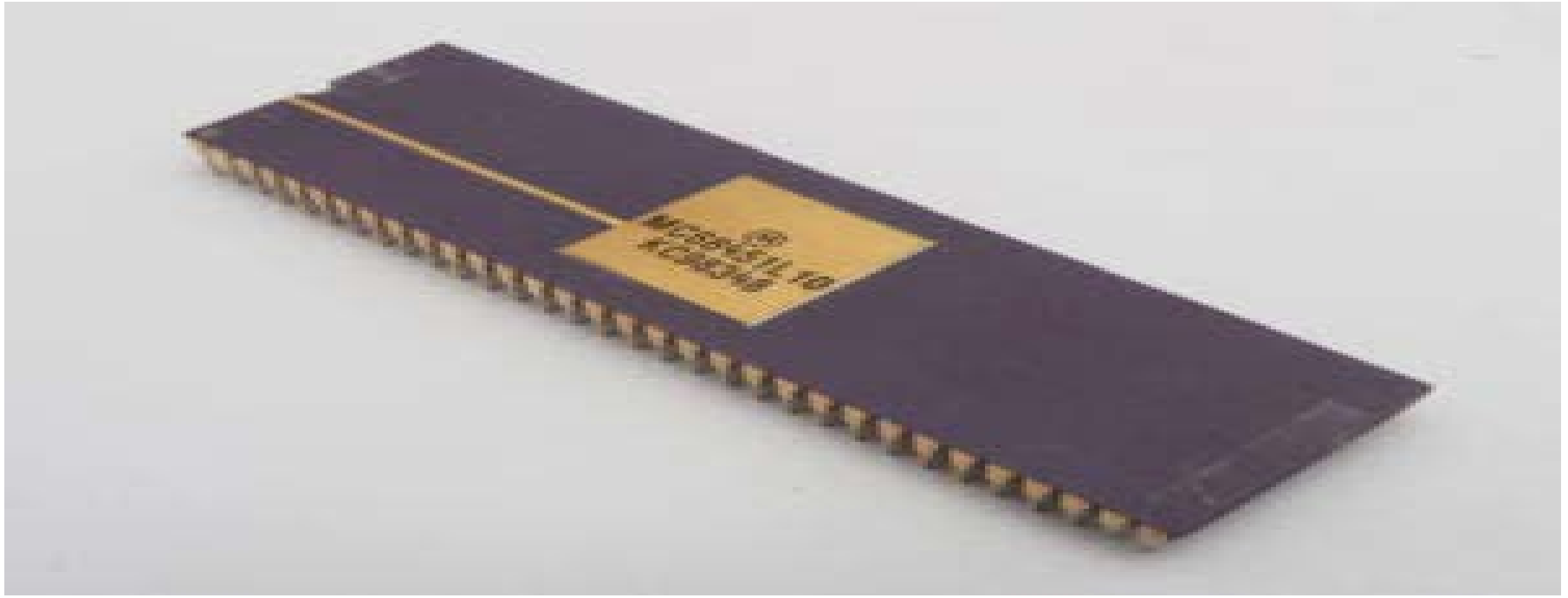
Operációsrendszerek 3. Memóriakezelés

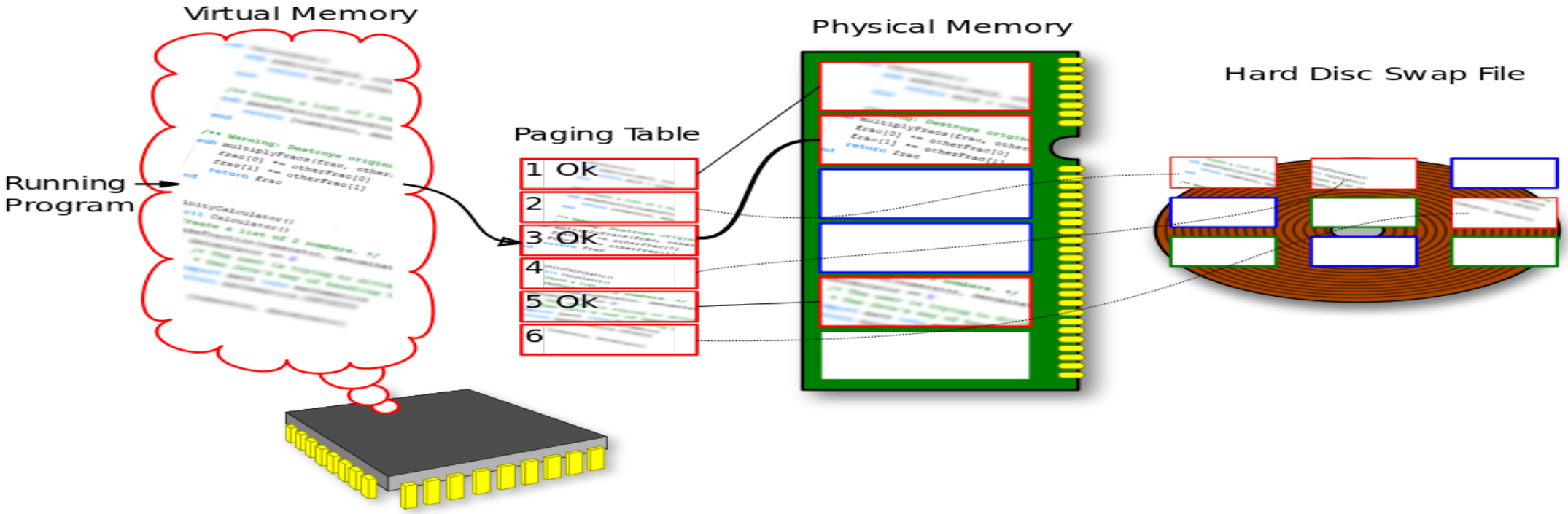


IT alapok

Varga Tibi 2019

A memóriakezelés feladatai

- Adatok elhelyezése a memóriában
  - Programok betöltése
  - Adatterületek biztosítása programok számára
- Memória megosztása a különböző folyamatok között



Programok betöltése

Feladat: végrehajtható programot be kell olvasni a háttértárról a memóriába és ott el kell indítani

Végrehajtható program meghatározott címekkel dolgozik

programot mindig ugyanoda kell tölteni

Memória megosztása:

minden folyamat más címtartományhoz férhet hozzá

a programjaik is különböző címeken vannak

Felhasználói adatterületek

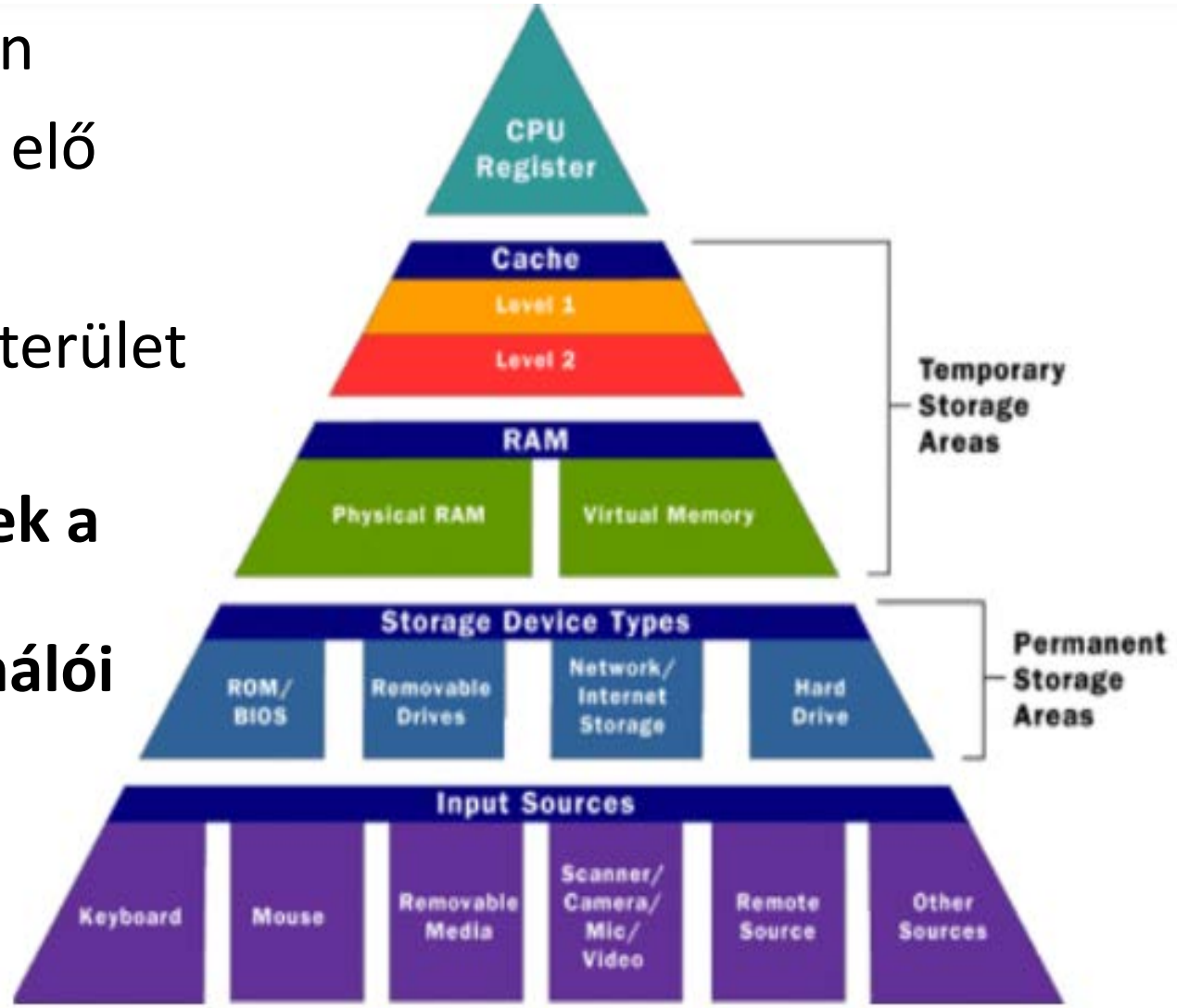
Adatok keletkezése a memóriában

- Felhasználói program állítja elő
- Háttértárról töltődik be

Mindkét esetben kell egy szabad terület ahol elfér

Operációsrendszer feladata ennek a területnek a biztosítása és kezdőcímének közlése a felhasználói programmal

Szükséges rendszerszolgáltatás: **memóriaigénylés**

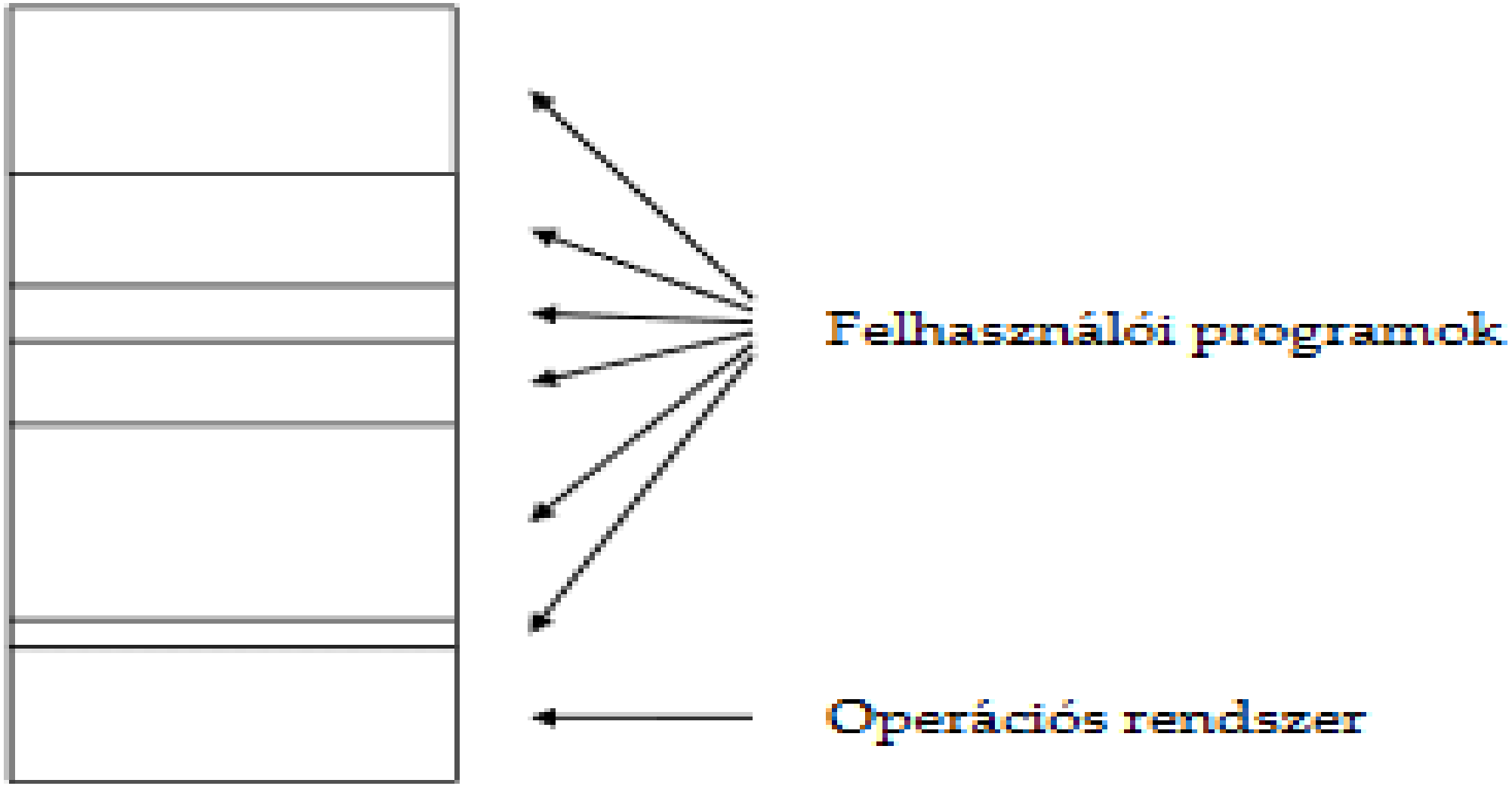


Memóriafelosztása

Több módszer ismert:

- **Memória partícionálás**
  - ✓ Fixszámú és méretű partíció (nem hatékony)
  - ✓ Dinamikusanváltozó számú és méretű partíciók
- **Virtuálismemória kezelés**
  - ✓ Lapozás
  - ✓ Szegmentálás
  - ✓ Hibrid megoldások
- Egyes módszereken belül is sok különböző megvalósítás és sok különböző algoritmus létezik

Memória partícionálás



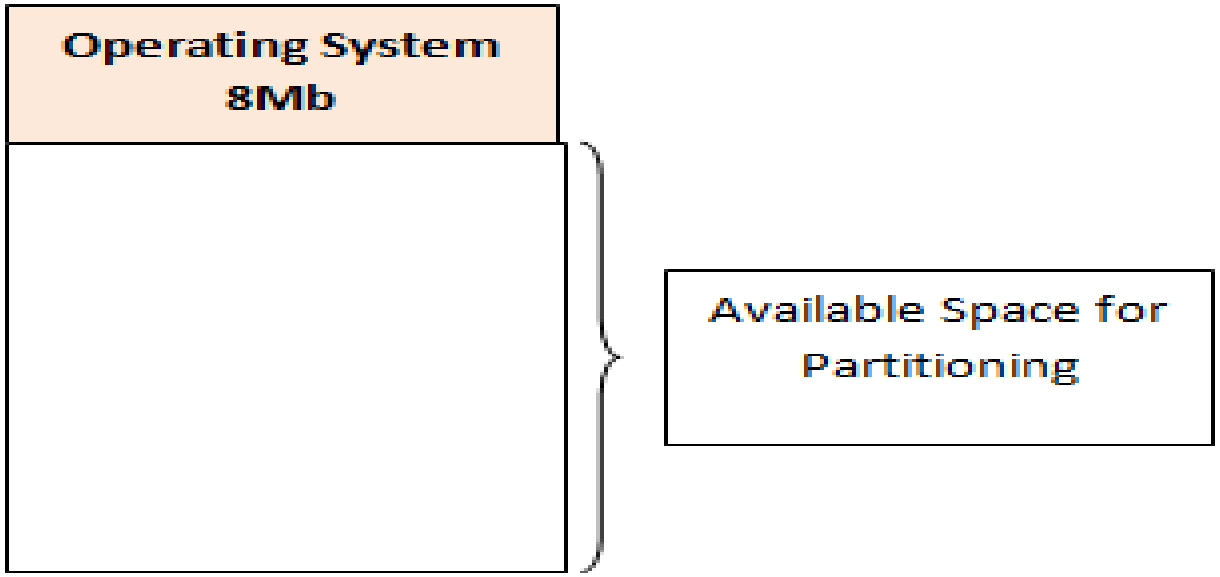
Rögzített memória partíciók

- Rendszergazda definiálja a partíciók számát és az egyes partíciók méretét
- Legelső partícióban az operációsrendszer kódja van
- Minden folyamat olyan partícióba kerül amelybe befér a programja
- Ha egy program nem fér be egyik szabad partícióba sem, a háttértáron várakozik

Partition Size	Memory Address	Access	Partition Status
100K	200K	Job 1	Busy
25K	300K	Job 4	Busy
25K	325K		Free
50K	350K	Job 2	Busy

Dinamikus memória partíciók

- Csak is az operációsrendszernek van előre rögzített méretű partíciója
- Minden folyamat létrejöttekor az OS keres neki egy helyet a memóriában ahol elfér és létrehoz számára egy memória partíciót
- Ha egy folyamat véget ér, megszűnik a partíciója is
- Ha nincs olyan összefüggő szabadterület, ahol elférne, a háttértáron várakozik



Dinamikus memória partíciók(2)

Operációsrendszer feladata:

- Partíciók nyilvántartása
- A szabad területek nyilvántartása(pl.láncoltlistában), szomszédos darabok összeolvasztása
- Megfelelő méretű szabadterület kiválasztás a egy új folyamat számára

Problémák:

- Pazarlás: kis folyamatok elfoglalják a nagy szabad területeket a nagyok elől
- Elaprózódás: sok kis szabad terület marad, amelyekben nem férnek el folyamatok, de összesen sok területet foglalnak el

Dinamikus memória partíció kezelés

A dinamikus memória kezelés esetén különböző memória kezelő algoritmusokkal keresik meg a szabad helyeket ahova a program és az adatok betölthetőnek:

- **First Fit** – első megfelelő méretű partíciót választjuk a szabadlistából
- **Next Fit** – ezelőző módosítása: nem az elejéről indulunk, hanem a legutóbb létre jött folyamat számára talált partíciótól
- **Best Fit** – Végig nézzük az összes szabad területet, és a legkisebb olyat választjuk, amelyikbe befér
- **Worst Fit** – A lehető legnagyobb szabad területet választjuk

Minden algoritmusnál célszerűbb bájtok helyett blokkokban foglalni a memóriát, mert a néhány bájtos részek partíciónak alkalmatlanok és csak lassítják a keresést

Virtuális memória

Alap problémák:

- Előfordulhat, hogy egy programteljesegészében egyáltalán nem fér be memóriába
- Mozgatása háttértárra és vissza sok erőforrást igényel
- Programok rövid idő alatt csak kis részét használják a memória területüknek
- Biztonsági probléma: nehéz elérni, hogy a programok ne nyúlhassanak ki a memória partícióból
- Nem mindig tudhatjuk a program indulásakor, hogy mennyi memóriára lesz szükség, azt dinamikusan kell lefoglalni

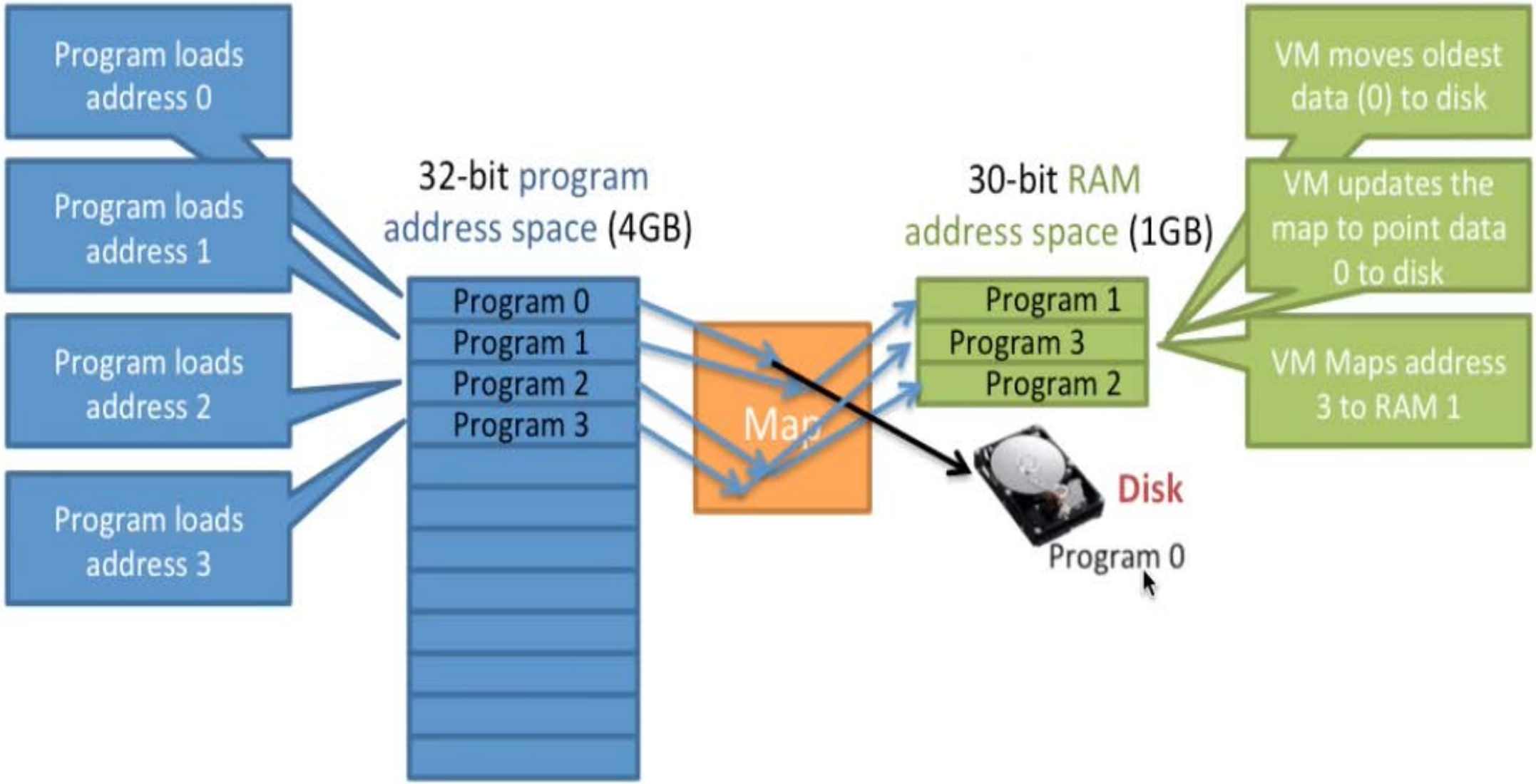
<u>Ötlet</u>: programok által látott memória terület különbözzön a fizikailag létezőtől!

Virtuálismemória követelmények

Követelmények:

- Minden program egy saját memóriaterületet lásson, mintha az egész memória az övé volna
- Bármely címre lehessen hivatkozni a területen belül, és az adatok permanensen tárolódjanak ott (mint a fizikai memóriában)
- Program ne vegyen észre semmit a megvalósítás módjáról
- Virtuálismemória elérésének hatékonysága ne legyen sokkal rosszabb, mint a fizikaié

Virtuális memória



Virtuális memória megvalósítása

Megvalósítás:

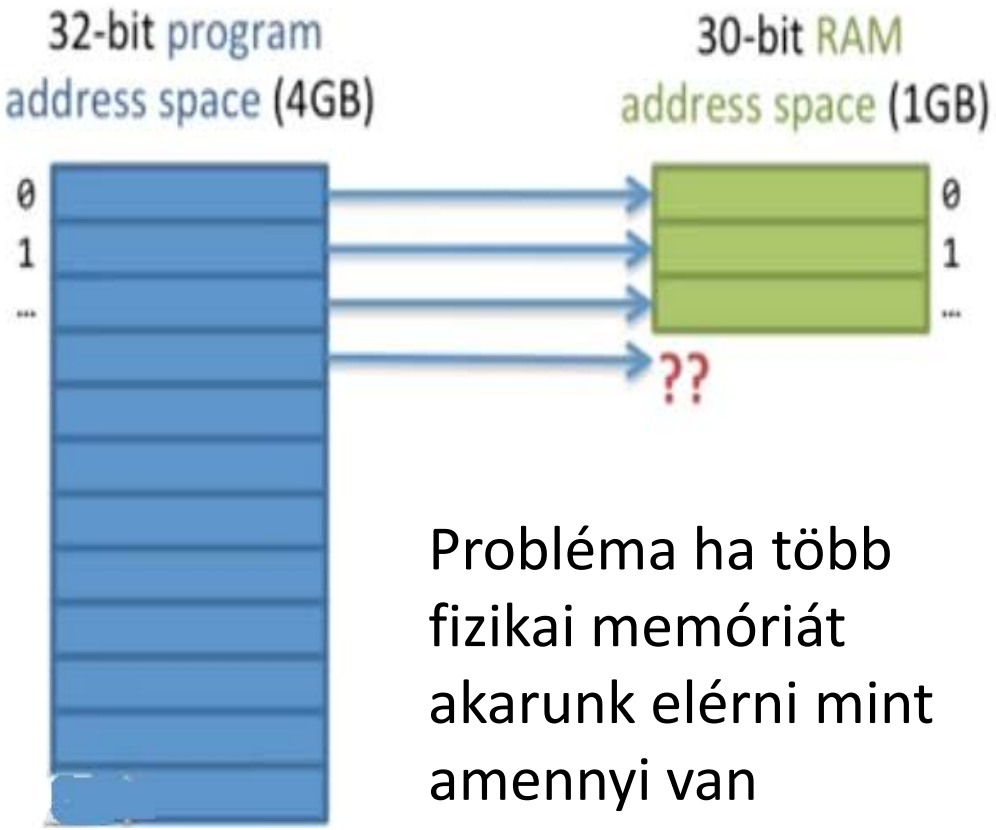
- Mind a virtuális, mind a fizikai memóriát felosztjuk, és a részeket egy memória térképben egymáshoz rendeljük
- Egyes virtuális memóriabeli területekhez nem rendelünk területet a fizikaimemóriából
    - Egyáltalán nem rendelünk hozzá területet: üres memóriaterület
    - Háttértáron tároljuk az adatait
- Ha hivatkozás történik egy olyan címre, amelyhez nem tartozik fizikai memóriaterület kivétel keletkezik:

Kivételkezelő feladata a hozzárendelések megváltoztatása úgy, hogy tartozzon hozzá

Virtuális memória megvalósítása

Virtuális memória nélkül

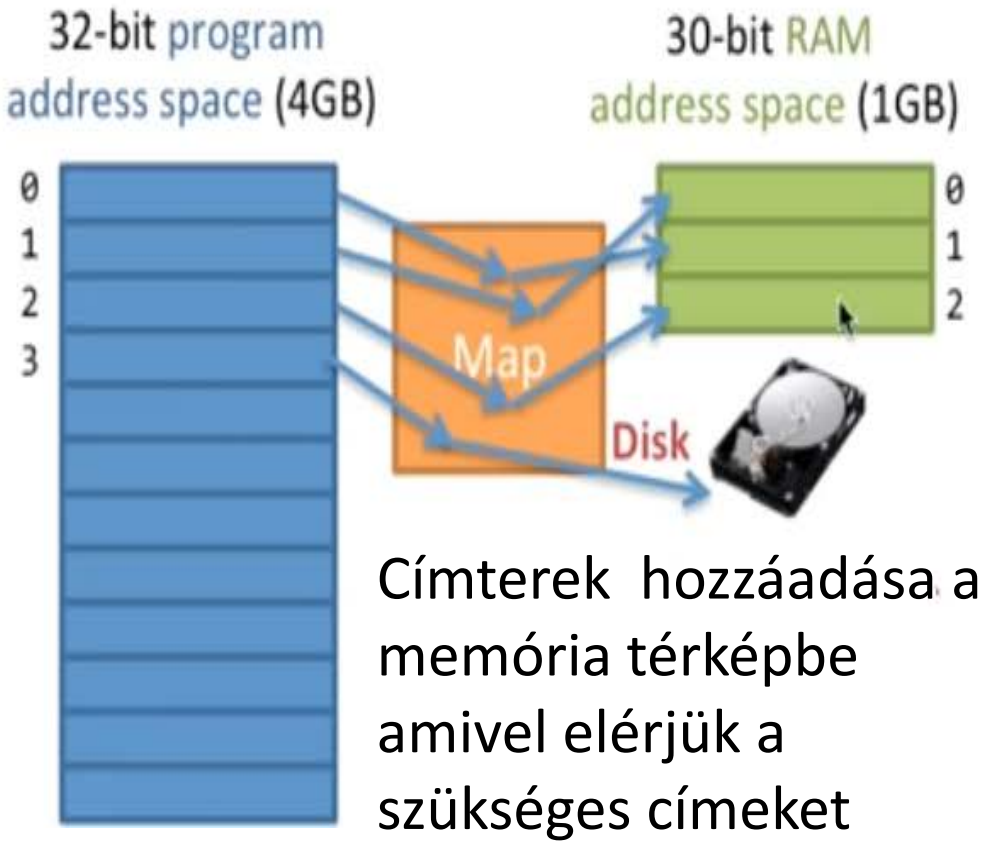
Program cím = RAM cím



Probléma ha több fizikai memóriát akarunk elérni mint amennyi van

Virtuális memóriával

Program cím = címtér a RAM címhez



Címterek hozzáadása a memória térképbe amivel elérjük a szükséges címeket

Virtuálismemória megvalósítás (folyatás)

Megvalósítás:

- Nem egy virtuális címtér van, hanem minden folyamatnak sajátja
- Más folyamatok adatait tartalmazó fizikai memóriabeli területekhez nem tartozik terület a folyamat virtuális memóriájában
- Több folyamat virtuális memóriájába is leképezett fizikai memória területek:
    - Az operációsrendszer kódja és adatai
    - Kommunikációs adatok

Virtuális memória leképezés

Leképezés:

Az összetartozó virtuális és fizikai területek címeit memória térképben tárolni kell

- **bájtonként**: rugalmas, de a táblázat mérete elfoglalná a fizikai memória nagyrészét
- rögzített méretű darabonként: **Lapozás**
- változó méretű darabonként: **Szegmentálás**

Lapozás

Mind a virtuális memóriát, mind a fizikai memóriát egyenlő méretű darabokra osztjuk fel. Tehát a lapozáshoz a fizikai és virtuális memóriaterületek ugyanazon fix méretű blokkokba vannak osztva.

- Fizikai memóriában lapkeretek
- Virtuális memóriában a lapok

Ezeket a fizikai memória rögzített méretű blokkjait lapkereteknek nevezik, és a virtuális memória fix méretű blokkjait lapnak nevezik.

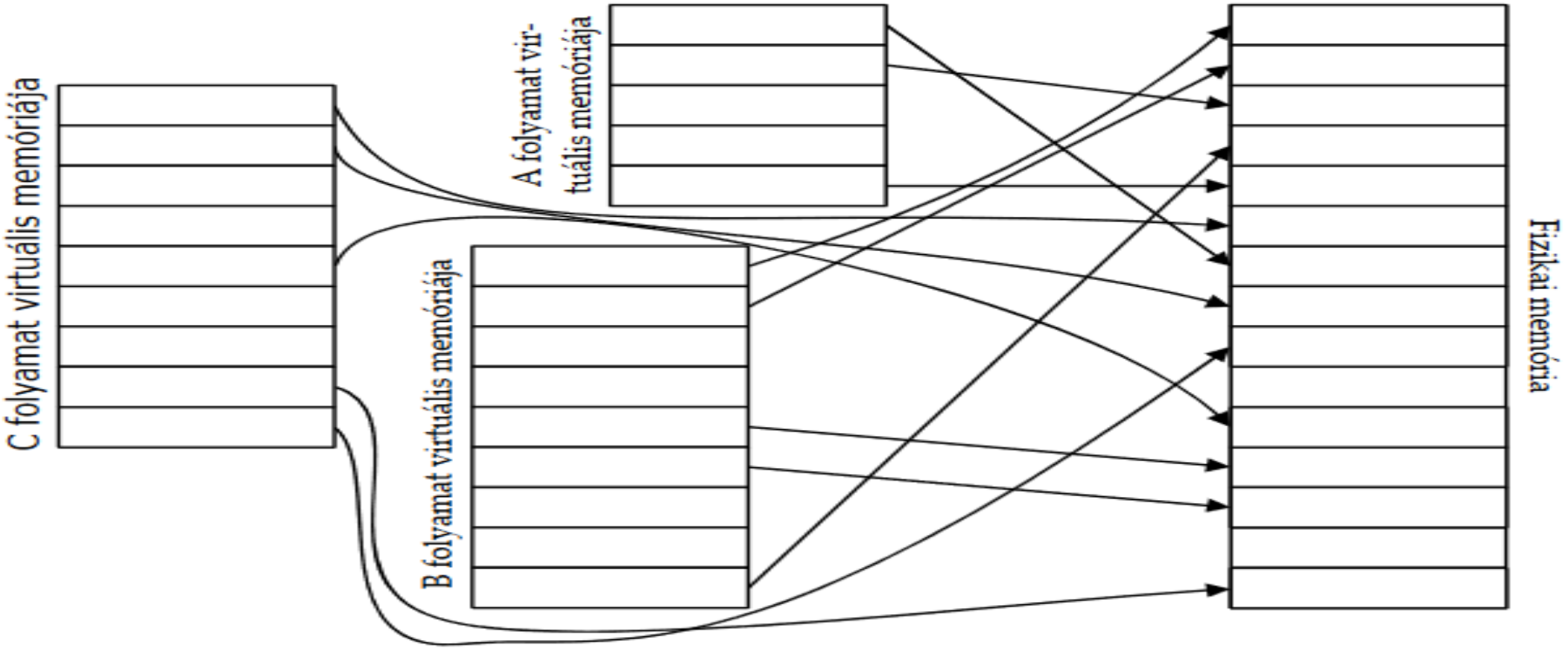
Szemléletes elnevezés: van egy csomó lapunk, de csak akkor tudunk dolgozni velük, ha keretbe tesszük őket;

A keretek száma azonban kisebb, mintalapoké

- Egy lapon belüli címek a fizikai memóriában is egy lapkereten belül lesznek

Lapozás működése

Amikor egy folyamatot végre kell hajtani, a virtuális memóriaterületről származó folyamat lapok betöltésre kerülnek a fizikai memória címkeret lapkeretébe.



Címfordítás

Lapok és lapkeretek egymáshoz rendelését egy táblázat tartalmazza: **laptábla**

- Minden folyamatnak külön laptáblája van
- Laptábla a lapcím szerint van indexelve
- Mezői tartalmazzák, hogy a laphoz tartozik-e lapkeret a fizikai memóriában, és ha igen, mi ott a címe
- Hivatkozás során ez a lapkeret cím a táblázatból alap cím helyére másolódik
- Ha nem tartozik hozzá lapkeret: **laphiba kivétel** történik, és a kivételkezelő feladata a lapot valamelyik lapkeretben elhelyezni

Laphiba kezelése

- Hiányzó lapot be kell tölteni a háttértárról, vagy létre kell hozni egy üres lapot
- Ha van üres lapkeret >> oda betehető a lap
- Ha nincs üres lapkeret >> ki kell dobni egy lapot
  - Kérdés: melyiket dobjuk ki
  - Válasz: sok különböző algoritmus létezik
  - Laphiba kezelése sok erőforrást igényel >> cél, hogy minél kevesebb legyen belőle

  Lapcserélő algoritmusok:

  LRU (least recently used) azt kell kidobni amire a legrégebben hivatkozott

  FIFO azt dobjuk ki, amelyik a legrégebb óta benn van

  NFU(not frequently used) számlálót iktatunk be

Szegmentálás

- A folyamat a változó méretű szegmensekre van osztva, és a virtuális memória címterületre van betöltve.

  - Minden szegmensnek egyedi azonosítója (neve és hossza) van, ami független a fizikai memóriában elfoglalt helyétől

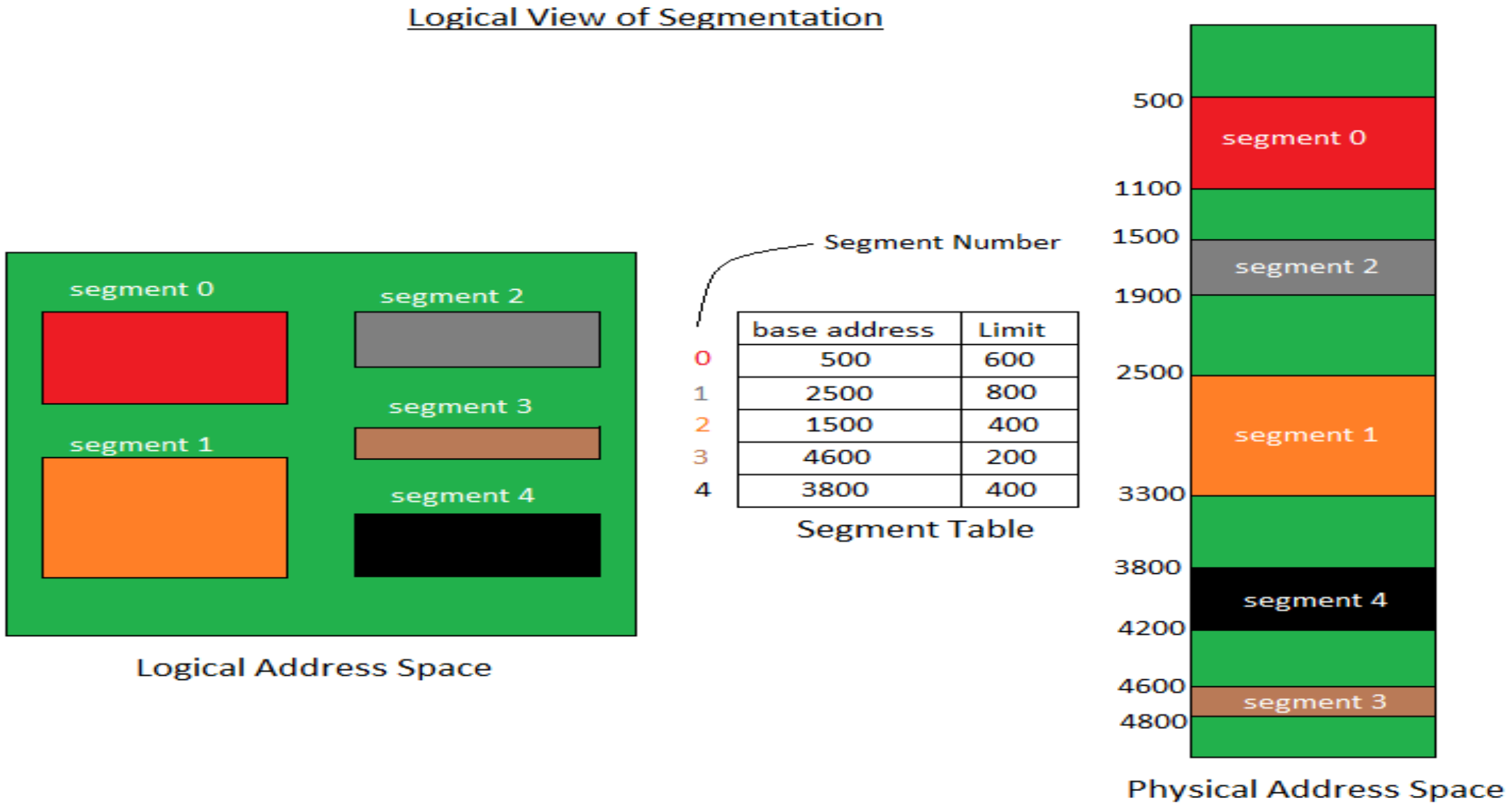
  - Szegmens azonosítók és fizikai memóriabeli kezdőcímek egymáshoz rendelése a szegmenstáblában van

- Szegmentált memória: több lineáris címtérből álló virtuálismemória
- Egy program különböző adatszerkezetei külön szegmensekben helyezhetők el

A program végrehajtáshoz a virtuális memóriaterületből származó szegmensek betöltődnek a fizikai memóriaterületre.

Szegmensek leírói: deszkriptorok

Szegmentálás



Szegmentálás előnyei a lapozáshoz képest

- Rugalmasság: ha egy adatszerkezet mérete változik, nem változik a többi címe
- Szegmentálás jobban illeszkedik a program szerkezetéhez: munkahalmaz jobban definiálható
- Különböző típusúak lehetnek:
    - csak olvasható / írható-olvasható adat,
    - csak végrehajtható/olvasható-végrehajtható kód
- Könnyebben megvalósítható, strukturáltabb adat megosztás folyamatok között

Hátránya: bonyolultabb algoritmus kell a szegmens cseréhez

Hibrid rendszer: Szegmentálás lapozással

- A lapozással történő szegmentálás esetén a virtuális memória lapozásakor általában egy-egy memórialapot kell csak az elsődleges és a másodlagos tár között mozgatni, hasonlóan egy nem szegmentált lapozásos rendszerhez. A szegmenshez tartozó lapok bárhol lehetnek a fő memóriában, és nem szükséges, hogy folytonosan helyezkedjenek el. Ez általában kisebb méretű swappeléssel és csökkent memória töredezettséggel jár

Virtuális memóriához kapcsolódó fogalmak

- **Lapozófájl** (swapfile, pagefile): Az a lemezterület, amely a virtuális memórialapokat tartalmazza. Némely OS esetén külön partíciót képez a lemezen (Linux).
- **Memóriavédelem:** Gyakran kombinálják a virtuális memóriakezelést védelmi rendszerrel, ekkor az egyes lapokat/szegmenseket **olvasható, írható és végrehajtható attribútummal** (illetve ezek kombinációival) látjuk el. Ilyen védelem esetén egy olyan memória-hozzáférés, amely egy adatterületet próbál végrehajtani, vagy egy nem írható területre akar írni, érvénytelen laphibát okoz.
- **Osztott memória (Shared/Common Memory):** Minden korszerű rendszer külön virtuális címtartományba helyezi az egyes folyamatokat (MVS), hogy azok ne zavarhassák egymást, **viszont külön eszközt biztosít arra, hogy a folyamatok, ha akarnak, használhassanak megosztott memóriatartományokat.**