

A számítógépek felépítése 6.b: A virtuális memória

Markó Tamás
PTE TTK, 2003

2003.08.25.

Markó Tamás, PTE TTK

1

A rádiótelefonokat kérem KIKAPCSOLNI!

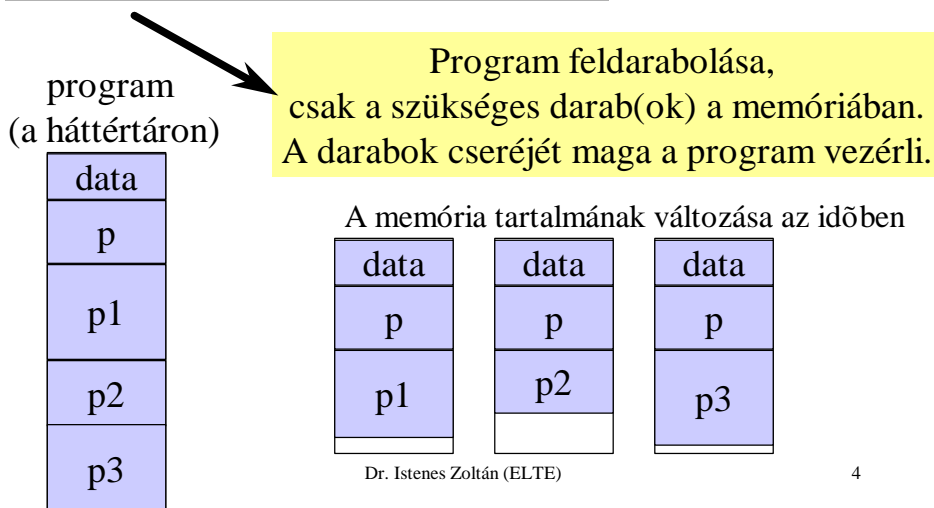
2003.08.25.

Markó Tamás, PTE TTK

2

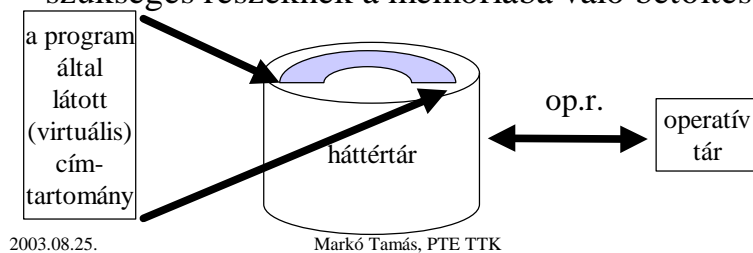
Overlay technika (átlapolásos technika)

Probléma : nagy program, kis memória

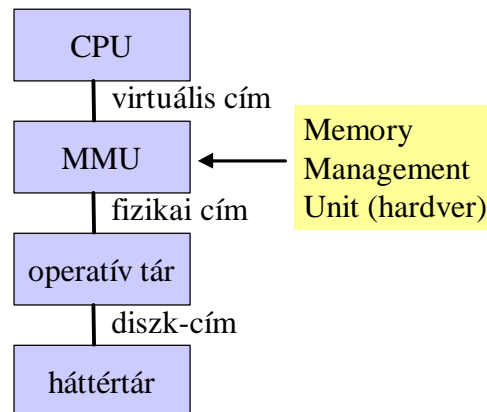


A virtuális memória ötlete (1961)

- A programok úgy készülhetnek, mintha a számítógépnek igen nagy memóriája lenne (virtuális címtartomány)
- Ennek a képzeletbeli memóriának a képét az operációs rendszer a háttértáron tartja
- Az operációs rendszer gondoskodik a mindenkor szükséges részeknek a memóriába való betöltéséről



A virtuális memória hardveres támogatása



2003.08.25.

Markó Tamás, PTE TTK

6

Fogalmak 1.

- Virtuális cím (logikai cím):
a programok által a CPU-ban használt cím
- Virtuális (logikai) címtartomány:
a CPU által használható teljes címtartomány
- Valós cím (fizikai cím):
az operatív tárba mutató cím
- Valós (fizikai) címtartomány:
az operatív tárba mutató címek halmaza

2003.08.25.

Markó Tamás, PTE TTK

7

Fogalmak 2.

- **Címleképzés:**
a virtuális címek átszámítása valós címekre
- **Memory Management Unit (MMU):**
a címleképzést megvalósító hardver
- **Virtuális memóriarendszer:**
a címleképzés láthatatlan (transzparens) a programok számára

2003.08.25.

Markó Tamás, PTE TTK

8

Fogalmak 3.

- **Statikus hozzárendelés (static relocation):**
a virtuális címek fizikai címekhez való hozzárendelése **a program betöltése során** megtörténik
- **Dinamikus hozzárendelés (dynamic relocation):**
a virtuális címek fizikai címekhez való hozzárendelése **a program futása során** történik meg
- Csak a dinamikus hozzárendeléssel foglalkozunk (a statikus hozzárendelésnek semmi köze a számítógép-architektúrához)

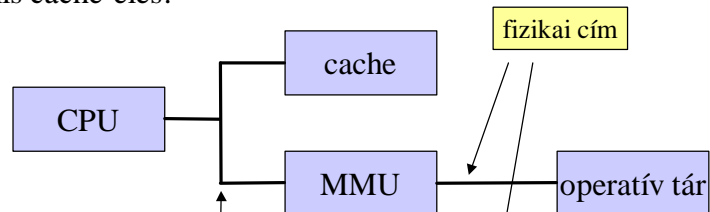
2003.08.25.

Markó Tamás, PTE TTK

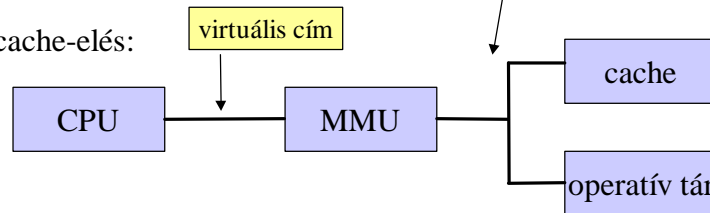
9

A cache és az MMU viszonya

Virtuális cache-elés:



Valós cache-elés:



2003.08.25.

Markó Tamás, PTE TTK

10

Dinamikus hozzárendelési eljárások

A használatos módszerek:

- Áthelyezés (relokálás, relocation)
- Lapozás (paging)
- Szegmentálás (segmentation)
- Paged Segmentation

2003.08.25.

Markó Tamás, PTE TTK

11

Relokálás

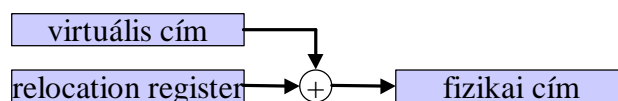
2003.08.25.

Markó Tamás, PTE TTK

12

Relokálás (relocation)

- Előfeltételek:
 - minden program kezdőcíme 0
 - minden program címtartománya ? valós címtartomány
- A módszer:
 - az op.r. dinamikusan megállapítja, hogy hova kell betölteni a programot
 - kezdőcím ? **relocation register** RR
 - valós cím := RR + virtuális cím
 - átkapcsolás másik programra: az RR átírásával



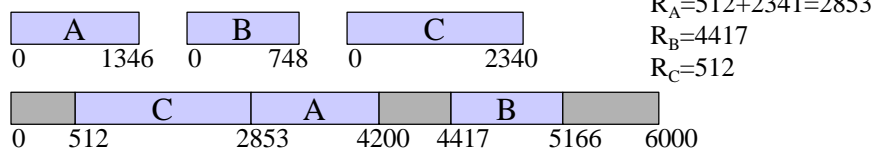
2003.08.25.

Markó Tamás, PTE TTK

13

Példa a relokálásra

- 3 program: A, B, C
- Méretük: A: 1347 bájt, B: 749 bájt, C: 2341 bájt
- Kezdőcímük a memóriában (ez kerül az RR-be):
A: 2853, B: 4417, C: 512



- Átkapcsolás B-ről C-re: 512 ? RR
- Más lehetőség: több RR, átváltáshoz csak a megfelelő RR-t kell kiválasztani

2003.08.25.

Markó Tamás, PTE TTK

14

A relokálás hiányosságai

- A virtuális címtartomány legjobb esetben is csak akkora lehet, mint a fizikai
 - Az egyes programokhoz összefüggő fizikai tárterületet kell rendelni
 - Nem védett az egyes programok tárterülete (ez megoldható **protection register** alkalmazásával)
- ? Manapság nemigen használatos

2003.08.25.

Markó Tamás, PTE TTK

15

Lapozás

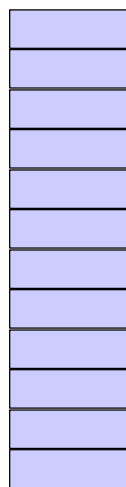
2003.08.25.

Markó Tamás, PTE TTK

16

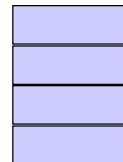
Lapok képzése

lapok = azonos, rögzített méretű adatblokkok



logikai
címtartomány

logikai címtartomány és a
fizikai címtartomány
felosztása lapokra



fizikai
címtartomány

Dr. Istenes Zoltán, ELTE

17

Lapozás (paging)

- A virtuális címtartományt azonos nagyságú blokkokra (**logikai lapokra, pages**) osztják
- A fizikai címtartományt ugyanekkora **fizikai lapokra (page frames)** osztják
- A virtuális címek fizikai címekké alakítása egy hozzárendelési táblázattal (**page table, PT**)
- Minden programnak saját *page table*, aminek kezdőcímét a **page table register (PTR)** tartalmazza
- A programok közti átkapcsolás a PTR átírásával történik

2003.08.25.

Markó Tamás, PTE TTK

18

Címzés lapozáskor 1.

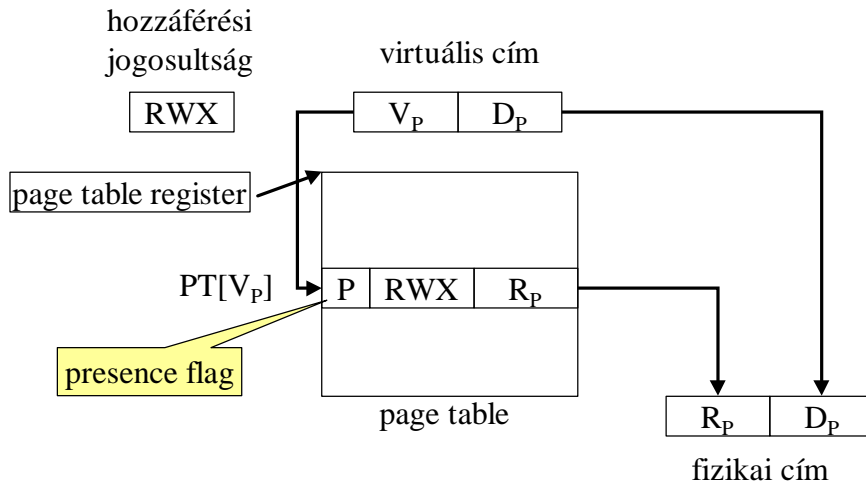
- A virtuális cím részei:
 - a logikai lap címe (V_p)
 - a lapon belüli cím (D_p)
- A fizikai cím részei:
 - a fizikai lap címe (R_p)
 - a lapon belüli cím (ugyanaz, mint a virtuális címnél, D_p)
- A *page table* egy bejegyzésének (**PTE, page table entry**) tartalma:
 - a fizikai lap címe (R_p)
 - hozzáférési jogosultság (**read, write, execute**)
 - **presence flag** (= 1, ha a lap bent van a fizikai memóriában)

2003.08.25.

Markó Tamás, PTE TTK

19

Címzés lapozáskor 2.



2003.08.25.

Markó Tamás, PTE TTK

20

Utántöltési stratégiák lapozáskor

- A legegyszerűbb megoldás:
 - Egy program összes lapját betöltjük (nem feltétlenül folytonos tárterületre)
- Probléma:
 - **Superfluity**: olyan lapokat is betöltünk, amiket később egyáltalán nem használunk (pazarlás).
- Megoldás:
 - **Demand paging**: csak azokat a lapokat töltjük be, amikre hivatkozás történik.
 - További előny: így a virtuális címtartomány nagyobb is lehet, mint a fizikai.

2003.08.25.

Markó Tamás, PTE TTK

21

Page fault

- *Demand paging* esetén nyilvántartjuk, hogy egy program melyik lapjai vannak az operatív tárban és melyek a háttértárolón.
- Ez a különbség a program számára nem látható (transzparens).
- Ha a hivatkozott lap nincs az operatív tárban: **page fault**.
- Ilyenkor ezt a lapot be kell tölteni.
Ez - mint minden I/O-művelet - a program futásának felfüggesztéséhez vezet.

2003.08.25.

Markó Tamás, PTE TTK

22

Helyettesítési eljárások

Melyik lapot kell cserélni (eltávolítani) ?

- optimal : „amire legkevesbé lesz szükség”...
- legrégebben bentlévő (FIFO)
- legrégebbi nem használt
- legrégebben használt (LRU - Least Recently Used)
- ...

Dr. Istenes Zoltán (ELTE)

23

Kalkuláció lapozáshoz 1.

- A memória bájtonként címezhető
- 32 bites virtuális cím (4 GB címtartomány)
- Lapméret: 4 kB ($= 2^{12}$ bájt)
 - ? a logikai lapok címe $32 - 12 = 20$ bites
 - ? 2^{20} számú logikai lap (és ennyi sor a PT-ben)
- 128 MB ($= 2^{27}$ bájt) fizikai memória
 - ? a fizikai lapok címe $27 - 12 = 15$ bites
 - ? $2^{27-12} = 2^{15}$ számú fizikai lap

2003.08.25.

Markó Tamás, PTE TTK

24

Kalkuláció lapozáshoz 2.

- A *page table* egy sora 3 bájtos:
 - 15 bit (? 2 bájt) a fizikai lap címének (R_p)
 - a harmadik bájtban a *presence flag* és a jogosultságok
- A *page table* mérete: 2^{20} sor \times 3 bájt = 3 MB
- A lapozás hátrányai:
 - Járulékos adminisztráció: a PT is a virtuális címtartományban van, lapozható (table superfluity).
 - A fix lapméret hátránya: vannak nem használt címek (belső fragmentálódás).

2003.08.25.

Markó Tamás, PTE TTK

25

Szegmentálás

2003.08.25.

Markó Tamás, PTE TTK

26

A szegmentálás igénye

- A fix blokkméret természet- (logika-) ellenes. Pl. a fordítóprogramok munkájuk során több, előre nem látható méretű, nagy táblázatot kezelnek:
 - szimbólumtábla (a változók neve és tulajdonsága)
 - forráskód (kilitázáshoz, debughoz)
 - a programban használt konstansok
 - a forráskód szintaktikus elemzéséhez használt elemzési fa
 - hívási verem a szubrutinokhoz
- Szegmens:
 - független, 0-val kezdődő címtartomány
 - változó méretű blokk, igény szerint nőhet

2003.08.25.

Markó Tamás, PTE TTK

27

Előnyök szegmentálásnál

- Feltesszük, hogy a végrehajtható program külön szegmensben tartja az utasításokat (kódszegmens) és az adatokat (adatszegmens)
- Takarékoság a memóriával:
 - Ha több felhasználó ugyanazt a programot futtatja, a kódszegmenset csak egy példányban kell betölteni (de mindenkinek külön adatszegmens)
- Programhibák elleni védelem:
 - a kódszegmens írásvédett, de olvasható és végrehajtható
 - az adatszegmens nem végrehajtható, de írható és olvasható

2003.08.25.

Markó Tamás, PTE TTK

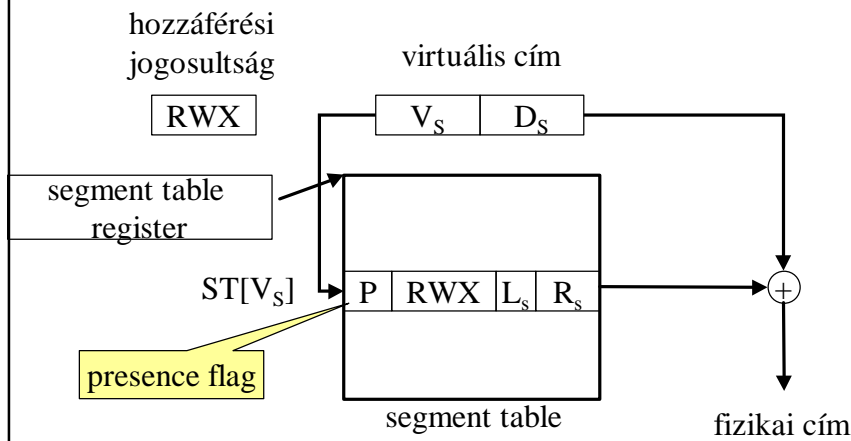
28

A szegmentálás adminisztrálása

- Hasonló a lapozáséhoz:
 - A virtuális cím a V_S szegmenscímből és a D_S szegmens belüli eltolásból (displacement) áll.
 - Van (programonként) egy **segment table (ST)**.
 - Erre mutat a **segment table register (STR)**.
 - A szegmenstábla sorait (**segment table entries, STE**) **segment descriptor**-nak is hívják. Egy-egy szegmens adatait tartalmazzák.
 - Az **STE** hasonló felépítésű, mint a **PTE**, de van benne még egy L_S mező, ami a szegmens hosszát tartalmazza.

29

Címképzés szegmentálásnál

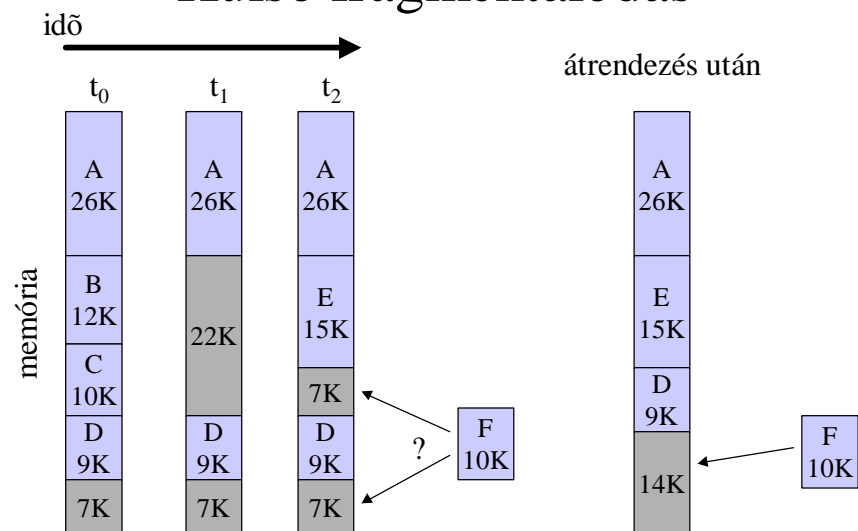


2003.08.25.

Markó Tamás, PTE TTK

30

Külső fragmentálódás



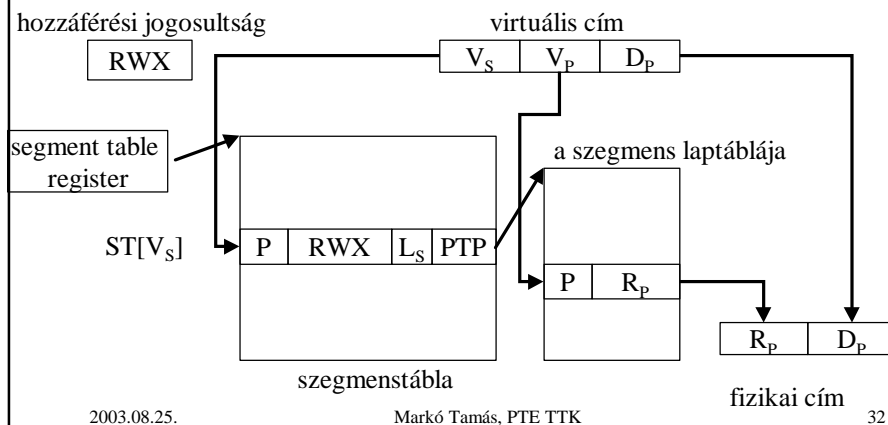
2003.08.25.

Markó Tamás, PTE TTK

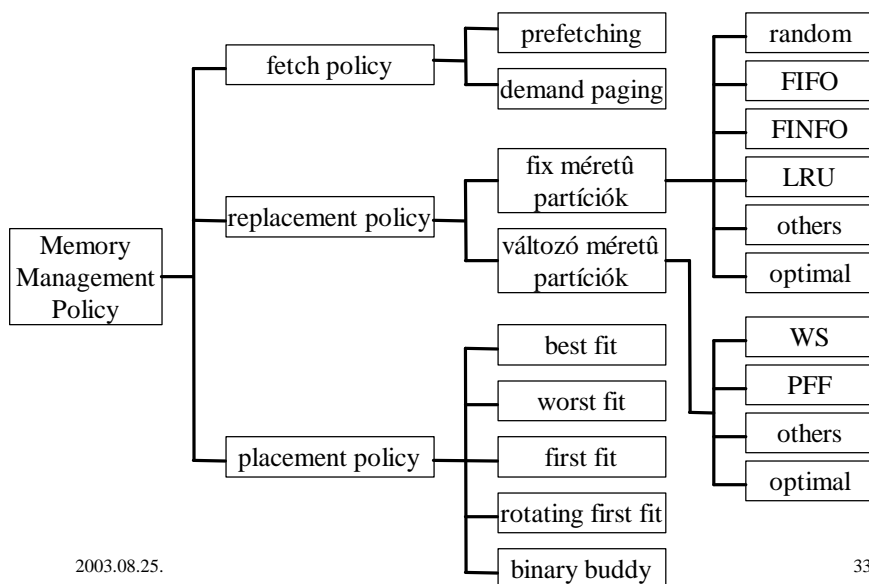
31

Lapozott szegmentálás

- A lapozás és a szegmentálás kombinációja:
1. szint: szegmentálás, 2. szint: lapozás
- Jelenleg széles körben elterjedten használják.



Memóriakezelési eljárások



Egyéb kérdések

2003.08.25.

Markó Tamás, PTE TTK

34

Fetch / placement

- Fetch:
 - két alap-módszer:
 - prefetching: nehéz, ritka
 - demand fetching: a szokásos eljárás
- Placement:
 - A probléma csak szegmentálásnál jelentkezik.
 - Különböző nagyságú szabad helyek vannak az operatív tárban. Egy szegmenset be kell tölteni. Hova?

2003.08.25.

Markó Tamás, PTE TTK

35

Hardver vagy operációs rendszer?

- A virtuális tárkezelési algoritmusokat alapvetően nem hardverben valósítják meg, hanem az operációs rendszerben
- Részletes tárgyalásuk az operációs rendszereknél (másik tantárgy)
- Itt csak a virtuális tárkezelés hardvertámogatása a téma

2003.08.25.

Markó Tamás, PTE TTK

36

A címkiszámítás hardver támogatása

Translation Lookaside Buffers (TLB)

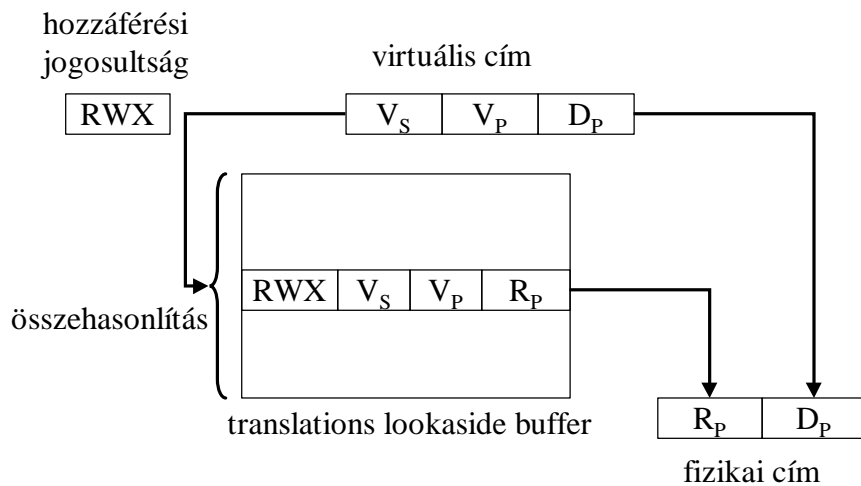
- Alapja: az időbeli lokalitás
- Az ötlet: az n utoljára használt címszámítás „cachelése“.

2003.08.25.

Markó Tamás, PTE TTK

37

Példa a TLB-re



2003.08.25.

Markó Tamás, PTE TTK

38

A TLB használata

- Összehasonlítjuk tehát $RWX || V_s || V_p$ -t és $TLB.(RWX || TLB.V_s || TLB.V_p)$ -t (a TLB összes sorával egyszerre!)
- Egyezés (legfeljebb egy lehet) az i -edik sornál:
 ? $TLB[i].R_p$ -t kiolvassuk
 ? $TLB[i].R_p // D_p$ lesz a kívánt cím
- Figyelem! A TLB-t nem a címmel „címezzük meg”, hanem a tartalommal
- Az ilyen memóriát **asszociatív tárnak** vagy **content adressable memory-nak (CAM)** hívják.

2003.08.25.

Markó Tamás, PTE TTK

39

A TLB problémái

- A TLB mérete korlátozott
 - ? gyakran nincs benne a keresett cím
- A TLB drága
 - ? összesen 1 TLB az összes program számára
 - ? a programok közti átkapcsolásnál a TLB összes sora érvénytelenné válik (minden programnak saját címtartománya van!)
 - page-fault esetén mindig átkapcsolás történik
 - ? a TLB gyakran válik érvénytelenné
- Ennek ellenére a TLB ma „szabvány” (VAX, MC 88200, Pentium, PowerPC, SPARC ...)

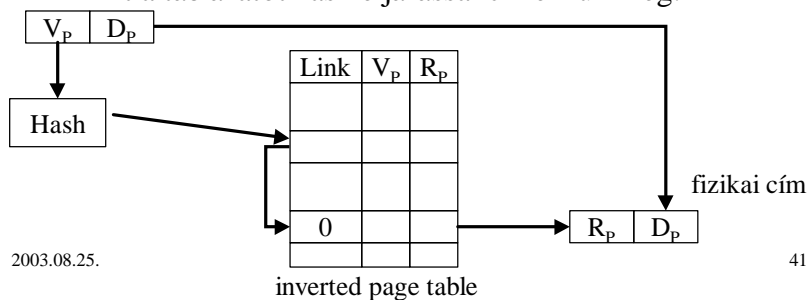
2003.08.25.

Markó Tamás, PTE TTK

40

Inverted page table (IPT) 1.

- A PT-ben minden logikai laphoz kellene egy bejegyzés ? nagyon nagy („superfluity“)
- Megoldás:
 - A táblában csak a **fizikai** lapokhoz van 1-1 sor (csak ezek érdekesek, az összes többi bejegyzésnél a presence flag úgyis 0 lenne).
 - Ezt a táblázatot hash eljárással címezzük meg.



2003.08.25.

41

Inverted page table 2.

- Az IPT szokásos hash-tábla ütközési listával.
- Ha szerepel benne egy V_p , akkor az valahol a link-listában van.
A hozzá tartozó R_p az, amit keresünk.
- Ha a $V_p[i]$ -nél $LINK[i] = 0$: nem ezt kerestük
? page fault.
- Az IPT közönséges memória ? olcsó
- Az IPT és a TLB kombinálható
- Alkalmazás pl. IBM RT, Motorola PowerPC

2003.08.25.

Markó Tamás, PTE TTK

42

A Pentium virtuális tárkezelése

2003.08.25.

Markó Tamás, PTE TTK

44

A memória szervezése

- Bájtanként címezhető
- 32 bites címbusz ? max. 2^{32} bájt (4 GB) fizikai címtartomány
- A memória kezelése:
 - szegmentálás és/vagy
 - lapozás
- Programonként max. 2^{14} (16.384) szegmens lehet, mindegyik max. 4 GB

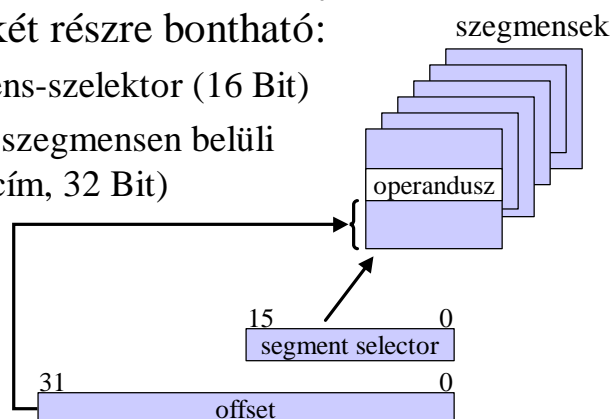
2003.08.25.

Markó Tamás, PTE TTK

45

A szegmentált címzés

- A szegmentált címtartományban egy cím két részre bontható:
 - Szegmens-szelektor (16 Bit)
 - Offset (szegmensen belüli relatív cím, 32 Bit)



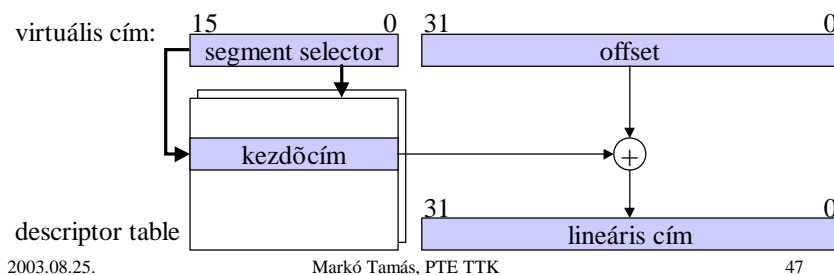
2003.08.25.

Markó Tamás, PTE TTK

46

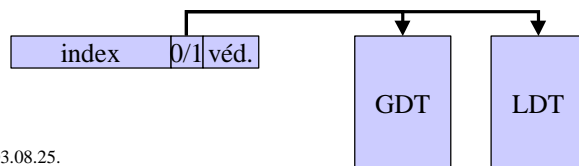
A fizikai cím kiszámítása

- A szelektor:
 - kiválasztja a *descriptor table*-t
 - megadja a táblán belüli indexet
- A tábla tartalmazza a szegmens kezdőcímét
 - kezdőcím + offset = a keresett fizikai cím



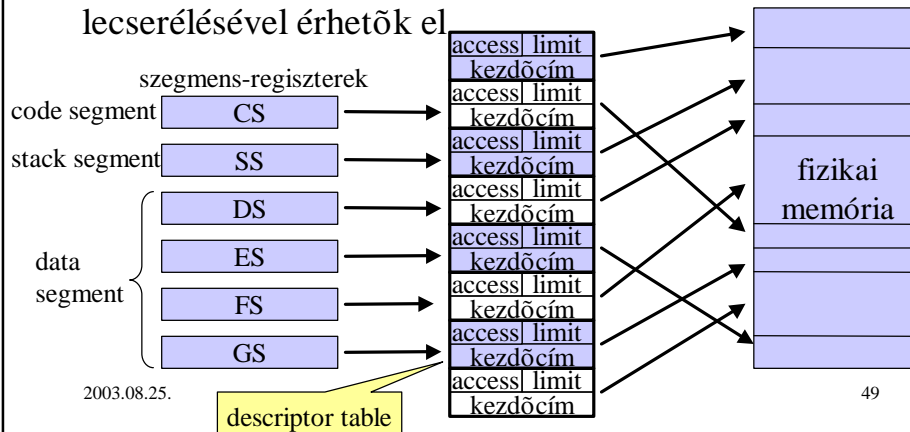
A segment selector

- 1 bit: melyik descriptor table
 - 0: global descriptor table (GDT)
minden programnak ugyanaz
 - 1: local descriptor table (LDT)
minden programnak egyéni
- 13 bit táblán belüli index
- 2 bit hozzáférési jogosultság (védelmi szint)



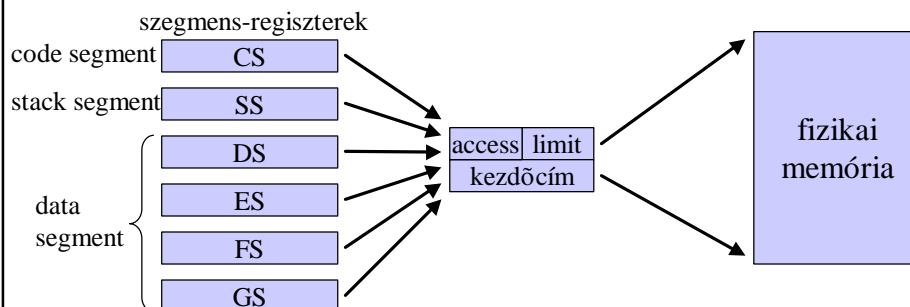
Többszegmensû memória-modell

- Programonként saját szegmensek
- 6 szegmens-regiszter ? 6 szegmens érhető el közvetlenül
- További szegmensek a szegmensregiszterek tartalmának lecserélésével érhetők el



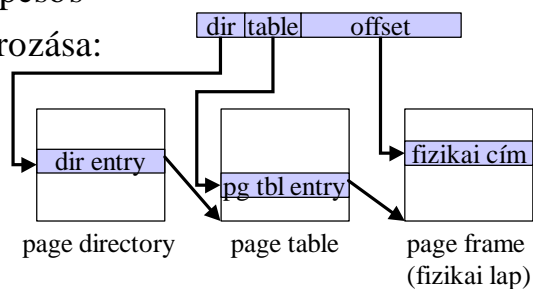
„Sík” (nem szegmentált) memória-modell

Minden szegmens mutathat ugyanarra a fizikai címtartományra is:



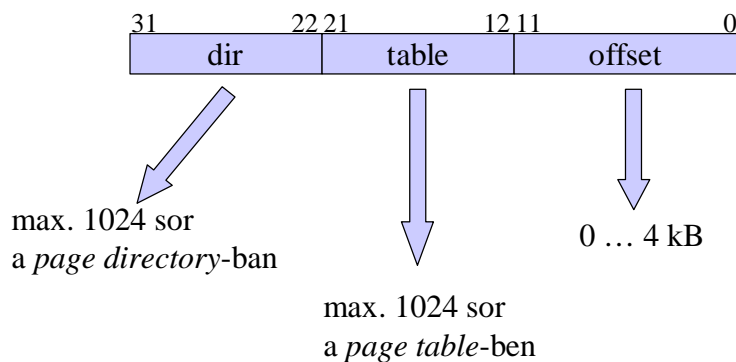
Lapozás

- A lapozás opcionális:
 - ha nincs lapozás: a szegmenscímzésből adódó cím a fizikai tárcím
 - ha van lapozás: a szegmenscímzésből adódó cím virtuális cím
- A lapméret 4 KB
- A lapozás kétlépcsős
- A cím meghatározása:



2003.08.25.

A virtuális cím felosztása

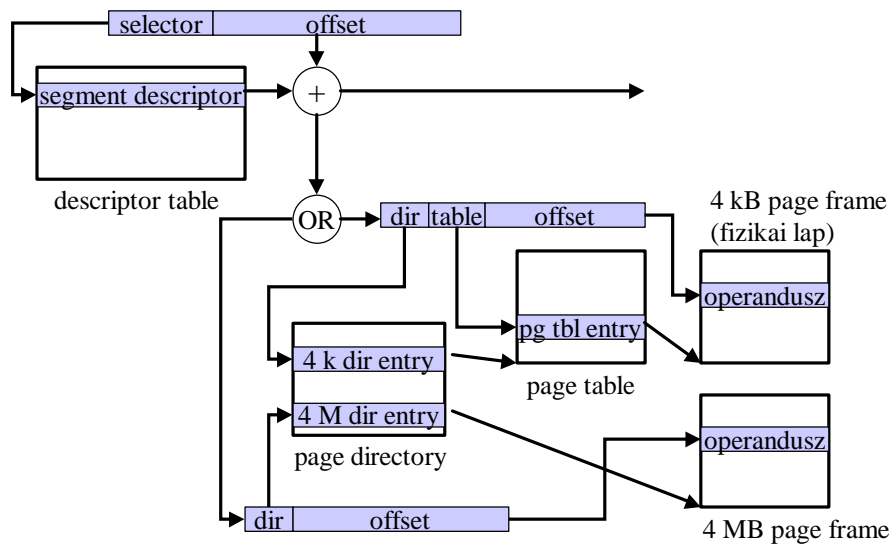


2003.08.25.

Markó Tamás, PTE TTK

52

A lapozás és a szegmentálás kombinálása



2003.08.25.

Markó Tamás, PTE TTK

53

A PowerPC virtuális tárkezelése

2003.08.25.

Markó Tamás, PTE TTK

54

A memória szervezése

- Memory Management Unit (MMU) read/write pufferral
- Változatos memória-modell
 - címszámítás 4 KB-os lapokra, változó blokkméretek, 256 MB-os szegmensek
 - 256 elemű TLB
 - Inverted Page Table
 - 52 bites virtuális címek, 32 bites fizikai címek

2003.08.25.

Markó Tamás, PTE TTK

55

A Memory Management Unit (MMU) 1.

- 2^{52} bájt (4 terabájt) virtuális memória
- 2^{32} bájt (4 gigabájt) fizikai memória
- Hozzáférési jogok és különböző lapméretek kezelése
- A címek az Instruction Unitból (IU) jönnek
- A cím felső részét az MMU fizikai címmé fordítja
- Az alsó rész (logikai = fizikai): index a cache-be (kiválaszt egy sort)
- Cache-miss: a lefordított felső részt konkaténáljuk az alsóval, ez lesz a fizikai cím
- Utasításcímek: kikeresés az ITLB-ből (4 bejegyzés)

2003.08.25.

Markó Tamás, PTE TTK

56

A Memory Management Unit (MMU) 2.

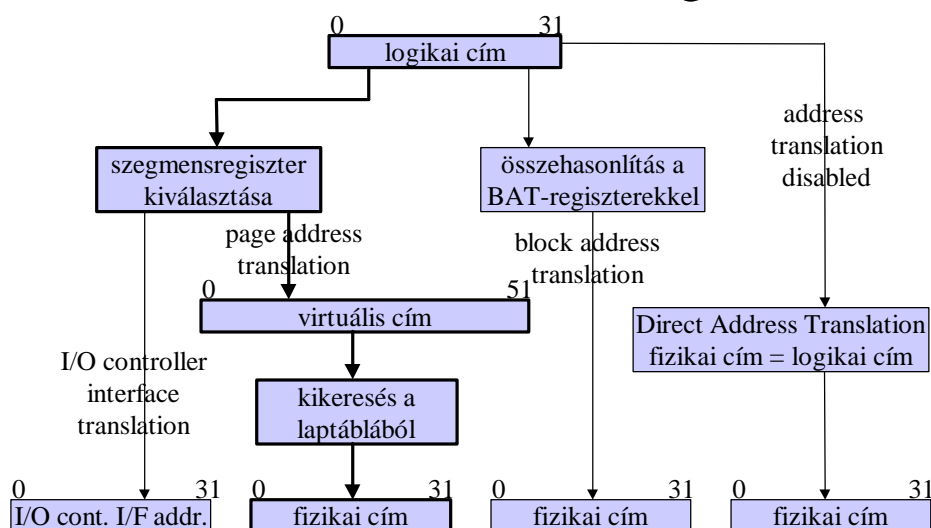
- A címtartomány felosztható
 - 256 MB-os szegmensekre, vagy
 - 128 KB - 8 MB-os blokkokra
- A szegmensek (kivéve az I/O-t) 4 KB-os lapokra oszthatók
- A lapok címének leírása: laptáblák (page table) a fizikai memóriában
- Gyorsítás: a laptáblák cache-elése a TLB-kben és a blokkinformációk cache-elése a BTLB-kben

2003.08.25.

Markó Tamás, PTE TTK

57

Címszámítási lehetőségek

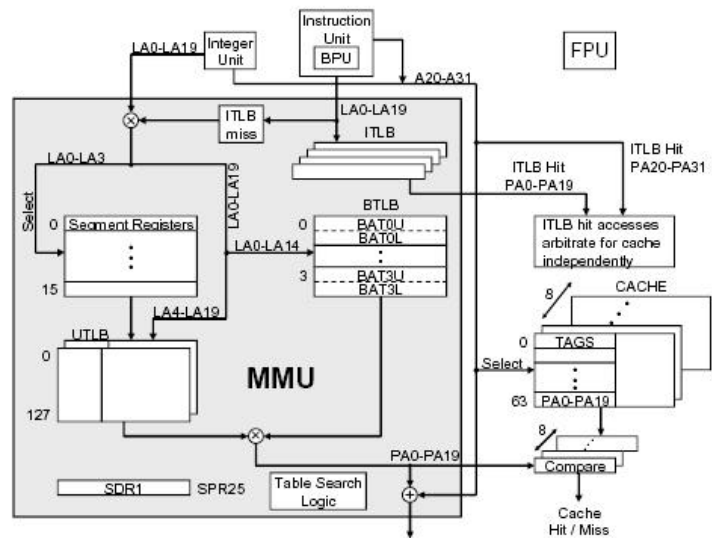


2003.08.25.

Markó Tamás, PTE TTK

58

Az MMU felépítése



2003.08.25.

Markó Tamás, PTE TTK

59

Az MMU szervezése

- Három TLB van:
 - Universal TLB (UTLB) az utasításoknak és az adatoknak, 256 bejegyzés, 2-es asszociativitás
 - Instruction TLB (ITLB) az utasításoknak, 4 bejegyzés, teljesen asszociatív
 - Block TLB, a supervisor-állapot BAT-registerei
- Utasításoknál:
 - 1. próbálkozás az ITLB-ben, találat esetén azonnal hozzáférés a cache-hez.
 - miss: 2. próbálkozás az UTLB-ben (a kiválasztott szegmensregiszterrel való átszámítás után)
- Adatoknál:
 - 1. próbálkozás a TLB-ben (a kiválasztott szegmensregiszterrel való átszámítás után)
 - UTLB-miss: keresés a laptáblában (PT)
 - Az UTLB-ben való kereséssel párhuzamosan: blokk keresése a BTLB-ben (siker esetén a blokkcím a domináns)

60

A blokkcímzés működése

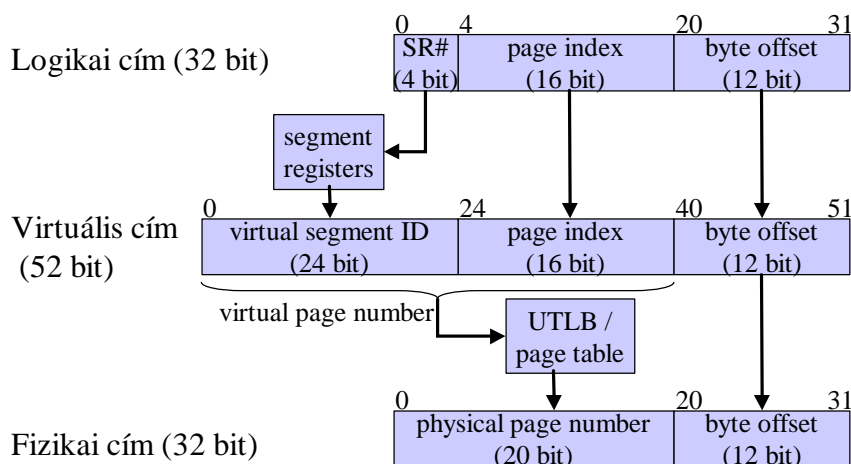
- A blokkcímzés eredménye felülírja a lapcímzést
- A cím meghatározása a BTLB segítségével történik:
 - 4 pár BAT-regiszter (*BATiU / BATiL, upper/lower*) (ezek az SPR528-SPR535 nevű speciális célú regiszterek)
 - minden pár tartalmazza egy blokk logikai kezdőcímét, a méretét és a fizikai kezdőcímét
 - a logikai cím első 15 bitjét összehasonlítjuk a BAT-regiszterekben lévő logikai kezdőcímekkel
 - egyezés esetén kiolvassuk a fizikai kezdőcímét

2003.08.25.

Markó Tamás, PTE TTK

61

A lapcímzés működése



2003.08.25.

Markó Tamás, PTE TTK

62