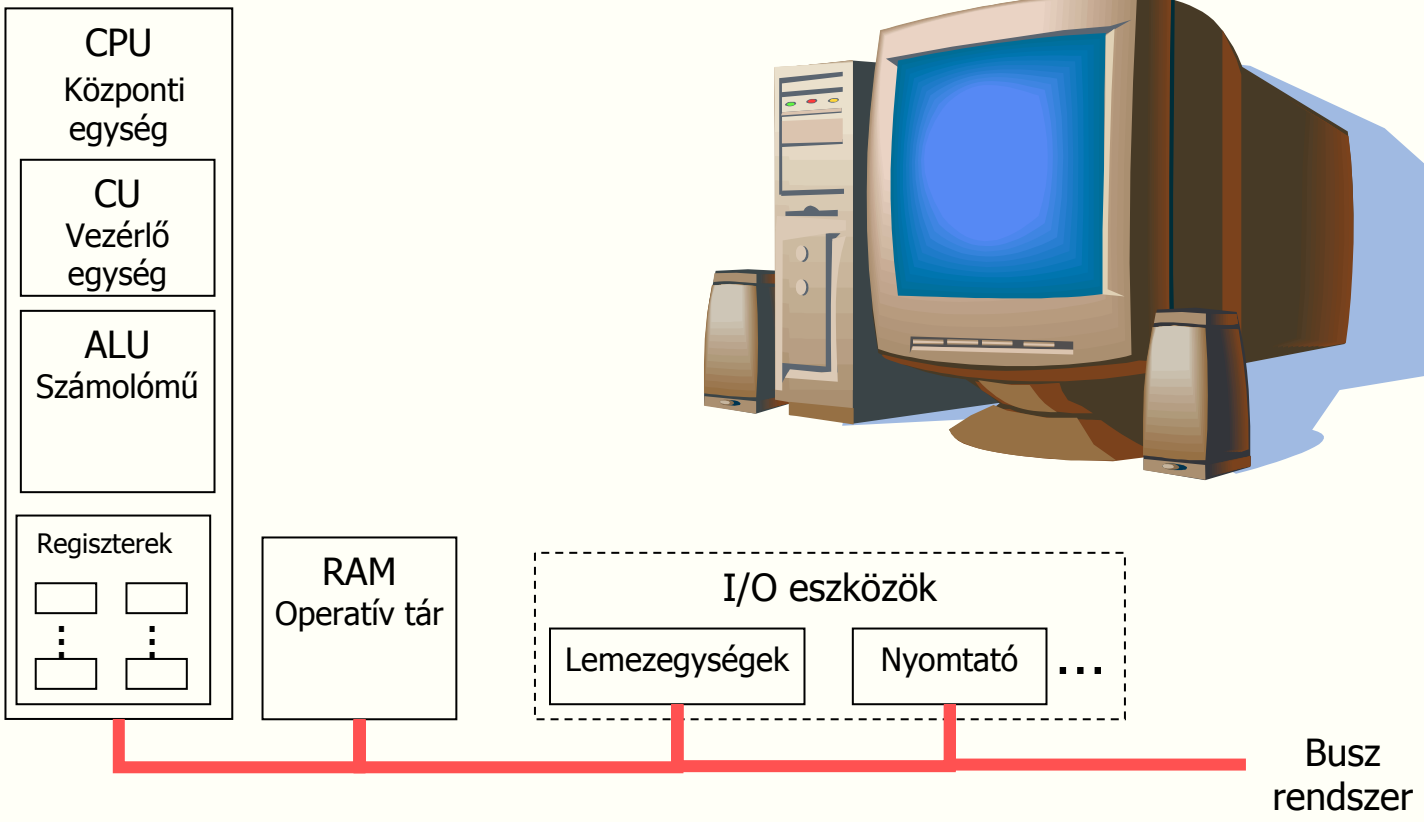
Budapesti Műszaki Főiskola
Regionális Oktatási és Innovációs Központ
Székesfehérvár

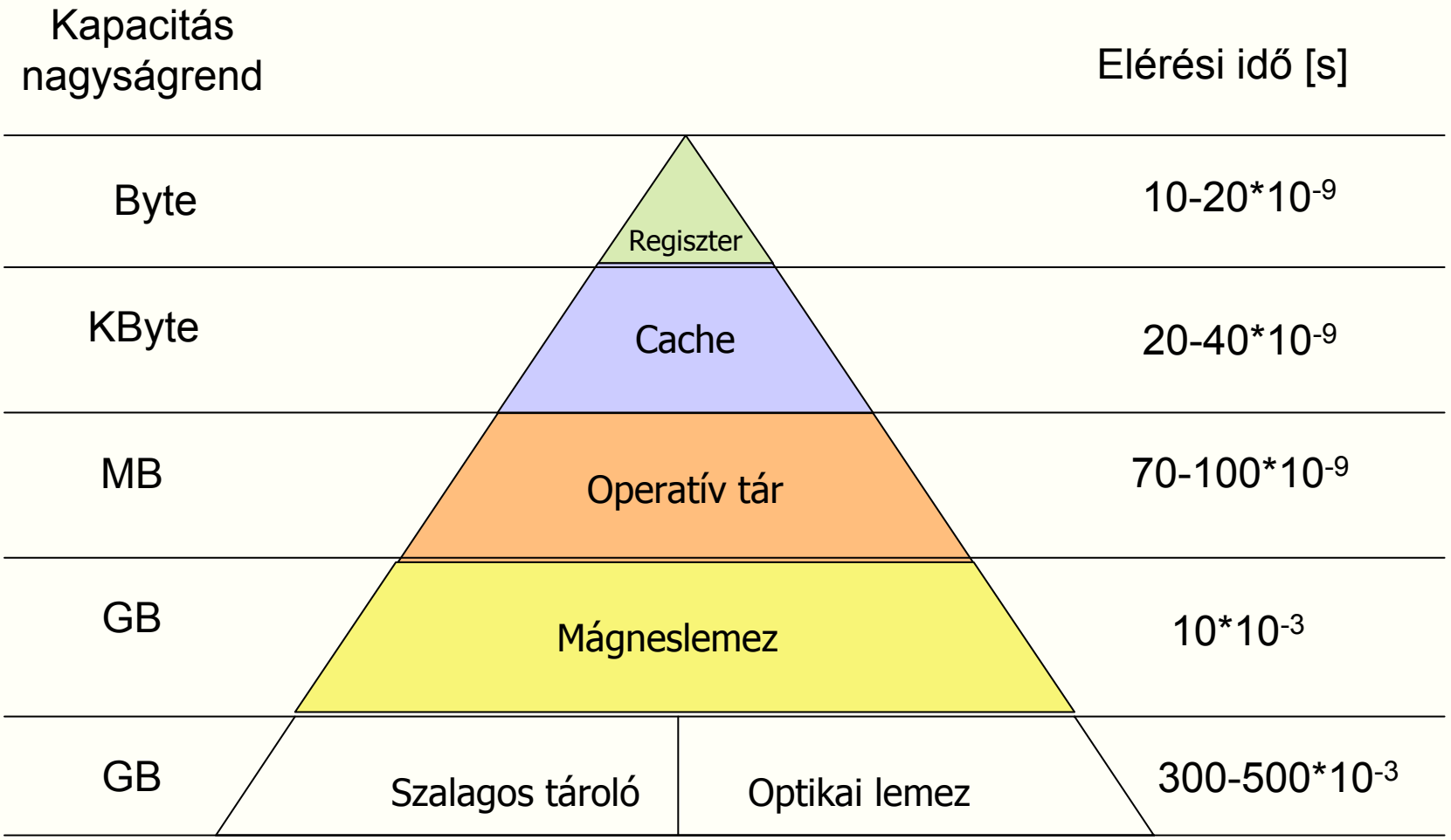
Tárak

Dr. Seebauer Márta
főiskolai tanár
seebauer.marta@roik.bmf.hu

Neumann-típusú számítógép általános felépítése



Tárhierarchia



Tárolási alapfogalmak

Az adat legkisebb tárolási egysége az egy bináris jel – **bit** (binary digit) értéke 0 vagy 1

A fizikailag egységként kezelhető legkisebb memóriaterület a **rekesz** (location, cell), mérete leggyakrabban 1 byte = 8 bit

A műveletvégzés során a számok ábrázolására 1 byte nem elegendő, ezért egy egységként 2-4-8 byte-ot használ a processzor. Ezt **szó**nak nevezzük. A szó az adatnak nem fizikai, hanem logikai mértékegysége.

Bájtsorrend

- nagy endián - a bájtok sorrendje egy szóban megegyezik a címsorrenddel
- kis endián – a bájtok sorrendje fordított

Minden rekesz önálló és egyedi **cím**mel rendelkezik, amelyet szigorúan növekvő sorrendben rendelnek a tárhoz.

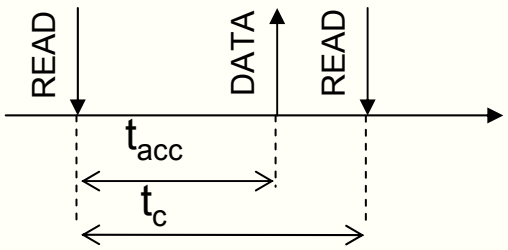
A processzor **címezhető tártartomány**át címvezetékeinek száma határozza meg

- valós tárolókezelés
- virtuális tárolókezelés

Alapvető tárjellemzők

- **tárkapacitás** [byte]
- **szószélesség** – egy tárművelettel elérhető bitek száma

- **tárhozzáférési idő** - az információkéréstől az adat megjelenéséig eltelt idő
- **ciklusidő** - két azonos tárművelet között eltelt idő



- **felvett teljesítmény**
- **átviteli sebesség** – egységnyi idő alatt írható vagy olvasható bitek száma
- **költséghatékonyság** – ár/bit
- **megbízhatóság** – MTBF – rendelkezésre állás

Tárak osztályozása

Alkalmazás szerint
- operatív tár
- háttértár
- speciális tár (puffer tárolók)

Hozzáférés módja szerint
- közvetlen
- soros
- asszociatív (tartalom szerinti)

Funkciója szerint
- írható/olvasható
- csak olvasható

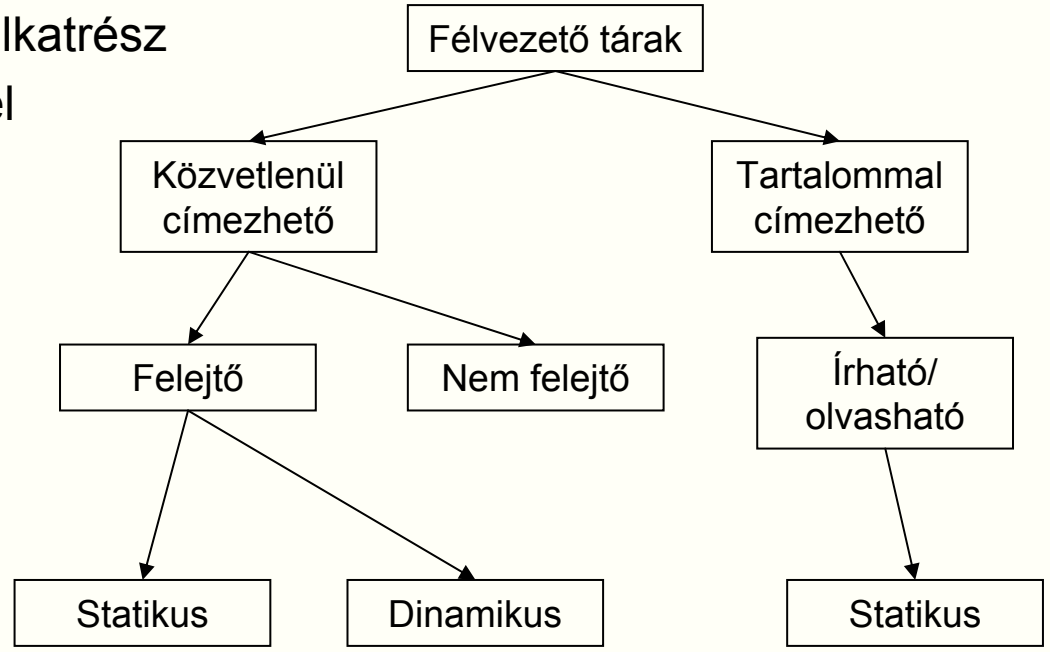
A tápfeszültség kikapcsolása után
- felejtő
- nem felejtő

Tárolási elv szerint
- félvezető
- mágneses
- optikai

Törlés szempontjából
- nem törölhető
- törölhető
  - dinamikus - működés közben törlődő
  - statikus - nem törlődő

Félvezető tárak

- mátrix szervezésűek
- modulrendszerű
- nagy működési sebesség
- kis méret
- nincs benne mechanikus alkatrész
- csekély teljesítményfelvétel
- tömeges gyártás
- kedvező ár



Félvezető tárak

Típus	Funkció	Törlés	Táp-feszültség nélkül	Tipikus alkalmazás
SRAM	Olvasás/írás	Elektromos	Felejt	2-es szintű gyorsítótár
DRAM	Olvasás/írás	Elektromos	Felejt	Operatív tár
ROM	Olvasás	Nem törölhető	Nem felejt	Fix programtár nagysorozatú berendezésekben
PROM	Olvasás	Nem törölhető	Nem felejt	Fix programtár kissorozatú berendezésekben
EPROM	Olvasás	UV fény	Nem felejt	Prototípusok
EEPROM	Olvasás	Elektromos	Nem felejt	Prototípusok
Flash	Olvasás/írás	Elektromos	Nem felejt	Cserélhető tár digitális eszközökhöz

Nem felejtő tárak

ROM Read Only Memory
- programozása a gyártás során történik fotómaszkos eljárással
- a maszk készítése drága, de nagy sorozatú gyártásnál költséghatékony
- tartalma nem változtatható

PROM Programmable ROM
- egyszer programozható elektromos úton, az adatmezőben a sorok és az oszlopok közötti kapcsolatok kiégetésével

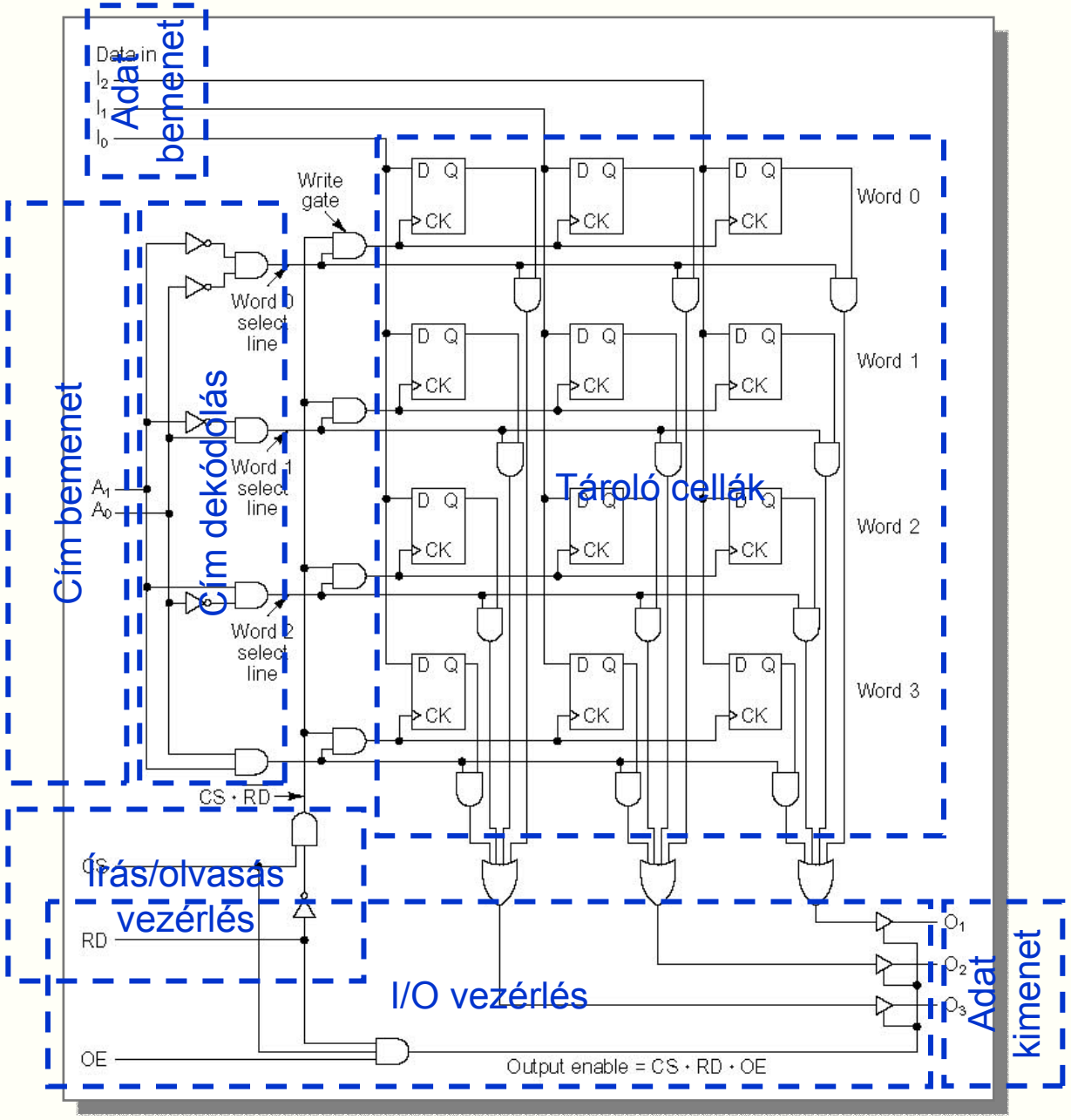
EPROM Erasable PROM
- kvarcüveg ablakon át ultraibolya fénnyel az információtartalom törölhető úgy, hogy az összes bitet 1-re állítjuk
- prototípusoknál gazdaságosabb, mint a PROM
- belső felépítése hasonlít az SRAM-hoz

EEPROM Electrical EPROM
- impulzusokkal törölhető
- nincs szükség a tok kiszerelésére
- kapacitása az EPROM 1/64 része és csak fele olyan gyors

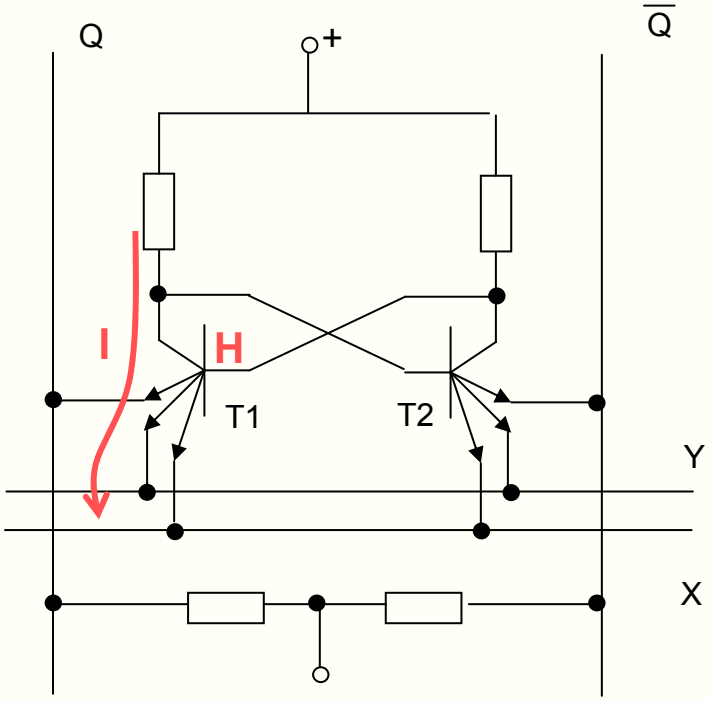
Flash memory
- blokktörlést és -újraírást alkalmaz
- megközelítően 10 ezer törlést bírnak ki

RAM általános felépítése

RAM –
Random Access Memory



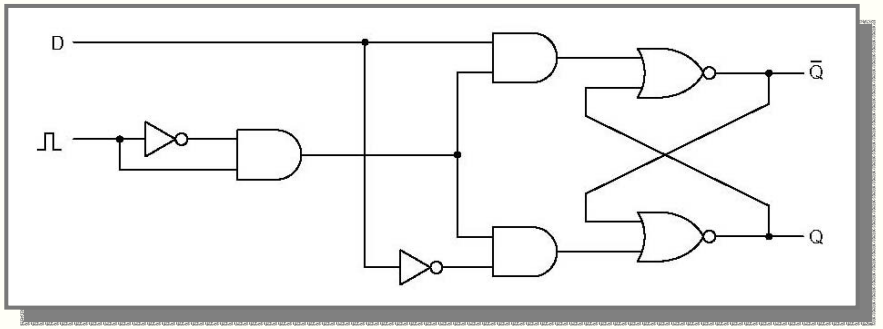
Tároló cella megvalósítása



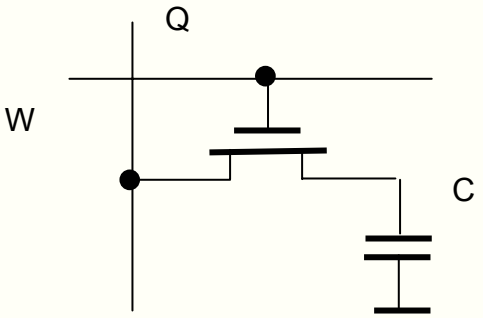
Ha T1 bázisán feszültség (H) van, akkor T1 vezet

Ha X,Y alacsony (L) szinten van, akkor T1 nyit, T2 zár

Ha X,Y magas (H) szinten van, kiolvasás



D-tároló

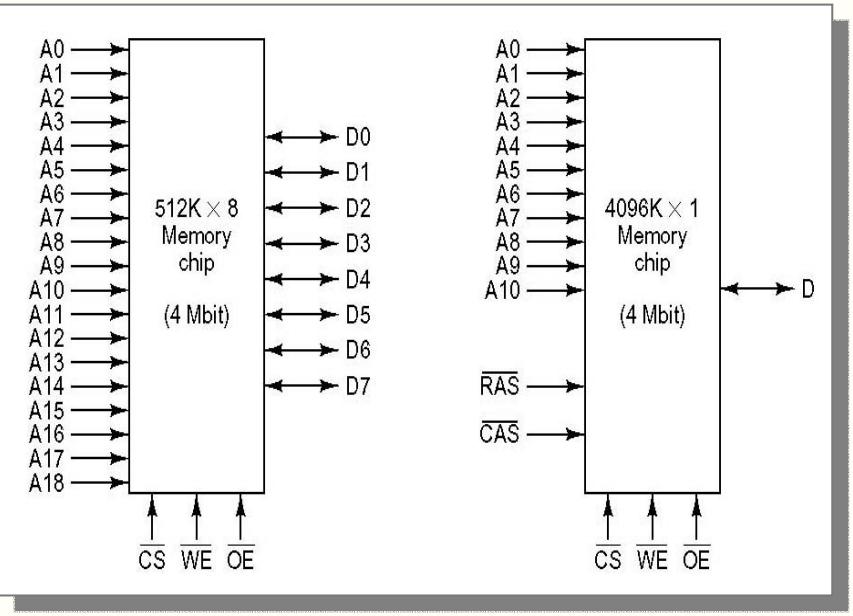


C kondenzátor töltés formájában tárolja az információt

Kiolvasás – a kondenzátor kisül, eredeti tartalmát vissza kell írni

C kis kapacitású, hamar elveszti a töltést, ezért rendszeresen frissíteni kell

Memórialapkák



Bájtszervezésű Bitszervezésű

Vezérlőjelek

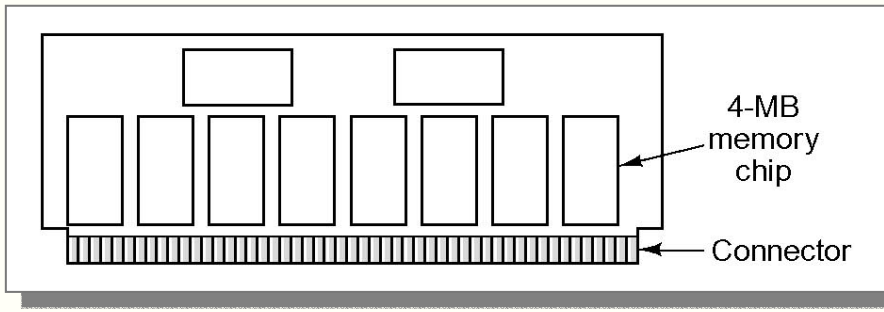
$\overline{CS}$ – memóriamodulok kiválasztása, címszelektálás

$\overline{WE}$ – adatok írásának/olvasásának kiválasztása

$\overline{OE}$ – kimenet engedélyezése, tiltott állapotban leválasztás a buszról

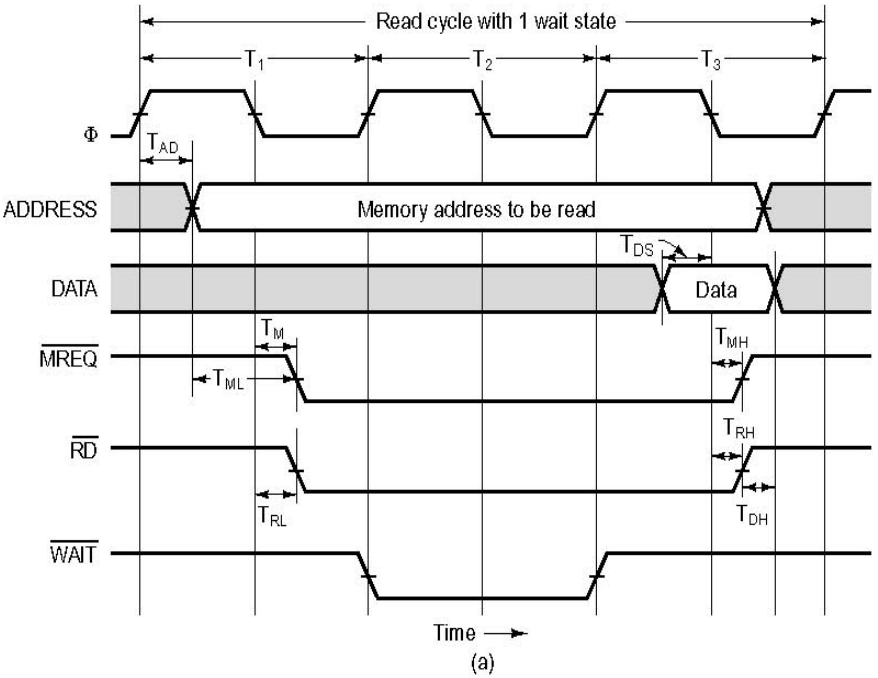
$\overline{RAS}$ – sorcím engedélyezése Row Address Strobe

$\overline{CAS}$ – oszlopcím engedélyezése Column Address Strobe



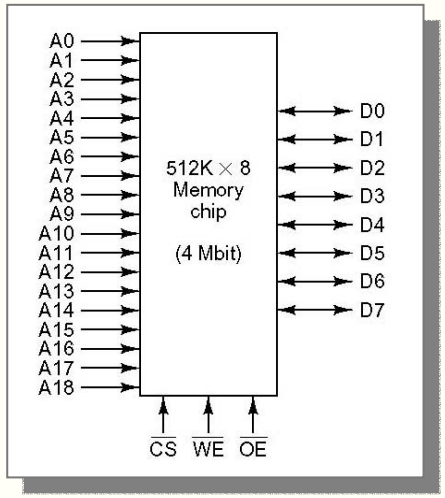
32 MB-os SIMM modul

Statikus RAM
SRAM



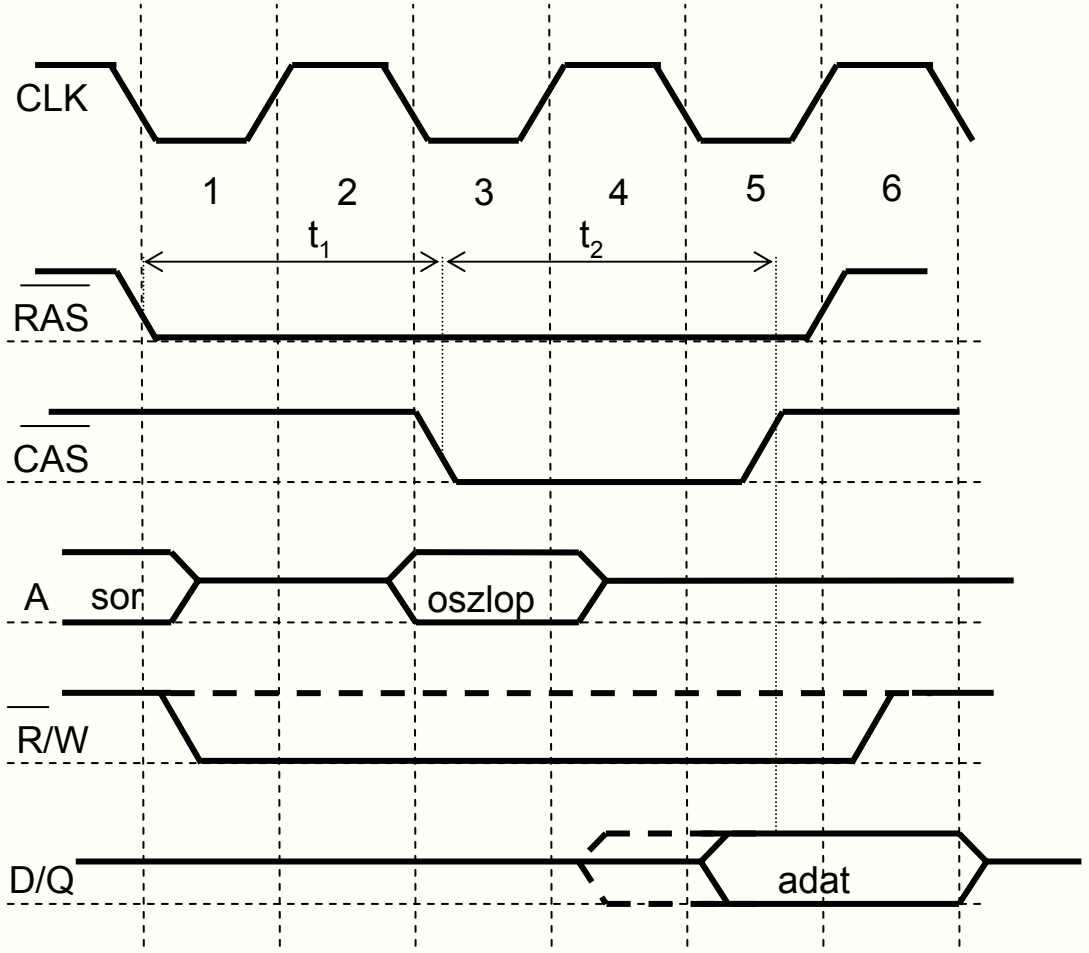
Symbol	Parameter	Min	Max	Unit
T_{AD}	Address output delay		11	nsec
T_{ML}	Address stable prior to $\overline{MREQ}$	6		nsec
T_M	$\overline{MREQ}$ delay from falling edge of Φ in T_1		8	nsec
T_{RL}	RD delay from falling edge of Φ in T_1		8	nsec
T_{DS}	Data setup time prior to falling edge of Φ	5		nsec
T_{MH}	$\overline{MREQ}$ delay from falling edge of Φ in T_3		8	nsec
T_{RH}	$\overline{RD}$ delay from falling edge of Φ in T_3		8	nsec
T_{DH}	Data hold time from negation of $\overline{RD}$	0		nsec

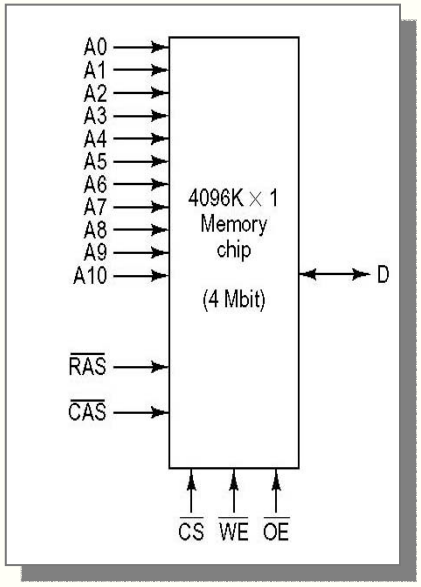
(b)



- tartalmát a tápfeszültség megszűnéséig megtartja
- gyors (15 ns)

Dinamikus RAM
DRAM





- tartalmát a kondenzátor lassú kisülése miatt elveszíti, ezért frissíteni kell (Refresh)
- nagy kapacitású
- lassabb, mint a statikus RAM

DRAM-ok frissítése

- belső frissítésű tárak
- külső logika oldja meg (ez lehet a processzor is, pl. Z80)

Distributed Refresh

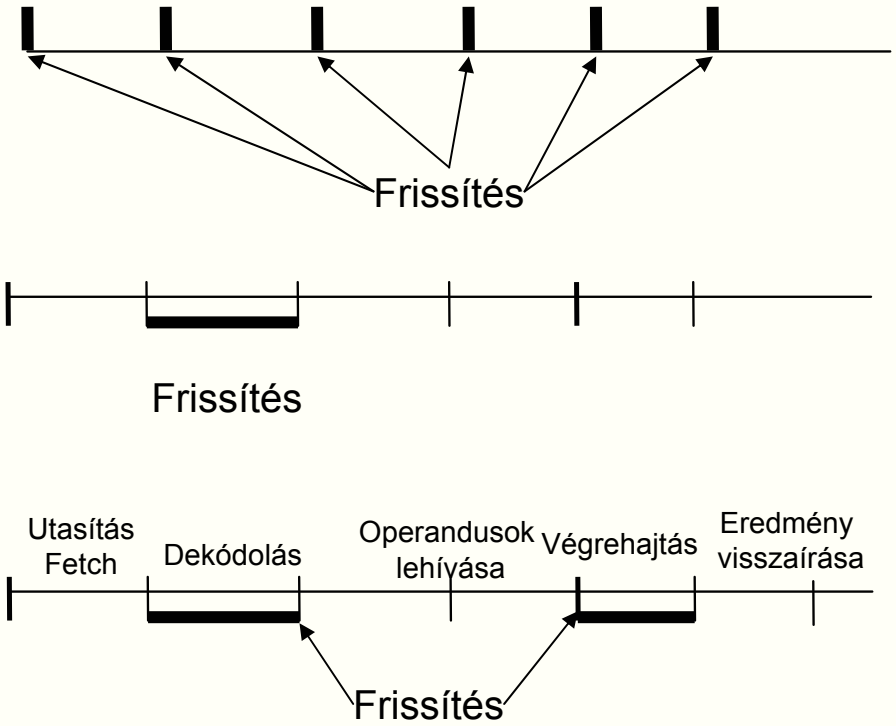
2 μs-onként a tárat kiolvassa, majd visszaírja

Burst Refresh

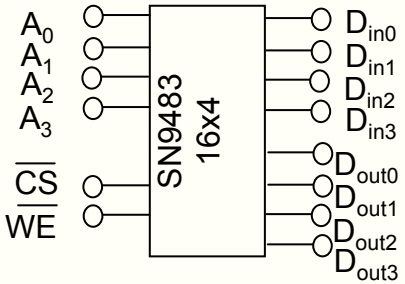
2 μs ideig egy blokkot kiolvas és visszaír

Synchronous (Hidden) Refresh

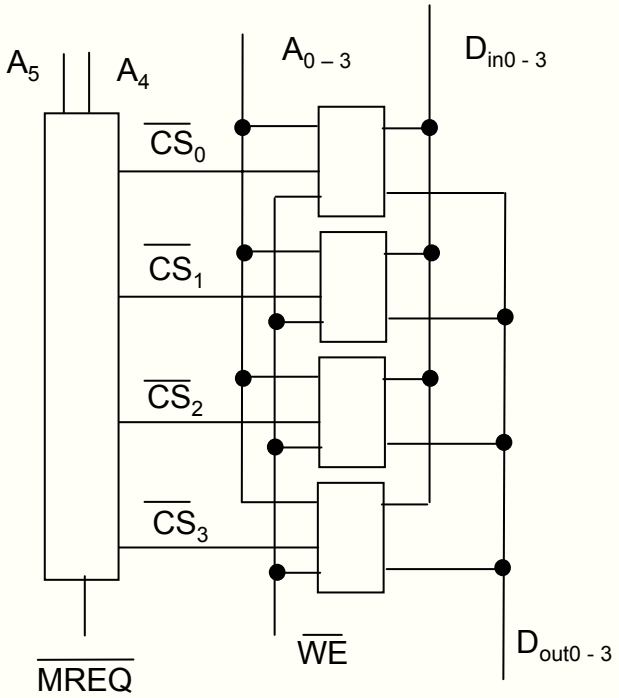
akkor frissít, amikor a processzor nem használja a memóriát



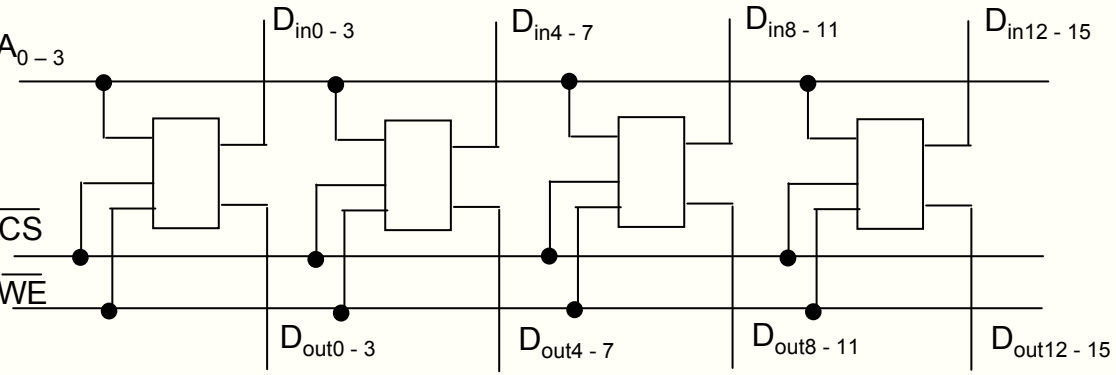
Tárbővítés



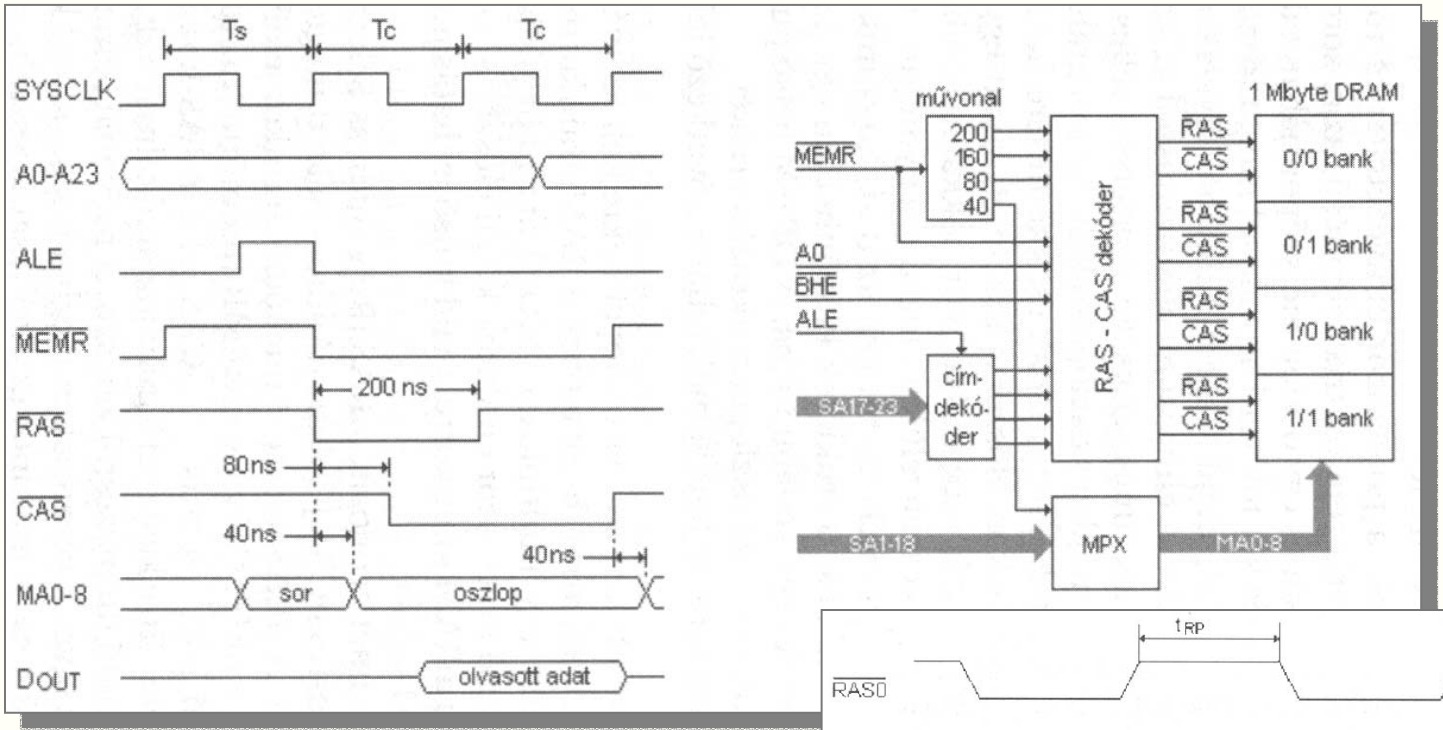
64x4 bites memóriablokk megvalósítása

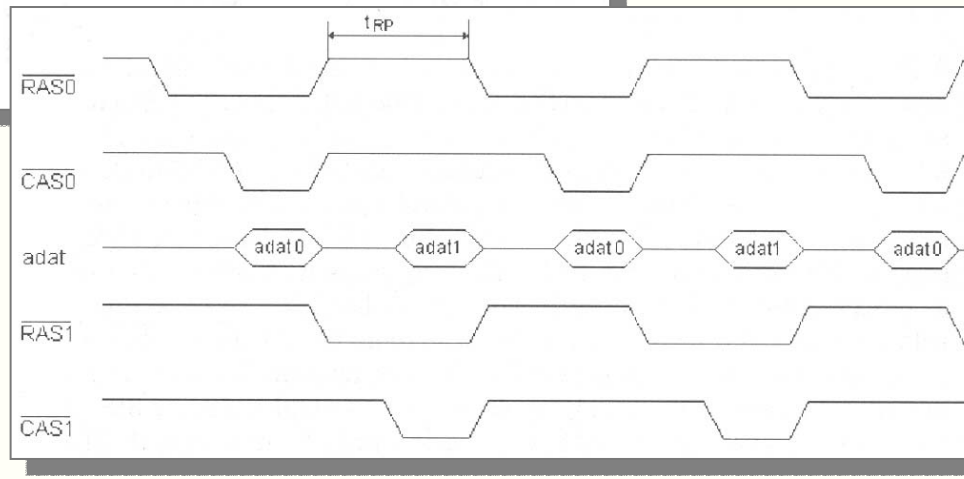


16x16 bites memóriablokk megvalósítása

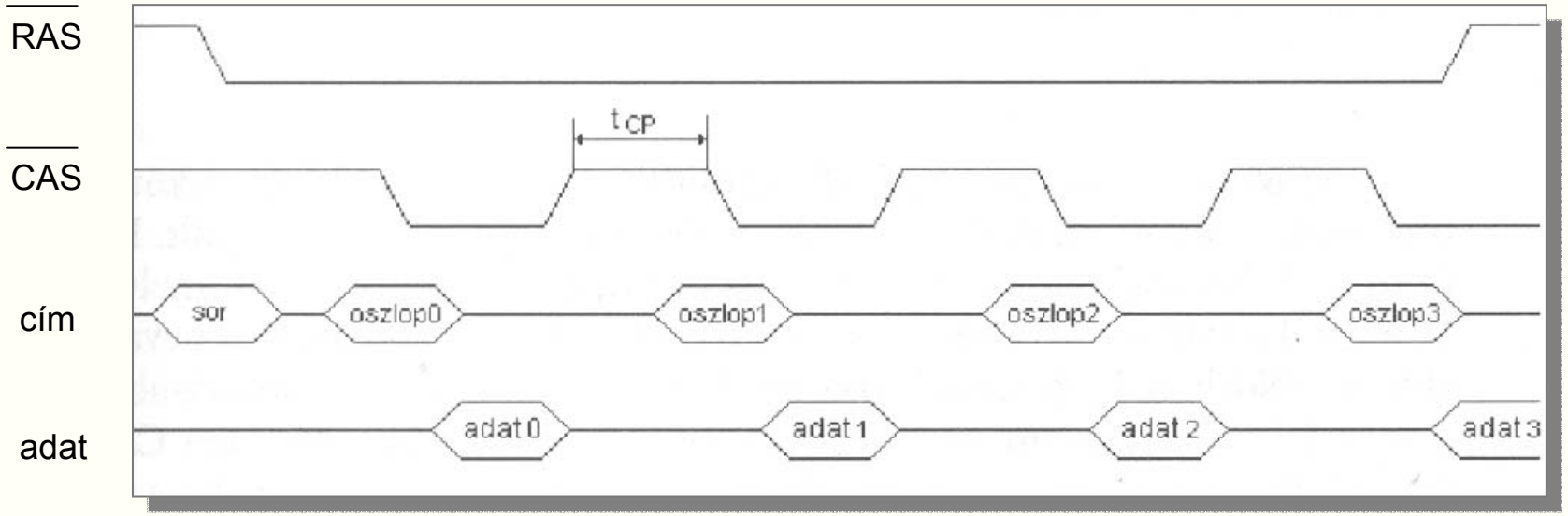


Átlapolt tár-hozzáférés (Interleaving) memóriabankok alkalmazásával





Lapmódú működés



A lapmódú működést három ok korlátozza

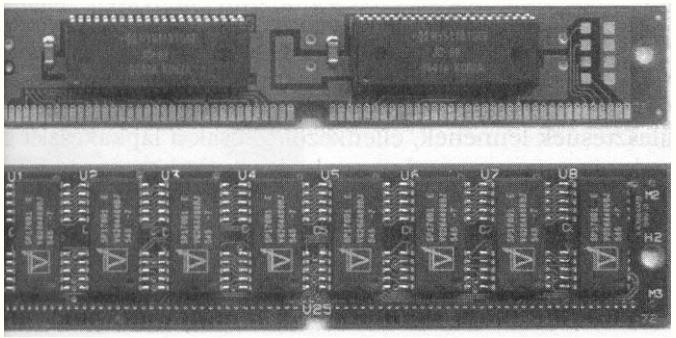
- elértük a lap végét
- nem címfolytonosan olvassuk a memóriát
- tárfrissítés miatt ki kell lépni a lapmódból

DRAM-ok típusai

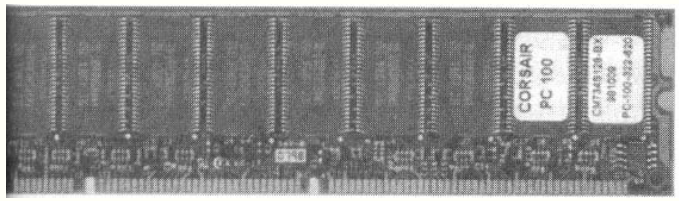
SDRAM (Synchronous DRAM)

- egyetlen szinkron órajel vezérli
- kötegelt hozzáférésű
- meg kell határozni
  - a csoportos átvitel hosszát
  - az átvitel típusát
    - folytonos
    - átlapolt
- futószalag elvű, amely megengedi a következő memóriahivatkozás megkezdését az előző befejezése előtt

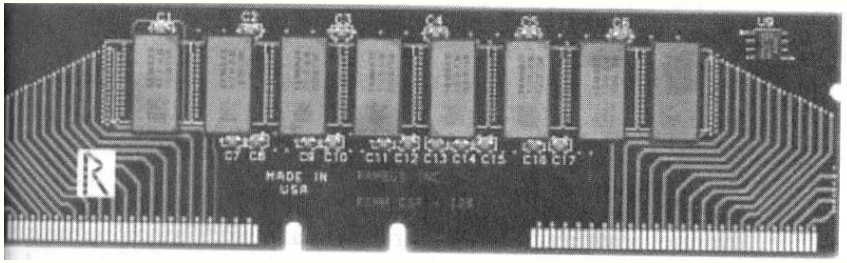
Memóriamodulok



SIMM



DIMM



RIMM

Asszociatív tárak

1	0	0	1

Kulcsregiszter

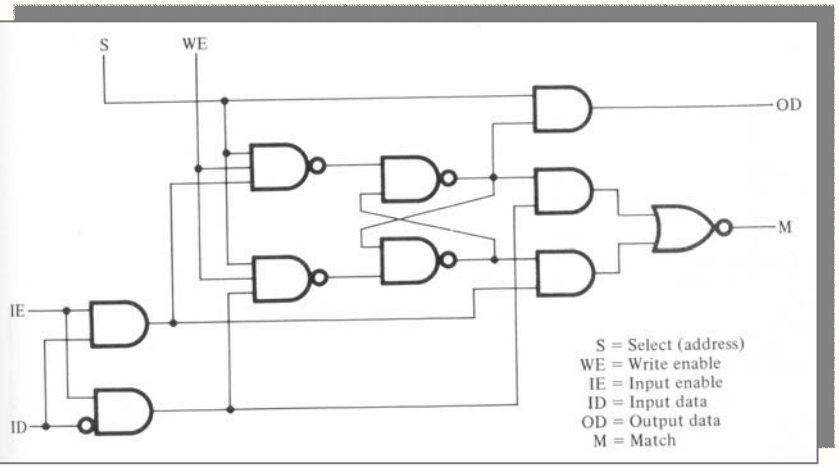
1	x	0	x

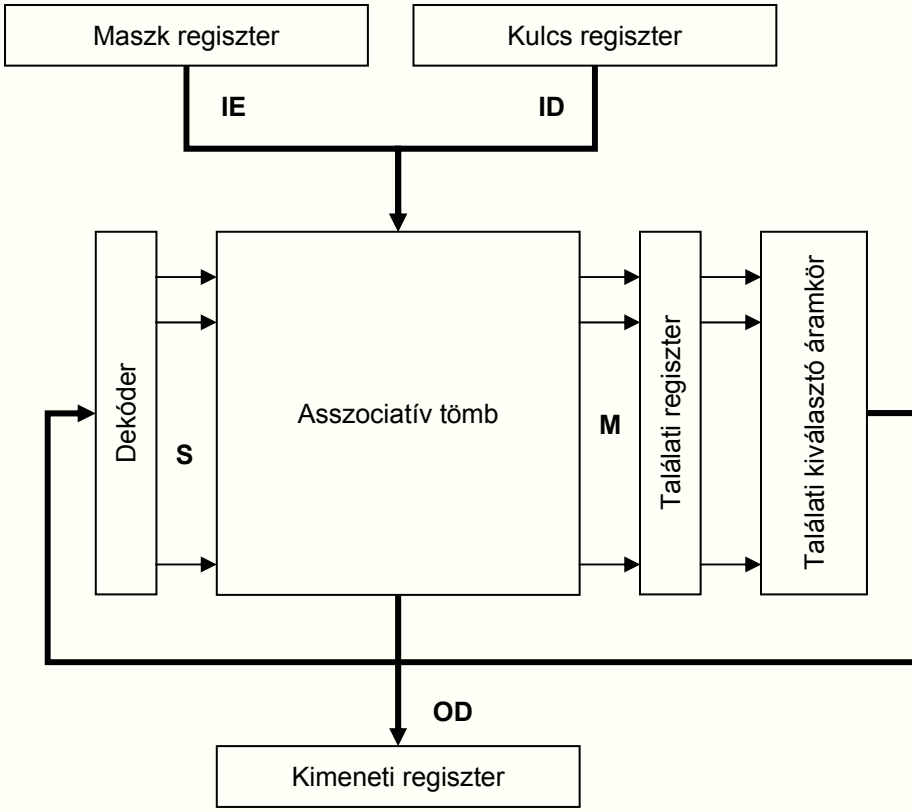
Maszkregiszter

1	1	0	0
0	0	0	0
1	0	0	0
0	0	0	1

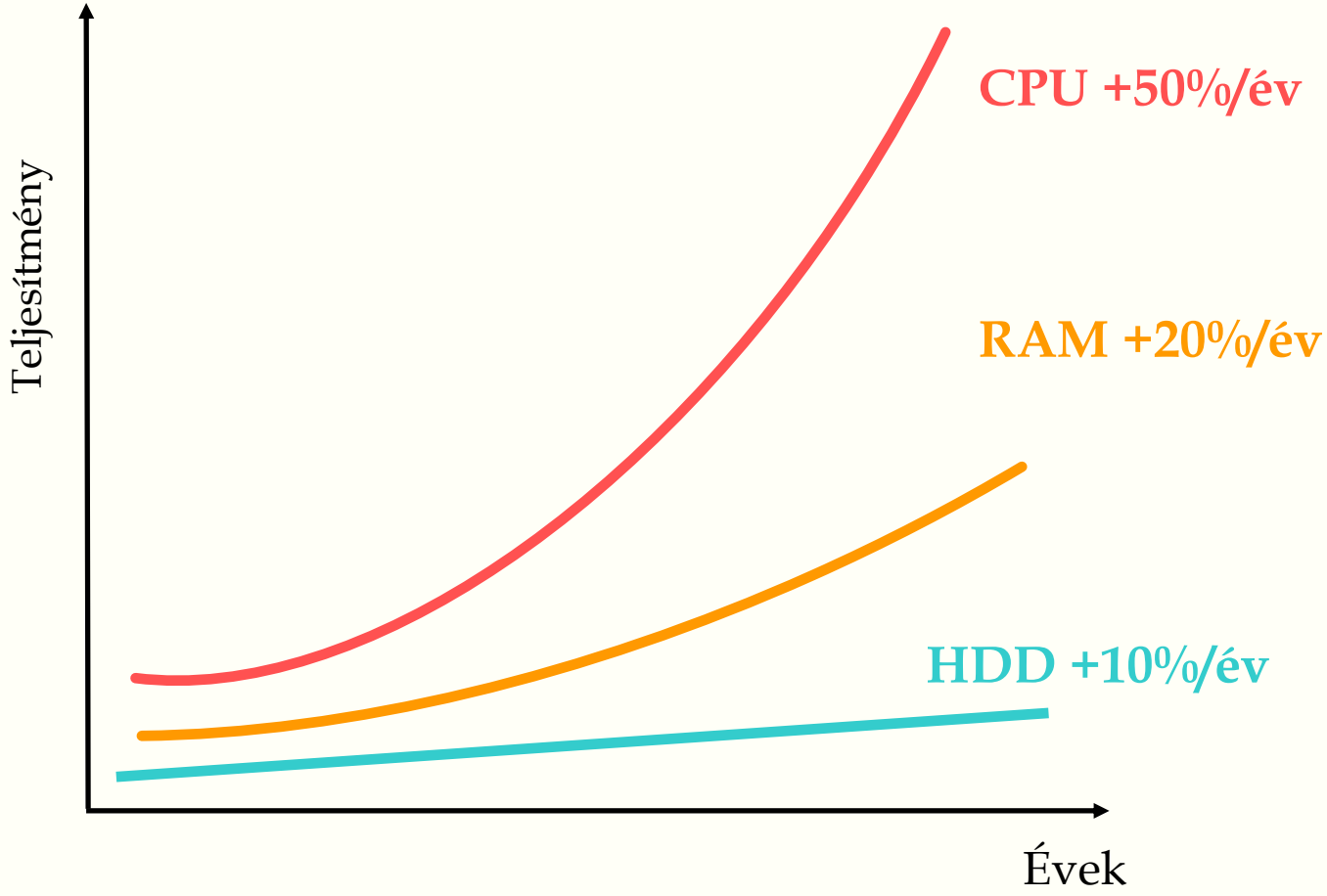
1
0
1
0

Találati regiszter





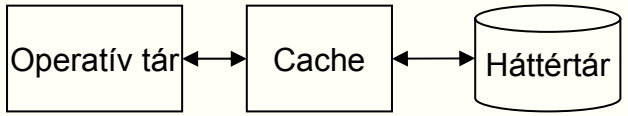
A számítógép egységek teljesítménynövekedése

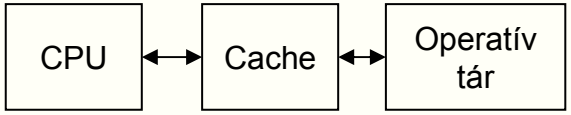


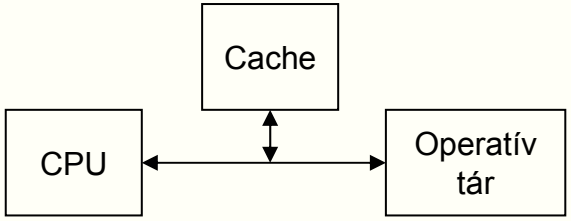
Gyorsítótárak

Gyorsítótár (cache) – utasítások és adatok átmeneti tárolására szolgáló, gyors működésű, a felhasználó számára nem elérhető tár.

Cél az adatforgalom gyorsítása
- Operatív tár és háttér tár között
- CPU és operatív tár között
  - Leválasztó (look-through)
  - Mellérendelt (look-aside)







Elhelyezkedés
- processzoron belül (on-chip cache) – L1 8-32KB
- processzoron kívül (off-chip cache) – L2 64KB–1MB

Tárolt információ
- csak adat
- csak utasítások
- adat és utasítások együtt

Az adatátvitel a cache és a memória között mindig blokkos formájú – **cache sor**

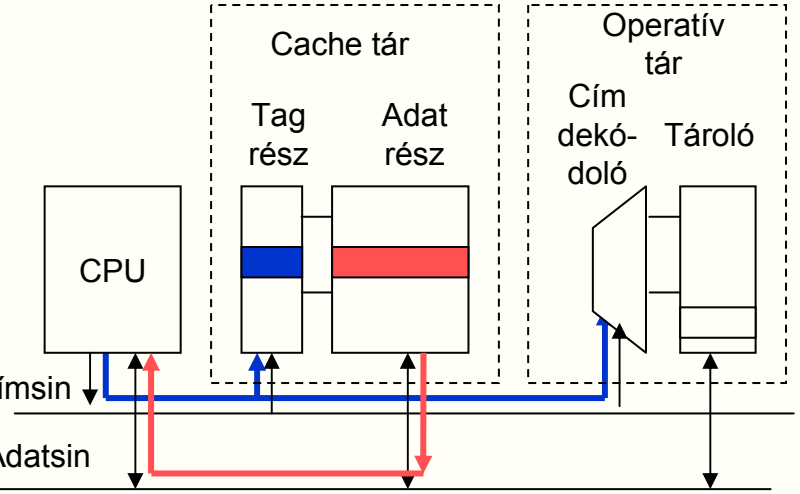
Hatékonyság
- találati arány (**cache hit**)
  - cache mérete
  - szervezési módja

$$h = \frac{találatok\quad száma}{összes\quad tárhozfordulás\quad száma}$$

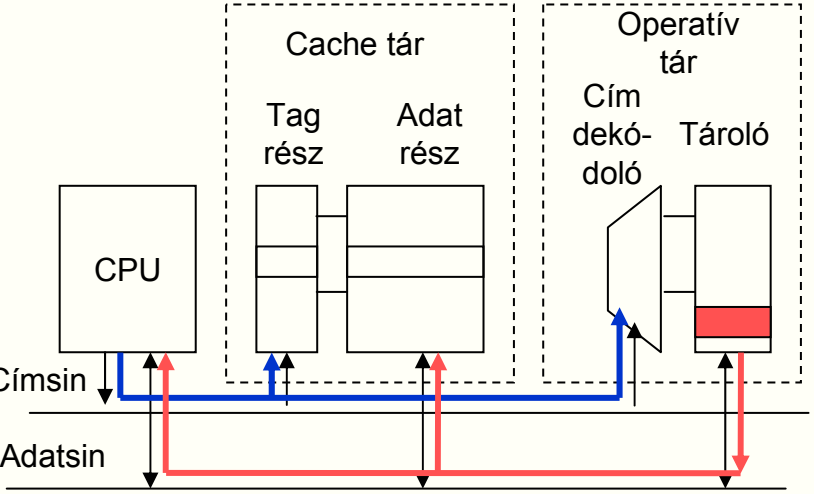
- találati hiba aránya (**cache miss**) (1-h) = max.10 %

Cache koherencia - a cache tár és az operatív tár megfeleltetett részei tartalmának az egyezősége.

Adatmozgatás
cache-hit és cache-miss esetén



cache-hit



cache-miss

t_c *a cache tár elérési ideje* $(\approx 25ns)$

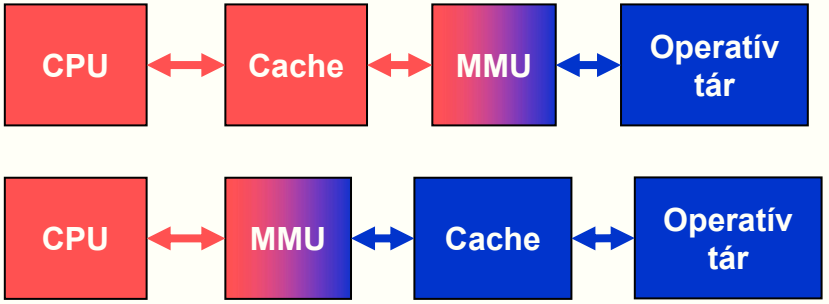
t_{miss} *blokk betöltési ideje a tárból* $(\approx 260ns)$

$$\bar{t}_a = h \cdot t_c + (1-h)t_{miss} =$$

$$= 0{,}95 \cdot 25 + 0{,}05 \cdot 260 = 36{,}75ns$$

Cache tárak

Tag rész – a címnek az a része, amely alapján a visszakeresés történik. Származhat a **virtuális címből** vagy a **fizikai címből**, attól függően, hogy a cache tár a processzor és a címfordító egység (MMU), vagy az MMU és az operatív tár között helyezkedik-e el.



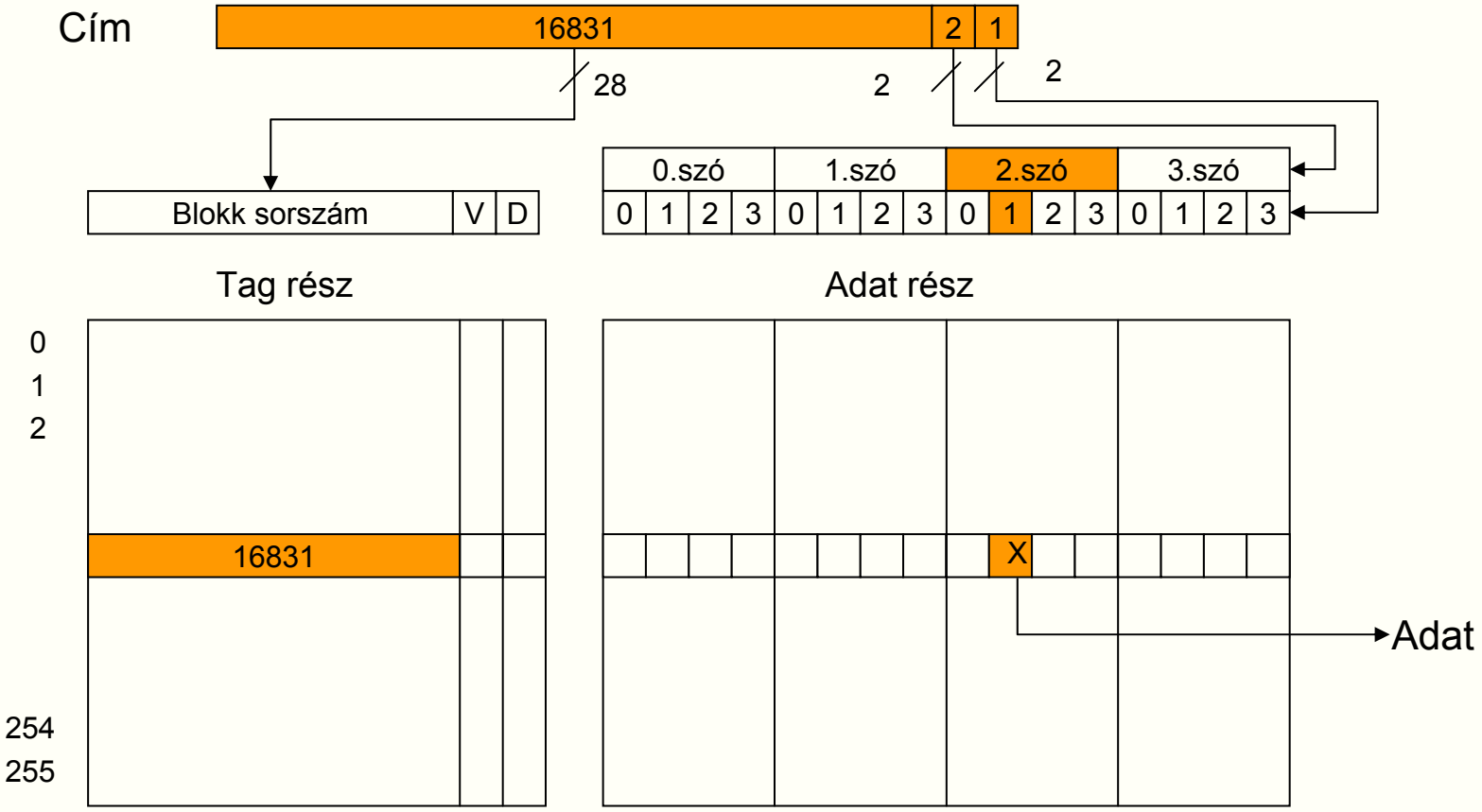
Állapotjelző bitek

- Valid bit (V) – a cache tár tartalmának érvényességét jelzi. Értéke új adat betöltésekor 1.
- Dirty bit (D) – a blokk valamely részének módosított állapotát jelzi. Az ilyen blokk helyére nem lehet új blokkot betölteni, előbb a blokkot vissza kell írni az operatív tárba.

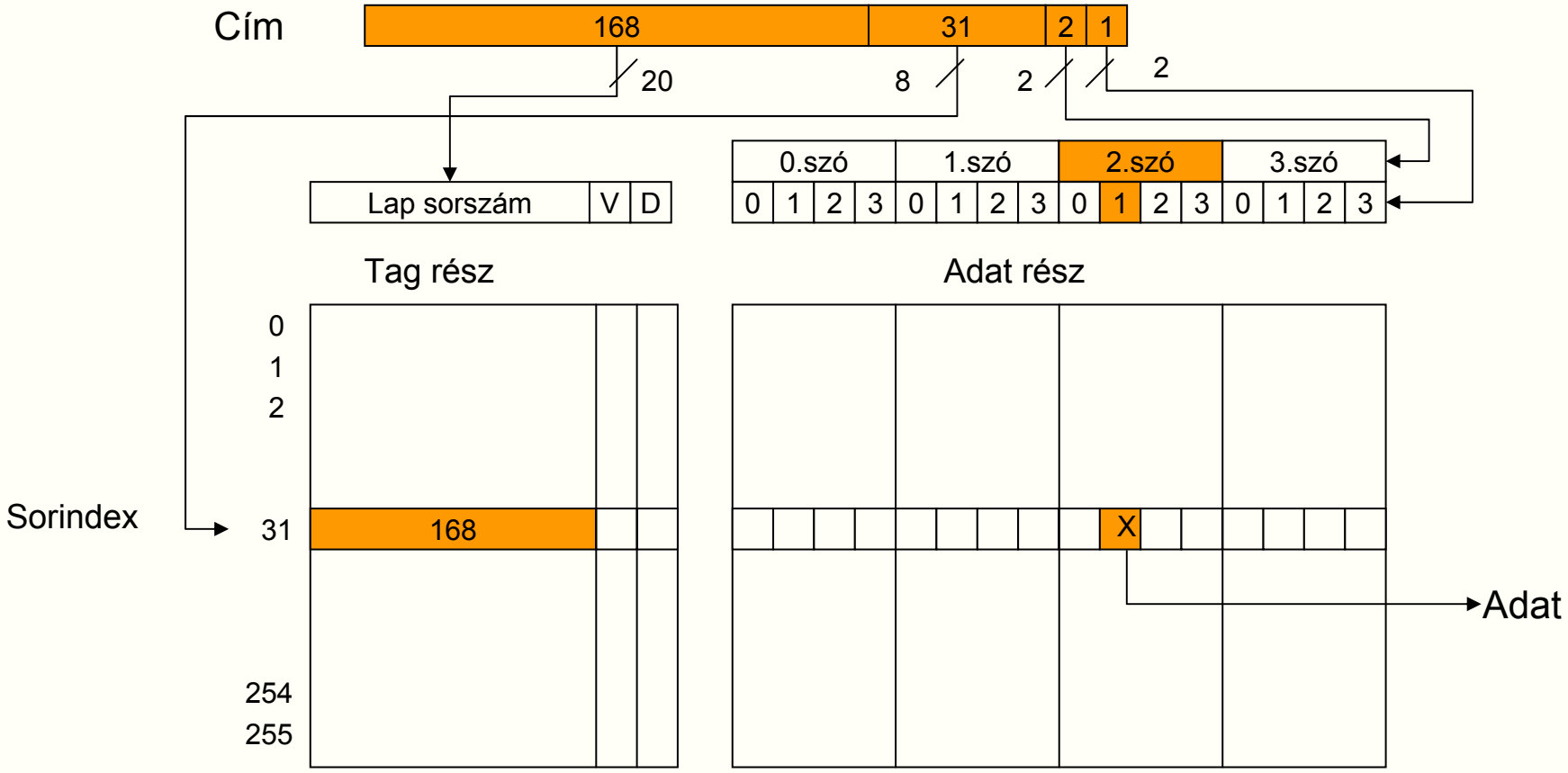
Cache tárak jellemzői

- cache méret –az elhelyezkedéstől függően általában 8kB – 1MB
- blokk méret – az operatív tár és a cache tár között egy egységben mozgatott adatmennyiség 4-32 byte (utasításoknál nagyobb, adatok esetében kisebb)
- sorméret – egy összehasonlítással kijelölhető adatmennyiség (általában a blokkal megegyező vagy annál kisebb)
- helyettesítési algoritmus (replacement policy) – új blokk betöltésekor a kicserélhető blokk meghatározásának algoritmusa
- adataktualizálási módszer (write strategy) – a módosítandó adatok cache tárba és az operatív tárba írásának módszere
- adategyezőség biztosítása (coherency mechanism) – az operatív tár és a cache tár tartalmának egyezőségét biztosító módszer

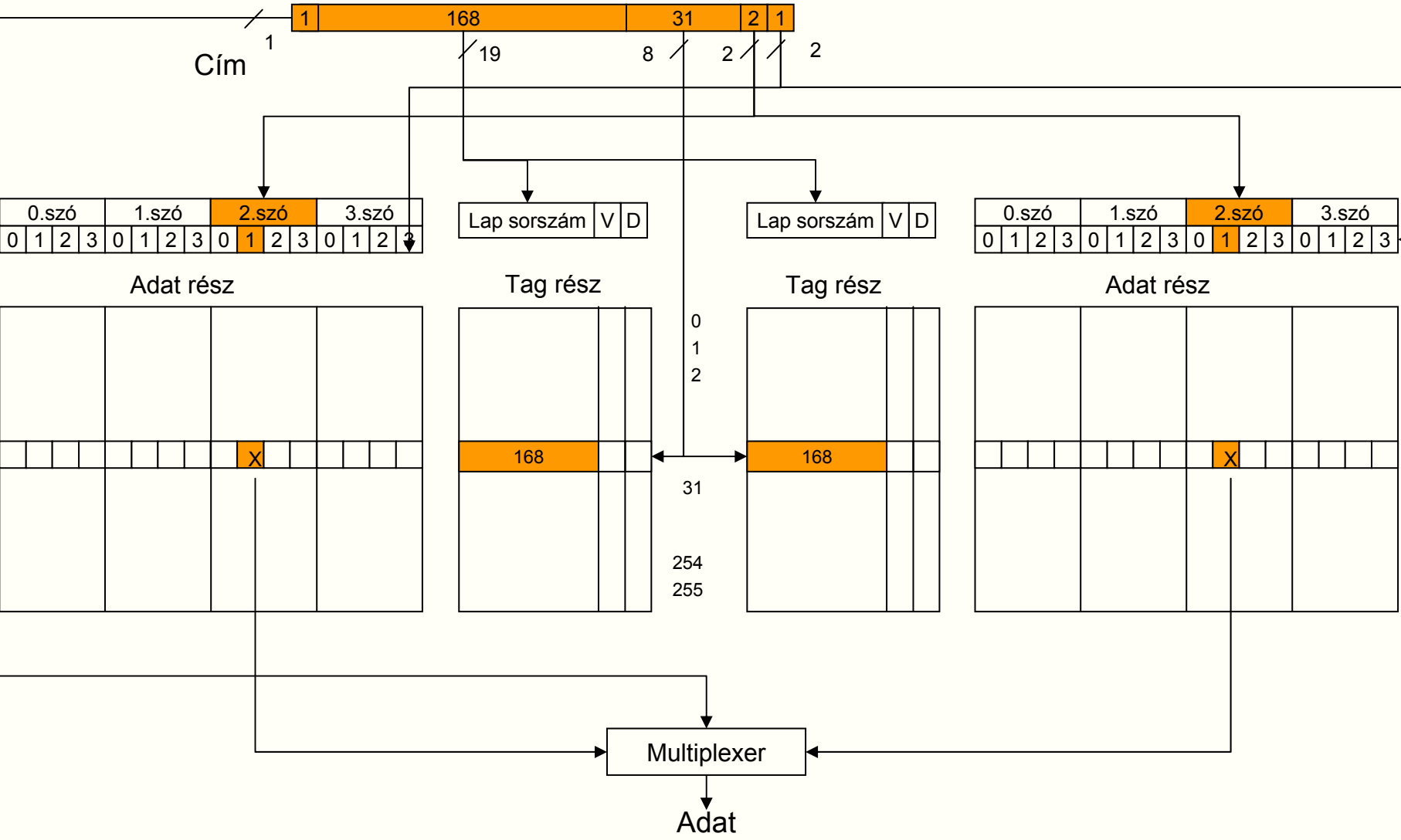
Teljesen asszociatív cache tár (fully associative)



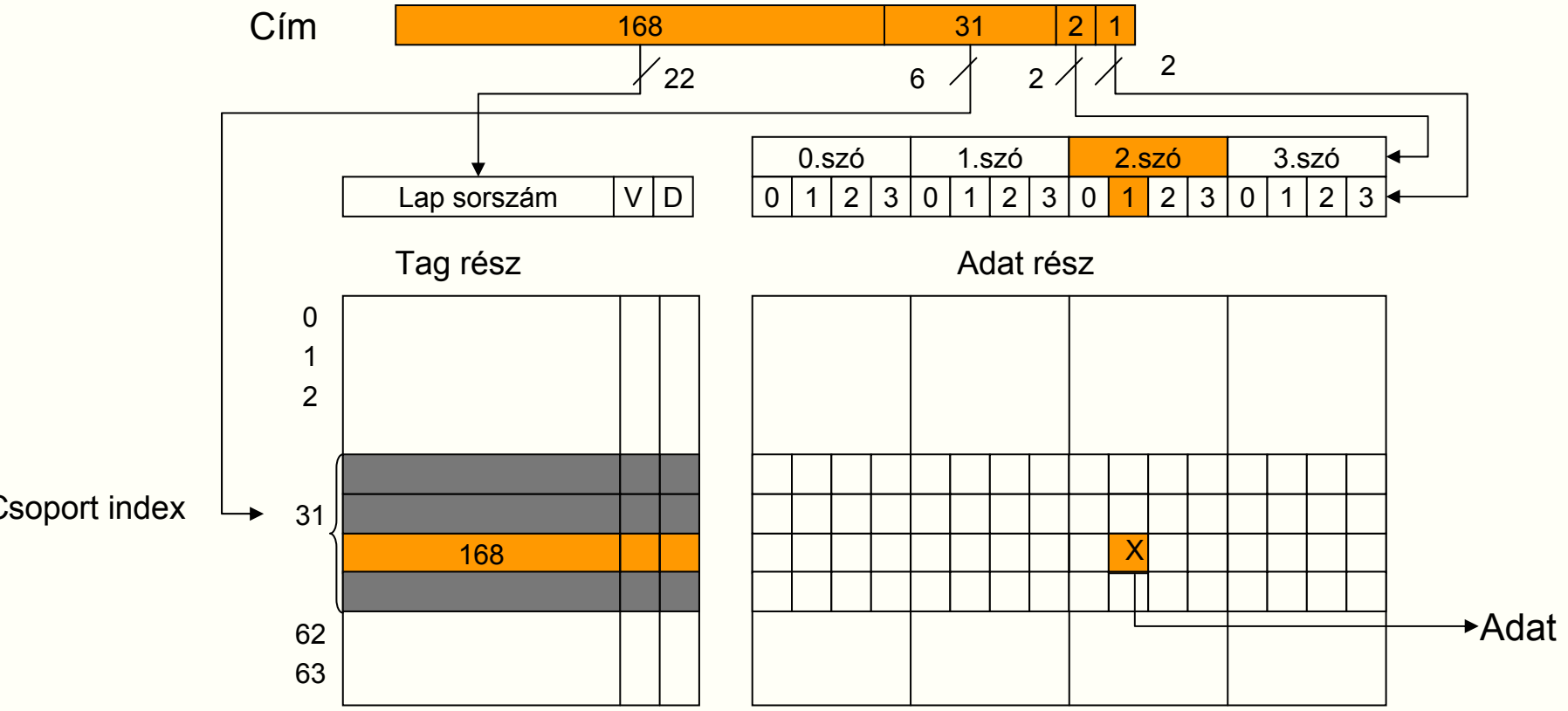
Közvetlen leképezésű cache tár (direct mapped)



Kétutas közvetlen leképezésű cache tár



Csoport asszociatív cache tár (set associative)



Tartalom betöltése

- aktuális igény felmerülésekor (demand fetching) – a keresett adat nincs a cache-ben, a blokk a cache-be töltődik. Párhuzamos a processzor is megkapja az adatot (leggyakrabban alkalmazott eljárás).
- következő blokk előkészítése (prefetching) – automatikusan előkészíti az olvasott után következő blokkot (kis blokk méretek esetén célszerű alkalmazni).
- szelektív előkészítés (selective fetching) – az írható adatokat a memóriában tartjuk, a cache-be csak olyan adatok kerülnek, amelyeket nem kell módosítani (bonyolult alkalmazni).

Aktualizálás

Adategyezőség (Cache koherencia) biztosítása - garantálni kell, hogy a processzor mindig aktuális adattartalomhoz fér hozzá.

- azonnali átírás (write trough)
- visszaírás (write back)
- egyszeri átírás (write once)

Egyes processzorok esetén (Power PC 601, i486) szoftver úton választható az átírásos vagy visszaírásos aktualizálási módszer egyike.

Azonnali átírás (write trough)

A módosított byte azonnal beírásra kerül az operatív tárba, függetlenül attól, hogy a blokk a cache van-e. Az átírás gyorsítható 1-2 blokkos puffer használatával, amelyben sorban állnak a memóriába írandó byte-ok (buffered write through).

- cache-hit esetén
  a cache tartalma is aktualizálódik (write-hit)
- cache-miss esetén
  - az átírást követi egy olvasási kísérlet read-miss, amelynek eredményeként a blokk betöltődik a cache-be
  - a blokk betöltése után aktualizálja a cache-t és a memóriát is (write through with write allocation)
  - az átírást nem követi betöltés (posted write through). Leggyakrabban ezt a módszert alkalmazzák.

Visszaírás
(write back)

A visszaírási eljárás

- cache-hit esetén
  csak a cache tartalmát aktualizálja, az operatív tár tartalmát csak a blokk cseréjekor módosítja
- cache-miss esetén
  - csak az operatív tár tartalmát aktualizálja vagy
  - a blokkot beolvassa, utána aktualizálja, visszaírás az operatív tárba csak a blokk cseréjekor történik. Ez a gyakrabban alkalmazott eljárás.

A visszaírási módszer hátránya, hogy a cache és az operatív tartalma nem mindig egyezik meg. Az átírás elmaradása miatt működése gyorsabb, és csökken a busz terheltsége.

Egyszeri átírás (write once)

A módosított byte első cache-be írásakor az operatív tár tartalmát is módosítja a processzor, majd a továbbiakban már csak a cache-ben aktualizálja a tartalmat. Az operatív tár aktualizálása csak a blokk cseréjekor történik.

Előnye

- a busz terhelése kisebb, mint az átírásos módszeré
- az operatív tár aktualizálási gyakorisága is megfelelő, gyakoribb, mint a visszaírásos módszer esetén

Helyettesítési eljárások

A cache tárak hatékonyságát a hozzáférési idő mellett az határozza meg, minél ritkábban kell a tartalmát cserélni. Ez függ

- a cache tár méretétől
- az alkalmazott helyettesítése stratégiától

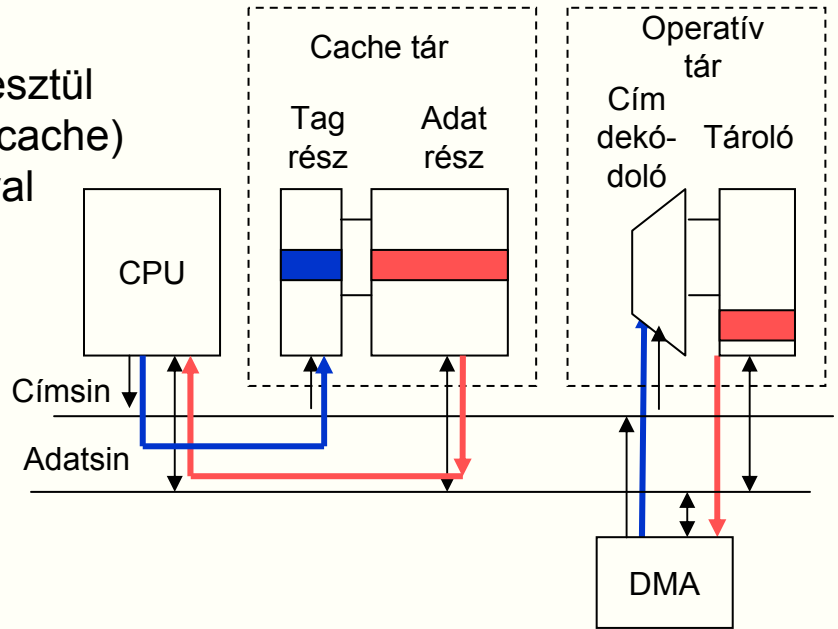
A cache táraknál és a virtuális címzésű táraknál hasonló helyettesítési eljárásokat alkalmaznak

- véletlenszerű (random)
  A cserélendő blokkot egy véletlenszám-generátor által szolgáltatott sorszám jelöli ki. A véletlen szám előállításához a számítógép rendszeróráját használják.
  Egyszerű eljárás, leghatékonyabban a teljesen asszociatív cache tárak esetében.
  Hátránya, hogy nem veszi figyelembe a blokkok tényleges használati gyakoriságát.
- FIFO (first in first out)
  A legrégebben betöltött blokkot cseréli ki. Egy számlálóval valósítják meg, amelynek tartalma a cserélendő blokkra mutat. Csoport-asszociatív tár esetén minden csoporthoz tartozik egy számláló.
  Egyszerű és olcsó megoldás, viszont nem garantált, hogy a legrégebben betöltött blokk a legkevésbé használt.
- legkevésbé használt (LRU - least recently used) blokkok cseréje

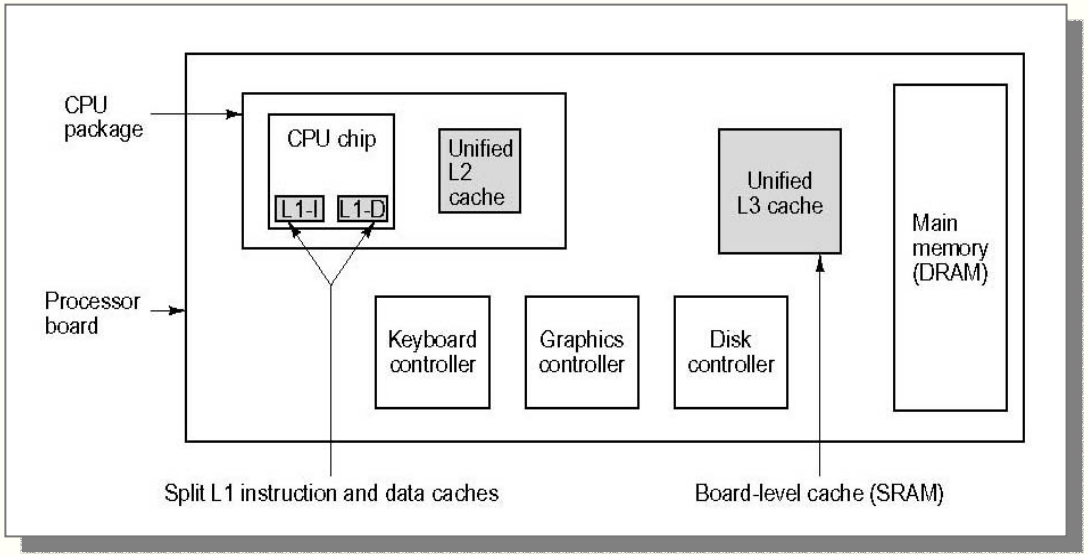
Adategyezőség biztosítása

Probléma

- a processzor olyan beírási módszert alkalmaz, amely csak időnként aktualizálja az operatív tár tartalmát
- az operatív tárat közben más egység olvashatja/módosíthatja
  - DMA-n keresztül lebonyolított I/O művelet
    - a buszon megjelenő címek ellenőrzése – szaglászó cache (snooping cache)
    - adatátviteli művelet cache-en keresztül történő lebonyolítása (I/O through cache) többkapus cache tár alkalmazásával
  - lapváltás virtuális lapkezelésnél



Többszintű gyorsítótárak



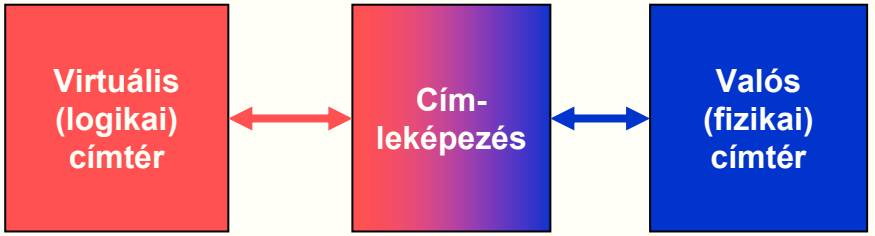
1. **szintű gyorsítótár** (szétválasztott gyorsítótár) L1-I utasításokat, L1-D adatokat tárol. A CPU lapkán helyezkednek el, méretük 16-64 KB
2. **szintű gyorsítótár** nincs rajta a lapkán, de a CPU sethez tartozik, általában egyesített (adatokat és utasításokat is tartalmaz). L2 jellemző mérete 512KB-1MB
3. **szintű gyorsítótár** a processzor alaplapon található és néhány MB SRAM-ot tartalmaz.

L1 teljes tartalma benne van L2-ben, és L2 teljes tartalma megtalálható L3-ban.

Virtuális tárkezelés

Oka

- operatív tár véges kapacitása
  nem növelhető olyan ütemben, ahogyan a programok és az adatok mérete növekszik
- gazdaságossági megfontolások
  nagy mennyiségű adatot olcsó adathordozón célszerű tárolni



- logikai címtér
  amit a programozó lát
- fizikai címtér
  rendelkezésre álló megcímezhető memória
- címleképezés (címfordítás)
  tárolókezelő rendszer (MMU -memory manager unit) feladata a virtuális címek átalakítása valós címekké

Virtuális tárkezelés módszerei

- lapozás
  A lapok olyan adatblokkok, amelyeknek a mérete azonos és rögzített, általában 512 byte – 64KB.
- szegmentálás
  A szegmens olyan adatblokk, amelynek a mérete nem rögzített, választható, lehet átlapolható osztott felhasználás céljából.
- lapozásos szegmentálás
  Szegmentáláskor a tár elaprózódik. A tár rugalmasabban kezelhető a lapozási technika bevezetésével.

A szegmens-, illetve laptáblázatok a következő adatokat tartalmazzák

- a logikai blokk sorszáma
- a blokk kezdőcíme az operatív tárban
- a blokk mérete szegmentált tárkezelésnél
- hozzáférési jogok
- állapotjelzők
- a blokk háttértárbeli fizikai címe (sáv, szektor)

Lapozás és szegmentálás

	Lapozás	Szegmentálás
Tudnia kell róla a programozónak?	Nem	Igen
Hány lineáris címtartomány létezik?	1	Sok
Meghaladja-e a virtuális címtartomány mérete a fizikai tartományt?	Igen	Igen
Könnyen kezelhetők-e a változó méretű táblák?	Nem	Igen
Miért alkalmazzák?	Nagy memória szimulálása	Több címtartomány biztosítása

Virtuális cím kiszámítása

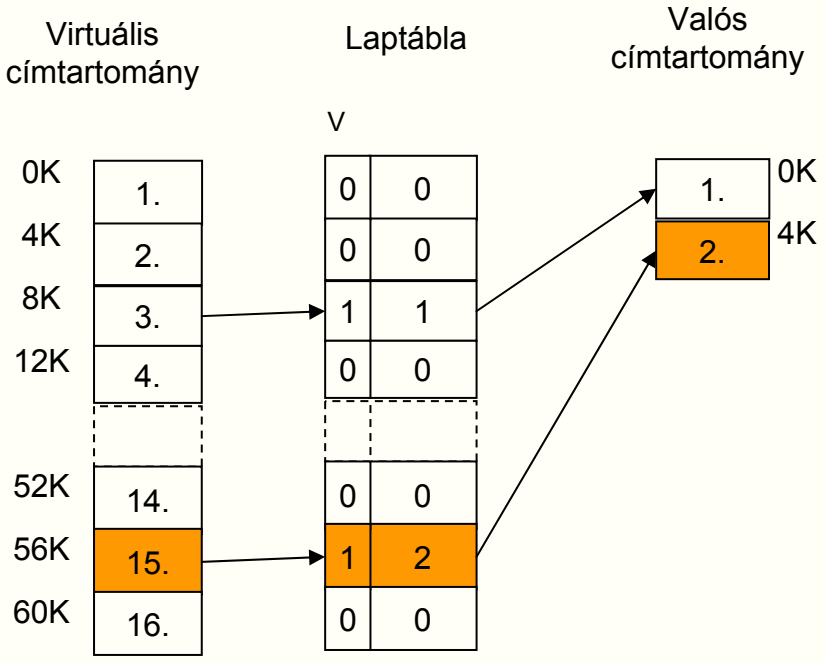
Legyen a
- lapkeretek mérete 4k
- virtuális címtartomány 64k
- fizikai címtartomány 8k

valós cím = lapkeret kezdőcíme + relatív cím

virtuális cím = 59124

relatív cím = 59124-14*4096 = 1780

valós cím = 4096+ 1780= 5876



Lapozás

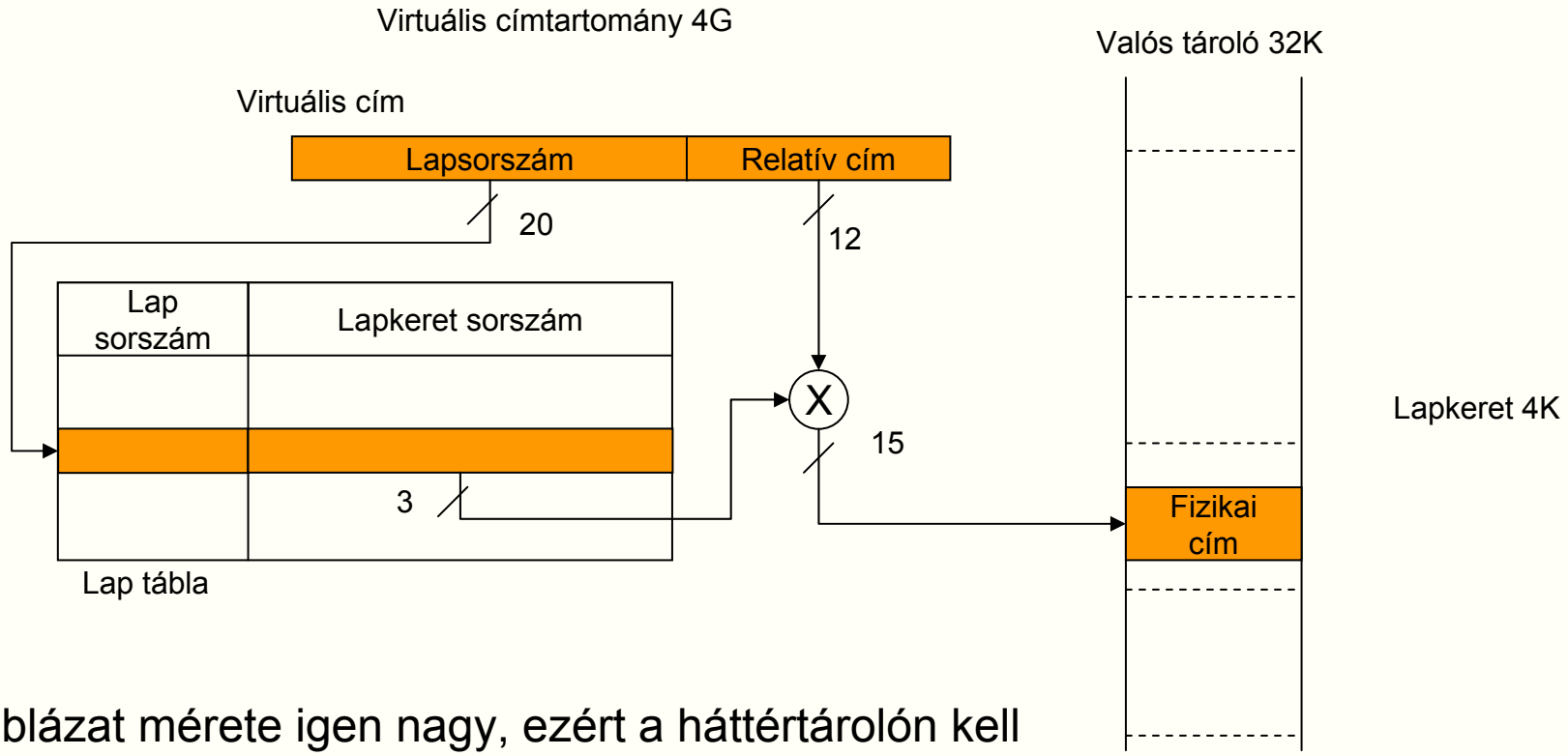
Lapozásnál az egyes lapok csak a lapkeretek által meghatározott helyre kerülhetnek. A lapozás egy összefüggő virtuális címteret képez le.

Lapbetöltés

- **felmerülő igény** (demand fetching) esetén történik. A hiányzó adat detektálása (page fault) lapváltási eljárást indít el. Az aktuális utasítás végrehajtása felfüggesztésre kerül
  - az **utasítás folytatása** a felfüggesztési fázistól (leggyakrabban az operandusok előkészítése). Ebben az esetben az aktuális állapotjelzőket el kell menteni, majd folytatáskor vissza kell tölteni.
  - az **utasítás újrakezdése** a lapváltás után. Vissza kell állítani az utasítás megkezdése előtti állapotot. Kevesebb állapotjelzőt kell elmenteni. Gyakrabban alkalmazott eljárás.

Amennyiben nincs üres lapkeret, a felszabadítandó lapkeretet ki kell választani.

Egylépcsős lapcím kiszámítás



A laptáblázat mérete igen nagy, ezért a háttértárolón kell elhelyezni. A leggyakrabban használt lapok adatait a TLB tartalmazza.

A fizikai cím a lapkeret sorszámának és a relatív címnek az összefűzésével áll elő.

Lap helyettesítési eljárás

- véletlenszerű választás
- legrégebben betöltött (FIFO)
- legrégebbi, nem használt
- adott idő alatt nem használt
- legkevésbé használt (LRU)
  lap cseréje

Az aktuálisan használt lapok (**munkahalmaz** - working set) a legutóbbi *k*

memóriahivatkozásban használt lapok halmaza.

Ha a munkahalmaz nagyobb a rendelkezésre álló lapkeretek számánál, a program folyamatosan laphibát generál, amit **vergődésnek** (thrashing) nevezünk.

Nagy lapkeretek esetén belső elaprózódás jön létre. Kisebb lapkeretek választásánál csökken a vergődés esélye, viszont kevésbé hatékony a lapok átvitele a tár és a háttértároló között.

Amennyiben az eltávolítandó lapon nem történt változtatást, nem kell a tartalmát visszaírni a háttértárolóra.

Szegmentálás

A szegmens olyan logikai egység, amely

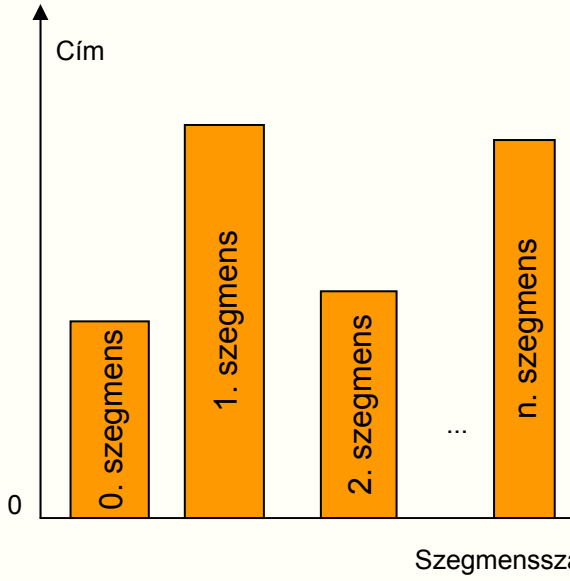
- a programozó számára látható
- külön címtartománnyal rendelkezik (eljárás, tömb, verem, stb)
- hossza változhat a végrehajtás során.

A szegmentálás elősegíti a különböző programok közötti kód- és adatmegosztást.

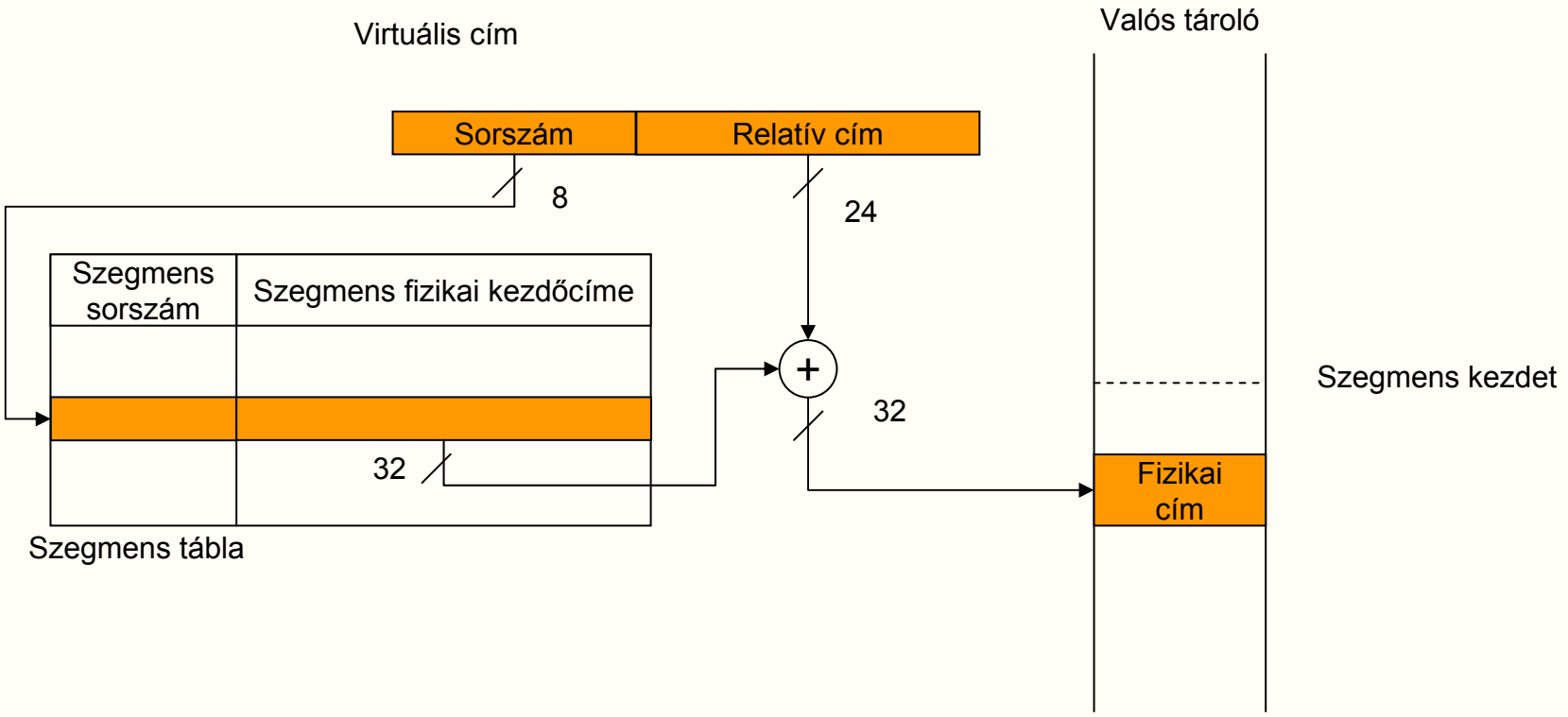
Megvalósítása

- cseréléssel (swapping)
- lapozással (paging)

Cserénél a nem azonos szegmensméretek miatt idővel külső elaprózódás jön létre. Ezt összepréseléssel (garbage collection) szüntetik meg.



Egylépcsős szegmenscím kiszámítás



A virtuális cím felső 8 bitje (beágyazott sorszám) jelöli ki a táblázat megfelelő sorát, amelyben a szegmens kezdetének fizikai címe található. A fizikai cím a szegmens kezdő címének és a relatív címnek az összege.

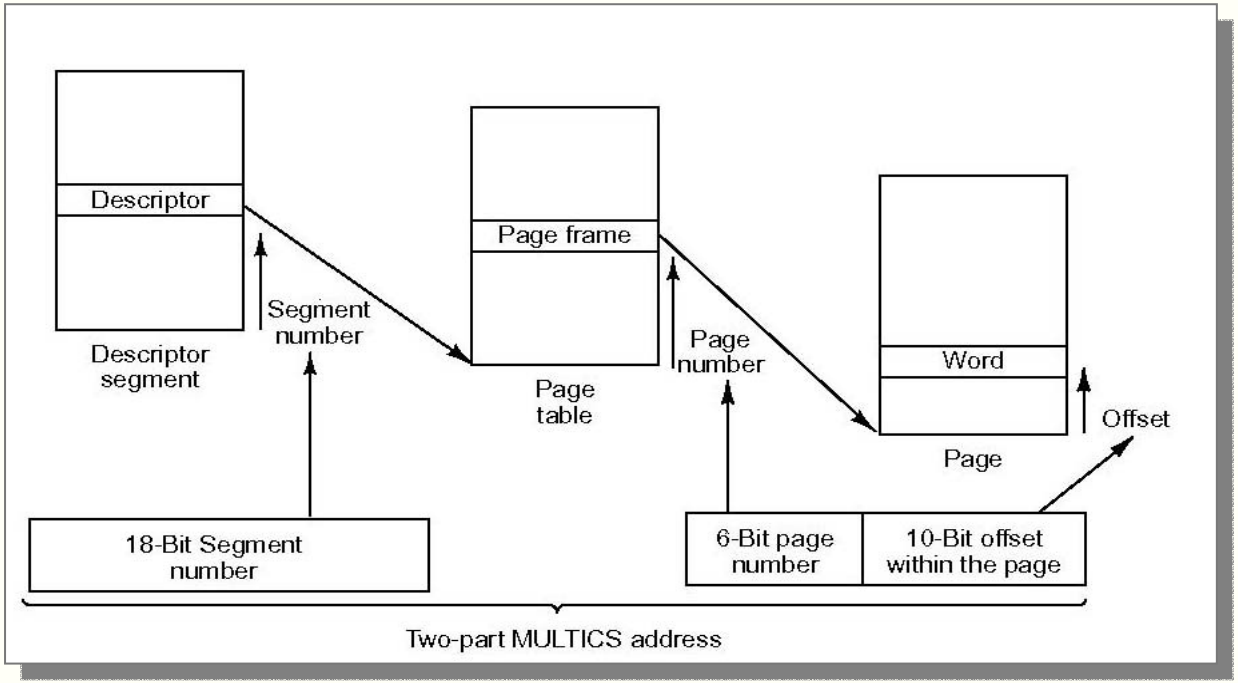
Szegmens helyettesítési eljárás

- első szabad hely (first fit).
  A memória kezdetétől vizsgálva az új szegmenst az első megfelelő üres helyre helyezi el. Kis szegmensek esetén jól bevált eljárás
- a következő szabad helyre (next fit).
  Az előzőleg elhelyezett szegmens követő első szabad helyet választja. A memória egyenletesebb kihasználtságára törekszik.
- legjobb illesztés (best fit).
  Megkeresi a legkisebb szabad helyet, amelyben a szegmens még elhelyezhető. Nagyobb összefüggő tárterületek hoz létre.
- legrosszabb illesztés (worst fit).
  A szegmenst úgy helyezi el, hogy lehető legnagyobb szabad terület maradjon mellette. Cél a minél nagyobb összefüggő szabad területek biztosítása.

Lapozásos szegmentálás

A szegmenseket fix méretű lapokra osztjuk.

Minden szegmenshez külön laptáblára van szükség, viszont nem kell egyszerre az egész szegmenst a memóriában tartani.



Virtuális címek leképezése

MMU feladata a virtuális címek átalakítása fizikai címekké a szegmens- és/ vagy lapleíró táblázatok adatainak felhasználásával.

A virtuális címzés táblázatok alapján megvalósított egy- vagy többlépcsős indirekt címzés.

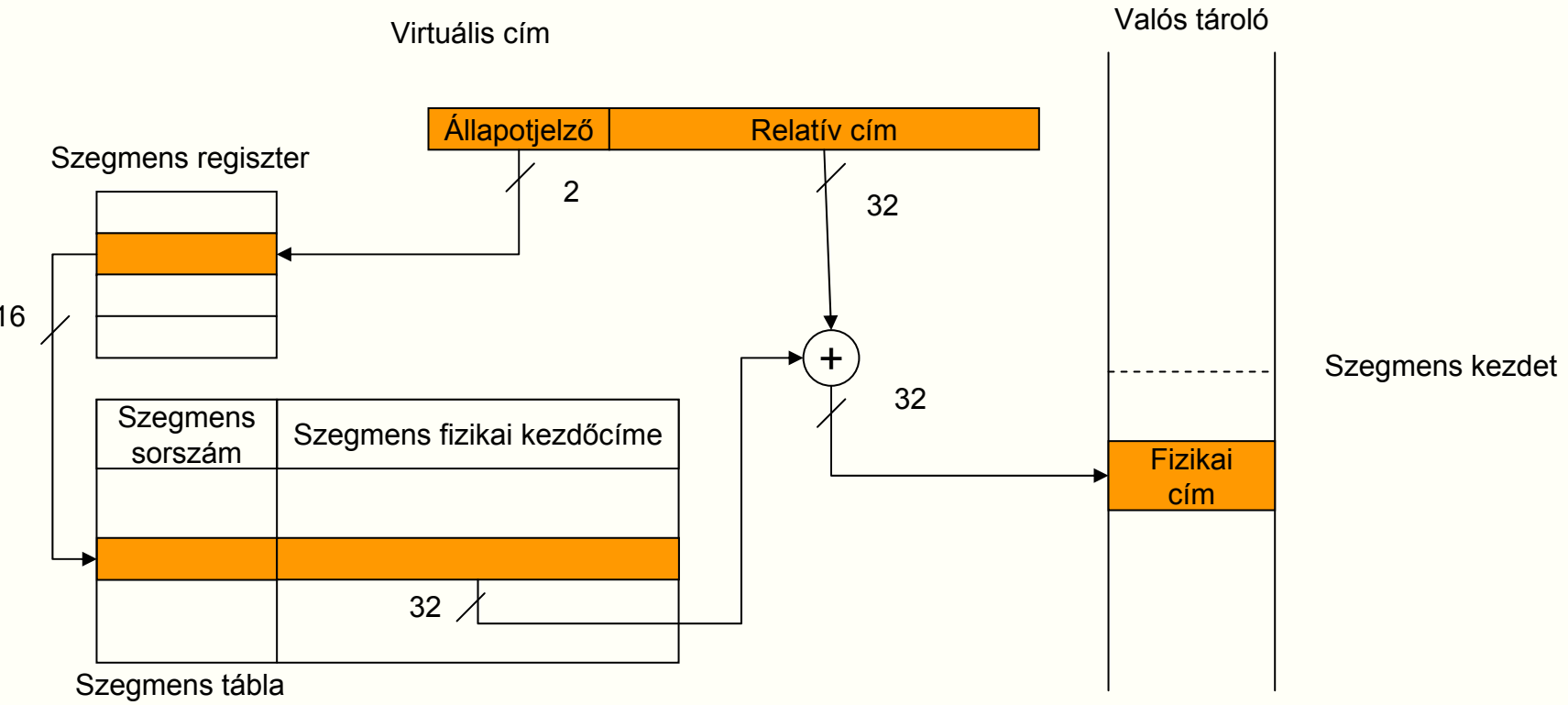
Lapcímfordító cache tár (TLB – Translation Lookaside Buffer) a leggyakrabban használt lapok adatait (descriptors) tartalmazza.

- lapváltáskor cache-hit. Virtuális lapcímet helyettesíti a fizikai lapcímmel, a védelmi jelzőket továbbítja a processzorhoz
- lapváltáskor cache-miss MMU előállítja a fizikai címet, a szükséges adatokat betölti a TLB-be, felszabadítva annak valamely sorát.

TLB 64-1024 sor méretű

- teljesen asszociatív
- 2-4 utas csoport asszociatív

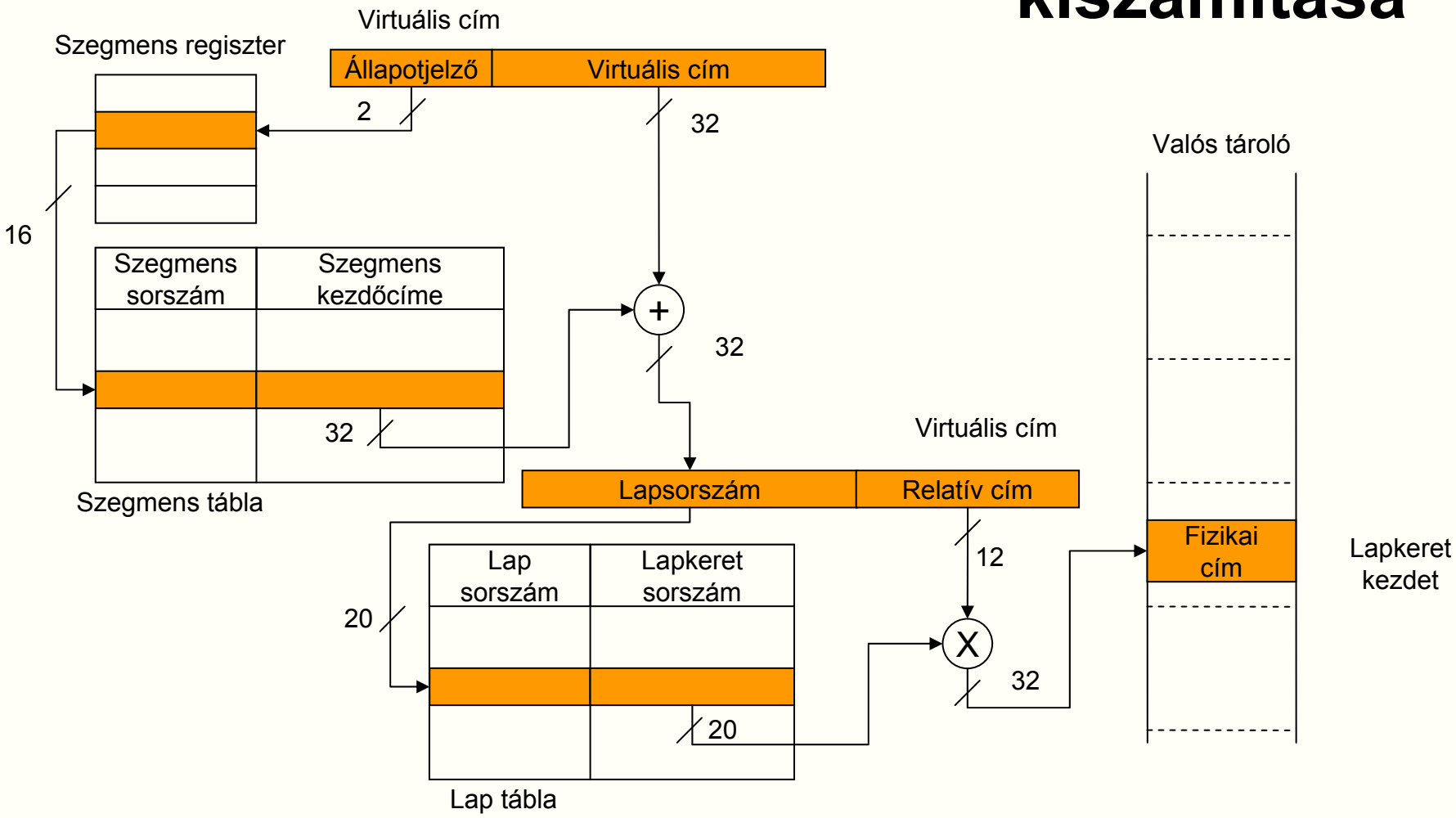
Egylépcsős szegmenscím kiszámítás szegmensregiszter felhasználásával



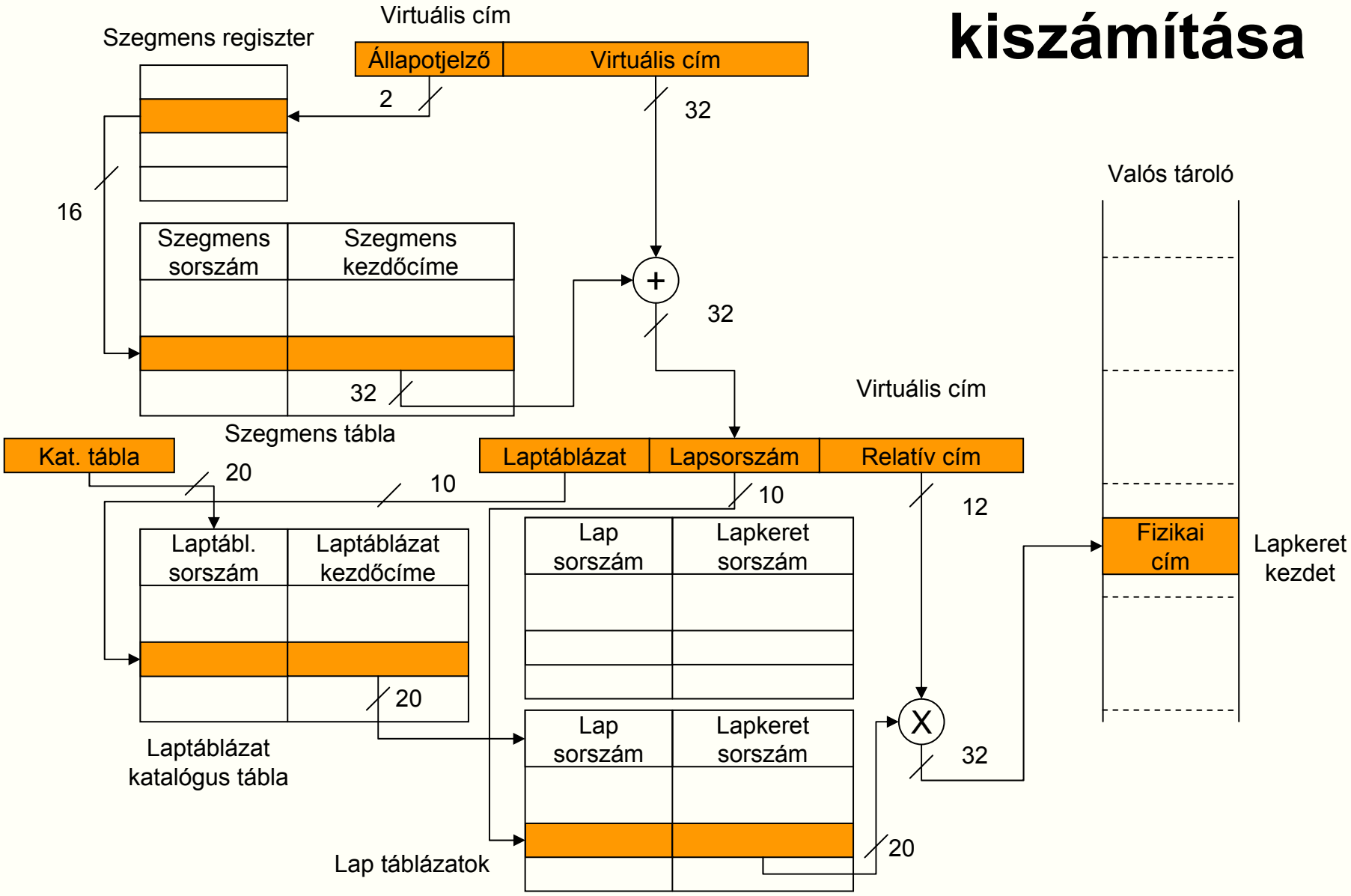
Az állapotjelző értéke a processzor működési módjától függ. A szegmens tábla nagy mérete miatt az operatív tárban helyezkedik el. A fizikai cím a szegmens kezdő címének és a relatív címnek az összege.

Ezt a módszert alkalmazza a Power PC 601

Kétlépcsős szegmentált lapcím kiszámítása



Háromlépcsős szegmentált lapcím kiszámítása



Tárvédelmi módszerek

MMU feladata

- címkiszámítás
- tárvédelmi feladatok
  - memóriaterület védelme, a címzések helyességének ellenőrzése
  - rendszerprogramok védelme a felhasználó beavatkozásaitól
  - a felhasználói taszkok egymástól elkülönítése
  - a tárolt adatokhoz történő hozzáférések ellenőrzése

Programok és felhasználói feladatok védelme

- hierarchikus rendszer (ring protect system)
- nem hierarchikus rendszer (capability based protect system) Minden taszkhoz egy táblát rendelnek, amely meghatározza a taszknak azon engedélyezett műveleteit, amely más taszkokat is érintenek. Nincs hardver megvalósítása, csak az operációs rendszer működteti.

Hierarchikus védelmi szerkezet

A legmagasabb védelmi szinttel az operációs rendszer rendelkezik. A legalacsonyabb védelmi szintje a felhasználói programoknak van.

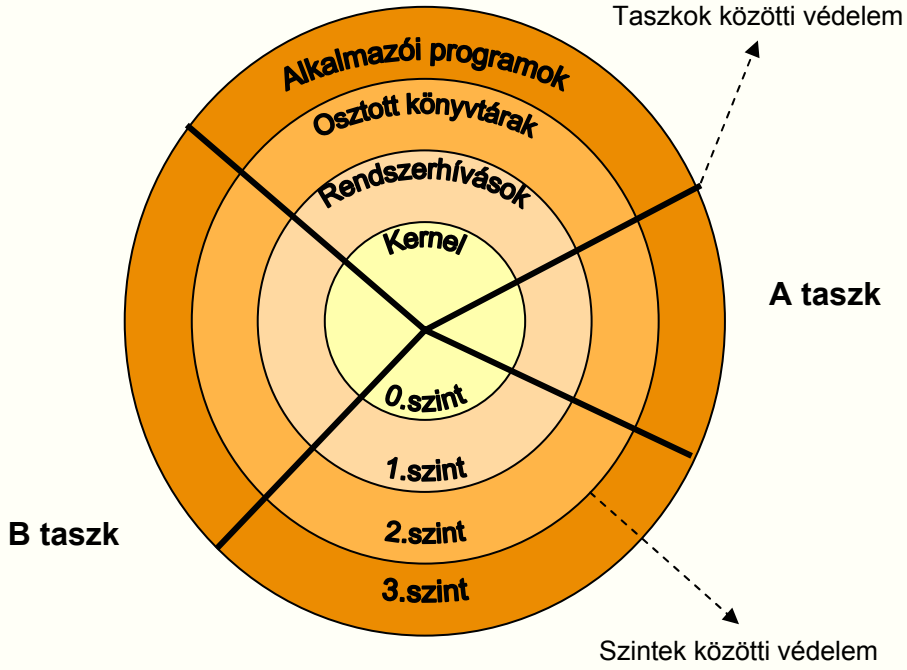
A programok más rutinokat csak a saját védelmi szintjükön vagy magasabb szinten hívhatnak. Az alacsonyabb szinteket és az azonos szinten futó többi taszkot csak védelmi kapukon keresztül érhetik el.

A feldolgozó programok adatokat csak a saját szintjükön, illetve alacsonyabb védelmi szinteken érhetnek el.

A szintek közötti védelmet az egyes szintek veremtárolója, a feladatok közötti védelmet az egyes feladatok saját leíró (descriptor) táblája biztosítja.

A hierarchikus rendszer előnye a hardver megvalósítás, és az ebből származó gyorsaság.

A hierarchikus védelmi rendszert alkalmazzák az Intel processzorok.



Az adatok védelme

MMU szabályozza a szegmensekhez, illetve lapokhoz történő hozzáférési jogokat

- olvasási jog (read access)
- írási jog (write access)
- végrehajtási jog (execute access)
  csak programot tartalmazó laphoz vagy szegmenshez rendelhető