



Óbudai Egyetem  
Alba Regia Kar  
Mérnöki Intézet

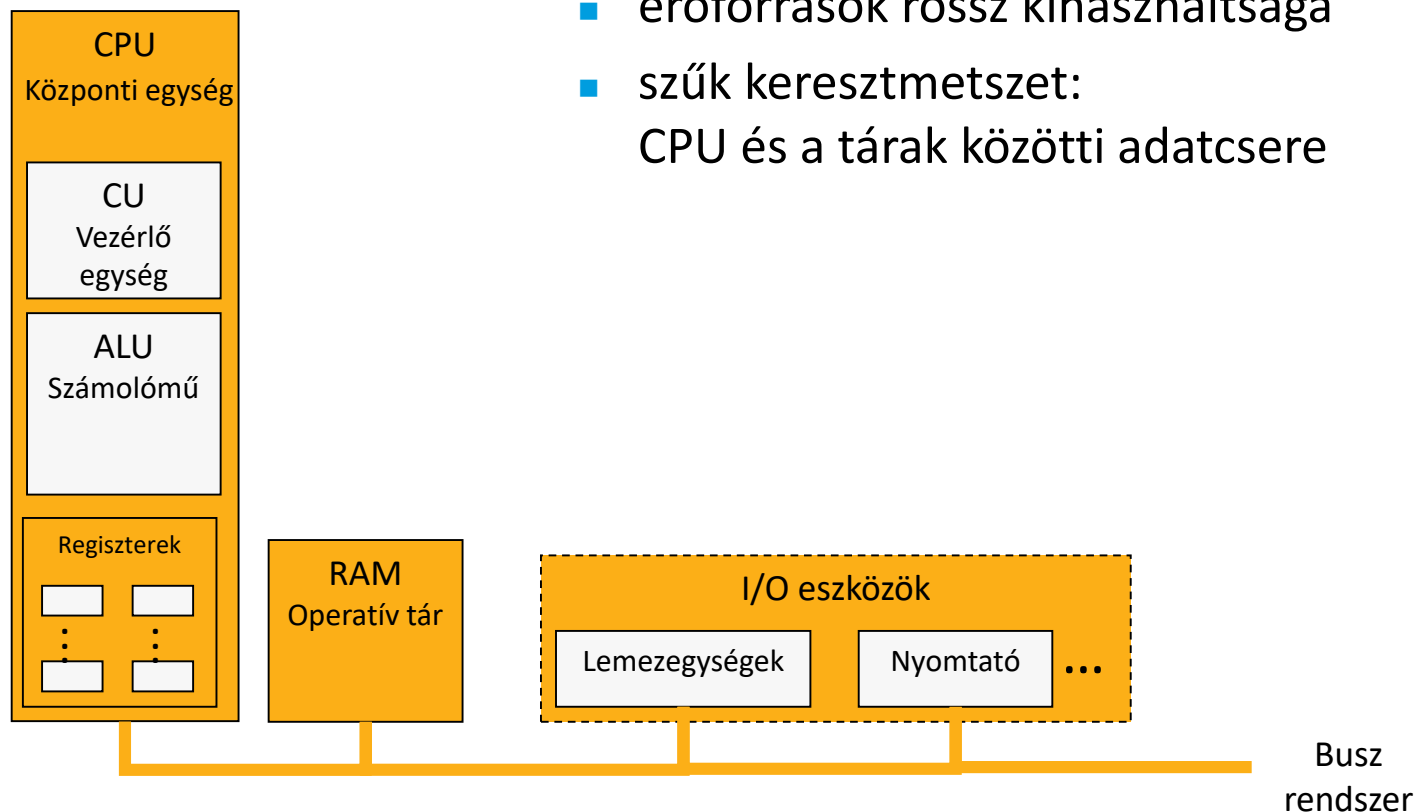
# Párhuzamos és átlapolt feldolgozás

**Dr. Seebauer Márta**  
egyetemi docens

[seebauer.marta@uni-obuda.hu](mailto:seebauer.marta@uni-obuda.hu)

# Neumann-típusú számítógép általános felépítése és korlátai

- egy időben egy program soros végrehajtása
- erőforrások rossz kihasználtsága
- szűk keresztmetszet:  
CPU és a tárok közötti adatcsere



**Extenzív** módszer – paraméterek változtatása

- órajel frekvencia növelése
- processzor szószélességének növelése
- processzor címtartományának növelése
- regisztertér növelése

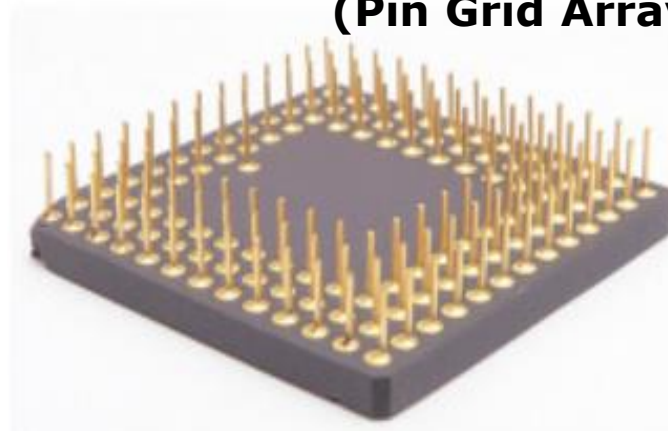
**Intenzív** módszer – architektúrális változtatás

- átlapolás – meglévő erőforrások jobb kihasználása
- párhuzamosítás – erőforrások többszörözése
  - adatpárhuzamosság
  - funkcionális párhuzamosság

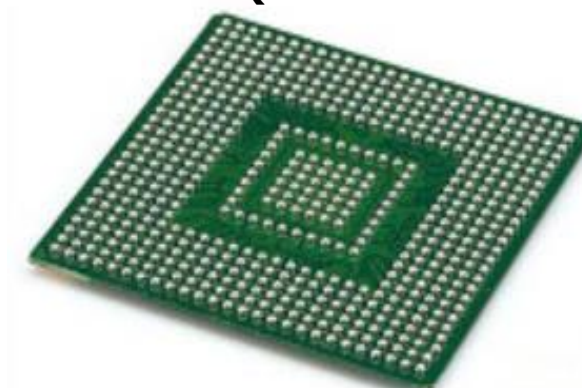
**LGA**  
(Land Grid Array)



**PGA**  
(Pin Grid Array)



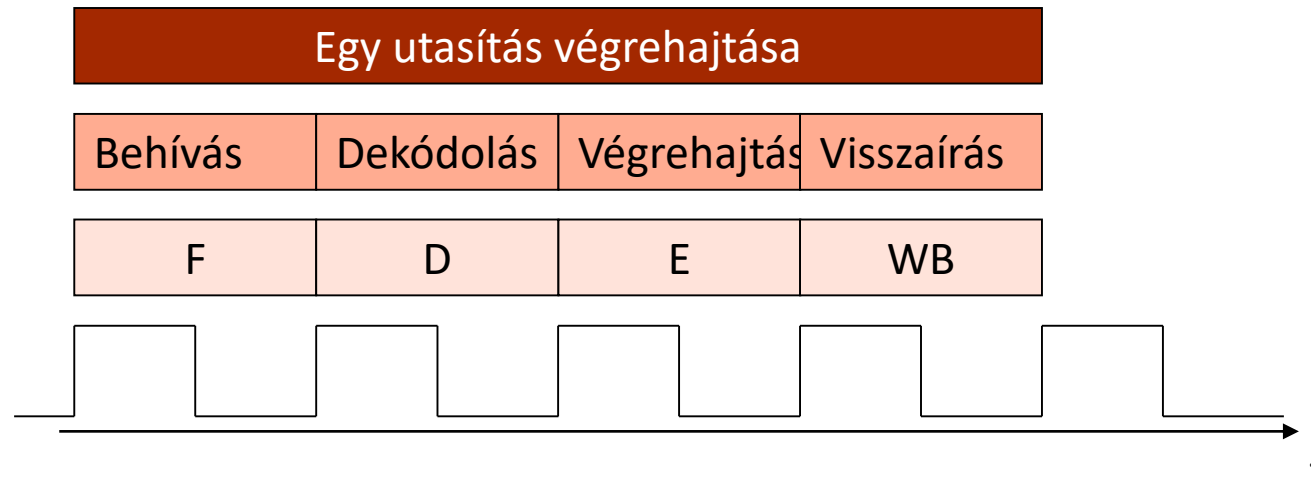
**BGA**  
(Ball Grid Array)



# Átlapolt végrehajtás - futószalag technika

A futószalag a az átlapolt végrehajtás megvalósításának eszköze. Alapfutószalagnak nevezzük az ideális, elvi futószalagot, amelynél

- minden végrehajtandó feladatot **részfeladatok sorozatára** kell bontani
- minden részfeladathoz **külön futószalag** fokozat tartozik
- az egyes részfeladatok végrehajtása **azonos ideig** tart
- minden fokozat **bemenete** az előző fokozat **kimenete**, és kimenete a következő fokozat bemenete
- **szinkron** működésű



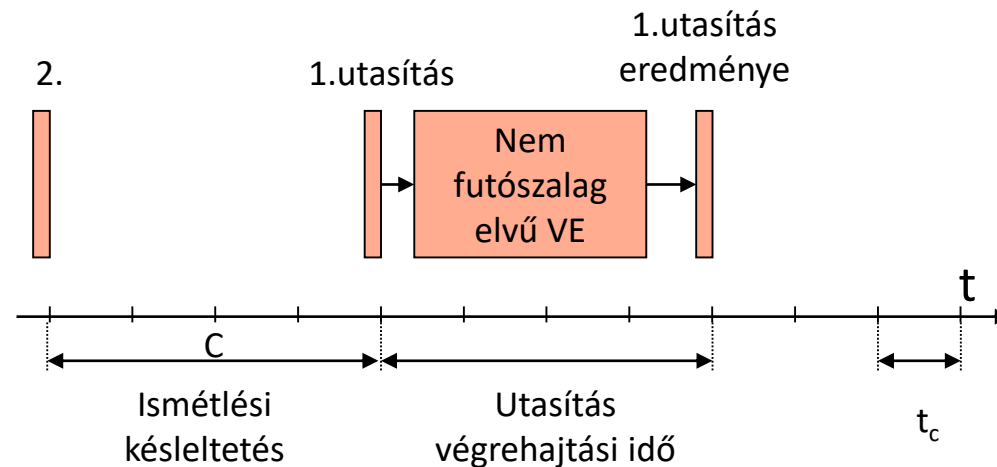
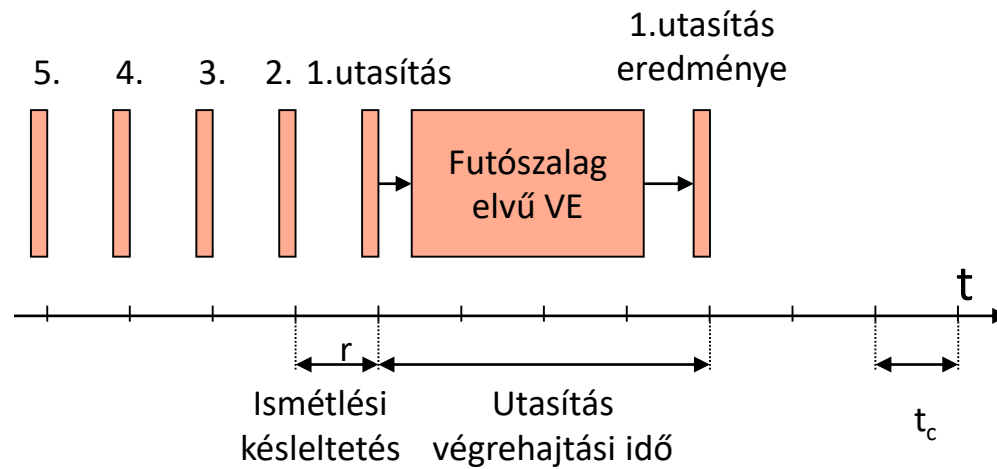
# Utasítássorozat feldolgozása

Alapfutószalag felhasználásával		F	D	E	WB	
Ciklus	Bemenet	Feldolgozás alatt				Kimenet
1. ciklus	1. utasítás	F1				
2. ciklus	2. utasítás	F2	D1			
3. ciklus	3. utasítás	F3	D2	E1		
4. ciklus	4. utasítás	F4	D3	E2	WB1	1. utasítás
5. ciklus	5. utasítás	F5	D4	E3	WB2	2. utasítás

Skalár feldolgozással		F	D	E	WB	
Ciklus	Bemenet	Feldolgozás alatt				Kimenet
1. ciklus	1. utasítás	F1				
2. ciklus			D1			
3. ciklus				E1		
4. ciklus					WB1	1. utasítás
5. ciklus	2. utasítás	F2				

# Ismétlési késleltetés

- Egy adott futószalag esetén az a legrövidebb időtartam, amely letelte után a futószalag meg tudja kezdeni egy újabb utasítás végrehajtását. Az alapfutószalag ismétlési értéke egy ciklus.
- Bonyolult műveletek végrehajtása (lebegőpontos szorzás) a futószalag egy-egy fokozatának egyszeri vagy többszöri ismétlését igényli. Ezeknél a műveleteknél a futószalag ismétlési késleltetése kettő vagy több ciklus.
- Az ismétlési késleltetés az a mérőszám, amely a futószalag teljesítőképességét leginkább meghatározza.



# Teljesítmény értékelése

- Nem futószalagelvű processzorok teljesítménye:
  - Futószalagelvű processzorok az utasítások végrehajtási ideje jelentőségét veszti, függ
    - a processzor ciklusidejétől - a leghosszabb feldolgozási idejű fokozat legkedvezőtlenebb feldolgozási ideje
    - az utasítások
      - műveleti
      - behívási
      - ismétlési
- késleltetésétől

$$R = \frac{f}{C_{eff}} \quad f = \frac{1}{T}$$

$$R = \frac{1}{r \cdot t_c} \quad t_c = \frac{T}{N}$$

ahol

f: órajel frekvencia

$C_{eff}$ : egy utasítás átlagos ciklusszáma

T: az utasítás végrehajtási ideje

N: a futószalag fokozatok száma

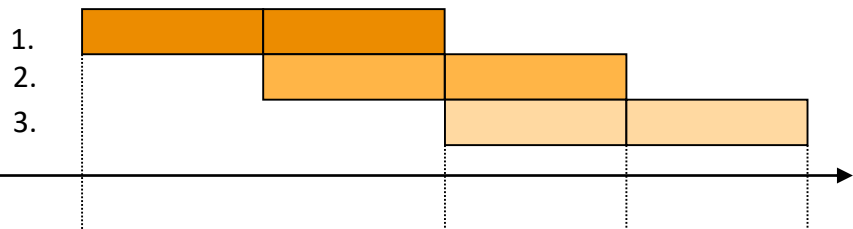
r: a futószalag ismétlési késleltetése ciklusokban

$t_c$ : a futószalag ciklusideje

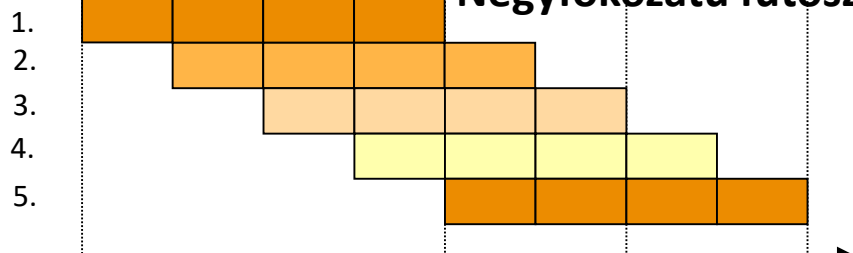
# Futószalag fokozatok száma

Utasítások

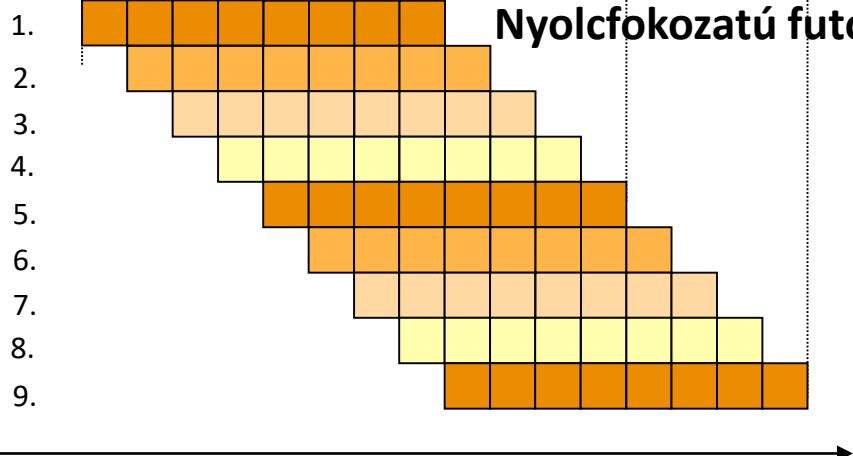
Kétfokozatú futószalag



Négyfokozatú futószalag

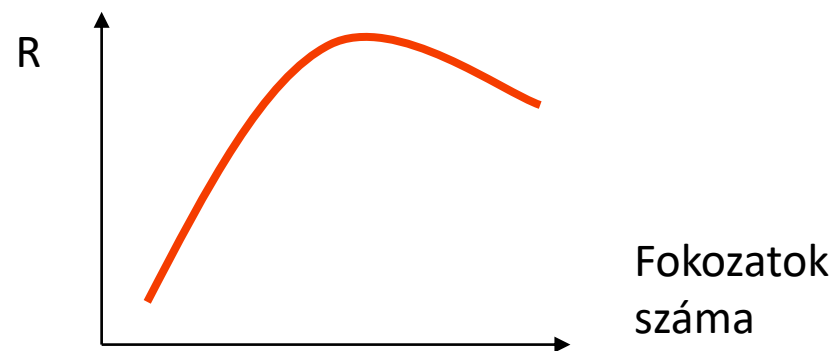


Nyolcfokozatú futószalag



Több fokozat használata esetén

- nagyobb mérvű párhuzamosság érhető el, ami nagyobb teljesítményt eredményez  
A fokozatok számának növelése miatt gyakoribbak lesznek a végrehajtási ablakban szereplő utasítások közötti **adat- és vezérlőfüggőségek**, ami teljesítménycsökkenéshez vezet.
- a végrehajtandó feladatot kisebb részfeladatokra kell bontani.  
A részfeladatok diszkrét jellege miatt az egyes részfeladatok **nem** lesznek **azonos idő** alatt végrehajthatóak. Az órajel frekvenciát a leghosszabb részidőhöz kell igazítani

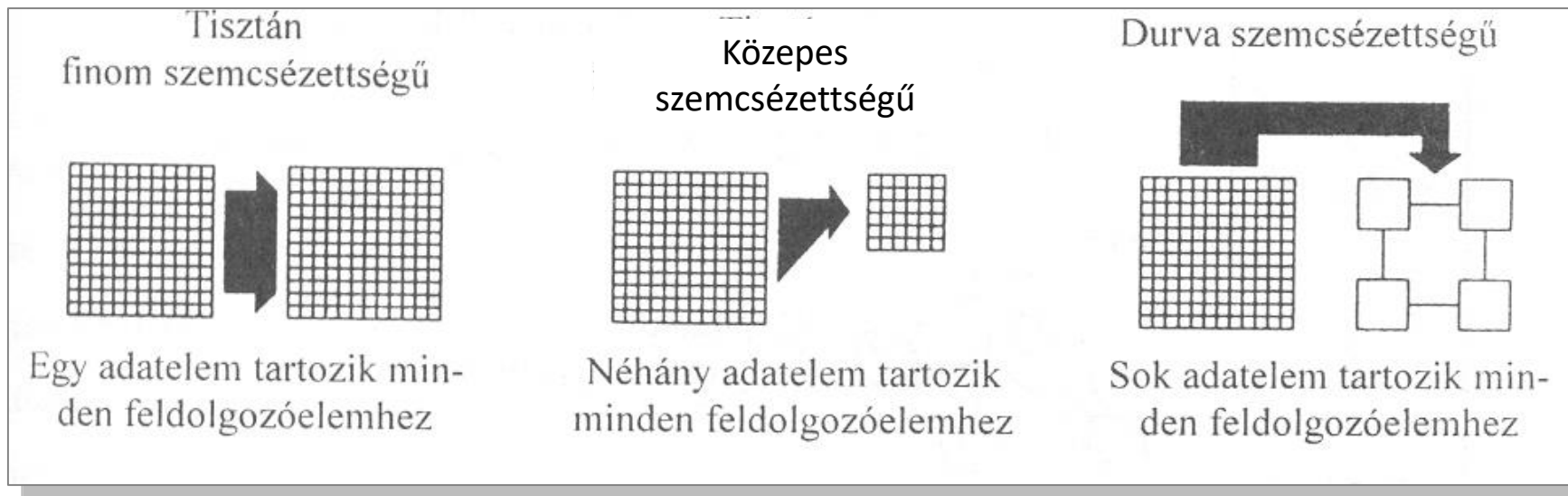




# A párhuzamosság típusai és szintjei

- Párhuzamosság
  - rendelkezésre álló párhuzamosság  
a programokban rejlő, de nem feltétlenül kihasznált
  - hasznosított párhuzamosság
- Rendelkezésre álló párhuzamosság
  - funkcionális párhuzamosság  
a feladatmegoldás logikájából következő párhuzamosság
  - adatpárhuzamosság  
olyan adatszerkezet használatából adódik, amelyek elemein a feladatmegoldás során a műveletek párhuzamosan elvégezhetők

# Adatpárhuzamosság



# CPU és GPU



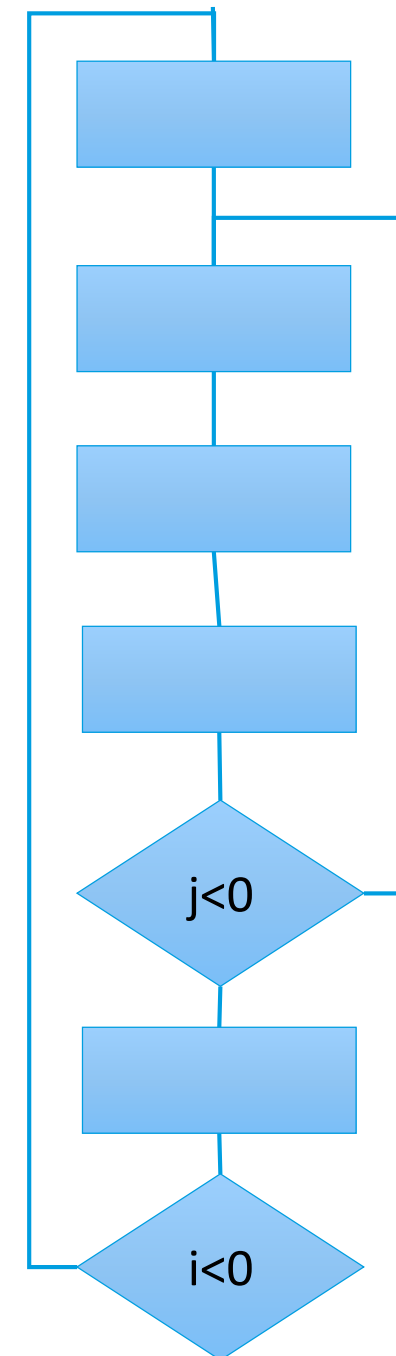
A memória elérés a CPU-nál mondhatni a leglassabb művelet. A cache elérése gyors, viszont mérete csak pár MB, és nagyon drága. GPU-nál a gyors memória-elérés megvalósítása jóval olcsóbb és gazdaságosabb.

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

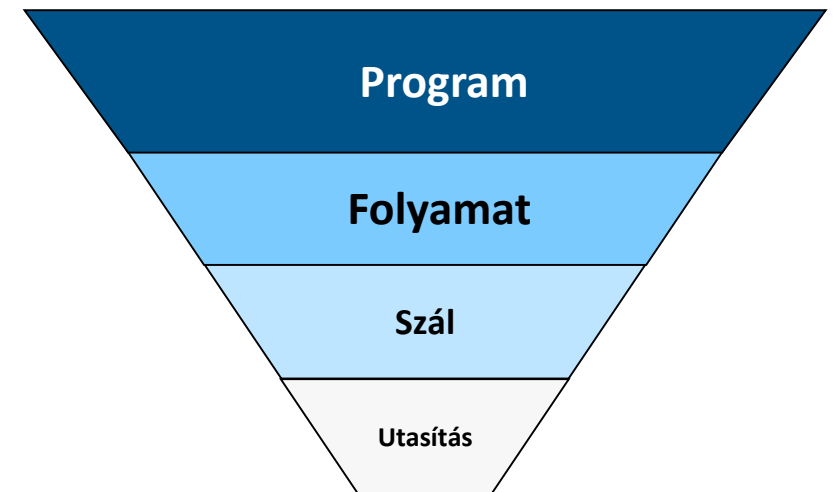
[https://www.youtube.com/watch?v=oi\\_9fujflhl](https://www.youtube.com/watch?v=oi_9fujflhl)

# Az adatpárhuzamosság hasznosítása

- párhuzamos vagy futószalag művelet-végrehajtás
- az adatpárhuzamosság átalakítása funkcionális párhuzamossággá
- adatelemeken végrehajtható műveletek ciklusokba szervezése



- **Program** az utasítások rendezett halmaza, végrehajtható állomány, amely a háttértárolón helyezkedik el.
- **Folyamat (process vagy task)** a végrehajtandó feladat, amellyel az operációs rendszer mint egységnyi megbízással foglalkozik, hozzárendeli a végrehajtáshoz szükséges erőforrásokat (memóriatartomány, processzoridő)
- **Szál (thread)** kisebb, önálló egységként kezelhető utasítássorozat a folyamaton belül, amely más szálakkal párhuzamosan vagy konkurensen hajtható végre. Egyazon folyamathoz tartozó szálak ugyanazokon az erőforrásokon osztoznak. Finomabban szemcsézett, mint a folyamat.
- **Utasítás (instruction)** egy adott szinten értelmezett művelet kódja

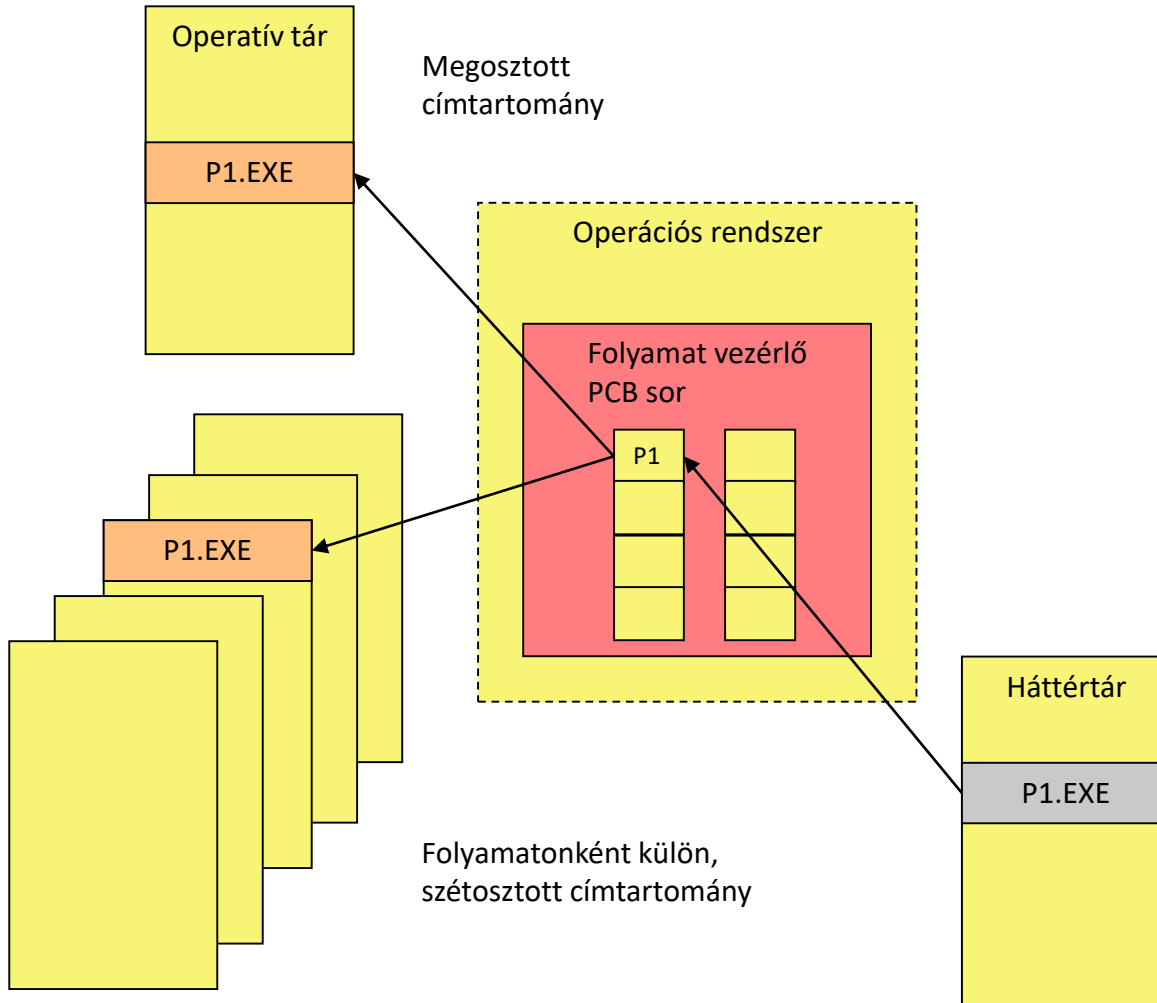


# A párhuzamosság mértéke

## Szemcsézettség

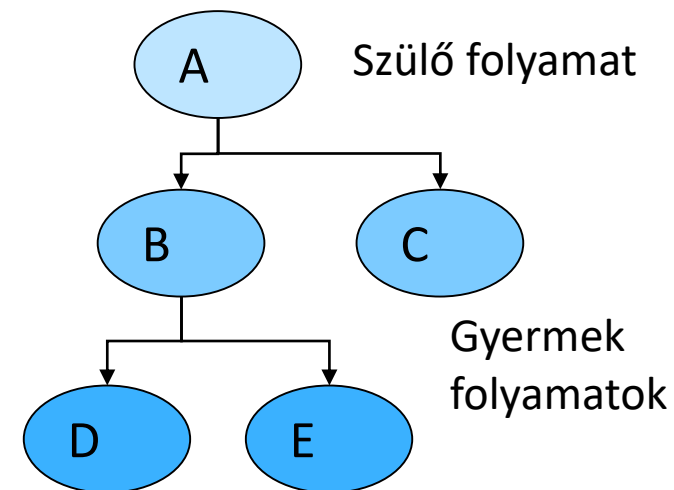
- **Durva szemcsézettség** – az adott program túlnyomó részének végrehajtása a programrészek közötti kevés kommunikációval vagy teljesen kommunikáció nélkül  
pl. **banki tranzakciós feldolgozó rendszerek**
- **Közepes szemcsézettség** – egy program egyes részeinek párhuzamos végrehajtása, ciklusszint vagy eljárásszint  
**funkcionálisan párhuzamos programok**
- **Finom szemcsézettség** – a program egészének utasításszinten párhuzamos végrehajtása,  
**futószalag és adatpárhuzamos programok**

# Folyamat létrehozása



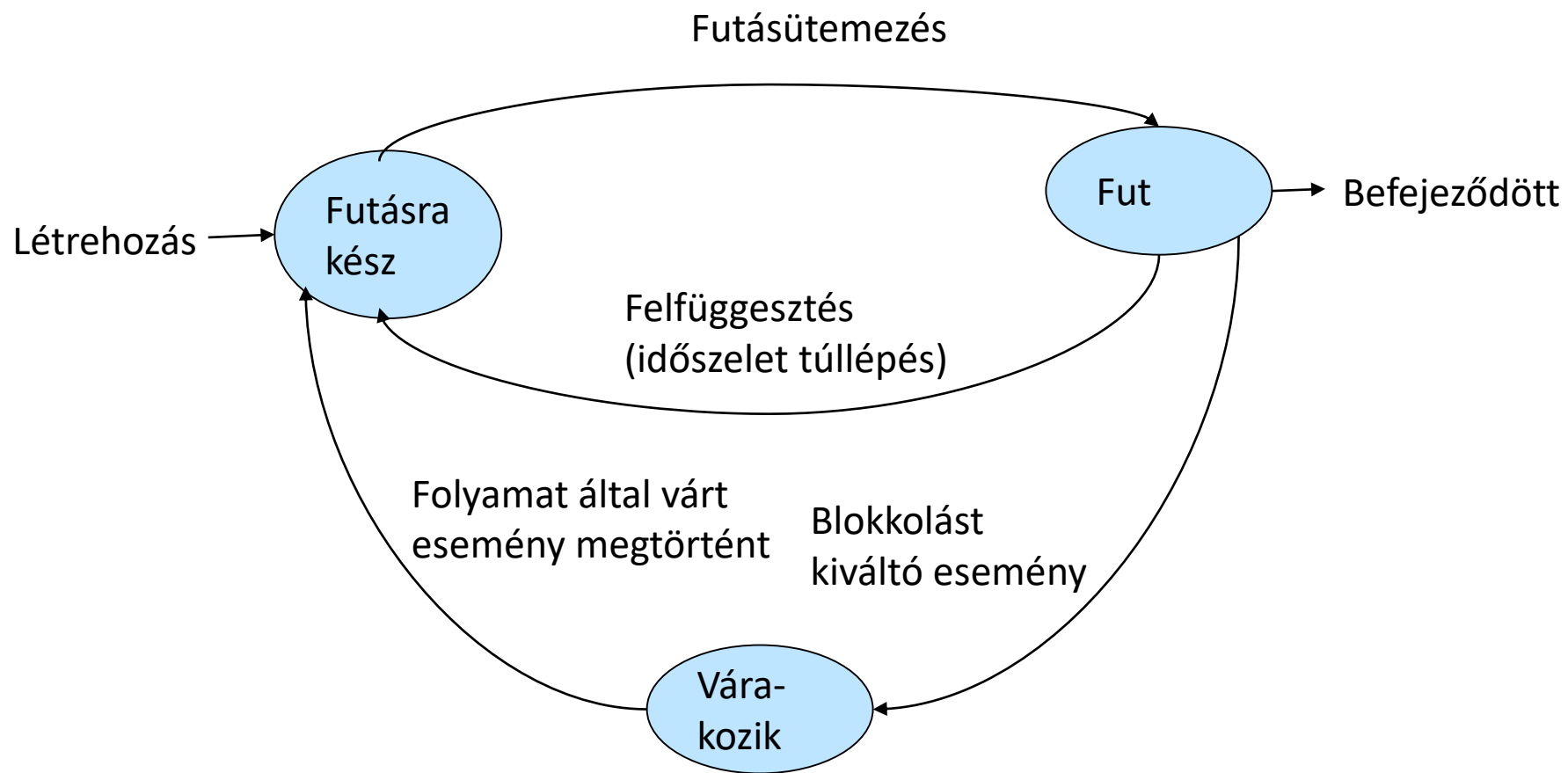
- **PCB (Process Control Block)** folyamatleíró tábla, amely tartalmazza a folyamat életciklusa alatt szükséges információkat
  - folyamat azonosítója
  - tulajdonosa
  - állapota
  - hozzárendelt címtartomány
  - implementációfüggő, folyamat specifikus információ, ami a memóriakezeléshez vagy az ütemezéshez szükséges

- A folyamatok életciklusa
  - folyamat létrehozása
    - folyamatleírás összeállítása
    - a címtartomány kijelölése és hozzárendelése
    - a program betöltése a kijelölt címterületre
    - a folyamatleírás átadása az ütemezőnek
  - folyamat végrehajtása
    - ütemezése az ütemezési politikának megfelelően
    - futásra kész állapot
    - futási állapot
    - várakozó vagy blokkolt állapot
  - folyamat befejezése



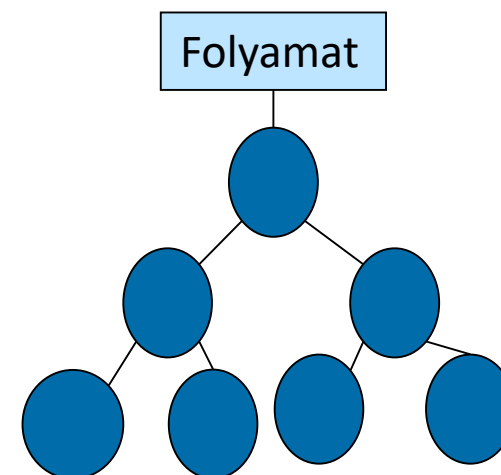


# Folyamat modell

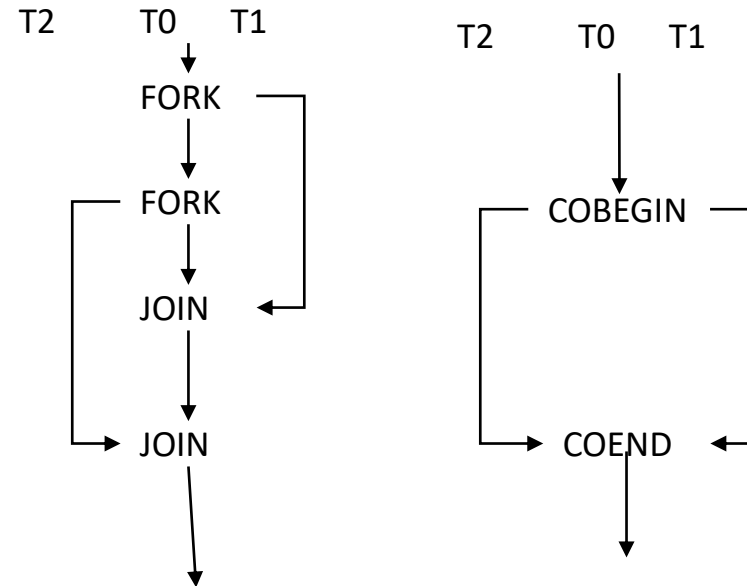


# Folyamatszál alapú modell

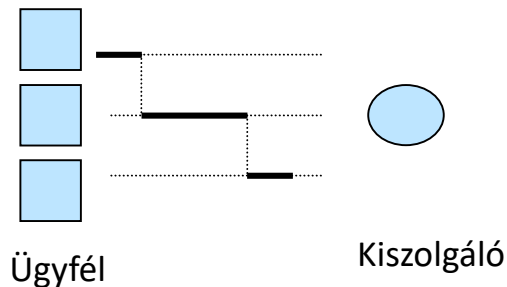
- **TCB** (Thread Controll Block) szálvezérlő blokk
  - finomabb szemcsézettsége révén a szálak használata nagyobb fokú párhuzamosítást tesznek lehetővé
  - a szálak létrehozása, a szálak közötti kommunikáció, a szálcserék kevésbé időigényes műveletek, mert a folyamaton belül ugyanazon erőforrásokat használják
  - a szálak alapállapota
    - futásra kész
    - futó
    - várakozó vagy blokkolt állapot
  - kiegészítő állapotok egyes operációs rendszerekben
    - inicializált
    - befejezett
    - félreállított
    - átmeneti (speciális erőforrásra várakozik)
- Szálkezelés
  - ütemezés az ütemezési politika szerint
  - állapotátmenetek kezelése az állapot-átmenet diagramnak megfelelően



- Szálak létrehozása és befejezése
  - nem szinkronizált létrehozás és befejezés
    - CREATE\_PROCESS, START\_PROCESS, CREATE\_THREAD, START\_THREAD
  - nem szinkronizált létrehozás és szinkronizált befejezés
    - FORK, JOIN
  - szinkronizált létrehozás és befejezés
    - COBEGIN, COEND



# Konkurens és párhuzamos végrehajtás

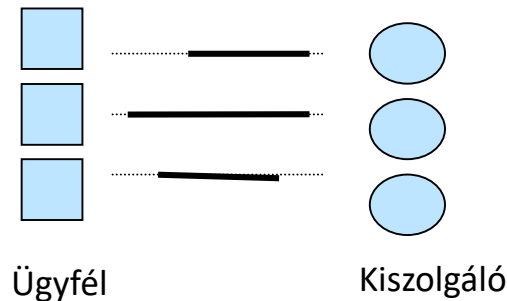


## Konkurens modell

„n ügyfél 1 kiszolgáló”

a kiszolgáló minden időpillanatban egy ügyfelet szolgál ki – időosztásos modell

- Megszakítási szabály
- Kiválasztási szabály



## Párhuzamos modell

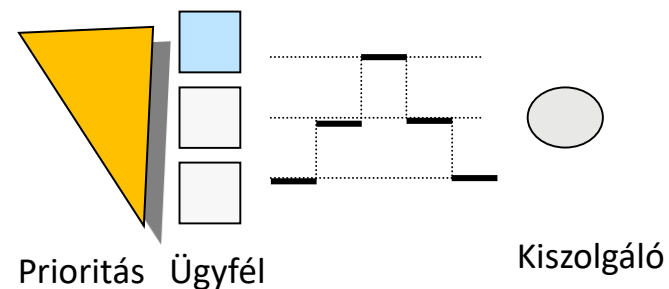
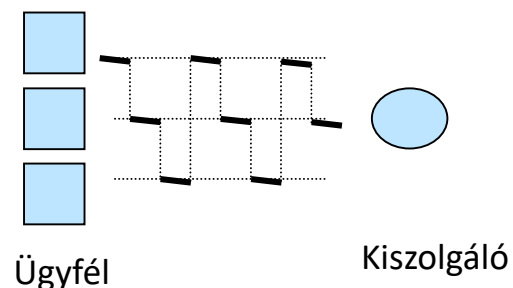
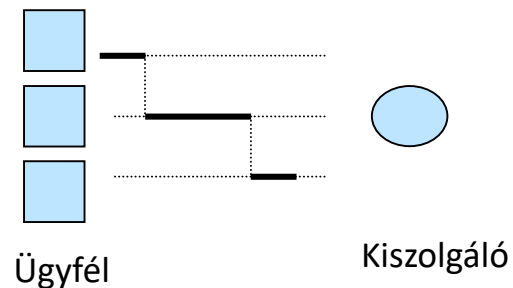
„n ügyfél n kiszolgáló”

a végrehajtott folyamatok összehangolása lehet

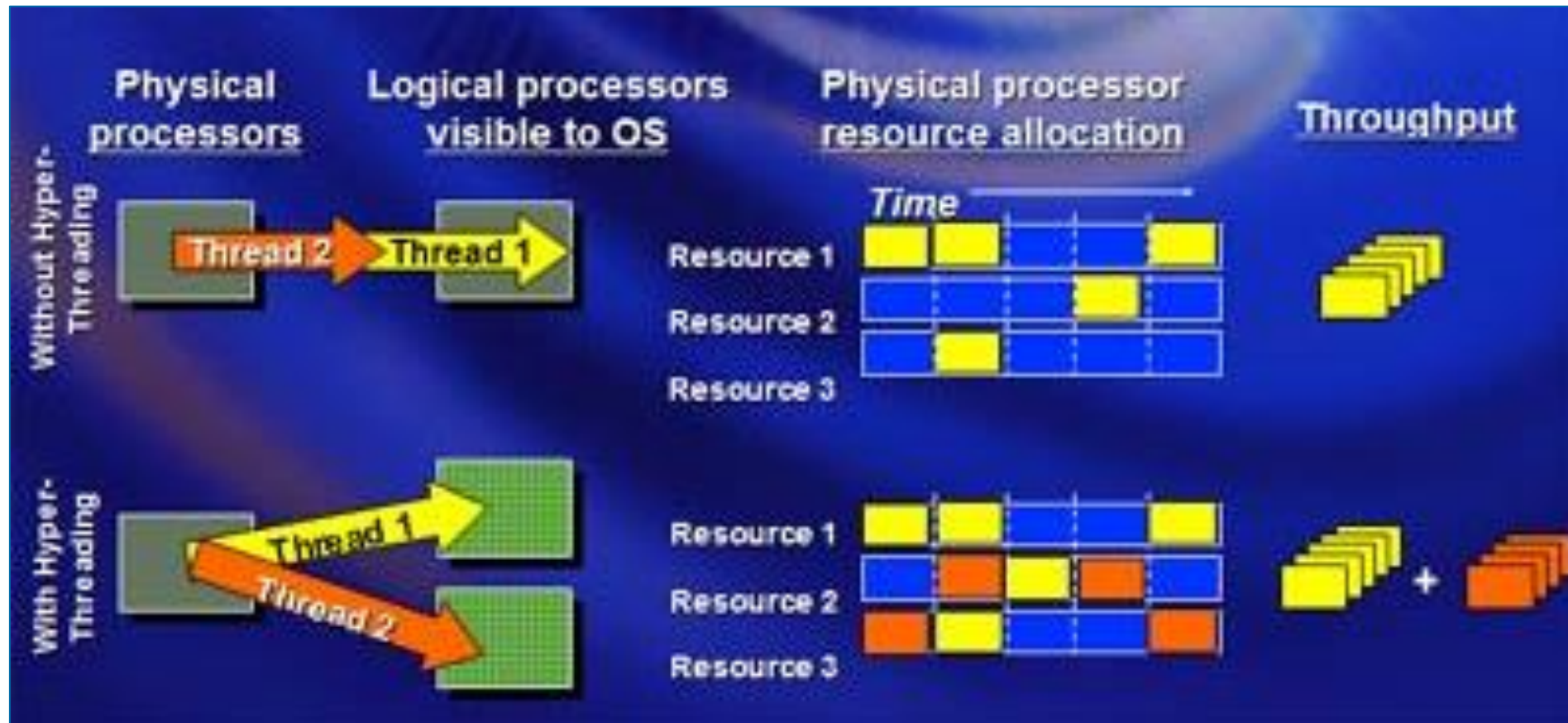
- szinkron – minden kiszolgáló ugyanabban a pillanatban kezdi meg a kiszolgálást
- aszinkron – a kiszolgálók működése nincs összehangolva

# Konkurens ütemezési politika

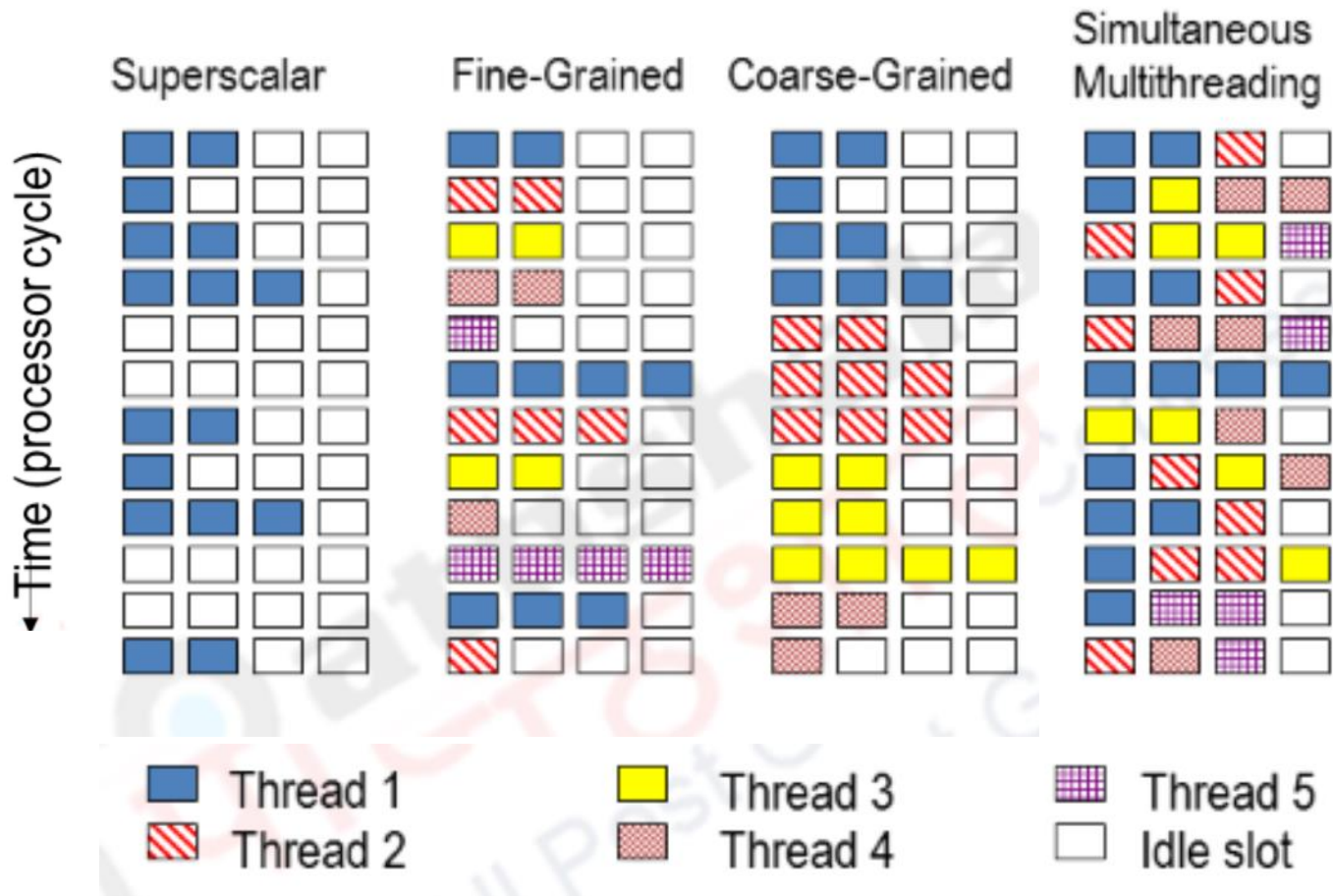
- Cél
  - hatékony kiszolgálás
  - nagyobb áteresztőképesség
  - kisebb válaszidő
- Megszakítási szabály
  - az aktuális kiszolgálás nem szakítható meg
  - az aktuális kiszolgálás megszakítható
    - időosztásos
    - prioritásos
- Kiválasztási szabály
 bizonyos jellemzők (prioritás, beérkezés időpontja, stb) alapján dönt a kiszolgálás sorrendjéről
  - prioritási érték kiszámítása
  - a legmagasabb prioritású ügyfél kiszolgálása



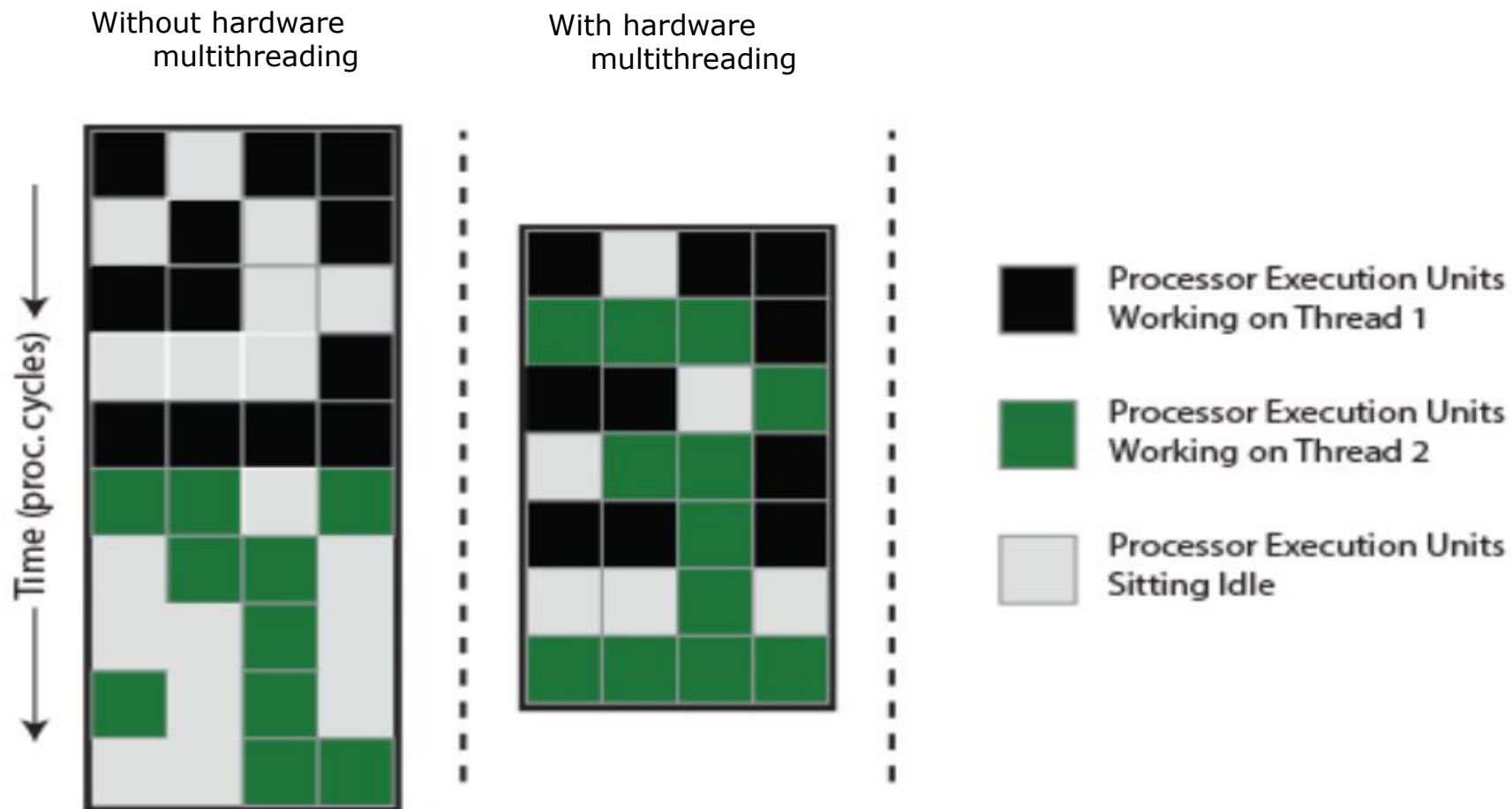
# A processzor erőforrásainak megosztása



# Többszálas végrehajtási modellek



# Többszálal végrehajtás hardveres támogatása





# Programozási nyelvek osztályozása

Nyelvek	Egy ügyfél egy kiszolgáló	n ügyfél egy kiszolgáló	Egy ügyfél n kiszolgáló	n ügyfél n kiszolgáló
Soros				
Konkurens				
Adatpárhuzamos				
Párhuzamos				

- **Tradicionális, soros programnyelvek**

nem rendelkeznek n ügyfél kiszolgálására alkalmas szerkezetekkel

- Fortran
- Pascal
- C
- LISP
- Prolog

- **Konkurens programnyelvek**

konkurens folyamatok és szálak deklarálásával alkalmasak n ügyfél és egy kiszolgáló modell megvalósítására, de nem tudnak n kiszolgálót kezelni

- ADA
- Concurrent Pascal
- Modula-2
- Concurrent Prolog

## Adatpárhuzamos programnyelvek

speciális adatszerkezeteket vezetnek be, amelyek az adatelemek szintjén párhuzamos feldolgozást tesznek lehetővé

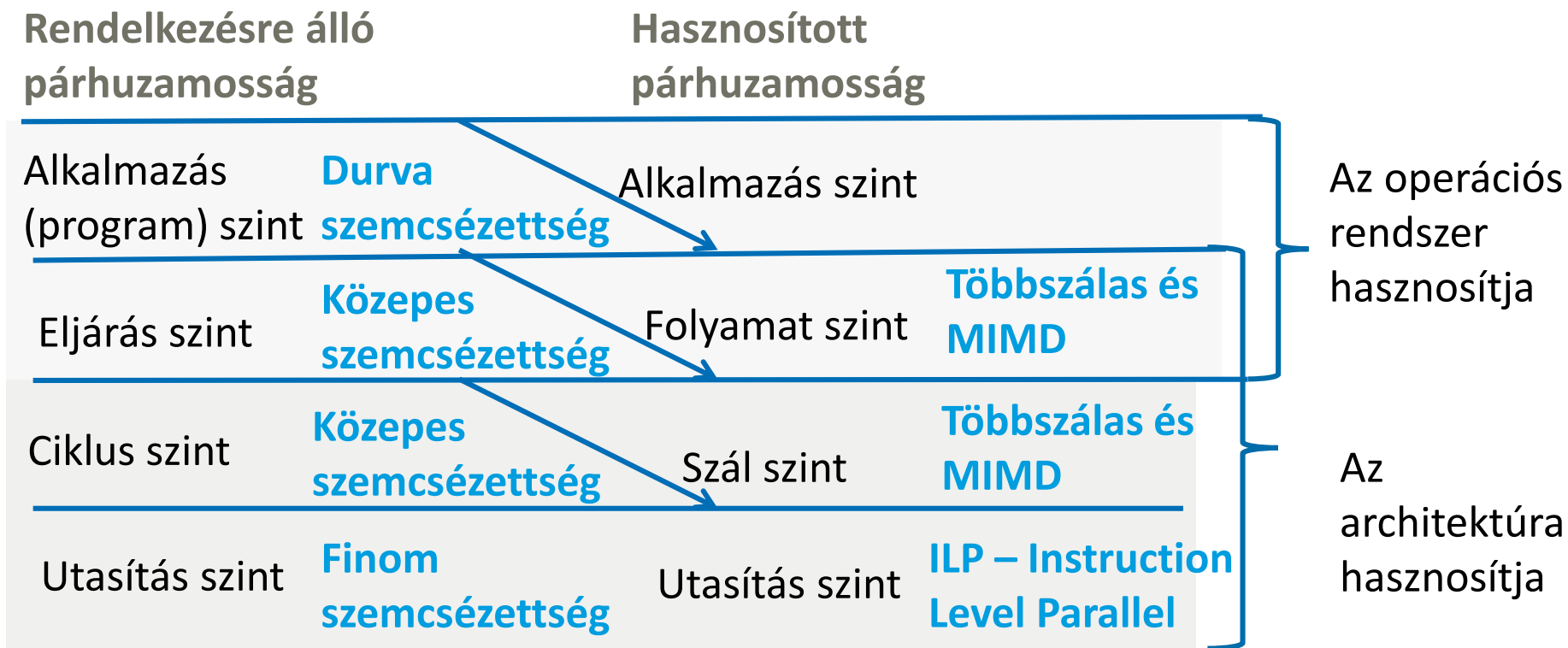
- High Performance Fortran
- DAP Fortran
- DAP Prolog
- Connection Machine LISP

## Párhuzamos programnyelvek

Megfelelő nyelvi eszközökkel lehetővé teszik az n ügyfél n kiszolgáló modellnek megfelelő processzor hozzárendelését

- Parallel C
- Occam
- Strand-88

# Funkcionális párhuzamosság szintjei

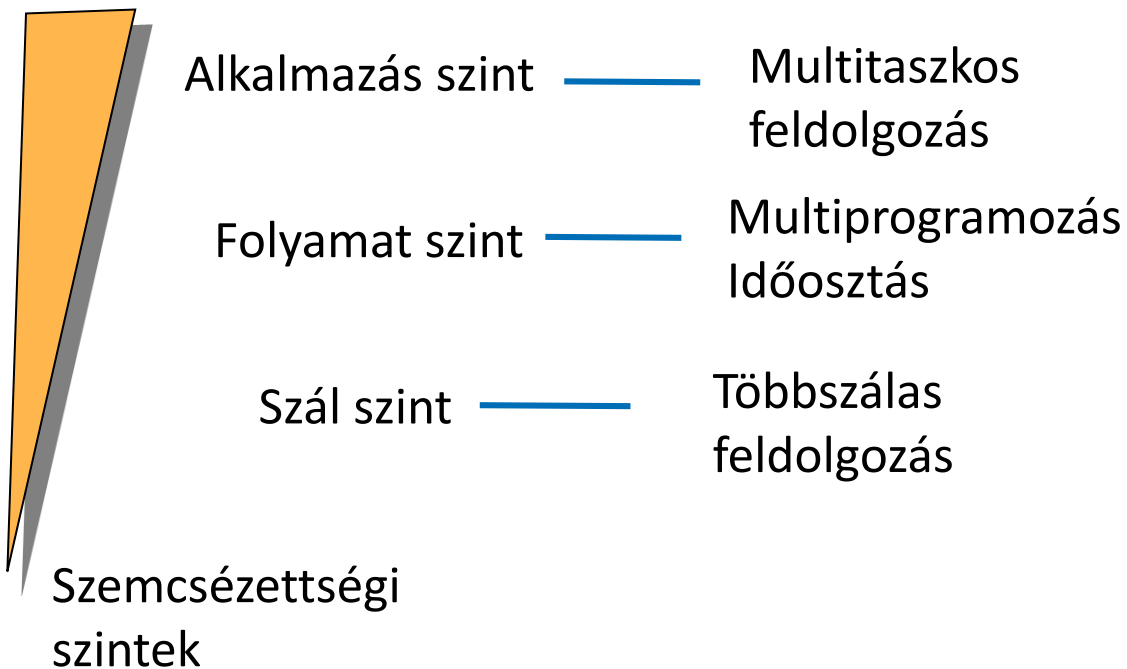


# Párhuzamosság hasznosítása

- **Cél** a számítások felgyorsítása
- Implementációs szintek
  - architektúra
  - párhuzamosságra optimalizáló fordítóprogramok
  - többfeladatos vagy többszálas operációs rendszer
  - speciális nyelvi szerkezetek a programnyelvekben
- Rendelkezésre álló párhuzamosság hasznosítása
  - **utasítások** párhuzamos végrehajtása
    - **ILP** (Instruction Level Parallel) architektúrák
  - **szál** és/vagy **folyamat** szinten a tárgykód legkisebb önállóan végrehajtható részfadatai
    - **többszálas** és MIMD architektúrák
    - **konkurens** végrehajtás

# Konkurens végrehajtási modellek

Egyetlen processzoron (SISD) az operációs rendszer támogatásával valósítják meg.



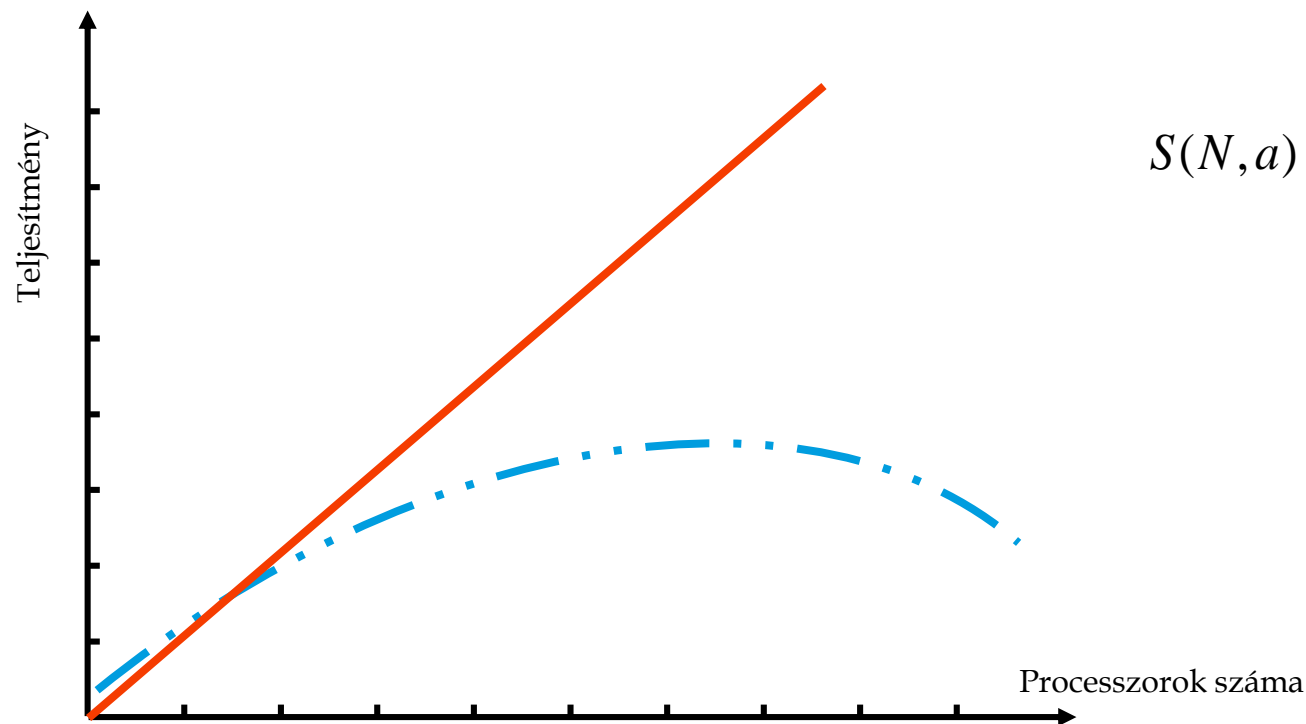
Felhasználói szintű konkurens végrehajtás. Az operációs rendszer által létrehozott folyamatok **különböző felhasználókhoz** tartoznak. Cél

- multiprogramozott futtatás – a processzor idejének hatékony kihasználása
- időosztásos futtatás – minden felhasználó erőforráshoz juttatása

Minden jelenleg használt operációs rendszer és a MIMD architektúrák támogatják

Minden folyamathoz több szál tartozhat. A processzor ütemezése a folyamat-szál modell alapján egyes operációs rendszerek (Windows NT, OS/2, Sun OS 5.0) és a többszálás architektúrák támogatják

# A párhuzamos rendszerek teljesítménynövekedése



$$S(N, a) = \frac{t_s}{t_p}$$

$$S(N, a) = \frac{1}{a + \frac{(1-a)}{N}}$$

$$S(N, a) = \frac{N}{a(N-1) + 1}$$

Amdahl törvénye

$S$  – a sebességnövekedés mértéke (speed-up)

$a$  – a program soros részének aránya

$N$  – a párhuzamosan működő processzorok száma

