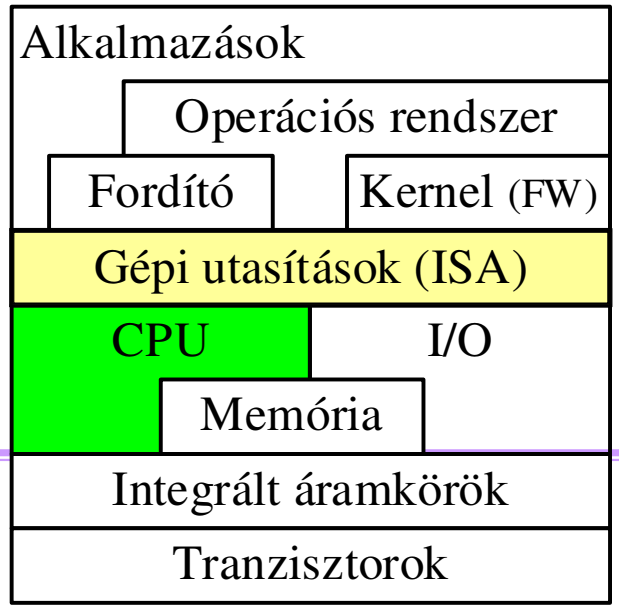
INFORMATIKA I.

BMEVIIIAB08

Többfelhasználós rendszerek támogatása



Multitaszking

- *Egy felhasználói program /alkalmazás/ - sok esetben - a korszerű számítógépek nagyteljesítményű erőforrásainak csak egy részét használja ki*

  Az OR. ilyenkor futtasson olyan másik feladatot /*taszkot* /***task***// is, amely egy éppen szabad erőforrást igényel

  - Nő a rendszer átbocsájtó képessége /*adott idő alatt elvégzett feladatok száma* /
  - Az OR is igényel erőforrásokat a feladat megvalósításhoz → /*ütemezés,stb*/

❖ **Taszk**

- ❑ *Tevékenység, amely a számítógépen más tevékenységgel elvben párhuzamosan folyhat*

Taszk

- *A taszk az algoritmusát megvalósító – utasításokon és kiinduló adatokon, illetve pillanatnyi állapotát leíró környezeten /kontext/ alapuló - program futtatásával realizálódik*
  - Tipikus *taszk lehet pl.: programszerkesztés, forrás file fordítás, szövegszerkesztő, böngésző,MATLAB, stb.*
  - Több taszk is futtathatja ugyanazt a programot

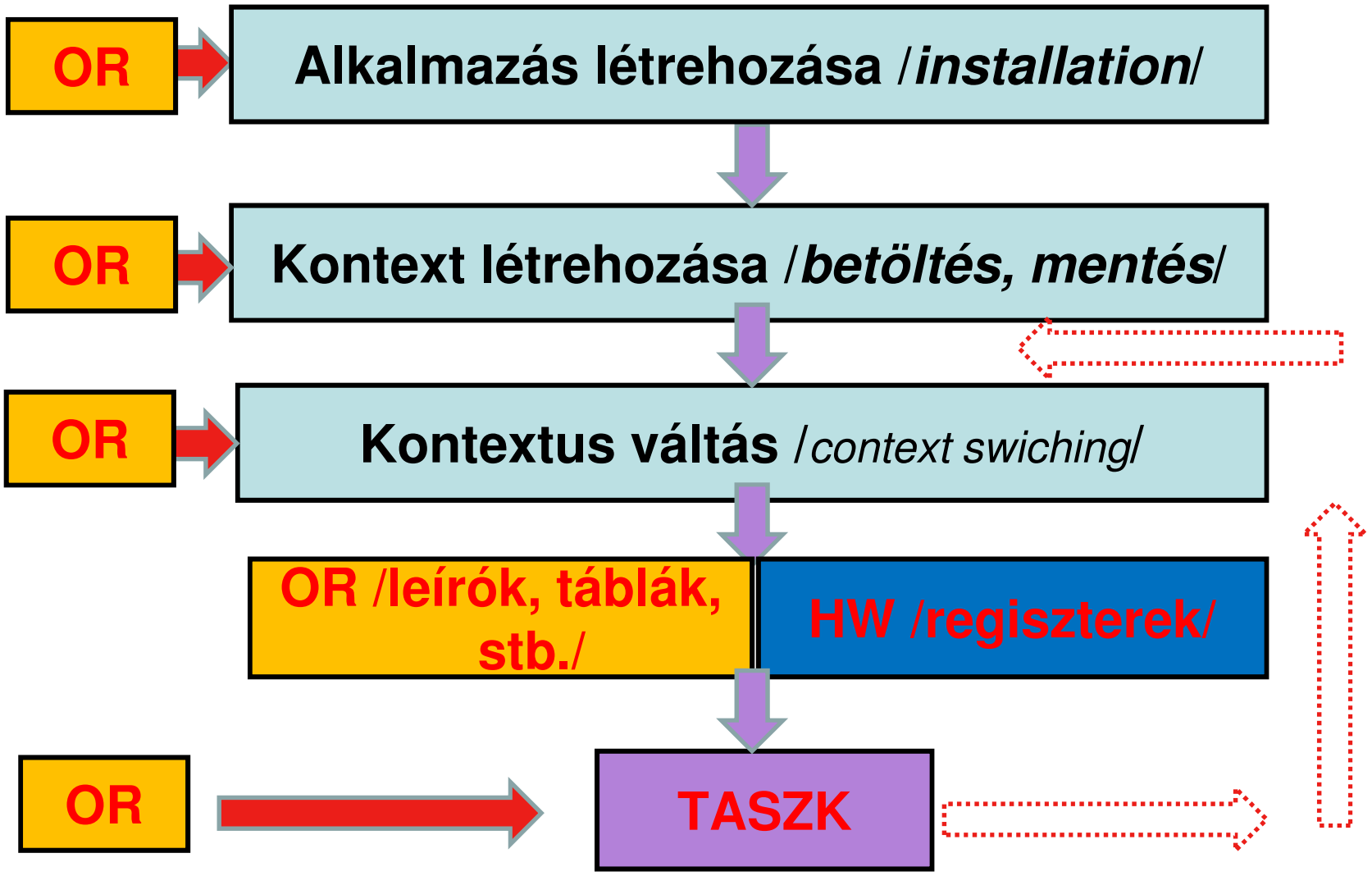
❖ Virtuális processzor

- Az OR a taszk létrehozásakor hozzárendel egy - *a taszkot leíró* - adatszerkezetet /***context***/, amely *pl.: a programot, a kiinduló adatokat, a CPU állapotát /HW/ leíró regisztereket, és a taszk futásához szükséges OR specifikus környezetet írja le*

Virtuális processzor

- Minden taszk-nak /*felhasználónak* / úgy kell éreznie, mintha saját, külön processzora lenne
  /*Virtuális processzor*/
- Látszólag /*virtuálisan*/ egy taszk teljesen párhuzamosan fut a többi taszkkal
  /*kivétel: a másik adatára vár*/
- Valójában ugyanazon a *fizikai processzoron* osztoznak
- Az OR ütemezi futásukat
  /pl.: időosztásos /***time-sharing***/ módszerrel /
  - A felhasználó számára transzparens módon a taszk pillanatnyi állapotát leíró kontextet hozzárendeli az erőforrásokhoz →
    /*fizikai processzor*/

Virtuális – Fizikai processzor összerendelése



Védelem

- Taszkon belül fellépő hibák ellen /*pl.:OR-t*/
- Egyik taszk megvédése a másik hibáitól

❑ Észlelni kell a védelem megsértését *korlátozás*

❑ Hiba hatását a taszk belsejére korlátozni

- Egyszerű megoldás
  - Tároló terület védelme
    pl.: lapszervezésnél a taszkhoz saját laptábla könyvtárat, a laptábla leírókhoz, illetve a lapleírókhoz egyedi védelmi attribútumokat rendelnek /pl.: csak olvasható, írható/olvasható, futtatható, stb.;/
  - A hozzáféréshez privilégium rendelhető
    /pl.: rendszer, illetve felhasználói hozzáférésű, stb./

→A fentiekkel végül is tárolóterületet védünk

/*pedig a feladat algoritmusa, adatai és logikai egységei- védendőek* /

Védelem

- Szegmentálás
  - A feladathoz /*algoritmus*/ rendelt logikai objektumok /*program, adat, verem, rendszer specifikus leírók, stb.;*/ szerint osztják fel /*szegmentálják* / a rendszert és ezeket rendelik a felhasználóhoz /taszkhoz/ 👍👎
  - 👍A szegmensek szétválaszthatók → ***/elkülöníthetők/***
  - 👍Szegmensen belül relatív címzés
    /szegmens kezdete+offszet/
  - 👍A szegmens hossza ismert →/ellenőrzés +védelem/
  - 👍A szegmensekhez rendelt attribútumok hatásos védelmet biztosítanak 👍
    pl.: a kód, stack, adat, szegmensekhez csak rendeltetésszerűen férhetünk hozzá 👍
    kódra nem lehet írni, stack-ből nem lehet utasítást felhozni, stb.; 👍

Védelem

- Fentiek csökkentik a rugalmasságot 👎
  ezt meg kell oldani
  - A szegmenseket a tárolóban tetszőleges helyre helyezhetjük /*ez úgy is kell az eltérő konfigurációk miatt*/ 👍
  - Átmenetileg, vagy véglegesen lehetnek azonos, vagy átfedő szegmensek👍
  - Speciális utasítással /*pl.: prefix*/ eltérhetünk a hozzárendeléstől 👍 /hibalehetőség!!!/ 👎
  - Védelem magasabb-logikai szinttel👍 👍→ *lásd IA-32*