Hardware alapismeretek

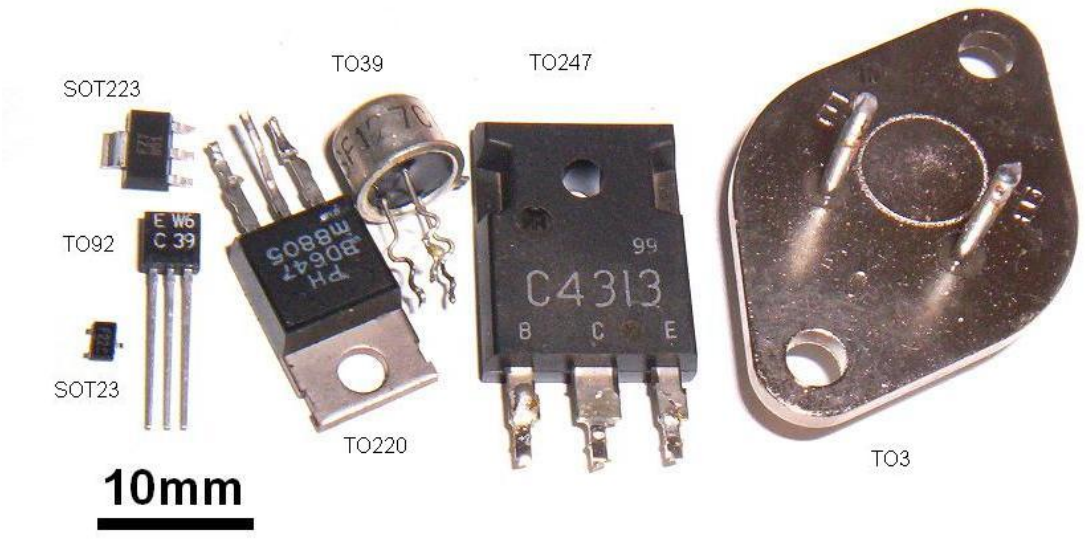
Sulyok András Attila

2018. 10. 04.[1]

Bevezetés a számítástechnikába

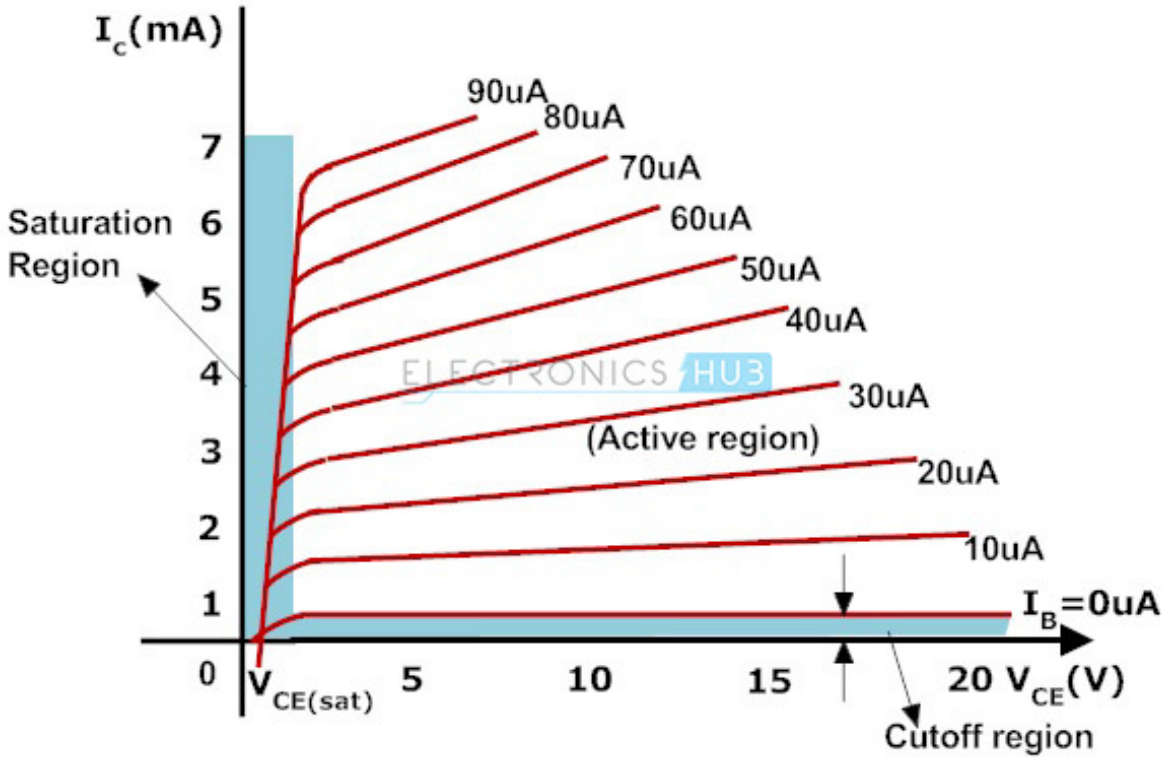
[1] Utoljára módosítva: 2018. október 18.

Tranzisztor



Kép forrása: Wikipedia

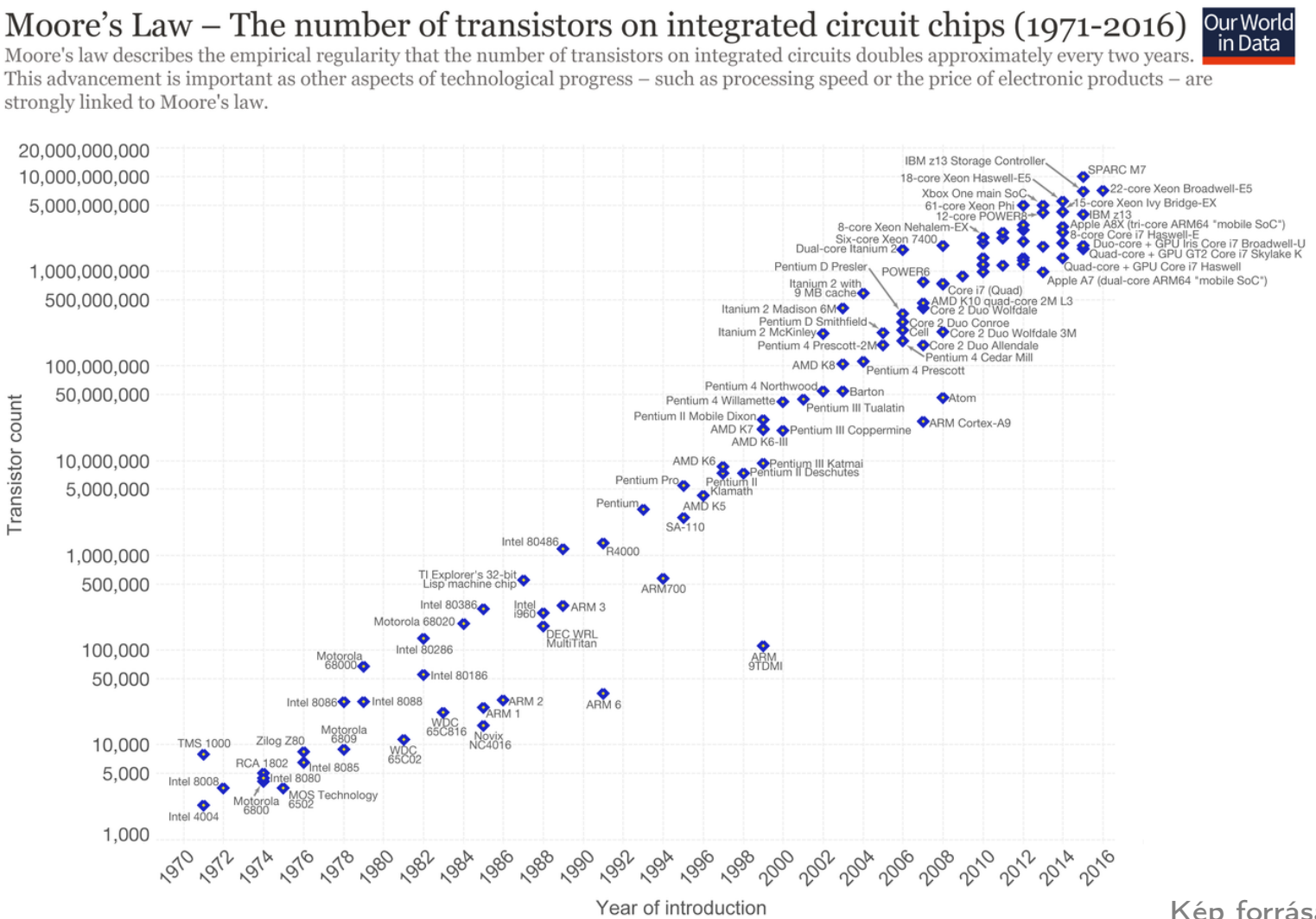
Tranzisztor



Kép forrása: Electronics Hub

Moore törvénye

Megfigyelt jelenség: a tranzisztorok száma exponenciálisan nő



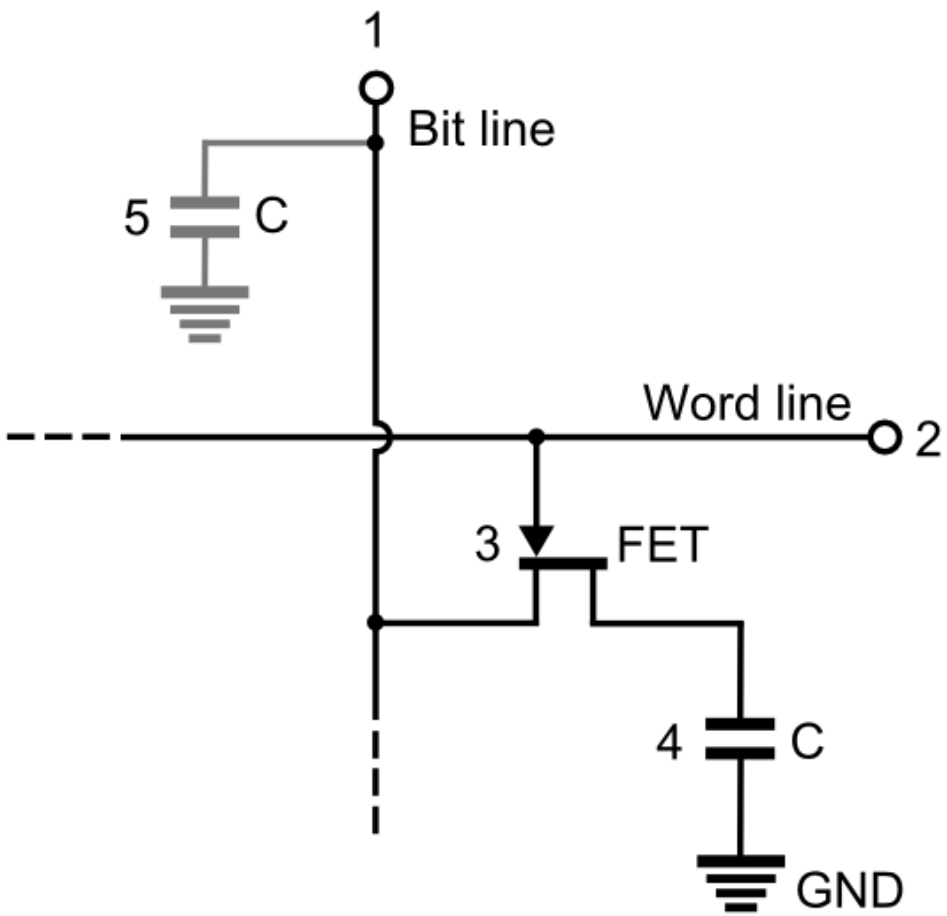
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

Kép forrása: Wikipedia

Tároló elemek

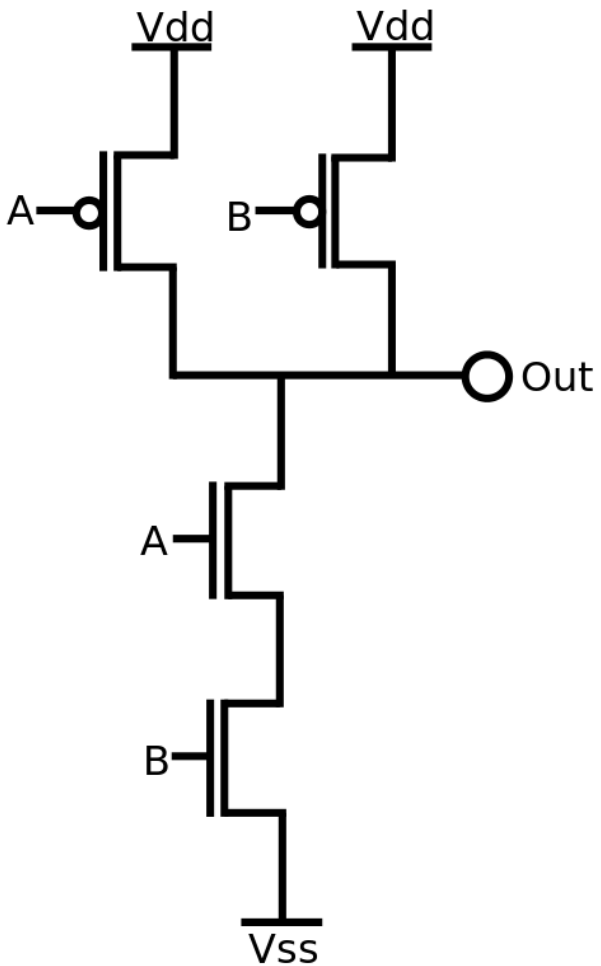
DRAM cella



Kép forrása: Wikipedia

Logikai kapuk

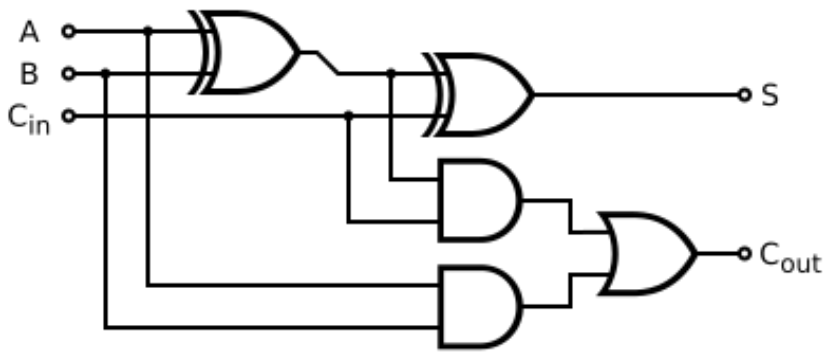
NAND kapu (Not-And: $\neg(A \wedge B)$)



Kép forrása: Wikipedia

Aritmetikai áramkörök

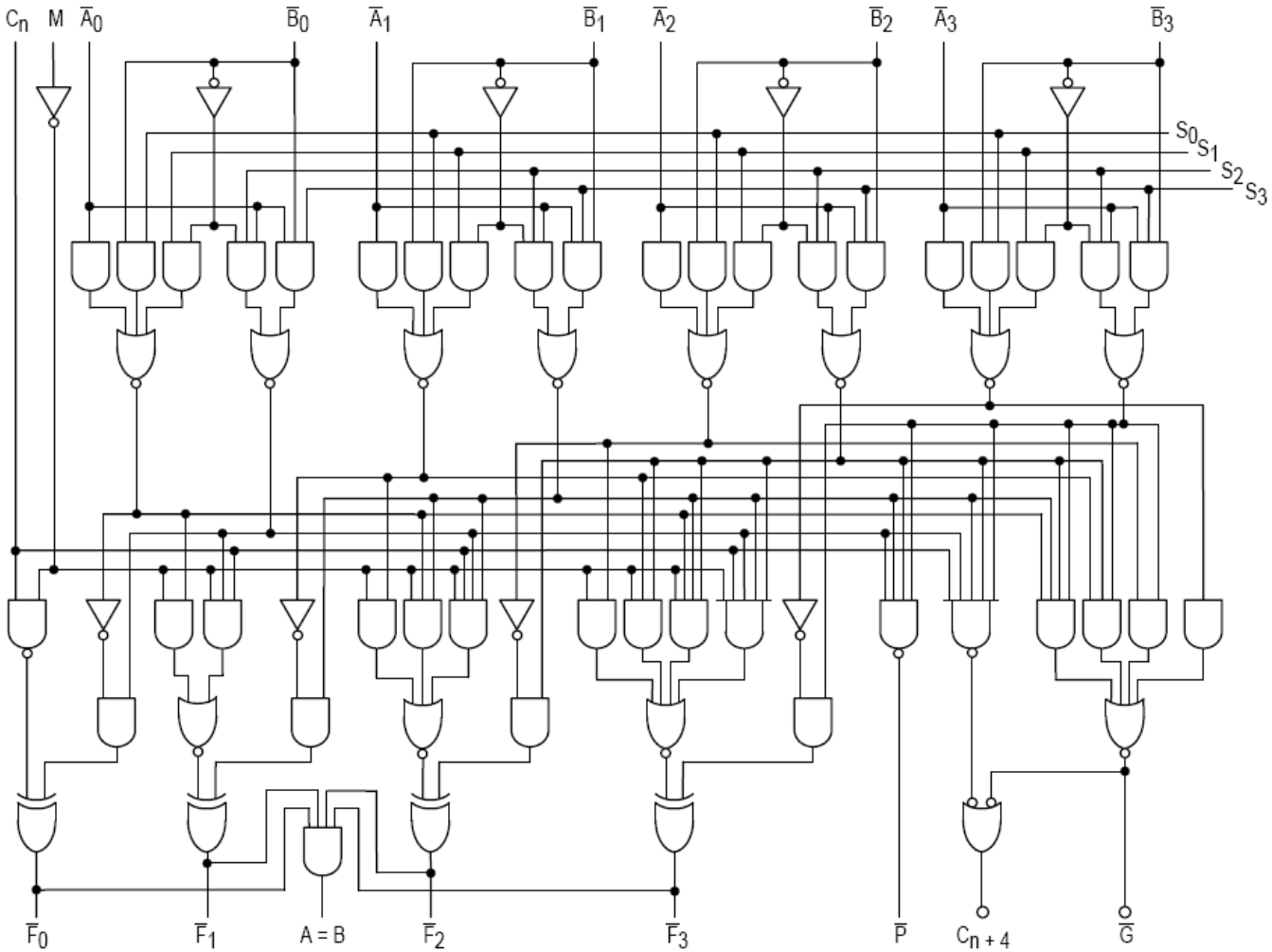
Összeadó áramkör (carry-vel)



Kép forrása: Wikipedia

Arithmetic Logical Unit (ALU)

A 74181, egy 4 bites ALU



Kép forrása: Wikipedia

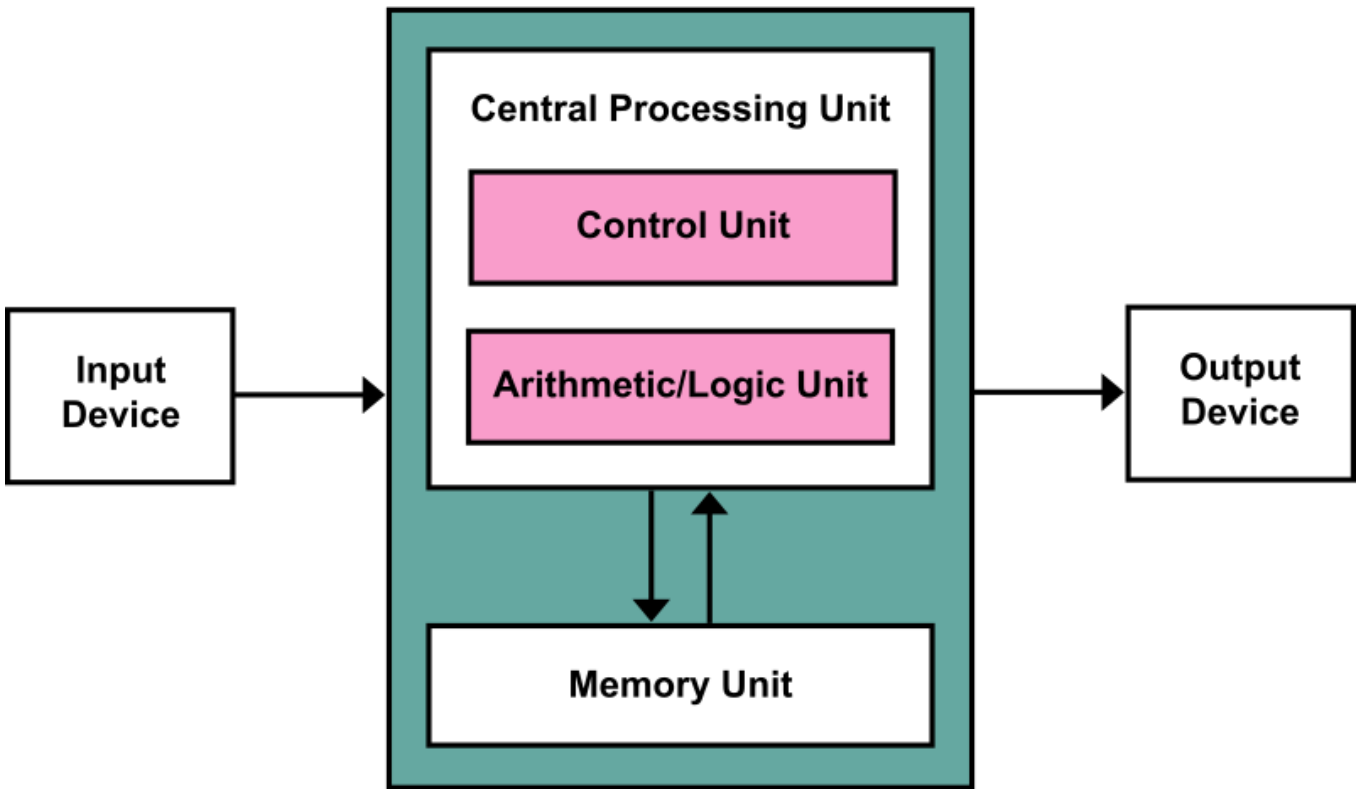
Neumann architektúra

Neumann elvek

- Részegységek:
  - központi aritmetikai egység
  - vezérlő egység
  - memória
  - Input/Output egység
  - Óra
- A végrehajtott program és a tárolt utasítások egy memóriában legyenek
- Elektronikus és digitális működés
- Bináris számrendszer használata

Neumann architektúra

A Neumann architektúra blokkdiagrammja



Kép forrása: Wikipedia

Neumann architektúra

Hátrányai

- program megváltoztathatja a saját kódját (pl.: buffer overflow-val)
- hatékonyság:
  - Nem lehet az adatot és az utasításokat párhuzamosan elérni
  - sávszélesség korlát (egy buszrendszer)

Harvard architektúra

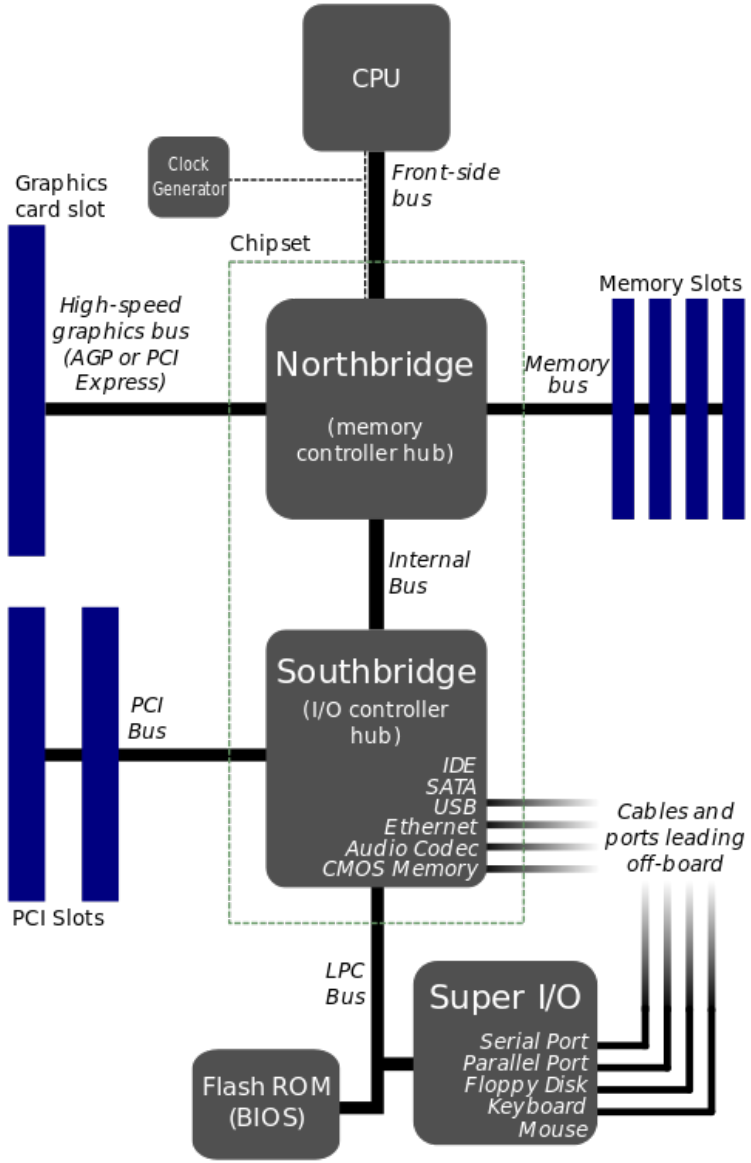
Adat- és utasításmemória különválasztva

Különböző paramétereket lehet alkalmazni, pl.:

- utasítások csak olvasható (ROM) memóriában
- szóhosszúság
- címzés

Általában együtt használják a két architektúrát, pl.: cache rendszerek

Egy modern alaplap blokkvázlata



Kép forrása: Wikipedia

Buszok

Különböző komponensek közötti összeköttetés

- vezérlőbusz: kommunikáció lefolytatása
- címbusz: üzenet célja
- adatbusz

A kommunikáció valamilyen protokoll alapján történik

- aszinkron
- szinkron

Master: a kommunikációt kezdeményező eszköz
Slave: a kommunikációt fogadó eszköz

Aszinkron kommunikáció

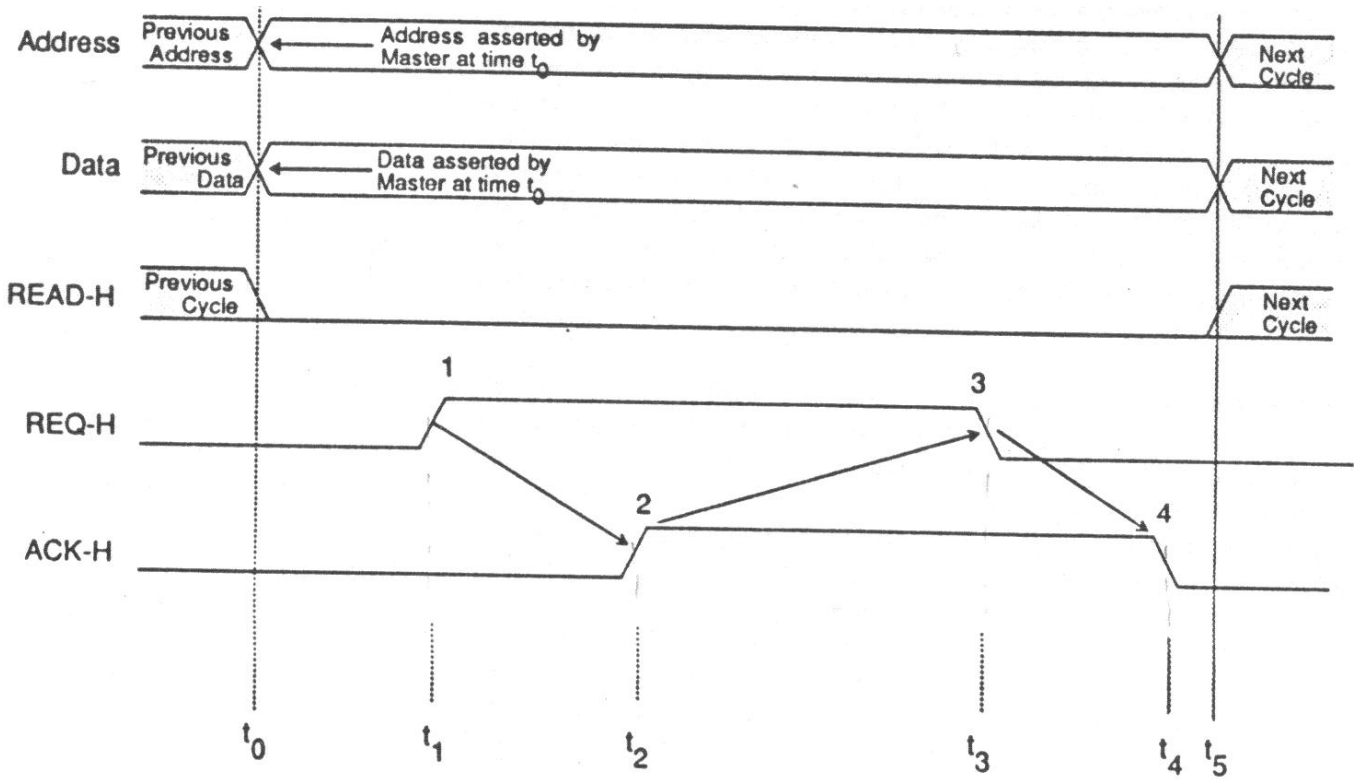
Nincs közös órajel,
a vezérlőjelekkel történik az időzítés. (Handshake)

Előny:
egyszerűbb, nem kell órajel

Hátrány:
késleltetések (meg kell várni, amíg a jel eljut a címzetthez),
vezetékek hossza korlátozott

Aszinkron kommunikáció

Master ír a Slave-nek



Kép forrása: DigRend előadás

Szinkron kommunikáció

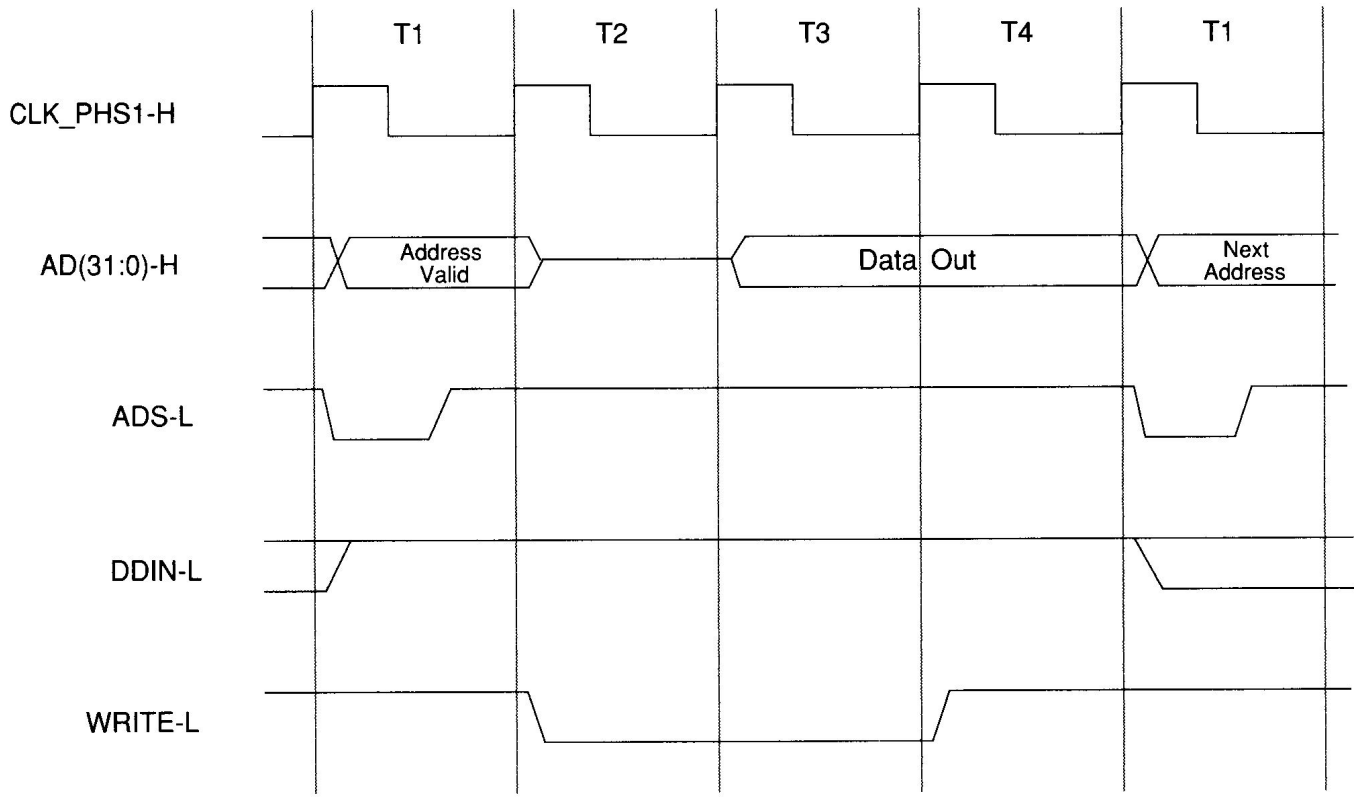
A kommunikációt közös órajel szinkronizálja.

Órajel: valamilyen periodikus jel, meghatározott frekvenciával és szélességgel

Előny: gyorsabb, nincs szükség handshake-re
Hátrány: az órajelet a leglassabb eszközhöz kell igazítani

Szinkron kommunikáció

NS32332 Busz írási tranzakciója



Kép forrása: DigRend előadás

I/O adatátvitel

Hogyan kerül a memóriába az adat a perifériákról?

- Programozott I/O: CPU (software) átmásolja az adatokat legegyszerűbb, de terheli a CPU-t
- Direct Memory Access (DMA): egy DMA vezérlő végzi az adatátvitelt
  CPU csak megindítja a tranzakciót

I/O adatátvitel

Hogyan címzi a CPU az adatátvitelt?

- Memory-mapped I/O: közös címtér a memóriával
  az eszközök figyelnek a címükre, és csak a sajátjukra válaszolnak
- Port-mapped I/O: külön CPU utasítással történik az adatátvitel

Honnan tudja a CPU, hogy adatot kell olvasnia egy eszközről?

Megszakítás (interrupt):
a CPU abbahagyja, amit csinál, és végrehajtja a megfelelő függvényt
külön vezetékek vannak erre rendelve

Peripherial Component Interconnect (PCI)

Párhuzamos, szinkron busz, gyors kiegészítések (expansions) összekötésére a rendszerbuszhoz.
Pl.: grafikus kártya, USB csatlakozók, SSD, hálózati csatlakozók bekötésésre

33, majd 66, illetve 133 MHz frekvencián működik, 32 vagy 64 bites.
A frekvenciát már csak a kábel hossza árán lehetett növelni, illetve a különböző hosszúságú vezetékeken a jel nem egyszerre érkezett meg.

Egy trükk a sebességnövelésre:
az eszköz több adatot küld, amíg az adatot kérő eszköz le nem állítja (burst)

Sávszélesség függ a paraméterektől, pl.: 64 biten és 66 MHz-en 533 MB/s

PCI Express (PCIe)

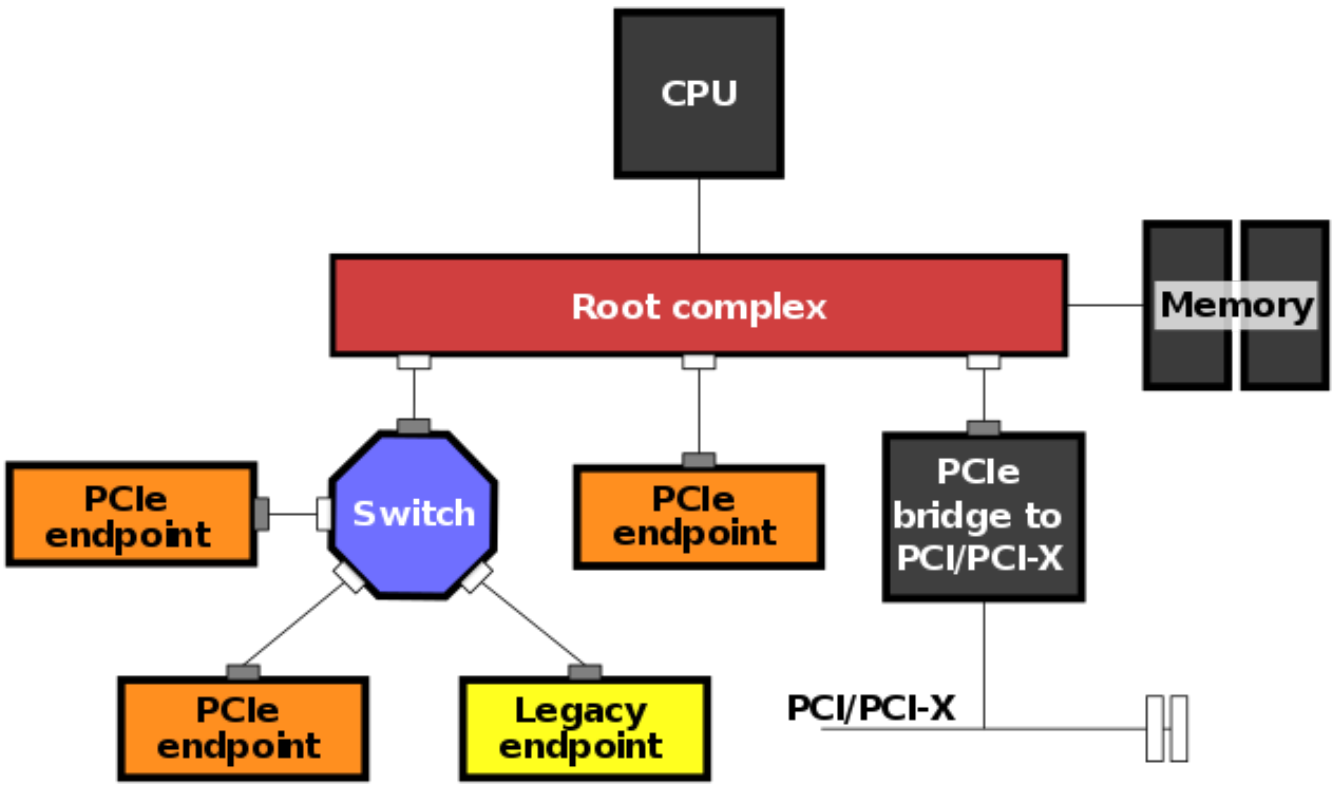
PCI utódja

- Soros busz
  Sávokból (lane-ekből) áll: $\times 1 \ldots \times 32$, melyeken párhuzamos a kommunikáció
  A sávok egymástól függetlenül tudnak adatot átvinni, közöttük bufferek biztosítják a szinkronizációt.
- Point-to-point topológiával
- Full-duplex kapcsolatra képes

Sávszélesség pl.: 1.969 GB/s (egy sáv, PCIe 4.0)

PCI Express (PCIe)

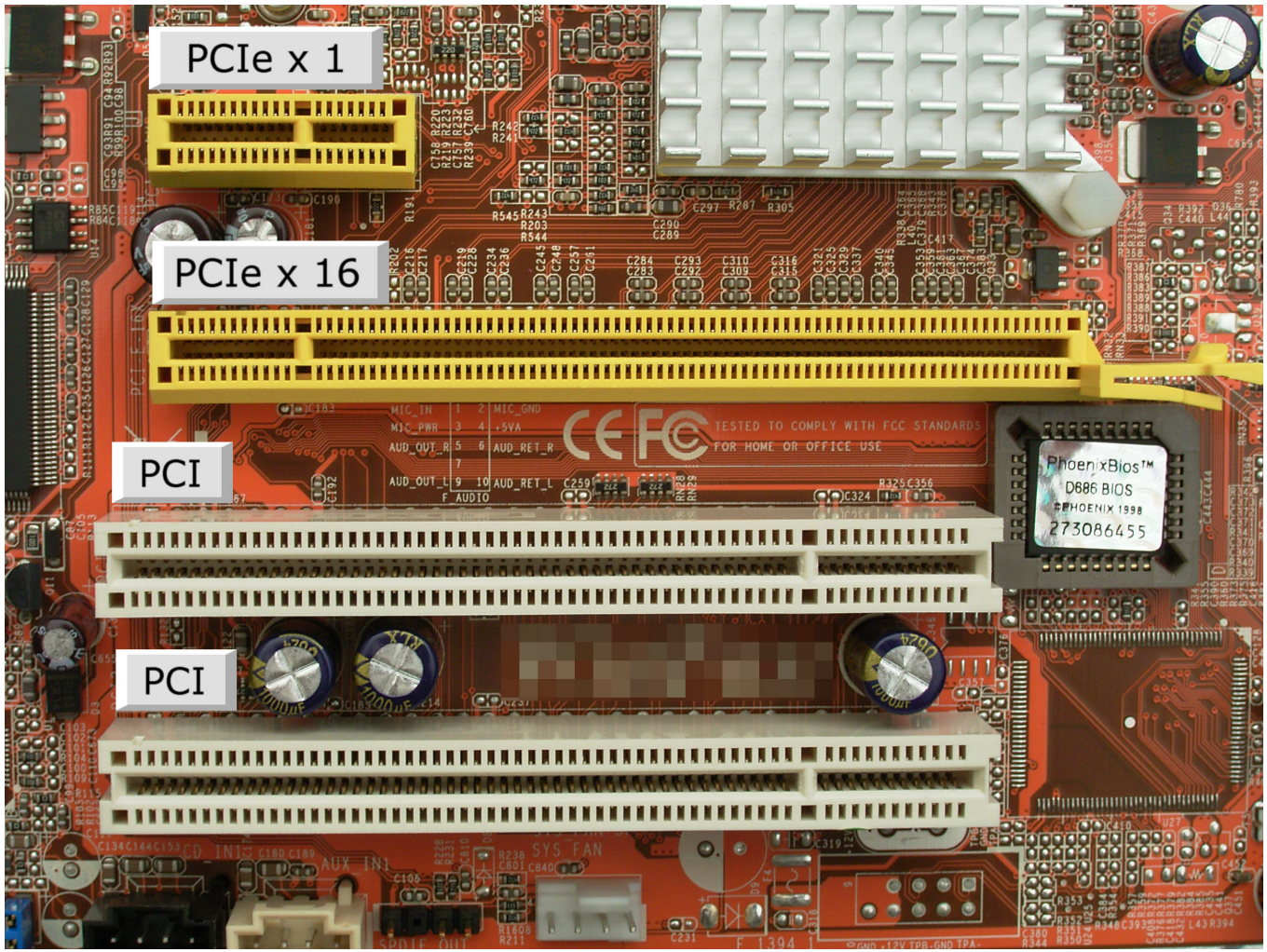
A PCIe blokkdiagrammja



Kép forrása: Wikipedia

PCI Express (PCIe)

Így néz ki az alaplapon



Kép forrása: Wikipedia

Small Computer System Interface (SCSI)

Cél: ellentétben a korábbi technológiákkal, minél több kommunikációs logika kerüljön az egyes eszközökbe (tehermentesítendő a CPU-t)
Pl.:
két eszköz tud egymás között adatokat cserélni,
több parancsot ki lehet adni egyszerre, nem kell megvárni az eredményt

A kommunikációs protokollt csatornától függetlenül több helyen is használják:
iSCSI (TCP/IP felett), Fiber Channel Protocol, Serial Attached SCSI
Sávszélessége nagyban függ ettől

Általában lassabb perifériák, pl.: merevlemezek összekötésére használják.

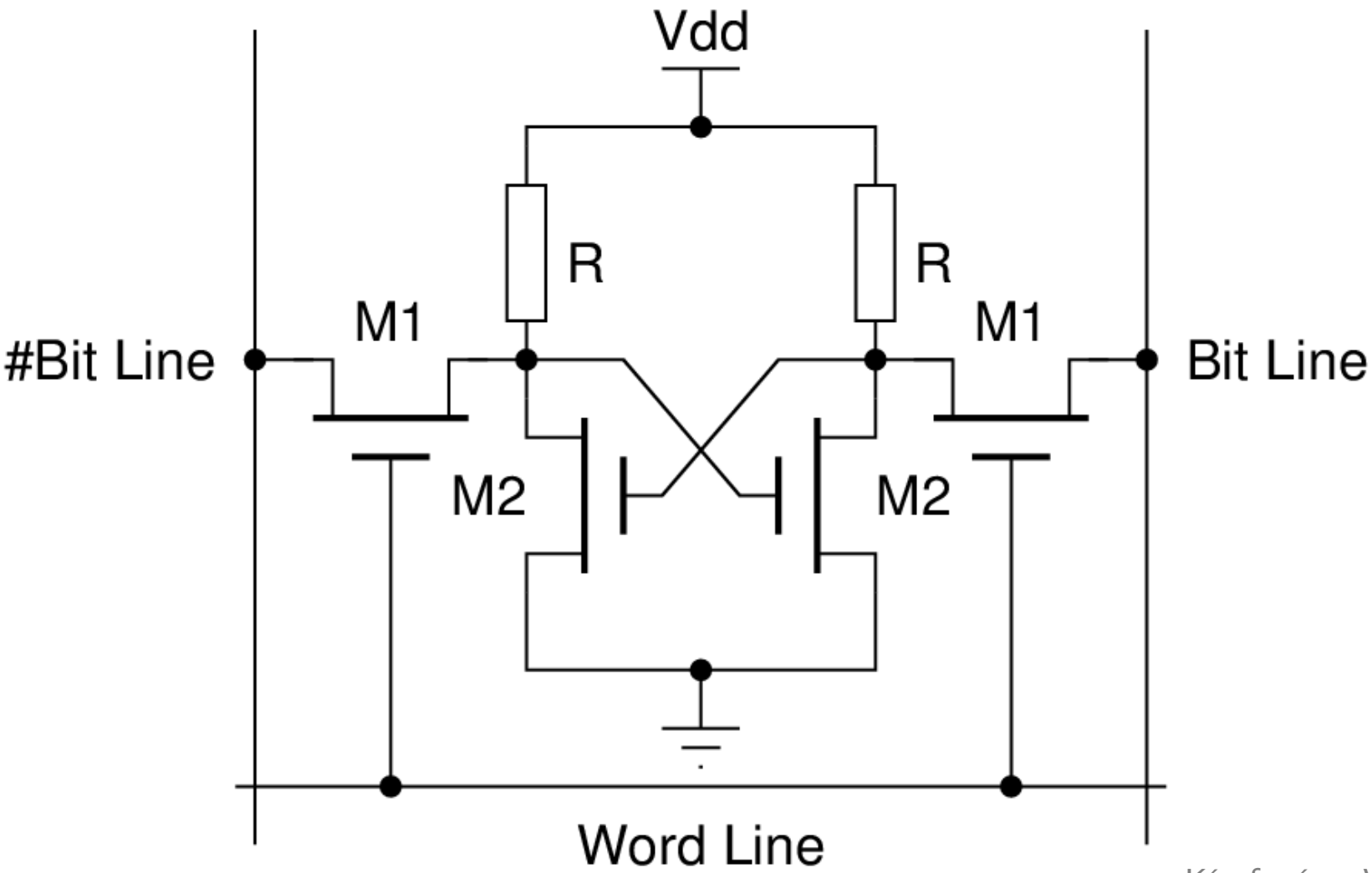
Memóriák

Különböző típusú memóriákat különböztetünk meg:

- *Read-Only Memory* (ROM): csak olvasható (vagy nehéz írni)
  pl. firmware-ket tárol
  - *Programmable ROM*: speciális eszközzel gyártás után programozható
    pl.: antifuse technológia
  - *Erasable PROM*: törölhető (pl.: UV fénnyel) és újraírható
  - *Eletronically Erasable PROM*
- *Random Access Memory* (RAM): bármely címet ugyanannyi idő elérni
  ellentétben pl. a merevlemezzel
  a legtöbb volatile memóriatípust is valamilyen RAM-nak hívják
- *volatile*: folyamatos energia kell a tároláshoz
  ellentétben a *nonvolatile* memóriákkal
- *Dynamic RAM* (DRAM): folyamatosan frissíteni kell a tárolt értéket
  ellentétben a *Static RAM*-al

Memóriák

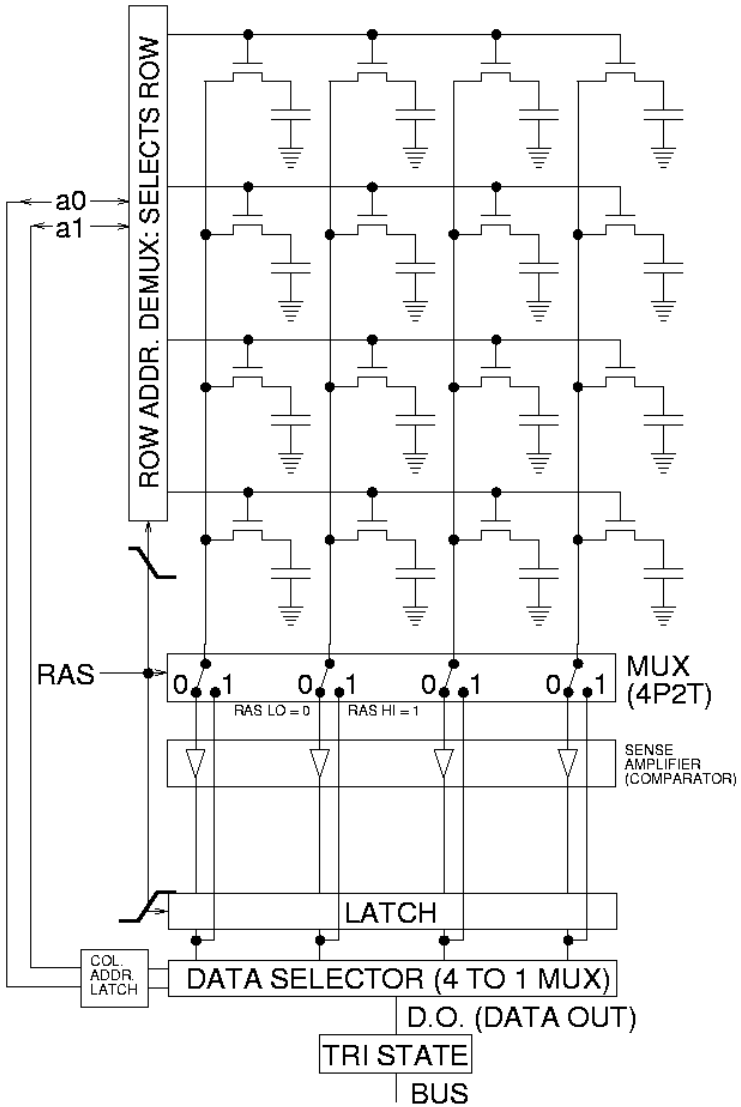
SRAM cella



Kép forrása: Wikipedia

Memóriák

DRAM címzési mechanizmusa



Kép forrása: Wikipedia

Synchronous DRAM (SDRAM)

A kommunikációt külső órajel szinkronizálja.

Egy cím elérésekor az egész sort ki kell olvasni,
így utána az adott sorban található többi cím kisebb késleltetéssel olvasható. Ezt ki lehet használni:

- egyszerre több adatot olvasva
- *prefetch buffer*: egyszere több adatot küld, kérés nélkül pl.: DDR4: egyszerre 8 szó
- *Double Data Rate SDRAM*: adat küldése emelkedő és ereszkedő órajelen is

Memória-hierarchia

Probléma: a gyors memóriák drágák és kicsik, a nagy memóriák nem férnek el közel a CPU-hoz

Megoldás ötletek:

- Alkalmazzunk sokféle memóriát
- A software válogasson közöttük (registerek, fő memória, merevlemez)
- Használjuk ki, hogy az elérések lokálisak (*locality of reference*): az egymást követő hívások
  - ugyanazt (temporal locality)
  - rákövetkező (spatial locality)

  címeket érnek el
  Ezek a *cache-hierarchiák*

Memória-hierarchia

Különböző technológiák, gyorsabbtól a lassabbig:

- Register: CPU chipen rajta, fordító dönti el általában, hogy mi kerül regiszterekbe
- L1, L2, . . . cache (gyorsítótár): egyre nagyobb és messzebb levő SDRAM memóriák
  a hardware dönti el, hogy mi kerül bele
- fő memória: általában valamilyen DRAM
  operációs rendszer ide cache-eli a háttértárakat
- Solid State Drive (SSD): gyors nonvolatile (perzisztens) tároló
- Hard Disk Drive (HDD): lassabb, mágneslemezes tároló

Little Man Computer

A CPU egy egyszerűsített modellje.
Egy pici embert bezártak egy szobába, melyben található:

- egy INBOX és egy OUTBOX,
- 100 sorszámozott fiók egy-egy háromjegyű számot tartalmazó cetlivel,
- egy számológép (összeadni és kivonni tud)
- és egy számláló (Program Counter).

A pici ember minden ciklusban:

- leolvassa a PC-ről a következő utasítás helyét;
- értelmezi a fiókban a cetlit;
- megnöveli a Program Countert;
- végrehajtja az utasítást.

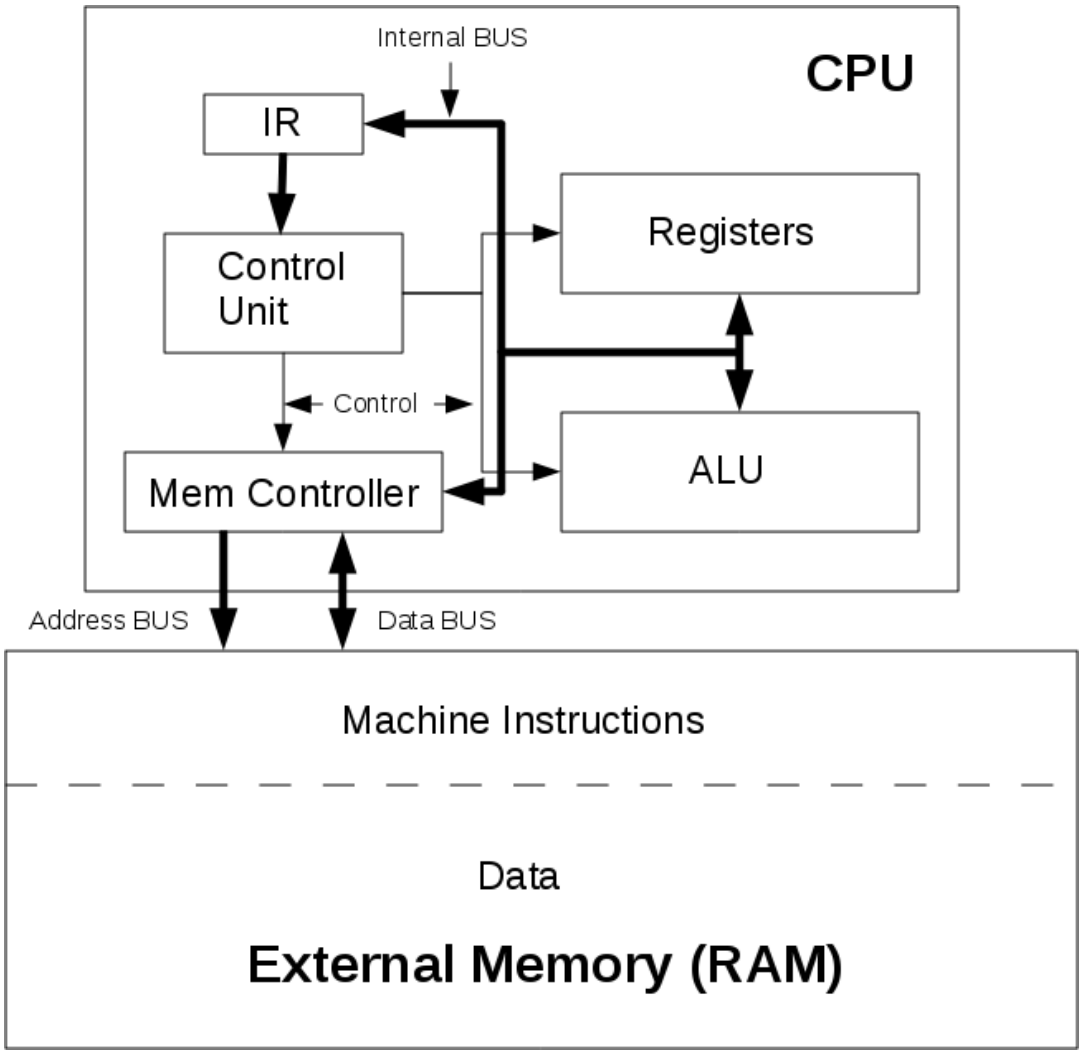
Little Man Computer

Példa

0	INPUT
1	STORE 99
2	INPUT
3	STORE 98
4	LOAD 99
5	BRANCH IF ZERO 12
6	SUBTRACT 15
7	STORE 99
8	LOAD 16
9	ADD 98
10	STORE 16
11	BRANCH 4
12	LOAD 16
13	OUTPUT
14	COFFEE BREAK
15	DAT 1
16	DAT 0

Vezérlő egység (Central Processing Unit)

Vázlatos blokkdiagramm:



Kép forrása: UWM

Fetch-Decode-Execute ciklus

Program Counter: regiszter, amely a következő utasítás címét tartalmazza
Instruction Register: az adott utasítást tartalmazza

- Fetch: a PC tartalmazta címről betölti a következő utasítást az IR-be
  megnöveli a PC-t eggyel
- Decode: értelmezi az utasítást
- Execute: végrehatja az utasítást

RISC – CISC

Kétféle utasításkészletet különböztetünk meg:

- Reduced Instruction Set (RISC):
  kevesebb, sokszor alkalmazás-specifikus utasítások
  az utasítások azonos hosszúságúak: egyszerűbb dekódolási és végrehajtási logika
  magasabb frekvenciával tud működni, nagyobb teljesítmény
  (lehet *pipeline* technikákat alkalmazni)
  pl.: Power ISA, ARM, RISC-V
- Complex Instruction Set (CISC):
  a programozási nyelvek megjelentével; a *semantic gap* csökkentésére:
  a magas szintű nyelveknek legyen gépi kódú megfelelőik
  változó hosszúságú utasítások
  általánosabb célra
  pl.: x86, amd64

További tanulmányozásra

- `https://www.youtube.com/watch?v=cNN_tTXABUA&pbjreload=10`
- Digitális rendszerek és számítógéparchitektúrák tárgy
- `https://godbolt.org`: fordítók kimenetének vizsgálgatására
- Little Man Computer Simulator: `https://peterhigginson.co.uk/LMC/`