# 7. Fejezet
# A processzor és a memória

**The Architecture of Computer Hardware and Systems Software:**
**An Information Technology Approach**

**3rd Edition, Irv Englander**

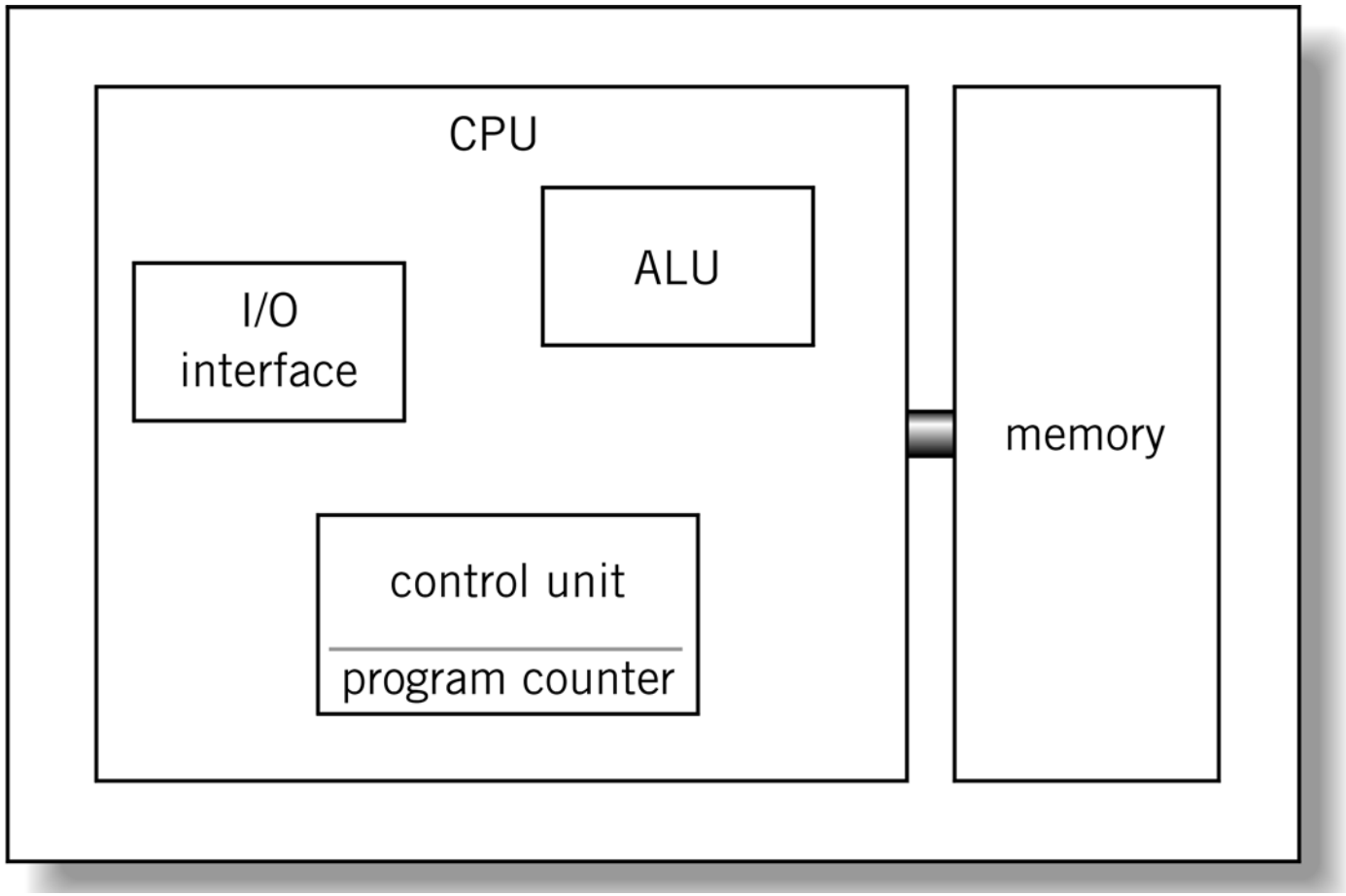


**Wilson Wong, Bentley College**
**Linda Senne, Bentley College**

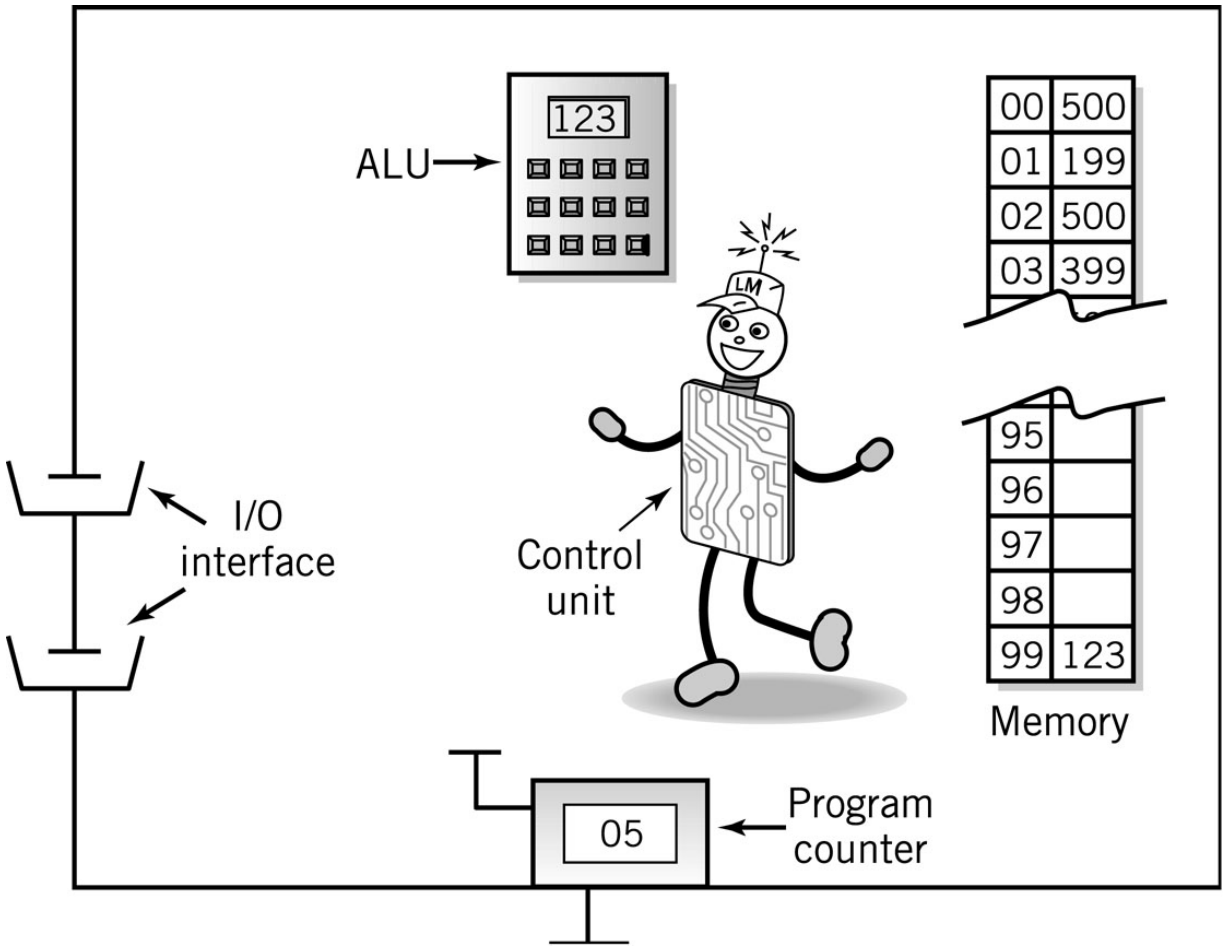
# A CPU három fő része

- *ALU (Arithmetic/Logic Unit = Aritmetikai/Logikai Egység)*
  - Számítási (aritmetikai és logikai), ill. összehasonlító műveleteket hajt végre a tárolt adatokon (regiszterek tartalma változik)
- *CU (Control Unit = Vezérlő Egység)*
  - Vezérli az utasítás-végrehajtást (fetch/execute ciklusokat)
  - Funkciók:
    - A CPU regisztereiben az adatokkal végez műveleteket (nem változnak meg az adatok)
    - Beolvassa a program utasításait és parancsokat ad az ALU-nak
  - Részei:
    - *Memory Management Unit (Memória Vezérlő Egység):*
      - Felügyeli a „fetch”-elését, az utasításoknak és adatoknak
    - *I/O Interfész*
      - néha egybeépítik a memória vezérlővel, melyet *Bus Interface Unit*-nak nevezünk
- *Regiszterek*
  - Példa: *Program counter (PC – program számláló)* vagy instruction pointer (utasítás mutató) meghatározza a következő utasítást a futtatáshoz

# Számítógép sematikus felépítése



# Little Man Computer (LMC)



# Regiszterek

# A regiszterek tulajdonságai

- Kicsi, *permanens (nem múlékony)* tárolóhelyek a CPU-n belül, amelyeket a számolások részeredményeinek tárolására használ
- A Vezérlő Egység (CU) vezérli őket
- *Speciális használatra* vannak tervezve
  - *általában mindegyik adott feladatot lát el*
- Tároló méretük néhány bit, ill. byte
- Tárolhat adatot, címet és utasítást
- Hány regisztere van az LMC-nek?

# Regiszterek

- Regiszterek használata
  - „Scratchpad” („vázlatfüzet”) az aktuálisan futó program számára
    - A gyakran használt, ill. gyorsan elérendő adatokat tárolja
  - A CPU és a futó programra vonatkozó információt tárolja
    - A következő program utasítás címe
    - Külső eszközökből érkező jelek
  - általános, ill. speciális célú regisztereket különböztetünk meg
- **Általános célú regiszterek**
  - *Felhasználó számára láthatóak*
  - Megszakítási és adat értékeket tartalmaz
  - Megegyezik az LMC számológépével
  - Minden processzorban van néhány tucat

# Speciális célú regiszterek

- *Program Count Register =*
  *Program számláló regiszter (PC)*
  - Instruction pointernek (utasítás mutatónak) is hívjuk
- *Instruction Register =*
  *Utasítás regiszter (IR)*
  - Memóriából származó utasításokat tartalmaz
- *Memory Address Register =*
  *Memória Cím Regiszter (MAR)*
- *Memory Data Register = Memória Adat Regiszter (MDR)*
- *Status Registers = Állapot Regiszterek*
  - A processzor és a futó program állapotát tartalmazza
  - *Flag*-eket *(jelzőbit)* (egybites logikai változókat) tartalmaz,
    - nyomon követhessük a számítógép működését
    - túlcsordulás, ill. carry, elektromos kimaradás, egyéb belső hibák.

# Regiszter műveletek

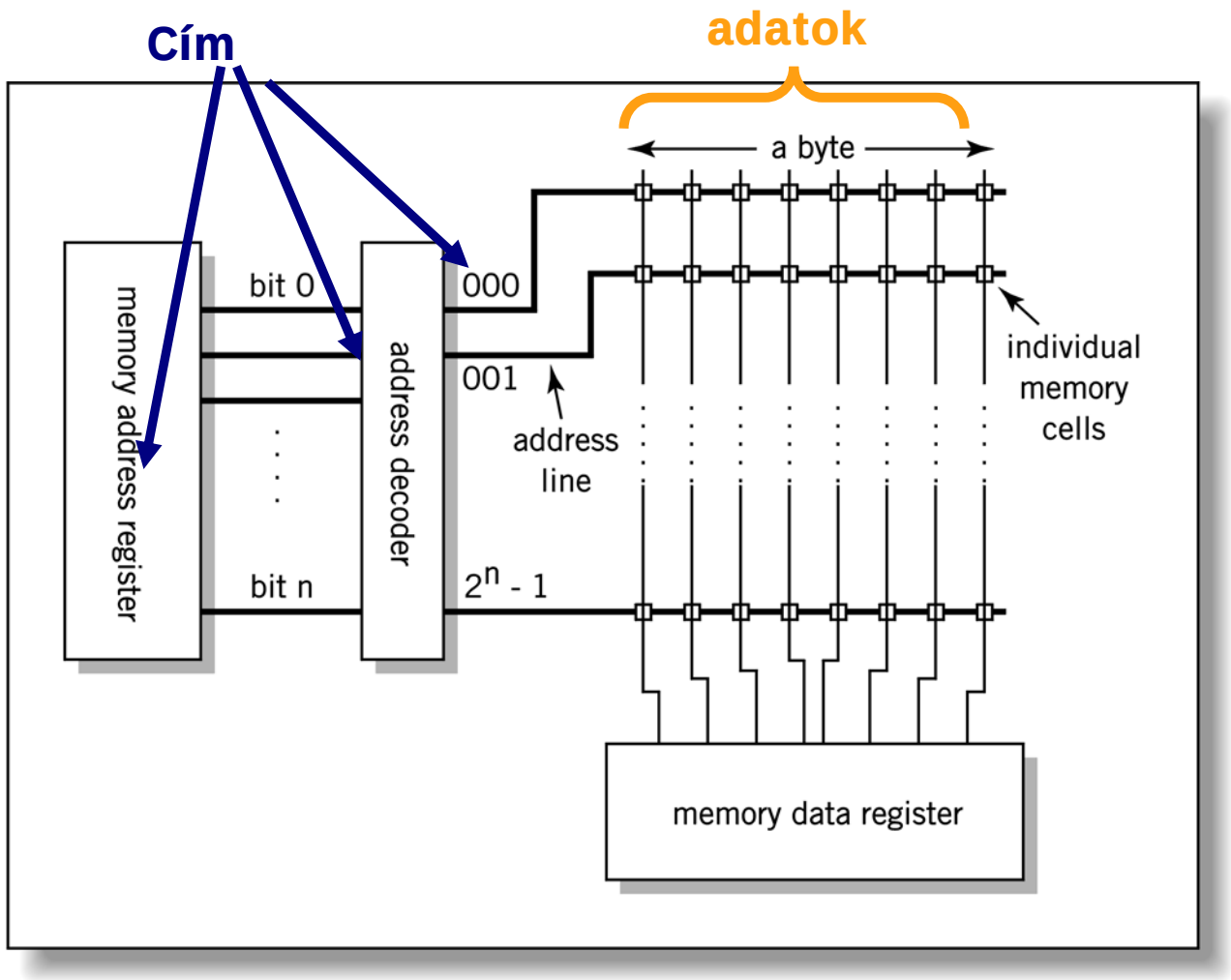
- Más helyekről származó értékek tárolása (regiszterek és memóriák)
- Összeadás és kivonás
- Bit műveletek (Shift, Rotate – Eltolás, Forgatás)
- A regiszter tartalmának ellenőrzése, pl.: nulla vagy pozitív

# Memória

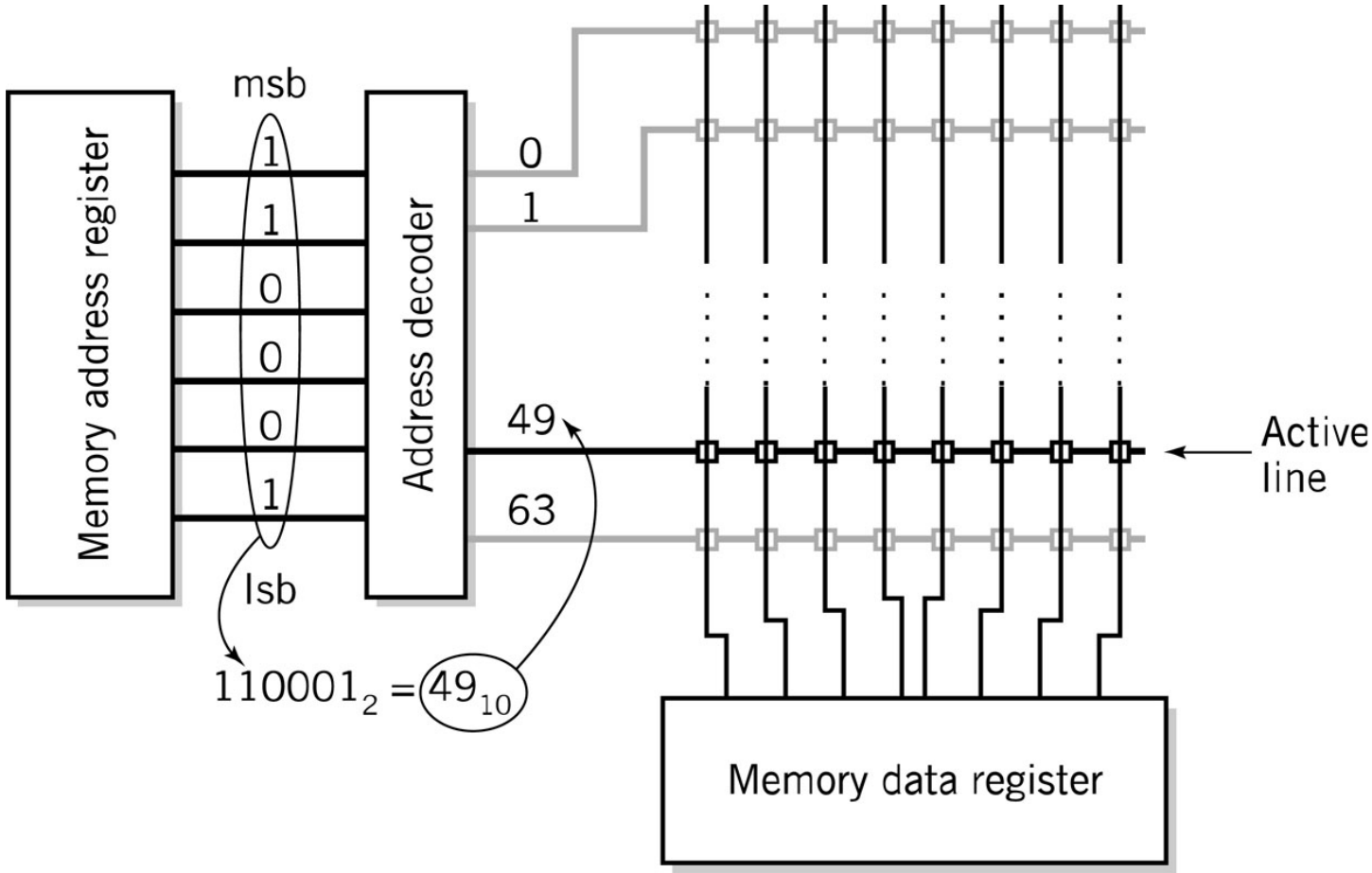
# A memória működése

- Minden memóriatartomány egyedi címmel rendelkezik
- Az utasítás címe a Memória Cím Regiszterbe kerül, amely megtalálja a helyét a memóriában
- A processzor határozza meg, hogy tároljon vagy olvasson adatot
- A Memória Adat Regiszter és a memória közötti szállítási helyet igényel
- A Memória Adat Regiszter kétirányú regiszter

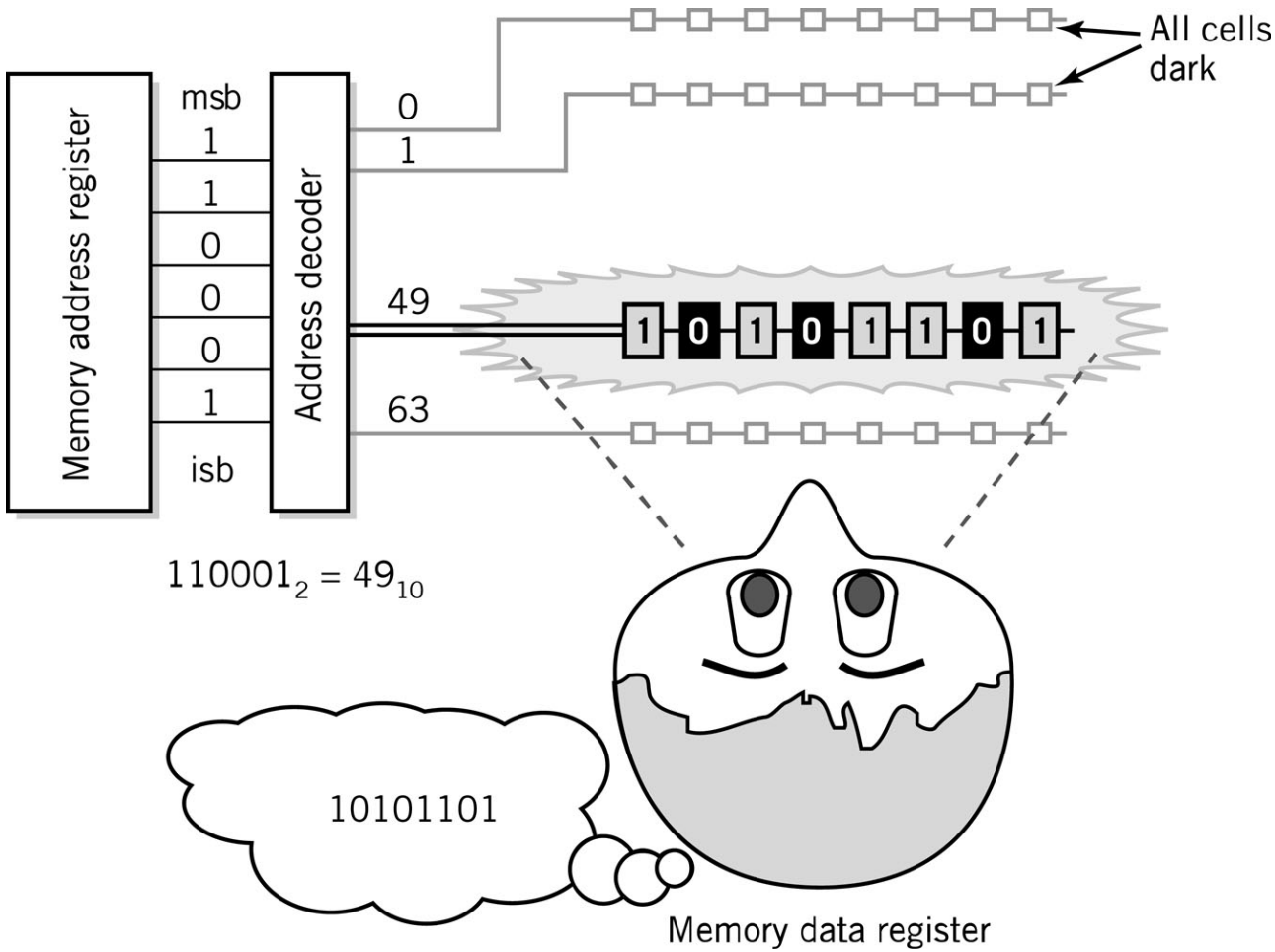
# Kapcsolatok a MAR, a MDR és a memória között



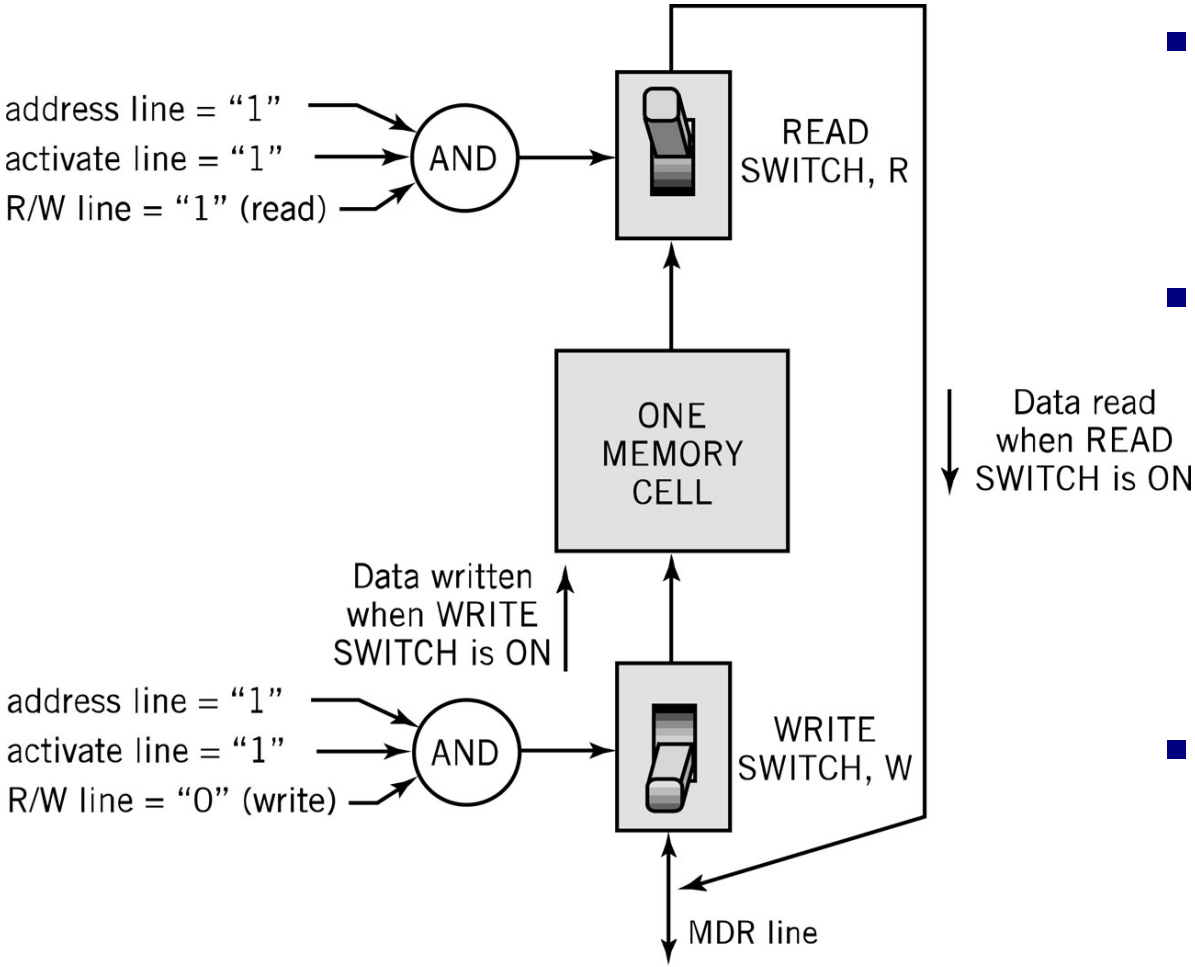
# MAR-MDR példa



# A memória működésének vizuális leírása



# Egy memória cella működése




- Két kapcsoló:
  - Olvasás (Read)
  - Írás (write)
- Két vezérlő áramkör (AND), három vezérlő jellel:
  - address line
  - activate line
  - R/W line
- Információt tároló memória cella

# Memória-kapacitás

- Két tényező határozza meg
  1. A MAR bit-jeinek száma
     - LMC = 100 (00 - 99)
     - $2^K$, ahol K = a regiszter szélessége bit-ekben
  2. Az utasításban szereplő cím rész (operandus) mérete
     - 4 bit 16 helyet tudunk megkülönböztetni
     - 8 bit enged 256 helyet tudunk megkülönböztetni
     - 32 bit 4,294,967,296 vagy 4 Giga helyet tudunk megkülönböztetni
- A memória mérete lényegesen befolyásolja a számítógép teljesítményét
  - A kevés memória arra kényszerítheti a processzort, hogy 50%-alatt dolgozzon

# RAM: Random Access Memory (Véletlen elérésű memória)

- *RAM*
  - eredetileg mágnesezhető tároló cellák, ma DRAM és SRAM
- *DRAM (Dynamic RAM=Dinamikus memória)*
  - Ma a legáltalánosabban használt, olcsó
  - Volatile-s tároló: feszültség kikapcsolásával elvész a tartalma
  - Tartalmát hamar elveszti : másodpercenként kb. 1000-szer kell újraírni a tartalmát az értékek tárolásához
- *SRAM (Statikus RAM)*
  - Gyorsabb és drágább, mint a DRAM (más gyártási technológiája)
  - Volatile-s tároló: feszültség kikapcsolásával elvész a tartalma
  - Keveset használnak belőle a **cache** memóriában, a gyors eléréshez

# ROM - Read Only Memory (Csak olvasható memória)

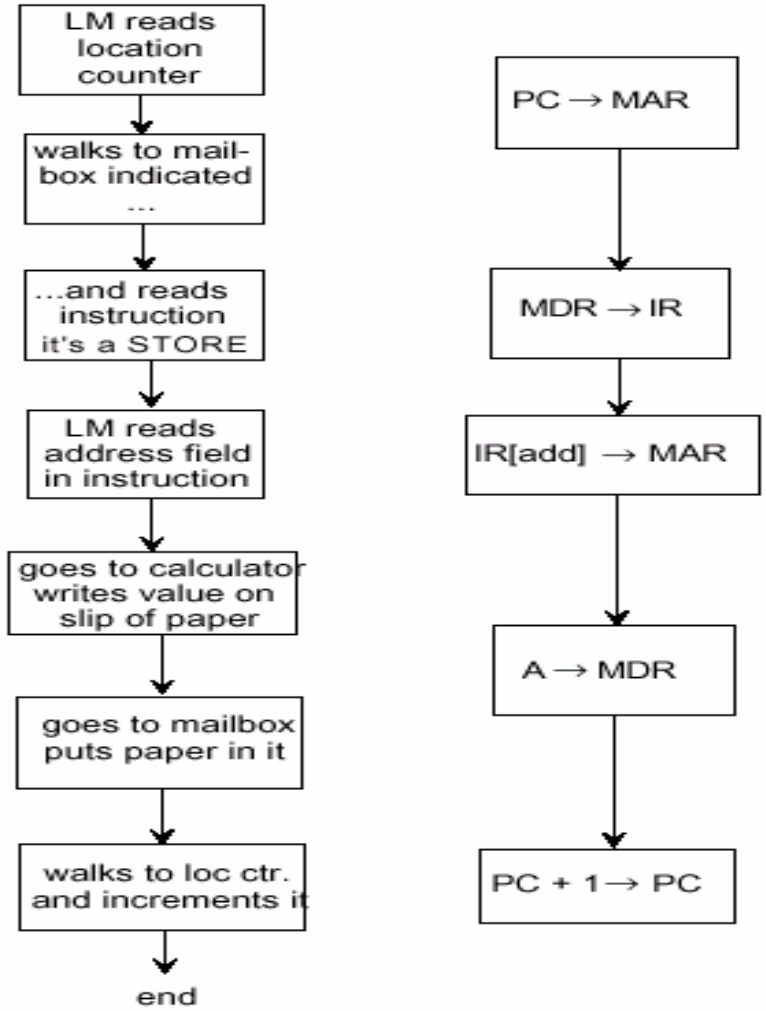
- Stabil, nem volatile-s tároló olyan programok tárolásához, amelyek nem változnak
- Mágneses magmemória
- EEPROM
  - Electrically Erasable Programmable ROM (Elektromosan törölhető és programozható ROM)
  - Lassabb és kevésbé rugalmasan használható, mint a FLASH ROM
- *Flash ROM*
  - Gyorsabb, mint a merevlemezek, de sokkal drágább is
  - Felhasználási területek:
    - BIOS: BOOT program, diagnosztikai programok
    - Digitális kamerák

# CPU utasítások végrehajtása

# Fetch-Execute ciklus

- A CPU műveletek végrehajtása két ciklusra osztjuk (két memória hozzáférésre)
  - mind a utasítás kódja, mind az az adat, amelyen a utasítást végrehajtjuk a memóriában van tárolva
- *Fetch*
  - Megkeressük, ill. dekódoljuk az utasítást, a memóriából a regiszterekbe töltjük és jelzést küldünk az ALU-nak
- *Execute*
  - Végrehajtjuk azt a műveletet, amit az utasítás kódol
  - Adatokat mozgatjuk, ill. módosítjuk

# LMC vs. CPU
# Fetch és Execute ciklus



# A LOAD utasítás Fetch/Execute ciklusa

| | |
|---|---|
| 1. PC -> MAR | Az adatokat a PC-ből a MAR-ba mozgatja |
| 2. MDR -> IR | Az adatokat az IR-be mozgatja |
| 3. IR(address) -> MAR | Az utasítás cím része (operandus) betöltődik a MAR-ba |
| 4. MDR -> A | Az aktuális adatokat a tároló regiszterbe mozgatja |
| 5. PC + 1 -> PC | A programszámláló növelése |

# A STORE utasítás Fetch/Execute ciklusa

| | |
|---|---|
| 1. PC -> MAR | Az adatokat a PC-ből a MAR-ba mozgatja |
| 2. MDR -> IR | Az adatokat az IR-be mozgatja |
| 3. IR(address) -> MAR | Az utasítás cím részletei betöltődnek a MAR-ba |
| 4. A -> MDR* | A tároló regiszterből az MDR-be mozgatja az adatokat |
| 5. PC + 1 -> PC | A programszámláló növelése |

* Vegyük észre, hogy a 4. pont hogyan változott meg a LOAD utasításhoz képest! Mikor történik a memória használata a két esetben?

# Az ADD utasítás Fetch/Execute Ciklusa

| | |
|---|---|
| 1. PC -> MAR | Az adatokat a PC-ből a MAR-ba mozgatja |
| 2. MDR -> IR | Az adatokat az IR-be mozgatja |
| 3. IR(address) -> MAR | Az utasítások cím részletei betöltődnek a MAR-ba |
| 4. A + MDR -> A | Az MDR tartalma a tár tartalmához adódik |
| 5. PC + 1 -> PC | A programszámláló növelése |

# LMC utasításainak Fetch/Execute ciklusai

**Kivonás**

PC → MAR

MDR → IR

---

IR[addr] → MAR

A – MDR → A

PC + 1 → PC

**Bemenet**

PC → MAR

MDR → IR

---

IOR → A

PC + 1 → PC

**Kimenet**

PC → MAR

MDR → IR

---

A → IOR

PC + 1 → PC

**Szünet**

PC → MAR

MDR → IR

---

**Ugró (elágazás) utasítás**

PC → MAR

MDR → IR

---

IR[addr] → PC

**Feltételes (elágazás) ugró utasítás**

PC → MAR

MDR → IR

---

Ha a feltétel hamis: PC + 1 → PC

Ha a feltétel igaz: IR[addr] → PC

# Bus-ok

# Bus

- Fizikai kapcsolat, ami lehetővé teszi az adatátvitelt a számítógépben két pont között
- Elektromos csatlakozások csoportja a két csomópont közötti jelátvitelhez
  - *Line (vonal)*: csatlakozási pont a buszon
- A jeleknek négy fajtáját különböztetjük meg
  1. Adat (alfa-numerikus, numerikus, utasítások)
  2. Cím
  3. Vezérlő jelek
  4. Táp (néha)

# Bus-ok a számítógépben

- Processzor és memória közötti bus
- Az I/O perifériák lehetnek azonos bus-on a CPU-val és a memóriával vagy külön buszon
- Ha a (plug-in) I/O elemek a CPU és a memória közötti bus-t használják, akkor a fizikai egységet *backplane-nek* hívják
  - *Rendszer bus-nak és külső bus-nak is hívják*
  - Ez jó példa a „*broadcast” bus-ra*
  - A PC-kben általában ezeket az elemeket egy nyomtatott áramköri lapra az *alaplapra (motherboard)* integrálják
    - ez tartalmazza a CPU-t és összekapcsolja azt a többi komponenssel

# Bus jellemzői

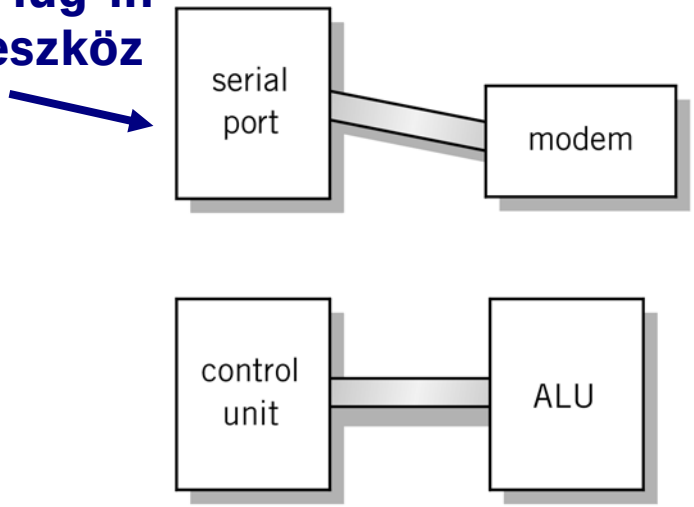
- Protokoll
  - A kommunikáció jól dokumentált leírása
  - Világosan megmagyarázza az összes vonal és a vonalakon lévő minden jel jelentését
- Adatátviteli kapacitás:
  - az adat átvitel mértékegysége bit/másodperc
- Adat-bus szélessége:
  - Az egyszerre átvihető adatok szélessége bit-ekben

# Bus topológiák és üzenetváltás

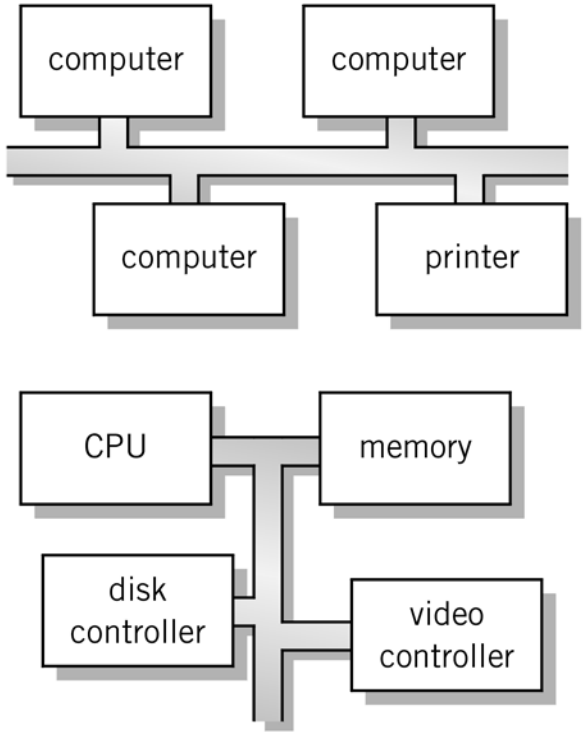
- Pont-pont kapcsolat
- Többpontos kapcsolat
  - „Broadcast bus”: minden elem megkapja a bus-ra kitett adatokat

# Pont-pont kapcsolat vs. több pontos

**Plug-in eszköz**




examples of point-to-point buses



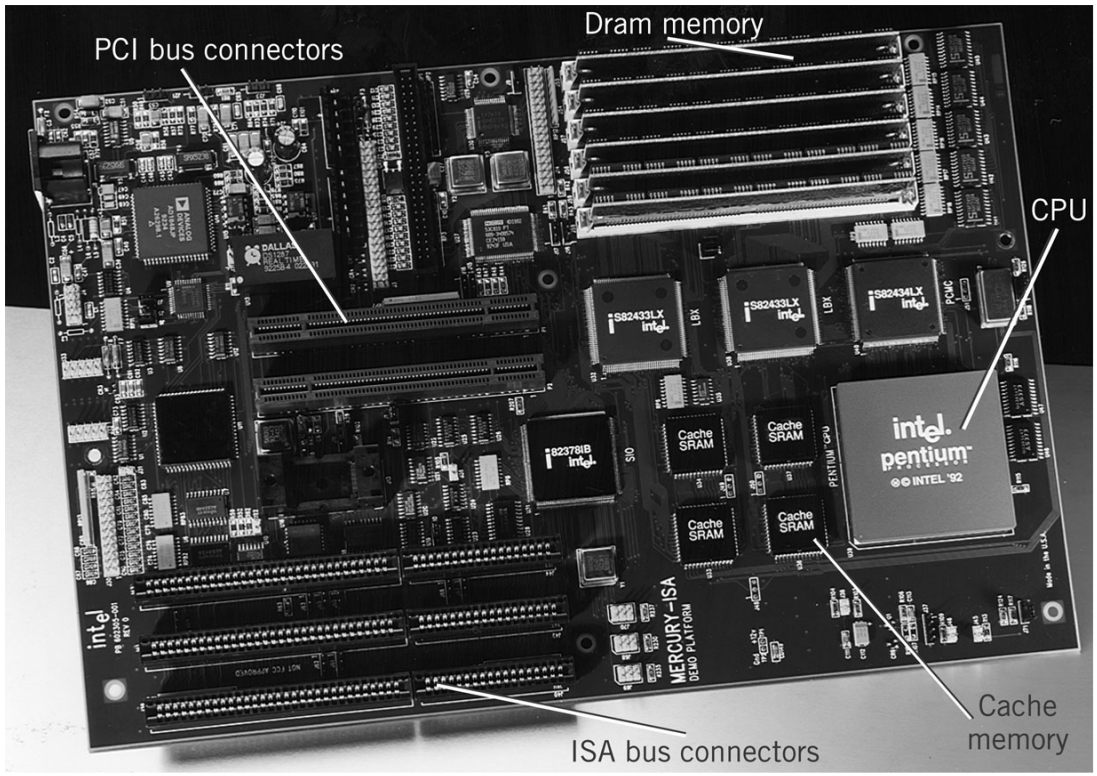

examples of multipoint buses

**Broadcast bus példa: Ethernet**

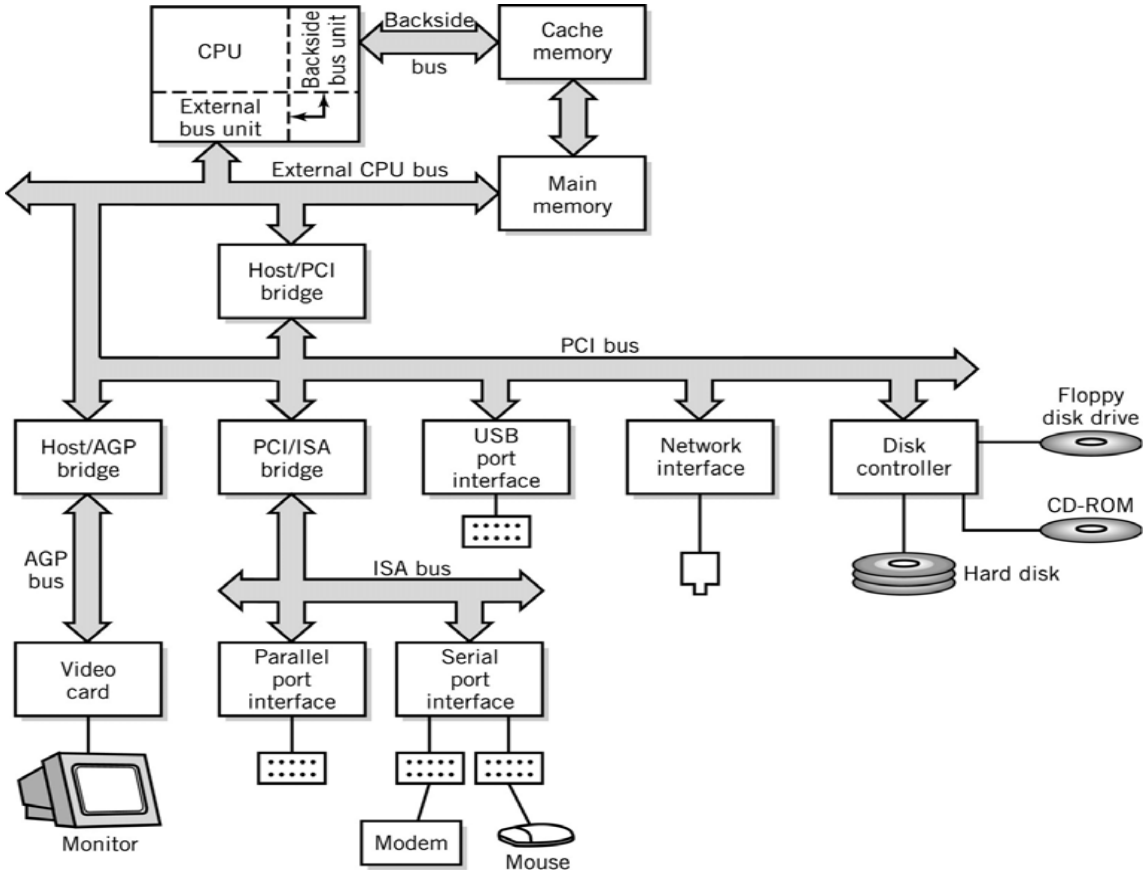
**Az összetett eszközök mindegyike között van kapcsolat**

# Az alaplap

- Nyomtatott áramkör, melyen a processzor és egyéb fő elemek találhatók

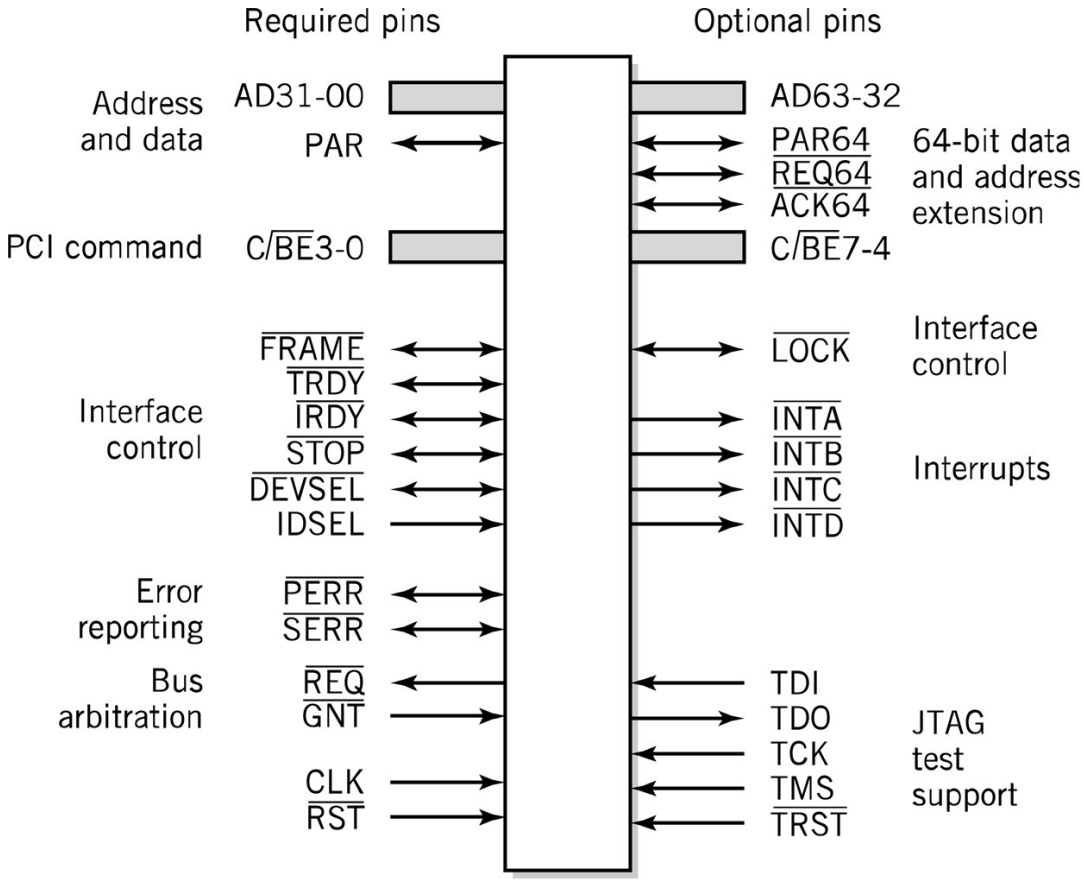


# Összeköttetések egy tipikus PC-ben




*Bus interface bridge-ek:*
különböző típusú bus-okat kapcsolnak össze, átviszik az adatokat az egyik bus-ról a másikra

# PCI bus-on használt jelek




Source: Copyright © PCI Pin List/PCI Special Interest Group, 1999.

# A CPU utasítás készlete

# Utasítások

- Utasítás
  - A számítógépet vezérlő parancsok
  - Végrehajtásukkor elektromos jelek hatására a számítógép egyes áramkörei különböző műveleteket hajtanak végre
- Utasítás készlet
  - A tervező által definiált funkciók, amelyet a processzor végre tud hajtani
  - A különböző architektúrájú számítógépek között különbséget tehetünk a következő szempontok alapján:
    - Végrehajtható utasítások száma
    - Az egy utasítással végrehajtható műveletek bonyolultsága szerint
    - Támogatott (feldolgozható) adattípusok
    - Utasítások formátuma (kötött vagy változó hosszúság)
    - Regiszterek használata
    - Címzés (cím mérete, címzési módok)

# Utasítás részei

- Műveleti kód (OP code):
  - a feladat, amit a számítógépnek végre kell hajtani
- Forrás operandus
- Eredmény operandus

**Címek**

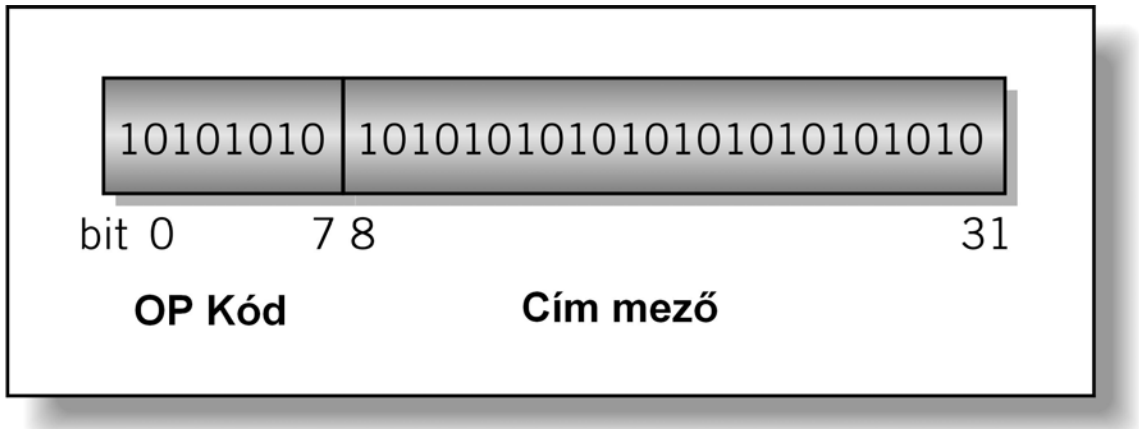
  - Adat helyének meghatározása (regiszter, memória)
    - Explicit: tartalmazza az utasítás
    - Implicit: alapértelmezés szerinti helyen

| OP code | Forrás operandus | Eredmény operandus |
|---|---|---|

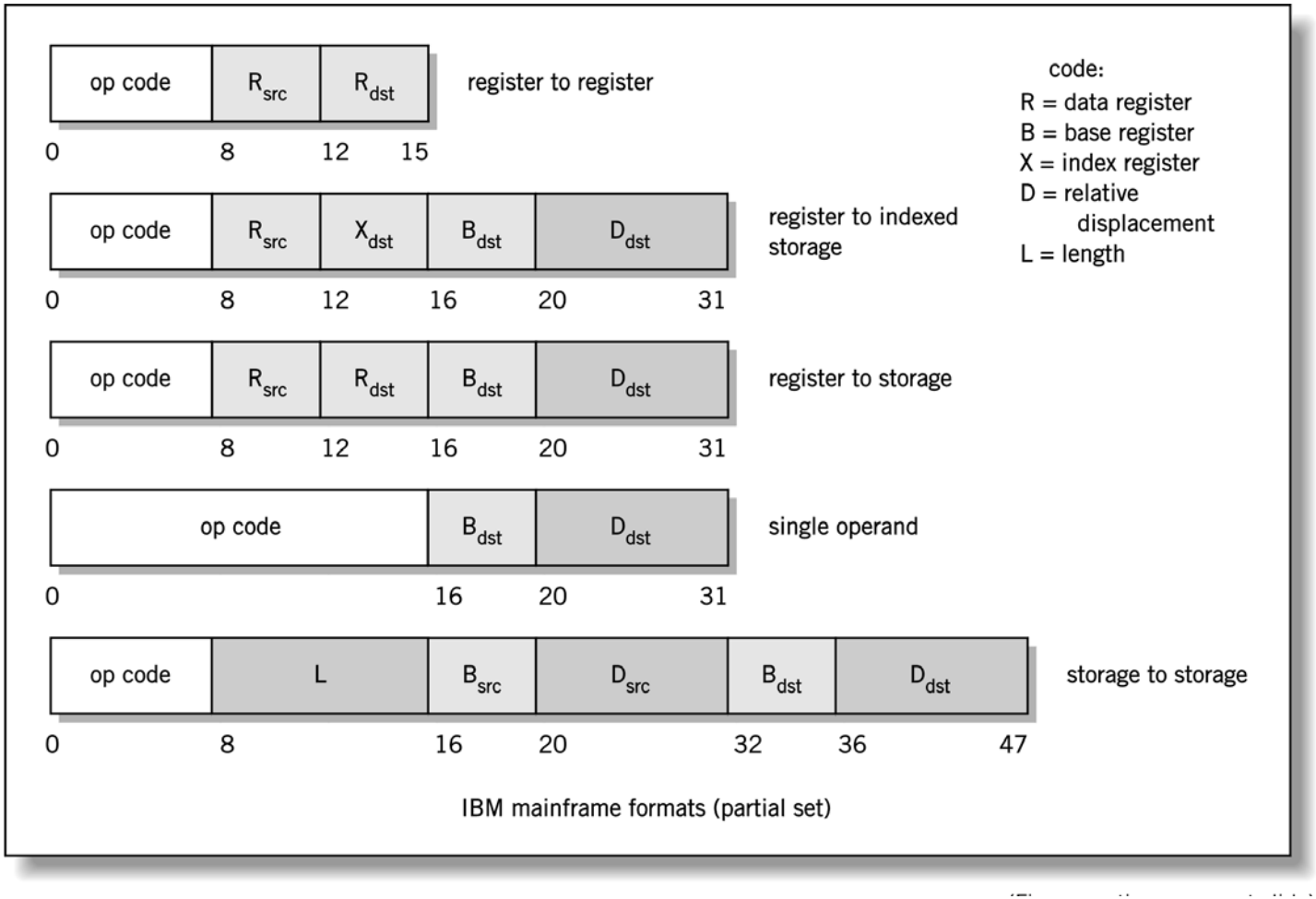
# Utasítások formátuma

- *Számítógép-specifikus* leírás, amely meghatározza:
  - műveleti kód hosszát
  - Operandusok számát
  - Operandusok hosszát

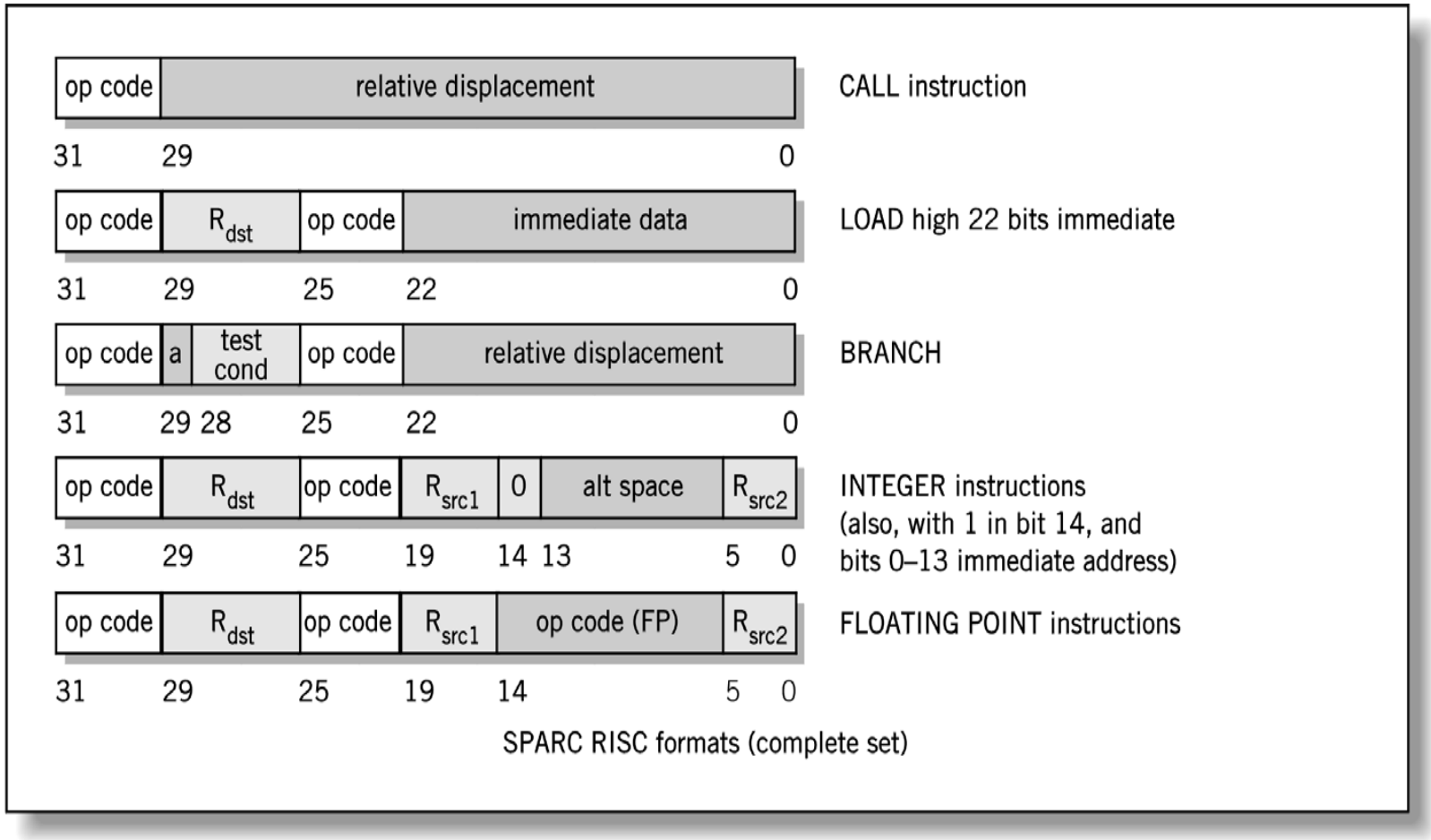
**Egy egyszerű 32-bites utasítás formátum**



# Utasítás formátumok: CISC



# Utasítás formátumok: RISC




SPARC RISC formats (complete set)

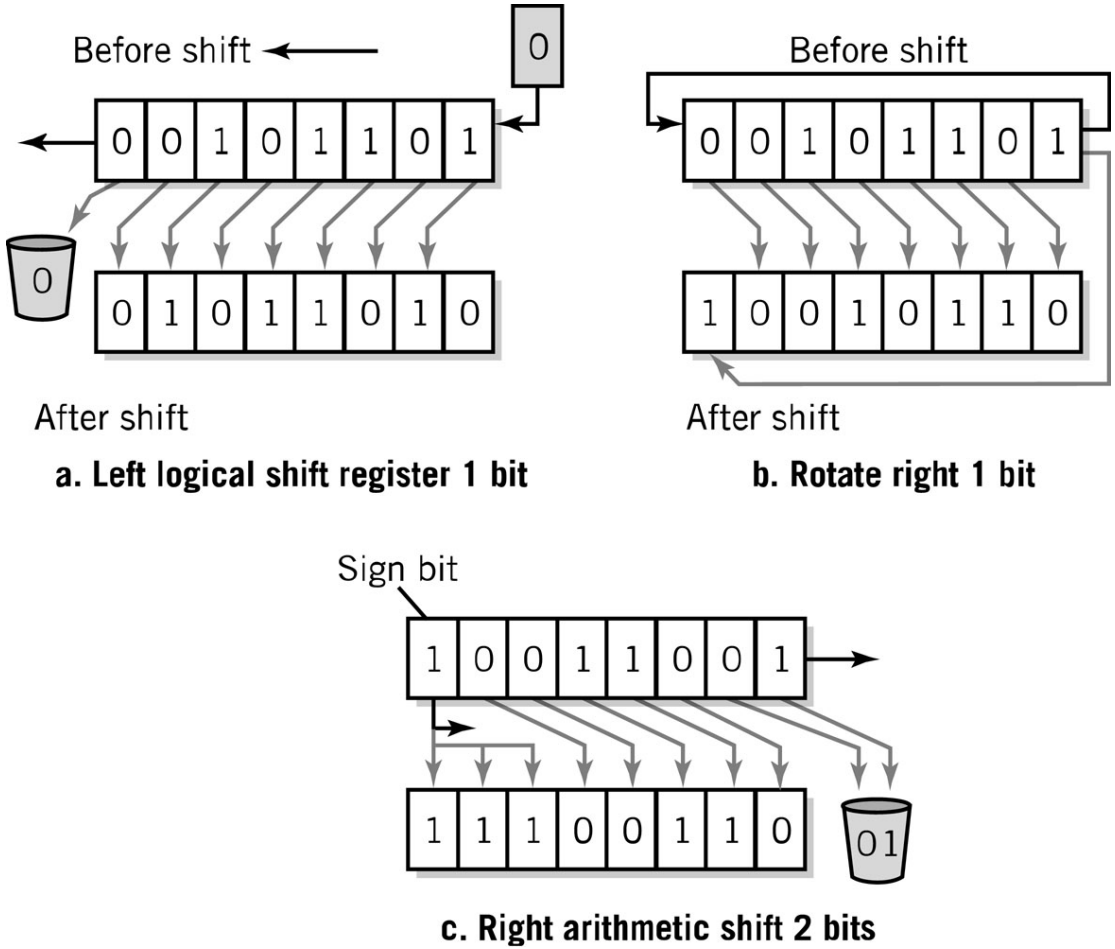
# Utasítások típusai

- Adat átviteli műveletek (load: betöltés, store: tárolás)
  - Általánosan használt, változatos megvalósítások
  - A memória és a regiszterek közötti kapcsolat
  - Mekkora a *word* adattípus?  16? 32? 64 bit?
- Aritmetikai utasítások
  - operátorok: **+ - / * ^**
  - Egész és lebegőpontos adatokon is végezhető
- Boolean logikai műveletek és relációs operátorok
  - Relációs operátorok:  **> < =**
  - Logikai operátorok: **AND, OR, XOR, NOR,** és **NOT**
- Egyetlen operandust használó utasítások
  - Negáció, dekrementálás, inkrementálás

# Egyéb utasítás típusok

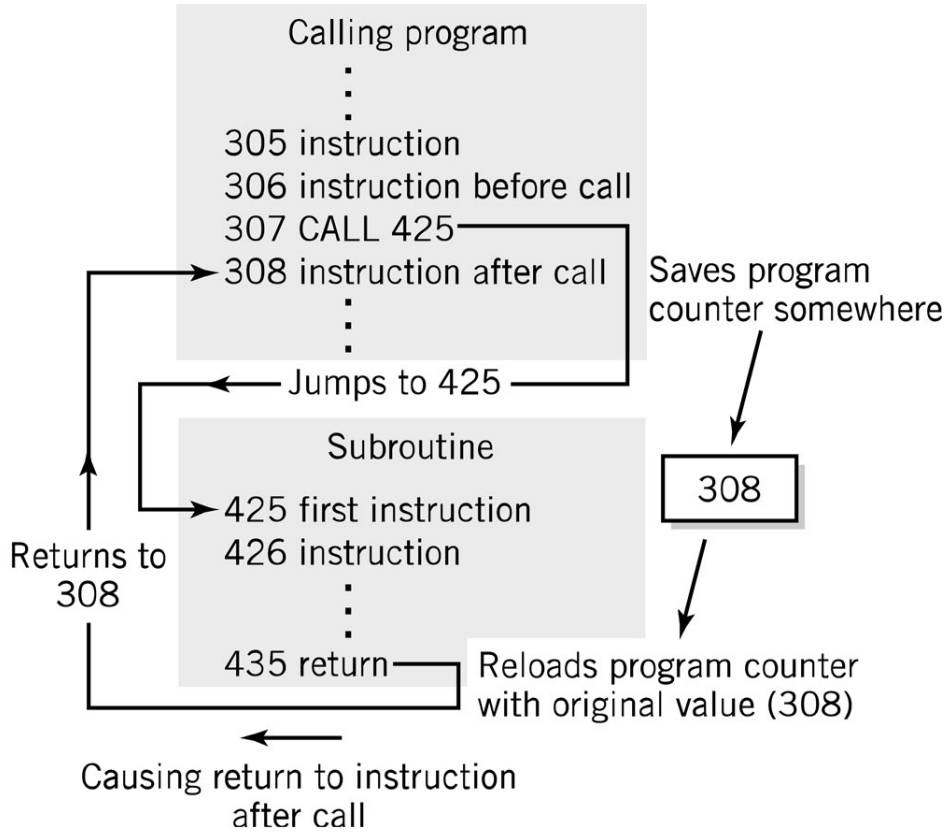
- Bit-manipuláló utasítások
  - Flag-ek (jelzőbitek) a feltételek tesztelésére
- Bit-enkénti  eltolás, ill. forgatás
- Program vezérlő utasítások
- Verem tár kezelő utasítások
- Összetett adatelemeken végzett utasítások
- I/O elemeket és számítógépet vezérlő utasítások

# Regiszterben tárolt értékek eltolása és forgatása



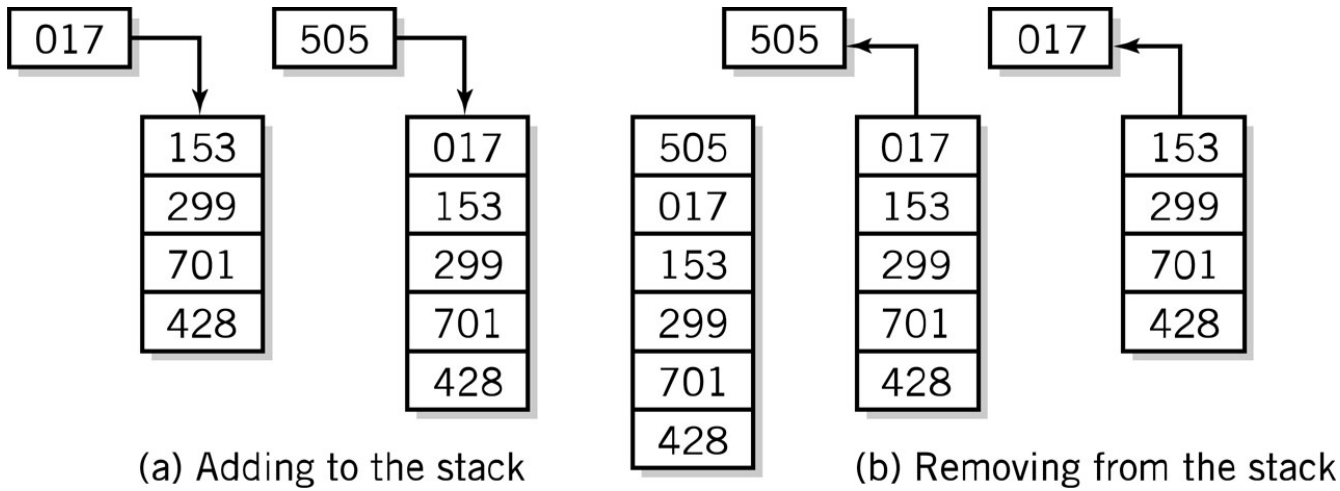
# Program vezérlő utasítások

- Program vezérlése
  - Ugrás és elágazás utasítások
  - Szubrutin hívása és visszatérés
    - CALL (CÍM)
    - RETURN



# Veremtár kezelő utasítások

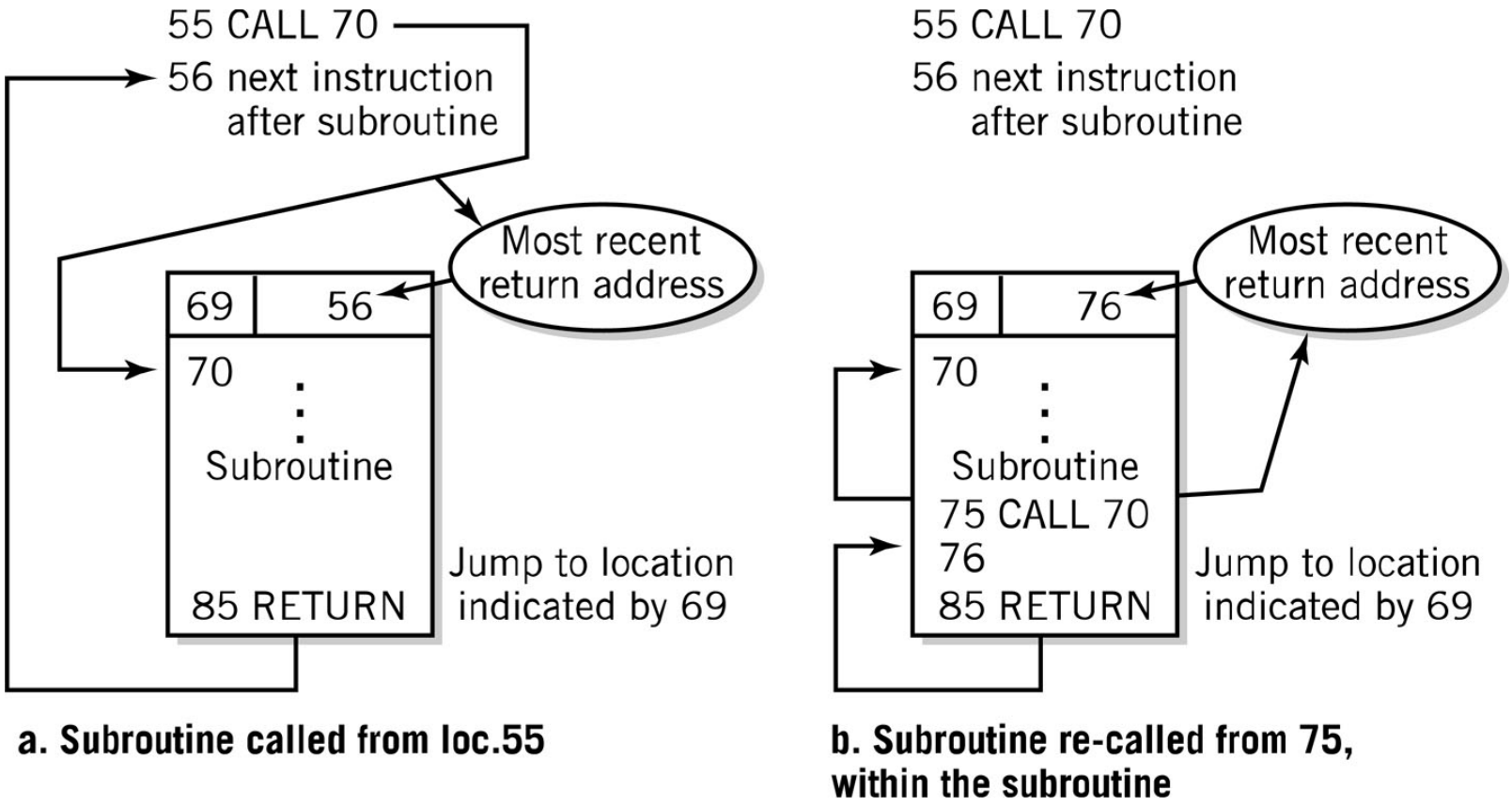
- Veremtár kezelő utasítások
  - LIFO (utolsó be, első ki) működés, információ tárolása
  - Elemek kivétele fordított sorrendben történik, mint az elemek belehelyezése




Beszúrás

Kivétel

# Szubrutin visszatérési címének tárolása állandó helyen : *Oops!*

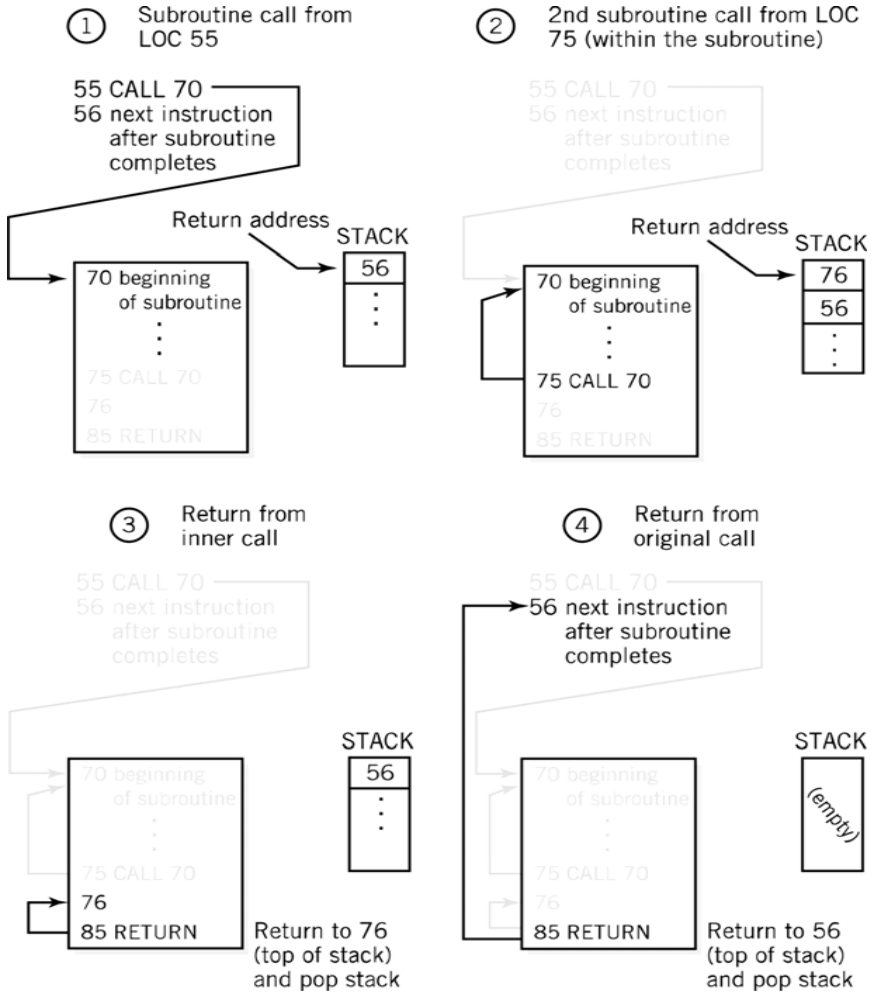



**a. Subroutine called from loc.55**

**b. Subroutine re-called from 75, within the subroutine**

***Hibás működés lehetséges!!***

# Veremtár a szubrutin visszatérési címének tárolására



# Összetett adatkezelő utasítások

- Párhuzamosan hajt végre egyszerű műveleteket adatok csoportján
  - SIMD:  Single Instruction, Multiple Data
    - egy utasítás, több adat
  - Intel MMX™:  57 multimédiás utasítás
  - Általánosan használják vektorok és tömbök feldolgozásakor