Hardver alapok

(VIIIBA01)

Multiprogramozott és több processzoros rendszerek

Pilászy György gpilaszy@iit.bme.hu

2018/19. tanév őszi félév

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) 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

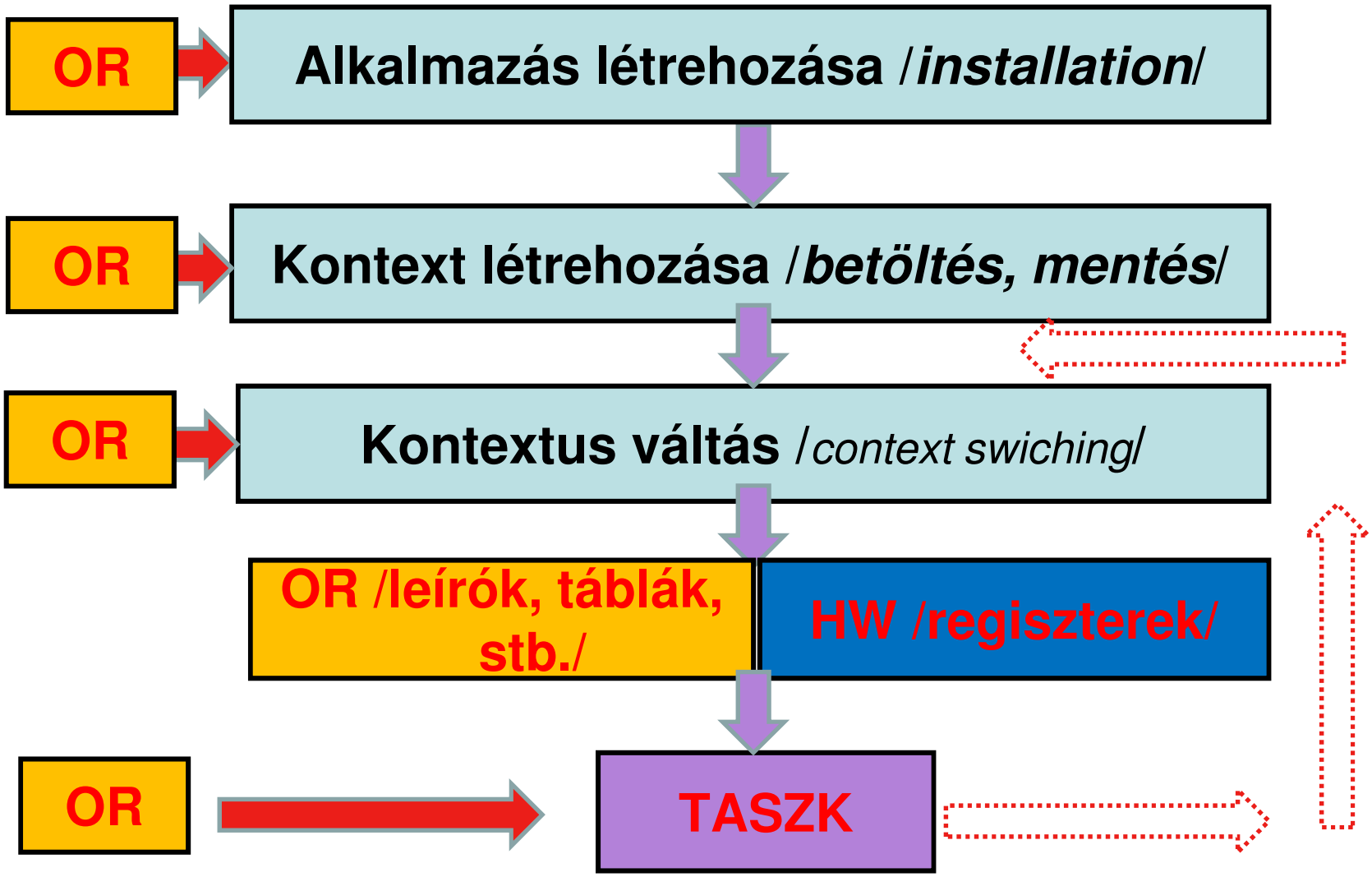
A taszk az algoritmusát megvalósító – utasításokon és kiinduló adatokon, illetve pillanatnyi állapotát leíró környezeten (kontext) alapuló - program futtatásával realizálódik

- 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
- Minden taszk-nak /felhasználónak úgy kell éreznie, mintha saját, külön processzora lenne → Virtuális processzor
- Látszólag (virtuálisan) egy taszk teljesen párhuzamosan fut a többi taszkkal kivéve, ha 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 Intel64*

Többprocesszoros rendszerek

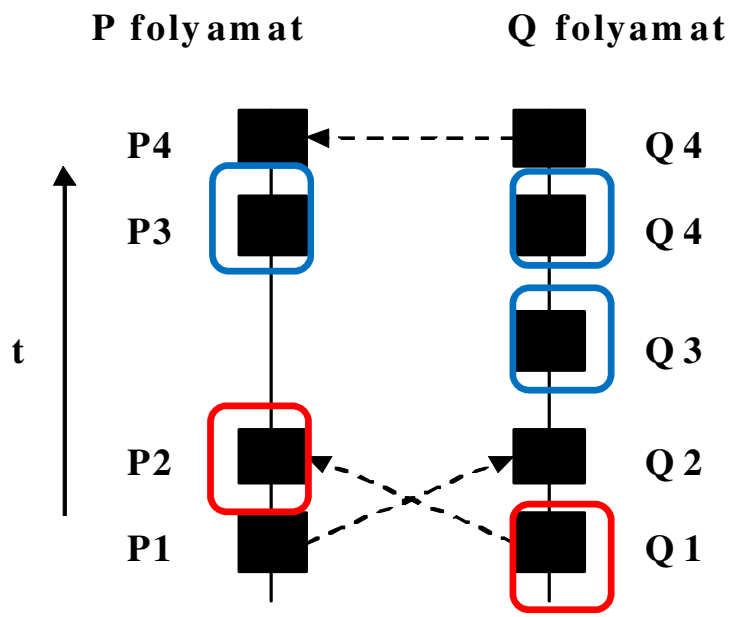
A mikroprocesszoros technika /és technológia/ lehetővé teszi, hogy egy nagyobb feladat kisebb elkülöníthető részeinek végrehajtását egy-egy külön processzorra bízzuk

→Kell egy mechanizmus, amelynek segítségével az egyes processzorok koordinálhatják működésüket a közös cél elérése érdekében

❖ **Multiprocesszoros rendszer**

Olyan struktúra, melyben ugyanazon rendszeralgoritmus egymástól független feladatainak konkurens végrehajtása folyik

Konkurencia



Két eseményt konkurensnek nevezünk, ha egyik sem tudja okszerűen befolyásolni a másikat

pl.: P_2 nem konkurens Q_1–el

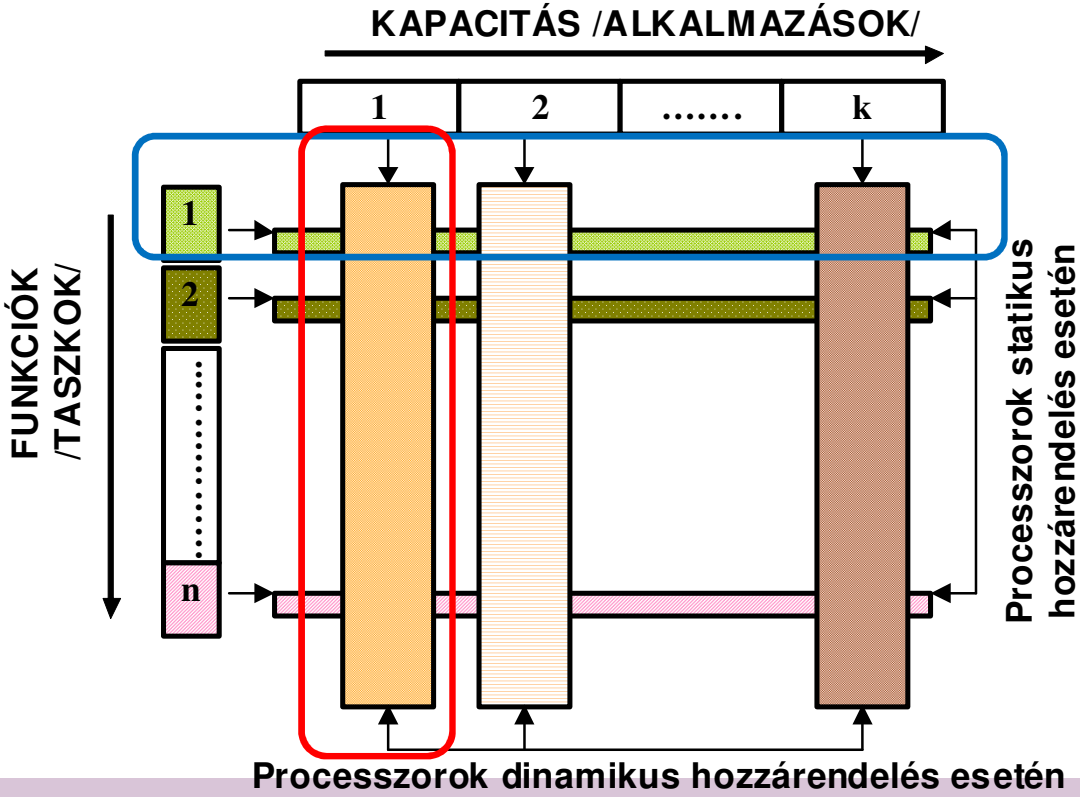
P_3 konkurens Q_3, Q_4 eseménnyel

- ***A multiprocesszoros rendszereket osztályozhatjuk a feladat-hozzárendelés módja és a processzorközi kapcsolat jellege szerint***
- **Feladat-hozzárendelés**
  - **Dinamikus feladat-hozzárendelés**
  - **Statikus feladat-hozzárendelés**

Feladat-hozzárendelés

- *A multiprocesszoros rendszernek bizonyos funkciókat /szolgáltatásokat/ kell biztosítania a felhasználó /alkalmazások/ számára*
- *Lehetővé kell tennie minél több felhasználó kiszolgálását*

→ *kapacitás biztosítás*



Dinamikus feladat hozzárendelés

- A rendszerkapacitás egy részére *minden funkciót* elláthat egy processzor

  Egy feladat-*terhelési viszonyoktól függően-* *bármelyik processzornak* *odaadható* →

  Logikailag egyenértékűek a processzorok

- Előnyök ←→Hátrányok

  Egy processzor meghibásodása a rendszernek csak egy részére terjed ki

  Megfelelő hibaterjedés védelem esetén

  A processzorok teljesítőképessége jól kihasználható

  Megfelelő terheléselosztó algoritmus esetén

  A processzorok teljesítőképessége *korlátozza* a megvalósítható funkciókat /*szolgáltatásokat*/

  Bonyolult szoftvert →sok taszkváltást igényel

Statikus feladat-hozzárendelés

- Egy processzor egy meghatározott szolgáltatást /*funkciót* / valósít meg az egész rendszer számára
  Pl.: I/O processzor
- Előnyök ←→Hátrányok

  Egyszerű szoftver→ *minimális a kölcsönhatás az egyes processzorok feladatai között*

  Elmarad a taszkváltással összefüggő adminisztráció

  Egyszerűbb az operációs rendszer ütemezése

  A processzorok felépítése a funkcióhoz optimalizálható →*jobb teljesítmény*

  Az egyes processzorok terhelése jelentősen eltérhet

  Érzékenyebb a meghibásodásra →*feladatát általában csak ugyanolyan típusú processzor veheti át*

Processzorközi kapcsolat

- *Az együttműködés megvalósításához mind hardver, mind szoftver szinten kapcsolatot kell kialakítani az egyes processzorok között.*
  - Lazán csatolt rendszerek
  - Szorosan csatolt rendszerek

❑ Lazán csatolt rendszerek

- A processzorközi kapcsolat üzenetorientált
  →*Lassú és kevés üzenet*

-Az egyes processzorokon különálló operációs rendszerek futhatnak

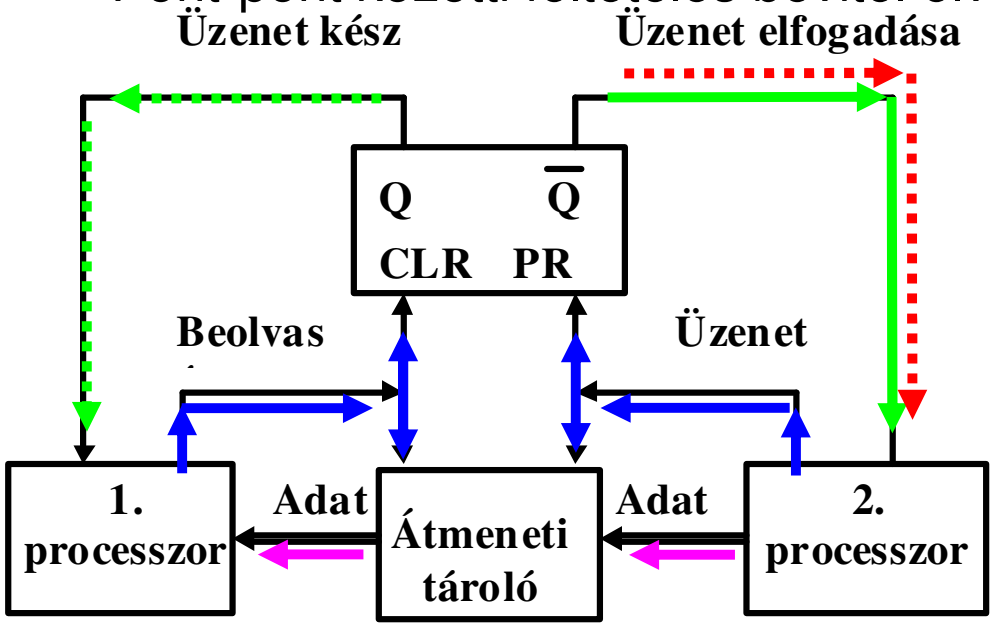
-A processzorokat kommunikációs alrendszerekből megvalósított csatornák kötik össze.

- A logikai és fizikai közeg függ
  -A rendszer nagyságától
  -Térbeli elhelyezkedésétől.

Lazán csatolt rendszerek

- *Elv: az üzenet átvitelének idejére a forrás processzor a fogadót perifériájának tekinti.*
  - Kis rendszernél, kevés információ átvitelére
    Pl.: hardverre alapozott szemafor

Pont-pont közötti feltételes bevitel elve



Előnyök←→Hátrányok

Egyszerű

Lassú

→DMA segíthet

Merev

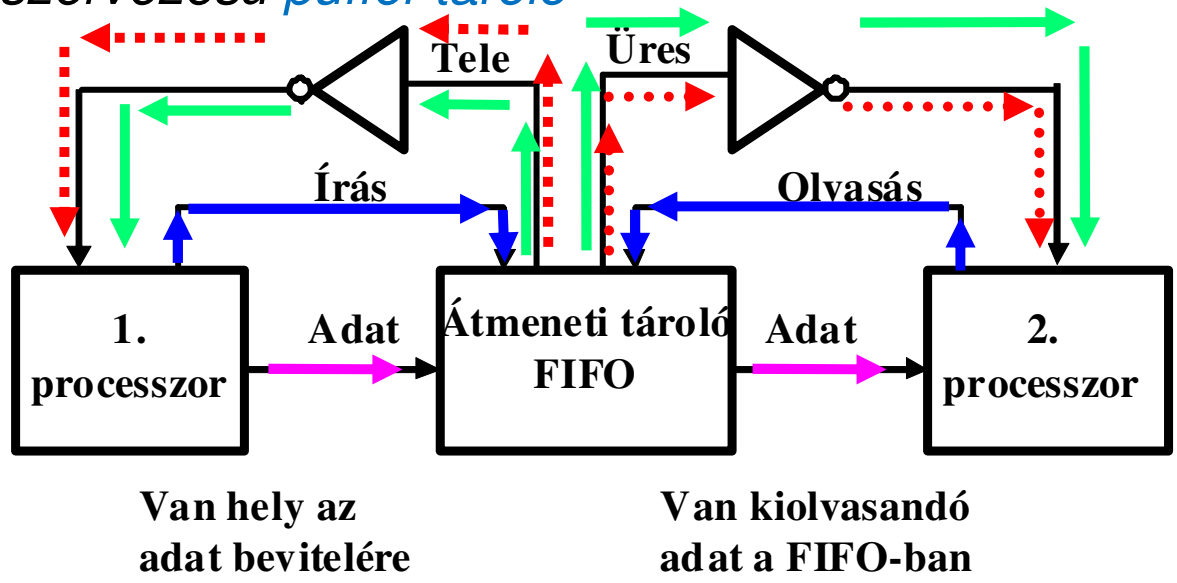
→Hálózatok segíthetnek

Nagyobb rendszereknél jelentőssé válhat az operációs rendszerek együttműködésének biztosítása

Lazán csatolt rendszerek

- *Probléma*: a forrás és a nyelő processzorok között nagymértékben ingadozik az adatátvitel sebessége, *ez függ az egyéb feladataiktól is*

  →Megoldás : *Mindkét oldal felé külön szemafor és FIFO szervezésű puffer tároló*



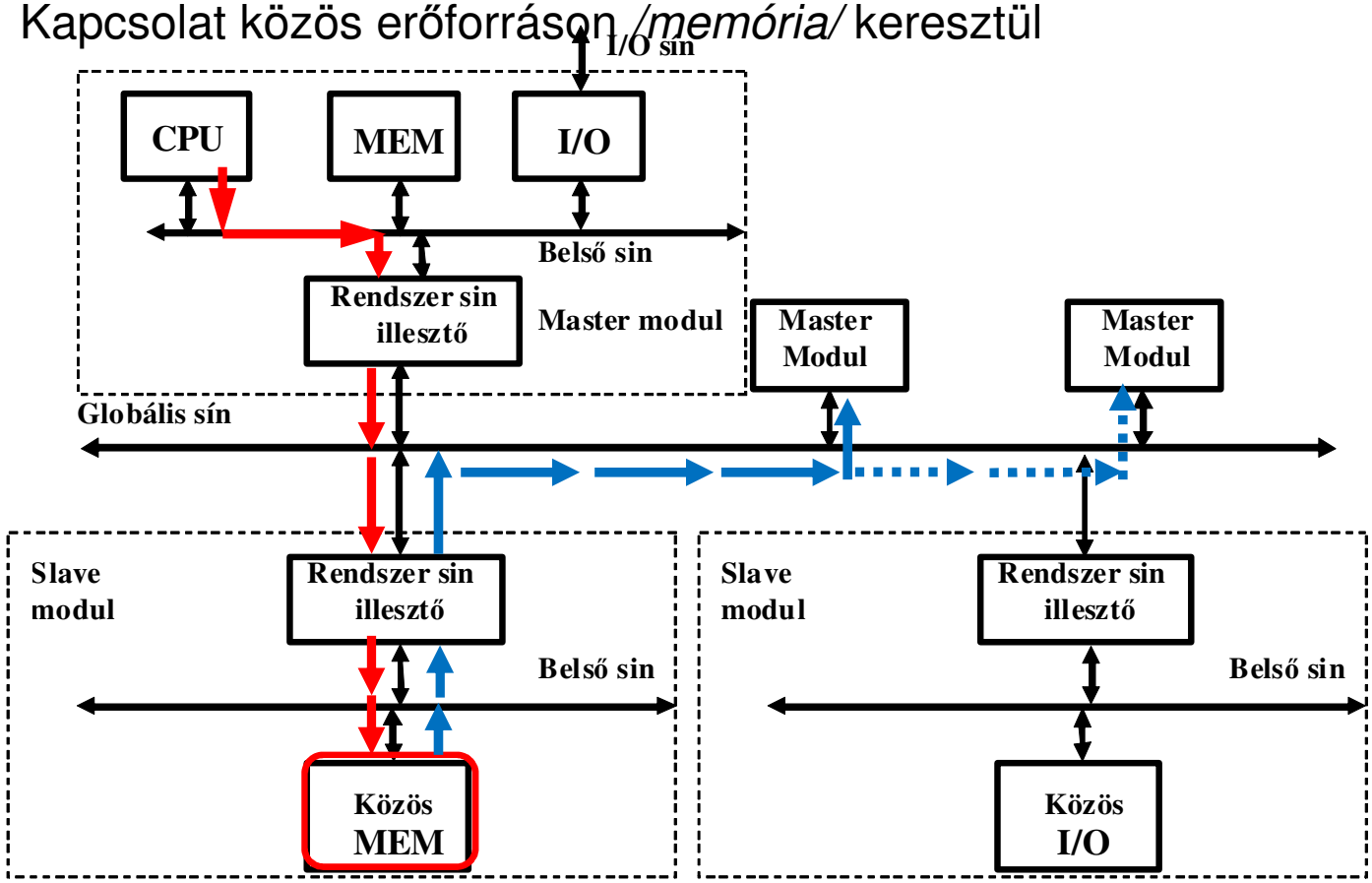
✧ **Számítógép hálózatok →lokális és globális → Kommunikációs processzorok**

Szorosan csatolt rendszerek

-A processzorközi kapcsolat közös erőforráson keresztül valósul meg

-Közös az operációs rendszer

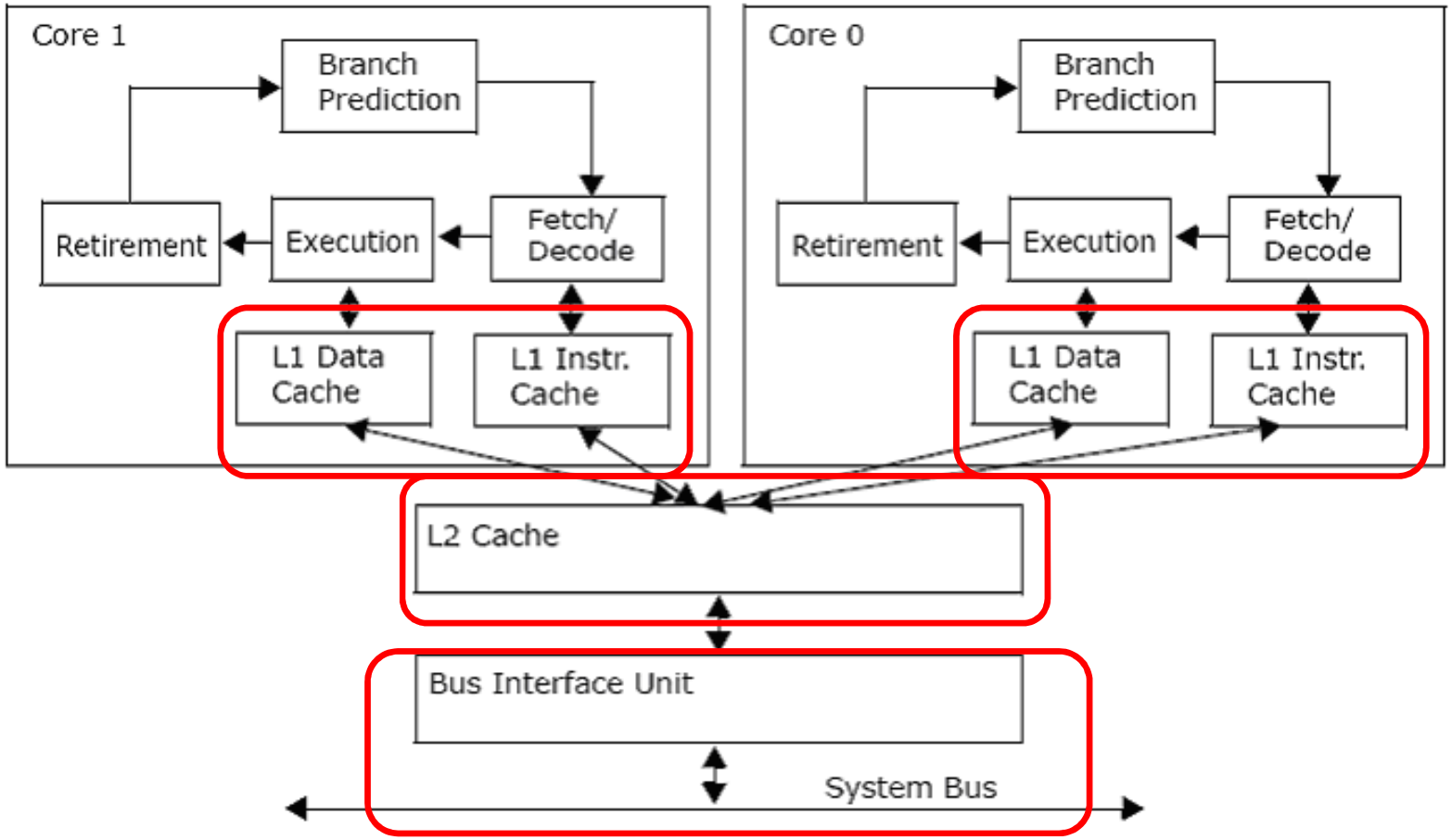
- Kapcsolat közös erőforráson *(memória)* keresztül



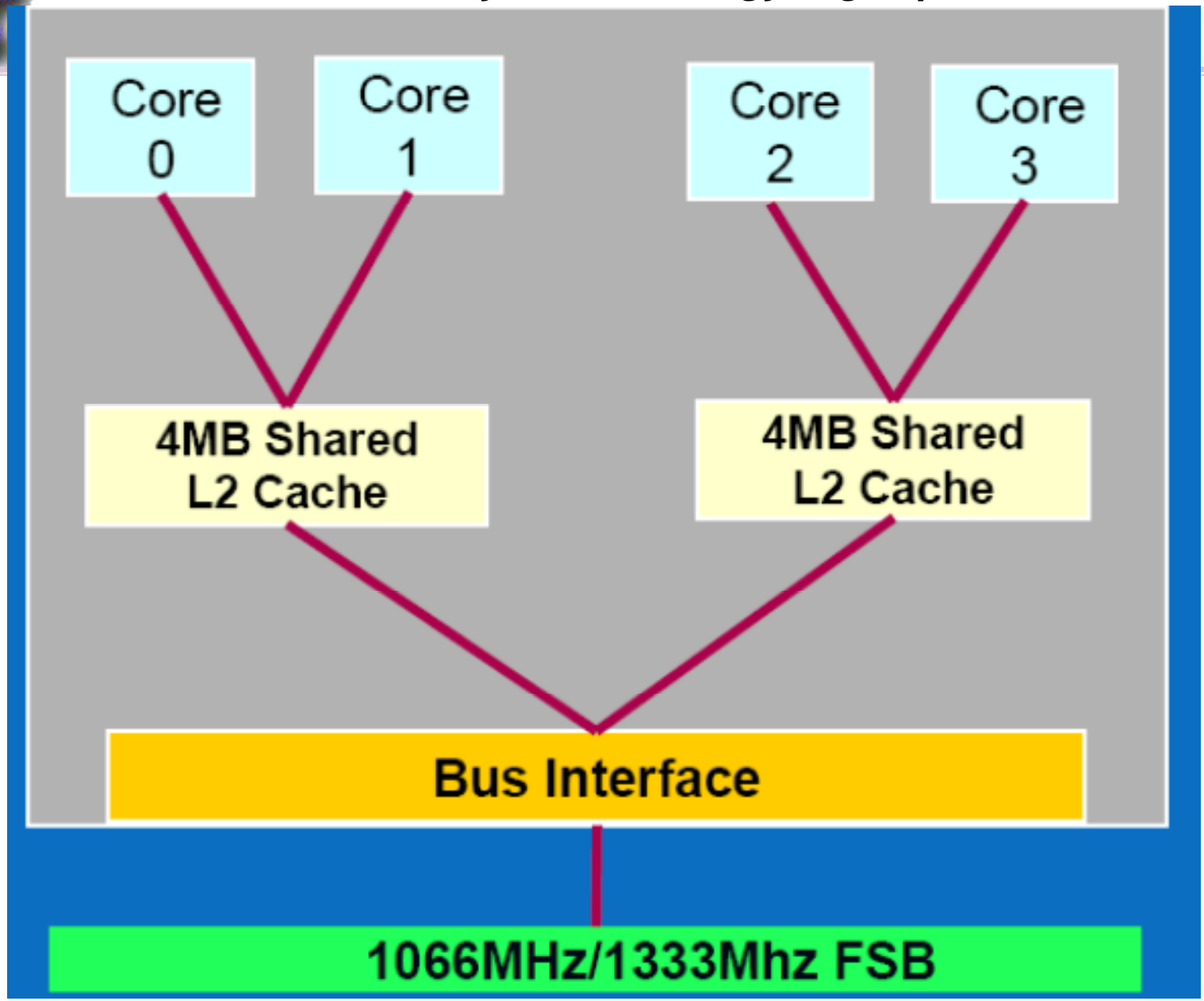
Kétsínes multiprocesszoros struktúra

Dual core

kétmagos multiprocesszoros struktúra



Esettanulmány - Pentium négymagos processzor



Típus	mag	szál	cache	órajel
Core i3	*2*	*4*	*3MB*	*1,6-2,4GHz*
Core i5	*2-4*	*4*	*3-6MB*	*2,7-3,1GHz*
Core i7-8086K	*6*	*12*	*12MB*	*1,7-4GHz*
Core i9-9900K	*8*	*16*	*16MB*	*3.6-5GHz*
Xeon E7	*10*	*20*	*30MB*	*2,4GHz*

Mag → *Core* → *AS* /*Architectural state* / → ~*CPU* → *Hw*

Szál → *Thread* → *SW*

Szorosan csatolt rendszerek

- A processzorközi kommunikáció
  - Logikai szinten

  Folyamatok közötti *adat-és vezérlés kommunikáció*

  →Postafiók elv

  A mesterek üzeneteket helyezhetnek el és olvashatnak ki egy közös memóriaterületről →postafiók →Az adatok elhelyezését – kiolvasását megfelelő szemaforokkal koordinálják

- A megoldást tekintve a fogadó modul szempontjából érdektelen ki tölti be az adatokat
- Fordítva ez igaz a küldő modulra is
- A többi modul a rendszersín művelet alatt dolgozhat a saját erőforrásaival
- A rendszersín szűk keresztmetszetté válhat

- Fizikai szinten

  A mester modul rendszersínhez →*közös erőforrás* való hozzáférését kell koordinálni →*ARBITER*

  – Központosított →***centralizált***

  – Elosztott →***decentralizált***

  – Soros⇔Párhuzamos megoldású

- A logikai szintű szinkronizáció *(szemaforkezelés)* megvalósításához hardvertámogatás kell

  →*Kölcsönös kizárás* megvalósítása

  Mester$_i$ számára a postafiók szemaforjának ellenőrzése és visszaírása idejére ki kell zárni mester$_k$ modult (→más mestereket) annak használatából

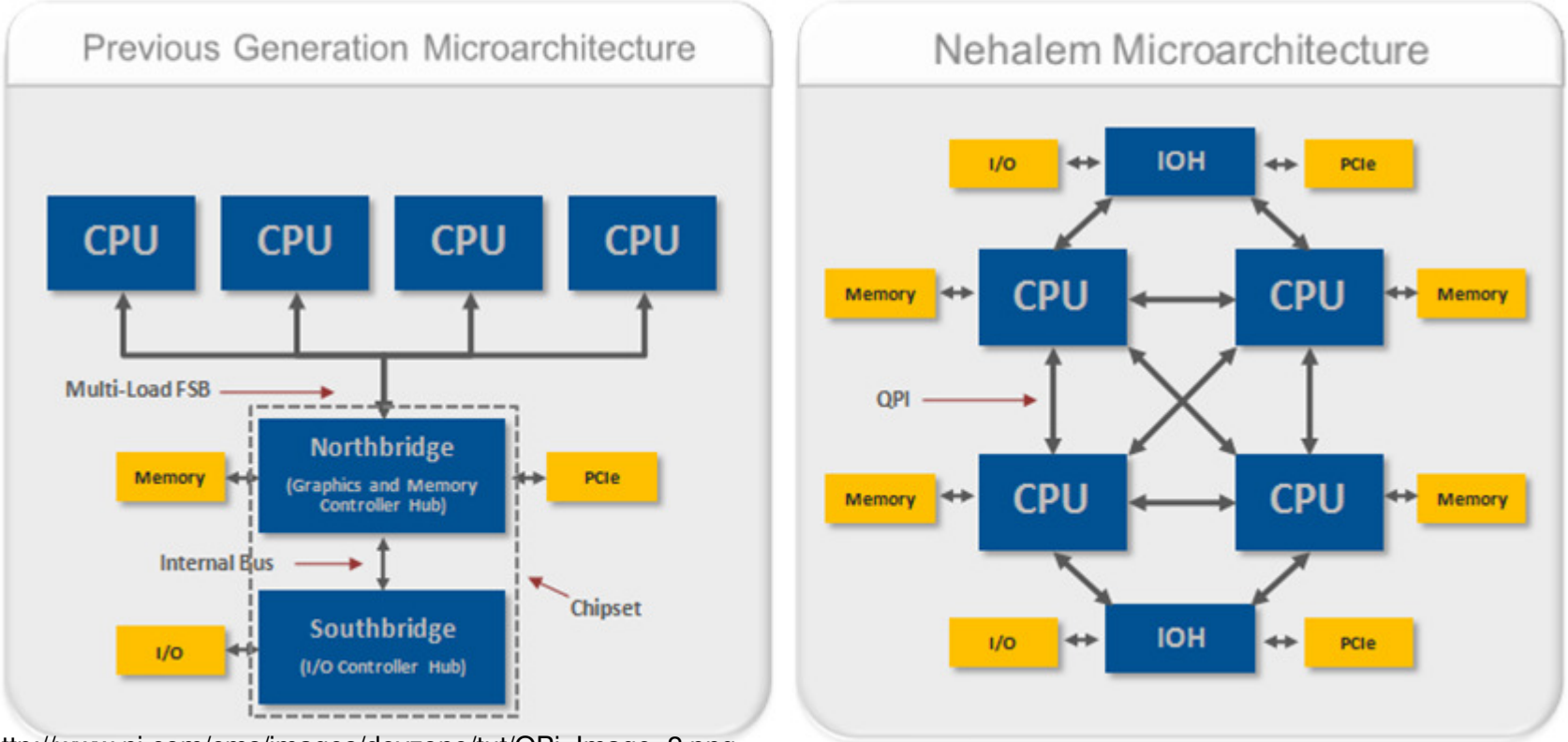
  – RMW típusú utasítás→*elemi oszthatatlan művelet*

  – BUS LOCK, BUS UNLOCK

  – Szoftver+hardver megoldás /*P és V primitív*/

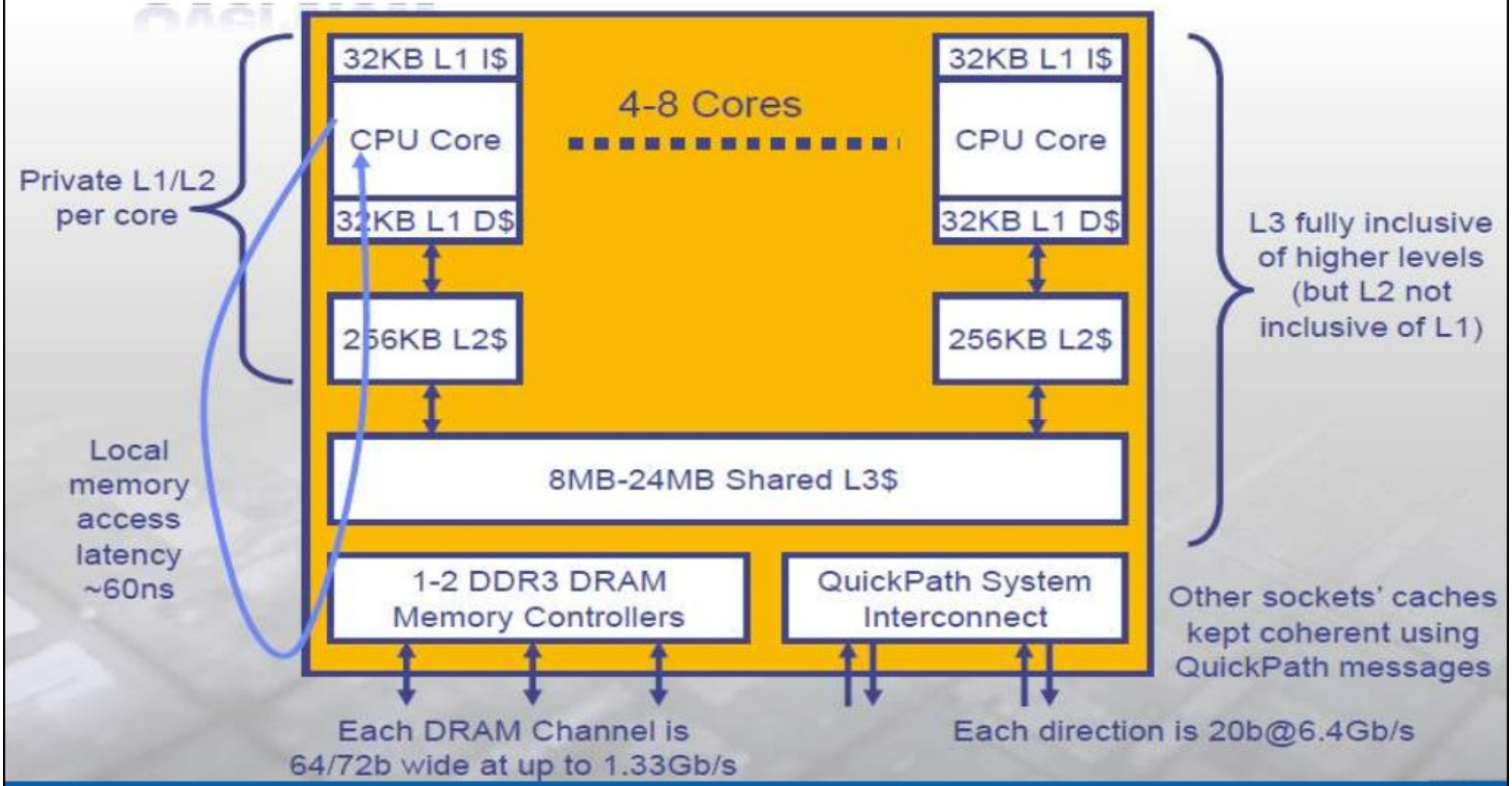
Processzor közi kapcsolat (Intel-példa)

QPI: QuickPath Interconnect
Processzorok közötti adatátvitel
20 lane (adatpálya) mindkét irányba + órajel (2.4GHz….4.8GHz)
Adatátvitel az órajel mindkét élére
Processzorok memóriáit egyetlen címtérbe képezi le



Forrás: http://www.ni.com/cms/images/devzone/tut/QPi_Image_2.png

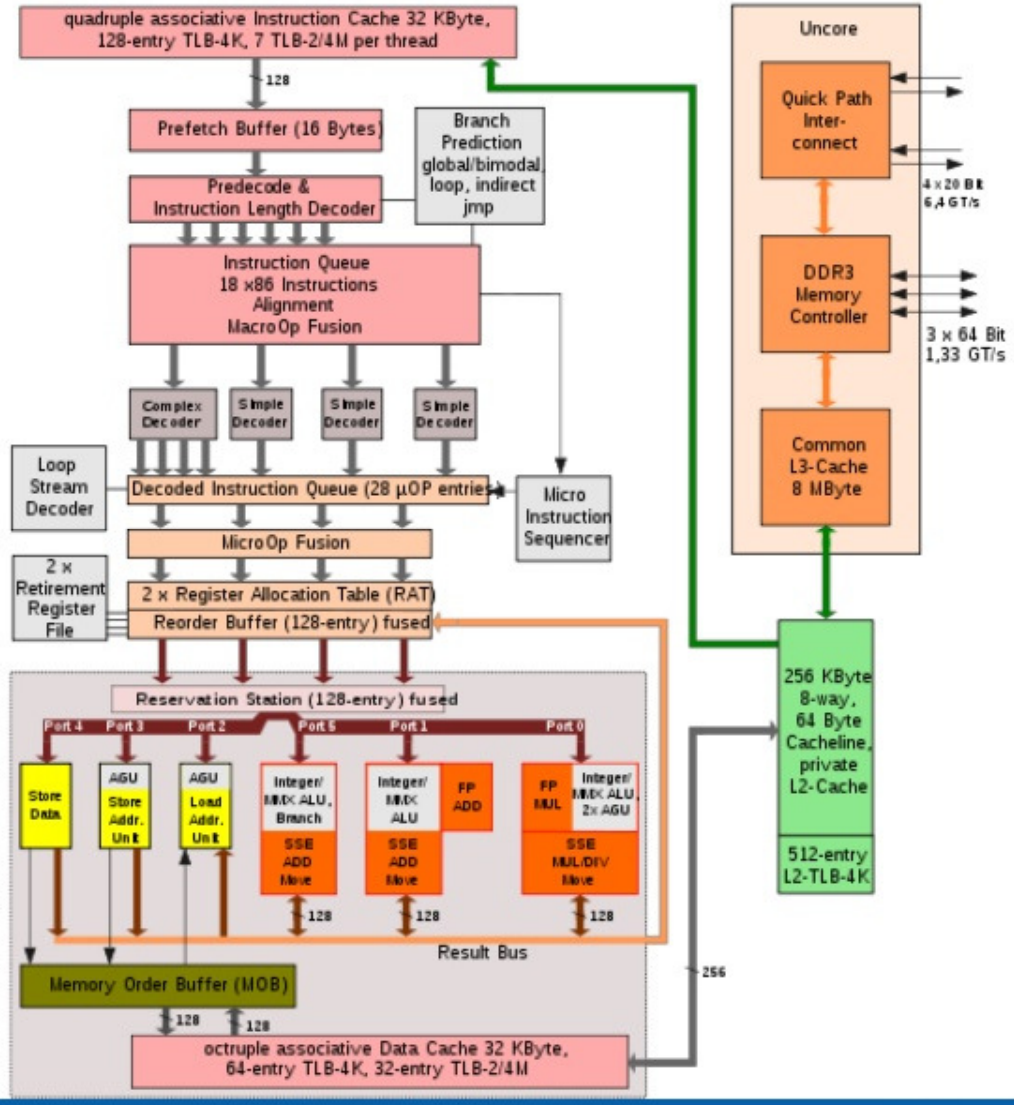
Nehalem Memory Hierarchy Overview



Cache Hierarchy Latencies

- L1 32KB 8-way, Latency 4 cycles
- L2 256KB 8-way, Latency < 12 cycles
- L3 8MB shared , 16-way, Latency 30-40 cycles (4 core system)
- L3 24MB shared, 24-way, Latency 30-60 cycles(8 core system)
- DRAM , Latency ~ 180 – 200 cycles

Nehalem Microarchitecture

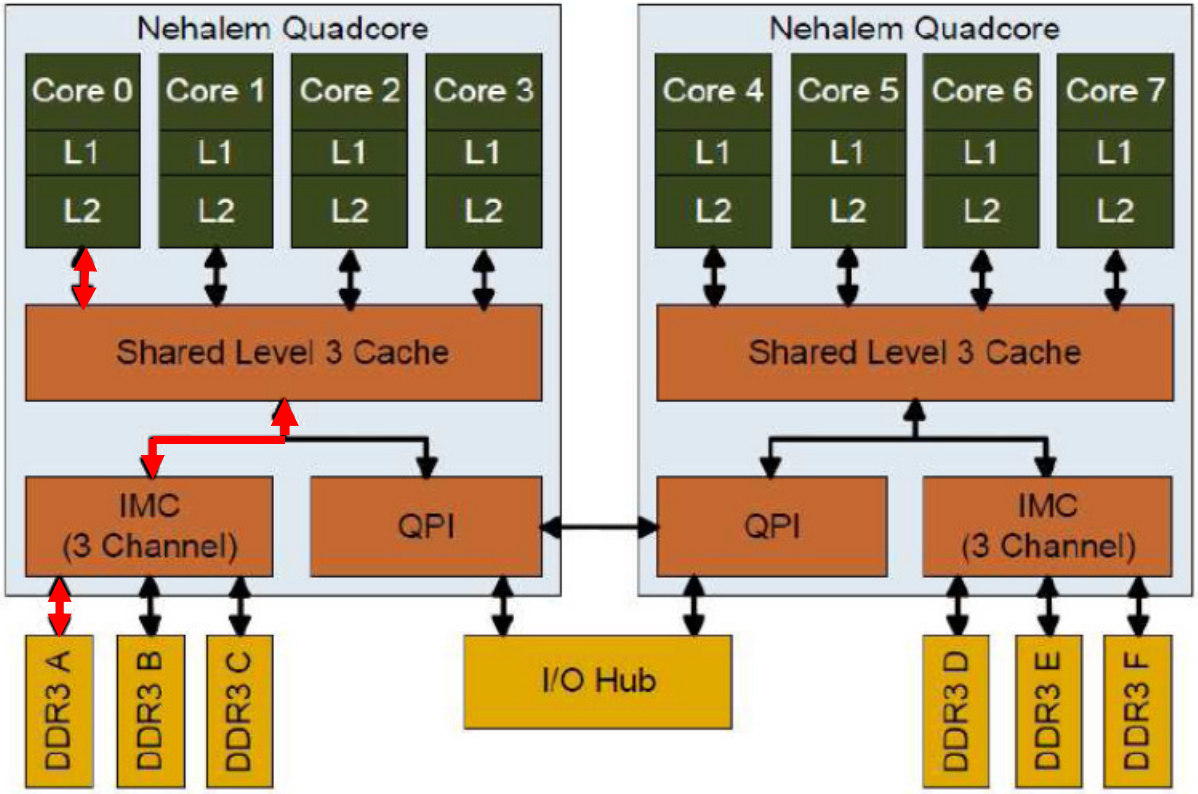


Adat elérése saját memóriából

Adat elérése a saját kezelésben lévő DDR3 memóriamodulból

a) Olvasási késleltetés a saját kezelésű memória hivatkozás esetén

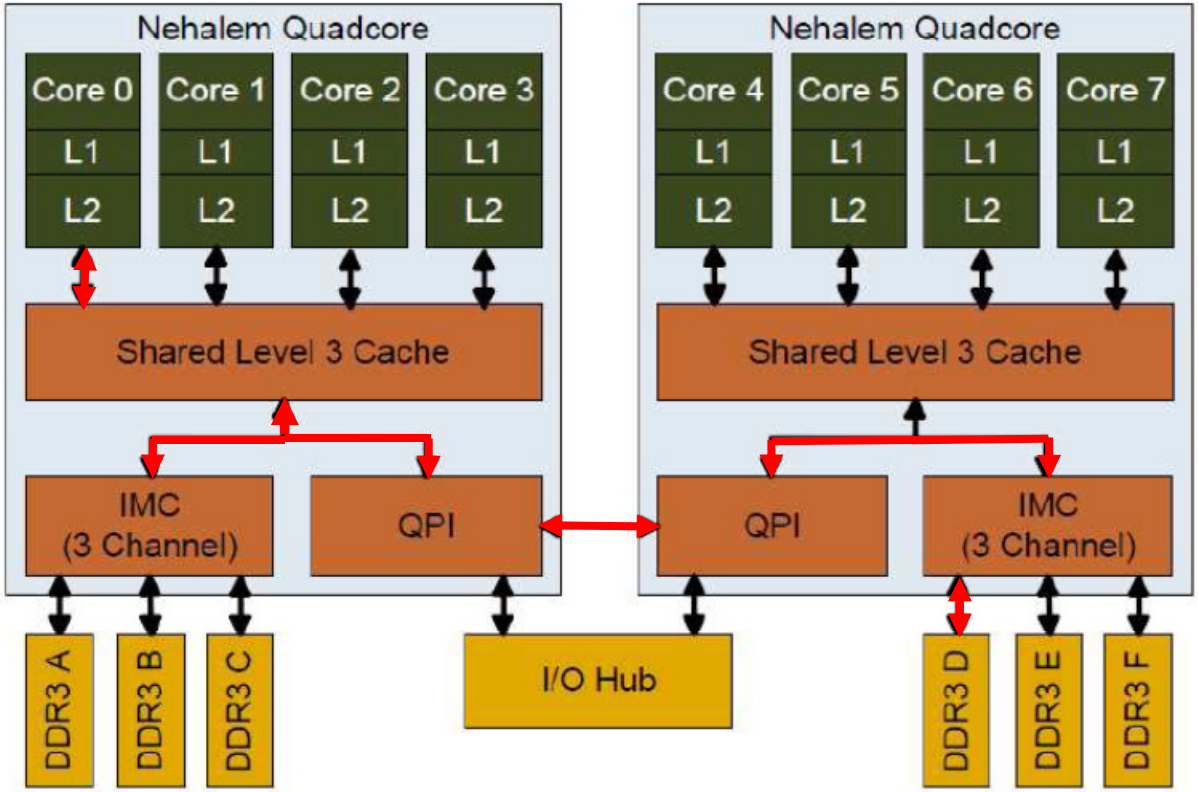
Read latency:
65.1 ns (190 cycles)



Molka D., Hackenberg D., Schöne R., Müller M. S., Memory Performance and Cache, Coherency Effects on an Intel Nehalem multiprocessor System, Proc. 18th Int. Conf on Parallel Architectures and Compilation, Techniques, PACT 2009, http://doi.ieeecomputersociety.org/10.1109/PACT.2009.22

Adat elérése másik processzor memóriájából

b) Olvasási késleltetés, ha a hivatkozott adat másik tartományban található



Read latency:
106 ns (~310 cycles)

Az olvasási késleltetés 41 ns-el nőtt ebben az esetben

Molka D., Hackenberg D., Schöne R., Müller M. S., Memory Performance and Cache, Coherency Effects on an Intel Nehalem multiprocessor System, Proc. 18th Int. Conf on Parallel Architectures and Compilation, Techniques, PACT 2009, http://doi.ieeecomputersociety.org/10.1109/PACT.2009.22

Összefoglalás

- Taszk fogalma
- Multitaszking elve
- Megoldandó feladatok több taszk futtatásakor
- Védelem
- Többprocesszoros rendszerek
- Statikus és dinamikus feladat hozzárendelés
- Lazán- és szorosan csatolt rendszerek
- Magok közötti kommunikáció