BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika
VIMIAA02
9. hét

Fehér Béla, Benesóczky Zoltán
BME MIT

Processzor utasítás rendszerek

- **A processzorok utasításkészlete az alábbi utasítás csoportokra osztható:**
  - **Adatmozgató** (MOV, LD, ST...)
  - **Aritmetikai** (ADD, SUB, MUL, DIV….)
  - **Logikai** (AND, OR, XOR, NOT…)
  - **Léptetés** (SHL, SHR, ASH…)
  - **Forgatás** (ROL, ROR,…)
  - **Feltétel vizsgálat** (CMP, TST,…)
  - **Vezérlő** (JMP, CJMP, CALL/RET,...)
  - **Egyéb** (NOP, EI/DI, HALT,...)
- **Jellemző RISC utasításkészlet: 50 – 150 utasítás**

Processzor utasítás rendszerek

- **A RISC processzorok utasítás formátuma**
  - Általában **fix hosszúságú,** 16/32 bit
  - Felépítése **2 regiszter címes (2R) vagy 3R típusú** (utóbbi a 32 bitnél)
  - Az **azonos utasítás formátumhoz tartozó utasítások egyes bitmezőinek elhelyezkedése és jelentése azonos**. (Pl. *műveleti kód, op1reg, op2reg, konstans adat, konstans cím...*)
  - **Kevés utasítás formátum** (bitmezők jelentése) **típus** (2-5 maximum)
    - **Egy formátumon belül minden utasítás azonosan használható**
  - **A bitmezők kiosztása** (mely biteken van) **rögzített**
    - **Egyszerű a dekódolása, sok érték közvetlenül használható az processzor adatstruktúrájában**
  - **Az utasítások végrehajtási ideje azonos (1 órajel).**
    Párhuzamosítás: Egy utasítás végrehajtási fázisában a következő utasítást már beolvassa. (A Mini RISC nem..)

Processzor utasítás rendszerek

- A processzor utasításaira a program írása során a nevükkel (**mnemonik**) hívatkozunk **assembly nyelvű programozás**kor.
- A különféle processzorok ugyanazon műveletet elvégző utasításainak nevei sokszor kissé eltérőek. (Pl. adatmozgatás:MOV, LOAD, LD, STORE, ST,..)
  - A továbbiakban többnyire a MiniRISC CPU utasításait használjuk a példákban.
- **Az assembly programot a compiler lefordítja a processzor utasítkódjára**.
- A lefordított **utasítás kódokat betöltik a processzor utasítás memóriába**.

Processzor utasítás rendszerek

- **A RISC processzorok címzési módjai**
  - ***A címzési mód az utasításban hivatkozott információ elérési módját jelenti***. (Pl. ***Hol vannak az operandusok? (utasítás kódban, melyik regiszterben, milyen memória címen) Hol van az ugrási cím?***
  - **A címzési módok jelentősen befolyásolják az utasítás ill. programvégrehajtás hatékonyságát.**
  - Különböző címzési módok léteznek a ***programmemória*** és az ***adatmemória*** elérésére.
    - Eltérő igények:
      - Programmemória esetén a cél ***a következő utasítás címének megadása***, ez ugrásoknál, feltételes ugrásoknál, szubrutinhívásoknál (függvény/eljárás) lényeges.
      - Adatmemóriánál általánosabb igények vannak.

Processzor utasítás rendszerek

- **Programmemória (kódmemória) címzési módjai**
- **A programmemóriát a PC (programszámláló) címzi**
- **A következő utasítás címe általában PC = PC + 1**
- **Az ugrás vagy szubrutinhívás esetén a PC értékét módosítani kell**
  - **Abszolút címzés**: A ***PC minden bitje módosul***: A teljes programmemória bármely címe szerepelhet, mint új cím. Kedvező, egyszerűen használható, de sok bitet igényel
  - **Relatív (PC relatív)**: **címzés** A ***PC bitjeinek csak egy része módosul***, az ugrás, szubrutinhívás elérési tartománya korlátozott. A programszervezés bonyolultabb (assembler/fordító feladata), de **a programok „lokalitása”** miatt ez ritkán okoz komoly problémát. Azért nem, mert **a programokban az ugrások legyakrabban kis címtávolságba történnek** (pl. ciklus szervezés). A relatív címzési tartomány 2^8 – 2^{16}
    *Pl. Egy **16 bites PC-hez 8 bites 2-es komplemens relatív cím adódik hozzá**. Így az **ugrási tartomány PC-128-tól PC+127-ig** terjed.*

Processzor utasítás rendszerek

- **Programmemória címzési módjai** (közvetlen, regiszter indirekt)
  - **Közvetlen cím:** Az ***ugrási cím értéke az utasításkód része***, ez gyakran relatív címzésre korlátoz (korlátos utasítás szószélesség)
    - Az **ugrási cím** FORDÍTÁS időben meghatározott, futáskor **konstans**, nem lehet adatfüggő
    - **MiniRISC**: 16 bit utasítás, 8 bit program cím: lehet abszolút is
    - Pl: jmp **addr** jmp 0b**10101010**

      Kódja: 10110000**aaaaaaaa** 10110000**10101010**

      Az ugrási cím (addr) az utasítás kód ‚a'-val jelölt bitjeiben van.

Processzor utasítás rendszerek

- **Regiszter indirekt:** ***Az ugrási címet egy regiszter tartalmazza*** (ez általában abszolút címzésre is elegendő)
  - Az ***ugrási cím adatfüggő lehet*** (regiszterben), több utas elágazások megvalósítására. (Előtte a ***regiszterbe be kell tölteni a megfelelő utasítás címét***.)
  - **MiniRISC**: ***van regiszter indirekt címzés*** (az ugrási cím egy regiszterben van), teljes tartományra működik, mert kicsi (8 biten címezhető) a teljes kódmemória.
    Pl: jmp (**ry**) jmp (r**5**)
    Kódja: 11111011**yyyy** 11111011**0101**
    Az ***ugrási cím az ry regiszterben van***. (A regiszter sorszáma az ‚y'-al jelölt biteken található.)

Processzor utasítás rendszerek

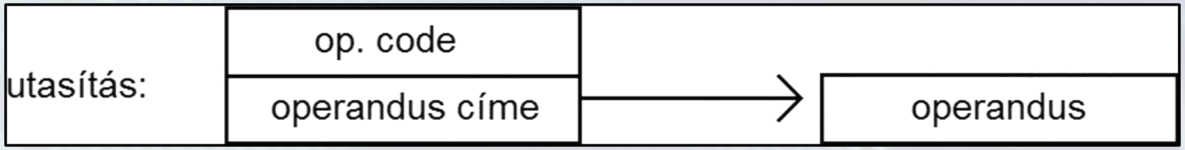
- **Adatmemória** címzése, illetve az **operandusok helyének megadása:**
  - **Közvetlen adat (adat az utasításban)**
  - **Közvetlen cím (az adat memóriacíme az utasításban)**
  - **Regiszter adat (adat a regiszterben)**
  - **Regiszter indirekt (adat címe a regiszterben)**
  - Regiszter + közvetlen ofszet című adat
  - Regiszter címzésű indexelt adat
  - Regiszter címzésű post-inkremens/pre-dekremens adat
- **Egy-egy utasításkészlet nem mindegyiket tartalmazza**
- **Léteznek egyéb speciális címzések is, ezeket nem tárgyaljuk**

Processzor utasítás rendszerek

- **Közvetlen (immediate) adat**:
  Az adat az utasításkód része
  - MiniRISC 16 bit utasítás, 8 bit adat → megoldható
    Pl: add rx, #**imm** add r3, #0b**10101010**
    kódja 0000xxxx**iiiiiiii** 0000001**10101010**
    Az adat (operandus) az utasítás kód i-vel jelölt bitjeiben van.
  - Általában pl. 32 bites processzor esetén nem egyszerű:
    32 bites utasításban 32 bites adat nem helyezhető el, mert akkor más információnak nem marad hely.
  - Gyakran használunk ***„kis értékű" konstansokat***, ezért a közvetlen adatra egy ***8-12-16-20 bites bitmező*** áll rendelkezésre

Processzor utasítás rendszerek

- **Közvetlen cím**: ***Az adat memóriacíme az utasítás része***



MiniRISC 16 bit utasítás, 8 bit adat → megoldható, mert az adatinterfész csak 256 címet tartalmaz

Pl: mov rx, **addr** mov r3, 0b**10101010**

kódja 1101xxxx**aaaaaaaa** 0000001**10101010**

Az operandus címe az utasítás kód ‚a'-val jelölt bitjeiben van.

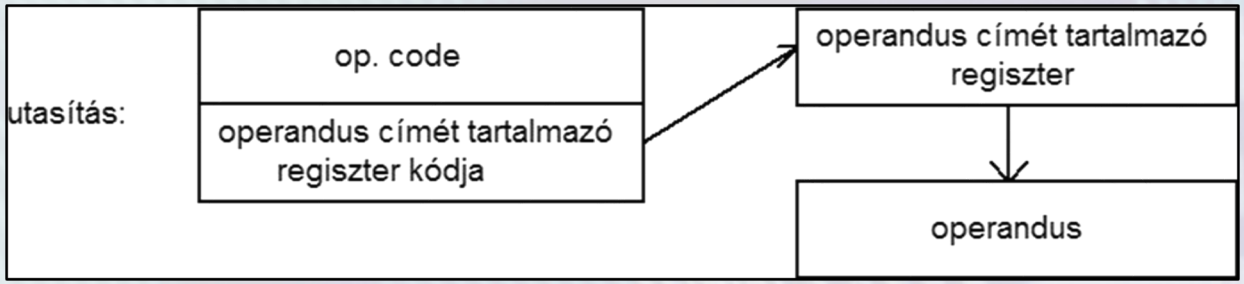
- Általában ugyanazok a korlátozások érvényesek, mint a közvetlen adatnál, bár beágyazott rendszerekben a memóriaméret korlátos (nem kell 32 címbit)

Processzor utasítás rendszerek

- **Regiszter adat**: *Az adatot regiszter tartalmazza* (*a regiszter címe az utasítás része*)
  - MiniRISC 16 bit utasítás:
    2 regiszter címes architektúra → max. 2 regiszterben lehet adat
  - Ez a leggyakoribb címzési mód
  - RISC processzorokban minden regiszter egyenértékű, egyformán minden utasításban használhatók. (Vannak processzorok, ahol ez nem igaz.)
    Pl: add **rx**, **ry** add r**3**, r**1**
    kódja 1111**xxxx**0000**yyyy** 1111**0011**0000**0001**
    Az adatok az rx és ry regiszterekben vannak. (*Az rx és ry regiszterek sorszáma az utasítás kódban található*.)

Processzor utasítás rendszerek

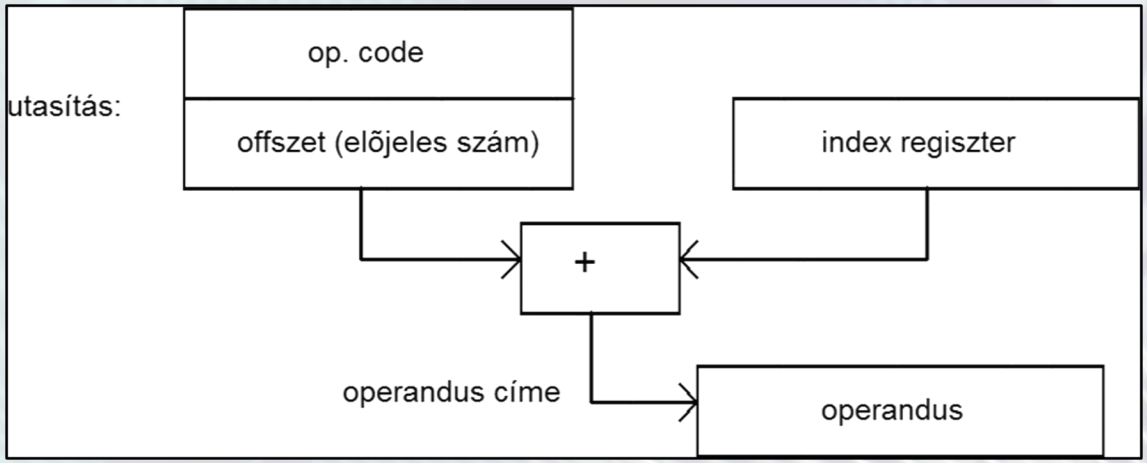
- **Regiszter (indirekt) címzésű adat:** ***Az adat memóriacímét regiszter tartalmazza***



- A MiniRISC utasításkészlet tartalmazza
  Pl: mov rx, **(ry)** mov r**3**, (r**1**)
  kódja 1111xxxx1100**yyyy** 1111**0011**1101**0001**
  Az operandus (adat) címe az ry regiszterben van.
  (Az rx és ry regiszterek sorszáma az utasítás kódban található.)
- Hatékony címzés ***adatvektorok/tömbök eléréséhez***

Processzor utasítás rendszerek

- **Regiszter + közvetlen ofszet adat:** ***Az adat címe egy regisztertartalom (bázis/index regiszter) és az utasításból származó közvetlen érték (ofszet) összege***



Pl: AVR mikrokontrollernél:

Mnemonik	kód formátuma	Rd	offset
LDD Rd, Y+q	10q0 qq0d dddd 1qqq	ddddd	qqqqqq

Az adat címe: Y regiszter tartalma + q

Processzor utasítás rendszerek

- **Regiszter címzésű indexelt adat**: ***Az adat címe két regiszter tartalmának összeadásából adódik***
  Az egyik a bázis cím a másik az index érték
  adat címe = báziscím + index_reg

- **Regiszter címzésű post-inkremens/pre-dekremens címzés**: Az adat elérése után/előtt a címregiszter tartalma automatikusan módosul. Hasznos pl. tömb címzésénél.

  Pl: AVR mikrokontrollernél:
  LD Rd, X+
  Az adatot betölti az X regiszterben levő címről, majd inkrementálja X tarlamát (a címet).

MiniRISC utasításkészlete

A MiniRISC processzor diasorozat 1 - 59 diája

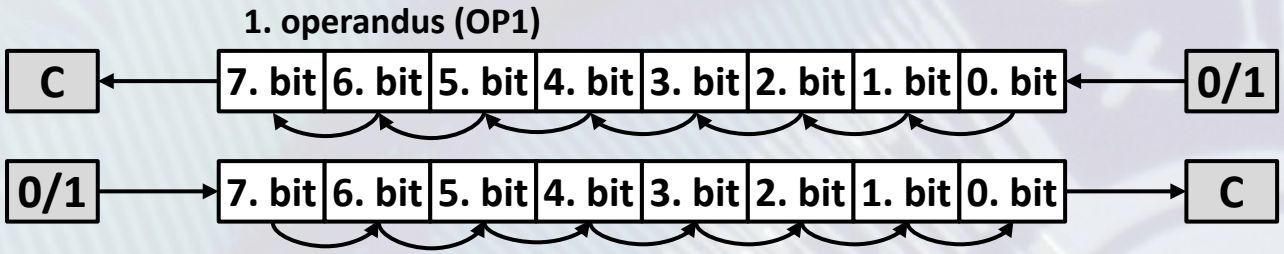
(nem ismertetjük részletesen, de kérjük mindenki nézze át!)

A továbbiakban néhány érdekesebb utasítás működését ismertetjük.

MiniRISC utasításkészlete

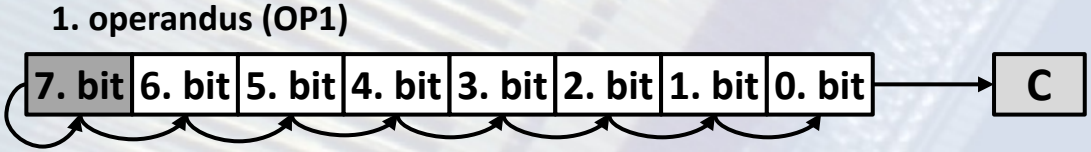
Logikai shiftelés

- A logikai shiftelés történhet balra vagy jobbra
- A beléptetett bit lehet 0 vagy 1, a kilépő bit a carry (C) flag-ba kerül



Aritmetikai shiftelés jobbra

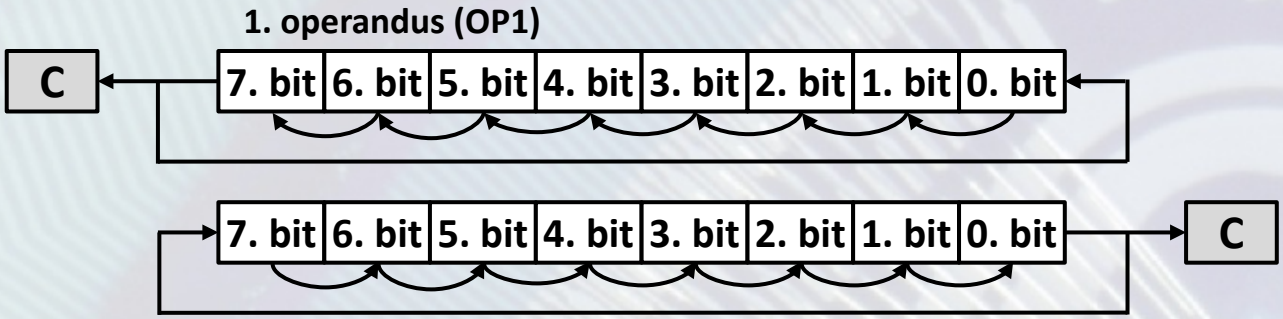
- Előjeles számok jobbra léptetése esetén az előjel bitet (MSb) helyben kell hagyni, hogy helyes eredményt kapjunk
- A kilépő bit a carry (C) flag-ba kerül
- Külön balra történő aritmetikai shiftelés nincs, mert ez megegyezik a balra történő logikai shifteléssel



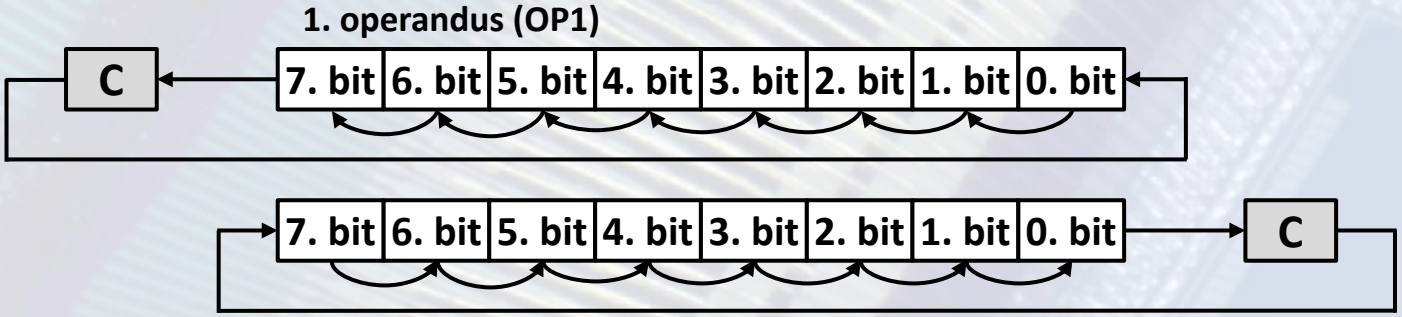
MiniRISC utasításkészlete

Normál forgatás

- A forgatás történhet balra vagy jobbra
- A kilépő bit beléptetésre kerül a másik oldalon
- A carry (C) flag a kilépő bit értékét veszi fel

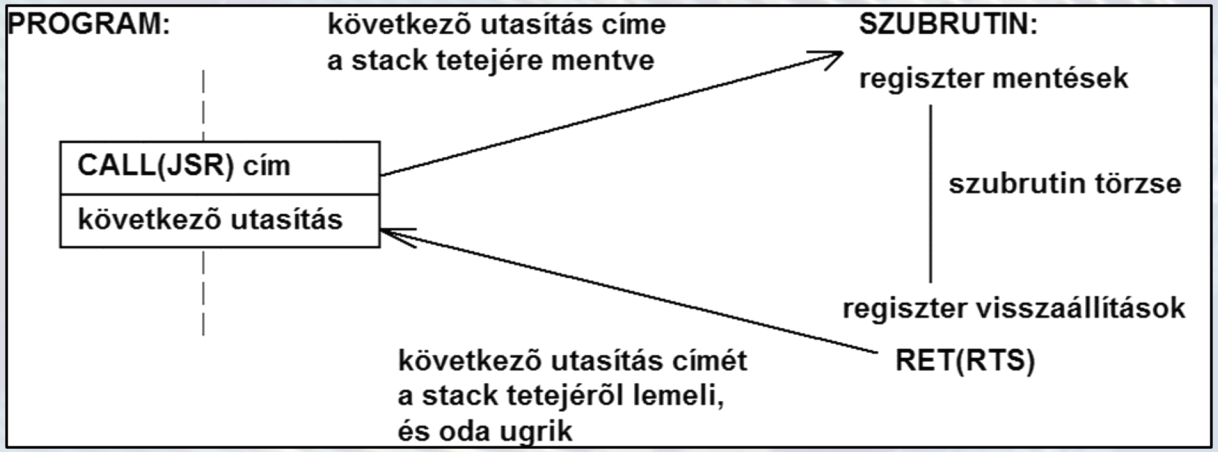


Forgatás a carry (C) flag-en keresztül



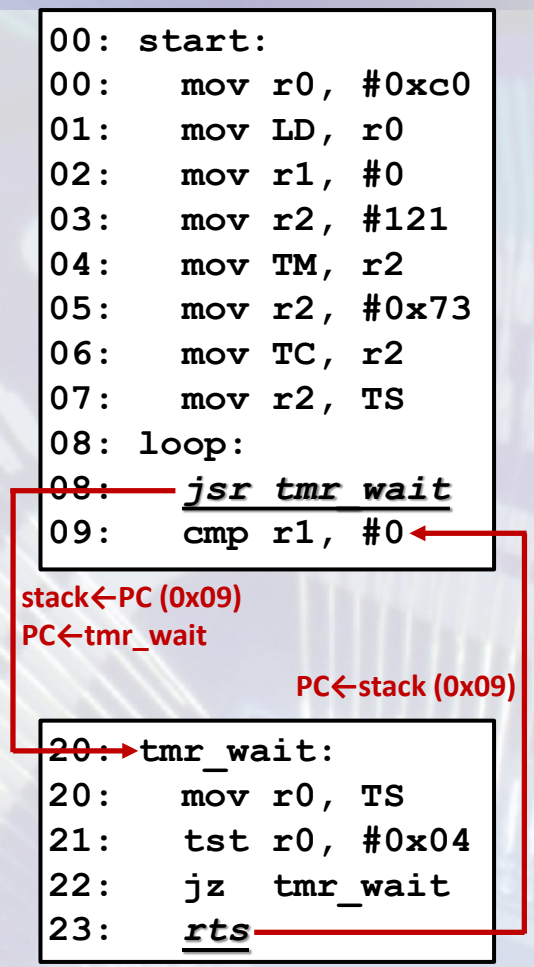
MiniRISC utasításkészlete

- **Szubrutin**
- A szubrutin hívó utasítás egy speciális ugró utasítás, amely az ugrás előtt a stack-re menti az utána következő utasítás címét. Elnevezése többnyire **CALL**, de a MiniRISC-nél **jsr.**
- Az ugrási címen egy valamilyen feladatot megvalósító program van elhelyezve. Ennek utolsó utasítása a szubrutinból visszatérő utasítás, neve többnyire **RET**, a MiniRISC-nél **rts.** Ez is speciális ugró utasítás, mely a stack tetejéről kiveszi a CALL/jsr által elmentett címet és oda ugrik, vagyis a szubrutin hívó utasítást követő utasításra.



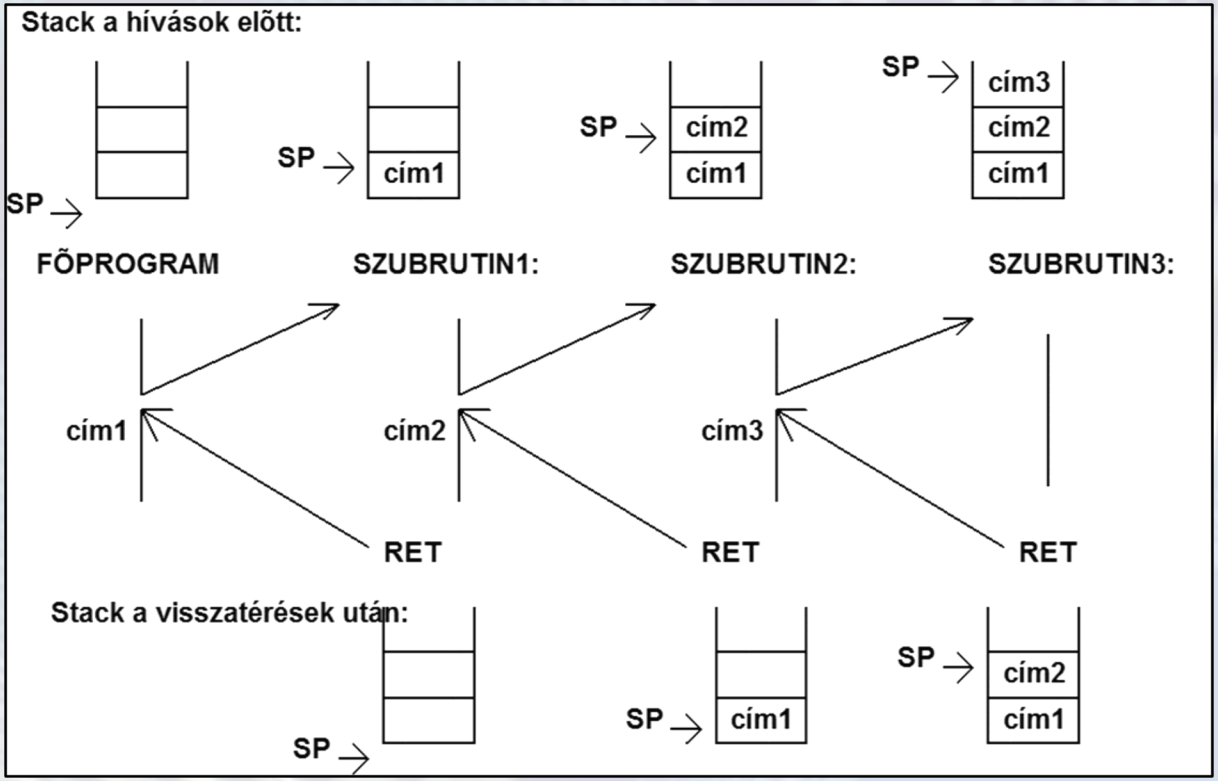
MiniRISC utasításkészlete

- Olyan részfeladatot megvalósító programrész célszerű szubrutinként megírni, amelyet többször is végre kell hajtani a program különböző részeiben.
- Így a programrészt csak egyszer kell leírni. A kódmemóriában is csak egyszer szerepel. Viszont többször felhasználható. Ha használni akarjuk, csak le kell írni a megfelelő címet tartalmazó hívó utasítást.
- Sokszor paraméterei is vannak (bemenő adatok és/vagy kimenő adatok). Ezeket átadhatjuk regiszterekben, de memória területen is.
- Bizonyos esetekben célszerű lehet a szubrutin által használt regiszterek egy részét a szubrutin elején elmenteni, a végén visszamenteni.



MiniRISC utasításkészlete

- **Egymásba ágyazott szubrutin hívások**
- A szubrutin hívásokat a stack terület által korlátozott mértékben egymásba lehet ágyazni, vagyis egy-egy szubrutinból újabbakat lehet hívni. Ezt a mechanizmust a stack teszi lehetővé.



Processzor belső kommunikációja

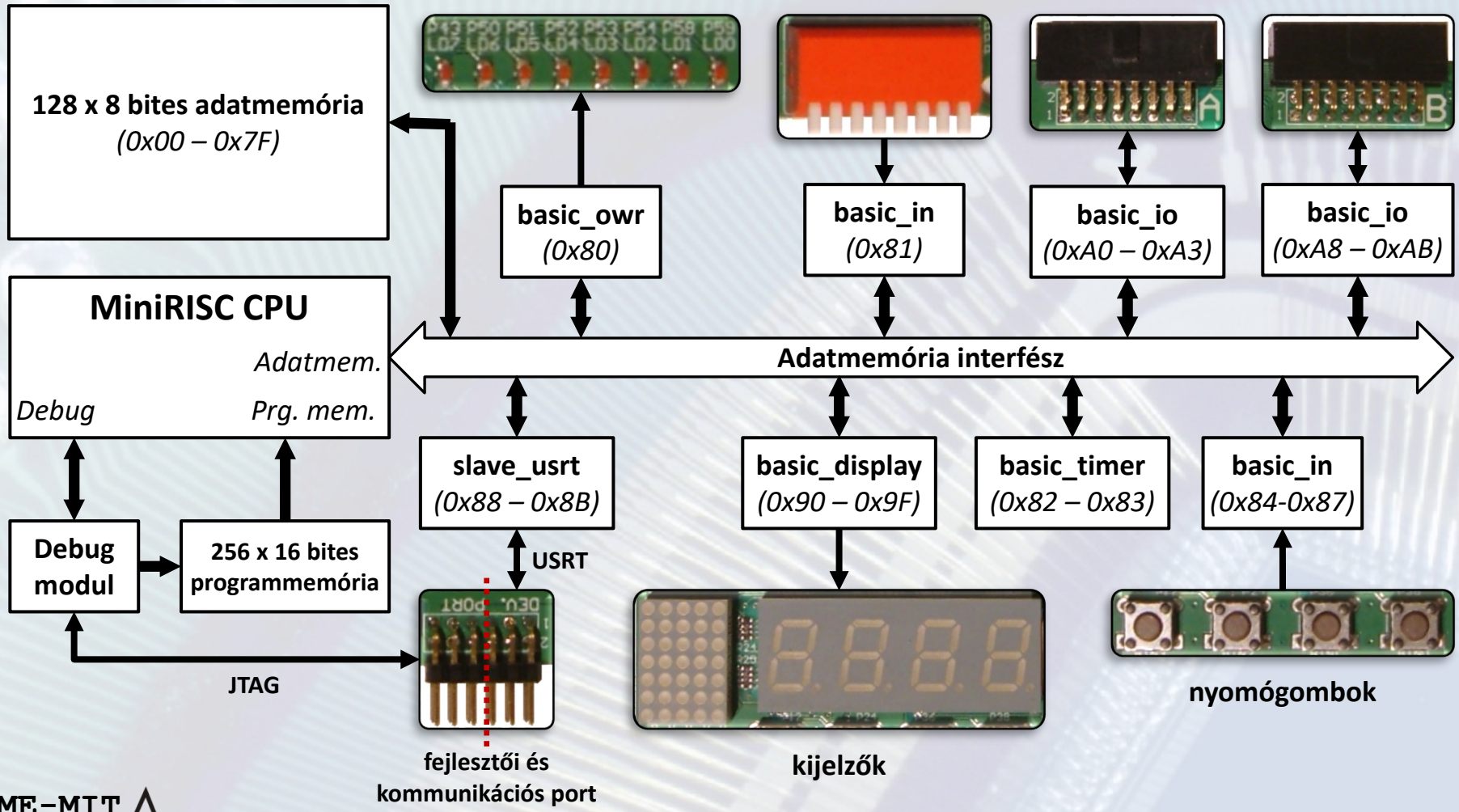
- **A mikroprocesszoros rendszerek összetett digitális rendszerek, több modulból állnak**
- **Eddig csak a központi egység felépítését, az un. CPU struktúráját és működését vizsgáltuk:**
  - Ez két fő részegységből áll: ***vezérlő*** és ***adatsruktúra*** (ALU + regiszterek) Sokszor ezt az adatstruktúrát is ALU-nak nevezik.
  - Ezek mindegyike bonyolult részrendszer, de önmagában szinte működésképtelen.
  - A vezérlő a működéséhez szükséges ***utasításokat*** a **programmemória interfészen** keresztül éri el.
  - Az ALU értelmes működésének feltétele az ***adatmemória és a perifériák*** elérése. Ezt az **adatmemória interfész** biztosítja

Processzor belső kommunikációja

- Az ***adat memóriában*** tárolja a CPU a változókat. Többnyire ennek egy részében van a szubrutinokhoz (és interrupthoz) szükséges STACK memória terület is. Azonban kisebb CPU-k esetén, azt önálló hardver STACK valósítja meg. (Pl. MiniRISC)
- A ***perifériák*** teszik lehetővé, hogy a CPU a környezettel kommunikáljon, onnan adatokat kapjon és oda adatokat küldjön. A perifériákat sokszor a processzor adat memória területére illesztik. (Mi csak ezt tárgyaljuk) Így a periféria regisztereket ugyanazokkal az utasításokkal lehet elérni, mint a memóriát.

Processzor belső kommunikációja

(MiniRSIC egyszerűsített blokkvázlata)



Processzor belső kommunikációja

- A mikroprocesszoros (uC) rendszerek adatátviteli interfészeit is ***busz***ok**nak** nevezzük. (Általánosan a ***busz*** valamilyen funkciót ellátó jelek halmaza.)
- A buszok teszik lehetővé az egységek közötti kommunikációt. A buszt vezérlő egységet ***busz masternek*** nevezik, a többi egységet ***slave***-nek. Elsősorban a CPU (processzor) a ***master*** (de speciális periféria is lehet) A továbbiakban masterként csak a CPU-ra hivatkozunk.
  - A ***címbusz*** a rákapcsolódó egységek regisztereinek megcímzésére (kijelölésére) szolgál.
  - Az ***adatbusz***on küld vagy fogad adatot a CPU a megcímzett egységtől.
  - A ***vezérlő busz***on a CPU kijelöli az adatmozgatás irányát (olvasás vagy írás) és vezérli ezek időbeli lezajlását. (A vezérlő busznak egyéb, itt nem részletezett funkciói is vannak.)

Processzor belső kommunikációja

- A mikroprocesszor rendelkezik belső busszal és rendelkezhet egyedi ***külső busszal*** (lokális busz)
- A ***belső busz*** a belső (chip-en belüli) memóriák és perifériák meg-címzésére szolgál. Itt áramköri okokból irányonként van egy-egy adatbusz (DATA_IN, DATA_OUT). (A MiniRISC-nek csak belső busza van.)
- A CPU-t és belső perifériákat, memóriát tartalmazó chipeket ***mikrokontrollernek*** nevezik
- ***A külső busz teszi lehetővé, hogy egy processzorhoz külső memóriát, perifériát lehessen illeszteni.***
- A ***külső és belső buszok*** némelyike processzor független, ún. rendszer busz, általános célú alkalmazásokra.
- ***Külső buszok esetén az adatbusz kétirányú.*** Ugyanazokon a jelvezetékeken adatot tud fogadni vagy adatot tud adni. Egyszerre csak az egyiket. Ezzel chip-en lábakat lehet spórolni, a belső buszhoz képest..

A mikroprocesszoros busz

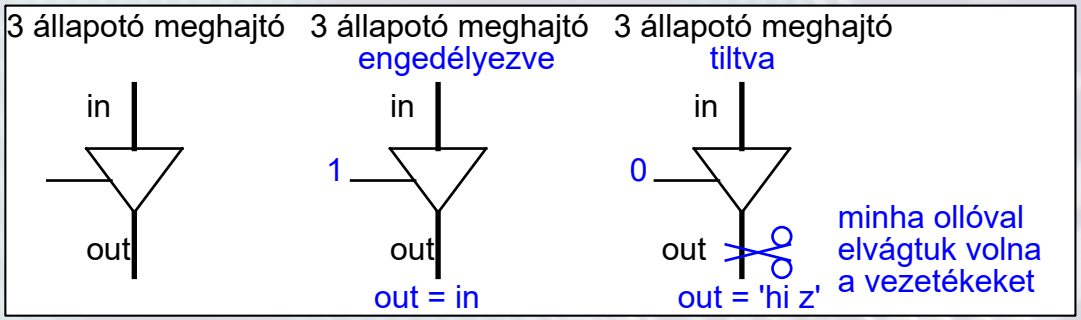
- **A busz részei részletesen:**
  - **Címbusz ADDR[n:0]**
  - **Adatbusz DATA[m:0],**
    belső busznál külön D_IN[m:0], D_OUT[m:0]
  - **Vezérlő busz (sok egyedi jel összessége):**
    - **Rendszerjelek: CLK, RST,**
    - Arbitrációs jelek: BUSREQ, BUSACK,
    - **Irányvezérlő jelek: READ, WRITE,**
    - Átvitelvezérlő jelek: FRAME, TS, TACK, AS, DS (nem tanuljuk)
    - **Megszakítás vezérlő jelek: IRQi,** IACK (utóbbi a MiniRISC-nél nincs).

A mikroprocesszoros busz

- **A busz használatának szabályai:**
  - A buszra kapcsolódó egységek között megkülön-böztetünk **MASTER** és **SLAVE** egységeket.
  - **MASTER:** ***Vezérelheti a buszt***. Kiadja a címet, meghatározza az adatáramlás irányát (írás/olvasás) íráskor kiadja az adatot, kezdeményezi az átvitelt és vezérli a működést. A MiniRISC rendszerben a processzor (és a DMA vezérlő) lehet MASTER. (DMA-val a tárgyban nem foglalkozunk.)
  - **SLAVE:** ***Figyeli a buszt***, felismeri/dekódolja a címet és a parancsokat (pl. írás/olvasás), azonosítás esetén (ha a címe egyezik a master által kiadottal) végrehatja a kérést a megfelelő adat fogadásával (ha írja a MASTER) vagy kiadásával (ha olvassa a MASTER).

A mikroprocesszoros busz

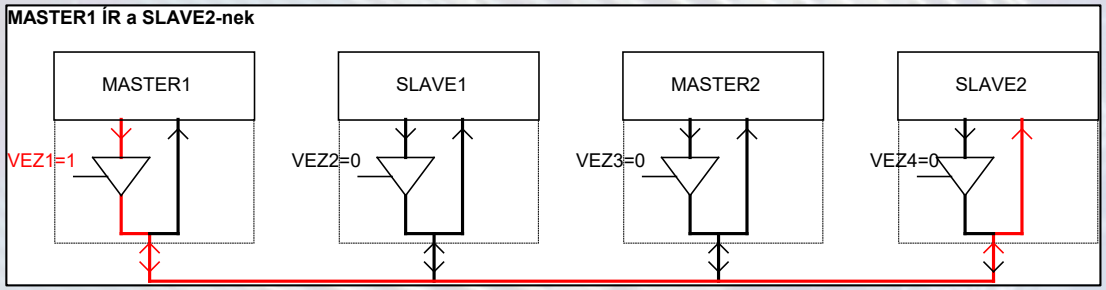
- **Buszhasználat <u>külső</u> buszon:**
- A busz meghajtására ***3 állapotú, HiZ, nagy impedanciás meghajtókat*** használnak (az oldalsó vezeték az engedélyezés):



- Ha ***a meghajtó engedélyezett***, akkor a ***kimenetén a bemeneti logikai szinteknek megfelelő feszültség*** jelenik meg.
- Ha ***a meghajtó tiltott, úgy vislkedik, minta elvágták volna a kimeneti vezetékeit***. Ezt nevezik ***magas impedanciás*** (Hi impedanciás) vagy „***HiZ" állapot***nak.

A mikroprocesszoros busz

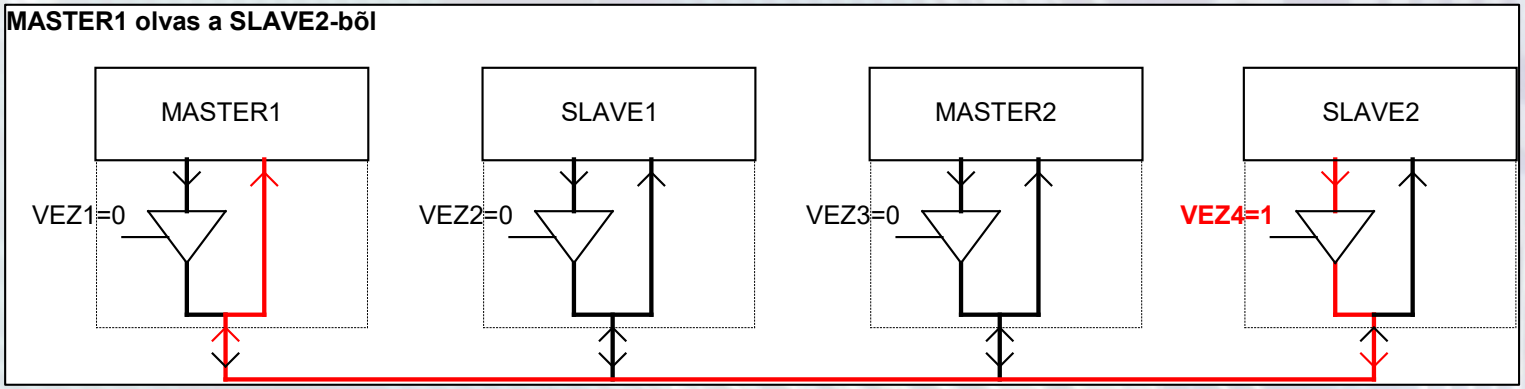
- **Buszhasználat külső buszon:**
- Közös vonalak, 3 állapotú, HiZ, nagy impedanciás meghajtókkal
  - Egy időben csak egyetlen adatforrás lehet aktív
  - A vezérlő/engedélyező jelek 1-az-N-ből kódolásúak. (Különben meghajtók szembe kapcsolódnának, ami meghibásodást okozna!)
  - Pl. Adatút, amikor a ***MASTER1 ír a SLAVE2-nek*** (SLAVE2 a címbuszon megjelenő címből tudja, hogy róla van szó, a vezérlő buszból, hogy írás lesz. Itt csak az adatbuszt tüntettük fel).



  - ***Csak a MASTER1 hajtja meg az adatbuszt, a többi egység meghajtója HIZ állapotban van.***

A mikroprocesszoros busz

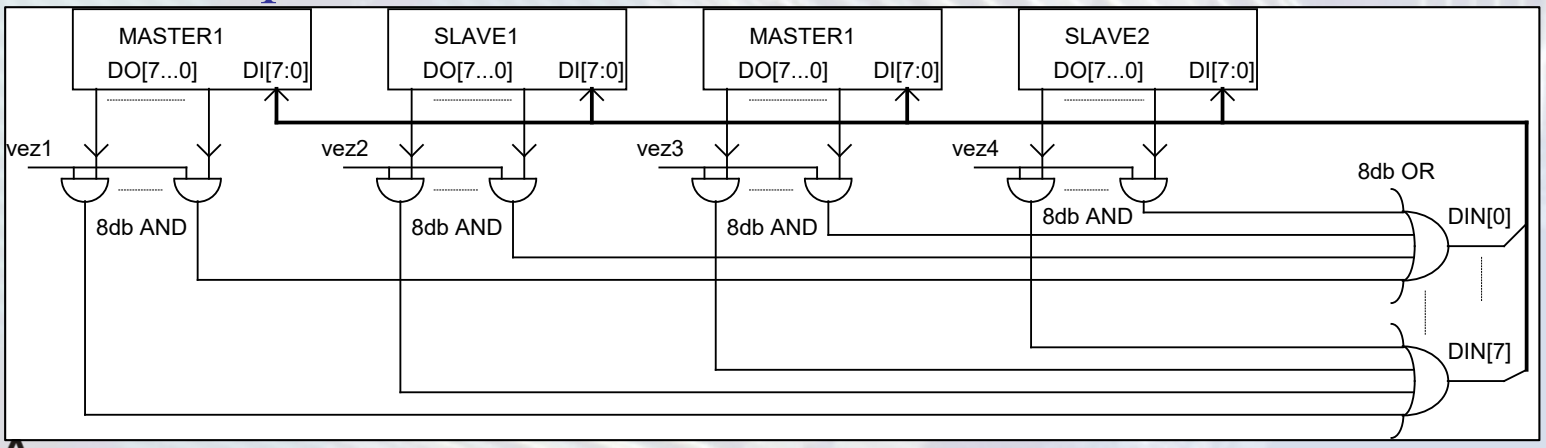
- Pl. Adatút, amikor a ***MASTER1 olvas a SLAVE2-ből*** (SLAVE2 a címbuszon megjelenő címből tudja, hogy róla van szó, a vezérlő buszból, hogy olvasás lesz, tehát neki kell meghajtani az adatbuszt. (Itt is csak az adatbuszt tüntettük fel).



- ***Csak a SLAVE2 hajtja meg az adatbuszt, a többi egység meghajtója HIZ állapotban van***.

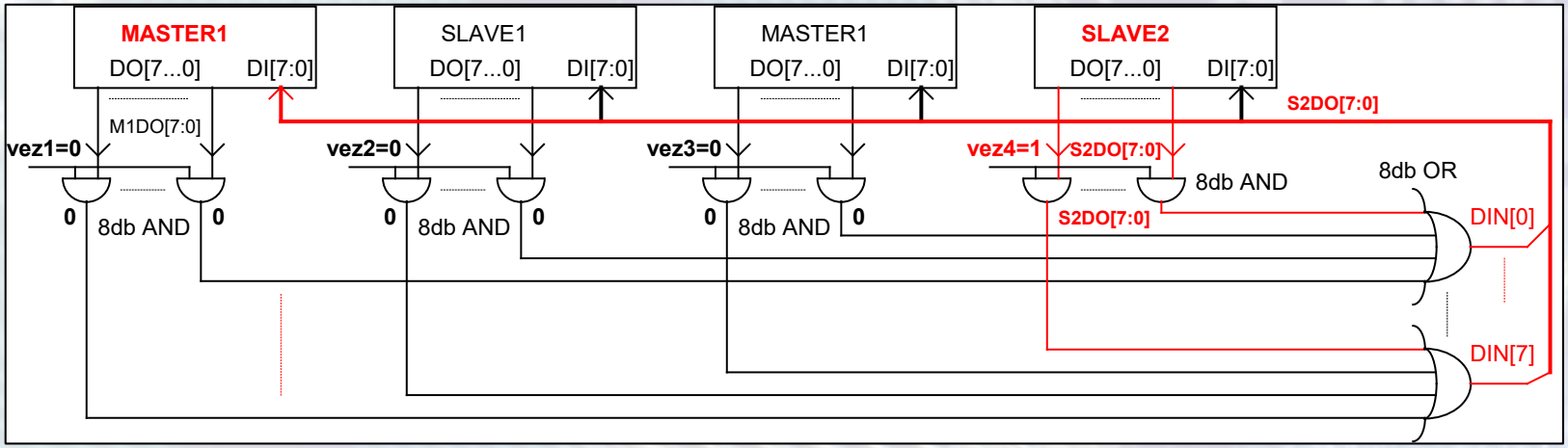
A mikroprocesszoros busz

- **Buszhasználat belső buszon:**
  - Az adatvonalak meghajtása AND-OR hálózaton. Ez tulajdonképpen egy elosztott, csatornánkénti kiválasztó (select) jellel rendelkező multiplexer). Azért elosztott, mert az AND kapuk az egyégekhez tartoznak, az OR kapuk a rendszerhez (a buszhoz).
  - A megoldás kizárja a több forrás kimeneteinek áramköri szembekapcsolását.



A mikroprocesszoros busz

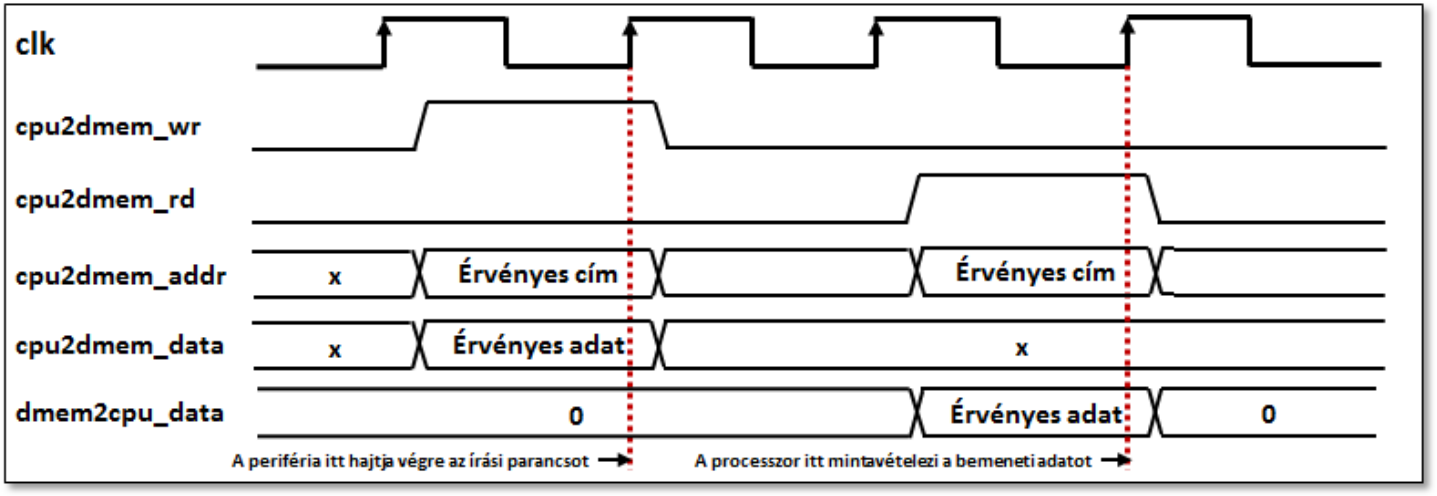
- Pl. Adatút, amikor a ***MASTER1 olvas a SLAVE2-ből*** (SLAVE2 a címbuszon megjelenő címből tudja, hogy róla van szó, a vezérlő buszból, hogy olvasás lesz. Itt csak az adatbuszt tüntettük fel).



- Csak a ***SLAVE2 hajtja meg a kimeneti adatával az adatbusz bitenkénti OR kapuinak bemeneteit***, ***a többi egység oda 0-át*** kapcsol. Ezért az OR kapuk kimenetein az SLAVE2 adata jelenik meg.

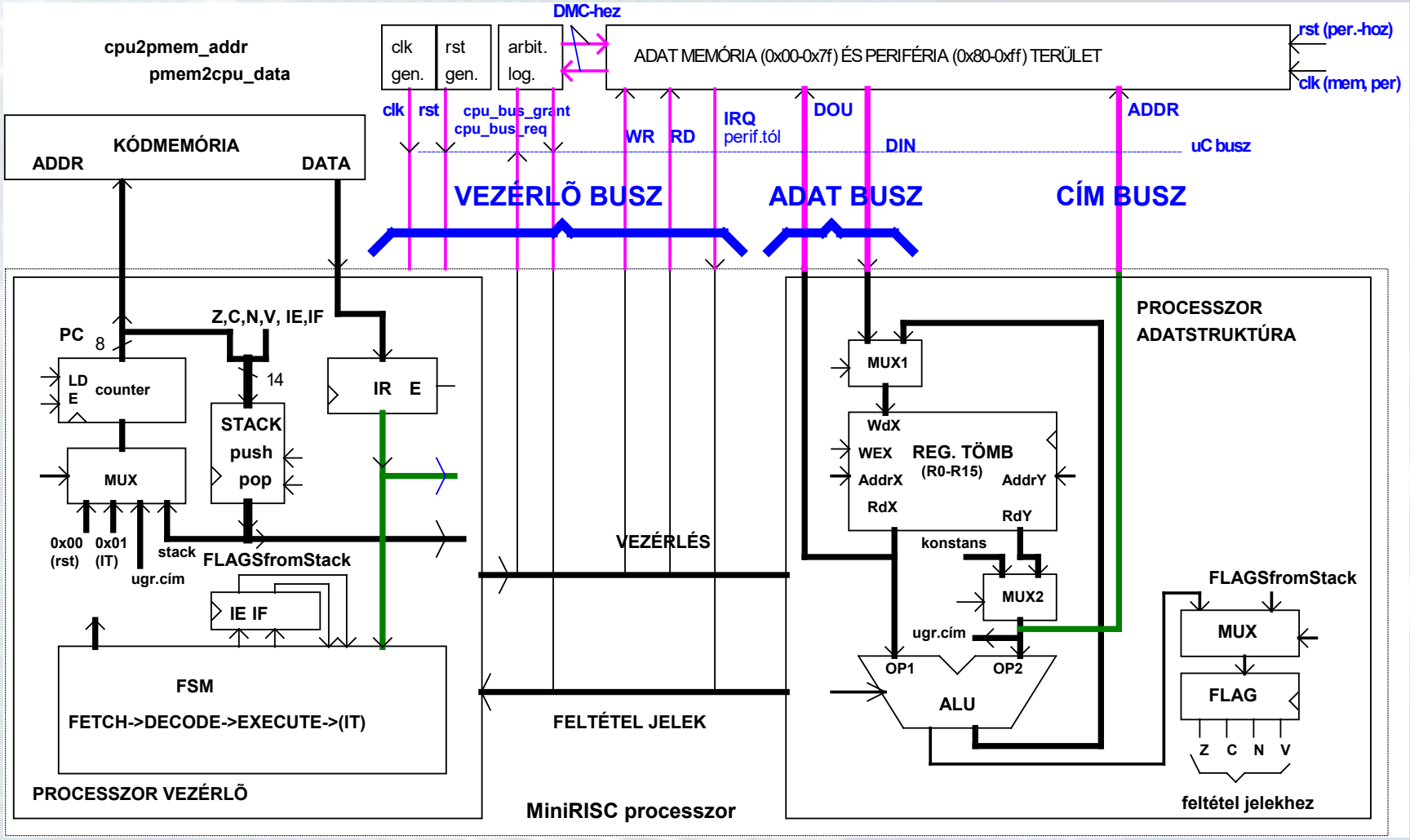
A busz adatátviteli ütemezése

- A buszon az adatátvitel ütemezése lehet **szinkron** és **aszinkron** (csak a szinkron ütemezéssel foglalkozunk)
- **Az adatátvitel ütemezése szinkron módon**
  - Az adatátvitelt a busz órajel ciklusai ütemezik
    - Sokfajta busz ütemezési protokoll létezik, gyakori a címzési és az adat fázis szétválasztása
      - Első ütem: a cím és a vezérlési paraméterek (pl: RD, vagy WR) beállítása
      - Második ütem: adatátvitel
    - Jellemzően egy átvitel több órajel ciklus ideig tart (START-ÁTVITEL-LEZÁRÁS)
    - A MiniRISC busz egyetlen órajel alatt hajtja végre az átvitelt:



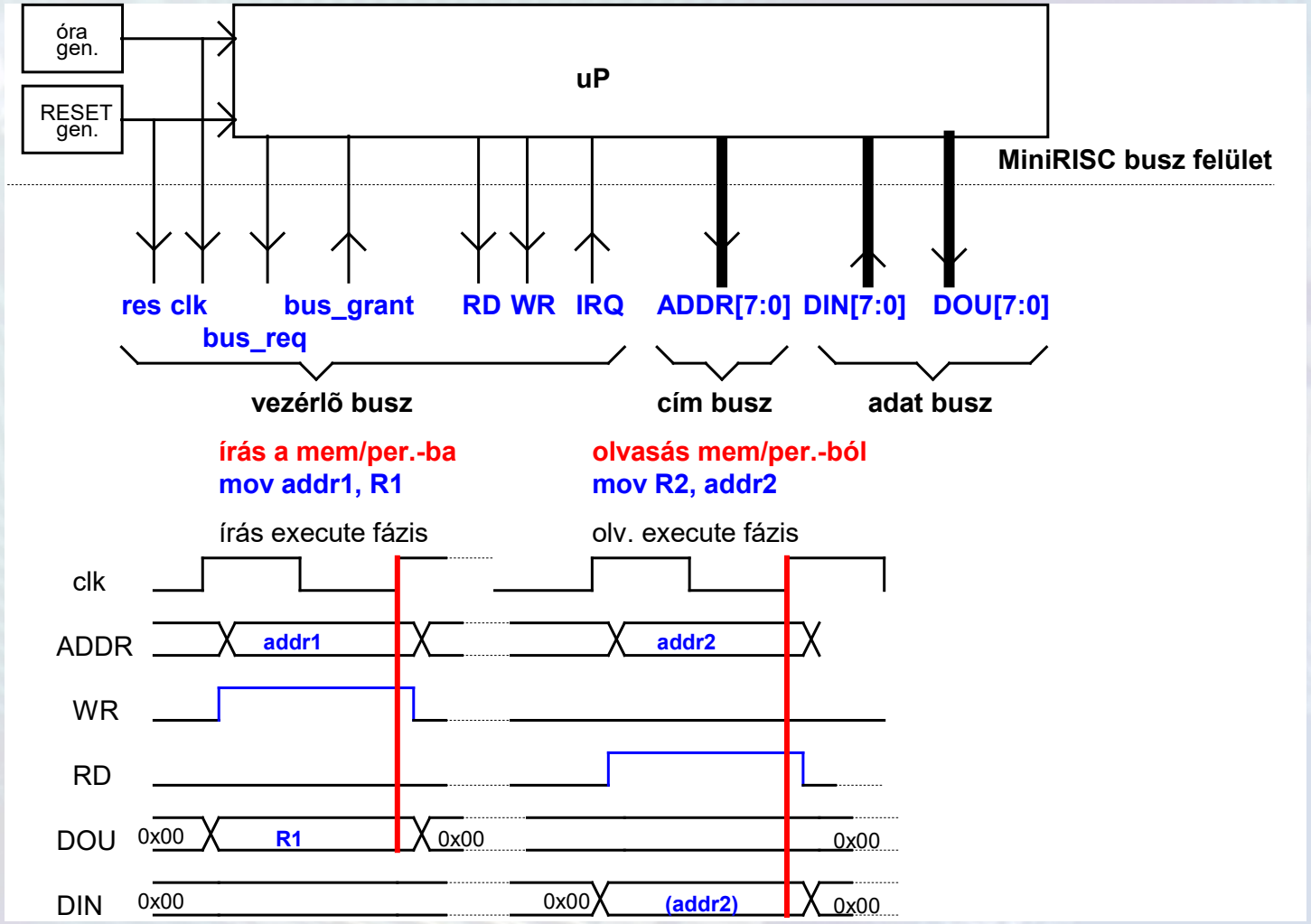
A mikroprocesszoros busz

MiniRISC busz



A mikroprocesszoros busz

MiniRISC írás és olvasás



A mikroprocesszoros busz

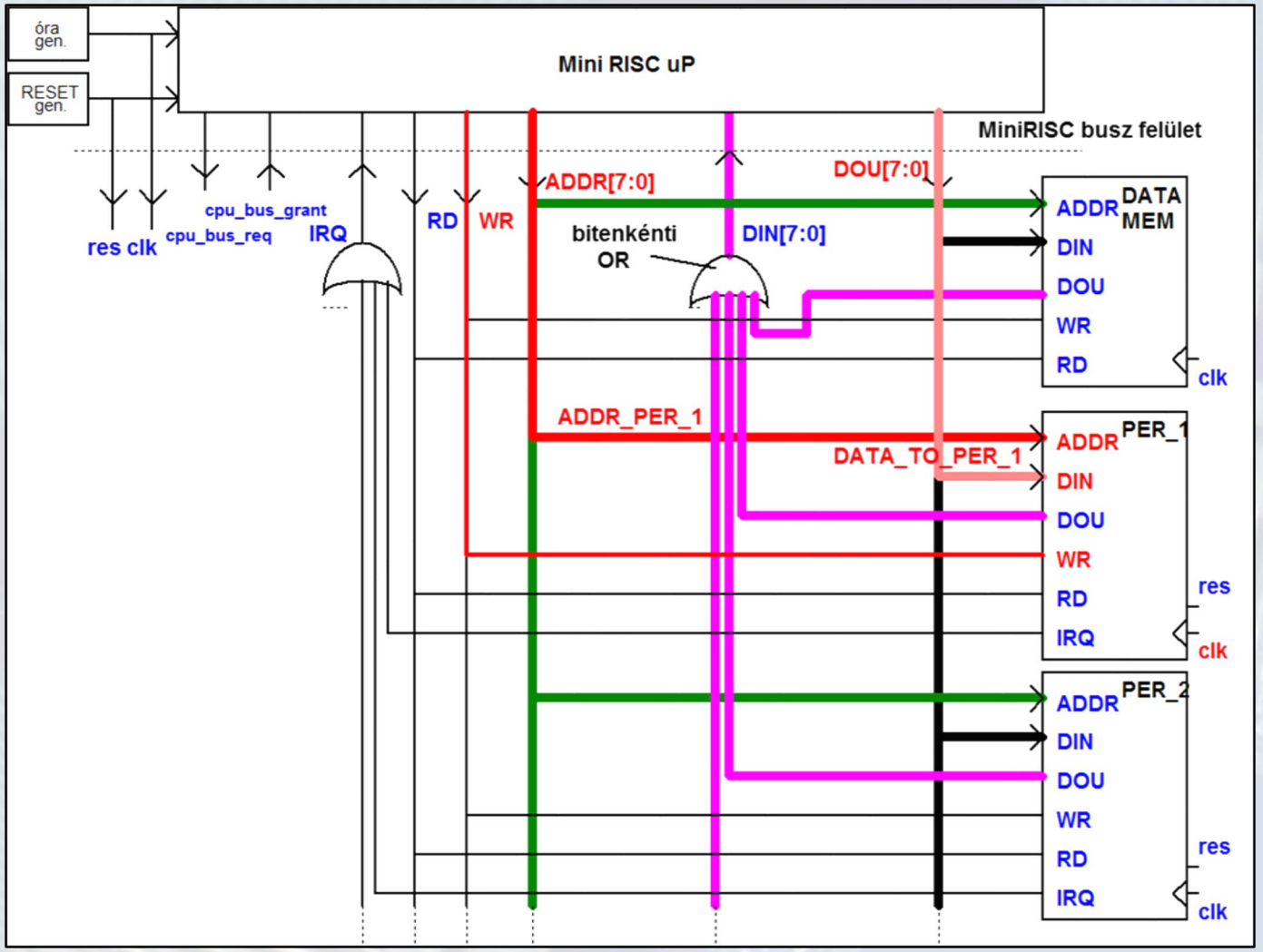
MiniRISC: Írás a PER_1 perifériába

Aktív jelek:

ADDR[7:0]

DOU[7:0]

WR



A mikroprocesszoros busz

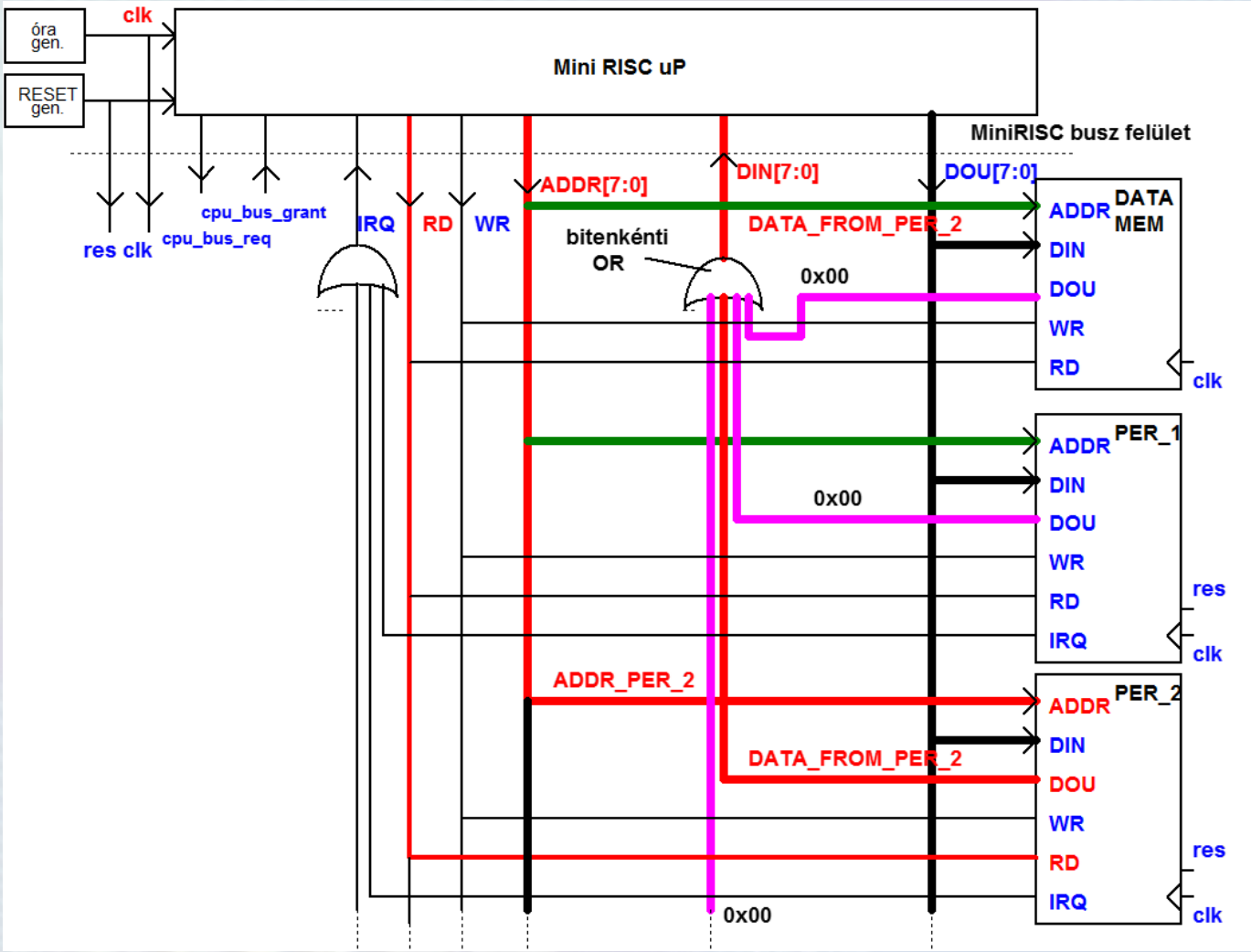
MiniRISC: Olvasás a PER_2 perifériából

Aktív jelek:

ADDR[7:0]

DIN[7:0]

RD



Digitális technika
10. EA vége

(Az alábbi diák nem részei a vizsga anyagának)

A busz adatátvitel aszinkron ütemezése

- Nincs ütemező órajel
- Az adatátvitelt a „kézfogásos” (hand-shake) szinkronizáció vezérli (REQ → ACK jelek)
- 4 állapot REQ_ACK → 00 – 10 – 11 – 01 – 00
- Ha az előző átvitelnek vége (ACK =0), akkor
- Átviteli paraméterek beállítása
- REQ → 1, start
- Vár ACK-ra, minden jel stabilan tartva
- Ha ACK megjön, REQ=0
- Vár ACK elengedésére

Arbitráció

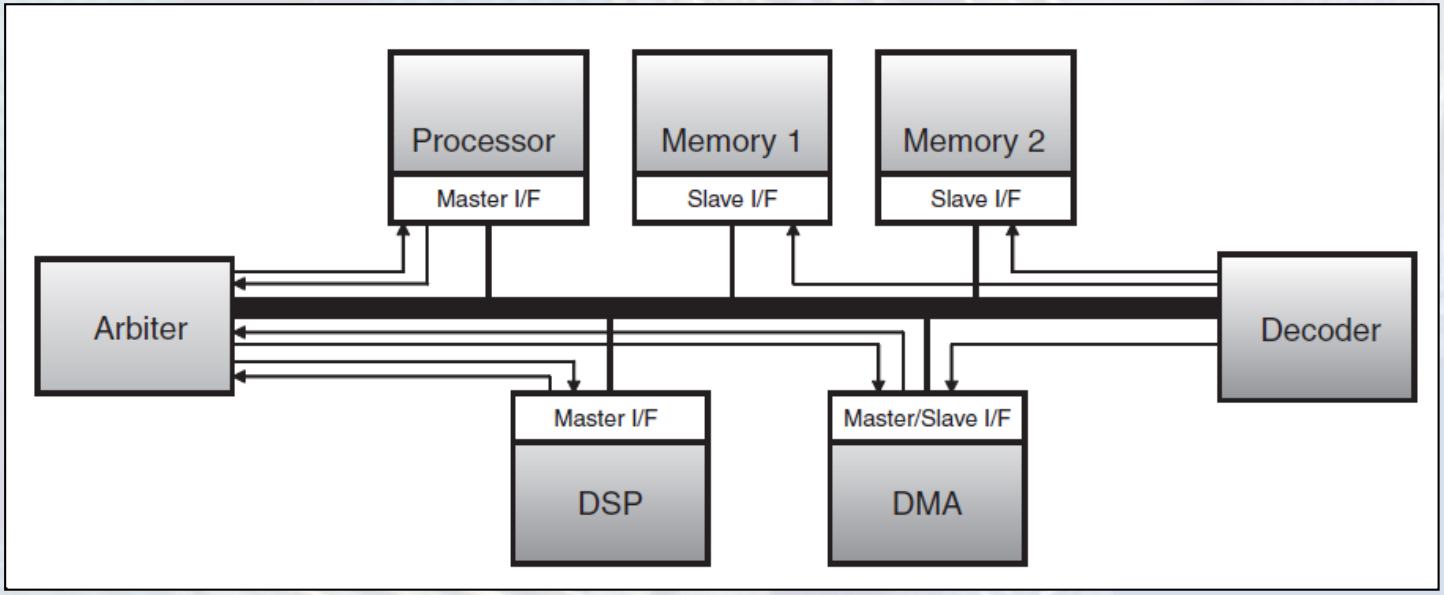
- Egy buszon lehet több MASTER és SLAVE is
  - Több SLAVE nem gond
  - Több MASTER : Arbitráció a buszhasználat jogáért
- Az arbitrációban (döntési folyamatban) csak a MASTER funkciójú egységek vesznek részt
- Az arbitráció a buszhasználat jogáért történik
- Az arbitráció dönti el, hogy versenyhelyzetben ki jogosult a busz használatára.
- Többfajta algoritmus létezik az arbitrációra:
  - Fix prioritás, körbenforgó, utolsó egység nagyobb prioritású, stb.

A buszhasználat szabályai

- **Egy buszon lehet több MASTER és SLAVE is**
  - **Több SLAVE nem gond**
  - **Több MASTER : Arbitráció a buszhasználat jogáért**
- **Az arbitrációban (döntési folyamatban) csak a MASTER funkciójú egységek vesznek részt**
- **Az arbitráció a buszhasználat jogáért történik**
- **Az arbitráció dönti el, hogy versenyhelyzetben ki jogosult a busz használatára.**
- **Többfajta algoritmus létezik az arbitrációra:**
  - **Fix prioritás, körbenforgó, utolsó egység nagyobb prioritású, stb.**

A buszhasználat szabályai

- **A buszhasználati feltételek eldöntése lehet:**
- **Központi:**
  - **Az arbiter értékeli ki a kéréseket és jelöli ki a következő MASTER egységet, aki indíthat egy adatátvitelt**
  - **A dekóder értelmezi a címet/vezérlést és jelöli ki a SLAVE egységet, aki válaszol a kérésre**



A buszhasználat szabályai

- **A buszhasználati feltételek eldöntése lehet:**
- **Elosztott:**
  - **Minden MASTER tartalmaz logikát az arbitrációhoz és így dől el, ki lehet a következő MASTER egység, aki indíthat egy adatátvitelt**
  - **Minden SLAVE egység tartalmaz dekóder áramkört, ami értelmezi a címet/vezérlést és engedélyezi az egységet, hogy a kérésre válaszoljon**
  - **Helyes tervezés esetén csak egy MASTER és egy SLAVE egység lehet egy időben aktív**