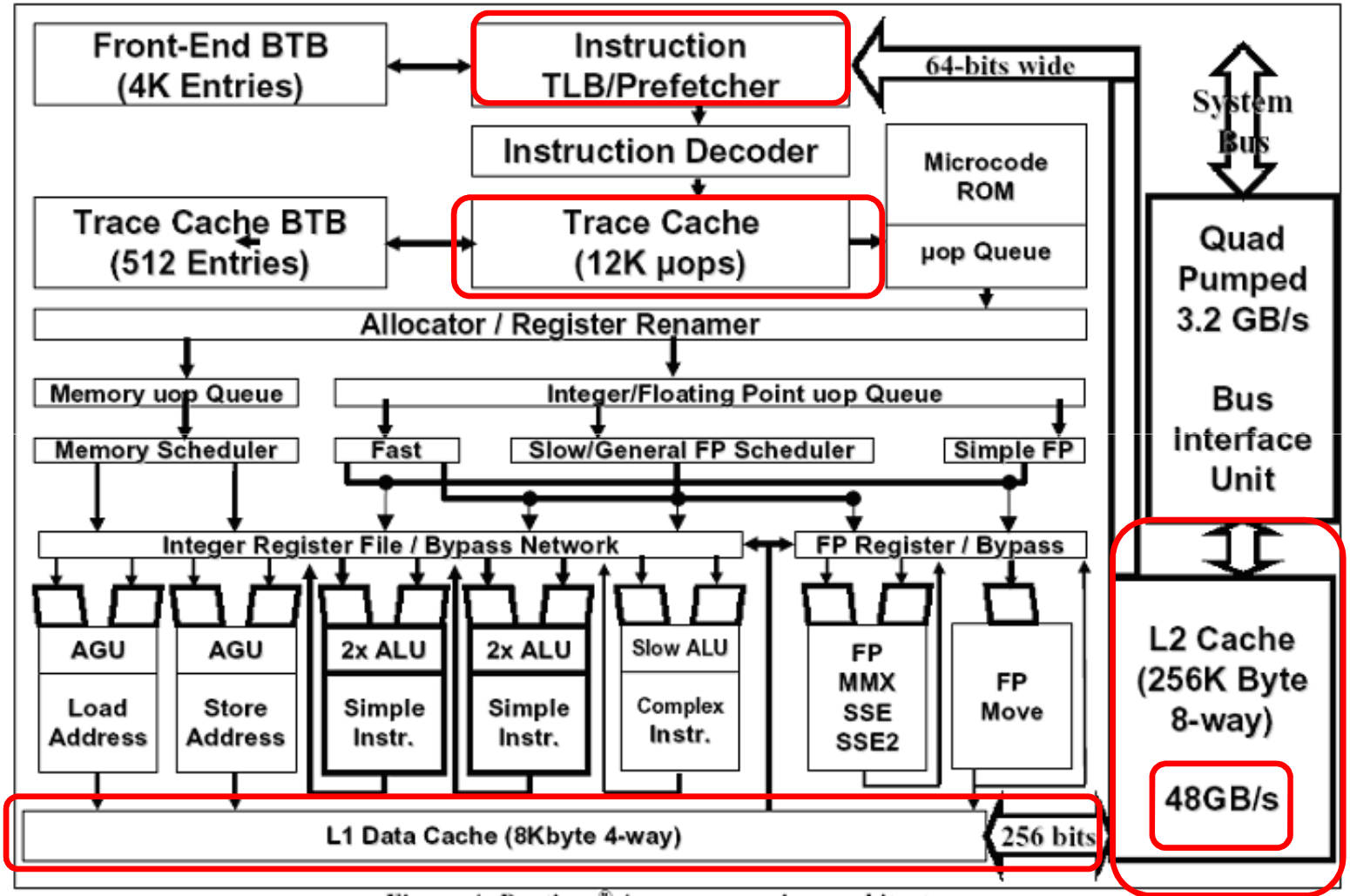
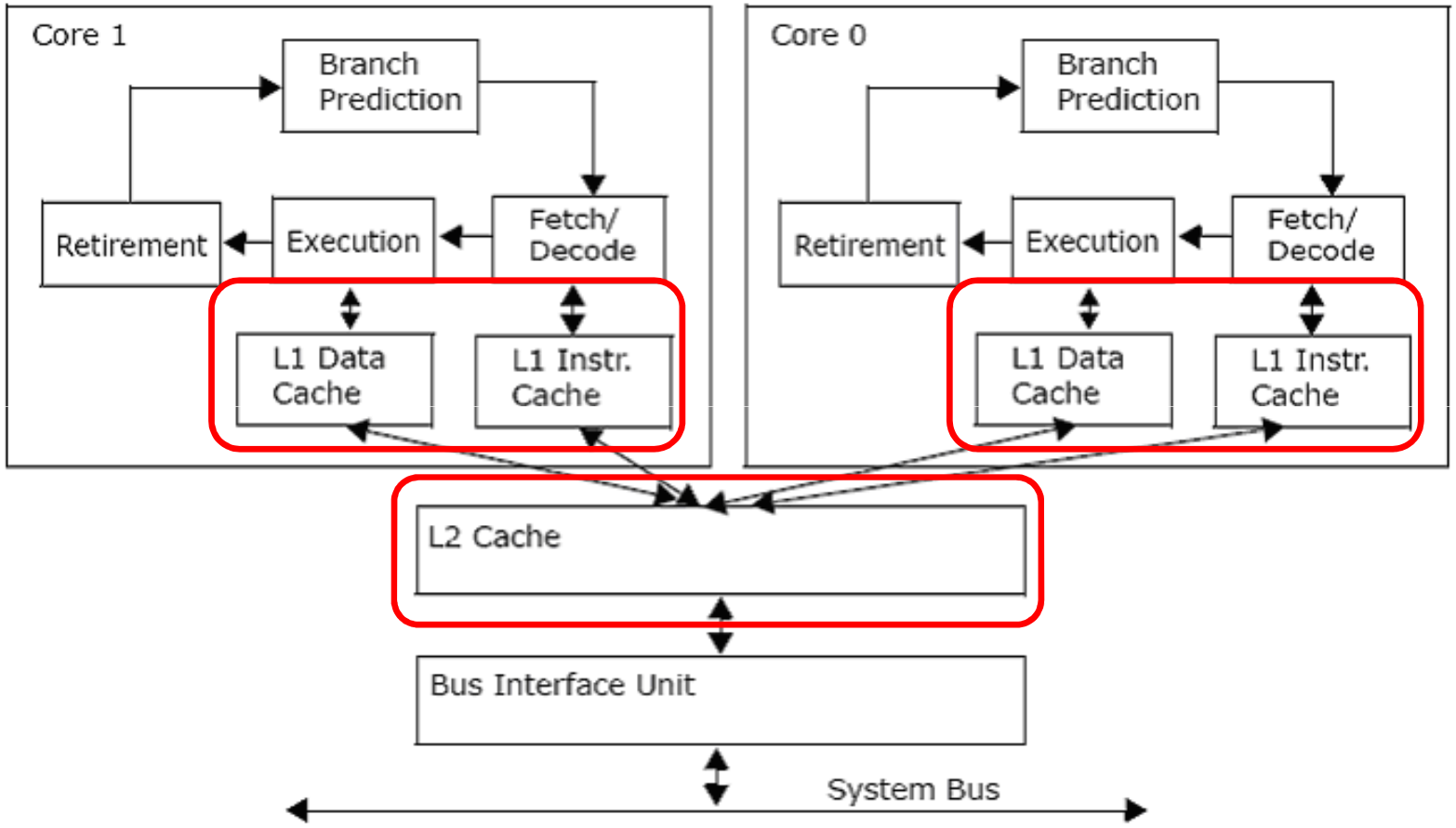
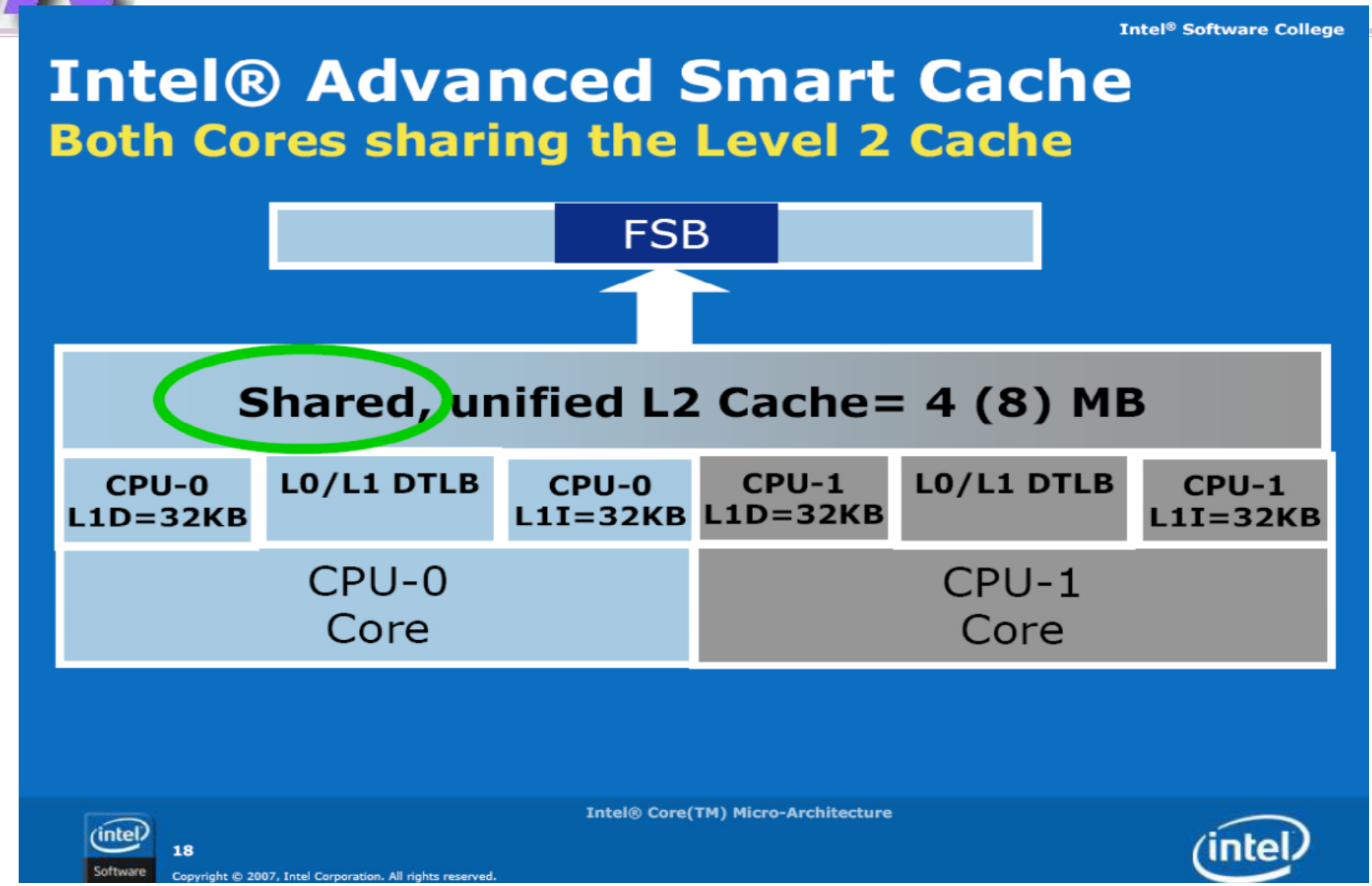


Figure 4: Pentium® 4 processor microarchitecture

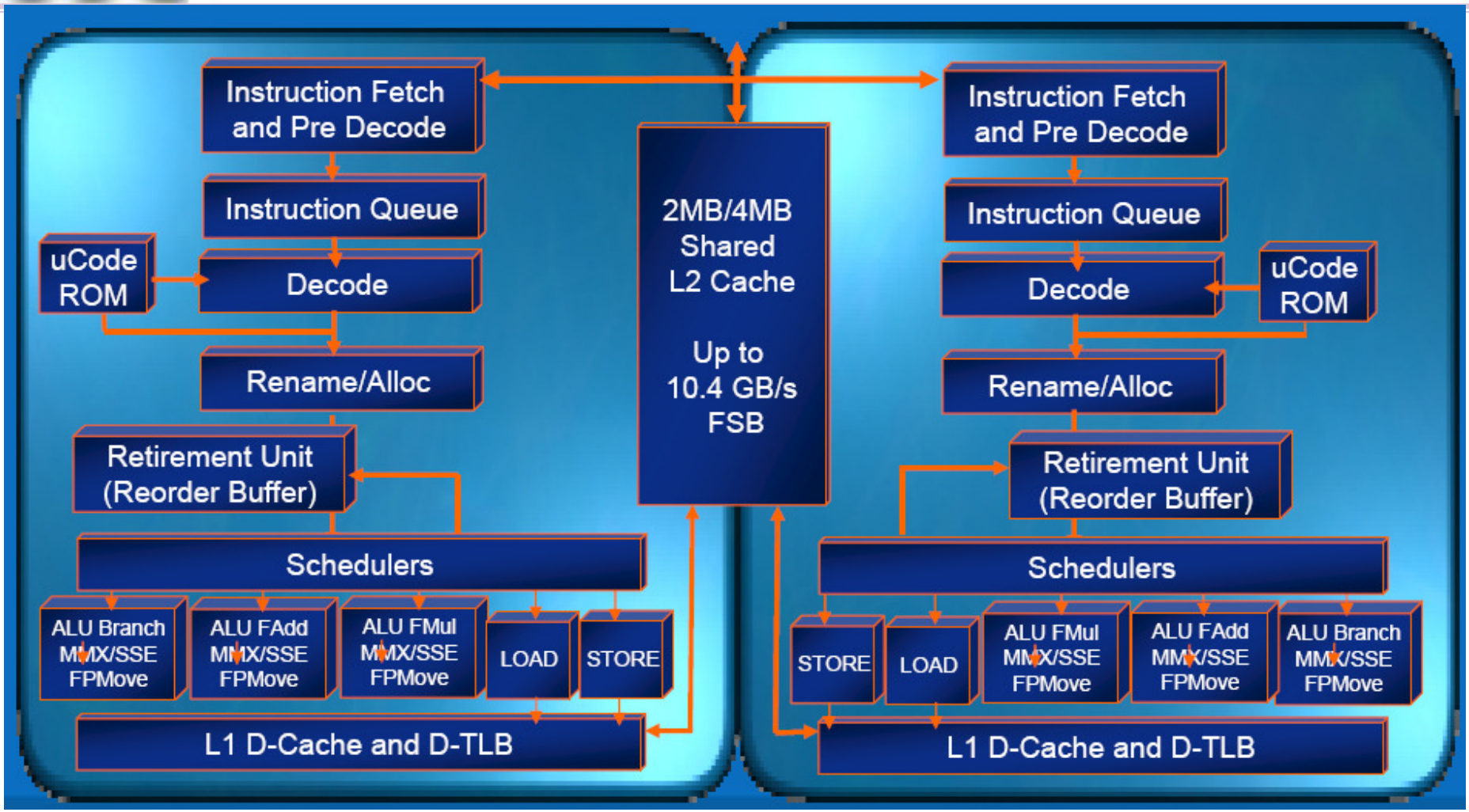
Esettanulmány - Pentium CACHE megoldások
Dual core



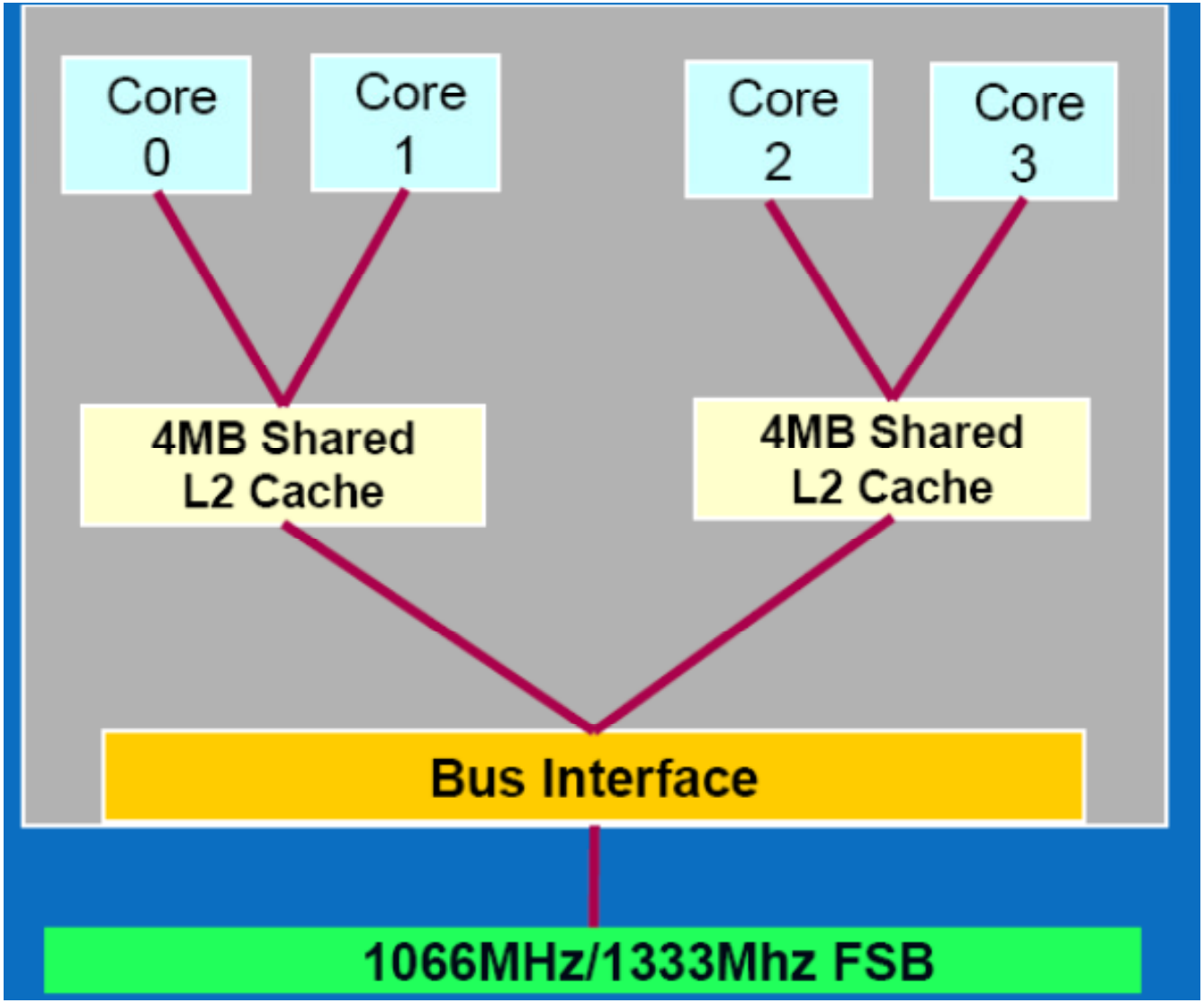
Esettanulmány - Pentium CACHE megoldások
Core 2 Duo



Esettanulmány - Pentium CACHE megoldások
Core 2 Duo

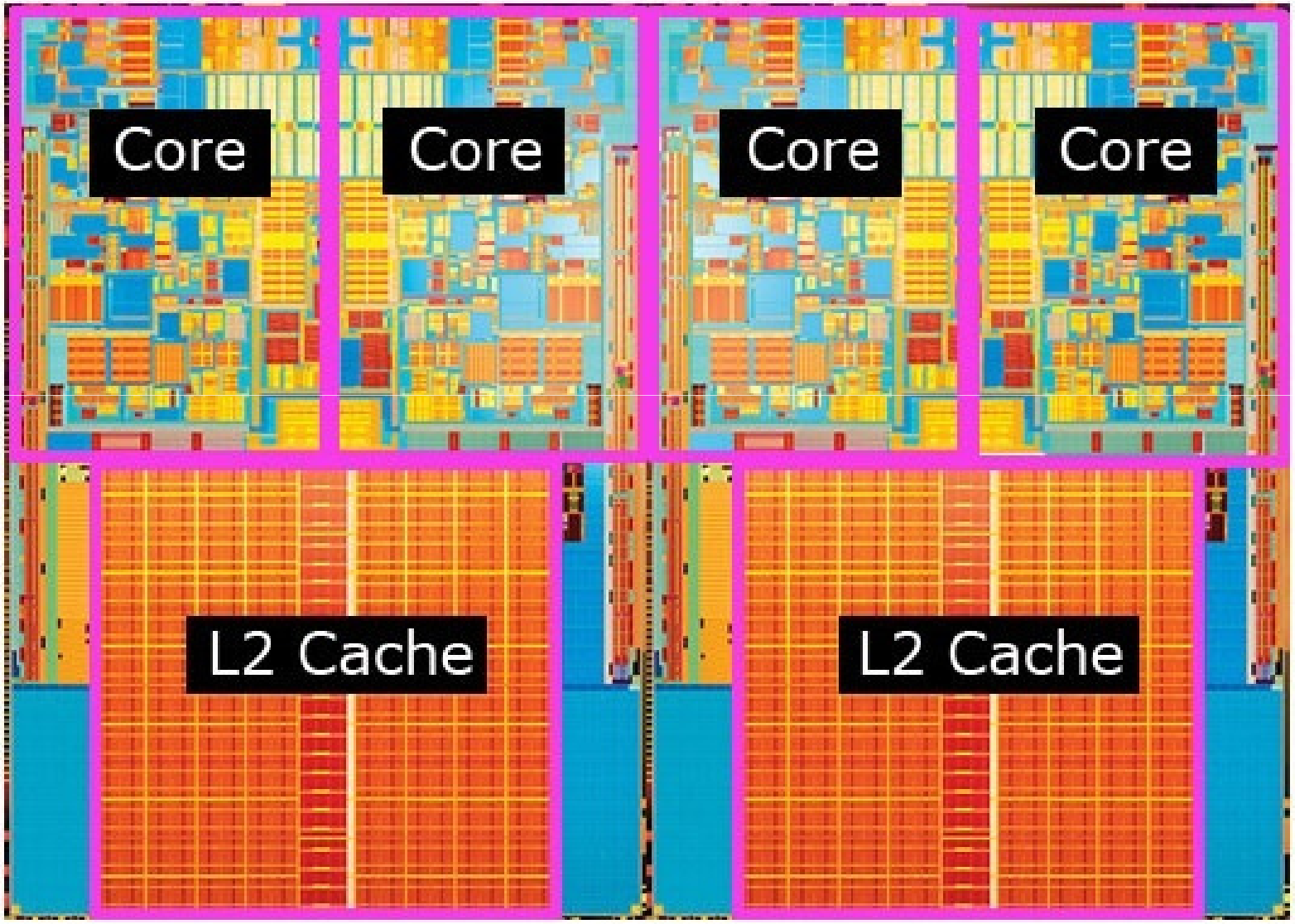


Esettanulmány Pentium CACHE megoldások



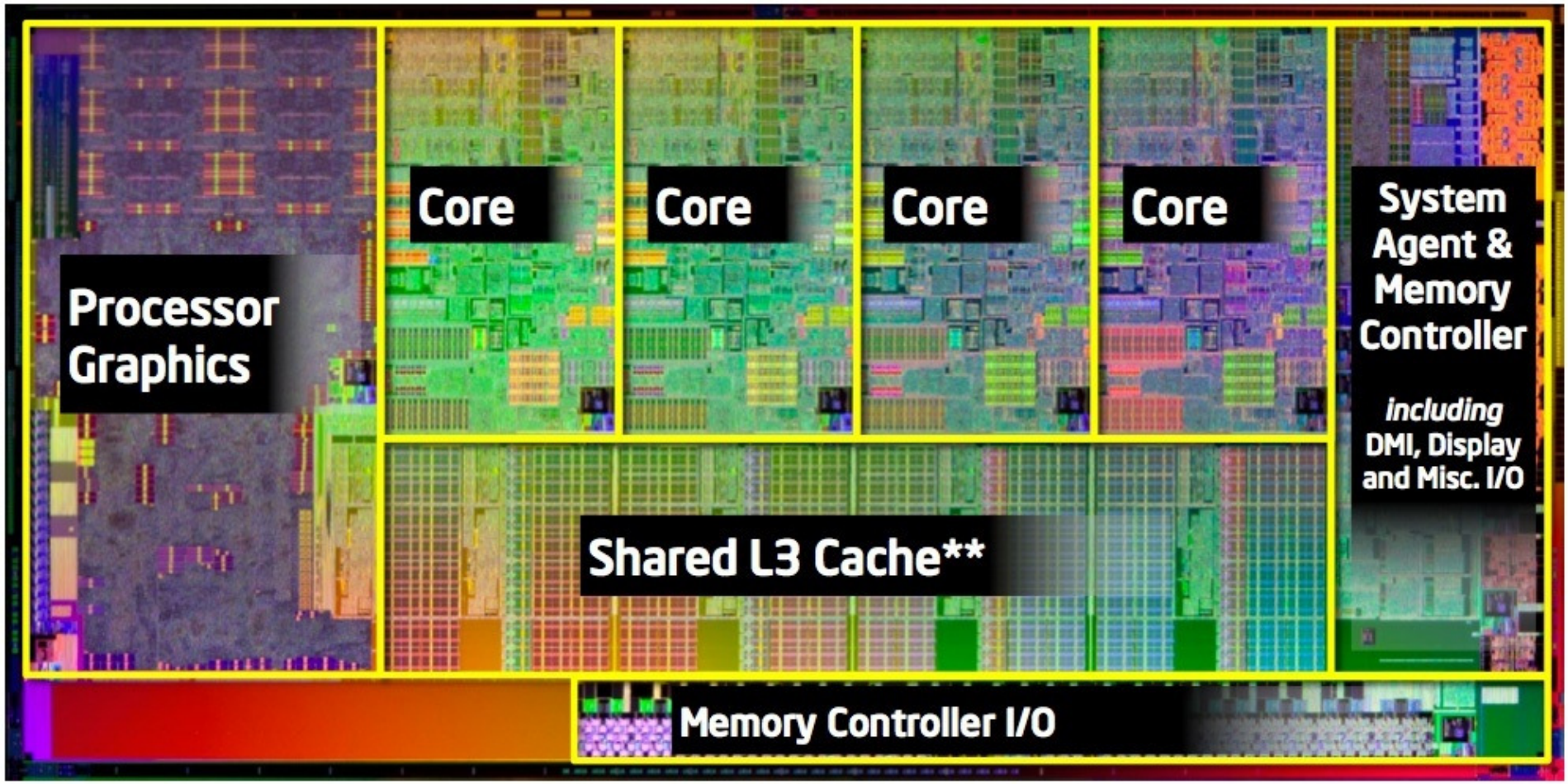
Esettanulmány Core2 CACHE megoldások

Quad-core Kentsfield package (2006)



Esettanulmány Core i7 CACHE megoldások

Quad-core Sandy Bridge lapka



Összefoglalás

- CPU-memória sebesség problémák
- Memória hierarchia
- Lokalitási elvek
- Gyorsítótár – Hr, Mr
- Cache szervezés, leképezés, blokkcsere, írási stratégiák
- Blokk betöltés gyorsítása
- DDR memória működési sajátosságok
  -blokk betöltés gyorsítása
- Korszerű processzorok gyorsító tár szervezései