A feltételes ugró utasítások eldugaszolják a csővezetéket

- Feltételes végrehajtás
- Predikáció

Feltételes végrehajtás (5.51-52. ábra):

C pr. rész	Általános assembly	Feltételes végrehajtás

if(R1 == 0)
R2 = R3;
	CMP R1, 0
	BNE L1
	MOV R2, R3
	L1: …
		CMOVZ R2, R3, R1

CMOVZ R2, R3, R1 csak akkor hajtja végre **R2 = R3** -t, ha **R1= 0**.

Feltételes végrehajtás (5.51-52. ábra):

Hasonlóan:

C pr. rész	Általános assembly	Feltételes végrehajtás

if(R1 == 0) {
R2 = R3;
R4 = R5;
} else {
R6 = R7;
R8 = R9;
}
	CMP R1, 0
	BNE L1
	MOV R2, R3
	MOV R4, R5
	BR L2
	L1: MOV R6, R7
	MOV R8, R9
	L2: …
		CMOVZ R2, R3, R1
		CMOVZ R4, R5, R1
		CMOVN R6, R7, R1
		CMOVN R8, R9, R1

CMOVN R6, R7, R1 csak akkor hajtja végre **R6 = R7** -et, ha **R1≠ 0**.

Predikáció, IA – 64 (5. 53. ábra)

64 predikátum regiszter: 1 bites regiszterek, többnyire párban. Az **IA – 64** minden utasítása predikátumos. **CMPEQ R1, R2, P4** beállítja **P4**-et és törli **P5**-öt, ha **R1 = R2**, különben **P5**-öt állítja be és **P4**-et törli.

if(R1 == R2)
R3 = R4 + R5;
else
R6 = R4 – R5;
	CMP R1, R2
	BNE L1
	MOV R3, R4
	ADD R3, R5
	BR L2
	L1: MOV R6, R4
	SUB R6, R5
	L2: …
		CMPEQ R1, R2, P4
		<P4>ADD R3, R4, R5
		<P5>SUB R6, R4, R5

Operációs rendszer szintje

Operating System Machine (OSM)

Ezen a szinten programozóknak rendelkezésre állnak a felhasználói módban használható **ISA** szintű utasítások és az operációs rendszer által hozzáadott utasítások: **rendszerhívások** (**system calls**). Ezeket az operációs rendszer eljárásai valósítják meg (értelmezés).

Virtuális memória

Régen nagyon kicsi volt a memória. Sokszor nem fért el az egész program a memóriában.

Overlay (átfedés): A program több része fut ugyanazon a memória területen, mindig az aktuálisan futó rész van a memóriában, a többi rész mágneslemezen van.
A programozó dolga a feladat átfedő részekre bontása, és a részek mozgatása a memória és a háttér tároló között.

Ma már sokkal nagyobb ugyan a memória, de még sokkal nagyobb lehet a **címtartomány** (**address space**).

Virtuális címtartomány: azok a címek, amelyekre a program hivatkozni tud.

Fizikai címtartomány: azok a címek, amelyek tényleges memória cellát címeznek.

A virtuális és fizikai címtartomány ugyanolyan méretű lapokra van osztva (**6.3. ábra**). A fizikai „lapokat” **lapkeret**nek (**page frame**) nevezzük.

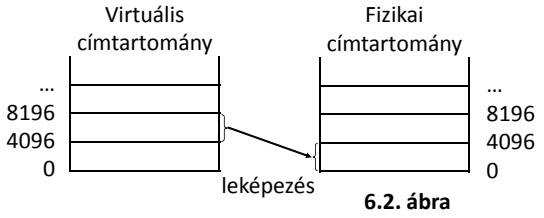
Lap méret: 512 B – 64 KB (– 4 MB), **mindig 2 hatványa**.

Lap	Virtuális címek	Lapkeret	Fizikai címek
N	-	n	-
…	… - …	…	… - …
4	16384 - 20479	4	16384 - 20479
3	12288 - 16383	3	12288 - 16383
2	8192 - 12287	2	8192 - 12287
1	4096 - 8191	1	4096 - 8191
0	0 - 4095	0	0 - 4095

A virtuális címtartomány sokkal nagyobb, mint a fizikai!

Mit kell tenni, ha olyan címre történik hivatkozás, amely nincs a memóriában?

1. Egy lapkeret (pl. a 0-4095) tartalmának lemezre mentése.
2. A kérdéses lap megkeresése a lemezen.
3. A kérdéses lap betöltése a lapkeretbe.
4. A memória térkép megváltoztatása: pl. a 4096 és 8191 közötti címek leképezése a betöltött lapkeret címtartományába.
5. A végrehajtás folytatása.

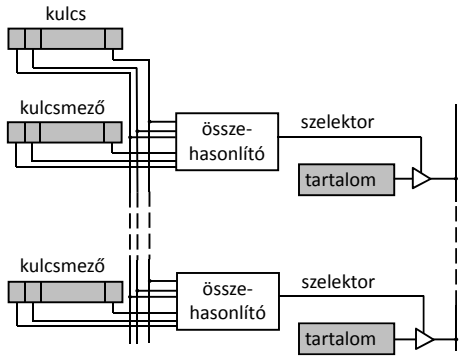


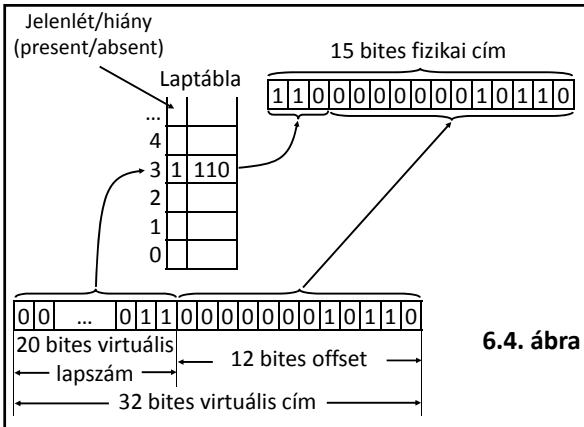
6.2. ábra

A virtuális címek fizikai címekre történő leképezését az **MMU** (Memory Management Unit) – memória kezelő egység végzi.

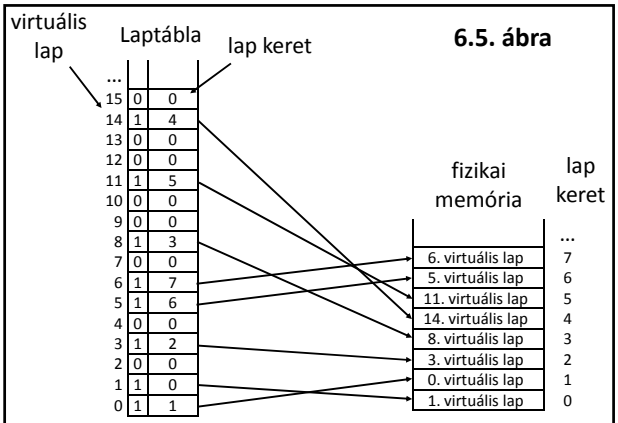
Memória térkép (**memory map**) vagy **laptábla** (**page map**) kapcsolja össze a virtuális címeket a fizikai címekkel. Pl. 4 KB-os lapméret és 32 bites virtuális cím esetén 1 millió virtuális lap van, ezért 1 millió bejegyzésű laptáblára van szükség. 32 KB fizikai memória esetén csak 8 lapkeret van, ezért a leképezés megoldható 8 cellás asszociatív memóriával is (a gyakorlatban több ezer lapkeret van, és az asszociatív memória igen drága).

Asszociatív memória





6.4. ábra



Máté: Architektúrák **Csak az elv kell** 12. előadás 13

Laphiány (**page fault**): a lap nincs a memóriában.

Kérésre lapozás (**demand paging**): lapozás csak laphiány esetén. A program egyetlen bájtja sem kell bent legyen a memóriában, csak a másodlagos tárolón.

Máté: Architektúrák 12. előadás 14

Munka halmaz (**working set**): az operációs rendszer feladata megállapítani

Tanenbaum: a legutóbbi ***k*** memória hivatkozásban szereplő lapok halmaza.

Wikipedia: a legutóbbi **τ** időben hivatkozott memória lapok halmaza.

Ha a munkahalmaz nagyobb, mint a lapkeretek száma, akkor gyakori lesz a laphiány. A nagyon gyakori laphiányt **vergődés**nek (**thrashing**) nevezzük.

Időosztásos rendszerekben a kérésre lapozás nem kielégítő, a munkahalmazban lévő lapok előre visszatölthetők.

Máté: Architektúrák 12. előadás 15

Lapkezelési eljárások: melyik lap **helyett** töltsük be a kért lapot?

LRU (Least Recently Used, legrégebben használt): általában jó, de nem jó pl. 9 lapon átnyúló ciklus esetén, ha csak 8 memória lap van (**6.6. ábra**).

7. Virtuális lap	7. Virtuális lap	7. Virtuális lap
6. Virtuális lap	6. Virtuális lap	6. Virtuális lap
5. Virtuális lap	5. Virtuális lap	5. Virtuális lap
4. Virtuális lap	4. Virtuális lap	4. Virtuális lap
3. Virtuális lap	3. Virtuális lap	3. Virtuális lap
2. Virtuális lap	2. Virtuális lap	2. Virtuális lap
1. Virtuális lap	1. Virtuális lap	0. Virtuális lap
0. Virtuális lap	8. Virtuális lap	8. Virtuális lap

Máté: Architektúrák **Csak az elv kell** 12. előadás 16

FIFO (First-in First-Out, először be, először ki): egyszerűbb (de most ez se jobb, mint **LRU**).

Csak a módosult (**dirty**, szennyezett) lapokat kell visszaírni, a tisztát (**clean**) nem (**szennyezés bit**). Most is előnyös, ha az utasítások és az adatok elkülönülten helyezkednek el a memóriában: az utasításokat nem kell visszaírni.

Máté: Architektúrák 12. előadás 17

Lapméret és elaprózódás

Ha egy program ***k*** lapon fér el, akkor általában a ***k***-dik lap nincs tele.

Ha a lap mérete ***n***, akkor programonként átlagosan ***n*/2** bájt kihasználatlan:
belső elaprózódás (**internal fragmentation**).

A belső elaprózódás ellen a lap méretének csökkentésével lehet védekezni, de ez a laptábla méretének növekedéséhez vezet.

A kis lap előnytelen a lemez sávszélességének kihasználása szempontjából is, viszont kisebb a vergődés kialakulásának valószínűsége.

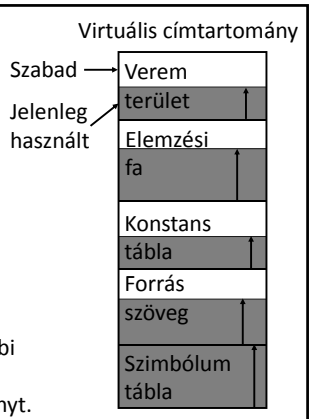
Máté: Architektúrák 12. előadás 18

Szegmentálás

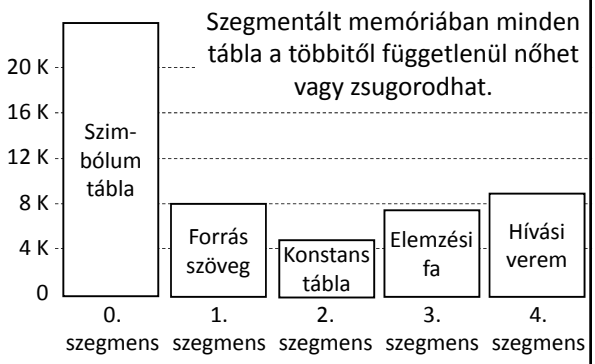
Egy fordítóprogramnak a következő célokra kellhet memória (**6.7. ábra**):

- szimbólum tábla,
- forrás kód,
- konstansok,
- elemzési fa,
- verem.

Rögzített memória felosztás esetén ezek egyike kicsinek bizonyulhat, miközben a többi nem használja ki a rendelkezésére álló tartományt.



Szegmentálás (6.8. ábra)



Szegmens (6.9. ábra)

A programozó számára látható logikai egység. Minden szegmens címtartománya 0-tól valamilyen maximumig terjed. A szegmens tényleges mérete ennél kisebb lehet. A program számára a címtartomány két dimenziós: (szegmens, offset).

Általában egy szegmensben csak egyféle dolgok vannak: vagy kód vagy konstans vagy … Különböző tárvédelmi lehetőségek:

- kód: csak végrehajtható, nem írható, nem olvasható,
- konstans: csak olvasható
- …

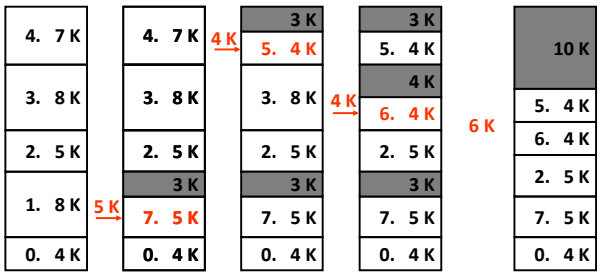
A szegmentálás és a virtuális memória összehasonlítása (6.9. ábra)

Szempontok	Lapozás	Szegmentálás
Tudnia kell róla a programozónak?	Nem	Igen
Hány lineáris címtartomány létezik?	1	Több
Meghaladhatja-e a virtuális címtartomány nagysága a fizikai memória méretét?	Igen	Igen
Könnyen kezelhetők a változó méretű táblák?	Nem	Igen
Mi ennek a technikának a lényege?	Nagy memória szimulálása	Több címtartomány biztosítása

A szegmentálás megvalósítása

Lapozással: Minden szegmensnek saját laptáblája van. A szegmens néhány lapja a memóriában van.

Cseréléssel: Teljes szegmensek mozognak a memória és a lemez között. Ha olyan szegmensre hivatkozunk, amely nincs a memóriában, akkor betöltődik. **Külső elaprózódás**hoz (**external fragmentation**) vezethet (**6.10. ábra**). **Lyukacsosodás**nak (**checkerboarding**) is nevezik.

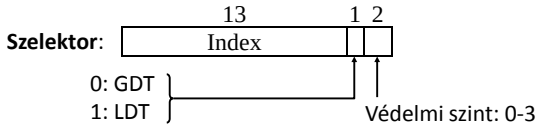


Összepréselés: idő igényes, de időnként kell.

Legjobb illesztés (**best fit**) és **első illesztés** (**first fit**) algoritmus. Az utóbbi gyorsabb és jobb is az általános hatékonyság szempontjából. **???**

Pentium 4 (6.12-14. ábra)

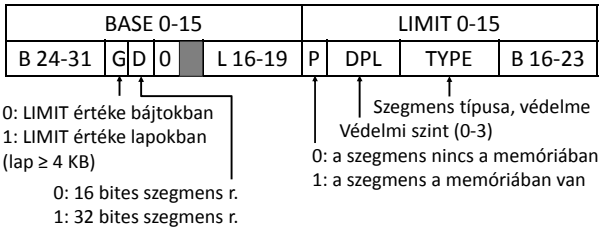
A szegmens regiszter tartalmazza a **szelektor**t.



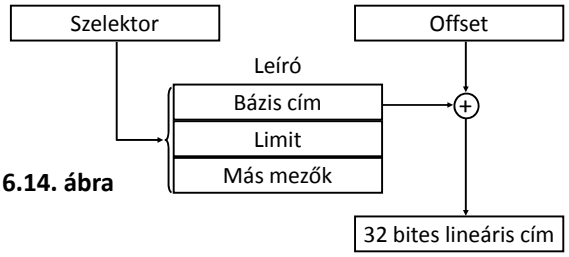
A szelektor (**6.12. ábra**) indexe választja ki a **leíró**t (**descriptor**) a **lokális** (**LDT, Local Descriptor Table**) vagy **globális leíró táblából** (**GDT**, **Global Descriptor Table**). (**6.13. ábra**).

A **0**. leíró használata csapdát eredményez (hiba).

Pentium 4 kódszegmensének leírója (6.13. ábra)



Ha **P=0**, csapda: nem létező szegmens, vagy be kell tölteni a szegmenst.



6.14. ábra

Ha offset (a szegmens elejéhez viszonyított relatív cím) a szegmens határán túl van, csapda (hiba).

Lapozást tiltó flag (a globális vezérlőregiszter bitje):

Ha engedélyezett: lineáris cím = virtuális cím

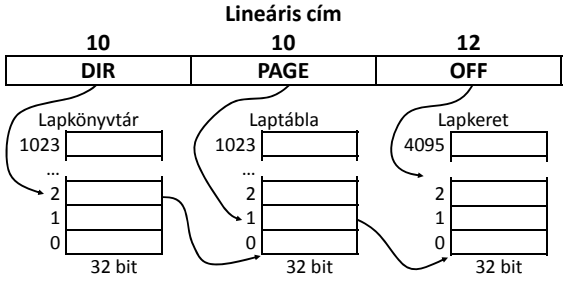
Ha tiltott: lineáris cím = fizikai cím

Lapkönyvtár (page directory 6.15. ábra)

A 32 bites lineáris címek és a 4 KB-os lapok miatt egy szegmenshez egymillió lap is tartozhat. Túl sok!

Minden futó programhoz egy l**apkönyvtár** tartozik.

Minden bejegyzés egy **laptáblára** mutat, vagy sehova.



A lapkönyvtárnak azokhoz a mutatóihoz, amelyek nem mutatnak sehova, nem kell helyet foglalni a laptábla számára (pl. csak két db. 1 K, és nem egy 1 M bejegyzésű tábla kell egy 4 MB-nál rövidebb szegmenshez).

A táblákban minden bejegyzéshez 32 bit áll rendelkezésre. A mutatókhoz nem használt biteket a hardver az operációs rendszer számára hasznos jelzésekkel tölti ki (védelem, szennyezettség, hozzáférés, …).

Speciális hardver támogatja a legutóbb használt lapok gyorsabb elérését.

A Pentium 4 védelmi rendszere (6.16. ábra)

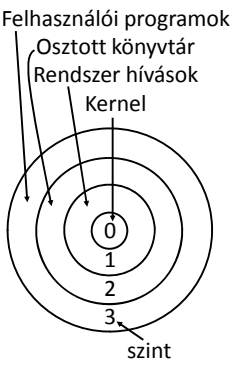
A futó program pillanatnyi szintjét a **PSW** tartalmazza.

A program a saját szintjén lévő szegmenseket szabadon használhatja.

Magasabb szinten lévő adatokhoz hozzáfér, de az alacsonyabb szinten lévők kezelése csapdát okoz.

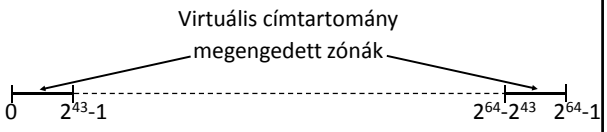
Más szinten lévő eljárás hívásánál **CALL** helyett szelektort kell alkalmazni, ez egy **hívás kapu**t (**call gate**) jelöl ki (más védelmi szintre csak szabványos – tehát ellenőrzött – belépési ponton lehet áttérni).

A szintek egy lehetséges felhasználása:



Az UltraSPARC III virtuális memóriája

Virtuális cím 64 bites, egyelőre 44 bitre korlátozva.



44 bitre korlátozva ez a címtartomány folytonos.

Fizikai címtartomány maximum 41 bites.

A kód és adat lapokat külön kezeli.

Lapméret: 8, 64, 512 KB és 4 MB (**6.17. ábra**).

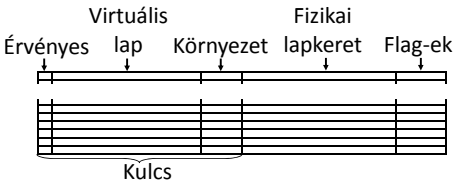
Lap mérete	Virtuális lap címe (bit)	OFFSET (bit)		Fizikai lap címe (bit)	OFFSET (bit)
8 KB	51 (31)	13	→	28	13
64 KB	48 (28)	16	→	25	16
512 KB	45 (25)	19	→	22	19
4 MB	42 (22)	22	→	19	22

44 bitre korlátozva maximum 41 bit

A memória kezelő egység (**MMU**) három szinten dolgozik:

- A legutóbb használt lapokat gyorsan megtalálja (hardver). A kód és az adat lapokat teljesen külön kezeli.
- A nem nagyon régen használtakat már lassabban (hardver segítséggel).
- A nagyon régen használtakat csak hosszas keresés után (szoftveres úton).

TLB (Translation Lookaside Buffer) a legutóbb használt 64 lap bejegyzését tartalmazza (**6.18. ábra**).



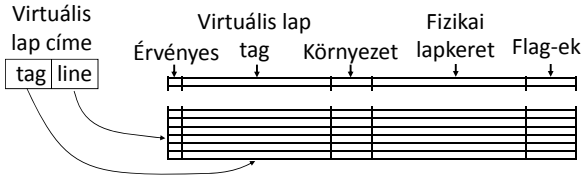
Környezet (context): processzus szám.

Asszociatív memória: Kulcs a keresett virtuális lap és a környezet.

TLB hiány (**TLB miss**) esetén: csapda.

TLB hiány esetén **TSB** folytatja a keresést (szoftver).

TSB (Translation Storage Buffer): olyan felépítésű, mint egy direkt leképezésű gyorsító tár (operációs rendszer építi fel, és kezeli a központi memóriában).



TSB találat esetén egy **TLB** sor helyébe beíródik a kért lapnak megfelelő bejegyzés.

TSB hiány esetén a **fordítótábla** (**translation table**) alapján keres. Ennek a táblának a szerkezetét az operációs rendszer határozza meg.

Egy lehetséges megoldás a tördeléses eljárás. Ebben az esetben a memóriába töltött virtuális lapok és a nekik megfelelő fizikai lapkeretek sorszáma listákba van helyezve. Ha a virtuális lap sorszáma ***p***-vel osztva ***q***-t ad maradékul, akkor csak a ***q***-adik listát kell végignézni.

Ha ez se találja a keresett lapot, akkor nincs a memóriában.

Virtuális memória és gyorsító tár

Két szintű hierarchia:

Virtuális memória használatakor az egész programot lemezen tartjuk, fix méretű lapokra osztjuk.
Lap hiány esetén a lapot a központi memóriába töltjük (operációs rendszer).

Gyorsító tár esetén a központi memóriát gyorsító sorokra osztjuk.
Gyorsító sor hiány esetén a gyorsító sort a gyorsító tárba töltjük (hardver).

Feladatok

Mit értünk feltételes végrehajtáson?
Mi a feltételes végrehajtás előnye?
Mit értünk predikáción?
Hogy küszöböli ki a feltételes végrehajtás és a predikáció a csővezeték elakadását? Jelent-e ez késleltetést a program futásában?
Milyen utasítások érhetők el operációs rendszer szinten?
Mi az overlay technika lényege?
Mi a virtuális címtartomány?
Mi a fizikai címtartomány?
Mi a lap és mi a lapkeret?

Feladatok

Mi a lapozás?
Mi a memória térkép (laptábla)?
Mi az MMU?
Hogy működik az asszociatív memória?
Mi a laphiány?
Mi a kérésre lapozás?
Mi a munka halmaz (working set)?
Mikor alakul ki vergődés?
Milyen lapkezelési eljárásokat ismer?
Mi a belső elaprózódás?
Mi az előnye, és mi a hátránya a kis lapméretnek?

Feladatok

Mit nevezünk szegmentálásnak?
Hogy valósítható meg a szegmentálás?
Mik a szegmentálás előnyei?
Mi a külső elaprózódás?
Mi az összepréselés (compaction)?
Hogy valósul meg a szegmens címzés a Pentium 4-en?
Mi a szelektor?
Milyen információt tartalmaz a Pentium 4 szelektora?
Milyen mezőket tartalmaz a kódszegmensek leírója?

Feladatok

Mire szolgál az LDT (Local Descriptor Table) és a GDT (Global Descriptor Table)?
Hogy képződik a lineáris cím?
Hogy valósul meg Pentium 4-en a virtuális címzés?
Milyen a Pentium 4 védelmi rendszere?
Hogy hívható más védelmi szintű eljárás?
Jellemezze az UltraSparc III virtuális memóriáját!
Mi a TLB (Translation Lookaside Buffer)?
Milyen memóriában van a TLB?
Mi történik TLB hiány esetén?
Hogy szervezett a TSB (Translation Storage Buffer)?
Mi történik TSB hiány esetén?
Hasonlítsa össze a virtuális memóriát a gyorsító tárral!

Az előadáshoz kapcsolódó Fontosabb témák

Feltételes végrehajtás, predikáció.
Operációs rendszer szintje. Virtuális memória.
Virtuális memória. Lapméret, elaprózódás
Szegmentálás.
A Pentium 4 és az UltraSPARC III virtuális memóriája.