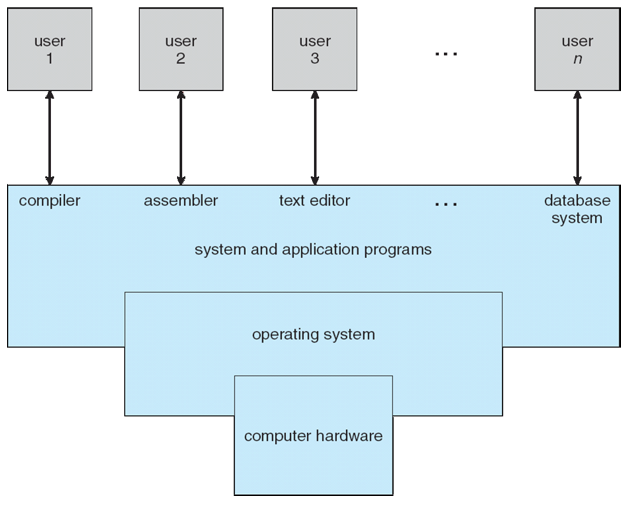
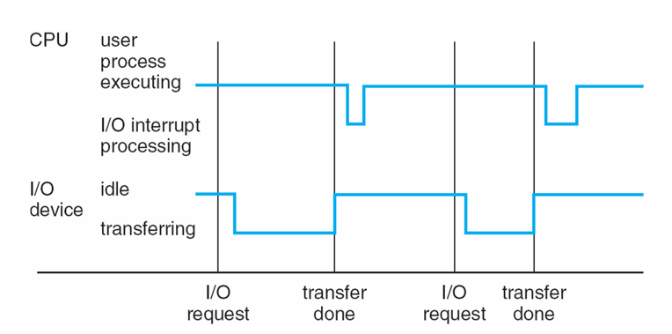
**Chapter 1**

* Operációs rendszer alapvetően a felhasználó és a számítógép közé beékelődő réteg
* Céljai
  + Végrehajtsa a programokat/feladatokat, könnyítse a felhasználó dolgát
  + Legyen kényelmes a felhasználónak
  + Hatékonyan használja a gép erőforrásait 🡪 A felhasználót nem érdekli a hogyan, csak az eredmény
* Mi is az oprendszer?
  + Erőforráskezelő
    - Kezeli a megadott erőforrásokat
    - Prioritások alapján ütemezi mely dolgok a fontosak
  + Egy irányító program
    - Végrehajt programokat, meggátolja a hibákat és a nem megfelelő használatot
  + Nincs egy univerzálisan elfogadott definíció
* Vas/Hardver-Oprendszer-Alkalmazások-Felhasználó
  + Az a lánc, amivel effektíve a felhasználó interaktál a számítógéppel
  + Erre a négy részre bontható egy számítógépes rendszer, de egymásra épülnek
* Mit kezel az oprendszer?
  + CPU
  + Memória
  + I/O eszközök
* Kernel – „*The one program running at all times on the computer”*
  + Az oprendszer része, ami mindennek az alapja, és a legfontosabb feladatokat végzi
  + Kernelen alapul a rendszer maga, és azon minden más
  + Minden, ami nem kernel az lehet rendszerprogram, vagy külső alkalmazás
* Számítógép indulása, bootolási folyamat
  + Egy bootstrap program betölt, ami legtöbbször EPROM-ban tárolnak, és firmware-nek hívnak (alapszoftver?)
  + Betölti a kernelt, majd minden mást
* Számítógép üzemelése
  + Ha több eszköz van csatlakoztatva a számítógéphez, akkor egy közös busszal vannak összekötve a közös memóriával
  + Konkurens execution probléma lehet
  + I/O és CPU tud egyszerre konkurensen működni
  + Minden device controllor feladata egy eszköz „irányítása”, és ezeknek van egy helyi buffere
  + A device controller szól a CPU-nak egy megszakítással, hogy elvégezte a feladatát

Első előadás vége

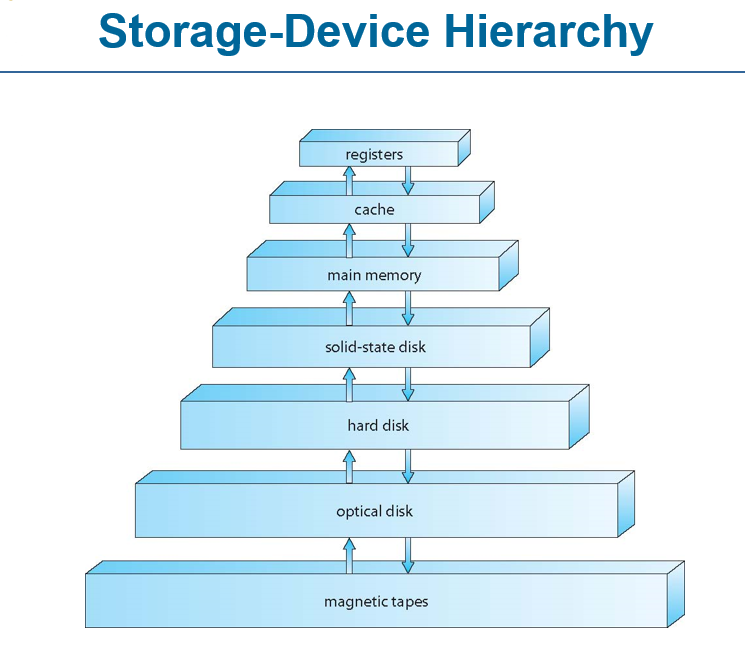
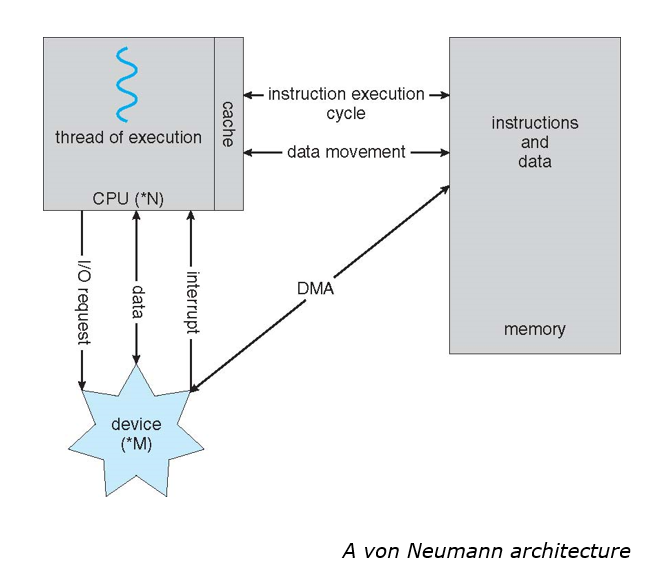
Megszakítások 🡪 Minden alapja lényegében

* interrupt service routine címét
  + Regiszterek állapotának tárolása basically
* A megszakítás elmenti az általa megszakított instruction címét
* Trap/Exception 🡪 Szoftver/program által tolt megszakítás, ami vagy hiba miatt vagy felhasználói kérés miatt jött létre
* Minden operációs rendszer megszakításokon alapul
* Példák megszakításra
  + Szoftveres megszakítás
    - 0-val való osztás 🡪 Kilövi a programot
      * Cizellelható, kivételkezeléssel
  + Pixel megjelenítése, billentyű lenyomása, akár 1 byte mozgatása
    - Ezek alapvetően a CPU feladatai
* Ki manageli a megszakításokat? 🡪 A kernel legfontosabb feladata
* Megszakítások két fő iránya
  + Lekérdezés alapú 🡪 Polling
  + Vektoralapú 🡪 Ez az elterjedtebb

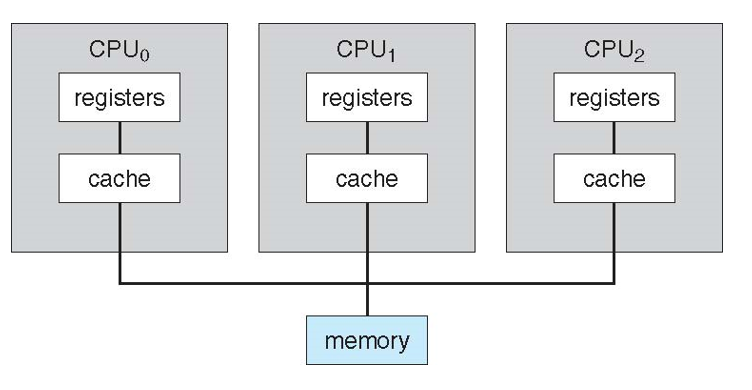
I/O struktúra, I/O műveletek

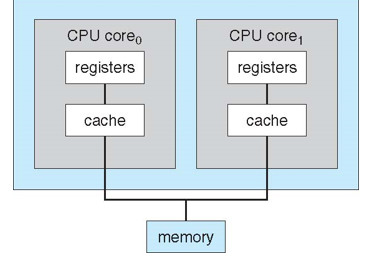
* 1. Amikor elkezdődik egy I/O művelet, elvesszük a felhasználótól az irányítást, és csak akkor adjuk vissza, ha végzett az I/O művelet
  + Wait instruction 🡪 CPU Idle
* 2. Amikor elkezdődik az I/O művelet azonnal visszaadjuk a felhasználónak az irányítást, nem várjuk meg a befejeződést
  + Rendszerhívások 🡪 Kérés az oprendszerhez, hogy a felhasználó várja meg a végét az I/O műveletnek

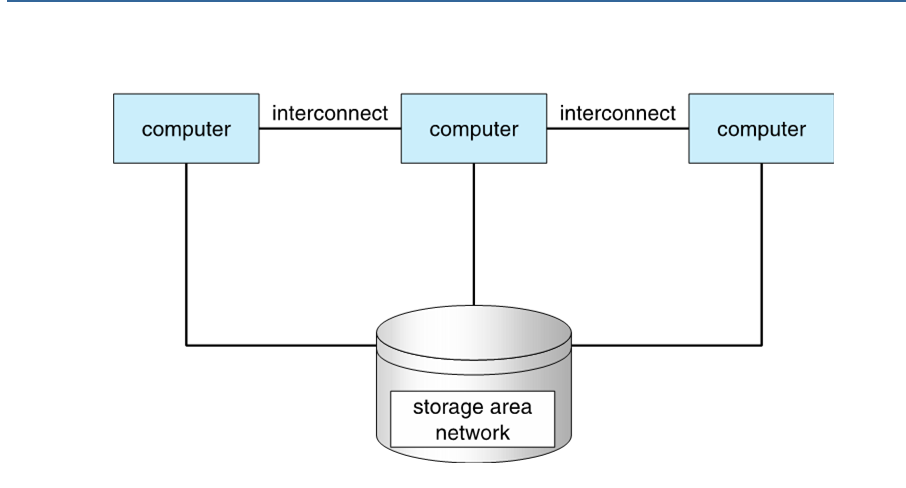
Storage

* Alapok
  + SI mértékegységek -gotta know
  + Alapegység 1 bit 🡪 0 / 1
  + Byte 🡪 8 bit 🡪 Legkisebb tárolási egység legtöbb számítógépen
    - Legtöbb számítógép ennél kisebb dolgot mozgatni sem tud
  + Szó 🡪 Gépi szó
    - Architektúrától függ, egy egységet jelent az adott architektúrán
    - 64 biten 🡪 Egy szó 8 byte
    - 32 biten 🡪 Egy szó 4 byte
    - A gép/CPU nagyrészt szavakban gondolkozik
    - Ezekkel ellentétben a hálózatszolgáltatók bitben dolgoznak 🡪 Ezzel „csalva”, hogy nagyobbaknak tűnjenek a számok
  + Átviteli egysége
    - Hálózat 🡪 Bit
    - Storage 🡪 Byte
    - Processzor 🡪 Gépi szó (architektúra függő, 64 biten 8 byte)
    - I/O rendszer 🡪 Egységek/blokkok 🡪 Fájlrendszertől függ
  + Random órán leírt dolgok még ide
    - 1 int 🡪 4 byte
    - 2 int 🡪 8 byte –> 64 bites processzornak 16 byte, mert 8-nál kisebbet nem tud kezelni
    - Bit 🡪 Byte 🡪 Gépi szó 🡪 I/O blokkok
* Storage Structure
  + Main Memory
    - Egyetlen nagy „adattároló” melyet a CPU közvetlen elér
    - Random Access
    - Volatile 🡪 Áram nélkül elveszik a rajta lévő adat
    - CMOS 🡪 Nem igazán ide tartozik, de nagyjából mégis
  + Secondary storage
    - Non Volatile 🡪 Kikapcsolás után is megmaradnak rajta a dolgok
    - Nagy méretű
    - HDD/SSD 🡪 Disk controller
* Storage Hierarchy
  + Minél nagyobb, annél lassabb
  + Regiszter leggyorsabb, és a legkisebb méretű
* Caching/Gyorsítótárazás
  + CPU sokkal gyorsabb mint egy HDD/SSD vagy az I/O alrendszer, de még a memóriánál is, ezért vezették be a Cachet
  + Átmenetileg „idehozunk” adatot valamelyik lassabb tárolóról
* Közvetlen Memória hozzáférés -DMA
  + Nem a processzoron keresztül mozgassunk adatot eszközről a memóriába, hanem közvetlenül az eszközről a memóriába

Számítógép Architektúrák

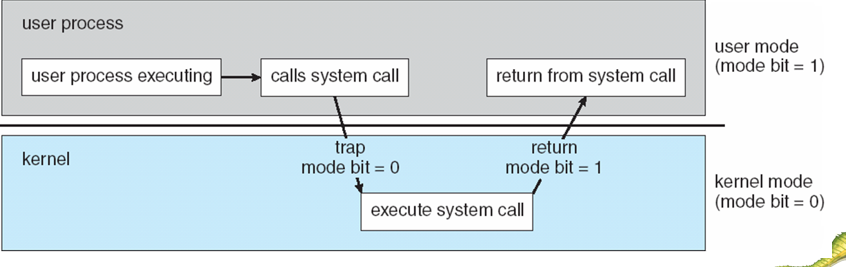
* Processzorok/Multiprocesszorok
* Legtöbb rendszer egy általános célú processzort használ
* Sok rendszernek vannak dedikált special purpose processzorai
* Multiprocesszorok
  + Nagyobb teljesítmény
  + Skálázható
  + Megbízhatóbb
  + Két típusa
    - Aszimmetrikus Multiprocessing
      * A processzorok nem egyenrangúak 🡪 Master and Slave
      * Business piacra jellemző
      * Saját CPU architektúrák
      * Céltevékenységek
    - Szimmetrikus multiprocessing
      * A processzorok egyenrangúak
      * Konzumer piacra jellemző
  + Nem ugyanaz, mint a több mag, fizikailag több processzororról beszélünk
* Multicore processzorok
  + Hoznak teljesítményt, de közel sem lineárisan, és nem annyit mint 2 fizikai processzor



* Clustering
  + Gépek összekötése 🡪 Sokkal gyorsabb, mint egy gép magában
  + Gépek közt optikai összeköttetés
  + Sata 3 🡪 Max 6 Gbit, Optika 🡪 Több mint 10 Gbit
  + SAN 🡪 Storage Area Network

Második előadás vége

Dual mode

* Egy védelmet jelent az operációs rendszernek, és rendszerkomponenseknek
* Dual mode
  + Kernel mode
    - Eszközhozzáférés, file elérése stb.
    - Privilegizált utasítás
  + User mode
  + Módbit 🡪 Egy bit ami azt jelöli éppen Kernel vagy User módban van az OS
  + A két mód közötti váltás rendszerhívásokkal valósul meg
  + A lényege, hogy a user nem fér hozzá sok dologhoz, ezzel védve a usert a saját hülyeségétől, és az OS-t is
  + Ára van 🡪 Sebesség

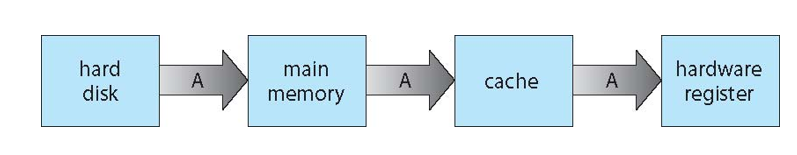
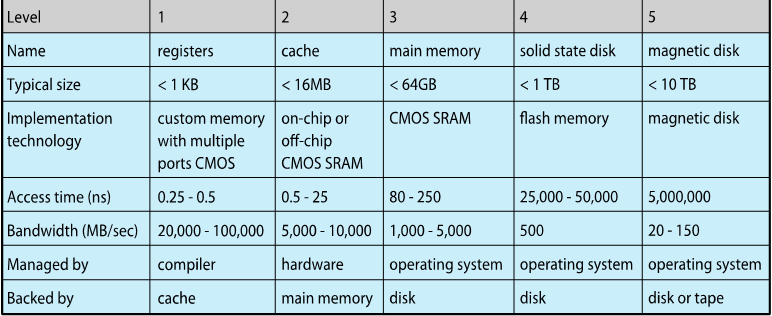
Folyamatkezelés

* A folyamat egy olyan program ami éppen fut
* Program alapból passzív entitás 🡪 Elindul 🡪 Aktív entitás
* Folyamatnak szüksége van
  + CPU, Memory, I/O
    - Ezeket a Kernel végzi
  + Kezdő adat
* Folyamat végezte után fel kell szabadítani minden felszabadítható erőforrást
* Utasításszámláló
  + Betöltött program melyik részén járunk
  + Kezdetben egyszerre egy folyamat megy, és a folyamatokat szekvenciálisan hajtjuk végre
    - Ha van egyszerre több folyamat
      * Közöttük kommunikáció/szinkronizáció megvalósítása
      * Erőforráselosztás több folyamat között
* Folyamatmenedzsment fontos része 🡪 Holtpont /Deadlock elkerülése
  + Két folyamat vár egymásra, ezért soha nem fog előre haladni

Memória Kezelés

* Egy program KIVÉGZÉSÉHEZ a programnak a memóriában kell lennie, a Neumann elvek szerint
* Az összes adatnak, ami a programnak kell, a memóriában kell lennie
* Memória kezelés dönti el
  + Mikor mi van a memóriában
    - Ezzel optimalizálja a CPU kihasználást, és a feedbacket a user felé
* Először fix területeket allokáltak
* Alapvető allokált memória 🡪 Stack/Verem
* Azon túli memória 🡪 Heap/Halom
* Minden allokált memóriát program végeztével fel kell szabadítani
* Az Oprendszer nem mindenhol tud segíteni, ha egy program nem deallokál memóriát az szívás lehet 🡪 Memory Leak

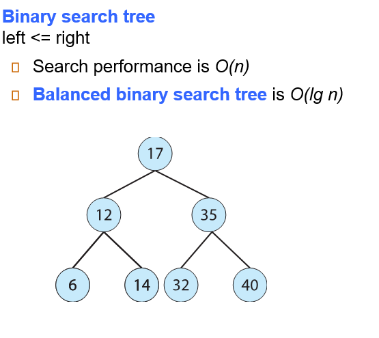
Storage kezelés

* Alapvető logikai egység 🡪 File
* Filekezelés központi kérdés
* Minden fájl a népszerű oprendszereknek
  + Egér, billentyű
  + Folyamat is fájl/mappa
  + Így egyszerűbb programozóknak 🡪 Mindent lehet olvasni, írni bele
* Filerendszer
  + Blokk méretének megadása
  + Másolás, törlés, létrehozás
* Adatmozgatás disk-ről regiszterekbe

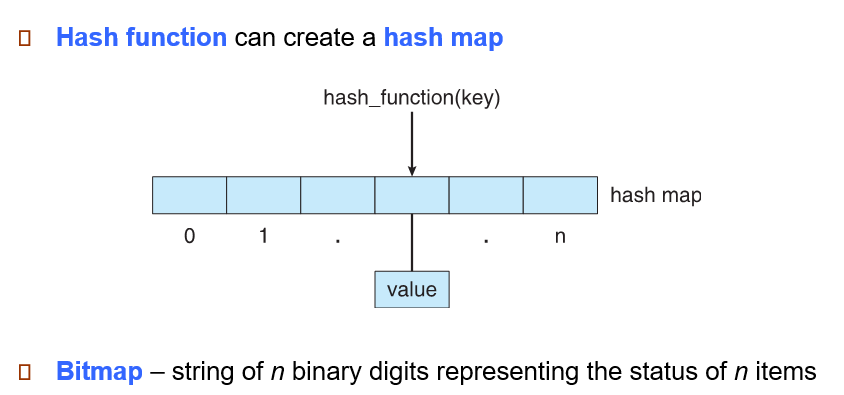
I/O alrendszer feladata

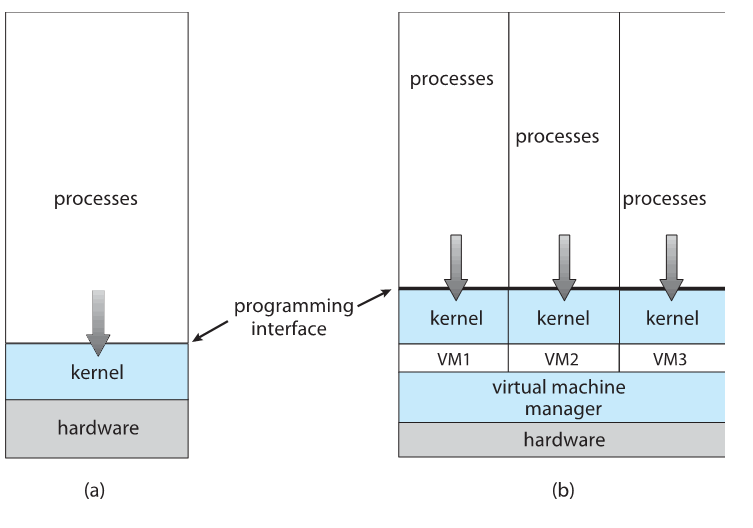
* I/O memory management
* Általános eszközkezelés interface
* Driverek harverekhez

Védelem és biztonság

* Védelem
  + Mechanizmusok, melyek segítségével az erőforráshoz való hozzáférést tudja szabályozni
  + User ID, Group ID, Privilégiumok
  + „Belső védelem”
* Biztonság
  + Kívülről érkező támadás/behatolás elleni védelem

Kernel Adatstrukturák

* Listák
  + Szimpla, Dupla, körkörös
* fák, táblázatok
* Binfa
* Hash Map

Computing EnvironmentsVirtualizáció

* + Erőforráskezelés miatt lett rá igény
  + OS az OS-ben
  + VMM 🡪 Virtual Machine Manager
* Valós idejű rendszerek
  + Minden lépésre jön valami válasz, nincs olyan, hogy „lefagy”
  + Például egy gyártósor, holdraszállás, bármi ilyen nagyon fontos dolog
  + Business szférában vannak jelen

Harmadik előadás fele