

# A számítógépek felépítése 6.a: A memória

Markó Tamás  
PTE TTK, 2003

2003.08.25.

Markó Tamás, PTE TTK

1

# A rádiótelefonokat kérem KIKAPCSOLNI!

2003.08.25.

Markó Tamás, PTE TTK

2

## Az ideális memória

- Nagy logikai címtartomány biztosítása (pl.  $2^{52}$  bájt = 4 Pbyte)
- Gyors válaszidő (? 1 CPU-ciklus)
- Eltolható programkód (relocation)
- Az egyes memória-tartományok védelme a más tartományokban futó programoktól (protection)

2003.08.25.

Markó Tamás, PTE TTK

4

## Problémák az operatív tárral

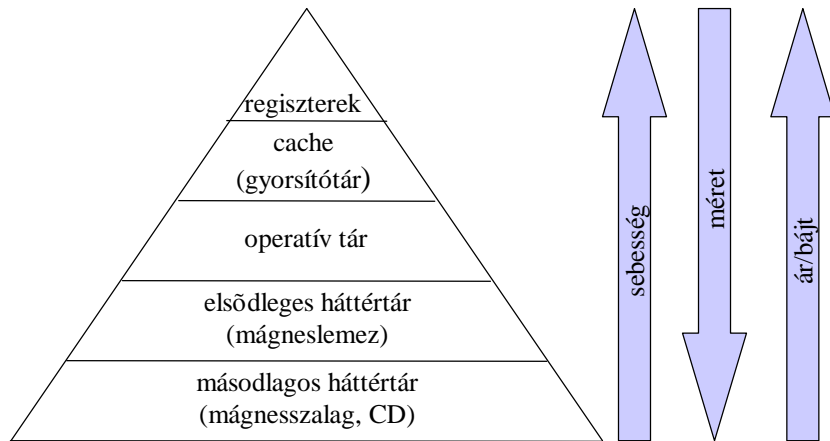
- A processzor sebességéhez képest - a mai technikai megoldásokkal - túl lassú  
? **cache**
- Sok feladathoz túl kicsi  
? **virtuális memória**
- Hosszú távú adattárolásra nem alkalmas  
? **háttértár**

2003.08.25.

Markó Tamás, PTE TTK

5

# Memória-hierarchia



2003.08.25.

Markó Tamás, PTE TTK

6

## Az operatív tár

2003.08.25.

Markó Tamás, PTE TTK

7

## Az operatív tár

- Main memory (MM), központi memória
- Lineáris szerkezet (sorszámozott, azonos méretű rekeszek)
- **Az egyetlen tár, amiből a processzor a programokat végrehajtja**
- Hosszú időn keresztül (kb. 1955-1975) ferritgyűrűkből készült
- Ma jellemző a félvezetőkkel felépített operatív tár

2003.08.25.

Markó Tamás, PTE TTK

8

## Bájtsorrend a memóriában

- **Nagy endián:** egy változó nagyobb helyiértékű bájtjai a kisebb címeken vannak (pl. SPARC, IBM nagygépek)
- **Kis endián:** egy változó nagyobb helyiértékű bájtjai a nagyobb címeken vannak (pl. Intel)

cím			
0	0	3	
0	1	0	
0	2	0	
3	3	0	

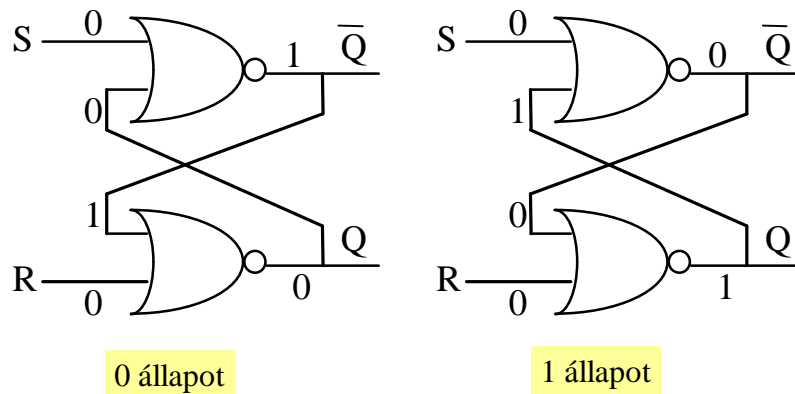
A 3 tárolása 4 bájtos egészként a 0 címen

2003.08.25.

Markó Tamás, PTE TTK

9

## 1 bites félvezetős memória (NOR-tároló, SR-tároló, set-reset latch)

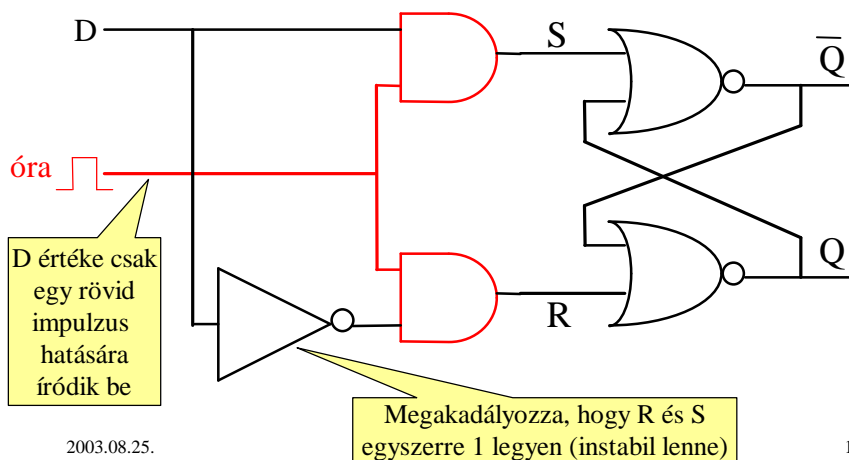


2003.08.25.

Markó Tamás, PTE TTK

10

## 1 bites félvezetős memória (időzített D-tároló)



2003.08.25.

Megakadályozza, hogy R és S egyszerre 1 legyen (instabil lenne)

11

## Statikus RAM (SRAM)

- RAM: írható és olvasható memória (random access memory - rossz név).
- D-tároló szerű elemekből épülnek fel: 1 bithez 11 (ügyesebb szervezéssel csak 6) tranzisztor
- Amíg áram alatt vannak, tartják a tartalmukat.
- Elérési idő: néhány nsec (cache-nek jók).

2003.08.25.

Markó Tamás, PTE TTK

12

## Dinamikus RAM (DRAM)

- Minden bit tárolásához egy tranzisztor és egy kondenzátor
- Egyszerűbb struktúra és nagyobb adatsűrűség
- Hátrány: néhány msec-onként fel kell frissíteni (bonyolultabb külső vezérlés)
- Lassabbak, de olcsóbbak a statikusoknál

2003.08.25.

Markó Tamás, PTE TTK

13

## Read-Only Memory, ROM

- Csak olvasható:
  - nem módosítható, nem törölhető
  - a tartalma áram nélkül is megmarad
- Tartalma a gyártás során állítódik be
- Nagy mennyiségben sokkal olcsóbb a RAM-nál
- Rugalmatlan: ha módosítás kell, ki kell cserélni

2003.08.25.

Markó Tamás, PTE TTK

14

## PROM

- PROM: Programmable ROM
- Tartalma egyszer elektronikusan (a gyártás után) megadható
- Gyakori megoldás: apró olvadó biztosítékokból álló tömb  
(1 biztosíték = 1 bit)
- A programozás az egyes biteknek megfelelő biztosítékok nagy feszültséggel való kiégetését jelenti

2003.08.25.

Markó Tamás, PTE TTK

15

## EPROM

- EPROM: törölhető (erasable) PROM
- Speciális berendezéssel elektronikusan írható
- Erős ultraibolya fénnnyel minden bit újra 1-re állítható
- Jól felismerhető a kis üveglakról (a véletlen törlés ellen ezt le szokták ragasztani)

2003.08.25.

Markó Tamás, PTE TTK

16

## EEPROM

- EEPROM: Electronically Erasable PROM
- Elektronikusan törölhető ultraibolya fény helyett
- A végleges helyén programozható
- Lassabb, kisebb és drágább, mint a RAM
- Flash memória:
  - speciális EEPROM
  - újfajta háttértár lesz?

2003.08.25.

Markó Tamás, PTE TTK

17



# A gyorsítótár (cache)

2003.08.25.

Markó Tamás, PTE TTK

18

# A gyorsítótár (cache)

- Cache: „elrejtett készlet“, „rejtekhely”
  - a processzor nem veszi észre a cache létezését
- Az alapötlet:
  - nagyon gyors tároló (ezért drága!)
  - elérési ideje összemérhető a regiszterek elérési idejével
  - olyan adatokat tartalmaz, amikre „a legnagyobb valószínűséggel” lesz szükség a közeljövőben
- A használhatóság előfeltétele:
  - lokáltság

2003.08.25.

Markó Tamás, PTE TTK

19

# Lokalitás 1.

## Időbeli lokalitás:

Bármely  $t$  időpillanatra igaz, hogy a közeli jövőben a program nagy valószínűséggel csak olyan objektumokra (utasításokra és adatokra) fog hivatkozni, amikre a közelmúltban hivatkozott.

2003.08.25.

Markó Tamás, PTE TTK

20

# Lokalitás 2.

## Térbeli lokalitás:

Bármely  $t$  időpillanatra igaz, hogy a közeli jövőben a program nagy valószínűséggel csak olyan objektumokra (utasításokra és adatokra) fog hivatkozni, amelyeknek címe a közelmúltban hivatkozott objektumok (utasítások és adatok) címének közelében van.

2003.08.25.

Markó Tamás, PTE TTK

21

## Lokalitás 3.

A lokalitás elve általában teljesül:

- Időbeli lokalitás:
  - ciklusok
  - verem
- Térbeli lokalitás:
  - szekvenciális kód
  - tömbök

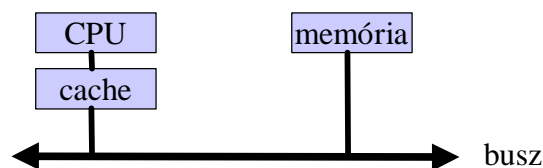
2003.08.25.

Markó Tamás, PTE TTK

22

## A cache technikai megvalósítása

- Mindenképpen a processzor és az operatív tár „között”



- Többszintű cache is lehetséges:
  - a processzorlapkán
  - külön lapkán, de a processzorral közös tokban
  - külön tokban

2003.08.25.

Markó Tamás, PTE TTK

23

## A találati arány

- Egy cache-rendszer találati aránya ( $p_h$ , hit ratio) annak a valószínűsége, hogy egy megcímzett objektum éppen a cache-ben van.
- A hibaarány ( $p_m$ , miss ratio) ennek a komplementere:  $p_m = 1 - p_h$

2003.08.25.

Markó Tamás, PTE TTK

24

## A cache méretezése

A megoldandó problémák:

- Mekkora legyen a cache?
- Mekkora adagokban történjen az adatok mozgatása a cache és az operatív tár között?

2003.08.25.

Markó Tamás, PTE TTK

25

## A cache adminisztrálása

A megoldandó problémák:

- „if and where“-kérdés:
  - A megcímzett adat a cache-ben van-e, és ha igen, hol?
- „cache-miss“-kérdés:
  - Ha a megcímzett dátum nincs a cache-ben, hova kell betölteni (azaz melyik cache-ben lévő adatokat kell felülírni?)

2003.08.25.

Markó Tamás, PTE TTK

26

## A gyorsítótár sorai (cache line)

- Sor: a memóriából a cache-be egyszerre átmásolt tartomány
  - a sor folytonos tárterület
  - a cache szervezése meghatározza a sor méretét (tipikusan 4-64 bájt)
  - a memória sorokra osztása rögzített (a 0 címtől kezdődő „feldarabolás”)
- A soronkénti másolás gyorsabb, mint a bájtonkénti

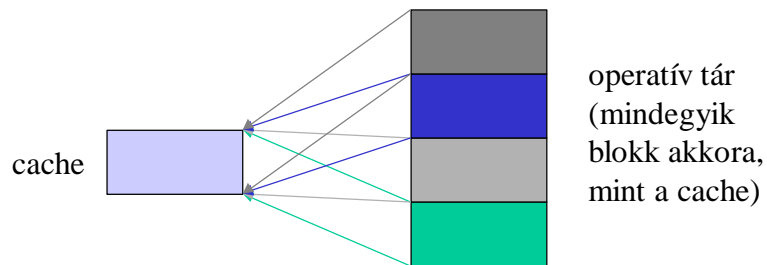
2003.08.25.

Markó Tamás, PTE TTK

27

## Közvetlen leképezésû (direct mapped) cache 1.

- A memória minden sora csak egy adott helyre kerülhet a cache-ben
- ? bizonyos memória-sorok nem lehetnek egyidejûleg a cache-ben



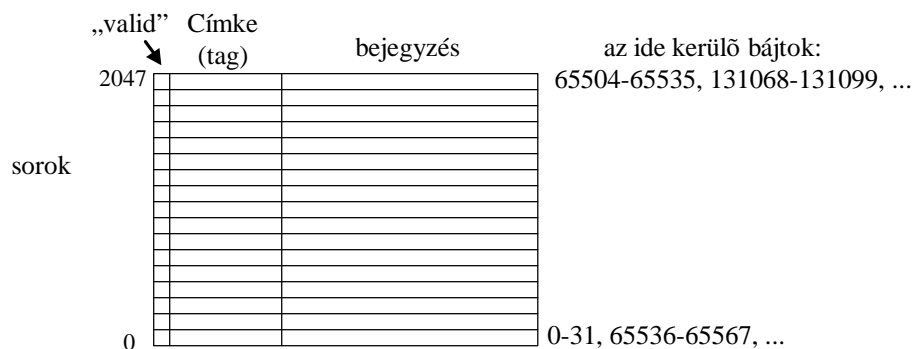
2003.08.25.

Markó Tamás, PTE TTK

29

## Közvetlen leképezésû (direct mapped) cache 2.

Egy 64kB-os cache felépítése (32 bájtos sorok):



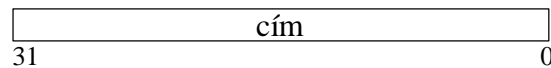
2003.08.25.

Markó Tamás, PTE TTK

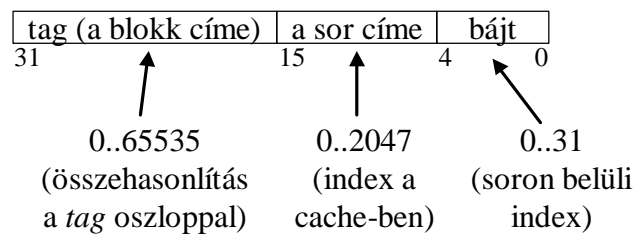
30

## A közvetlen leképezésû cache címzése

- A processzor homogén tárterületet címez meg. A processzor által használt cím alakja:



- A cache ezt másként értelmezi: blokkonkénti címzést használ, tehát



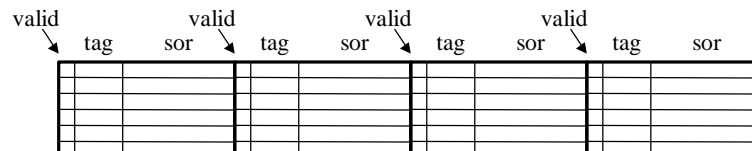
2003.08.25.

Markó Tamás, PTE TTK

31

## Halmazkezelésû (set associative) cache

- Az egymást a cache-ből kizáró memória-sorok problémájának kezelésére
- A cache sorait megtöbbszörözik, így több, eredetileg egymást kizáró sor is a cache-ben lehet
- A sorok n-szerezése: n utas halmazkezelésû gyorsítótár (tapasztalat: n=2 vagy n=4 már jó)
- Az adminisztráció bonyolultabb



2003.08.25.

Markó Tamás, PTE TTK

32

## Utántöltési stratégiák

- A kérdés: mikor kell a cache tartalmát utántölteni?
- Két stratégia:
  - demand fetch
  - prefetch
- A *demand fetch*:
  - Ha a megcímezett blokk nincs a cache-ben, betöltődik.
  - Direct mapping esetén az egyértelműen definiált blokkot írja felül.
  - Egyébként lehet
    - random
    - lru (least recently used)
    - fifo (first in - first out).

2003.08.25.

Markó Tamás, PTE TTK

39

## Prefetching

- Mit töltünk be előre?
  - Hardver-okok miatt egyszerű stratégia kell.
  - Közelben lévő (a következő nagyobb című) blokkot választunk (lokalitás!!)  
(One block look-ahead OBL)
- Mikor töltünk be előre adatokat?
  - always (minden cache-hez való hozzáférés után)
  - on miss (ha nincs a cache-ben)
  - tagged prefetch: on miss + on reference to prefetched line (nincs a cache-ben vagy már hivatkozunk az előre betöltött adatra)

2003.08.25.

Markó Tamás, PTE TTK

40



## A prefetch-stratégiák elemzése

- Always
  - a legjobb találati arányt adja
  - a legnagyobb forgalmat generálja a buszon (Több processzor! DMA!)
- On miss
  - minimális busz-forgalom
  - alacsony találati arány
- Tagged prefetch
  - majdnem olyan jó találati arány, mint az „always“ esetén
  - de kisebb busz-forgalom
- A *prefetching* nem helyettesítheti a *demand fetching*-et, néha felesleges utántöltésekhez vezethet

## A Pentium cache-kezelése

## Cache-kezelés

- A Pentium különböző célokra használ cache-t:
  - belső cache
    - utasításoknak
    - adatoknak
  - címfordítás (TLB)
  - prefetching (alkalmanként 1 cache-sor)
  - write buffers
- Az adat- és utasítás-cache-ek jellemzői:
  - méretük 4 KB
  - 32 bájtos sorok
  - halmazkezelésűek, az asszociativitás 2
  - LRU frissítési algoritmus

2003.08.25.

Markó Tamás, PTE TTK

44

## Aktualizálási stratégia

- Csak az adat-cache esetében érdekes
- Azt szabja meg, hogy mikor hat egy írási művelet az operatív tárra
  - **Write-through:**  
minden írás az operatív tárra is hat
  - **Copy-back:**  
az írás csak akkor hat az operatív tárra, ha
    - a cache adott sora törlődik új sor beolvasása miatt
    - MESI-protokollt használunk (többprocesszoros rendsz.)
- A Pentium mindkét stratégiát támogatja (keverten is)

Markó Tamás, PTE TTK

45

## Table Lookaside Buffer (TLB)

- Külön TLB az utasítások és az adatok részére
- A felhasználói programok nem férnek hozzá
- Az operációs rendszer használja
- A programok közötti átkapcsoláskor az operációs rendszer érvényteleníti a TLB-k régi tartalmát

2003.08.25.

Markó Tamás, PTE TTK

46

## A PowerPC cache-kezelése

2003.08.25.

Markó Tamás, PTE TTK

47

## A Memory Unit

- Processzoron belüli puffer a a cache és a külső csatlakozófelület között
- Az operatív tárhoz való hozzáférésnél kell
  - cache miss
  - MESI-protokoll szerinti írási és olvasási műveletek
  - a táblázat végigolvasása, ha a TLB-ben nincs meg a szükséges adat
- Olvasási sor (read queue): puffer 2 cím részére
- Írási sor (write queue):
  - 3 hely, mindegyik a cím és max. 8 szó adat részére (tehát egy teljes cache-sor)
  - 1 hely a legnagyobb prioritással a MESI szerinti íráshoz (SNOOP)

48

## A külső csatlakozófelület

- A legfontosabb üzemmód: burst read/write (a processzor-chipen lévő cache miatt)
- Kifelé külön adat- és címbusz
- Másodlagos (a chipen kívüli) cache támogatása
- Az írási és olvasási műveletek sorrendje megváltoztatható, ha nincs adatfüggőség (scoreboard vezérli)
  - ? a busz forgalma optimalizálható

2003.08.25.

Markó Tamás, PTE TTK

49

## A Cache Unit

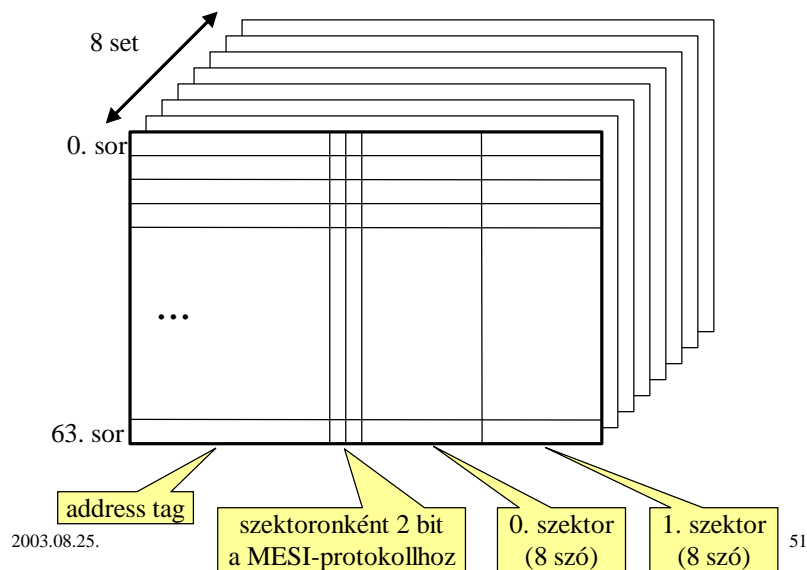
- 32 KB, 8-as asszociativitás
- 1 közös cache az utasítások és az adatok részére
- Cache-sor: 64 bájt, 2 db 8 szavas szektorra osztva
- A két szektor egymástól függetlenül érhető el
- A sorok lecserélési módszere: LRU
- Írási módszer: copy back vagy write through
- Csatlakozófelület: 8 szó szélességű

2003.08.25.

Markó Tamás, PTE TTK

50

## A cache szervezése a PowerPC 601-nél



## Cache-műveletek

- **Cache Reload:**

Sikertelen olvasási kísérlet esetén egy szektor betöltése.

Az utasítás-prefetch nem vált ki Cache Reload-ot.

- **Cache Cast-out:**

Az LRU miatt kiszoruló szegmens memóriába írása. A sor másik szegmensét is kiírjuk (alacsonyabb prioritással).

- **Cache Sektor Push:**

Szektor írása a MESI-protokoll miatt.

52