1. **Az asszociatív tárak jellemzői, belső felépítése és alkalmazási területei. (251104\_26:30)**

* Drága technológia: SRAM, ami bit szinten minden cellában tartalmaz egy ***komparátort***
* Angolul: CUM – Content Adressable Memory
* Eddigiekben csak a közvetlen hozzáférésű tárakról volt szó, azaz cím szerint férünk hozzá
* Asszociatív tár az tartalom szerint címezhetőek

**Logikai architektúrája:**

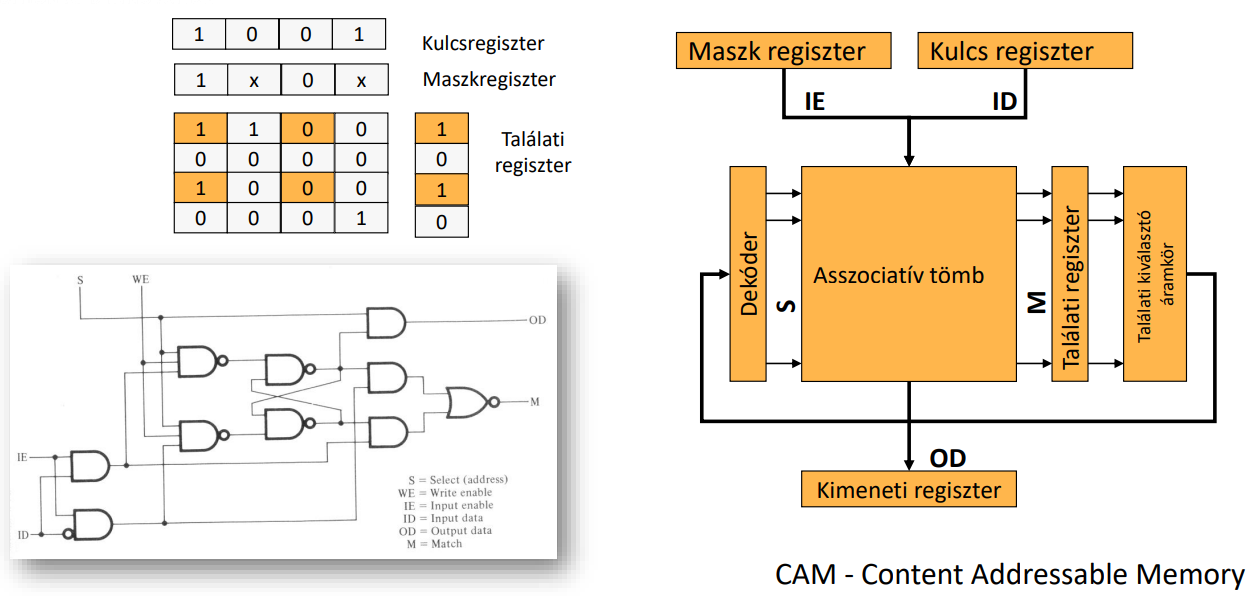
* Kulcs regiszter: ebben van a keresett szó
* Maszk regiszter: kimaszkolhatja azokat a biteket, amiket ki szeretnénk hagyni a keresésből
* Tároló mátrix: tartalmazza a bináris adatokat
* Találati regiszter: megmondja, hogy az adott rekeszben van-e találat

**Működés szempontjából:**

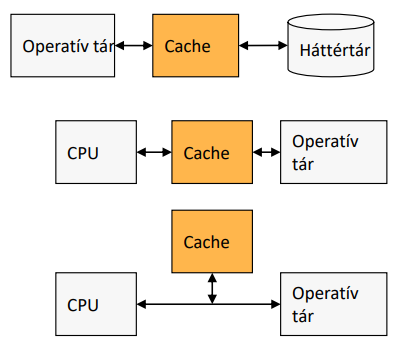
* párhuzamosan összehasonlítja a kulcs regiszter bitjét (ott, ahol nincs maszkolva) a beírt adatokkal, és ha az összes egyezett, akkor tesz egy találatot
* Az asszociatív tömbnek minden egyes cellája tartalmaz egy *komparátort* és egy vízszintesen *kaszkádba* kötött ***ÉS*** kapcsolatot, hiszen az összes nem maszkoltnak meg kell egyeznie
* Tehát a találatnak ***ÉS*** kapcsolata van és az beíródik a találati regiszterbe
* Specifikáció szerint:
* Több találat is lehet
* Ezeket úgy lehet kiolvasni, hogy kódoljuk és dekódolóval utána cím szerint a megfelelő rekeszt ki tudjuk olvasni a kimeneti regiszterbe

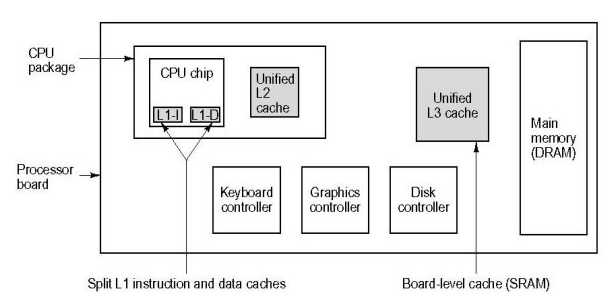
**Mire is lehet használni:**

* Számítástechnika korai szakaszában: használták Hash tárnak
* Számítástechnika elején használták vezérlésekre (különösen hadiiparban) – mivel akkor nem voltak még olyan mikrokontrollerek, amik helyettesíteni tudták volna
* Mai világban pl. a gyorsító tárakban, amikor szeretnénk gyorsan keresni
* Virtuális tárkezeléseknél

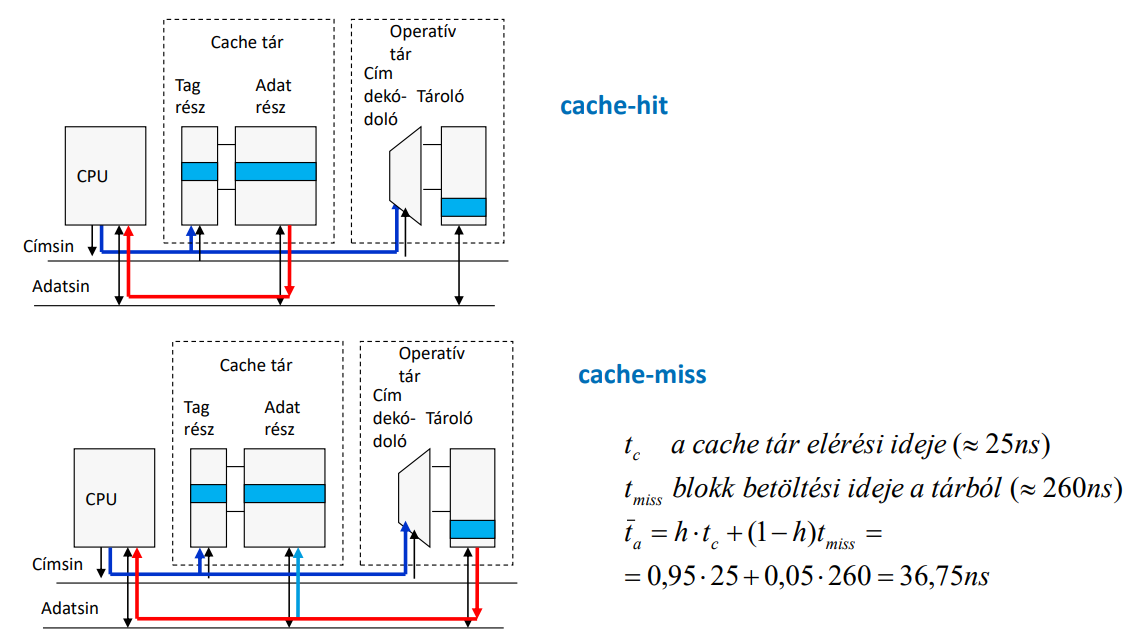


1. **A gyorsítótárak fogalma, elve, értékelése: cache-hit és cache-miss. Gyorsítótárak osztályozása elhelyezkedés és tárolt információ szerint. Gyorsítótárak alapvető szervezési módjai. Teljesen asszociatív, közvetlen leképezésű és csoportasszociatív gyorsítótár. Gyorsítótárak betöltési és aktualizálási algoritmusai. A gyorsítótárak főbb paraméterei és fejlődésük. Többszintű gyorsítótárak. Cache koherencia fogalma. Gyorsítótárak helyettesítési algoritmusai. (251104\_35:40)**

* Gyorsítótár (cache) – utasítások és adatok átmeneti tárolására szolgáló, gyors működésű, a felhasználó számára nem elérhető tár.
* Cél az adatforgalom gyorsítása
  + Operatív tár és háttértár között
  + CPU és operatív tár között
    - Leválasztó (look-through)
    - Mellérendelt (look-aside)
* Elhelyezkedés szerint
  + processzor magon belül (Privat)
  + processzormagon kívül (Shared)
* Tárolt információ szerint
  + csak adat
  + csak utasítások
  + adat és utasítások együtt
* Elve:
  + Nem byte-os a mozgatású, hanem blokkos, ezt cache sornak hívjuk
  + Lehelyezünk egy köztes tárat, amin mindkét irányba párhuzamosan történik az adatátadás
* Működése teljesen hardware-es, nem programozható, hanem a hardware vezérli
* Automatikus mechanizmus: pl. a CPU ugyanúgy címzi, mint az operatív tárat és nem tudja, hogy a gyorsító tárhoz fog kerülni az adat
* cache-hit: a processzor által kiadott címtartomány benne van a gyorsítótárban
* cache-miss: a processzor által kiadott címtartomány nincsen benne a gyorsító tárban
* akkorára kell szabni a cache kapacitását és a cache sorokat, hogy a cache-miss 10% alatt legyen
* cache koherencia: a cache tárt és az operatív tárt egymásnak megfeleltetett, azaz megegyezik benne az adat, azon a címtartományon:
  + ha nem egyezik meg benne az adat azt úgy hívják cache koherencia probléma 🡪
  + valamilyen cache koherencia algoritmussal fel kell tartani, mivel az operatív tárat más is használhatja
* **Többszintű gyorsítótárak:**
  + **1.szintű gyorsítótár (L1-I, L1-D):** benne van a magban az L1-I és az L1-D, ezt úgy hívják, hogy Harvard-architektúra, mivel az utasítás és az adat párhuzamosan lehívható. Méretük 16-64KB
  + **2.szintű gyorsítótár (L2):** általában egyesített, mivel tartalmazza az adatokat és az utasításokat is. Általában a lapkán vagy a processzor magban található. mérete általában 512KB-1MB
  + **3. szintű gyorsítótár (L3):** a processzor lapkán vagy a chip set-ben található és néhány MB SRAM-ot tartalmaz
  + A legnagyobb blokkot az L3**-**ba helyezik, annak egy részét behelyezik az L2-be, annak egy részét berakjak az L1-be

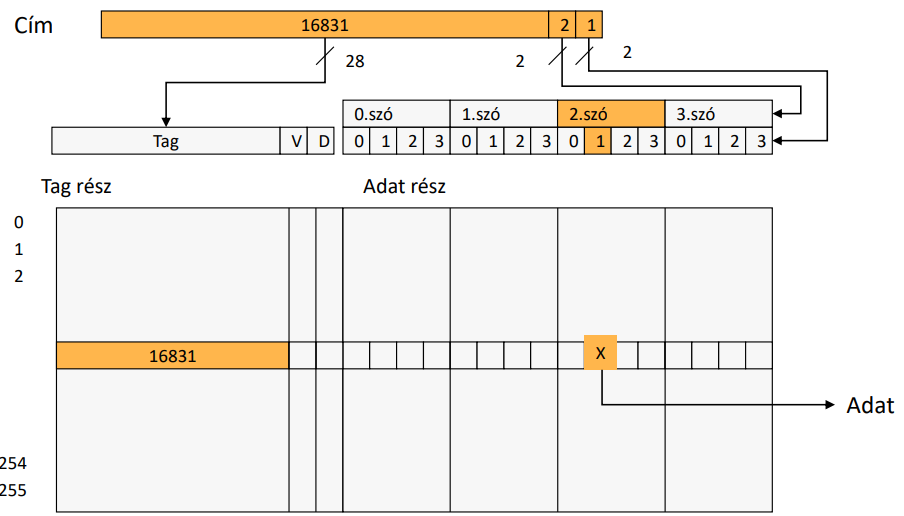


* L1-I(utasítás), L1-D(adat), L2 és L3 vegyesen tartalmaz adatot és utasítást

****

* A gyorsítótár két részre oszlik: tag rész (azonosító), adat rész
* CPU kitesz egy címet vagy byte-osan vagy szavasan, de az átvitel blokkosan történik
* Tag definíció: Tagnek hívjuk a címnek azt a szegmensét, amellyel a cache sor azonosítható
* helyettesítési algoritmus (replacement policy) – új blokk betöltésekor a kicserélhető blokk meghatározásának algoritmusa
  + előbb-utóbb betelik a gyorsítótár, olyan nincsen a gyorsítótárban, hogy törlünk (olyat lehet, hogy frissítjük a tárat). Amikor kell hely, egy helyettesítési algoritmussal meghatározzuk, hogy melyik az a cache sor, amit felül fogunk írni
* Gyorsítótár jellemzői:
  + cache kapacitás – az elhelyezkedéstől és típustól függően általában 8kB – 2MB
  + blokk méret – az operatív tár és a cache tár között egy egységben mozgatott adatmennyiség 32 byte – 1 KB (utasításoknál nagyobb, adatok esetében kisebb)
  + sorméret – egy összehasonlítással kijelölhető adatmennyiség (általában a blokkal megegyező vagy annál kisebb)
  + adat-aktualizálási 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
* **Koherencia:**

**Teljesen asszociatív gyorsítótár (fully associative)**

****

Tag része tartalom szerint címezhető

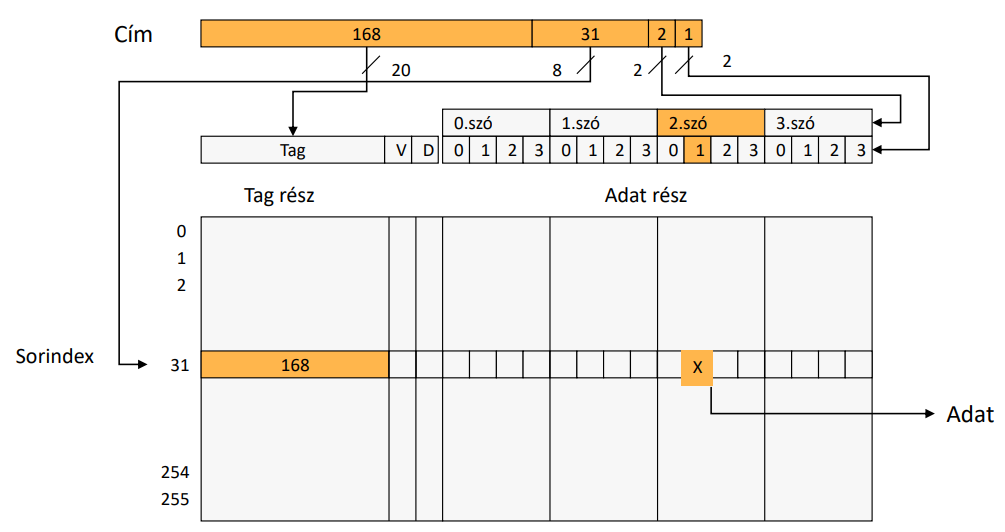
A tag mellett minden sorhoz jár 2db állapot bit:

**valid**: van az adott soron érvényes betöltött adat

**dirty**: a cache soron változtatás történt 🡪 ha bármelyik byte-ot megváltoztattuk, akkor a dirty 1-es lesz

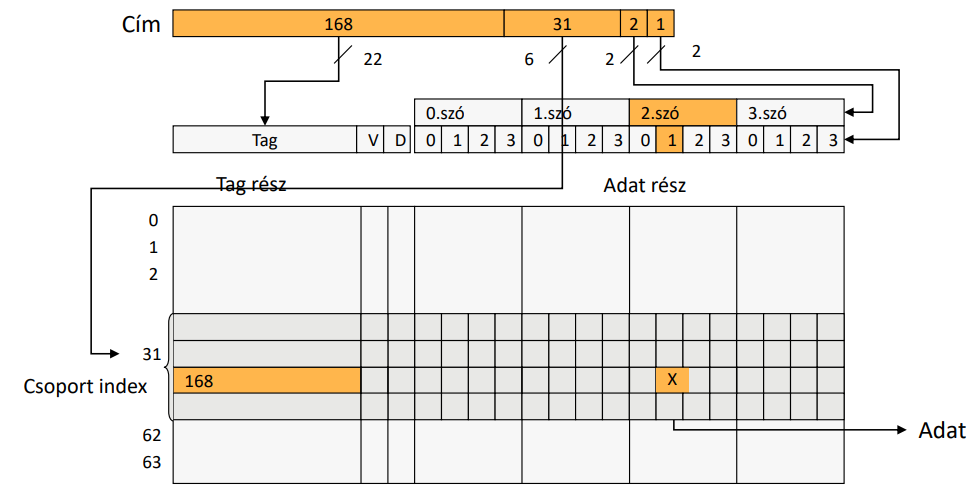
A teljesen asszociatív tár a legdrágább, ezért a legkisebb kapacitású gyorsítótárnak (általában az L1) szokták használni

**Közvetlen leképzésű gyorsítótár (direct mapped)**

****

L3-ban van általában

**Csoport asszociatív gyorsító tár**



1. **Virtuális tárkezelés fogalma, módszerei. A virtuális cím kiszámítási módja. A lapozás és a szegmentálás összehasonlítása. MMU feladatai. Lapozás jellemzői. Lapbetöltés, helyettesítési eljárások. Lapcím kiszámítási eljárások: egylépcsős, kétlépcsős és háromlépcsős. Szegmentálás jellemzői, megvalósítása. Szegmenshelyettesítési eljárások. Szegmenscím kiszámítási eljárások: egylépcsős szegmens regiszter felhasználásával, kétlépcsős és háromlépcsős szegmentált lapcím kiszámítás. (251104\_1-0:0)**

* 3 címtér létezik:
  + Fizikai címtér:
  + Logikai címtér: amit a CPU megtud címezni
  + Virtuális címtér: megegyezik a logikai címtérrel (IBM találja fel)

Virtuális tárkezelés fogalma: A virtuális tárkezelés alapvető célja az operatív tár véges kapacitásából adódó korlátok leküzdése. Mivel a programok mérete gyorsabban nő, mint a fizikai memória, gazdaságosabb a nagy mennyiségű adatot olcsóbb háttértáron tárolni.

A virtuális tárkezelés során megkülönböztetünk logikai (virtuális) címteret, amelyet a programozó lát, és fizikai címtartományt, amely a ténylegesen rendelkezésre álló memóriát jelenti.

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 elaprózóidik. A tár rugalmasabban kezelhető a lapozási technika bevezetésével.

A virtuális cím kiszámításának módja:

A lapozás és szegmentálás összehasonlítása:

Lapozás:

-A hardware fogja intézni.

-Egy bináris címtartományt hoz létre

-Azonos méretűek (512 byte – 64 KB)

-Nagy memóriát szimulál

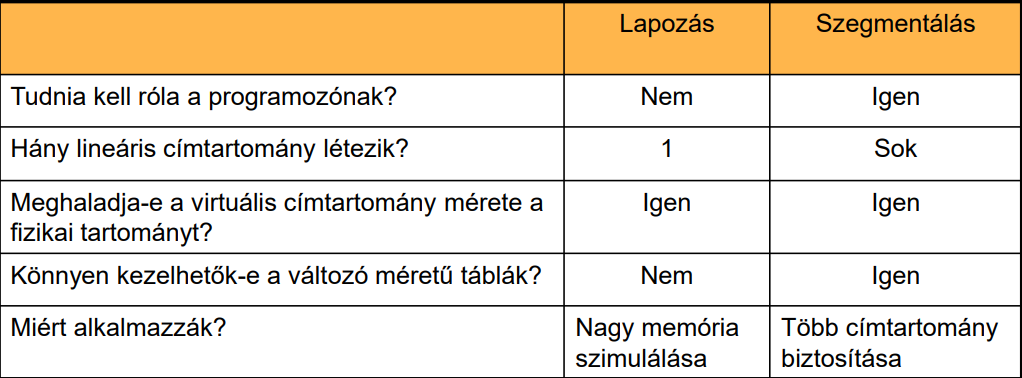
Szegmentálás:

-Programozó kezelheti.

-Szegmensenként saját címtartománnyal rendelkezik

-Változó méretűek

-Adatmegosztás és logikai tagolás



MMU feladatai:

* Címleképezés: A virtuális címek átalakítása valós fizikai címekké táblázatok segítségével.
* Tárvédelem: A memóriaterületek védelme, a címzések helyességének és a hozzáférési jogoknak (írás, olvasás, végrehajtás) az ellenőrzése.
* Izoláció: A felhasználói feladatok elkülönítése egymástól és a rendszerprogramok védelme.

Lapozás jellemzői:

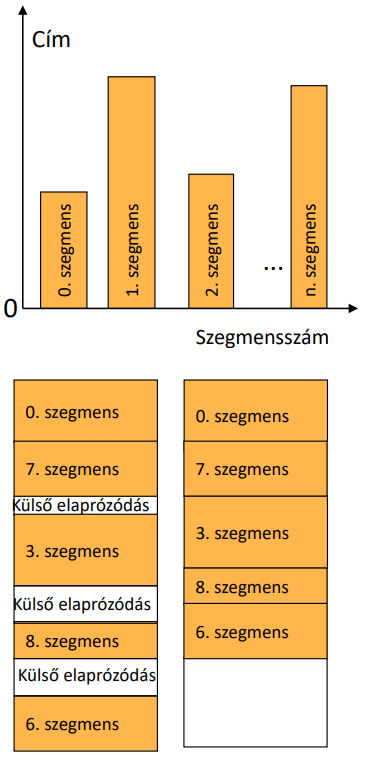
A virtuális címteret fix méretű lapokra osztják, amelyek csak meghatározott lapkeretekbe kerülhetnek a RAM-ban.

* **Lapbetöltés (Demand fetching):** Csak igény esetén történik; ha az adat hiányzik, **laphiba (page fault)** keletkezik, ami elindítja a betöltési eljárást.
* **Helyettesítési eljárások:** Ha nincs üres lapkeret, választani kell egyet a felszabadításra. Módszerei: véletlenszerű, FIFO (legrégebben betöltött), vagy **LRU** (legkevésbé használt).
* **Vergődés (Thrashing):** Akkor fordul elő, ha a program munkahalmaza nagyobb, mint a rendelkezésre álló lapkeretek száma, így a rendszer folyamatosan laphibákat generál.

Lapcím kiszámítási eljárások:

A címfordítást gyorsítja a **TLB (Translation Lookaside Buffer)**, amely a leggyakrabban használt lapok adatait tartalmazza.

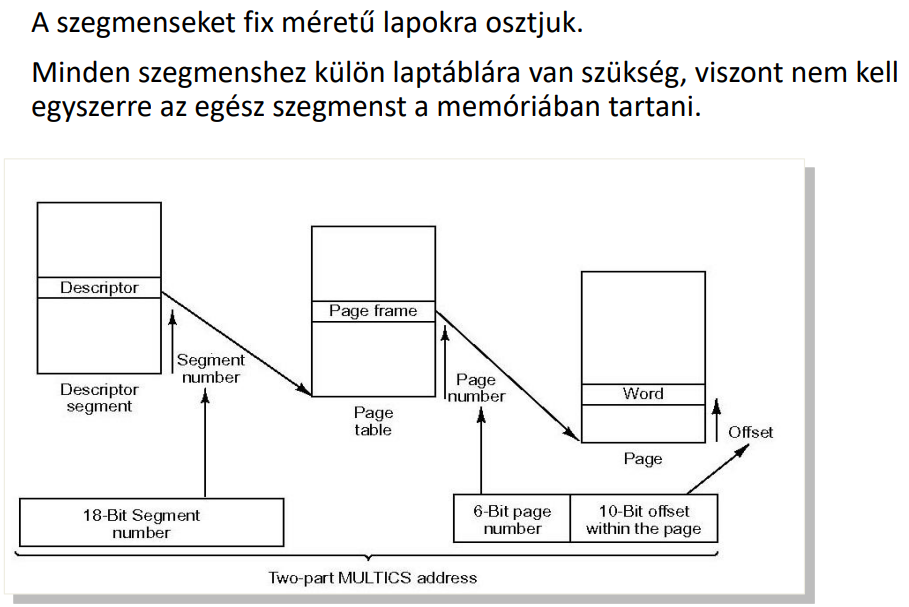
1. **Egylépcsős lapcím számítás:** A virtuális cím egy lapsorszámból és egy relatív címből áll. A lapsorszám alapján a laptáblából kikeresik a lapkeret sorszámát, majd ezt összefűzik a relatív címmel.
2. **Kétlépcsős szegmentált lapcím számítás:** Először a szegmens tábla alapján meghatározzák a szegmenst, majd azon belül a laptábla segítségével a konkrét lapkeretet.
3. **Háromlépcsős eljárás:** A bonyolultabb rendszereknél egy köztes **lap katalógus tábla** is beépül a szegmens tábla és a laptáblázatok közé a cím kiszámításakor.

Szegmentálás jellemzői, megvalósítása:

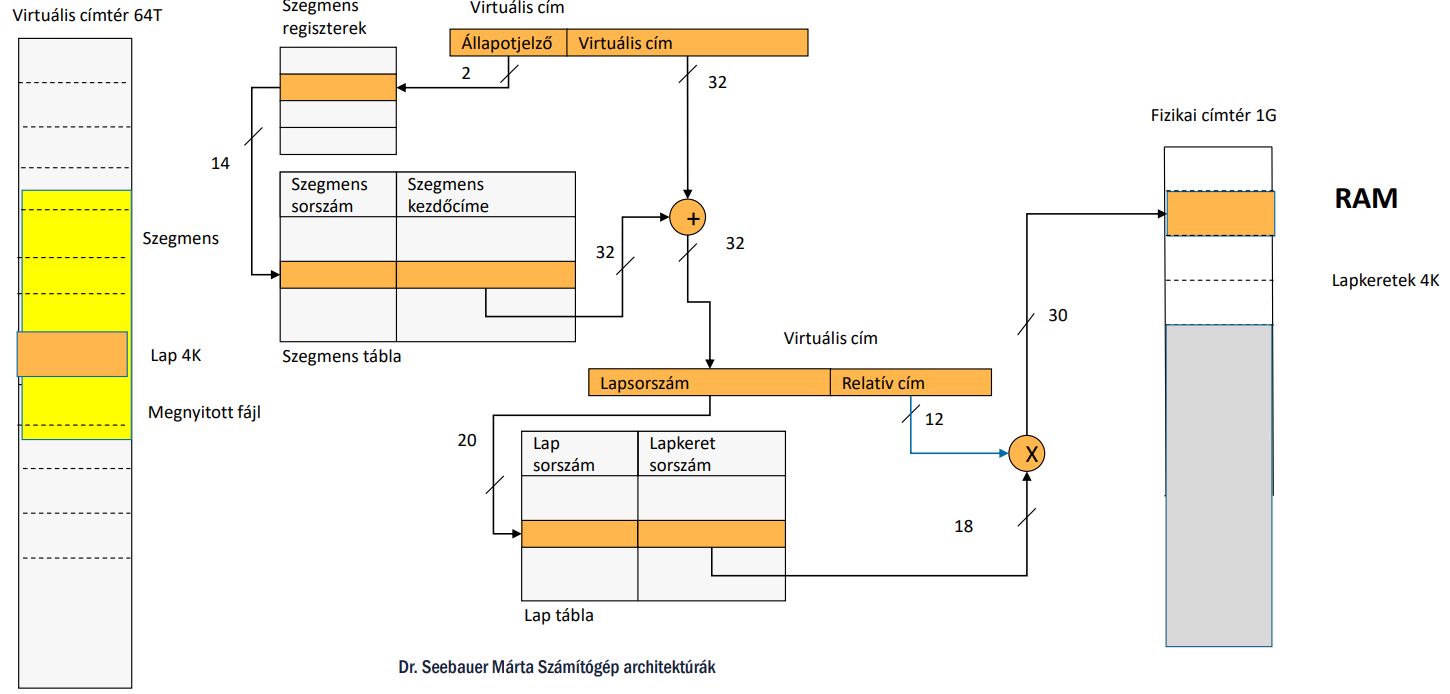
A szegmens egy logikai egység (pl. eljárás, tömb), amelynek hossza a végrehajtás során változhat. Megvalósítható **cseréléssel (swapping)** vagy **lapozással**.

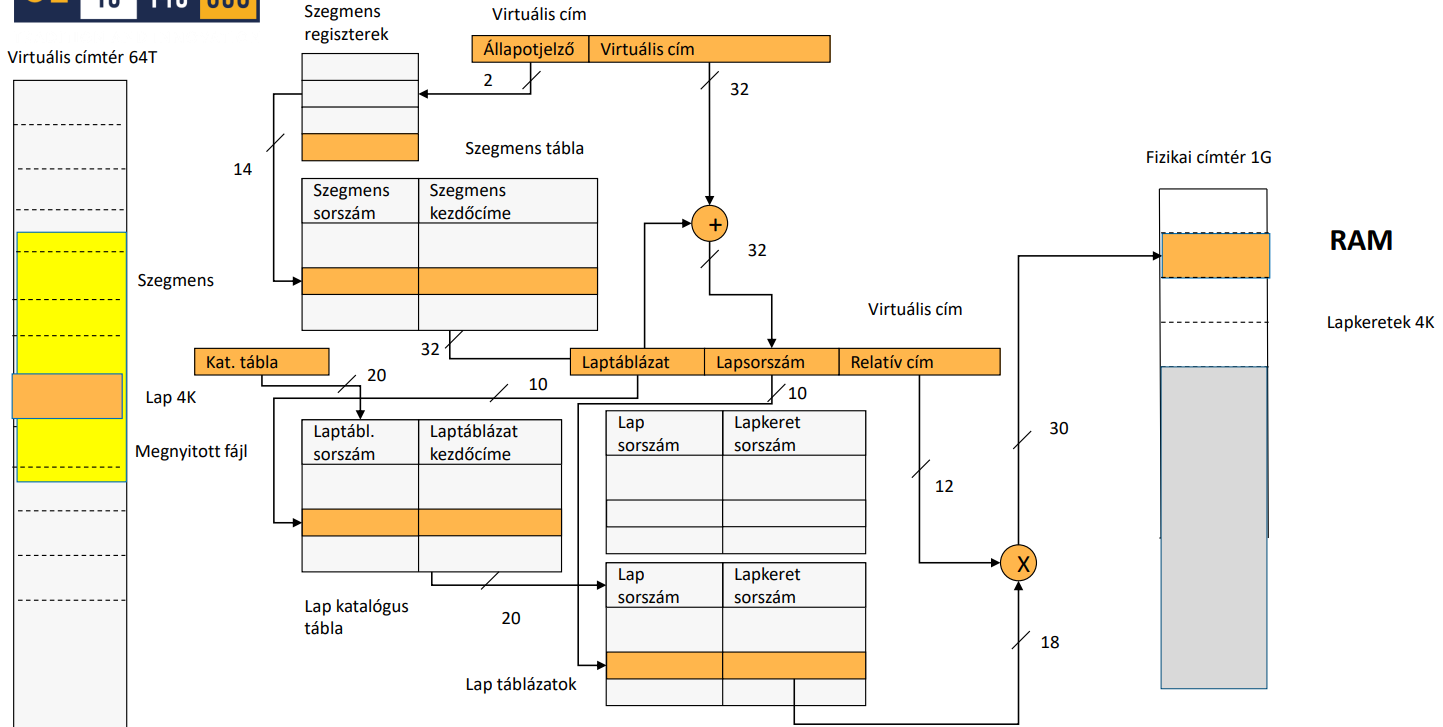
* **Szegmenshelyettesítési eljárások:**
  + **Első szabad hely (first fit):** Az első megfelelő üres helyre teszi.
  + **Következő szabad hely (next fit):** Az előző utáni első szabad helyet választja.
  + **Legjobb illesztés (best fit):** A legkisebb még elegendő helyet keresi meg.
  + **Legrosszabb illesztés (worst fit):** A legnagyobb szabad területet választja.

Szegmenscím kiszámítási eljárások:



Kétlépcsős szegmentált lapcím kiszámítása



Háromlépcsős szegmentált lapcím kiszámítása

**4. Tárvédelmi módszerek. Programok és adatok védelme. A hierarchikus védelmi szerkezet.**

**Tárvédelmi módszerek és az MMU szerepe**

A tárvédelem elsődlegesen a memóriakezelő egység (MMU) feladata. Az MMU felelős a következőkért:

* A címzések helyességének ellenőrzése és a memóriaterületek védelme.
* A rendszerprogramok megóvása a felhasználói beavatkozásoktól.
* A különböző felhasználói taszkok (folyamatok) egymástól való elkülönítése.
* A tárolt adatokhoz való hozzáférések szabályozása.

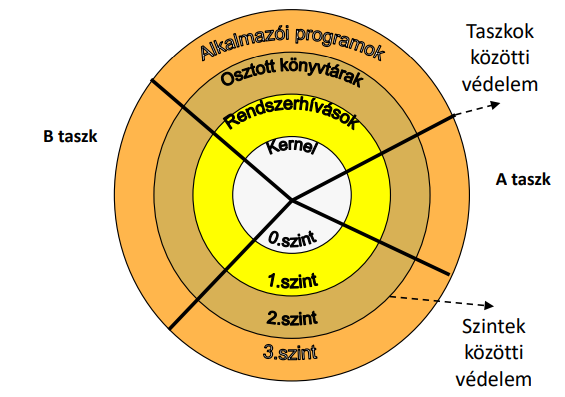
A védelem megvalósítása történhet hierarchikus (ring protect system) vagy nem hierarchikus (capability based protect system) módon. Ez utóbbinál minden taszkhoz egy táblát rendelnek, amely meghatározza az engedélyezett műveleteket, de ennek nincs hardveres megvalósítása, csak az operációs rendszer működteti.

**Programok és adatok védelme**

Az adatok védelmét az MMU a szegmensekhez és lapokhoz rendelt **hozzáférési jogok** szabályozásával biztosítja:

* **Olvasási jog (read access):** Az adat kiolvashatóságát teszi lehetővé.
* **Írási jog (write access):** Az adat módosítását engedélyezi.
* **Végrehajtási jog (execute access):** Kizárólag programkódot tartalmazó szegmensekhez vagy lapokhoz rendelhető.

**A hierarchikus védelmi szerkezetek (Ring Protection)**

Ez a rendszer különböző védelmi szinteket (gyűrűket) határoz meg, ahol a jogosultságok felülről lefelé szűkülnek.

* **Védelmi szintek:** A legmagasabb védelmi szinttel (0. szint) az operációs rendszer rendelkezik, míg a legalacsonyabbal a felhasználói programok.
* **Hívási szabályok:** A programok csak a saját szintjükön vagy magasabb védelmi szinten lévő rutinokat hívhatnak meg közvetlenül.
* **Védelmi kapuk:** Az alacsonyabb (szigorúbb) szintek és más, azonos szinten futó taszkok elérése csak speciális **védelmi kapukon** keresztül lehetséges.
* **Adathozzáférés:** A programok csak a saját vagy náluk alacsonyabb védelmi szinten lévő adatokat érhetik el.
* **Elkülönítés:** A szintek közötti védelmet a szintek saját veremtárolója, a feladatok közötti védelmet pedig az egyedi leíró (descriptor) táblák biztosítják.
* **Hardveres támogatás:** A hierarchikus rendszer nagy előnye a hardveres megvalósítás, ami gyors működést tesz lehetővé; ilyen rendszert alkalmaznak például az **Intel processzorok** is.

**5. I/O eszközök kezelési módjai. Programozott I/O fogalma, memóriában leképezett és különálló I/O címtér. I/O csatorna, I/O processzor. (251111\_0:00)**

**I/O eszközök kezelési módjai**

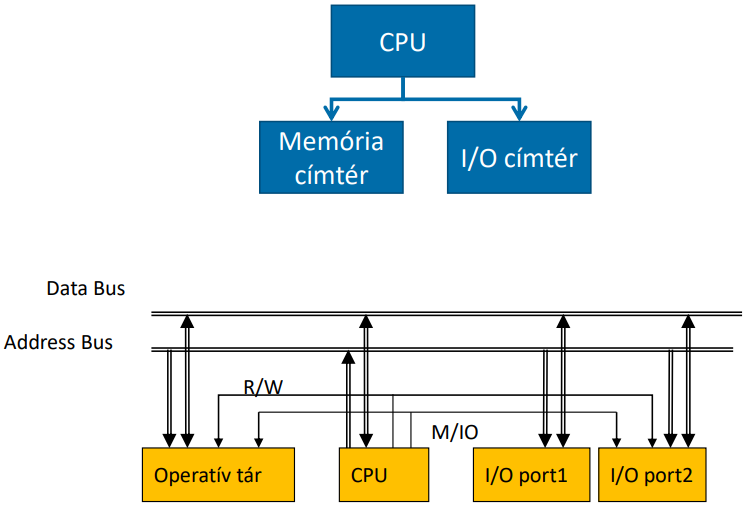
* **feltétel nélküli, közvetlen (direkt) átvitel**, egyszerű eszközök esetében, amikor az adatátvitelt minden feltétel nélkül végre lehet hajtani, ellenőrzésre sem előtte, sem utána nincs szükség. Pl. LED beállítása.
* **feltételes átvitel**, az átvitel lehetősége valamely feltétel teljesülésétől függ. Pl. az I/O eszköz foglaltsága.
* **közvetlen tároló-hozzáférés (DMA – Direct Memory Access),** az adatátvitel az I/O eszköz és a memória között a processzor kezdeményezésére, de attól függetlenül a DMA vezérlő irányításával történik. Jellemzően adatblokkok átvitelére használják.
* önálló, programozott vezérlővel rendelkező **adatátviteli csatorna**, a csatorna a saját vezérlőjének az irányítása alatt, a processzortól függetlenül végzi az adatátvitelt a periféria és a memória között. Nagygépekre jellemző.
  + **szelektor** csatorna – egyetlen gyors periféria kiszolgálása
  + **multiplexor** csatorna - több lassú periféria kiszolgálása időosztásos üzemmódban
* **I/O processzor** (IOP) és csatorna alkalmazása, a csatorna továbbfejlesztése. Az IOP általában a perifériához csatlakozik, a CPU-tól függetlenül működik, saját utasításkészlettel rendelkezik, és saját sínt vezérel (I/O, PCI, AGP sín)

**Programozott I/O fogalma**

Az összes I/O utasítás a CPU közvetlen vezérlése alatt kerül végrehajtásra, azaz minden, az I/O egységet bevonó adat-átviteli művelet a CPU egy bizonyos utasítására történik.

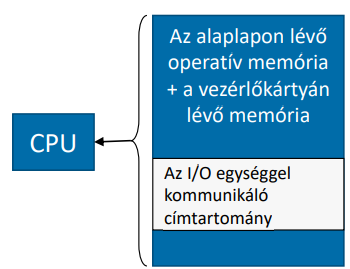
* **eszközszintű periféria-kezelés** - a perifériális eszközök fizikai sajátosságainak megfelelő illesztési felületet és utasításkészletet biztosítunk. A kis szóhosszúságú mikroprocesszorokra jellemző.
* **logikai szintű eszközkezelés**, ahol általánosított illesztési felületeket biztosítunk a periféria-kezelés számára. A Pentium-os környezetben az utóbbi használata kizárólagos.

**A különálló I/O címtér**

****A programozott I/O-val rendelkező rendszerekben a CPU, a memória és az I/O egységek egy közös használatú sínen (a rendszersínen) keresztül kommunikálnak egymással.

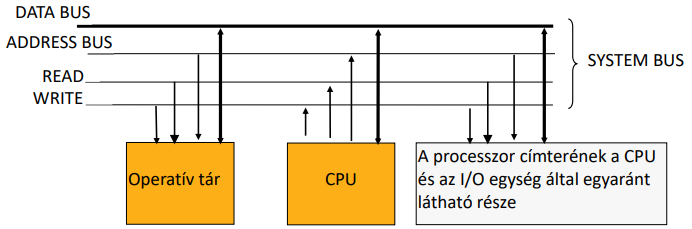
Az a címsín, melyet CPU a memóriahozzáférésre használ, szolgálhat az I/O egység kiválasztására is. A két cím megkülönböztetésére létezik egy memória - MREQ és egy I/O - IORREQ nevű vezérlővonal, mely jelzi, hogy a címsínen az adott időpillanatban memória vagy I/O port cím található-e.

Az I/O cím rendszerint rövidebb, mint a Operatív tár memória cím..

**Memória címterében leképezett I/O**

A memóriában leképezett I/O lényege, a megosztás, azaz, hogy a memória mapped addressing) - címtérnek egy közös részét használhatja mind a processzor, mind pedig a periféria.

Annak érdekében, hogy a processzor adatot küldjön az adott perifériához, elegendő az, hogy az adott perifériával közösen használt memória-területre beírja egy közönséges STORE művelettel, akár egy későbbi felhasználásra szánt adatot. A periféria aztán ugyanezt a memória-területet el tudja olvasni, és így hozzájut a számára szükséges adathoz.

A memóriában leképezett címzésű egységeknek ezért közvetlenül el kell érniük a memóriasínt. Ezen kapcsolat révén akkora sebességre tesznek szert, amekkorát csak a memória-rendszer és a síncsatlakozása biztosít.

A legtipikusabb memória-címtérben leképezett egység a PC képernyője. A legtöbb grafikus rendszer megengedi a mikroprocesszor számára, hogy közvetlenül megcímezze azt a frame buffert, amely a képernyőn az adott pillanatban megjelenő képet tartalmazza.

**I/O processzor**

A periféria eszközök sokfélesége miatt a számítógépek beviteli/kiviteli rendszere nem kötődik meghatározott eszközök használatához, hanem általánosított beviteli/kiviteli eljárásokat és illesztési felületeket biztosítanak számukra. A központi egység és a perifériák nagy sebességkülönbsége miatt általában nem használnak közvetlen processzorirányítást.

A legegyszerűbb mikroprocesszoros rendszerektől eltekintve a beviteli/kiviteli hardver részleteit az **operációs rendszer** jórészt eltakarja.

A **csatornára** és az **I/O processzorra** alapozott perifériakezelés lényege, hogy rögzített feladatú modulok hajtják végre, így a processzort felszabadítják a periféria részletes kezelése alól, melyet az csak megfelelő szoftver segítségével tudna végrehajtani. Ezek tehát **intelligens, programozható eszközök**, melyek a processzortól egy parancssorozatot véve a továbbiakban autonóm módon működve hajtják végre a perifériakezelést.

Az I/O processzorok az átvitel szervezésén túlmenően meglehetősen általános adatfeldolgozási képességekkel is rendelkeznek. Ez különbözteti meg őket a csatornáktól, melyek elsősorban a bevitel/kiviteli műveletek autonóm irányítására szolgálnak. Tehát a csatorna felfogható egyszerűsített I/O processzorként is.

A beviteli/kiviteli folyamatok kezelésére háromféle utasítástípus szolgál:

* vezérlőutasítások;
* periféria-állapot lekérdező utasítások;
* adatátviteli utasítások.

**6. Közvetlen memória-hozzáférés (DMA) fogalma, működése, blokkos és cikluslopásos átvitel.**

**A közvetlen tárhozzáférés DMA – Direct Memory Access fogalma**

A hardver-komplexitás egy igen szerény növelésével elérhető, hogy az I/O egység a CPU közreműködése nélkül legyen képes adatblokk átvitelére a memória felé, illetve a memóriától. Ez azt igényli, hogy az I/O egység, vagy pedig a vezérlője

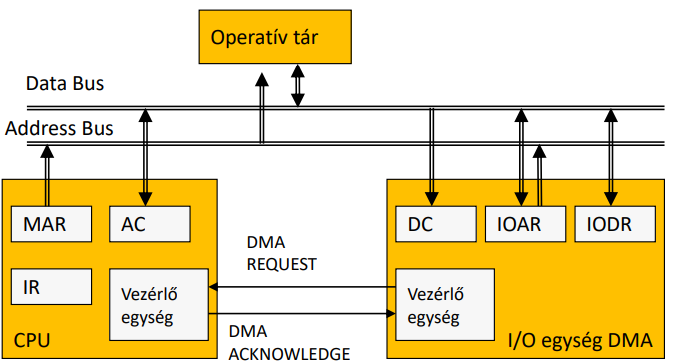
* legyen képes memória-címek generálására
* adatátvitelre a memória-sín felé és síntől
* sín-igénylési és kiválasztási mechanizmussal kell rendelkeznie.

Továbbra is a CPU a felelős minden egyes blokk-átvitel kezdeményezéséért. Az I/O egység pedig a CPU további program-végrehajtása nélkül gondozza az átviteli folyamatot. Ezt a folyamatot hívják közvetlen tárhozzáférésnek vagy DMA-nak.

A DMA-t célszerű alkalmazni

* a nagyobb sebességű eszközök használatakor
* nagyobb tömegű adat átvitele esetén.

A DMA átvitel előnye, hogy a szükséges megszakítások száma nagyságrendekkel csökken, így a processzor ideje jelentős mértékben felszabadul.

**DMA működése**

A I/O egység részei

* IODR adatregiszter
* IOAR címregiszter tárolja a következő átviendő szó címét. Ez automatikusan inkrementálódik minden szó átvitele után.
* DC (data counter) adatszámláló. regiszter tárolja a még átviendő szavak számát. Minden átvitel után automatikusan dekrementálódik és nullára tesztelődik. Amint eléri a nullát, az I/O egység leáll

Az adott regiszterek lehetővé teszik az I/O egység számára az adatátvitelt a memória egy összefüggő területére.

Az I/O egységet általában felszerelik egy megszakítási képességgel is, ennek révén egy megszakítást küld a CPU felé, hogy jelezze az adatátvitel végét.

A DMA vezérléséhez szükséges logikát könnyen el lehet helyezni egy IC-ben, amelyet DMA vezérlő IC-nek hívnak. Ez több I/O egység számára képes a DMA adatátvitel felügyeletére, melyek mindegyike különféle prioritással érheti el a memóriasínt

**Blokkos átvitel (burst cyrcle mode)**

A DMA képességgel rendelkező I/O egység átviheti szavak egy hosszú blokkját egyetlen folyamatos műveletként, ezt hívják blokkos átvitelnek. A közvetlen tárhozfordulásos átvitelnél a processzor által elindított DMA vezérlő - a processzor kihagyásával - önállóan irányítja az adatátvitelt a tároló és a kijelölt I/O eszköz között.

A processzor és a DMA vezérlő közötti kapcsolat a megszakítási vonalak segítségével jön létre. A DMA segítségével bonyolított adatátvitel a következő lépésekkel írható le:

Processzor oldal

* a processzor megvizsgálja az I/O eszköz állapotjelzőjét, hogy fogadni tudja-e az átviteli kérelmet. Ha nem tudja fogadni, akkor egy hibavizsgáló rutin elindításával a processzor megpróbálja megállapítani a hiba okát;
* a processzor kiadja az átviteli utasítást és a szükséges paramétereket (az átviendő blokk kezdőcímét, hosszát, az átvitel irányát) az előírt módon előkészíti a DMA vezérlő számára;
* elindítja a DMA vezérlőt.

**DMA oldal**

* a sín előkészítése az adatátvitelre;
* DMA sínkérelem (BUS REQUEST) jelzése;
* a sínciklus befejezése;
* a buszvezérlő engedélyezi a sín használatát (BUS ACKNOWLEDGE);
* a cím sínre tétele;
* az adat sínre tétele;
* a számláló csökkentése; ha nem nulla, akkor további adatok címzése, átvitele;
* ha a számláló = 0, akkor a sínfoglaltság megszüntetése;
* megszakítási kérelem a processzor felé, annak jelzésére, hogy az átvitel véget ért.

**Processzor oldal**

* a processzor ellenőrzi a DMA vezérlőt az átvitel végrehajtásának sikerességéről;
* a sínhasználat befejezése.

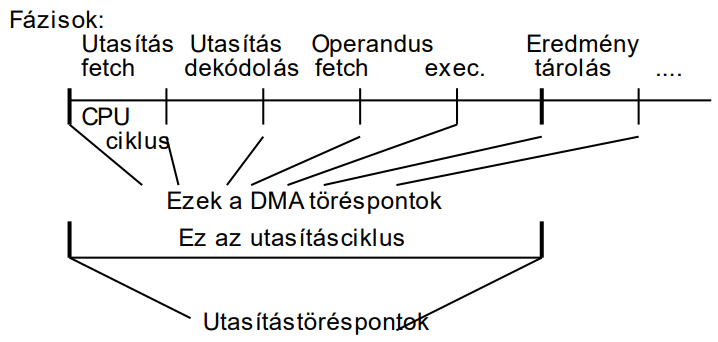
**Cikluslopás (cycle stealing)**

Ha az átviendő adatok nem blokkos formájúak, akkor csak egy-egy adat átvitelére kell igénybe venni a sínt. Ezt az eljárást nevezik cikluslopásnak, amely tulajdonképpen a sín időosztásos használata a processzorral közösen.

A DMA átvitelhez használt ciklusok átlapolhatnak a CPU-ciklusokkal. Mivel nem minden CPU-ciklus igényli a memóriasín használatát, a CPU képes lehet az utasítás végrehajtás folytatására, miután lemondott a memóriasín vezérléséről. Így lehetségessé válik a CPU műveletek és a DMA átvitel átlapolására.

A valamelyik I/O egység által bejelentett DMA megszakítási igény a CPU-tól csak a memóriasín vezérlésének a megszakító egységhez történő átadását jelenti. A CPU átadhatja a vezérlést minden olyan művelet végén, amely ezt a sínt használja. Egy utasításciklust egy sor CPU vagy gépi ciklusra oszthatunk, amelyek közül több igényelheti a memóriasín használatát. Általánosan használt technika, hogy minden CPU ciklus végén lehetővé kell tenni a gépnek, hogy válaszoljon a DMA igényre.

Amennyiben ilyen megszakítás érkezik a CPU-hoz, vár a következő töréspontig, felszabadítja a memóriasín vezérlését és az igénylő I/O egység felé DMA\_ACKNOWLEDGE vezérlővonal aktiválásával jelzést küld.

**A cikluslopásos DMA átvitel folyamata**

1. A DMA vezérlő felprogramozása: a CPU végrehajt két I/O utasítást, amely letölti az IOAR és a DC regiszterbe a kezdeti értéküket. Az IOAR-nek az adatátvitelre szolgáló memóriarégió báziscímét kell tartalmaznia, a DC-nek pedig az átviendő egységek (bájt, félszó, szó...) számát.
2. Amikor az I/O egység kész az adatátvitelre, az aktiválja a CPU DMA REQUEST vezérlővonalát. A CPU vár a következő DMA töréspontig. Ezután lemond a memória adat és címsín vezérléséről és aktiválja a DMA ACKNOWLEDGE jelzést. A DMA REQUEST és a DMA ACKNOWLEDGE lényegében a BUS REQUEST és a BUS ACKNOWLEDGE vonal a memóriasín számára. A szimultán DMA megszakítások kezelése a sín prioritásos vezérlési technikáknál megtárgyaltakhoz hasonlóképpen oldhatók meg.
3. Most az I/O egység és a memória között megvalósul a közvetlen adatátvitel. Miután egy adategységet átvittünk, az IOAR és a DC sorrendben inkrementálódik és dekrementálódik.
4. Amennyiben a DC-t nem dekrementáltuk nullára, de az I/O egység nincs READY állapotban a következő adat küldésére vagy fogadására, visszaadja a vezérlést a CPU-nak
   * a memóriasín felszabadításával és
   * a DMA REQUEST vonal deaktivizálásával.

A CPU a DMA ACKNOWLEDGE vonal deaktiválásával és normál művelet visszajelzéssel válaszol.

1. Ha a DC nullára dekrementálódik, akkor az I/O egység ismét lemond a memóriasín vezérléséről. Ez küldhet egy interrupt jelzést a CPU számára. A CPU válaszolhat az I/O egység leállításával vagy egy új I/O átvitel kezdeményezésével.

**7. Számítógépek szabványos külső illesztő felületei.**

**A külső illesztő felületek**

A perifériák egy része a nagyobb méretük vagy a speciális funkciójuk miatt nem közvetlenül a számítógép bővítő sínére, hanem egy külső illesztő felületre csatlakozik. Ez lehet

* szabványos soros port
* szabványos párhuzamos port
* sínszerű SCSI felület.

Az illesztők fontos jellemzője, hogy pont-pont vagy multipont csatlakozást biztosítanak.

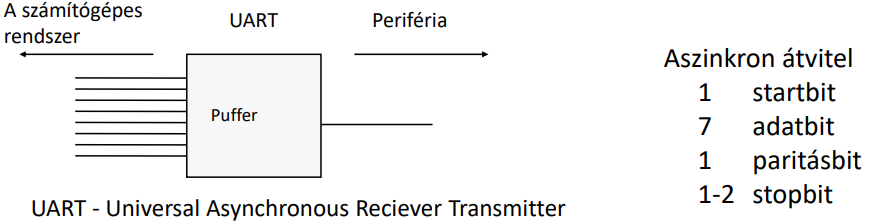
A **pont-pont csatlakozás** egy kábellel köti össze a számítógépes rendszert és a perifériát. Az IBM PC-vel kompatibilis gépeknél ez jelenti a tipikus megoldást, így csatlakozik a számítógéphez, például, a billentyűzet, az egér, a nyomtató, a modem stb.

Ma már a **multipont csatlakozást** használunk. A multipont illesztők lényegüket tekintve külső sínek, mivel ugyanazon logikával működnek, mint a sínrendszer. A mai párhuzamos illesztők közül, például, ilyen a merevlemez-vezérlő, mely tipikusan két merevlemezt képes vezérelni, a SCSI vezérlő, továbbá a multimédia vezérlő, mely alkalmas CD-ROM, audio és video csatlakoztatására. A korszerű soros illesztők között is találunk multipont csatlakozóra példát: ezek közé tartozik az USB és a Thunderbolt szabvány.

**Átviteli módok**

* párhuzamos, az adatszó minden bitje egyszerre kerül átvitelre
* soros, az adatbitek időben egymás után kerülnek átvitelre

**Soros (serial) adatátvitel**

****

Ennél az adatátvitelnél az egyes biteket **egy vezetéken** "sorban" egymás után, tehát időben eltolva továbbítják a két eszköz között.

Bár ez nyilvánvalóan lassabb átvitelt eredményez, mint a párhuzamos, az előnye viszont a zavarállósága, akár 50-100 méterre is biztosítható a kapcsolat.

A rendszer előnye:

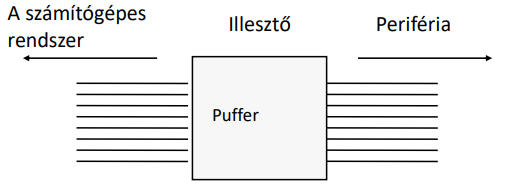
* kevés vezeték, így olcsóbb
* a külső zajoktól jobban védhető, mint a párhuzamos esetén
* mindezek lehetővé teszik, hogy nagyobb távolságra is biztonsággal továbbítsuk az adatokat.

A rendszer hátránya:

* kisebb átviteli sebesség a párhuzamoshoz képest.

Pl. Az RS-232C vagy CCITT V.24

**Párhuzamos (parallel) adatátvitel**

****

Az adatszó minden bitje egyszerre kerül átvitelre. A párhuzamos adatátvitel azt jelenti, hogy az egyes bitek egyszerre, egymás mellett több vezetéken haladhatnak. Az adatvezetékek száma megegyezik az egyszerre átvihető bitek számával. Ez a szám általában követi a gép szóhosszának alakulását.

A rendszer előnye:

* mivel egyszerre több bit vihető át, ezért az átvitel sebessége nagyobb, mint a "konkurens", soros átvitelé.

A rendszer hátrányai:

* sok vezeték kell a megvalósításához, ezért drágább;
* mivel több vezetéken egymás mellett haladnak az adatok, így azok főleg nagyobb távolság esetén zavarhatják egymást. Nagyobb távolság áthidalása esetén az interferencia valószínűsége növekszik, és a vezetékek hosszából adódóan az egyes bitek késleltetése is különböző lehet. Ezért az ajánlott maximális távolság 4-6 méter.

Pl. Centronics és az IEEE 1284

**USB szabvány (Universal Serial Bus)**

Egy USB rendszer egy fa topológia. Ennek gyökere egy központi elosztó (root hub), amely a rendszersínre csatlakozik. Ez az elosztó több csatlakozót tartalmaz az I/O eszközök és az újabb elosztók számára. A teljes USB sávszélesség 1,5MB/s.

A kábelek végén két különböző csatlakozó van, az egyik csak I/O eszközbe, a másik csak elosztóba dugható. A kábel négy vezetéket tartalmaz, kettőt az adatok egyet-egyet a tápfeszültség (5V) és a föld számára.

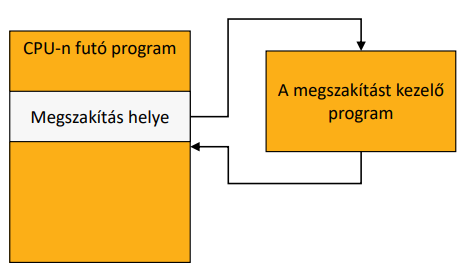
1. Egy új eszköz csatlakozik.
2. A központi elosztó érzékeli az eseményt és megszakítást kezdeményez az operációs rendszerben
3. Az operációs rendszer lekérdezi az eszköz típusát és a sávszélesség igényét.
4. Ha van elegendő sávszélesség, az operációs rendszer egy egyedi azonosítót (1-127) rendel az eszközhöz. Ezt és minden szükséges konfigurációs adatot betölti az I/O eszköz konfigurációs regisztereibe.

Logikailag az USB rendszer bitcsatornáknak tekinthető a központi elosztó és az I/O eszközök között. Minden eszköz feloszthatja a csatornáját legfeljebb 16 alcsatornára a különböző típusú adatok számára.

Minden 1,00ms-ban a központi elosztó egy új üzenetváltási keretet (frame) indít, amely mindig egy bitcsatornához csatlakozik. Egy keret csomagokból áll, amelyek közül az elsőt mindig a központi elosztó küldi az eszköznek, a többi csomag iránya tetszőleges. Az USB szabvány négyféle **keret típust** különböztet meg

* vezérlő - eszközök konfigurálása, parancsok küldése, eszközök állapotának lekérdezése
* izoszinkron - valós idejű eszközök kommunikációja, hiba esetén nem igényli az adatok megismétlését
* csoportos - nagy tömegű adatok átvitele
* megszakítás - az USB nem támogatja a megszakításokat, ezért az operációs rendszer kérdezi le

**8. Megszakítás és kivételkezelés fogalma. Megszakítási rendszer fogalma, megszakítási okok, szintek. Többszintű megszakítások kezelése. A megszakítási folyamat részfeladatai, prioritások kezelése. Megszakítások hardveres és szoftveres kezelése. (251125\_0:00)**

**Megszakítási rendszer:** A CPU-nak rugalmasan reagálnia kell bizonyos, a rajta futó program és a hozzá kapcsolódó eszközök által generált eseményekre.

A megszakítás bekövetkezésekor az éppen futó programról vezérlés ideiglenesen átadódik egy másik program számára, amely kiszolgálja a bekövetkezett eseményt. Ez vezérlés-átadó jel segítségével történik, amely a megszakítás konkrét okától függően meghatározza a szükséges rutin kezdőcímét, és hardver úton tárolja

* a megszakított programnak a megszakítás pillanatában fennálló legfontosabb állapotjellemzőit és regisztereinek tartalmát (együtt: kontextus), hogy azok később, a megszakított program folytatásakor visszaállíthatók legyenek
* beállítódnak a megszakító rutin induló állapotjellemzői és regisztertartalmai
* a megszakító rutin végén megtörténik a vezérlés visszaadása a megszakított programra, az utasításszámláló új tartalma a megszakított program végrehajtásra következő utasításának címe lesz.

**Megszakítások**: a program végrehajtásával nem közvetlen összefüggésben álló okok miatti állapotok

* Külső források - a külső eszközök által generált megszakító jelek
  + összekapcsolt gépek esetén a másik gép jelzése.
  + reset-gomb benyomása
* I/O források - a periféria eszközök - megszakítás-kérő jelzései
  + az I/O tevékenységek befejeződésekor vagy üzenetközlést kezdeményező perifériák bevezető állapotjelzései
* Géphibák az automatikus hibafigyelő áramkörök jelzései alapján
  + paritás-ellenőrzés útján feltárt adatátviteli vonalak
  + CPU regiszterek vagy az operatív memória hibái
  + az energia-ellátás és hűtő rendszer

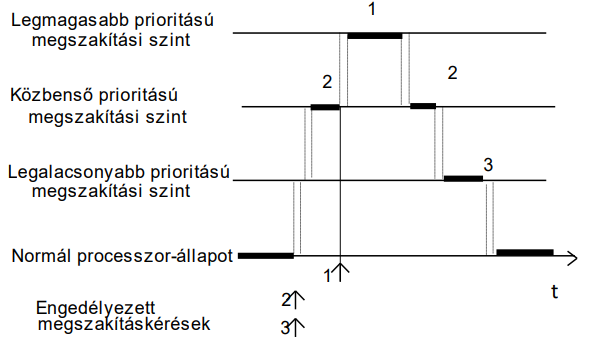
**Kivételek**: a futó program utasítás-végrehajtása vagy végrehajtásának megkísérlése következményeként a CPU-n kialakult állapotok.

* Memóriakezelés
  + lapváltási igény
  + memóriavédelem megsértése egy időben több, egymástól független feladat programok adatainak védelme
  + tényleges tárkapacitás túlcímzése
  + címzési előírások megsértése ugróutasítás páratlan címre, bájthatárok megsértése
  + veremtúlcsordulás
* Aritmetikai-logikai műveleteknél
  + integer és a lebegőpontos számok túl/alulcsordulása
  + nullával osztás kísérlete
  + definiálatlan műveleti kód
* Nem létező I/O eszköz indítása

**Megszakítási szintek**

**Egyszintű**: Egyetlen megszakítási szint létezik; amíg egy megszakítás kezelése folyik, újabb nem jöhet létre.

**Többszintű**: Minden forráshoz vagy forrásosztályhoz külön szint tartozik, lehetővé téve a prioritások szerinti egymásba ágyazott kezelést.

**Többszintű megszakítások kezelése**

Ezzel az algoritmussal a CPU az összes, az aktuálisnál magasabb prioritásszintű engedélyezett megszakításkérésre szinte azonnal reagál, és a legmagasabb prioritás-szintű kérés feldolgozását kezdi meg elsőnek.

Egy megszakítás-kérés feldolgozása után a megszakító rutin "Load PSW" utasítással mindig annak a szintnek adja vissza a vezérlést, ahonnan kapta. Így a megszakítás-kérések egymás utáni feldolgozása is fontossági sorrendjükben következik be.

Mivel egy megszakításkor az új PSW-vel egy új maszk-szó is aktiválódik - a további megszakítások engedélyezése szempontjából - már az új maszk-szó az érvényes.

Többszintű megszakítási rendszerben a kiválasztó logika működésének egy másik lehetséges algoritmusa a következő:

* a kiválasztó logika keresi a pillanatnyi CPU-szintnél magasabb prioritási szintű engedélyezett megszakítás-kéréseket.
* ha talál ilyeneket, kiválasztja közülük a legmagasabb prioritási szintűt, és megszakítást hajt végre erre a szintre.
* a megszakított program állapotjellemzői a főtárnak a megszakított szinthez tartozó kijelölt területén tárolódnak, majd az új szinthez tartozó, szintén kijelölt területről új állapotjellemzők töltődnek be.

**Megszakítási folyamat részfeladatai:**

1. **Előkészítés:** A forrás aktiválja az INTR vonalat, a CPU befejezi az aktuális utasítást, majd az INTACK jellel nyugtázza a kérést.
2. **Hardveres mentés:** A CPU menti a környezetet (PC, állapotregiszterek) a verembe.
3. **Szoftveres mentés:** A kiszolgáló rutin elején a regiszterek tartalmát mentik.
4. **Kiszolgálás:** A konkrét feladat végrehajtása (pl. adatbeolvasás).
5. **Visszatérés:** Regiszterek visszaállítása és a megszakított program folytatása.

**Prioritások kezelése:**

Egyszerre több forrás is kérhet kiszolgálást, ezért a megszakítás forrásának azonosítása után a rendszernek el kell tudni dönteni a megszakítás-kérés kiszolgálásának sorrendjét.

* **Prioritások nélküli rendszer**

A megszakításokat beérkezésük sorrendjében kerülnek kiszolgálásra. Nagy hátránya, hogy az időérzékeny megszakítások kiszolgálása így megkéshet, és ezért esetleg adatvesztés következhet be.

* **Prioritásos megszakítási rendszer**

A prioritásos megszakítási rendszer esetén egy prioritási sorrendet rendelünk az egyes forrásokhoz

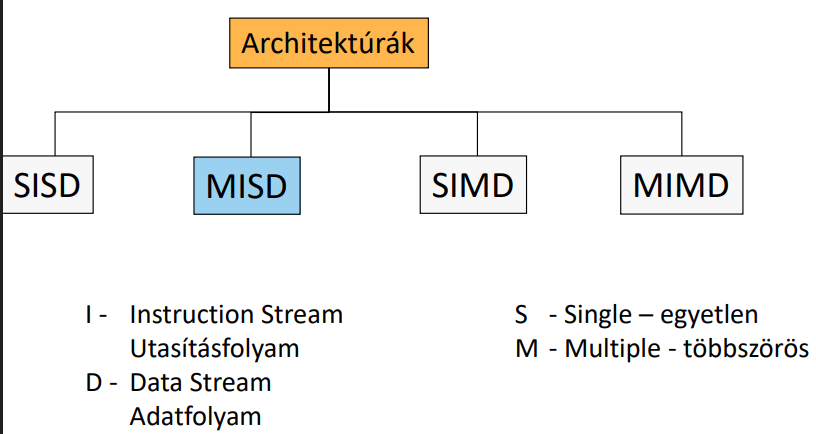
**A megszakítás kiszolgálásának a hardver által végzett feladatai**

* a CPU elkezdi annak az előkészítését, hogy az eddig futó program helyett a megszakítást kiszolgálását végző programot kezdhesse végrehajtani. Ennek során a programtól általában függetlenül (tehát hardver úton) egy erre a célra kijelölt memória-tartományba, a veremtárolóba kimenti azokat az állapot-információkat (a PC és az állapotregiszterek tartalma), amelyek a megszakított programnak a megszakítás bekövetkezése utáni folytatásához szükségesek.
* a CPU betölti a megszakítást feldolgozó program első utasításának címét a PC-be, továbbá esetleg (nem minden architektúra esetén) be kell töltenie a megszakítást kiszolgáló program futásához szükséges állapot-információkat. Szintén architektúra-függően a megszakítást kérő program lehet egyetlen program az összes megszakítás feldolgozására, lehet egy-egy önálló program minden egyes megszakítás-típushoz.

**A megszakítás kiszolgálásának a szoftver által végzett feladatai**

* a megszakítást feldolgozó program első utasításai a megszakított program regisztertartalmak mentését végzik, szintén a veremtárolóba.
* amennyiben egyetlen megszakítás-kiszolgáló program van, vagy egy megszakítástípushoz több egység is tartozik, először beazonosítja a megszakítás-kérő egységet.
* a megszakítás tényleges kiszolgálása. Ennek során például I/O megszakítás esetén beolvassa a megszakítást kérő I/O port adatregiszterének tartalmát, esetleg további parancsokat küldhet az I/O egység számára, és azok végrehajtását is ellenőrizheti az I/O egység státuszregiszterének újabb beolvasásával
* a megszakítás kiszolgálásának befejeződése után (amennyiben a megszakítás nem okozott rendszer-leállítást) gondoskodni kell arról, hogy egy programmal visszaírjuk a megfelelő regiszterekbe a megszakítási veremtárolóból a megszakított program folytatásához szükséges adatokat, majd ezután megkezdődhet a megszakított program következő utasításának végrehajtása.

**9. A többprocesszoros rendszerek értékelési szempontjai, előnyei az egyprocesszoros rendszerekkel szemben, a korlátaik. A párhuzamos architektúrák Flynn féle és modern osztályozása. A többprocesszoros rendszerekben jellemzően alkalmazott topográfiák. A különböző topográfiákon megvalósítható topológiák. A párhuzamos architektúrák tervezési szempontjai, fejlődési trendje és jellemző alkalmazási területei. A többprocesszoros rendszerek osztályozása logikai struktúra, a csatolás foka szerint. ()**

**A számítógép architektúrák Flynn-féle osztályozása**

A Flynn-féle osztályozás számítógéparchitektúrák párhuzamosság szerinti osztályozási módja, amit Michael J. Flynn publikált 1966-ban.

Flynn osztályozási modellje az egyidejű utasítás- (vagy vezérlési), illetve adatfolyamok alapján különbözteti meg az architektúrákat.

Flynn-féle osztályozás korlátja, hogy nem kezeli a párhuzamosság szintjét (alkalmazás/ folyamat/szál/utasítás).

Szakmai körökben vitatott, hogy létezik-e MISD architektúra, azaz lehet-e egy adaton egyidejűleg több műveletet elvégezné. Egyes vélemények szerint ide tartozhatnak a futószalag processzorok, mert az előző utasítás még nem fejeződött be, amikor hozzálátunk a következő utasítás végrehajtásához.

Modern Flynn- féle osztályozás

**(SISD) Single Instuction Single Data Stream – egy utasítás-, egy adatfolyam**

A klasszikus, szekvenciális Neumann architektúrájú számítógép, amely sem adat-, sem utasítás szinten nem alkalmaz párhuzamosságot.

Egyetlen vezérlőegység (CU) kéri le az egyetlen utasításfolyamot (I) a memóriából. A CU azután létrehozza a megfelelő vezérlőjeleket, hogy az egyetlen feldolgozó egység (PE) egyetlen adatfolyamon (D) elvégezze a műveletet. Tehát a végrehajtó egyszerre egy utasítást végez egy adaton.

A **SISD** eszközök közé tartoznak a hagyományos egyprocesszoros rendszerű gépek, mint a korai PC-k vagy a régi mainframe-k.

**Párhuzamos számítógépek tervezési szempontjai**

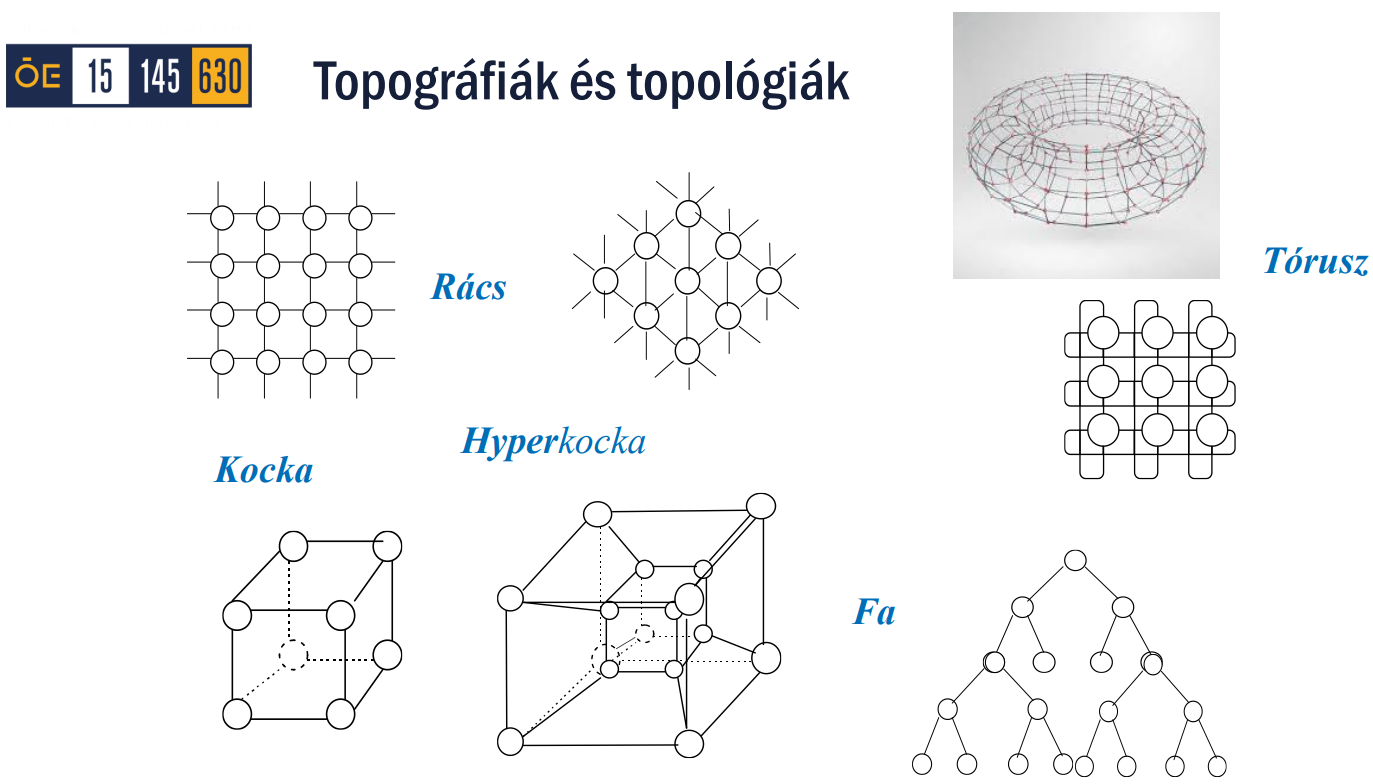
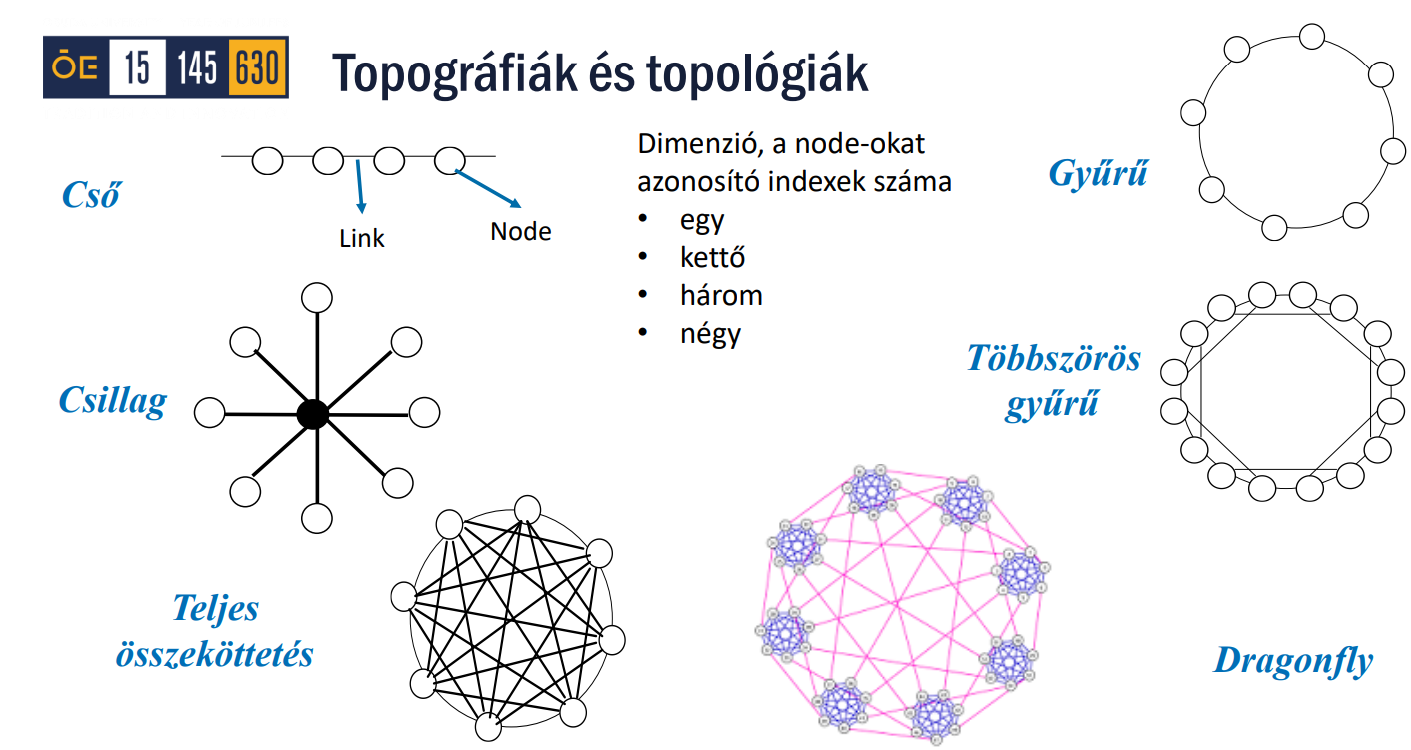
* **Processzorelemek** - típusa, teljesítménye és száma a minimális ALU-tól a komplett számítógépekig
* **Memóriamodulok** - típusa, mérete és száma a CPU-ba integrálttól a nagyméretű dinamikus memóriamodulokig két-, három- vagy négyszintű cache-sel

**A processzorelemek és a memóriamodulok összekapcsolási módja** - statikus vagy dinamikus

**Többprocesszoros rendszerek előnye az egyprocesszoros rendszerekkel szemben**

* nagyobb hatékonyság/ár arány a többszörözött olcsó hardveregységek használatának köszönhetően
* nagyobb teljesítő -vagy átbocsátóképesség
* nagyobb és összetettebb feladatok megoldása
* könnyű átkonfigurálhatóság, rugalmasság
* nagyobb megbízhatóság redundáns elemek viszonylag olcsó alkalmazásával
  + rendelkezésre állás: MTBF + MTTR
  + hibatűrés
* szervizelhetőség
  + karbantarthatóság
  + javíthatóság

**Topográfiák és topológiák**

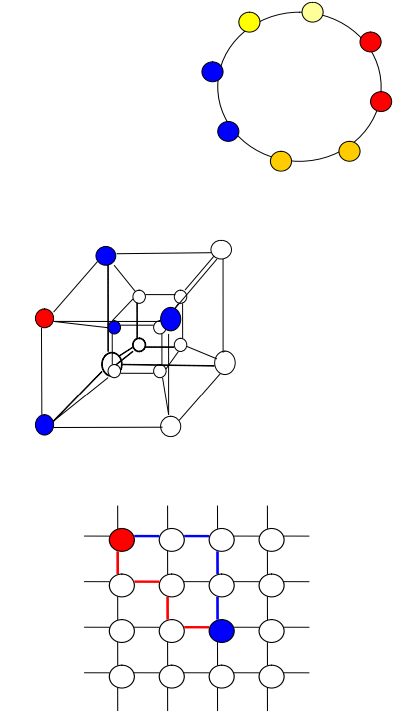
****

**A kapcsolatok jellemzői**

Egyidejűség (simultaneity) a rendszer egyidejűleg kommunikáló processzorainak az aránya

* S = m/N
* m – az egy időben üzenetet küldő processzorok száma

Üzenetterítés (broadcast scope) a rendszer egy adatelemet max. hány processzorhoz képes egyszerre továbbítani

* B = b/(N-1)
* b – az egy időben üzenetet fogadó processzorok száma

Összekötöttség (connectivity) egy processzorközi kapcsolat lehetséges útvonalainak az aránya

* C = r/N! r – az átvitel különböző lehetséges útvonalainak a száma
* N – a rendszer összes processzorának száma

**Többprocesszoros rendszerek osztályozása csatolás foka szerint**

**Szorosan csatolt rendszerek** (valódi többprocesszoros rendszerek)

* általános célú processzorok azonos jellemzőkkel
* a processzorok közös tárterülettel rendelkeznek, a közös vagy osztott táron kívül lehet saját táruk is
* közös operációs rendszer
* a processzorok terhelése közel azonos
* gondosan kialakított szinkronizációs eljárásokra van szükség

**Lazán csatolt rendszerek** (számítógép hálózatok is)

* önálló, egymástól független számítógépek alkotják
* az adatátvitel kommunikációs interfészeken át bonyolódik
* szigorú átviteli protokollok szerint nagysebességű soros átviteli csatorna
* a hálózat csak a kommunikáció számára létezik
* bármelyik számítógép hozzáférhet bármely más gép erőforrásaihoz és adataihoz

**Mérsékelten csatolt rendszerek** (elosztott intelligenciájú rendszerek)

* önálló rendszerelemek alkotják, saját központi egység, programtár, adattár és be/kiviteli egységek
* a processzorok saját hozzárendelt feladatokat látnak el
* a processzorok az alkalmazói program be/kiviteli műveleteit és a rendszerkommunikációval kapcsolatos feladatokat is ellátják
* a processzorok komplexitása különböző lehet, a processzorok HW és SW kiépítettsége a feladathoz optimalizált
* a processzorok közötti kommunikáció főleg adatátvitelre korlátozódik, statikus terhelés

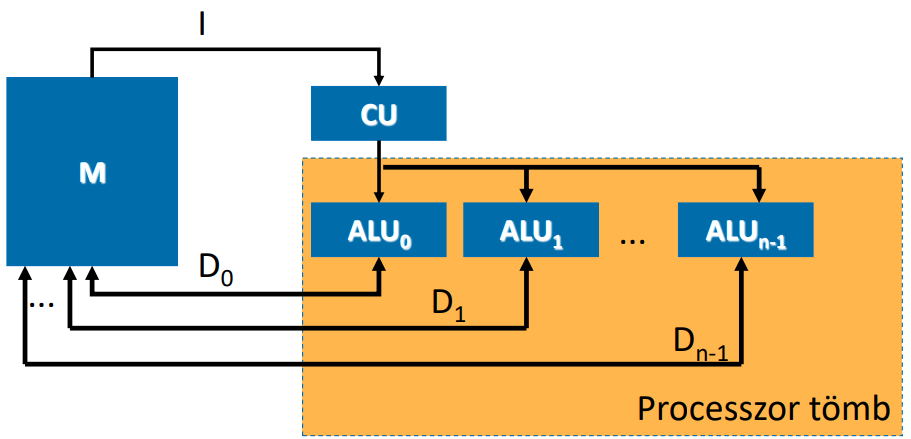
**10. SIMD architektúrájú számítógépek általános felépítése, jellemzőik. Jellemző géptípusok. A tömbprocesszoros számítógépek. Jellemzőik, előnyeik, hátrányai. A vektorprocesszoros számítógépek általános felépítése, fejlődési trendje, jellemző géptípusok**

**SIMD architektrúrája:**

**Single Instruction, Multiple Data Stream - egy utasítás-, több adatfolyam**

A számítógép egyetlen utasítás folyamot (I) hajt végre több adatfolyamon (D0 -Dn-1 ) egyszerre, az adatstruktúrában rejlő párhuzamosságot kihasználva. A processzor elemek egy közös vezérlőegységet használnak. Tehát a végrehajtó egyszerre egy utasítást végez több adaton.

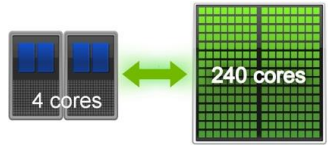
Az ilyen elvű egységeket vektorfeldolgozónak is nevezik. Példa: tömbprocesszor, vektor processzor, GPU.



**11. Grafikus processzorok. A klasszikus grafikus futószalag és a CUDA architektúra össszehasonlítása. GPGPU-k feldolgozási paradigmái, mikroarchitektúrája. A GPU-k főbb alkalmazási területei.**

**Grafikus processzorok: GPU**

A memória elérés a CPU-nál mondhatni a leglassabb művelet. A cache elérése gyors, viszont mérete csak pár MB, és nagyon drága. GPU-nál a gyors memória-elérés megvalósítása jóval olcsóbb és gazdaságosabb.

A **GPU SIMT (Single Instruction Multiple Thread) architektúrájának** köszönhetően egyetlen utasítás folyamot több szálon tud végrehajtani. Ez azt jelenti, hogy ugyanazt az utasítást több 10.000 szálon tudja párhuzamosan végrehajtani.

**GPGPU:**

A GPGPU (General-Purpose Computing on Graphics Processing Units) a grafikus processzorok (GPU) általános célú számításokra való használatát jelenti, amelyek nem kapcsolódnak a grafikus megjelenítéshez. A GPU-k masszív párhuzamossága, ami az ábrák megjelenítéséhez szükséges, lehetővé teszi számukra, hogy drámaian felgyorsítsanak olyan számításigényes feladatokat, mint a gépi tanulás, a tudományos kutatás, az adatfeldolgozás és a szimulációk, amelyeket hagyományosan a központi processzor (CPU) végzett el.

2007-ben jelennek meg az első, felhasználó számára is programozható GPU-k a GPGPU-k. A CUDA (Compute Unified Device Architecture) az NVIDIA által kifejlesztett párhuzamos számítástechnikai platform és programozási modell, amely lehetővé teszi a fejlesztők számára, hogy a kompatibilis GPU-k erejét általános célú feldolgozásra használják. Ezáltal a szoftverek jelentősen felgyorsíthatják a számításigényes feladatokat, mint például a tudományos kutatás, a mesterséges intelligencia, vagy a grafikai feldolgozás. A fejlesztők C++, Python, Fortran és más nyelveken programozhatnak a CUDA kiegészítőkkel.

**GPU modell**

A modern GPU-k megértéséhez először a klasszikus grafikus futószalagot kell megismerni. A folyamat a geometriai csomópontok – a vertexek – felépítésével kezdődik, melyeket a térben eltolni – transzformálni – és világítani kell. Ezután következik a csomópontok háromszögekbe rendezése.

Az utóbbi időben a harmadik lépés, a pixelszintű számolás kapta a legnagyobb hangsúlyt. Az egyes képpontok színének meghatározása folyik itt bonyolult matematikai számítások – árnyalási egyenletek – és textúrázási eljárások segítségével. A negyedik lépést az NVIDIA ROP (raster operation) vagyis raszteres műveleti résznek, a Microsoft pedig Input Mergernek nevezi. Itt összegződnek a pixel shaderek eredményei és itt kerül a képre az anti-aliasing (élsimítás). Az ötödik elem a memória, melyben a végleges kép fölépül, a textúrákat és egyéb adatokat tárolja.

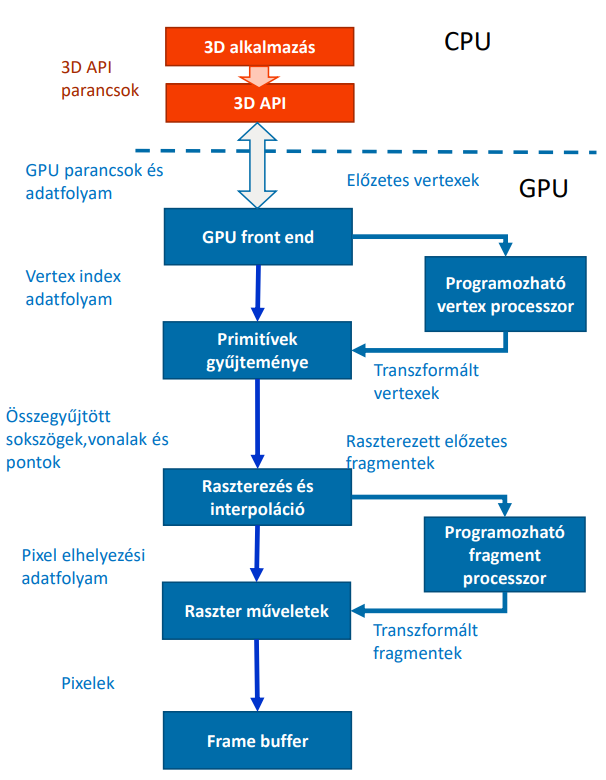
Az évek során ez az elvi felépítés semmit sem változott, csak az egyes lépcsők fejlődtek. A DirectX 7-ben a vertexes rész kikerült a CPU-ból, és a grafikus processzor T&L (Transform & Lighting) egysége foglalkozott vele. A DirectX 8-ban a pixel egységek programozhatósága debütált, ami a DirectX 9 folyamán fokozatosan finomodott. A ROP részen fejlődtek a Z-buffer technikák és az élsimítás, a memória pedig egyre gyorsabb és gyorsabb lett.

**CPU és GPU kapcsolata**

Három lépésben tudunk bármilyen műveletet végrehajtatni a GPU-val

* adatokat át kell másolnunk a GPU-ra a hostról (CPU)
* a GPU-ra írt programmal az adatokon végrehajtjuk a műveleteket
* az adatokat vissza kell másolnunk a GPU-ról a hostra

A GPU PCI-Express 16x nagy sávszélű busszal kapcsolódik a hosthoz. Ez a busz legfőképpen a DMA-n keresztüli adatforgalmat biztosítja a CPU és GPU memóriák között. Egyik process unit sem képes megcímezni a másik memóriáját. (DMA végzi)

Memória elérés

* Paged memory
* Page-locked memory
* Mapped memory
* Write combining

https://www.youtube.com/watch?v=oi\_9fujflhI

**Klasszikus grafikus futószalag:**

A különböző típusú adat formátumok miatt a vertexek és a pixelek előállítása programozhatóvá válik. A futószalag többi funkciója továbbra is fixen kerül végre hajtásra.

A programozható és a fixen beépített programok végrehajtási ideje kritikus a futószalag optimális működése szempontjából.

**CUDA architektúra**

Tesla a termék neve, amelyet GPGPU (General Purpose GPU) célra fejlesztett és dobott piacra az nVIDIA. Ez az igazi első CUDA-kompatibilis architektúra.

A nagy fejlődés az architektúrában: Thread Processing Clusters (TPC)

Mindegyik TPC közvetlen eléri a DRAM vezérlőt (crossbar hálózat). A memória fel van osztva 8 vezérlőre (Core). A memória a GPU chipjén kívül található. Mindegyik TPC-nek van 2-3 stream processzora.

CPU-memória: 25GB/s

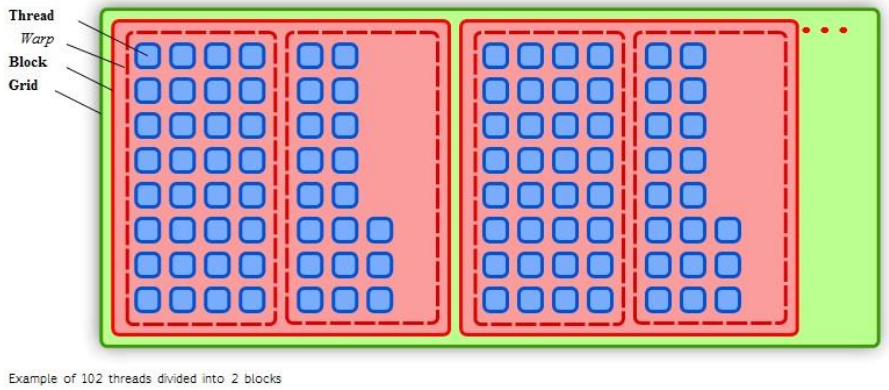
GPU-memória: 77GB/s

CPU-GPU: 8GB/s DMA-n keresztül

Tehát a memória elérés a GPU-nál 3- szor gyorsabb, mint a CPU-nál, azóta ez többszörösére nőtt.

Ezen kívül core-oknak van megosztott memóriájuk, mindegyik rendelkezik saját cache-sel.

**Szoftver oldal**

CUDA egy kiterjesztett változata a C nyelv GPU programozási ágának A programozási modell legfőképpen az nVIDIA grafikus kártyák architektúrájával párosult.

Thread – Szál

Warp – 32 szálból álló köteg

Block – ezek tartalmazzák a szálakat :

CC1.x – 512, CC2.x – 1024, CC3.x – 2048

Grid – rács, mely a blokkokat

tartalmazza : architektúra-függő a mérete

**Fermi architektúra (nincsen a tételben kimondva, de lehet rá fog kérdezni)**

Legelső olyan GPU architektúra, melyet kifejezetten a szuperszámítások elvégzésére terveztek. 512 vagy annál több CUDA core a legújabb modelleknél

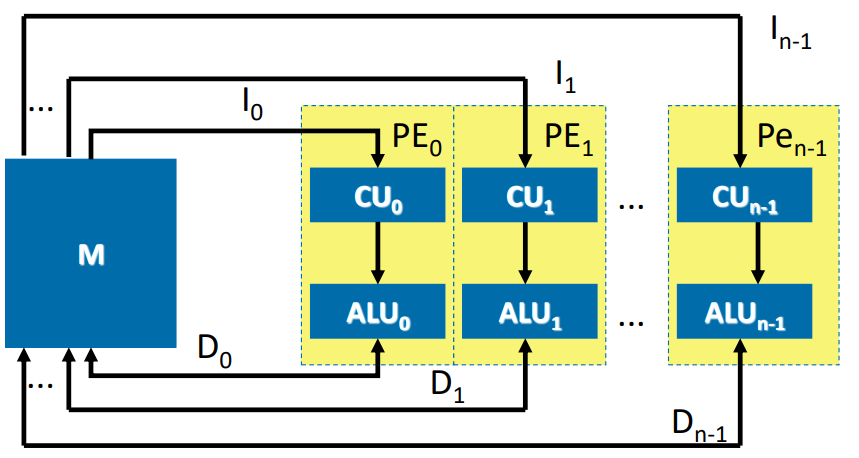
**12. Multiprocesszorok. Sínrendszerű SMP architektúrák. UMA és NUMA rendszerek. Gyorsító tár koherencia. A „WriteThrough“gyorsító tár koherencia protokoll. Gyorsító tár koherencia. A MESI gyorsító tár koherencia protokoll. Az NCNUMA és CC-NUMA multiprocesszorok általános jellemzői. Stanford DASH multiprocesszor architektúrája, a DASH katalógus felépítése, előnyei, hátrányai. A DASH cache koherencia protokoll algoritmusa. A nem blokkoló kapcsolóhálózatok. Sun Enterprise 10000 SMP architektúrája. Az SGI Origin 2000 felépítése, jellemzői. Topográfiája, a cache koherencia protokoll elvi alapjai. (18\_Multiprocesszorok.pdf dia 1-24 kb. minden kell. Ajánlott 251202\_0:00)**

**Multiprocesszorok:**

A multiprocesszoros rendszer egy olyan számítógép, amelyben minden CPU egy közös címzésű tárterületet használ. A processzorok a memórián keresztül kommunikálhatnak egymással, azaz közös memória használat (**shared memory**) jellemzi. Az operációs rendszernek egy példánya fut.

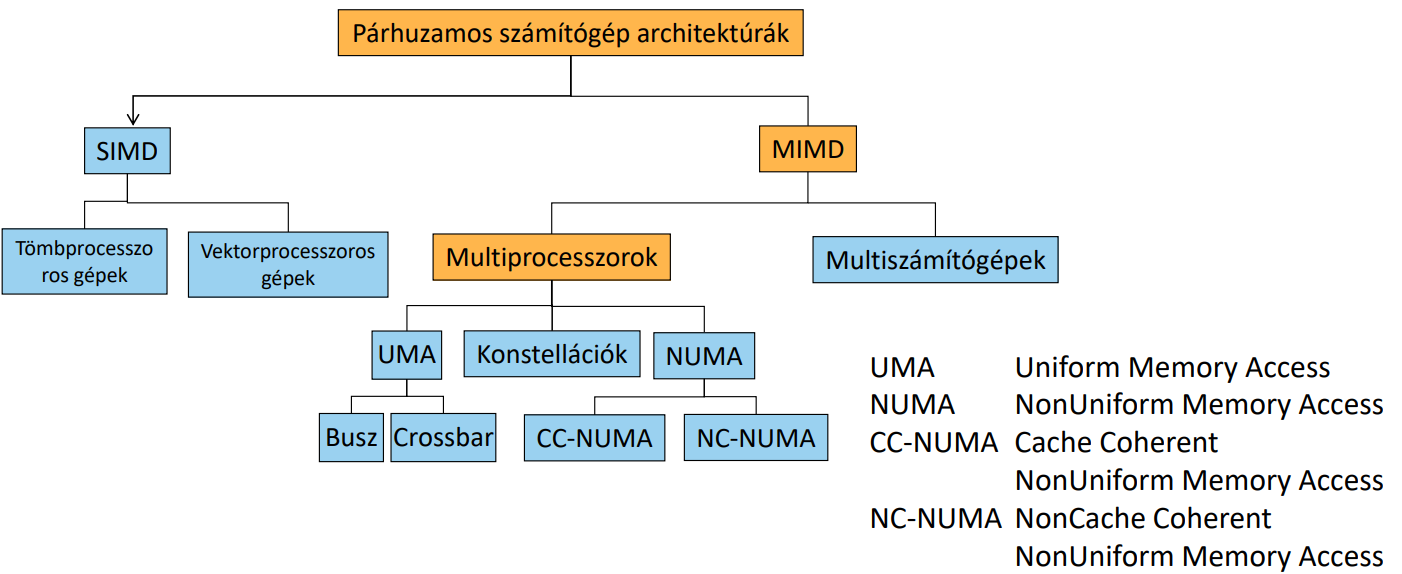
Egy speciális osztálya az **SMP** architektúra (Symmmetric MultiProcessor), amelynél minden CPU egyformán éri el az összes memóriamodult és be/kiviteli eszközt.

A multiprocesszorok az 1990-es évek második felének legsikeresebb szuperszámítógépei voltak.

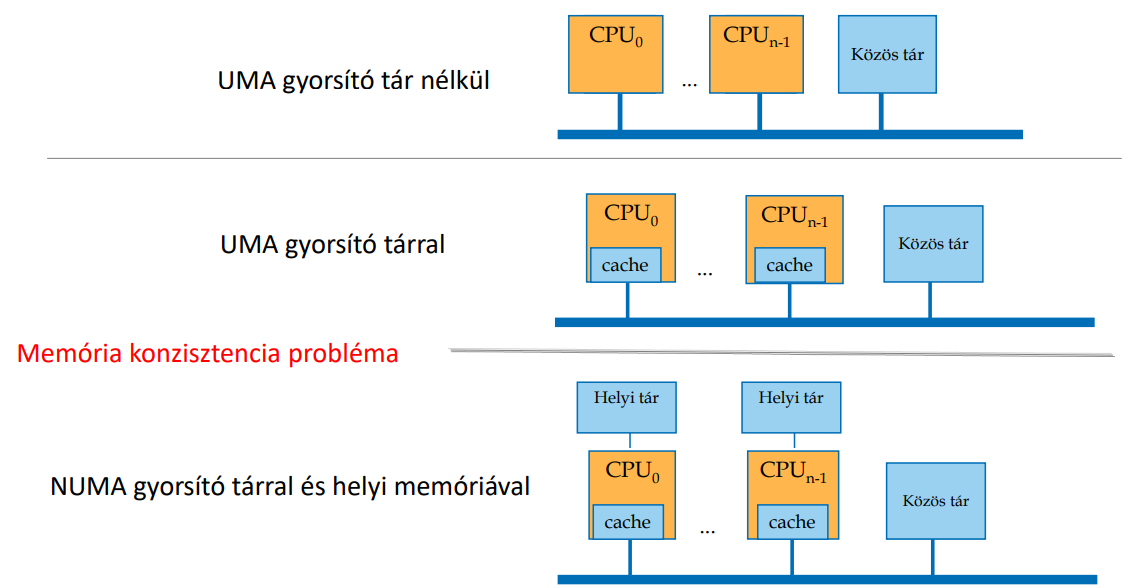
**MIMD architektúra**

Multiple Instruction, Multiple Data Stream - több utasítás-, több adatfolyam

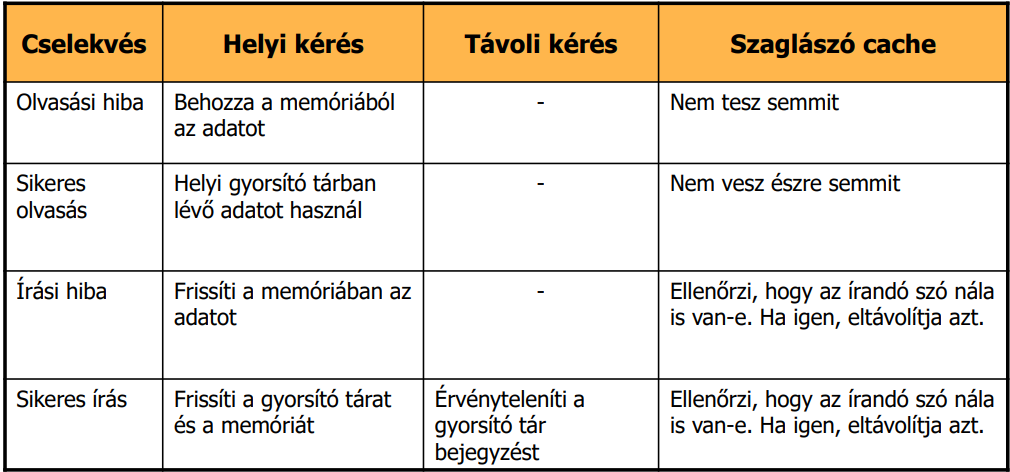
Több autonóm processzor egy időben végez különböző műveleteket különböző adatokon.

MIMD-architektúrának tekinthetők az elosztott rendszerek, akár megosztott, akár elosztott memóriaterületet használnak. Egy többmagos, szuperskalár processzor is MIMD-processzor.

**Sínrendszerű SMP-architektúrák**

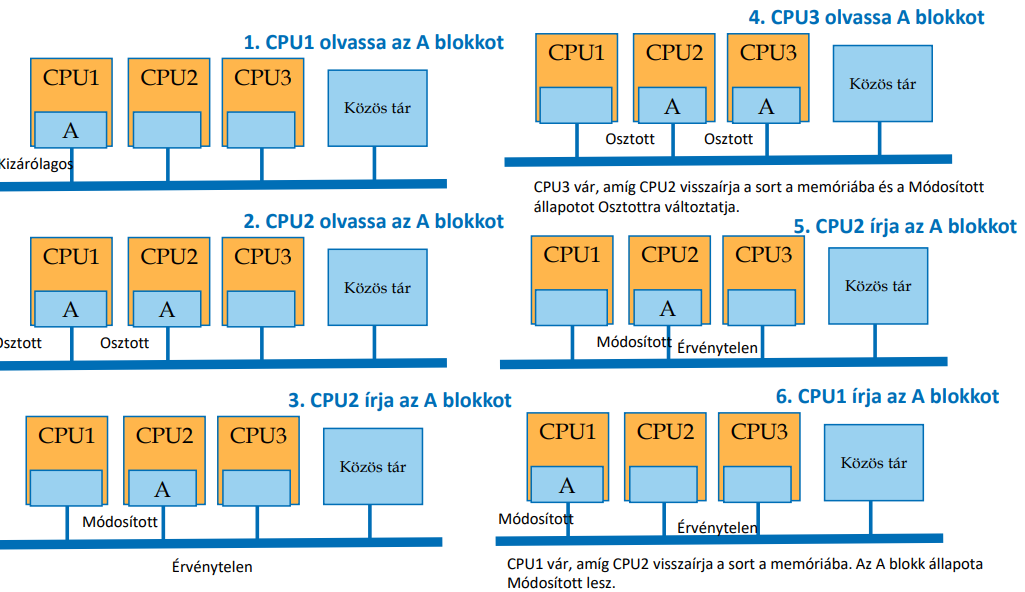
****

**Gyorsító tár koherencia protokoll „Write-Through“ – írásáteresztő protokoll**

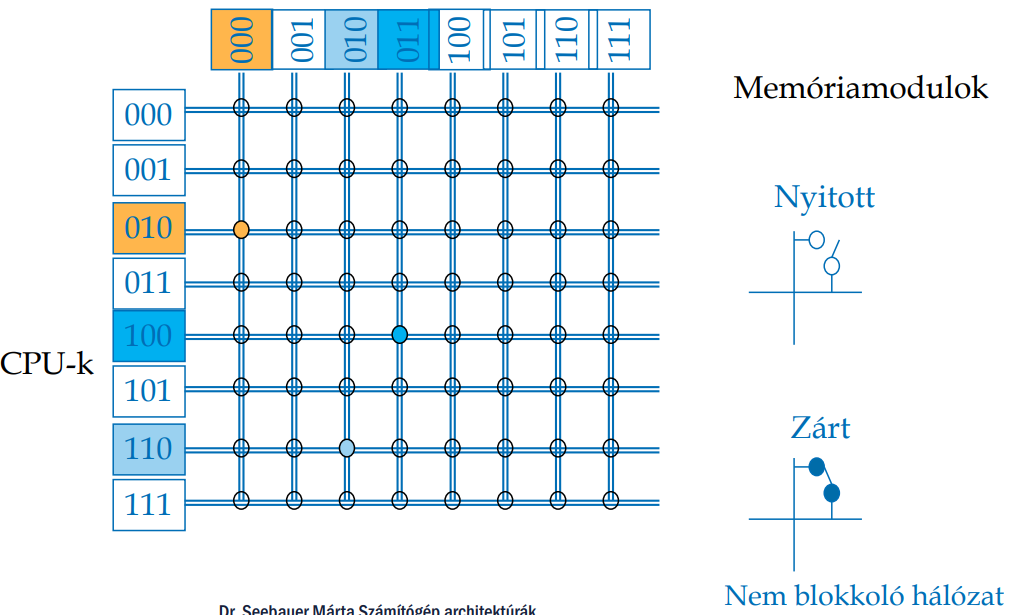


**A MESI gyorsító tár koherencia protokoll**

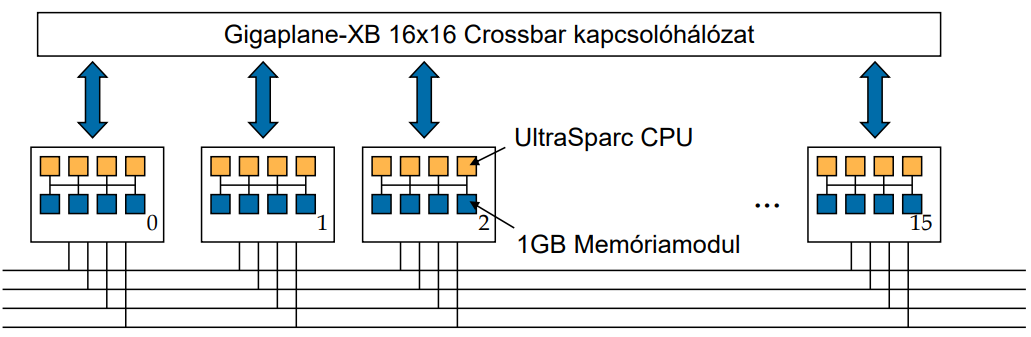
1. **Érvénytelen (Invalid)** – A gyorsító tár bejegyzés nem tartalmaz érvényes adatot.
2. **Osztott (Shared)** – Több gyorsító tár tartalmazhatja a sort, a memória frissítve van.
3. **Kizárólagos (Exclusive)** – Más gyorsító tár nem tartalmazza a sort, a memória frissítve van.
4. **Módosított (Modified)** – A bejegyzés érvényes, a memória nincs frissítve, másolatok nincsenek.



**UMA multiprocesszorok crossbar kapcsolóhálózattal (Crossbar Switch)**

****

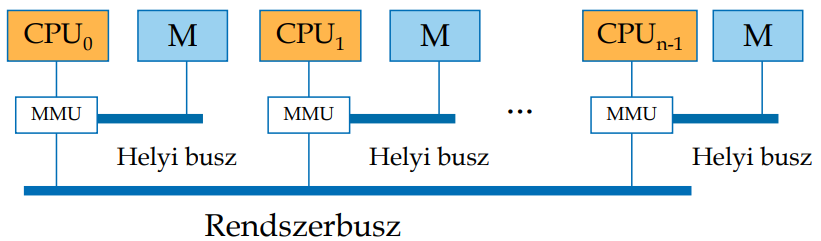
**Sun Enterprise 10000 SMP architektúrája**

****

4 címbusz a szaglászáshoz

A crossbar kapcsoló hálózaton keresztül az egyes processzor kártyák sínei egymáshoz csatlakoztathatóak. Mivel ezt a többi processzor nem észleli, ezért a négyszeres gyűrűben körbefutó címbuszt tudják a gyorsító tárak szaglászni.

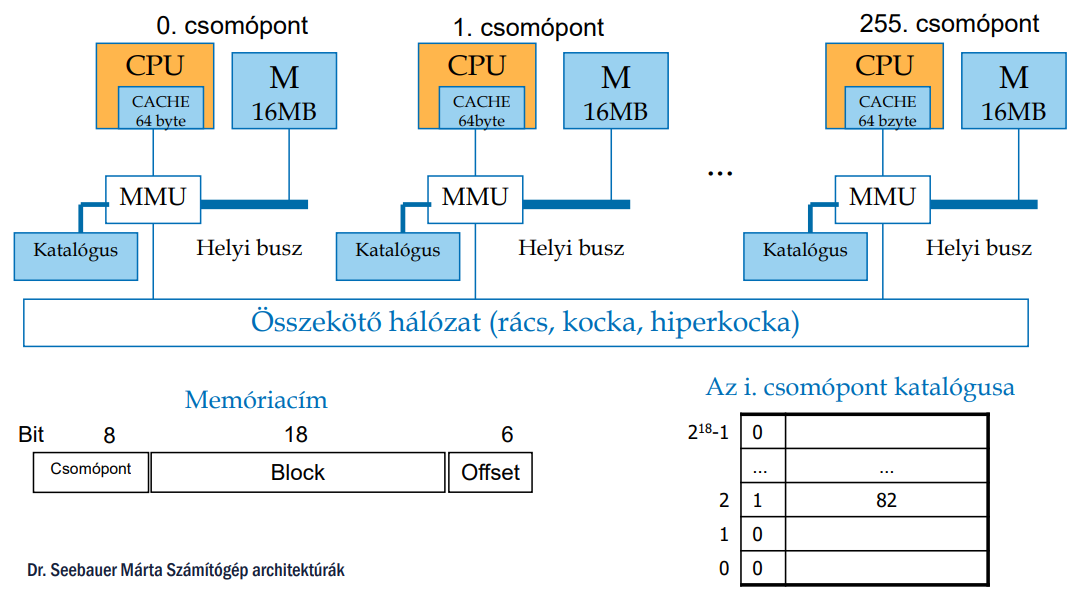
**NC-NUMA multiprocesszorok**

****

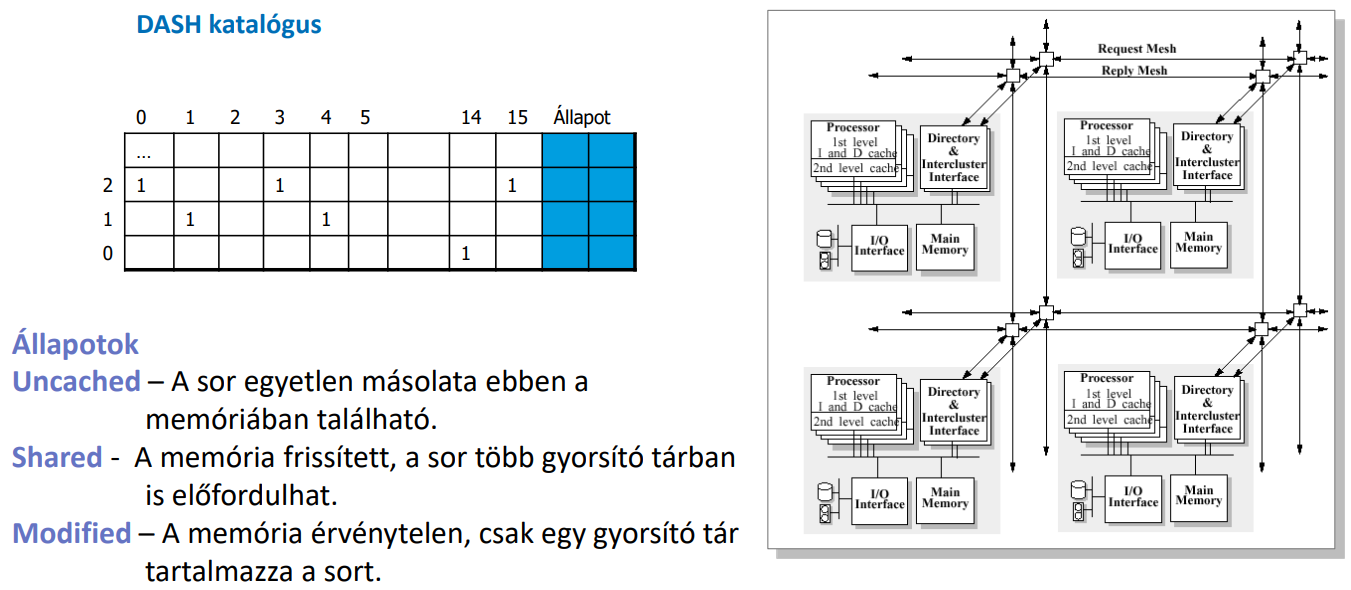
**Non-Cache NonUniform Memory Access**

Nincs cache, minden memóriaszó egy példányban létezik, így nem léphet fel koherencia probléma. Fontos a fordítóprogram szerepe a memória lapok szervezésében. A lapok cserélhetőek.

**CC-NUMA multiprocesszorok**

****

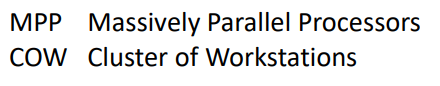
**Stanford DASH multiprocesszor (Directory Architecture for Shared Memory)**

****

**Az SGI Origin 2000 felépítése (251202\_35:00)**

**13. A hibrid architektúrák (konstelláció). Az Earth Simulator architektúrája, jellemzői és alkalmazási területei. (251202\_44:00 18\_Multiprocesszorok.pdf 25-29 oldal)**

**14. Multiszámítógépek általános felépítése, típusai és jellemzői. Az MPP rendszerek általános jellemzői, architektúrája, alkalmazási területei. Az USA ASCI programjának a célja, és a program keretében fejlesztett szuperszámítógépek jellemzői. Az EU és Magyarország szuperszámítógép programja. (251202\_48:30 19\_Multiszámítógépek)**

****

**Multiszámítógépek:**

A multiszámítógépek a MIMD architektúrák azon osztálya, amelyben a processzorok elosztott memória címtérrel (**distributed memory**) rendelkeznek. A multiszámítógépek a 2000-es évek tipikus nagyteljesítményű számítógépe (**HPC**).

Minden CPU saját lokális memóriával rendelkezik, amely a többi CPU számára közvetlenül nem elérhető.

A kommunikáció a processzorok között egy összekötő hálózaton üzenetátadással SEND és RECEIVE utasításokkal (**message passing**) folyik, esetenként külön kommunikációs processzorok vezérlésével.

Sokféle topográfia, kapcsolatépítési mód és útvonalválasztási algoritmus alkalmazható.

A multiszámítógépek jól skálázhatóak.

**Multiszámítógépek általános felépítése**

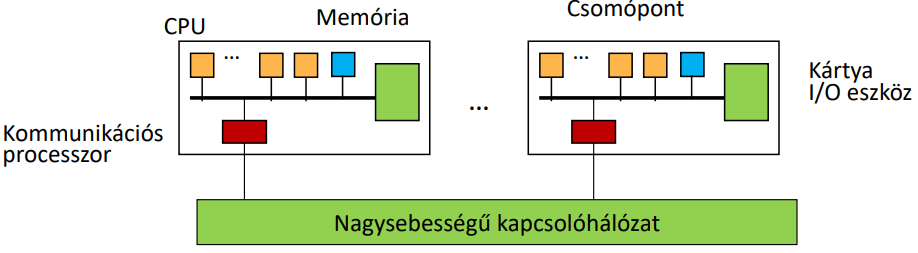
A multiszámítógépeknek két osztálya létezik

* szorosan csatolt rendszerek, MPP - Massively Parallel Systems
* lazán csatolt rendszerek, COW - Cluster of Workstations

Egy csomópontban (node) 2-4 CPU helyezkedik el, amelyek egy közös buszra csatlakoznak és közös memóriát használnak. A node-on belül tipikusan alkalmazott cache koherencia protokoll a MESI protokoll.

A node-okat összekötő kapcsoló hálózat lehet LAN, WAN vagy speciális, pl. Myrinet, InfiniBand.

A számítógépek virtualizált környezetben működnek. Az alkalmazott köztesrétegek (middleware) pl. MPI (Message Passing Interface), PVM (Parallel Virtual Machine)

****

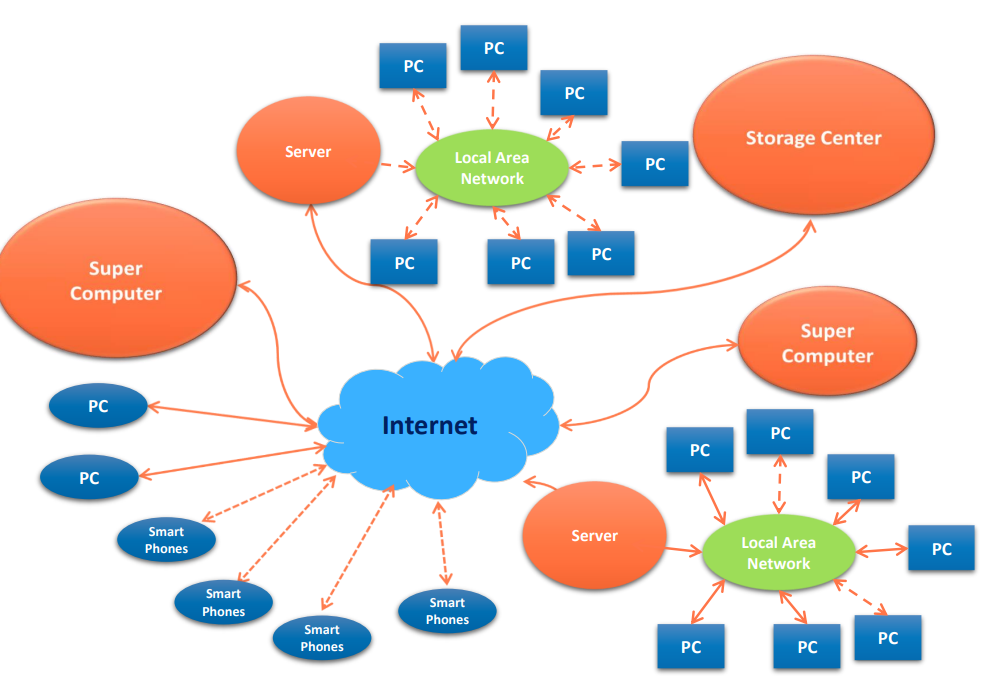
**Magyar szuperszámítógép a Komondor, ami a Debreceni Egyetemen található (19\_Multiszámítógépek.pdf 32-35 oldal)**

**251202\_1\_0:00 – 19\_Multiszámítógépek 7.oldal**

**Az USA ASCI programjának a célja, és a program keretében fejlesztett szuperszámítógépek jellemzői. – (251202\_1\_10:45)**

**15. COW architektúrák. A PC clusterek típusai, hardver és szoftver szintű kommunikációs módszerei. (251202\_1\_0:00 19\_Multiszámítógépek 10-13)**

**16. GRID rendszerek kialakulásának okai és feltételei, típusai, működése, alkalmazási területei. (251202\_1\_37:20/42:00 - 20\_Grid\_Cloud.pdf)**

****

**GRID rendszerek:**

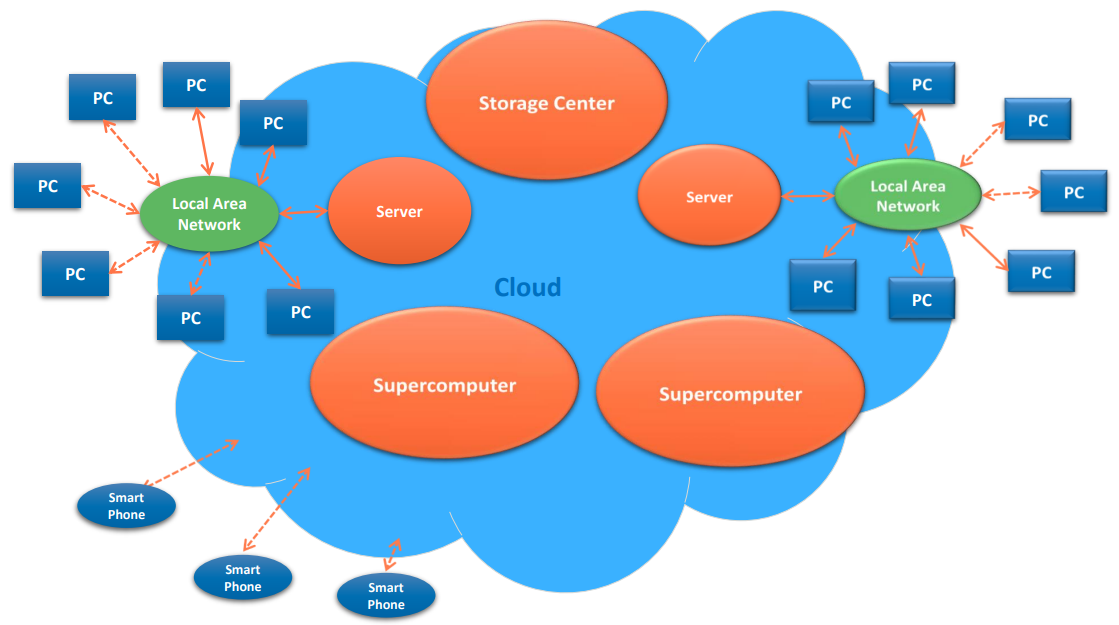
Heterogén számítástechnikai eszközök virtuális rendszerbe kötve nagy számításigényű feladatok elvégzésére

* számítási és tárolási kapacitás a szerver oldalra kerül
* virtualizált szerverek
* kötegelt feldolgozás – visszatérés a mainframe-ekhez

Három típusa alakul ki

* desktop GRID – személyi számítógépek összekötése virtuális szuperszámítógépbe, egy központi gép (bróker gép) küldi ki a feladatokat és gyűjti össze az eredményeket - SETI@Home
* alkalmazói GRID – egy vállalat vagy intézmény saját hálózatán kialakított GRID rendszer – gyártásoptimalizálás, szimuláció, modellezés, digitalizálás
* szolgáltató GRID – gépidő bérbe adása

**17. A Cloud computing kialakulása, működése, típusai. A felhőszolgáltatások szintjei és alkalmazási területei.**

****

**Cloud computing**

A szolgáltatói GRID-ből alakul ki. Az alkalmazások elosztott virtuális eszközökön futnak elfedve a valós hardver eszközöket a felhasználó elől.

A felhasználók ezeket a szolgáltatásokat lokális vagy távoli hálózatokon keresztül érik el. Jellemző a mobil applikációk használata és az alkalmazásokhoz és az adatokhoz történő interaktív, online hozzáférés.

Működéséhez stabil Internet kapcsolatra van szükség.