PANNON EGYETEM, Veszprém
Villamosmérnöki és Információs Rendszerek Tanszék



Számítógép Architektúrák
(MIKNB113A)

7. előadás: Mikroszámítógépek - I/O operációk I. aszinkron-, szinkron- protokollok, arbitráció.

Előadó: Dr. Vörösházi Zsolt
voroshazi.zsolt@virt.uni-pannon.hu

Jegyzetek, segédanyagok:

- Könyvfejezetek:

http://www.virt.uni-pannon.hu → Oktatás → Tantárgyak → Számítógép Architektúrák (Gazdaságinformatikus – Programozó Info.)

(chapter06.pdf)

- Fóliák, óravázlatok .ppt (.pdf)
- Feltöltésük folyamatosan

I/O műveletek

- **Aszinkron protokoll**
- **Szinkron protokoll**
- **Arbitráció (döntési mechanizmus)**
- Megszakítás – kezelés (Operációs Rendszerek)
- Buszok – Buszrendszerek:
  - PCI,
  - PCI-Express,
  - SCSI buszok

I/O egységek

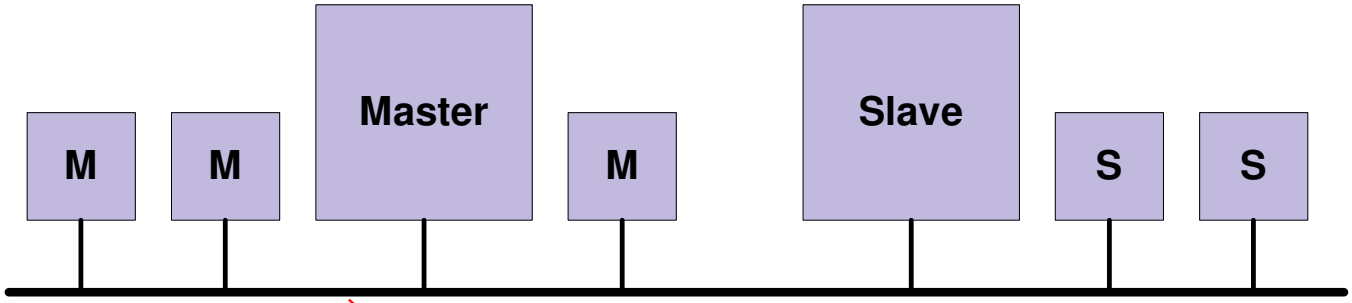
- A számítógép a külvilággal, *perifériákkal* az I/O egységeken keresztül tartja a kapcsolatot. Az információ továbbítását az egységek között buszok végzik, amelyek interfészekkel kapcsolódnak egymáshoz.
- **Interface**: azon szabályok összessége, amelyek mind a fizikai megjelenést, kapcsolatot, mind pedig a kommunikációs folyamatokat leírják. Egy busz általában 3 fő kommunikációs vonalból állhat:
  - *vezérlőbusz,*
  - *adatbusz, és*
  - *címbusz*.

I/O kommunikációs protokollok:

- Kommunikáció során általában egy eszközpárt különböztetünk meg: a *Master*-t és *Slave*-t. A Master (pl. CPU) általában, mint kezdeményező, birtokolja és ellenőrzi a buszt, és átadja (ír / olvas) az adatokat a Slave-nek (pl. Memória).
- A kommunikációhoz előredefiniált **protokollokra** (szabályok és konvenciók gyűjteménye) van szükség, amelyek meghatározzák az események sorrendjét és időzítését. A kommunikáció feltétele a másik egység állapotának pontos ismerete. Lehetséges több M-S modul (pl. multi-master rendszer több kezdeményezővel) is egy rendszerben.
- **Két alapvető protokoll különböztethető meg:**
  - *1.) Aszinkron* kommunikációs (pl. SCSI busz), és
  - *2.) Szinkron* kommunikációs (pl. PCI busz) protokoll.

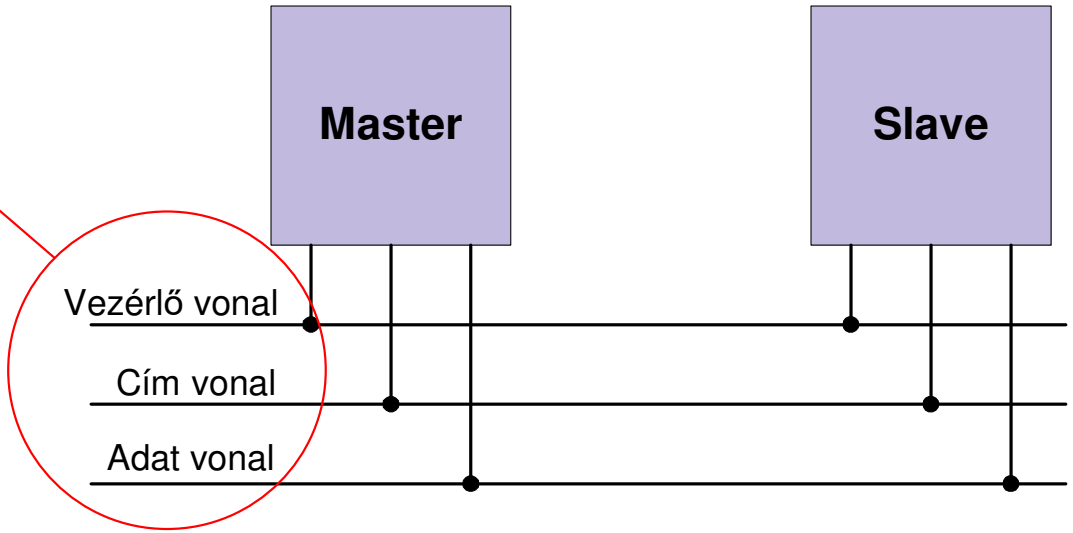
Busz rendszer Master-Slave moduljainak szervezése

- Modul-szervezés



Busz-Master (busz-vezérlő) kommunikál a Busz-Slave-el (responder).

- Busz-szervezés: Master - Slave



1.) Aszinkron busz protokollok

Aszinkron busz protokollok

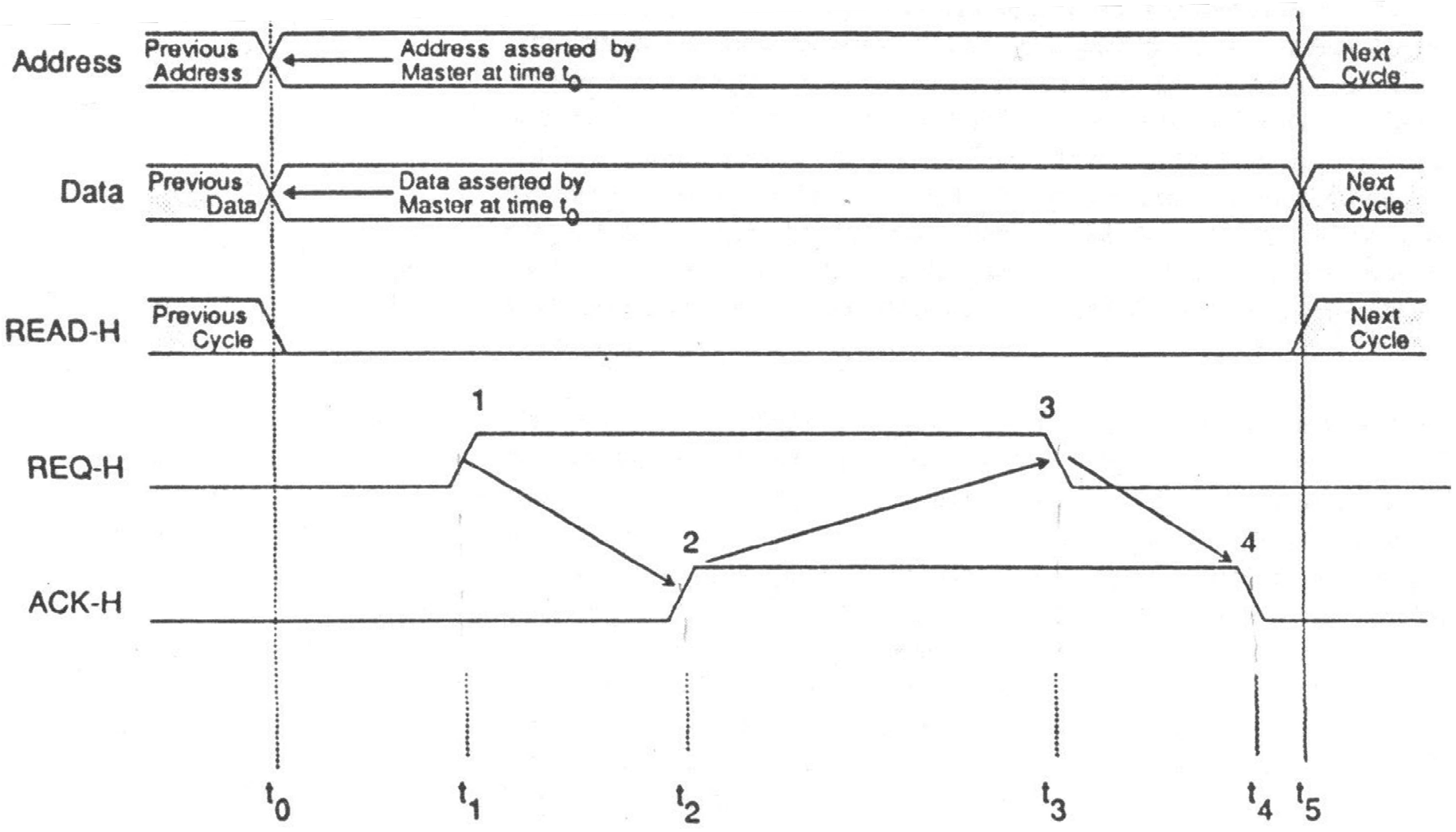
- Aszinkron handshaking: ”2x-es kézfogás”
- Ebben az esetben a Master-Slave modulok **nem** közös órajelet (SysCLK-t) használnak: azaz ha az egyik egység végez, elindít egy másik tranzakciót. A Master, mint (commander) kezdeményező, aktiválja a megfelelő vezetékeket. A Slave (responder) válaszol. A Slave címének azonosítására egy külön egység szolgál.
- **Vezérlőjelekkel** zajlik a kommunikáció: *írási / olvasási* tranzakciókat különböztetünk meg. Ilyen vezérlőjelek:
  - READ, (ha pl. READ-H = Master olvas a Slave-től, ha READ-L= Master ír a Slave-nek, vagyis a Slave olvas.),
  - REQUEST (REQ)
  - ACKNOWLEDGE (ACK)
- *Előnye***:** egyszerűen felépíthető, nem kell (közös) órajel, gyors átvitelt biztosít (modulok sebességétől függően)
- *Hátránya*: belső késleltetések (skew-propagation time), ill. csak korlátozott hosszúságú buszok /vezetékek használhatóak.

Aszinkron Handshaking protokoll (kétszeres „kézfogás”)

- A buszrendszer moduljai nem közös órajellel működnek, hanem **vezérlő jelek** segítségével. (READ-H, REQ, ACK)
- Háromféle buszvonalat ismerünk: *cím-, adat- és vezérlő*- buszt. A Master által *címvonal*ra rakott cím (kezdeményezés) egyértelműen meghatározza a tranzakció célállomását. Meghatározott idő áll rendelkezésre, hogy a slave modulok összehasonlítsák saját címükkel a célcímet. Ha megegyezik, akkor válaszol a master-nek, és a tranzakciót a *vezérlővonalak* megfelelő beállításaival szabályozza. A vezérlővonalakat tehát a master-slave közötti kommunikáció szinkronizálására használjuk. Ezt nevezzük **handshaking** protokollnak.
- Címvonalak csoportjából (Address), adatvonalak csoportjából (Data), és három vezérlőjelből (READ-H, REQ-H, ACK-H) áll. Ha READ-H magas, akkor a Master olvas a Slave-ről, ha alacsony, akkor ír a Slave-re. A REQ (kérés) és ACK (nyugtázás) vezérlőjelek határozzák meg az események időzítését és pontos sorrendjét.

Handshaking – írási ciklus

WRITE (READ_L) ciklus: „a Master modul ír a Slave-nek”

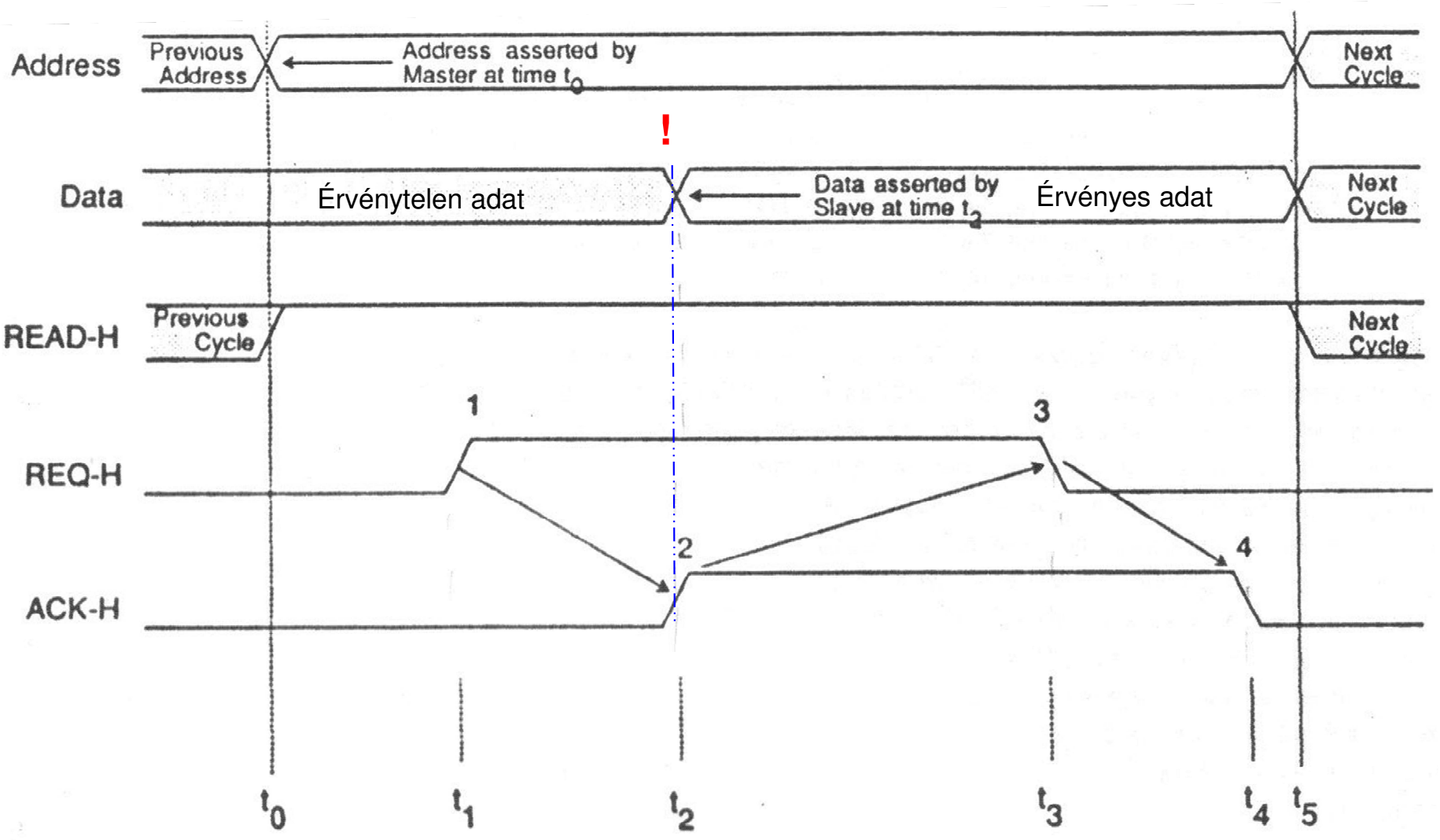


Írási (Write) ciklus lépései:

- t_o-ban a master (amely az aktuális arbitráció után már megkapta a busz vezérlését) megadja a kívánt slave címét.
- véges idő kell hogy a jel a slave modulokhoz érjen, azok dekódolják a címet, ezért a master vár bizonyos ideig mielőtt beállítja a request vonalat t_1-ben. (*skew time*= a slavekhez elsőként ill. utolsóként érkező címek közötti időkülönbség $\Delta t=t_1-t_0$)
- ezt a request jelet minden slave veszi ugyan, de csak az fog válaszolni, akinek a címe megegyezett a master által kért címmel.
- amikor a slave megkapta az adatokat a mastertől, nyugtázza t_2-ben.
- a master megkapja a nyugtát, így tudja, hogy az átvitel megtörtént, ezért felszabadítja (alacsony állapotba helyezi) a request vonalat t_3-ban.
- ezt (a request felszabadítását) érzékeli a slave, és felszabadítja a nyugtázó vonalat t_4-ben
- a master a címvonalat tartja még egy bizonyos ideig (t_5-ig) a request vonal felengedése után is, a *cím esetleges megváltozása* miatt (amelyet a Slavek dekódolnak).

Handshaking – olvasási ciklus

READ (READ_H) ciklus: „a Master olvas a Slave-től”



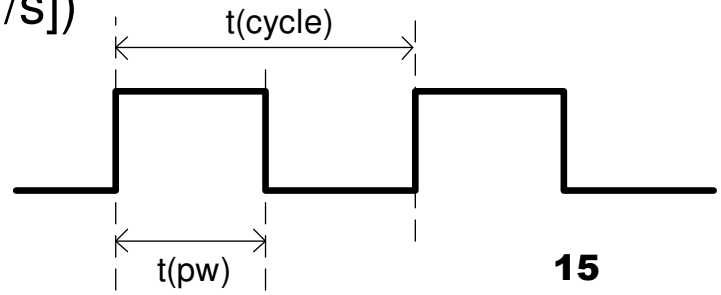
Olvasási (Read) ciklus lépései:

- Nagyon hasonlít az írási folyamathoz, de abban különbözik, hogy a READ-H jel magas szinten van, és az adatvonalat a slave állítja be. Ez jelenti az olvasást.
- t_o hasonlóan történik, a master (amely az arbitráció után már megkapta a busz vezérlését) megadja a kívánt slave címét
- véges idő kell hogy a jel a slave modulokhoz érjen, azok dekódolják a címet, ezért a master vár bizonyos ideig mielőtt beállítja a request vonalat t_1-ben. (*skew time* = a slavekhez elsőként ill. utolsóként érkező címek közötti időkülönbség $\Delta t=t_1-t_0$). Tehát t_1-ben a master a request vonal beállításával a megcímzett slave-től adatot kér, olvasni szeretne
- t_2-ben veszi a kérést a slave, nyugtázza és beállítja magas szintre a nyugtázó vonalat.
- t_3-ban a master megkapja az adatot a slave-től, felszabadítja a request vonalat.
- t_4-ben érzékeli a slave, hogy a master felszabadította a requestet, ezért így ő is felszabadítja a nyugtázó vonalat.
- végül a master felszabadítja a címvonalat (t_5-ben)

2.) Szinkron busz protokollok

Szinkron busz protokollok

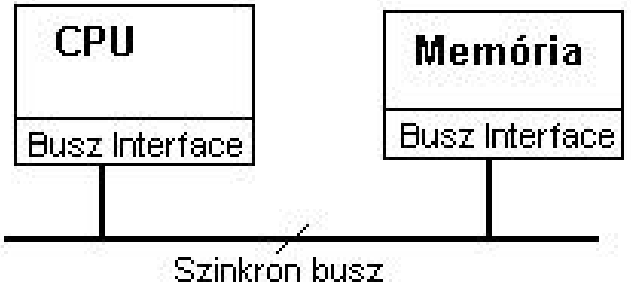
- Ebben az esetben a Master-Slave modulok **közös órajelet** (CLK-t): használnak. Itt a Master, mint Commander (kezdeményező), a Slave pedig (Responder) válaszol.
- *Előnye*: Tehát a műveleti időt az órajelciklus határozza meg. Mivel nincs szükség párbeszédre (pl. handshaking), gyorsabb lesz az aszinkron működésénél.
- *Hátránya*: Az órajelet mindig a leglassabb (legtávolabb lévő) egységhez kell igazítani.
- A digitális áramkörökben az események történésének *sorrendje kritikus* (megfelelő időzítés kell ⇒ órajel vezérléssel)
- **Óra**: impulzusok sorozatát bocsátja ki, pontosan meghatározott szélességgel [t(pw)], és időintervallummal.
- **Ciklus-idő** (clock-cycle): két egymást követő pulzus élei közötti időintervallum [t(cycle)].
  - Példa: Órajel Frekvencia: f=100 000 000 [Hz] ([1/s])
  - Ekkor T=1/f=1 / 100 000 000 = 10 [ns]
- Kristály-oszcillátor szolgáltatja ált. az órajelet.



Szinkron írási / olvasási ciklus

- **Írási ciklus (pl. CPU → MEM)**

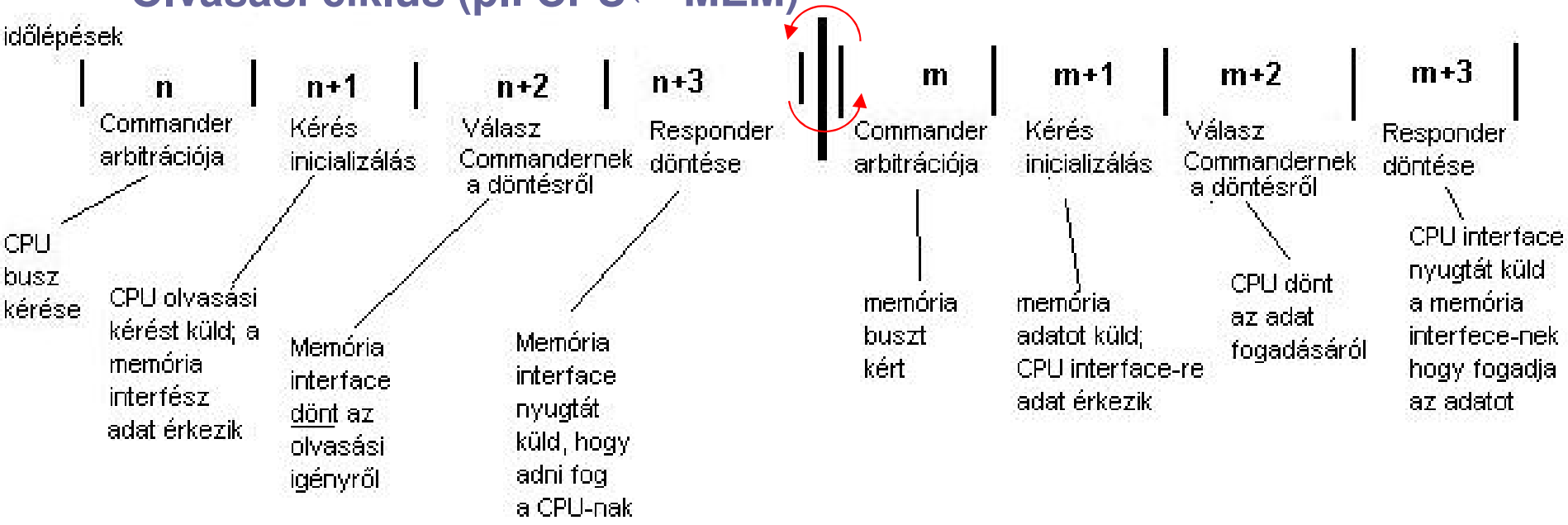




Szinkron adatátvitel 4 fő lépése:

1. **Commander arbitrációja, kiválasztása**
2. **átvitel megkezdése (de *még nem fogad*)**
3. **válasz a kérésre, döntéshozás**
4. **nyugta, Responder *veszi* az adatot**

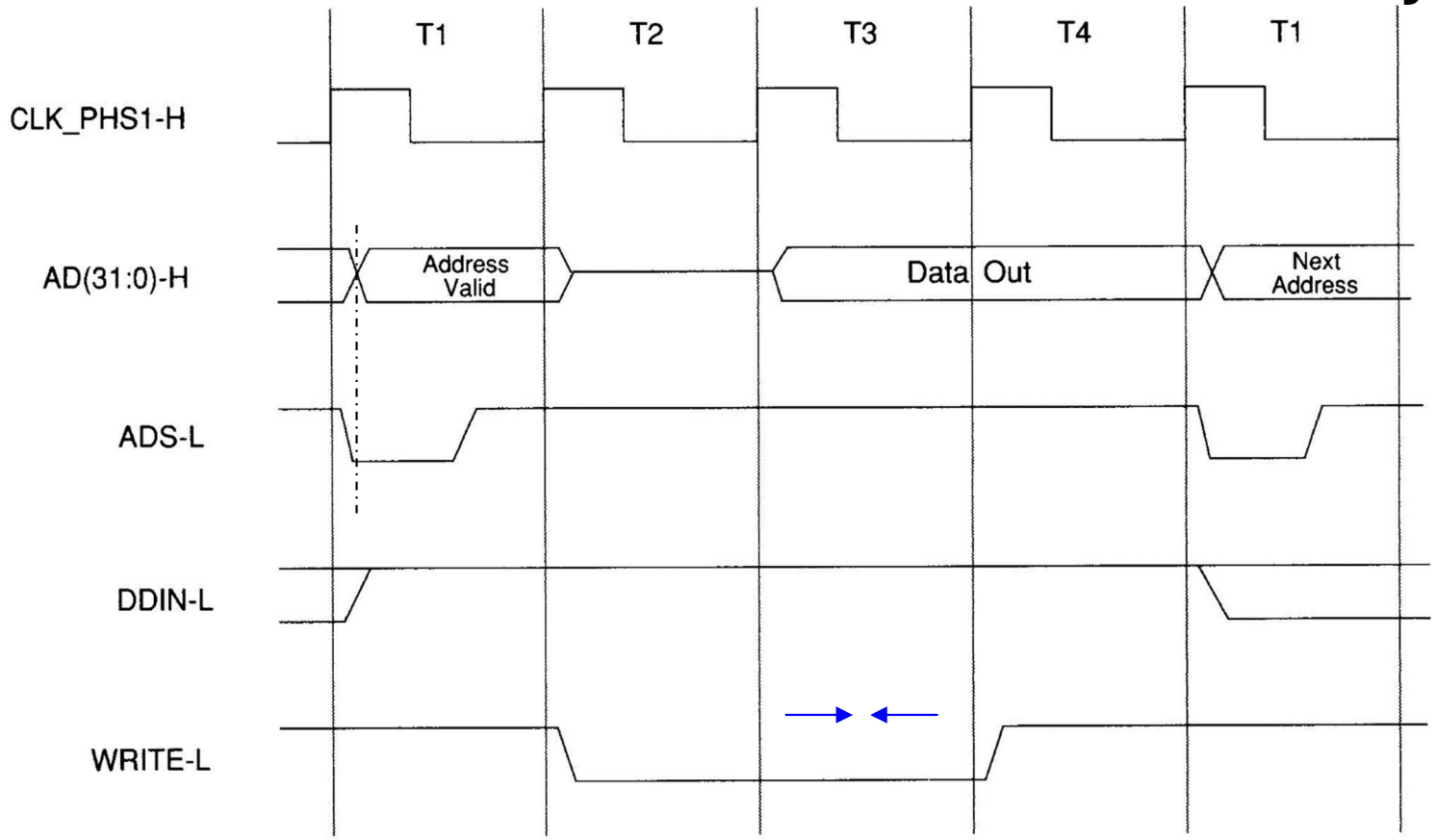
- **Olvasási ciklus (pl. CPU← MEM)**



Példák: Szinkron busz protokollok fajtái

- 1.) **Általános célú busz**: CPU közvetlen vezérlése alatt áll (határozza meg a forrást ill. célt)
- 2.) **Motorola 68020**: nagyteljesítményű rendszer. Dinamikus buszméret. CLK-val szinkron kommunikáció. Ha a Slave rövid időn belül nem képes egy kérést kielégíteni, akkor a CPU automatikusan egy tétlen („idle”) állapotot, *várakozó* ciklust generál, amíg a Slave nem válaszol.
- 3.) **NS32332** buszrendszer: 32-bites szinkron rendszer. Idő-multiplex Addr/Data kommunikáció (IC lábszám csökkentése miatt).
- 4.) **PCI buszrendszer!** (Lásd később)

Példa: NS32332 Busz **írási** tranzakciója



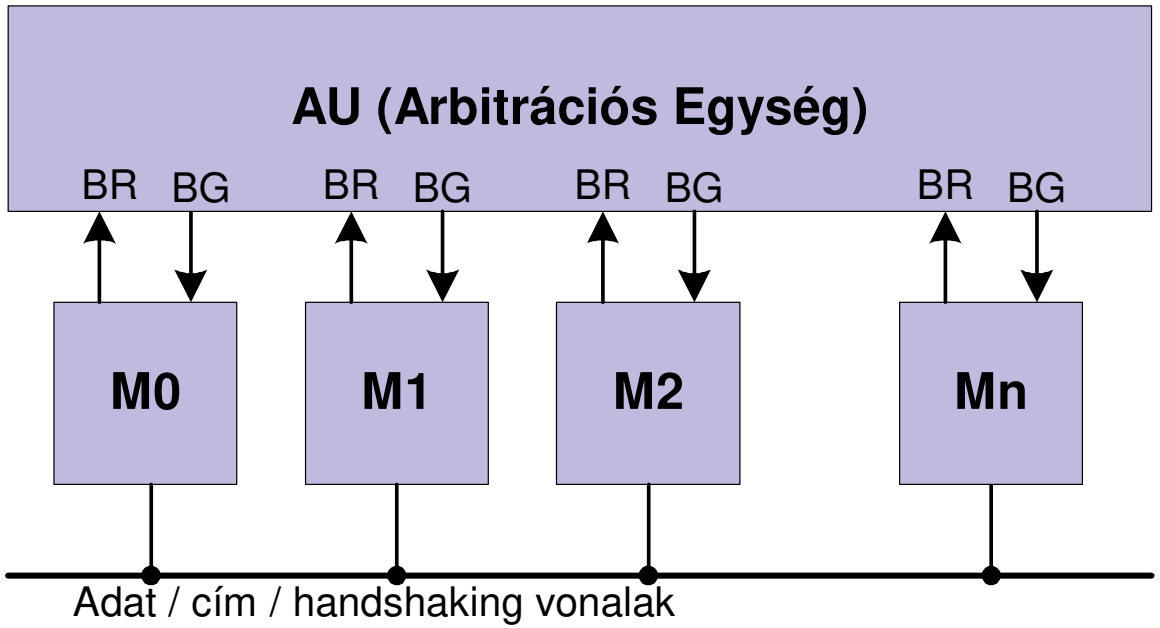
- CLK_PHS1_H: közös rendszer-órajel (felfutó élre)
- AD (31:0)_H : 32-bites multiplexált cím/adat vonal (érvényes adat T_3-ban)
- ADS_L: érvényes cím érkezését jelzi T_1 ciklusban
- WRITE_L: írás engedélyezés T_2-ben
- DDIN_L: adatátviteli irány (DIN: -L: out/olvasás, -H: input/írás)
- Minimális tranzakcióhoz 4 ciklus szükséges (T_1-T_4):

Busz Arbitráció

Arbitráció

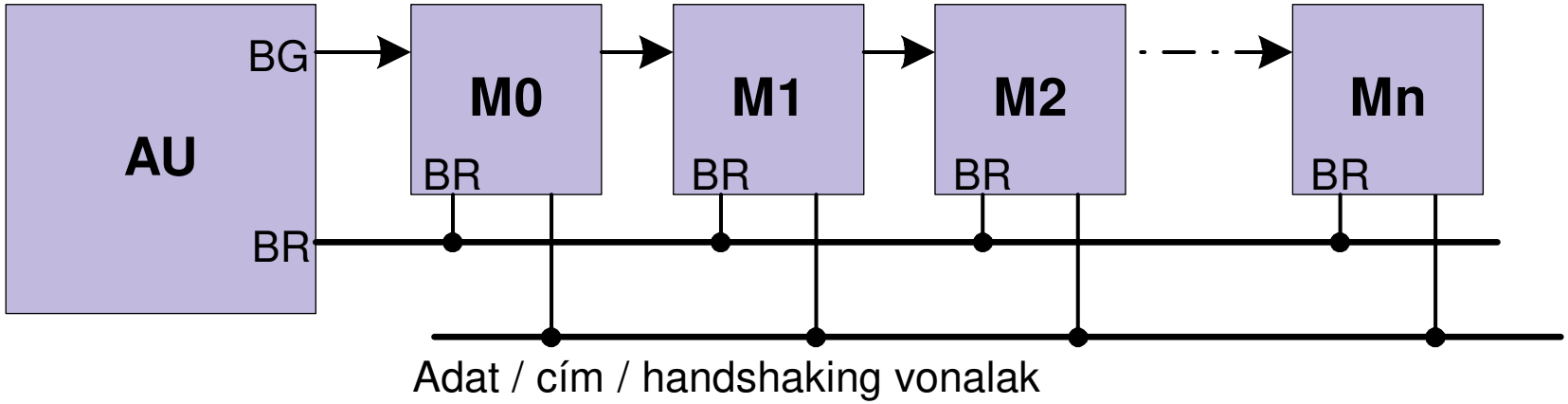
- Egy tetszőleges I/O művelet esetén (aszinkron v. szinkron buszos átvitel) több Master (commander) egység is meg akarja szerezni egyszerre a busz irányítását. Különböző előredefiniált algoritmusok segítségével egyértelműen azonosítható, hogy a „**versenyhelyzetben**” a következő átvitelt (a következő ciklusban) melyik Master fogja megvalósítani. Az ***arbitrációs eljárás*** egy **döntési folyamat (mechanizmus),** amely az adatátvitellel párhuzamosan zajlik le, és még az aktuális adatátvitel befejezése előtt eldől, hogy melyik következő master adhat. A Mastereket M1...Mn-el jelöljük, a BR: Bus Request (busz kérése, igénylése) a Master által, míg a BG: Bus Grant (igénylés elfogadása, engedélyezés) jelet a központi AU: Arbitration Unit (arbitrációs egység) bocsátja ki.
- **Az arbitrációnak 3 típusa** van:
  - a.) párhuzamos,
  - b.) soros (daisy chain), és
  - c.) lekérdezéses (polling).

a.) Párhuzamos



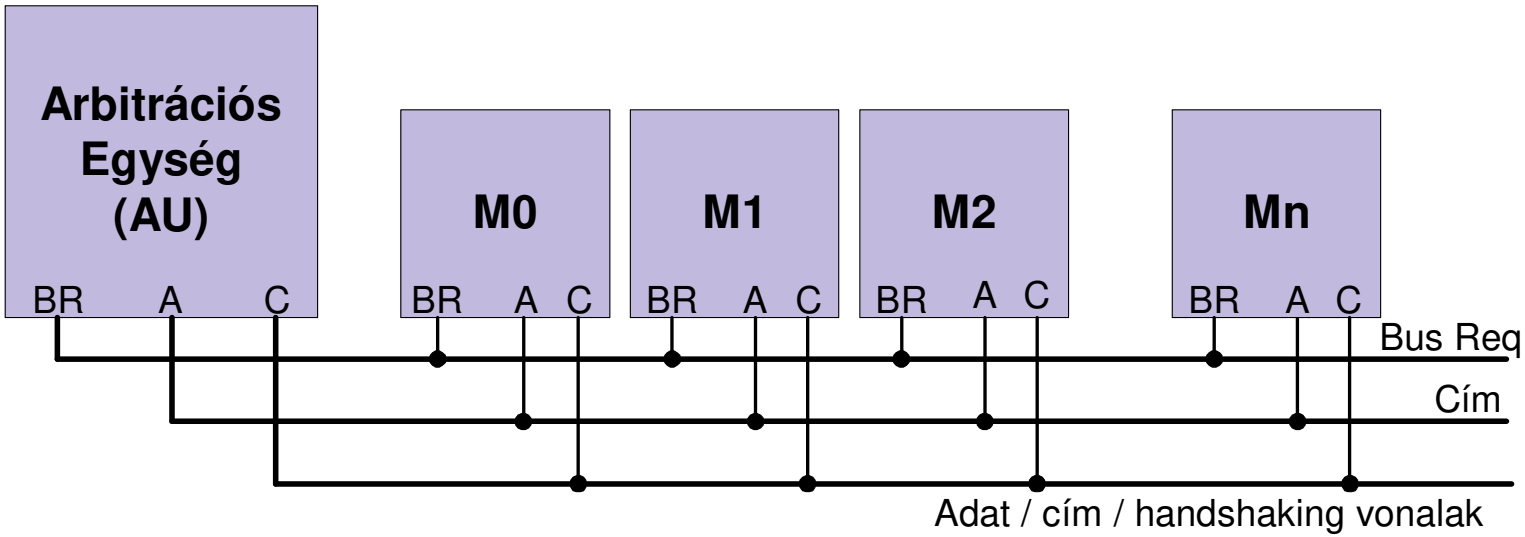
- Ez a leggyorsabb módszer, minden M_i-nek kitüntetett, egyenrangú kapcsolata van az AU-val (két vonalon: BG és BR-en). Az arbitráció lehet *(i) first asserted/first served (FIFO),* vagy (ii) *round robin*, vagy (iii) *prioritásos alapú.* Ezek a módszerek többfajta lehetőséget biztosítanak mind egyszerű, mind pedig komplex esetben.
- Az AU vezérli a közös buszon az adatátvitelt. Az adat-cím-handshake vonalat a kijelölt master kezeli, az tranzakció befejeztével pedig kiadja ismét a BR-jelet. Hátránya, hogy drágább, mint a többi módszer (a párhuzamos ágak miatt minden M_i-hez 2 vonal kell), és az M_i-k száma korlátozott.

b.) Soros (daisy chain)



- A BG vonal sorosan van kötve, míg a BR vonal mindegyik M_i-hez csatlakozik. Az AU nem ismeri, hogy pontosan melyik M kívánja elérni a buszt, ezáltal az átvitel leegyszerűsödik: csupán azt tudja, hogy bizonyos ciklusonként BG-jelet kell kibocsátania. A soros csatlakozás miatt a legelső Master (M_0) rendelkezik a legnagyobb prioritással (*fizikai prioritás*), így ha ő igényelt, minden esetben megkapja a buszt.
- Bármennyi eszközt is sorba köthetünk, nincs felső korlátja. Hátránya, hogy az arbitrációs idő (ha a legutolsó M_i igényel és az előtte lévők nem) egyenes arányban van a sorba kapcsolt M_i-k számával.

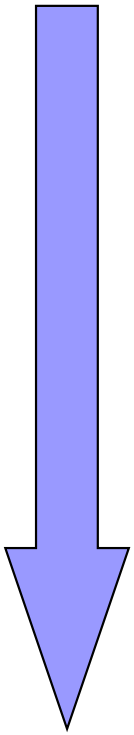
c.) Lekérdezéses (polling)



- Mindegyik Master egy **közös** BR buszon igényelhet (ID), és az AU dönti el, hogy melyik Masternek adja a buszt. Annak a **címét** az *address* vonalra rakja. Minden ciklusban megnézi az igényléseket, és a legmagasabb prioritással rendelkezőt fogadja el. Itt is többféle prioritásos módszer megvalósítható: pl. (FIFO, round robin stb.).
- Hátránya, hogy nagyobb az időszükséglete a párhuzamosnál, így ritkábban alkalmazzák (pl: I/O kérések arbitrációjánál), a processzor egy program futtatásával folyamatosan monitorozza az I/O eszközök busz kéréseinek állapotát (a megszakításos I/O műveleteknél rosszabb, lásd később).

Adatmozgatás:
I/O kommunikációs technikák

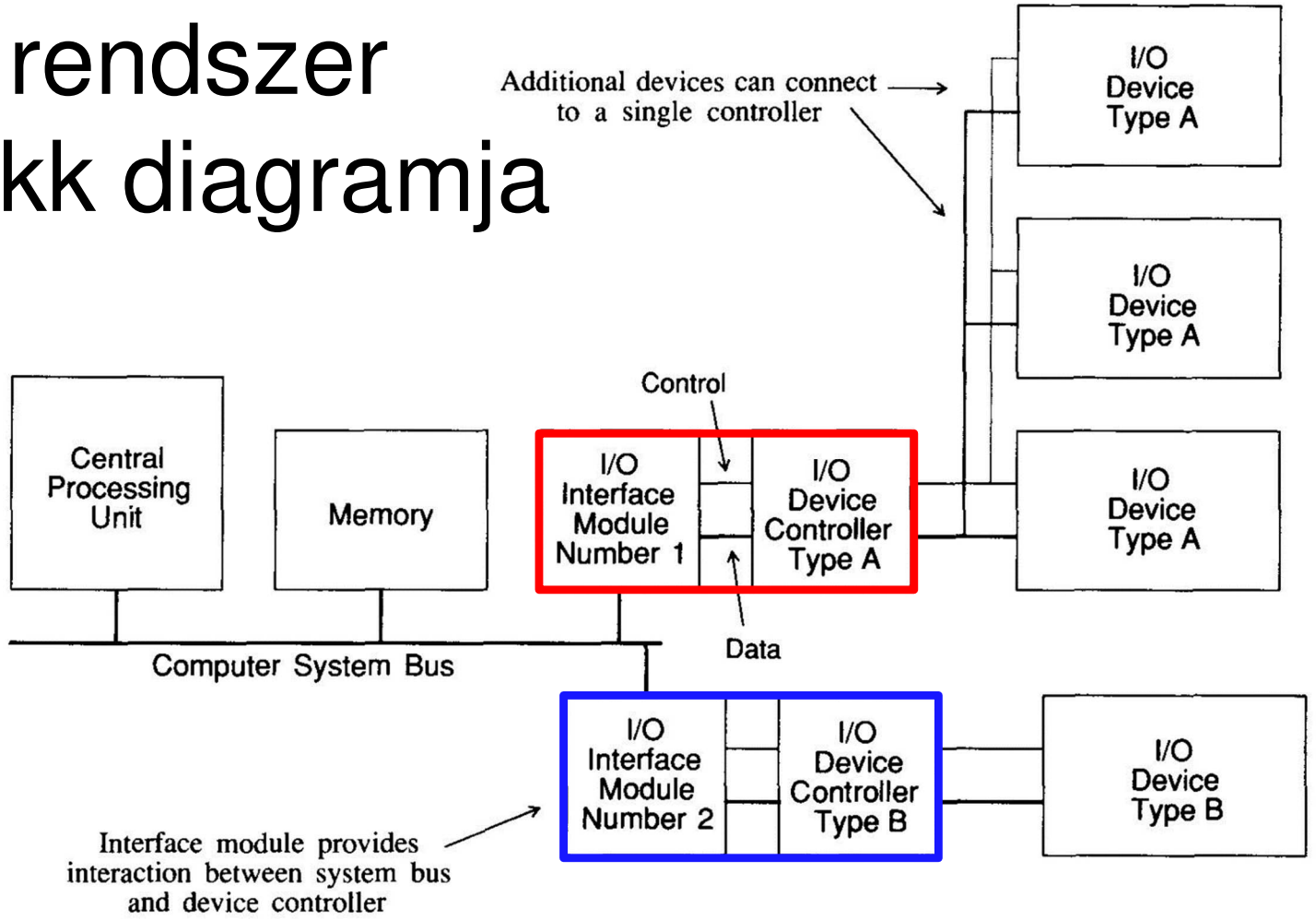
I/O adatátvitel típusai:



- Programozott I/O átvitel (polling)
- Megszakításos (interrupt) átvitel
- Direct Memory Access (DMA) - Közvetlen memória-hozzáféréssel rendelkező átvitel
- IOP - I/O processzoros átvitel

Feladat: CPU tehermentesítése

I/O Interfész rendszer általános blokk diagramja



Két fő részre osztható: 1. ***rendszerbusz*** (CPU, memória) 2. *I/O* ***eszközvezérlők*** a különböző típusú I/O eszközökkel tartják a kapcsolatot. A rendszerbuszt és az I/O eszközvezérlőket az I/O interfacek kapcsolják (illesztik) össze. Az eszközök gépi kódú utasításokkal (assembly) vezérelhetők. Egy új I/O eszközt (A v. B típusú) a megfelelő típusú eszközvezérlőkhöz kell kapcsolni.

1.) Programozott I/O átvitel (Polling)

- Legegyszerűbb technika
- De leginkább ez a módszer terheli a CPU-t (adatátvitel teljes ideje alatt) – lassú eszközök esetén
  - Teljes vezérlésért, adatmozgatásért felel
  - Pl. periféria állapotának ciklikus lekérdezés folyamatosan terheli
- Csak a CPU közbeiktatásával érheti el a periféria a memóriát
- Lehet Memory Mapped I/O: amikor a program és az I/O eszköz is ugyanazt a címtartományt használja (cím leképezés = „mappelés”)

2.) Megszakításos (interrupt) átvitel

- Megszakítással jelezhető a CPU-által az I/O eszköz számára az adatátviteli igény, illetve az adatátvitel befejeződése
- Megszakítás kérelem (**i**nterrupt **re**q**u**est) I/O eszköz által – **IRQ** szintek
  - Megszakítási vektorok (maszkolható megszakítások – SW interrupt)
- Bővebben: Operációs rendszerek tárgyból

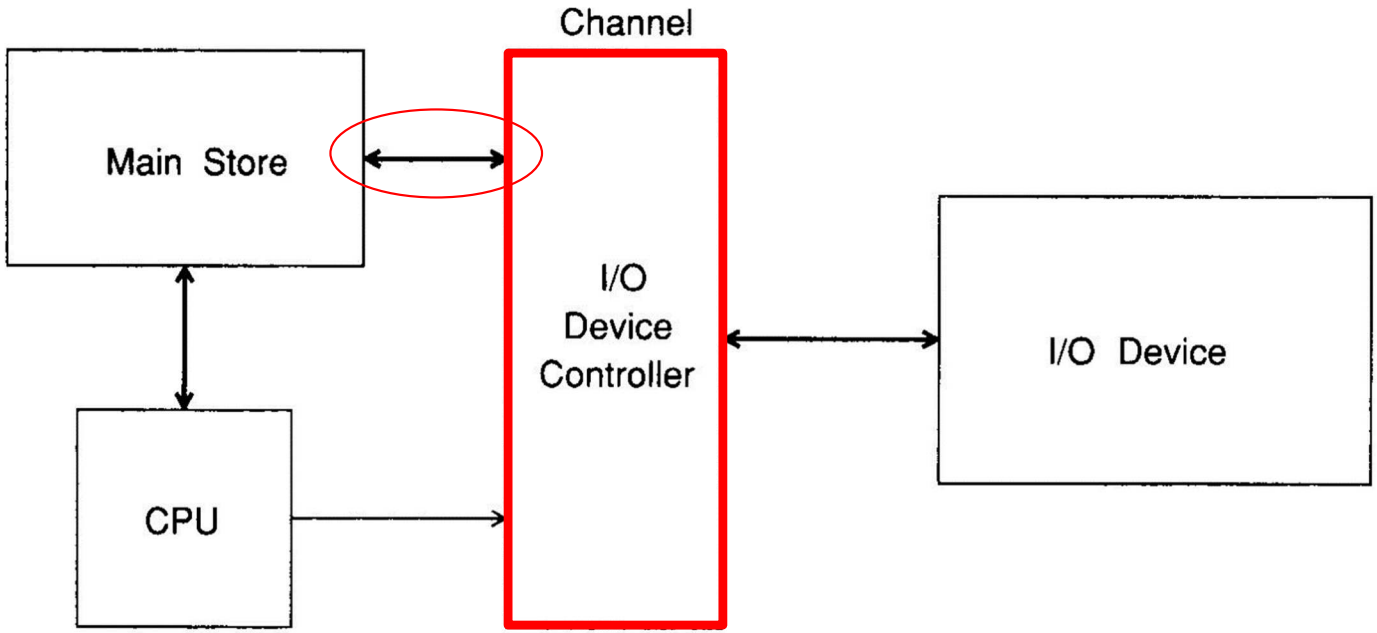
3.) Direct Memory Access (DMA)

- Közvetlen memória hozzáférés:
  az I/O eszköz és memória közötti adatátvitelt a processzortól függetlenül egy DMA-controller (eszközvezérlő) végzi.
- Cél a CPU tehermentesítése a tranzakció idejére
- CPU feladatai csupán:
  - az átvitel előkészítése (kezdőcím és adat hossza),
  - minimális vezérlés: eszköz állapot vizsgálata (busy)
  - és a befejezett művelet hibátlanságának ellenőrzése (megszakításos alapú is egyben)
- Gyors módszer

4.) I/O processzor (I/O csatornák)

- A cpu átadja az I/O műveletet és a végrehajtáshoz szükséges összes adatot ez intelligens eszközvezérlőnek = I/O (társ)processzornak, amely teljesen önállóan szabályozza a tranzakciót
- Főleg a mainframek-re jellemző módszer
- Rendkívül gyors
- I/O csatornák: eszköz sebessége szerinti osztályozás

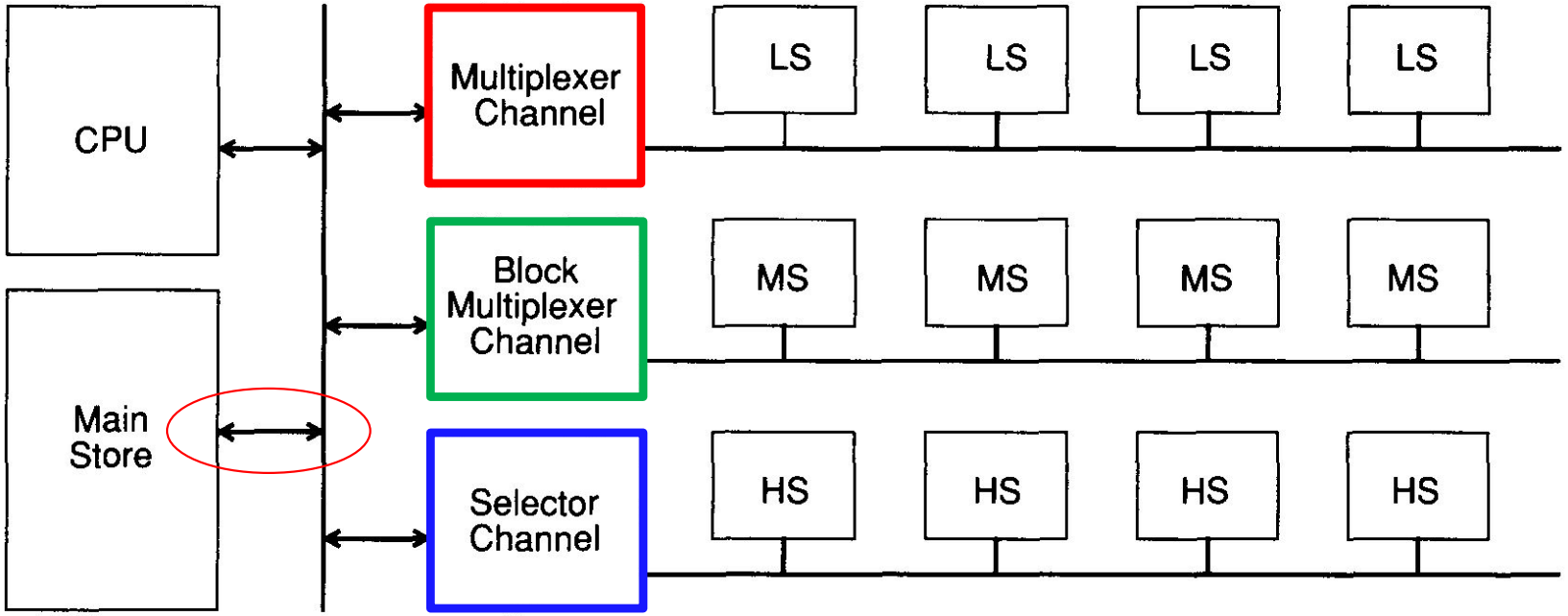
a.) I/O channels – I/O csatornák



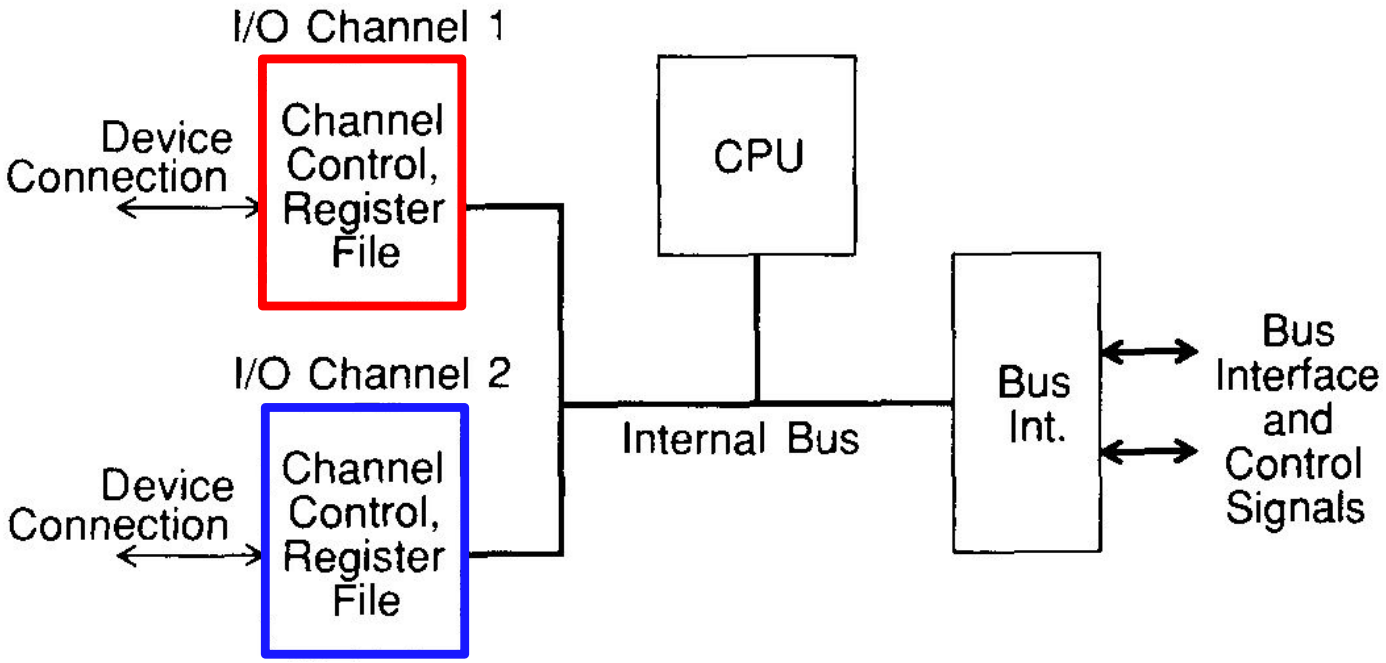
- A cpu csak az eszközvezérlőn keresztül (közvetetten) érheti el a perifériát.
- Channel = I/O Device Controller: általános célú processzáló elem.
- Channel feladata:
  - Konverzió
  - Adatmozgatás
  - Hibaellenőrzés és kezelés

Példa: I/O channel – I/O csatornák

- I/O csatornák típusai (sebességük szerinti kategóriákat képeznek):
  - Multiplexeres: LS (lassú)
  - Blokkos: MS (közepesen gyors)
  - Selector channel: HS (nagysebességű)



b.) I/O Processzorok (IOP)



- IOP: Intelligens eszközvezérlők/processzorok
- Saját, dedikált funkciókkal rendelkeznek (vezérlést, és interfészt biztosít más rendszerekkel – különböző sebességen)
- Példa: SCSI rendszer is egy IOP