HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

MŰEGYETEM 1782

Budapest,
2025.02.18.

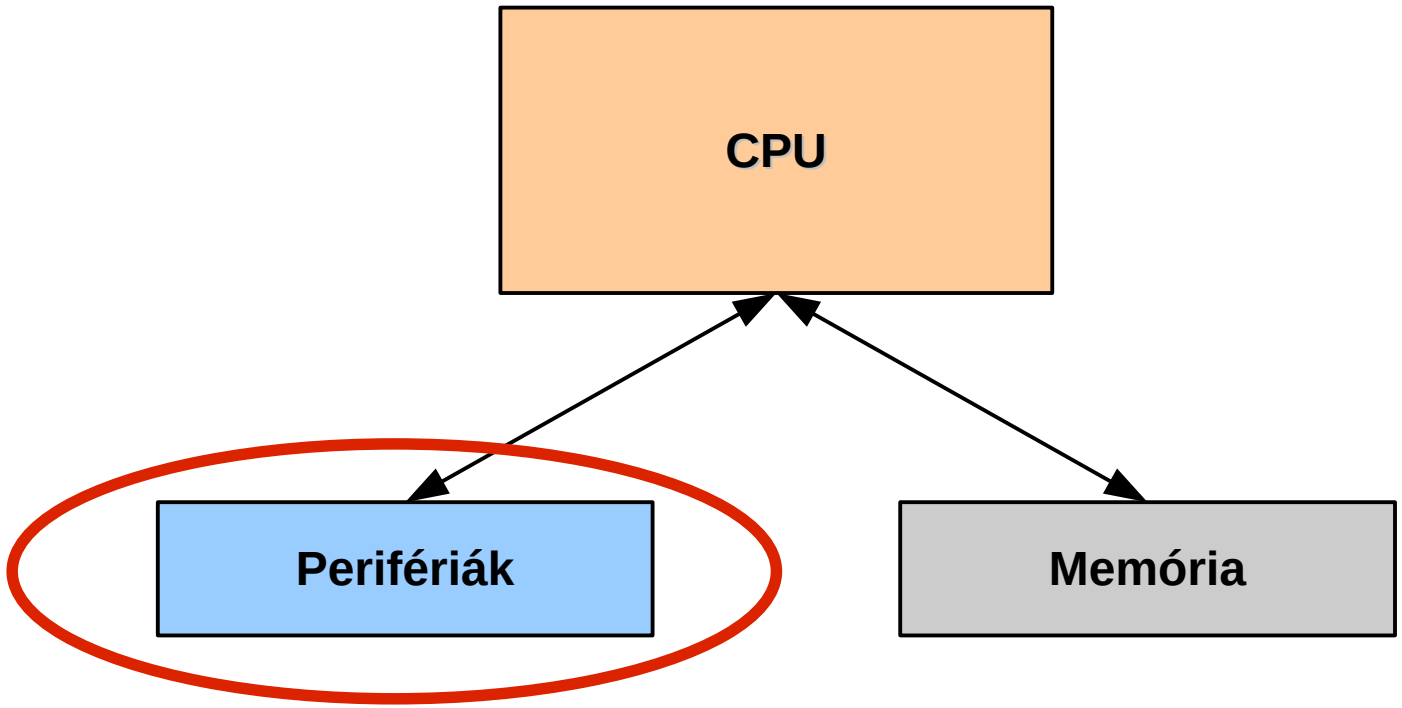
SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Perifériakezelés

Horváth Gábor, Belső Zoltán

BME Hálózati Rendszerek és Szolgáltatások Tanszék
ghorvath@hit.bme.hu, belso@hit.bme.hu

HOL TARTUNK?



PERIFÉRIÁK

- Sokféle van:
  - Bemeneti / kimeneti
  - Késleltetésérzékeny / nem az
  - Sávszélesség-igényes / nem az
  - Bithibát toleráló / nem az
  - Stb.

	Partner	Irány	Adatforgalom
Billentyűzet	Humán	Bemeneti	kb. 100 byte/s
Egér	Humán	Bemeneti	kb. 200 byte/s
Hangkártya	Humán	Kimeneti	kb. 96 kB/s
Printer	Humán	Kimeneti	kb. 200 kB/s
Grafikus megjelenítő	Humán	Kimeneti	kb. 500 MB/s
Modem	Gép	Ki/Be	2-8 kB/s
Ethernet hálózati interfész	Gép	Ki/Be	kb. 12.5 MB/s
Diszk (HDD)	Gép	Ki/Be	kb. 50 MB/s
GPS	Gép	Bemeneti	kb. 100 byte/s

PERIFÉRIÁK

- Kérdések:
  - Hogyan tud a processzor adatátvitelt kezdeményezni?
  - Hogyan tudnak a perifériák adatátvitelt kezdeményezni?
  - Hogyan lehet az adatátvitelt hatékonyan, hibamentesen levezényelni?
  - Hogyan/hová kössük a perifériákat a számítógéphez?

PERIFÉRIÁK

- Kérdések:
  - **Hogyan tud a processzor adatátvitelt kezdeményezni?**
  - Hogyan tudnak a perifériák adatátvitelt kezdeményezni?
  - Hogyan lehet az adatátvitelt hatékonyan, hibamentesen levezényelni?
  - Hogyan/hová kössük a perifériákat a számítógéphez?

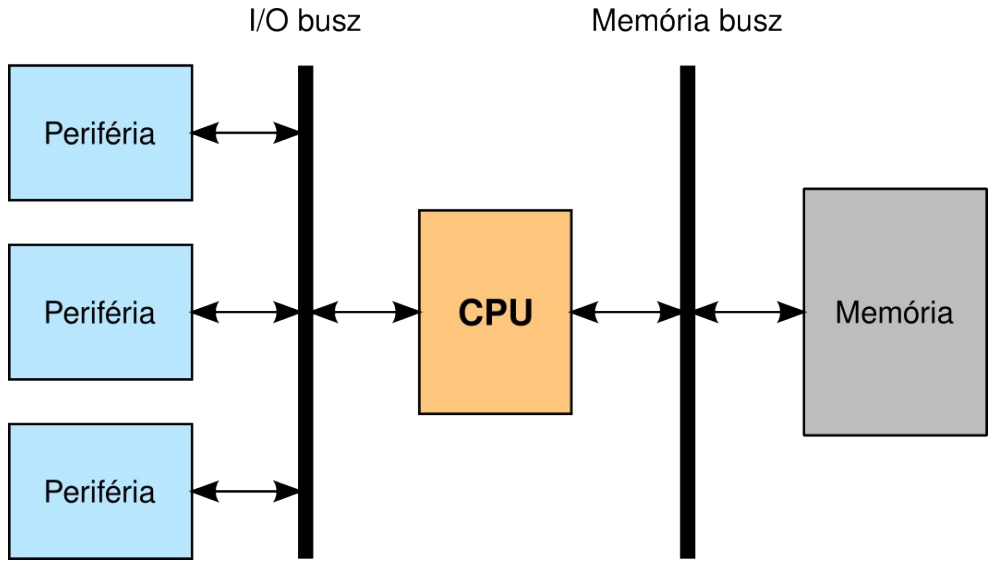
CÍMZÉS

- Memóriatartalom elérése: címzéssel
- Perifériák elérése: címzéssel

- Mennyire különülnek el a címek?
  - Lehet **külön** memória és I/O címtartomány
  - Lehet **közös** (osztott) címtartomány

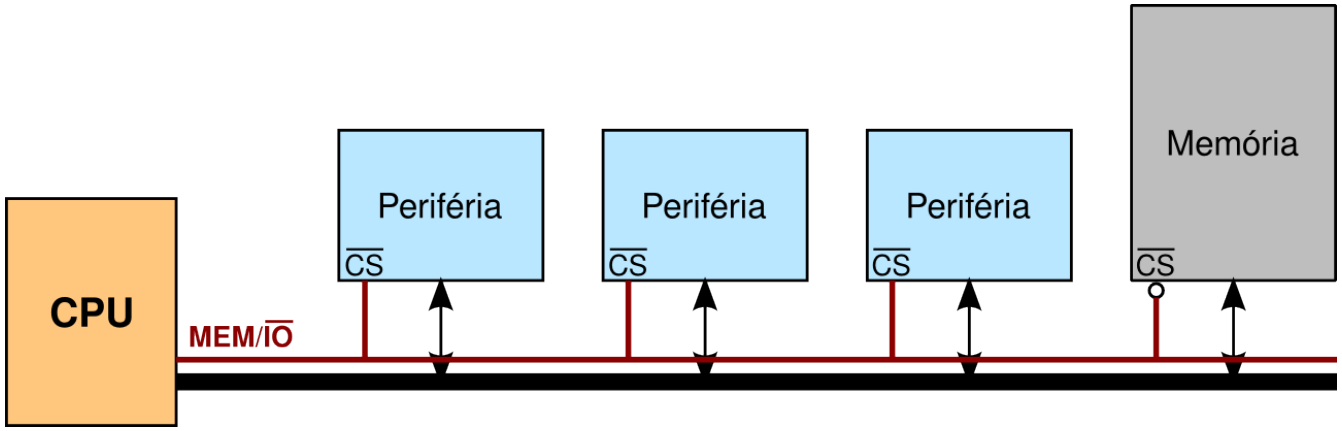
KÜLÖN MEMÓRIA ÉS I/O CÍMTÉR

- Két független címtér
  - x86: memória: 0 – 4GB, I/O: 0 – 64kB
- Külön perifériakezelő utasítások
  - `R0 ← MEM[0x60]`: memóriára vonatkozik
  - `R0 ← IO[0x60]`: perifériákra vonatkozik
- Megvalósítás:
  - Szeparált buszokkal:



KÜLÖN MEMÓRIA ÉS I/O CÍMTÉR

- Két független címtér
  - x86: memória: 0 – 4GB, I/O: 0 – 64kB
- Külön perifériakezelő utasítások
  - `R0 ← MEM[0x60]`: memóriára vonatkozik
  - `R0 ← IO[0x60]`: perifériákra vonatkozik
- Megvalósítás:
  - Multiplexált buszokkal:

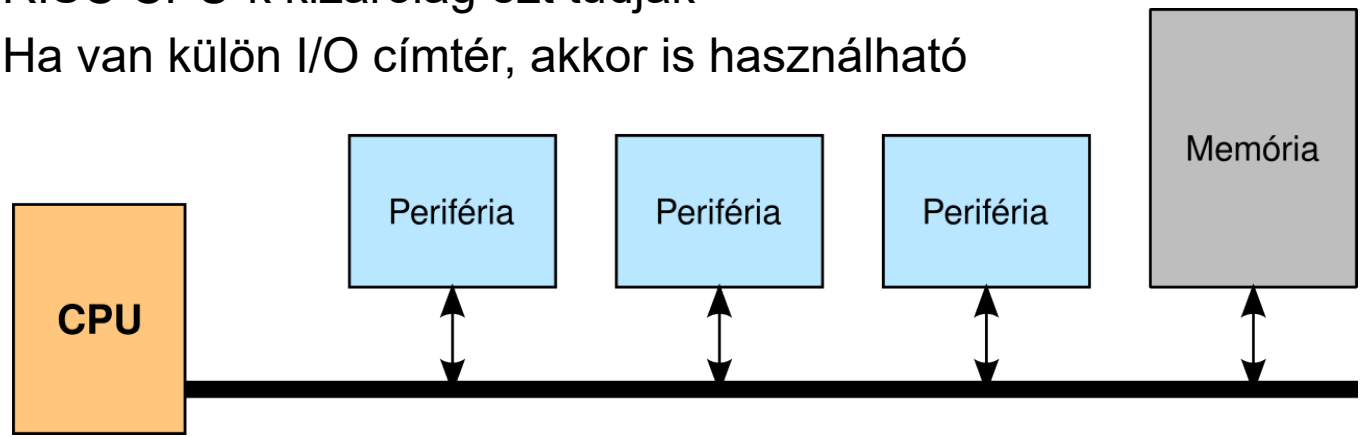


MEMÓRIÁRA LEKÉPZETT PERIFÉRIAKEZELÉS

- Egyes memóriacímekre perifériák válaszolnak
- Nagyon egyszerű a kommunikáció:

```
char* p = 0x60;
int x = *p;
```

- **Előny**: memóriakezelő utasítások többen vannak, és kényelmesebbek, mint az I/O kezelő utasítások
- **Hátrány**: nem elérhető lyukak a memóriában
- Megvalósítás:
  - RISC CPU-k kizárólag ezt tudják
  - Ha van külön I/O címtér, akkor is használható



PERIFÉRIÁK

- Kérdések:
  - Hogyan tud a processzor adatátvitelt kezdeményezni? √
  - **Hogyan tudnak a perifériák adatátvitelt kezdeményezni?**
  - Hogyan lehet az adatátvitelt hatékonyan, hibamentesen levezényelni?
  - Hogyan/hová kössük a perifériákat a számítógéphez?

MEGSZAKÍTÁSKEZELÉS

- CPU rendszeresen kérdezgeti a perifériákat: Polling
- CPU „INT” lábán lehet megszakítást kérni
  - Gond: interrupt források száma > INT lábak száma (1-2)
  - Muszáj többeket ugyanarra az INT lábra kötni
    - Honnan tudjuk, hogy melyik kért megszakítást?
    - Mi van, ha egyszerre többen is kérnek megszakítást?

- 1. megoldás: **osztott interrupt**
  - Mindenki ugyanazon a vonalon jelez
  - Az interrupt szubrutin mindenkit körbekérdez, hogy ki volt
  - Ha többen is jeleznek egyszerre?
    - Kiszolgálási sorrend = körbekérdezési sorrend

MEGSZAKÍTÁSKEZELÉS

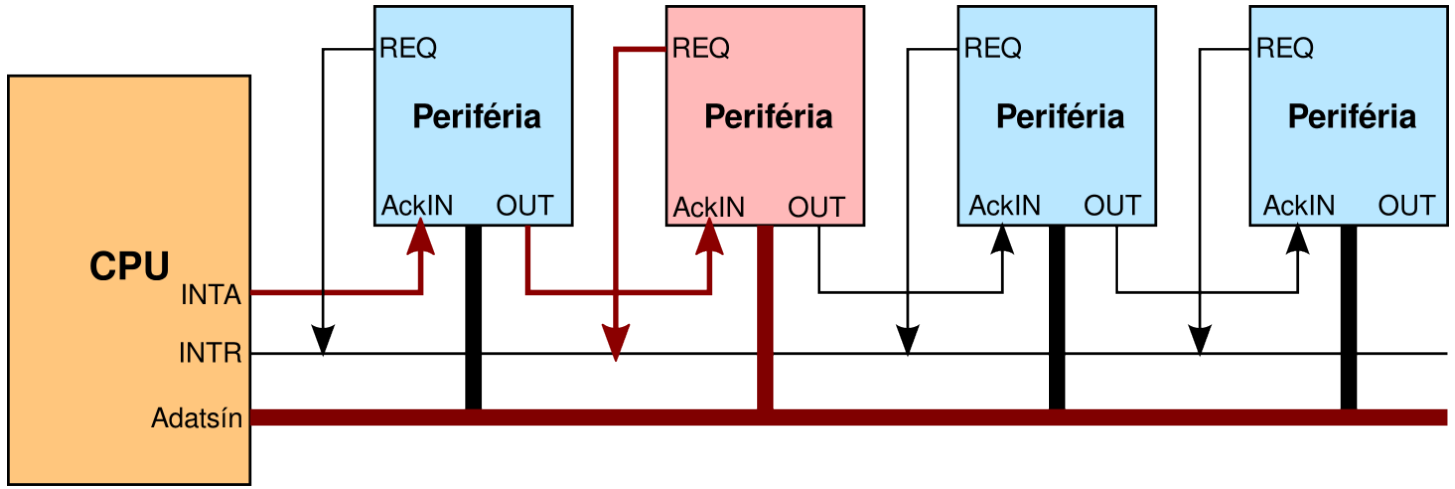
- 2. megoldás: **vektoros megszakításkezelés**
  - Megszakításkor a CPU kiad egy INTA (elfogadás) jelet
  - A jelezni kívánó periféria az adatbuszra teszi a számát
    - Ez a szám megmondja, hogy hányas számú szubrutinnak kell őt kiszolgálnia
    - A CPU-nak van egy táblázata: **interrupt vektor tábla**. Mutatókat tartalmaz interrupt kiszolgáló szubrutinokra.
    - Amilyen számot mondott a periféria, a vektor tábla annyiadik szubrutinja fut le
- A kettő kombinációja
  - Néha a vektortábla elemeinek száma sem elég
  - Az azonos vektoron levő perifériák között marad a körbekérdezés

MEGSZAKÍTÁSKEZELÉS

- Na de mi van, ha többen is kérnek egyszerre interruptot?
  - Nem is ritka dolog
  - Miközben a CPU kiszolgál egy interruptot, több más interrupt is érkezhet
  - Valahogy sorba kell állítani a kéréseket
    - Daisy chaining-el
    - Interrupt vezérlővel

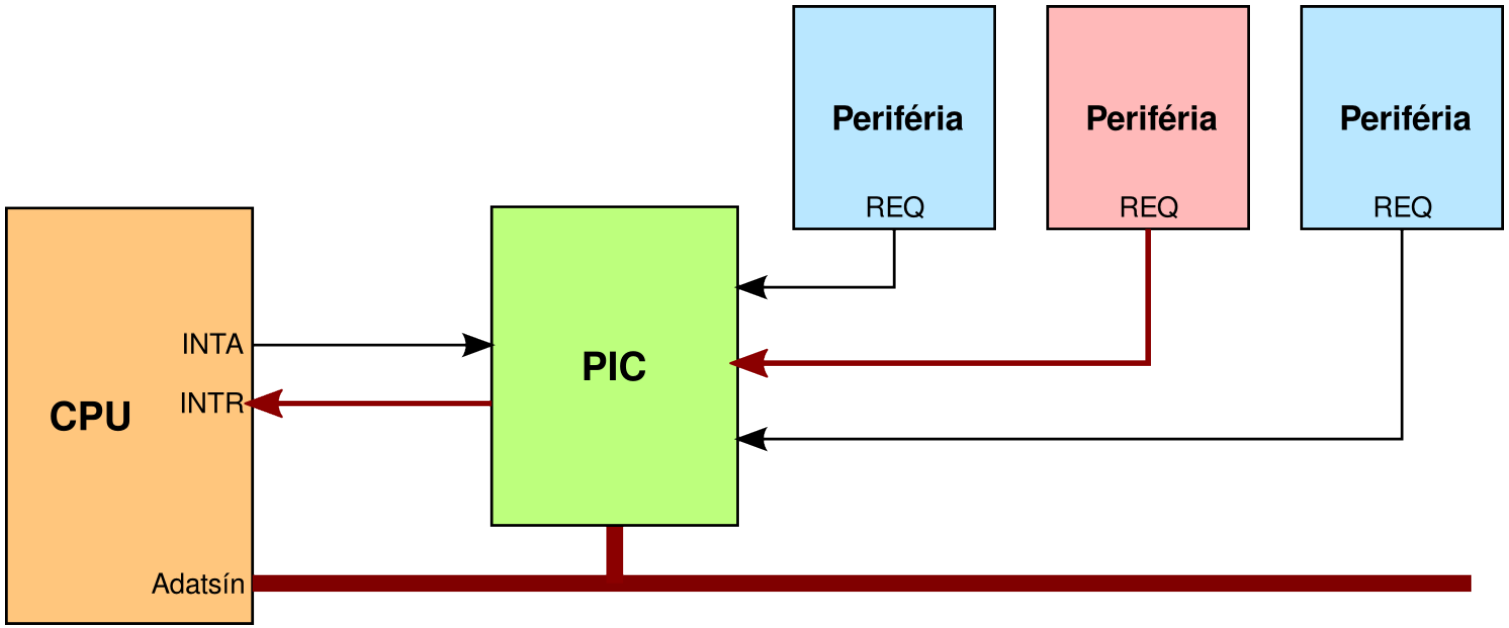
MEGSZAKÍTÁSKEZELÉS

- **Daisy chaining**:
  - A CPU interrupt elfogadás jele sorban végighalad minden periférián
    - Aki nem kér megszakítást, továbbadja
    - Aki kér, megállítja a jelet, és az adatsínre teszi a számát
  - A sorrend a prioritást is meghatározza
    - A sorban hátul lévők éheznek



MEGSZAKÍTÁSKEZELÉS

- **Interrupt vezérlő** (Programmable Interrupt Controller):
  - Több bemenete van
  - A PIC maga is egy periféria
  - A CPU (op. rendszer) perifériaműveletekkel beállíthatja,
    - hogy mi történjen, ha több interrupt kérés van egyszerre,
    - hogy mely kapcsolódó perifériák interruptjait engedélyezi



MEGSZAKÍTÁSKEZELÉS

- Interrupt kezelése **multiprocesszoros rendszerekben**
  - Egyszerű megoldás: minden megszakítás a boot processzorhoz fut be
  - Alternatív megoldás: fejlett interrupt vezérlővel (Intel: APIC, ARM: GIC, stb.)
    - Összetevők:
      - Minden processzornak van egy **lokális interrupt vezérlő**je
      - Van egy **rendszerszintű interrupt elosztó**
    - Ha egy periféria megszakítást jelez, a rendszerszintű elosztó továbbítja a megfelelő processzornak → **interrupt routing**
      - Az op. rendszer állítja be a rendszerszintű elosztót, hogy melyik megszakítást melyik CPU lokális interrupt vezérlője kezelje
    - A lokális interrupt vezérlők egymásnak is küldhetnek megszakítást
      - Ez egy lehetséges módja a processzorok egymás közötti kommunikációjának!

MEGSZAKÍTÁSKEZELÉS

- Amikor túl sok a jóból...
  - Vannak perifériák, melyek túl sok interruptot generálnak
    - Pl. a gigabit sebességű hálózati interfészek

Interrupt forrás	Tipikus ráta	Maximális ráta
10 Mbps Ethernet	812	14880
100 Mbps Ethernet	8127	148809
Gigabit Ethernet	81274	1488095

- A CPU megállás nélkül megszakítást kezel, mással nem tud haladni
- Irányelvek:
  - A megszakításkezelő szubrutin legyen rövid
  - Kritikus rendszerekben az interrupt rátát maximálják
  - A futó program kritikus szakaszaiban le kell tiltani a megszakítást
  - **Interrupt moderation**
    - A periféria több eseményt összevár, és akkor jelez, egyszer
    - A CPU egy interrupt-tal lekezeli a periféria összegyűlt mondanivalóját

PERIFÉRIÁK

- Kérdések:
  - Hogyan tud a processzor adatátvitelt kezdeményezni? √
  - Hogyan tudnak a perifériák adatátvitelt kezdeményezni? √
  - **Hogyan lehet az adatátvitelt hatékonyan, hibamentesen levezényelni?**
  - Hogyan/hová kössük a perifériákat a számítógéphez?

HATÉKONY ADATÁTVITEL

- Végső cél:
  - Adatok átvitele a perifériából a memóriába
- Kérdések:
  - Mi van, ha a küldő és a fogadó sebessége eltérő?
    → **Forgalomszabályozás**
  - Milyen útvonalon juttassuk az adatokat a memóriába?

FORGALOMSZABÁLYOZÁS

- Alapprobléma:
  - Készen áll-e mind a periféria, mind a CPU az adatátvitelre?

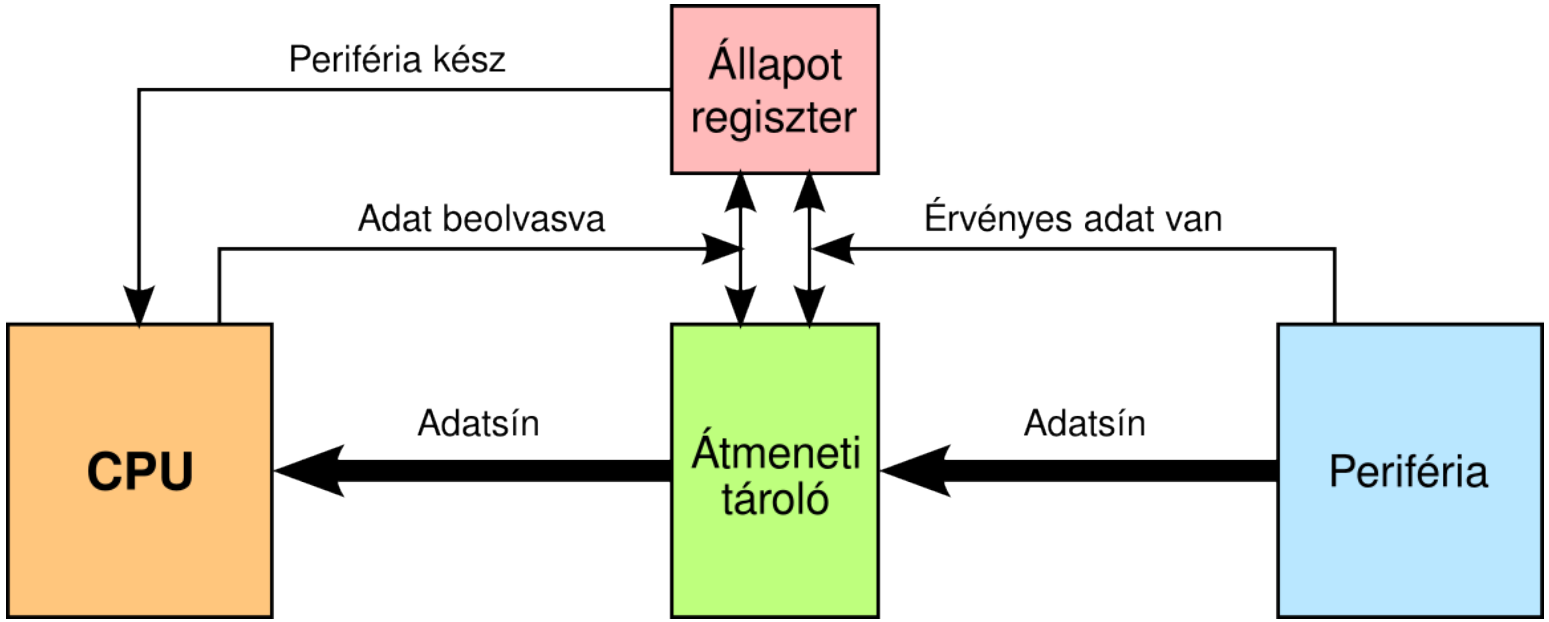
1) Feltétel nélküli adatátvitel

- Nincs forgalomszabályozás
- Egyik fél sem tudja közöli a másikkal, hogy kész az átvitelre
- Kétféle probléma léphet fel:
  - Adat egymásra-futás (küldő gyors, fogadó lassú)
  - Adathiány (küldő lassú, fogadó gyors)
- Példa: kapcsoló leolvasás, LED kigyújtás, ...

2) Egyoldali feltételes adatátvitel

- Az egyik oldal sebessége nem befolyásolható (a másiké igen)
- Állapotregiszter: van-e érvényes adat?
- Példa: hangbemenet, hálózati kártya
- Példa: olvasás művelet

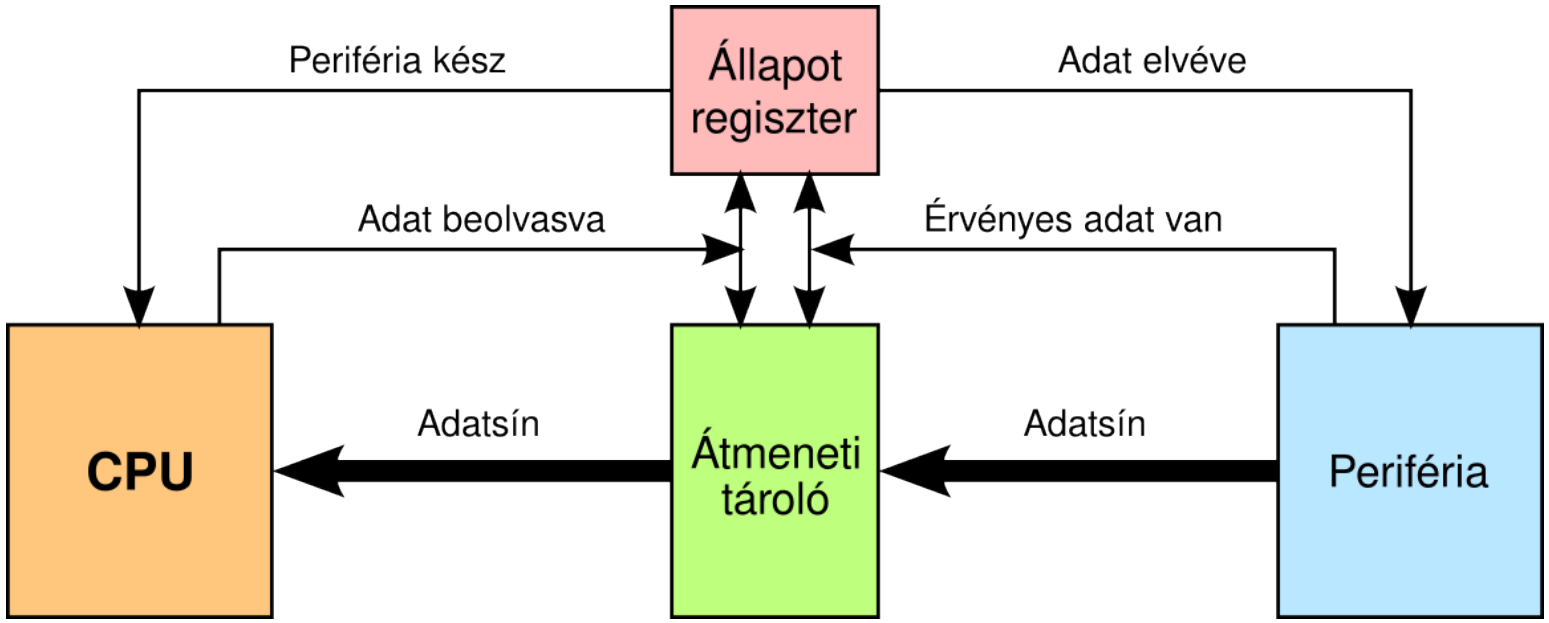
  → az állapotregiszterrel az adathiány elkerülhető!



3) Kétoldali feltételes adatátvitel

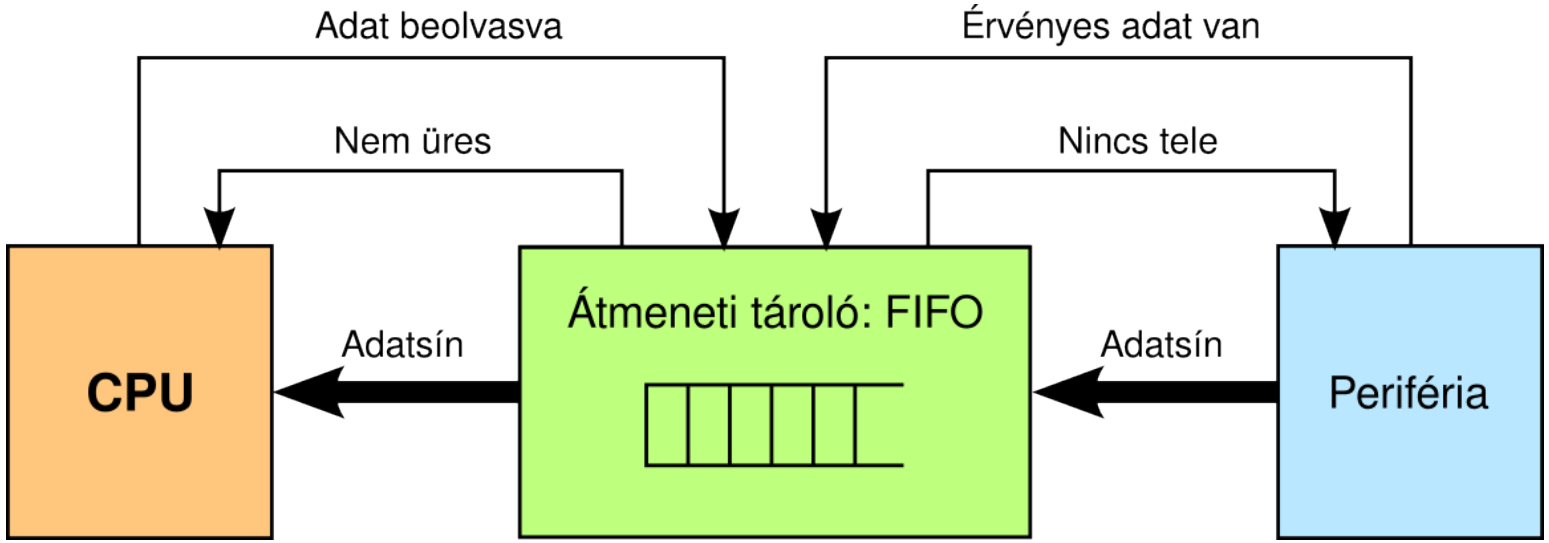
- Mindkét oldal sebessége befolyásolható
- Állapotregiszter: van-e érvényes adat?
- Példa: olvasás művelet

  → az állapotregiszterrel az adathiány és az adat egymásra-futás is elkerülhető!



4) Feltételes adatátvitel FIFO tárolóval

- Mindkét oldal sebessége befolyásolható
- Feleknek nem kell mindig bevárniuk egymást
- Akkor jó, ha:
  - Ingadozó az adatforgalom
  - Ingadozó a felek rendelkezésre állása

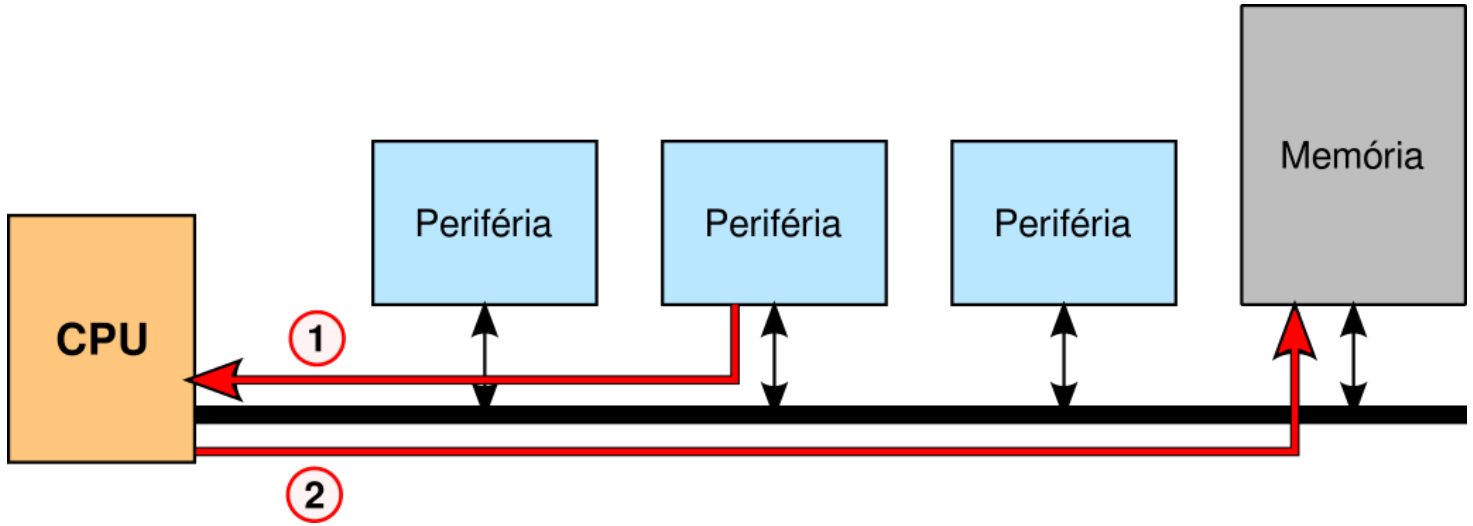


HATÉKONY ADATÁTVITEL

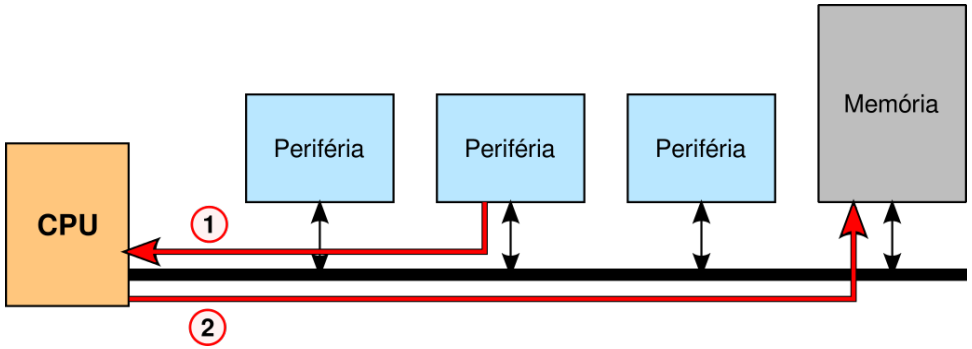
- Végső cél:
  - Adatok átvitele a perifériából a memóriába
- Kérdések:
  - Mi van, ha a küldő és a fogadó sebessége eltérő?
    → Forgalomszabályozás √
  - **Milyen útvonalon juttassuk az adatokat a memóriába?**

HATÉKONY ADATÁTVITEL

- Végső cél:
  - Adatok átvitele a perifériából a memóriába
- Milyen útvonalon juttassuk az adatokat a memóriába?
- 1. megoldás: **a processzoron keresztül**



HATÉKONY ADATÁTVITEL

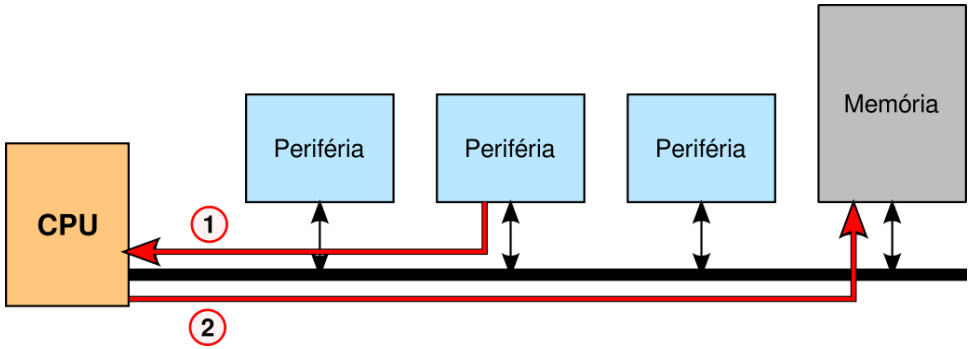


- **Polling:**
  - A periféria állapotát folyamatosan figyeljük:

```
ciklus: R0 ← IO[0x64]
        JUMP ciklus IF R0==0
        R0 ← IO[0x60]
        MEM[0x142] ← R0
```

  - Kritikus kérdés: polling periódus
    - Túl gyakori → nagy CPU terhelés
    - Túl ritka → adatvesztés

HATÉKONY ADATÁTVITEL



- **Megszakításra alapozott adatátvitel:**
  - A periféria készenlétét megszakítással detektáljuk
  - Megszakításkezelő szubrutin:

```
billkezelo: PUSH R0
            R0 ← IO[0x60]
            MEM[0x142] ← R0
            POP R0
            RET
```

  - Csak annyira terheli a CPU-t, amennyire muszáj

POLLING VS. INTERRUPT

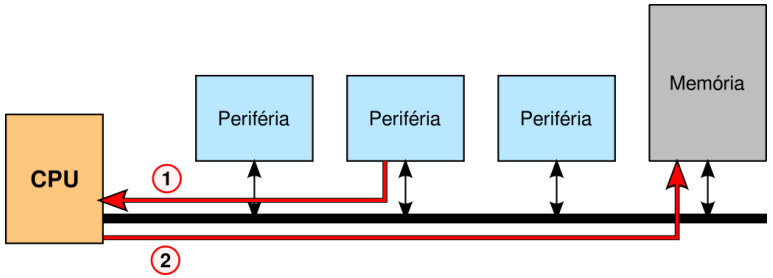
- Példák pollingra (CPU: 1 GHz, 1 lekérdezés 600 órajel)
  - Egér:
    - másodpercenként 30-szor kérdezzük le
    - 30 poll/s * 600 órajel/poll = 18000 órajel/s
    - CPU: 10^9 órajel/s → 18000 / 10^9 = 0.0018%, OK.
  - Diszk:
    - Interfész: $100 \cdot 10^6$ byte/s, 500 byte/blokk
    - Polling periódus: $100 \cdot 10^6$ byte/s / 500 byte/blokk = 200 000 poll/s
    - 200 000 poll/s * 600 órajel/poll = 120 000 000
    - CPU: 10^9 órajel/s → $120 \cdot 10^6$ / 10^9 = 12%
    - Megengedhetetlenül nagy: egyetlen periféria egyetlen jelzése!

POLLING VS. INTERRUPT

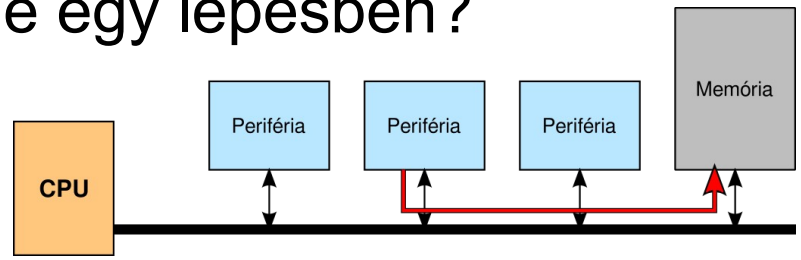
- Példa interrupt-ra alapozott adatátvitelre:
  - Diszk
    - A diszk az idő 10%-ában aktív
    - Interrupt feldolgozási idő: 600 órajel
    - Tényleges adatátviteli idő: 100 órajel
    - Interrupt feldolgozásra fordított idő:
    - $0.1 \cdot (100 \cdot 10^6$ byte/s / 500 byte/blokk · 600 órajel/interrupt)
      = 12 000 000 órajel/s
    - CPU: 10^9 órajel/s → $12 \cdot 10^6 / 10^9 = 1.2\%$
    - Adatátviteli idő:
    - $0.1 \cdot (100 \cdot 10^6$ byte/s / 500 byte/blokk · 100 órajel/átvitel)
      = 2 000 000 órajel/s
    - CPU: 10^9 órajel/s → $2 \cdot 10^6 / 10^9 = 0.2\%$
    - Összesen: 1.2 % + 0.2% = 1.4%

A PROCESSZOR TEHERMENTESÍTÉSE

- Periféria → memória átvitel az eddigiek szerint:



- Nem lehetne egy lépésben?

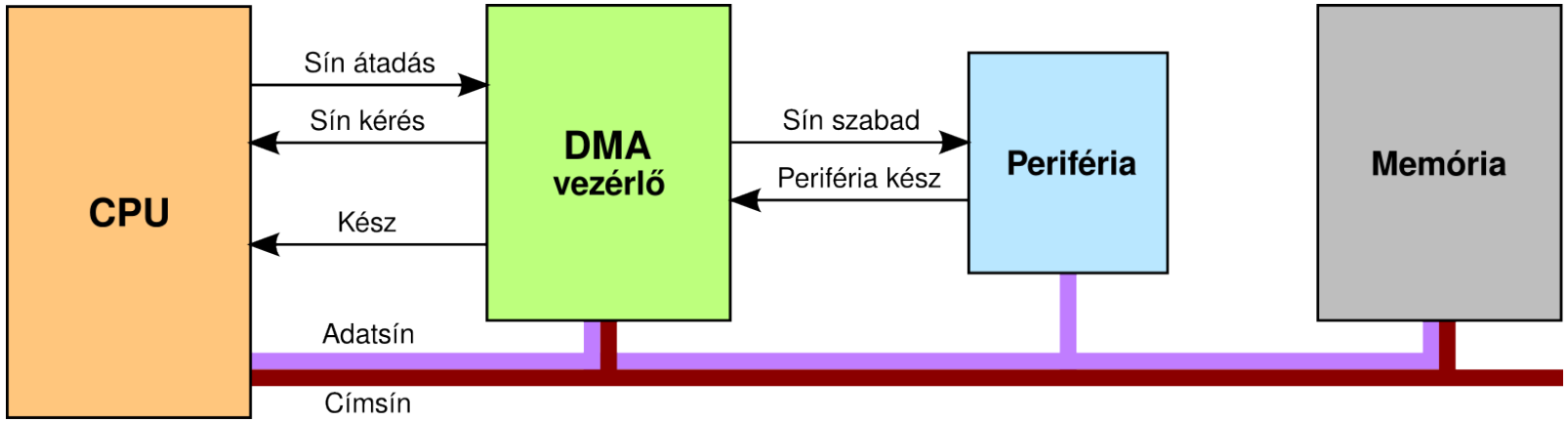


  - Gyorsabb lenne
  - CPU közben mást csinálhatna
  - Megoldások:
    - DMA
    - I/O processzor

DMA

- Periféria → Memória a CPU megkerülésével
  1. DMA vezérlő felprogramozása:
     - Melyik csatornán lévő perifériától,
     - melyik memória kezdőcímtől,
     - írjon-e, vagy olvasson,
     - hány adategységet kell átvinni.
  2. DMA vezérlő koordinálja az adatátvitelt
     - Memóriabusz használati jogának megszerzése
     - Adatátvitel forgalomszabályozással
       → CPU szerepét játssza
  3. A DMA vezérlő megszakítással jelzi, hogy végzett

DMA



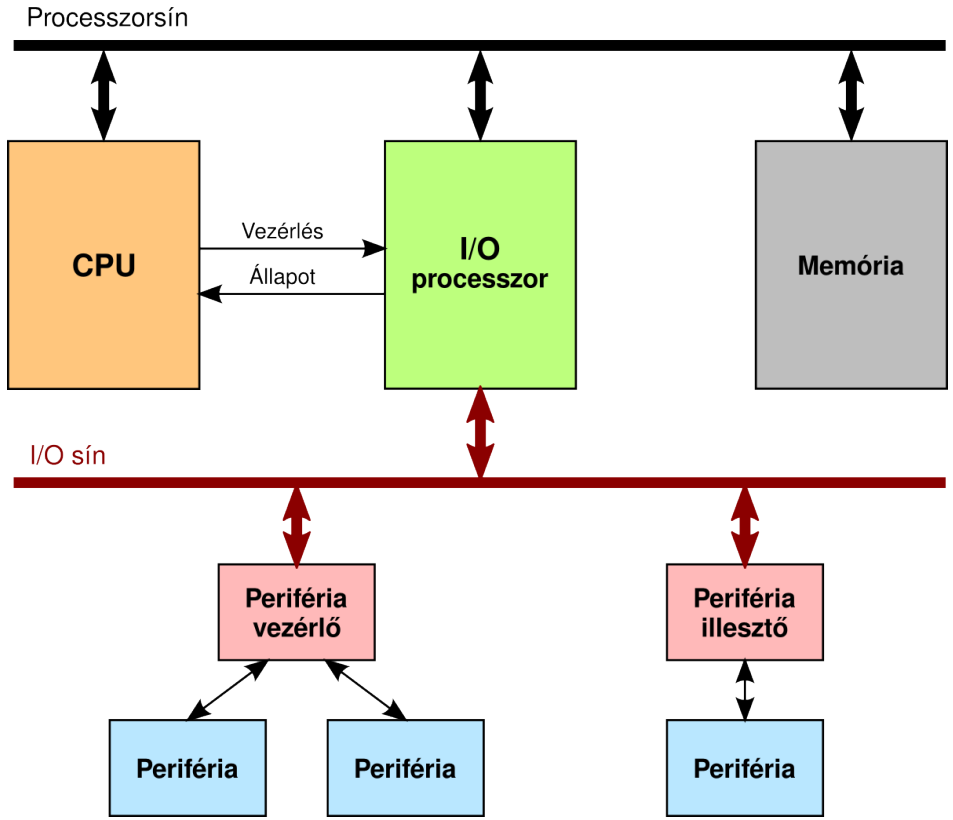
- **A CPU-nak csak a blokkos átvitel kezdeményezésekor és a teljes blokk átvitelének végén van dolga**
- Ez a rendszerszintű (third party) DMA vezérlő
- Ma már nem használják
- Amit ma használnak: first party DMA vezérlő
  - A perifériáknak saját DMA vezérlője van
  - Önállóan tudnak versenyezni a buszért
    - Egymással és a CPU-val
  - Aki nyer, az átviheti az adatait a memóriába

I/O PROCESSZOR

- A rendszerszintű DMA koncepció továbbfejlesztése
- Az I/O processzor **saját utasításkészlet**tel is rendelkezik
- I/O program:
  - Periféria és memóriaműveletek sorozata
  - Egyszerű adatfeldolgozási műveletek
    - CRC/paritás ellenőrzés
    - Tömörítés/kicsomagolás
    - Bájtsorrend konverzió
    - Stb.
- Végrehajtás:

  1. CPU odaadja az I/O processzornak az I/O program memóriabeli kezdőcímét
  2. Az I/O processzor kiolvassa, és sorról sorra végrehajtja
  3. A program végén megszakítással jelzi a CPU-nak, hogy kész

I/O PROCESSZOR



- A CPU minden perifériaműveletet az I/O programmal ad meg
  → **perifériafüggetlenség!**
- A periféria illesztő/vezérlő feladata:
- Perifériaspecifikus viselkedés ↔ I/O sín protokoll fordítás

PERIFÉRIÁK

- Kérdések:
  - Hogyan tud a processzor adatátvitelt kezdeményezni? √
  - Hogyan tudnak a perifériák adatátvitelt kezdeményezni? √
  - Hogyan lehet az adatátvitelt hatékonyan, hibamentesen levezényelni? √
  - **Hogyan/hová kössük a perifériákat a számítógéphez?**

ÖSSZEKÖTTETÉSEK

- Hogy kössük össze egymással a
  - processzort,
  - memóriát,
  - perifériákat?
- Eddig: memória & perifériák közvetlenül a CPU buszon
- Máshogy is lehet.

BUSZ VS. PONT-PONT

- **Pont-pont** összeköttetések
  - Felek között dedikált csatorna
    - nincs osztozkodás, versengés, várakozás → gyorsabb
    - ahány eszköz, annyi pont-pont összeköttetés → drága
- **Busz alapú** összeköttetések
  - Osztott csatorna
    - osztott erőforrás → szűk keresztmetszet lehet (torlódás, egymás feltartása, stb.)
    - egy buszra kapcsolódik mindenki → olcsóbb
  - Az osztott erőforrással való gazdálkodás speciális megoldásokat igényel

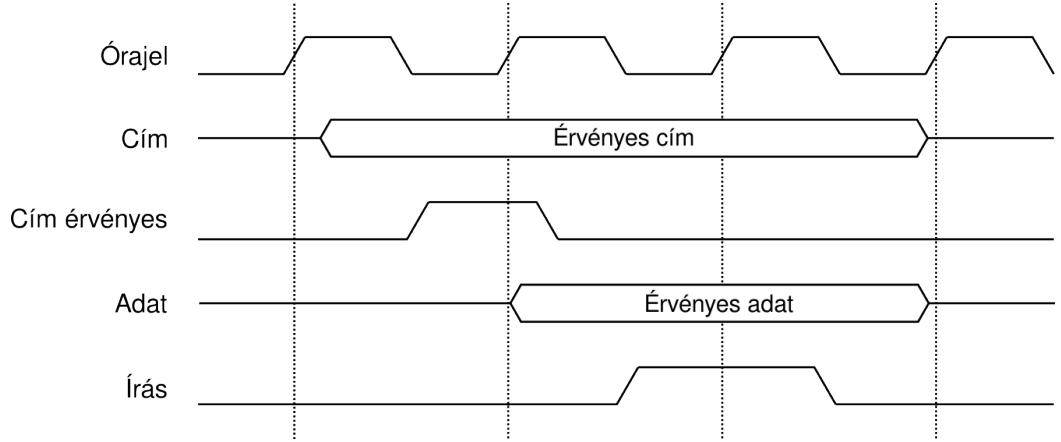
ÖSSZEKÖTTETÉSEK SZÉLESSÉGE

- Szélesség = a jelvezetékek a száma
- Széles összeköttetés:
  - Több bit vihető át egyidejűleg → gyorsabb lehet
  - Drágább
- **Ellentmondás: a leggyorsabb buszok sorosak!**
  - Széles busz → sok jelvezeték
  - A jelvezetékek hossza és elektromos tulajdonságai nem egyformák

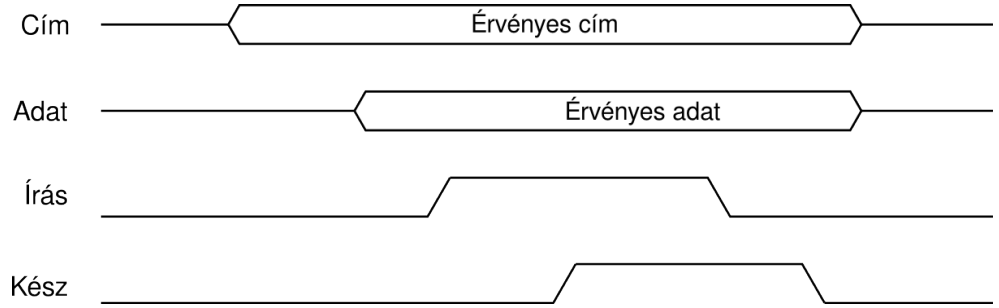
    → az egy időben küldött jelek egymáshoz képest csúszva érkeznek meg!
  - Akkor okoz gondot, ha a csúszás összemérhető az órajellel
- Trend: soros átvitelt mindenhova → nincs csúszás, ami korlátozza a sebességet

IDŐZÍTÉS

- **Szinkron:**
  - Közösen látott órajel
  - Adatok érvényességét az órajelhez kötik



- **Aszinkron:**
  - Nincs órajel
  - Adatok érvényességét a vezérlőjelek határozzák meg



- Szinkron – aszinkron: **melyik jobb?**

IDŐZÍTÉS

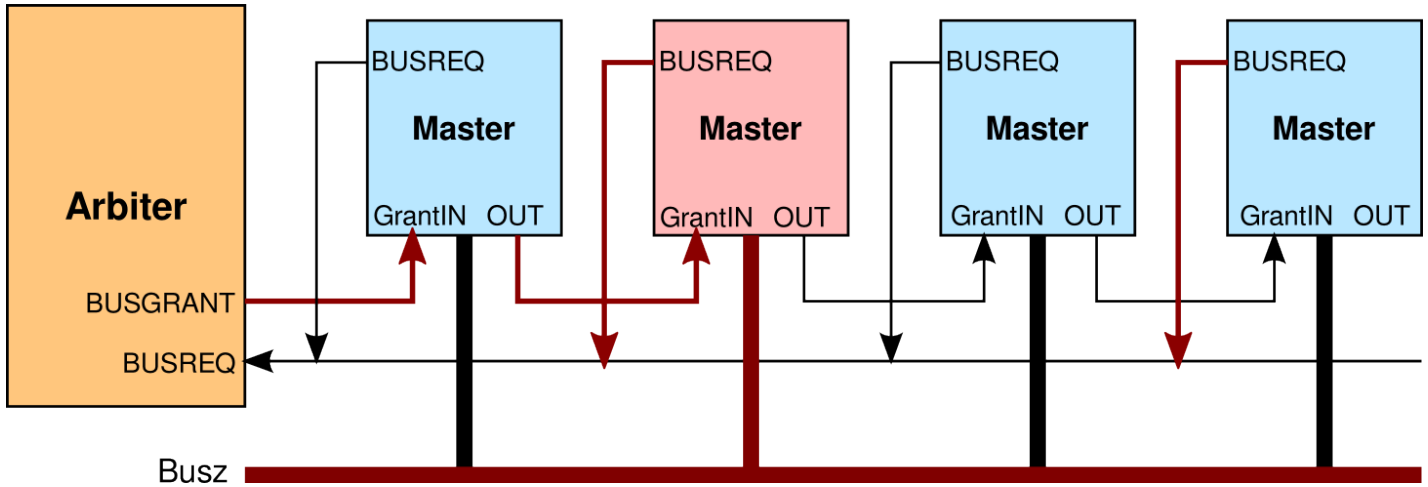
- Egyik sem:
  - **Adat vezeték – órajel vezeték között csúszás léphet fel!**
- Megoldás:
  - Ne legyen órajel
  - Ne legyenek adat érvényességére vonatkozó vezérlőjelek
  - Csak 1 szál drót az adatnak (soros átvitel)
- Honnan tudja a periféria, hogy hol vannak a bithatárok?
  - Jöjjön rá a 0-1 bitek váltakozásából
    → önidőzitő adatátvitel
  - Csak akkor megy, ha van 0-1 váltakozás
- **8b/10b kódolás**:
  - Csinál 0 – 1 átmenetet (csupa 0 – csupa 1 sorozat max. hossza: 5)
  - 20 hosszan max. 2 lehet a 0-k és 1-k száma közt a különbség
  - Gigabit Ethernet, PCI Express 1.0/2.0, USB 3.0, SATA, HDMI, DVI, stb.
  - Hátrány: overhead
  - Továbbfejlesztés:
    - 64b/66b: 10 GB Ethernet, 100 GB Ethernet
    - 128b/130b: PCI Express 3.0/4.0/5.0, USB 3.1, SATA 3.2, DisplayPort 2.0
    - 256b/257b: Fibre Channel

ARBITRÁCIÓ A BUSZON

- Busz: osztott erőforrás
  - Meg kell oldani, hogy egyszerre csak egyvalaki használja
- Szereplők a buszon
  - **Bus Master**: az az eszköz, ami képes a buszon önálló módon adatátvitel lebonyolítani
  - **Bus Slave**: nem képes
- A busz osztott erőforrás
  - Egyidejűleg több Master is jelenthet be igényt
  - Csak egy nyerhet
- **Arbitráció**
  - A buszért zajló verseny eredményének eldöntése

CENTRALIZÁLT ARBITRÁCIÓ

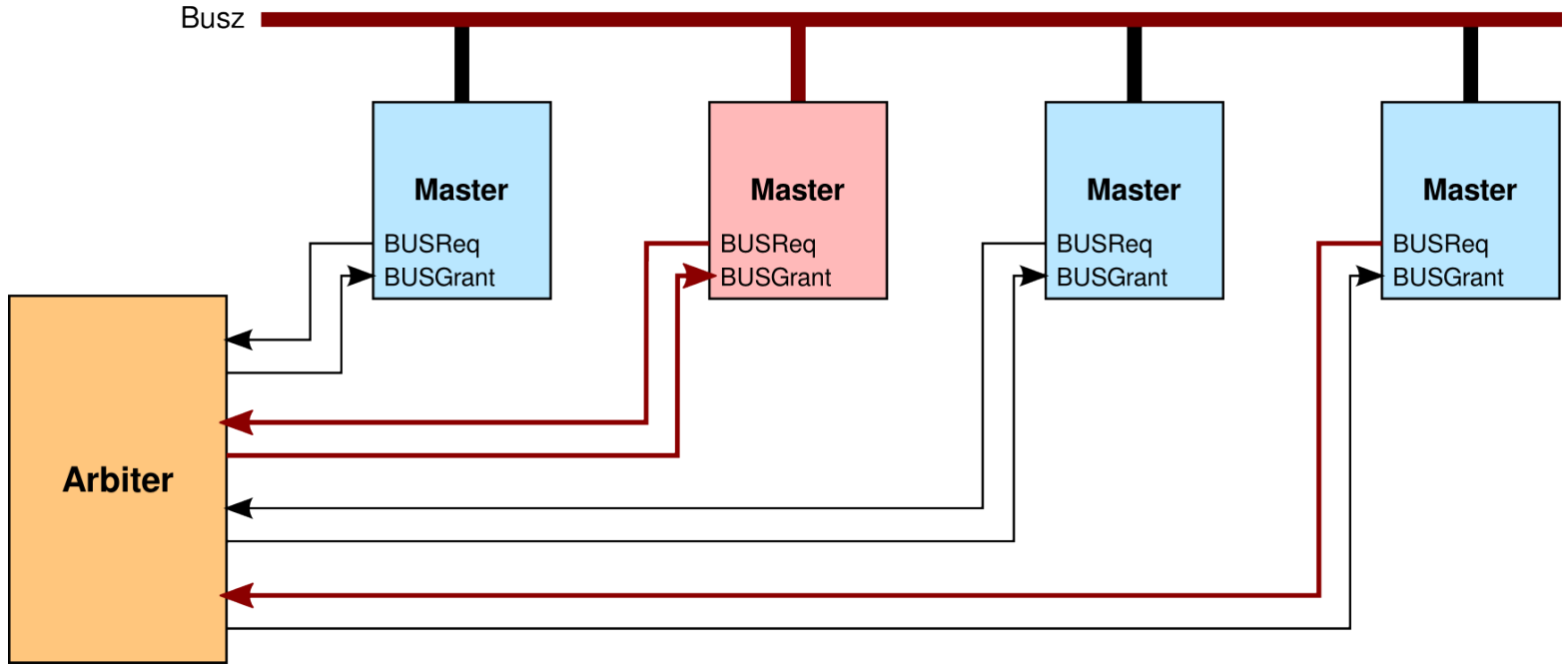
- Speciális egység: **Arbiter**
  - Eldönti, ki kapja a buszhasználat jogát
- Soros arbitráció: **Daisy Chain**



- Előny:
  - Könnyen bővíthető
- Hátrány:
  - Nem fair

CENTRALIZÁLT ARBITRÁCIÓ

- **Párhuzamos** centralizált arbitráció
- Rugalmasabb priorizálás
  - Körbenforgó
  - Késleltetésérzékeny perifériák gyakrabban nyernek
  - Stb...

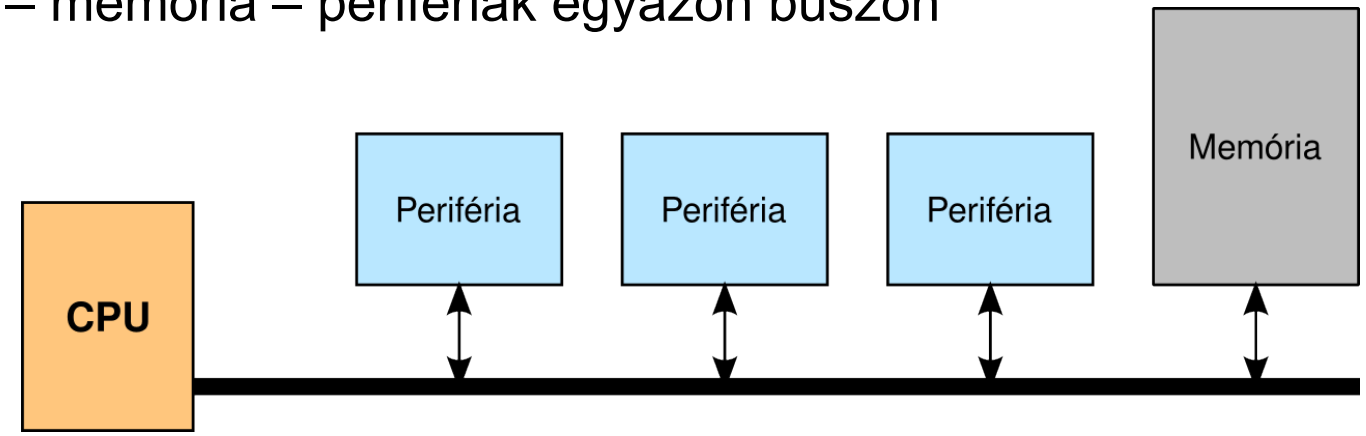


ELOSZTOTT ARBITRÁCIÓ

- Nincs arbiter
- **Önkiválasztó arbitráció** (pl. SCSI):
  - Mindenki látja a többiek buszfoglalási kéréseit
  - Mindenki tisztában van a saját és a többiek prioritásával
  - A legnagyobb prioritásún kívül mindenki visszavonja igényét
- **Ütközésdetektáláson alapuló** busz használat
  - Nincs is arbitráció
  - Ha valaki használni akarja a buszt, rögtön neki is lát
  - Adatátvitel közben hallgatózik a buszon
  - Ha tisztán kivehető a saját „adása”, akkor jó
  - Ütközés esetén zagyvaságot hall → elhallgat, később újra próbálkozik
- Elosztott arbitráció előnye:
  - Nincs arbiter, ami, ha elromlik, megáll az élet

EGYBUSZOS RENDSZEREK

- CPU – memória – perifériák egyazon buszon

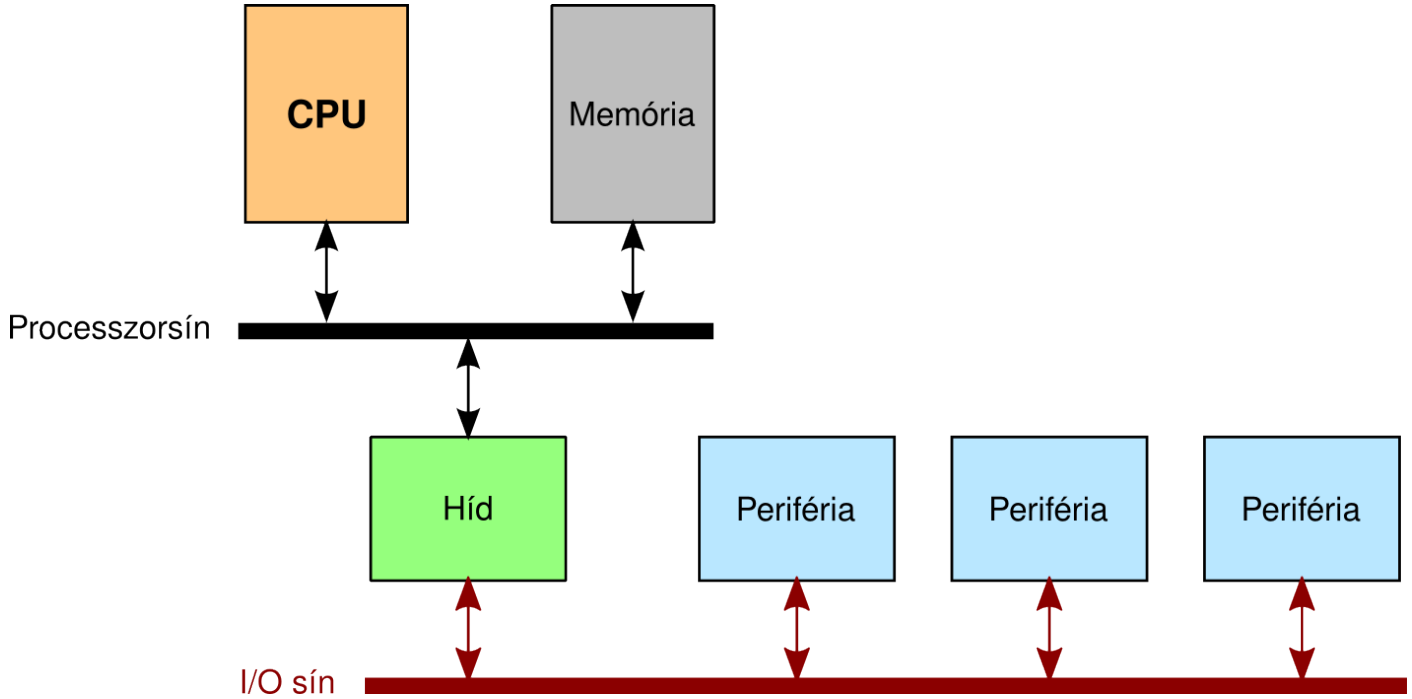


- Egyszerű megvalósítani
- Hátrány:
  - Magas CPU órajel → magas busz órajel → drágább perifériák → korlátozott fizikai távolság
  - A CPU órajele / busz órajele és a busz protokoll típustól függően változik

    → A perifériák állandó működési környezetet igényelnek
    - Nem szeretnénk kidobni a perifériákat, ha új CPU-t veszünk

KÉTBUSZOS RENDSZEREK

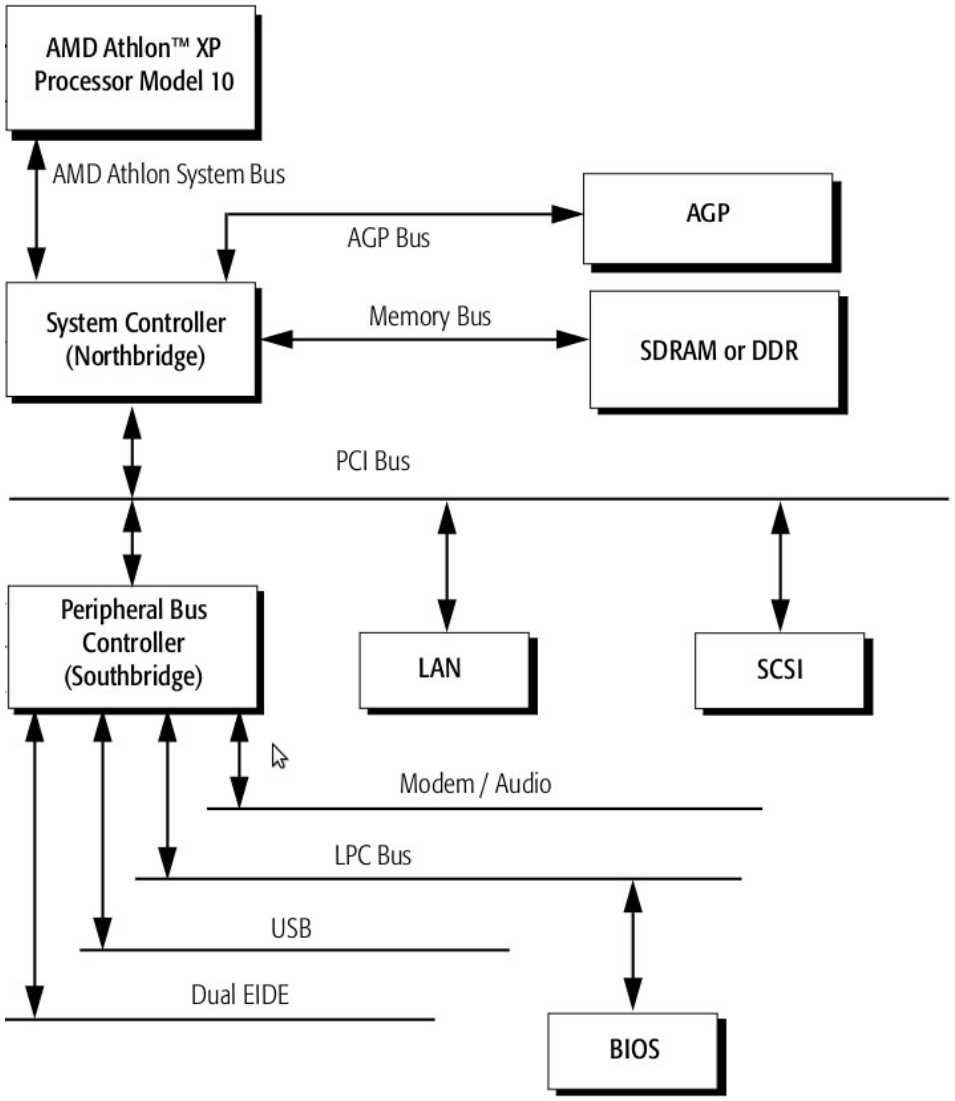
- Perifériák külön buszon, CPU-memória egy másik buszon
- Átjárás: **híd**



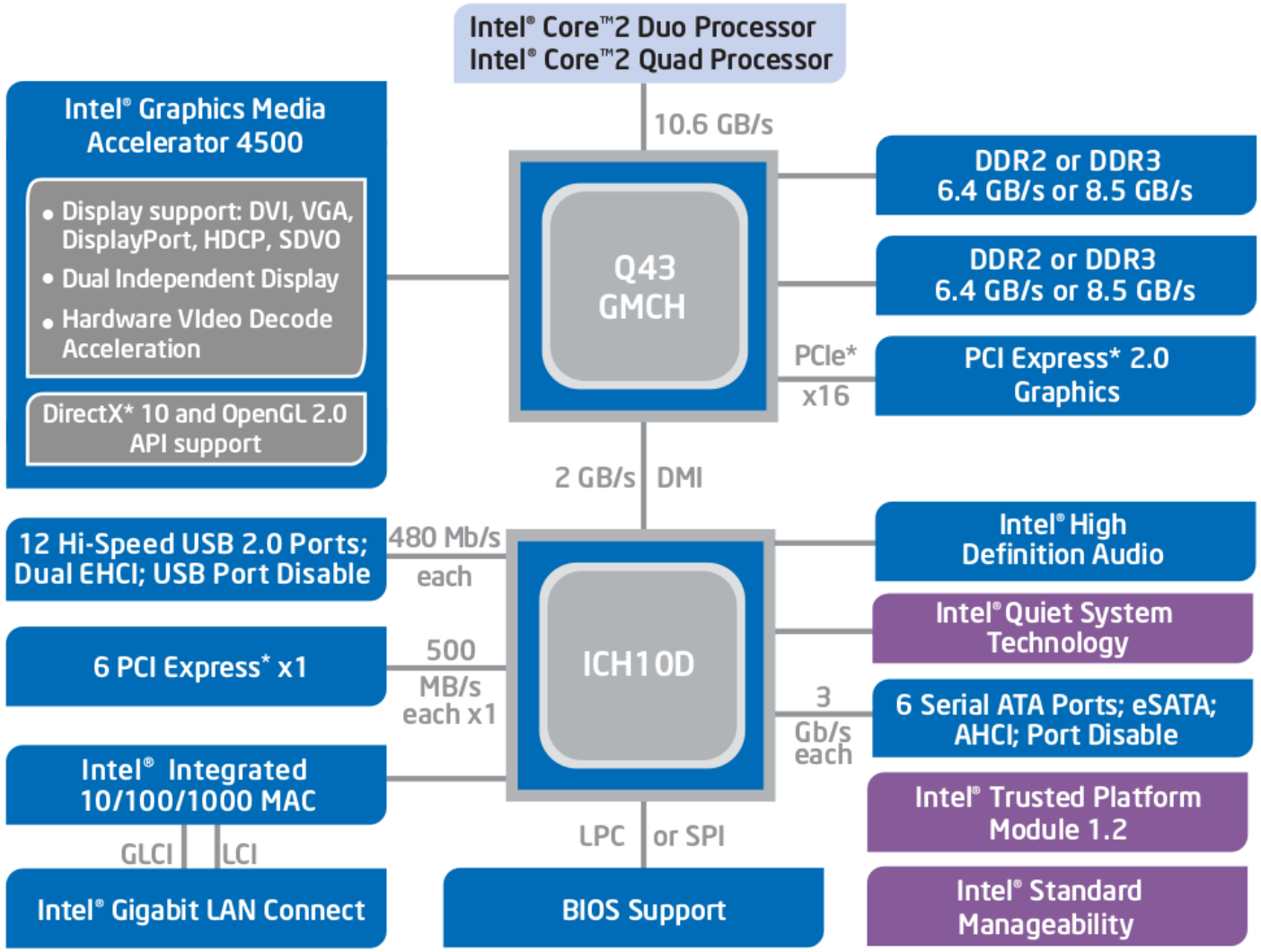
- Processzorsín: típusfüggő sebesség és protokoll
- I/O sín: állandó, szabványos sebesség és protokoll

HÍD ALAPÚ RENDSZEREK

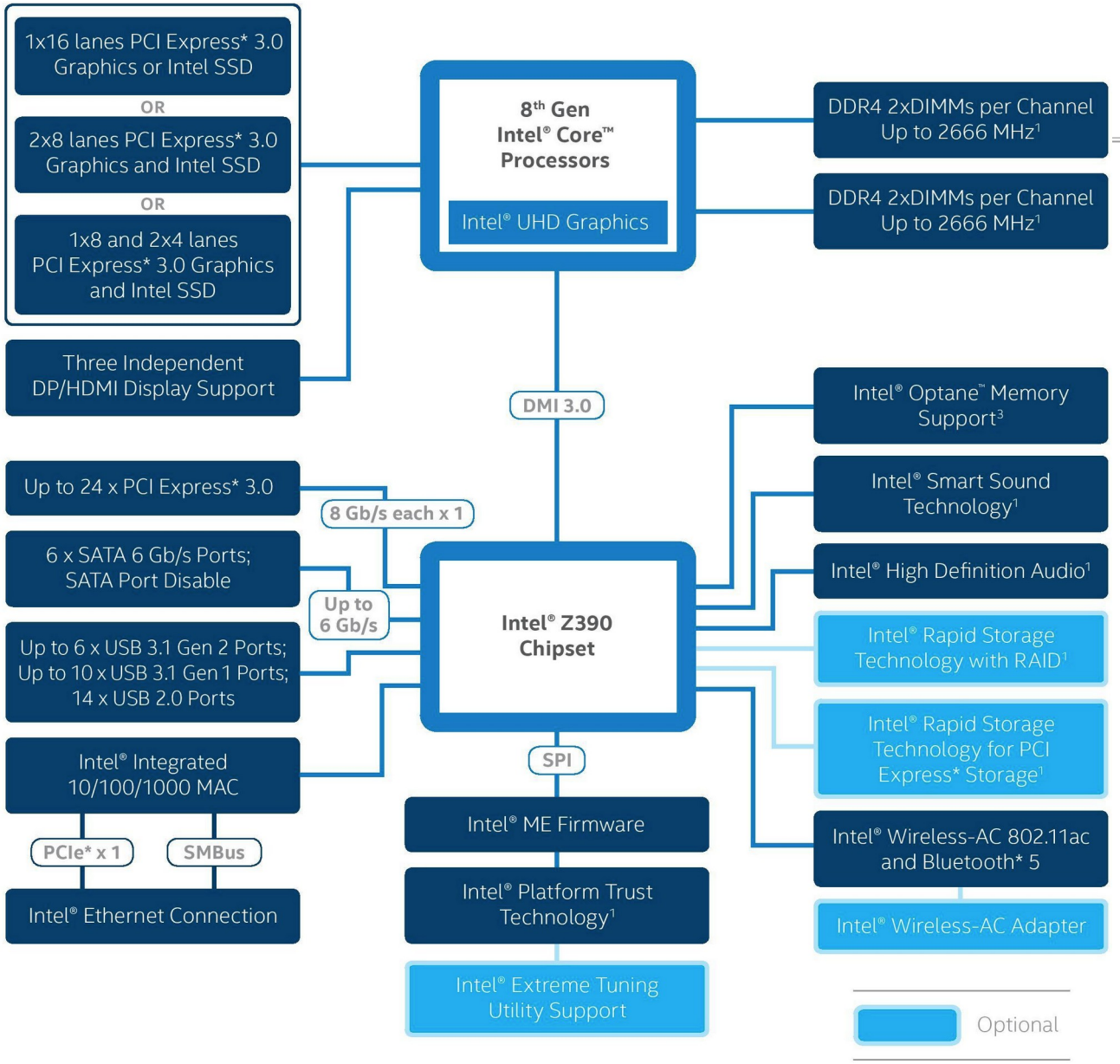
- Külön buszon:
  - CPU
  - Memória
  - Perifériák
- Processzorsín:
  - Típusfüggő
- Memóriasín:
  - Állandó, szabványos
- Perifériasín (PCI):
  - Állandó, szabványos



HÍD ALAPÚ RENDSZEREK



HÍD ALAPÚ RENDSZEREK



HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK