Verzió: 2018.10.01. Történelmi áttekintés

- Mai számítógép: általános célú
- Régen: konkrét feladatok megoldására alkalmas
- Cél: gyorsabb munkavégzés, automatizálás

Az automatizálás kezdetei (ókor, középkor eleje)

- Algoritmusok
- digitus – digit
- Görögök: szenteltvíz-adagoló automata
- Kína, Mezopotámia

    a helyiérték és a tízes számrendszer társulása =>

    abacus (golyós számológépek, négy alapművelet)

    abacus kövei: calculusok
- Al-Khvarizmi (IX. század): algoritmus,

  algebra (al-dzebr=csontok helyreállítása), arab számok
- Fibonacci: Liber Abaci (1202)

A szellemi munka gépesítése (az újkor)

- **Simon Stevin** (XI. század vége): a logaritmus első használója

- **Jost Börgi** (1552-1632) svájci órásmester
  Első logaritmustáblázat (1603-1611), nyomtatásban 1620-ban.
  „Arithmetica” c. könyv (1592) – a tizedestörtek mai írásmódja
  első logarlécek

- **Wilhelm Sickard** (1623) thübingeni csillagász
  1 és 10 fogó fogaskerekek illeszkednek egymáshoz
  négy alapművelet

- **Blaise Pascal**

  1642 (19 éves) - 1644 Arithmometer – az első szériában gyártott számológép (7 példány)
  óraalkatrészekből mechanikus, fogaskerekes, forgatókarral működtetett szerkezet adószedő apja segítésére.
  +, -

- **Gottfried Wilhelm Leibniz (1671)**

Az ő szintén **mechanikus** gépe már közvetlenül szorozni és osztani is tudott, kivonást kiegészítő művelet nélkül.
Négyműveletes kalkulátor.

- **Matthieu Hahn** (1779), gépészeti érdeklődésű lelkész
  Fogazott dobok körkörös elrendezésben
  1820-ig sorozatgyártás

- **XIX. sz. eleje**
  Ipari termelés kialakulása, ipari megmunkálás fejlődése → számos tekerős gép

Úton az univerzális gép felé (XIX. század közepe – XX. század közepe)

- **Francia forradalom konventje**
  Megbízás: 19 jegyű logaritmustáblázatra
  14 jegyű trigononometrikus logaritmusra
  De Prony:
  5 matematikus – alapműveletek – szerkesztés
  8 „számoló” – programozás
  80 „rabszolga” - aritmetika

- **Charles Babbage (1791 – 1871)**

„A lyukkártya alkalmas elemeire bontott számítási utasítások gépbe táplálására.” (1830)

Differencia mozdony (difference engine, 1823-33):
Táblázatok kiszámítására (tengeri navigáció).
+, -.
Kimenő adatai rézbevonatú lemezbe lyukasztotta.

Gépek matematikai (szám)táblázatok kiszámításánál való alkalmazásának tapasztalatai – Királyi Csillagászati Társaság Aranyérme

pénzügyminiszter támogatása (1823–1842)

Egyetlen algoritmus: véges differenciák módszere polinomokra alk.

x	x^2+x+41	d1	d2
0	41		
1	43	2	
2	47	4	2
3	53	6	2
4	61	8	2
5	71	10	2
6	83	12	2

Weierstrass tétele

Dél-Kensigtoni Természettudományos Múzeum

- **Georg Scheutz**, svéd nyomdász

egyszerűsített differenciagép
táblázatok készítése, nyomtatása (negyedik differenciák)

- **Analitikus mozdony (Babbage, analytical engine)**

  Automatizált (digitális) számítógépek őse
  - technika miatt,
  - felismerte, hogy fontos a részeredmények tárolása,
  - algoritmusok alkalmazására törekedett.
  Programozható, programok, utasítások lyukkártyáról (kártyacsomag), pl. összeadás számokon, előtte tárolóból kiolvasni és az eredményt oda visszaküldeni

  Kudarc a gigantománia miatt (50 helyiérték, Scheutz: 8 helyiérték), az analitikus gép (nem készült el).

  Egységek:
  – memória (1000 db 50 jegyű szám),
  – malom (+, -, *, /),
  – input (lyukkártya),
  – output.

  Kártyacsomagok:
  - műveleti kártyák (végrehajtandó műveletek)
  - változókártyák (amelyeken a műveleteket)

  Ada Byron (Ada Augusta Lovelace, 1815-1852)
  A gép részletes leírása, példaprogramok (egyszerű assembly nyelven programozható az analitikus gép) -> szoftver.

  Működőképesség nem biztosítható ebben az időszakban: pontos fogak, kerekek és áttétek igénye ezerszámra.

- **Csebisev** (1878)
  olyan számológép, amely a szorzást nem összeadásra vezeti vissza

- **T.W. Odhner (1887)**

állítható fogazású számkerék (ma is gyártanak ilyen típusokat!)

- **Herman Hollerith (1889)**

Lyukkártyás vezérlés a népszámlálás adatainak rendezésére

- **XX. század**

  - mechanikus (elekromos hajtómű)

  - elektro-mechanikus (dugaszoló tábla programhoz)

  - jelfogók használata

    - **Kozma László** (négy alapművelet)

    - **Konrad Zuse** (1910-1995), 1936: elektromágneses relék használata, 1941: programvezérlés
      Z1 (1934-36): az első jelfogós számológép
      mechanikus, bináris gép (igen-nem kapcsolók), mikronyelv használata
      Z2 (1937-39) jelfogós, rögzített tizedespontú aritmetikai egység
      Z3 (1939-41) programvezérlésű gép (lyukszalaggal)
      tárolt program elve
      külön megépített +,-,*,/, gyökvonó
      lebegőpontos aritmetika, 64 szó tárolása
      beviteli egység: decimális klaviatúra
      kiviteli egység: lámpás megjelenítő

Z1-Z3: 2-es számrendszer

- **Howard H. Aiken (Harvard), T.J. Watson (IBM)**

  Babbage nyomán: jelfogós gép (Mark-I, 1943, ballisztiki számítások, később Mark-II)

  MARK-II: - központi vezérlés
  - összeadás 0,5 sec, szorzás 6 sec, osztás 15 sec,

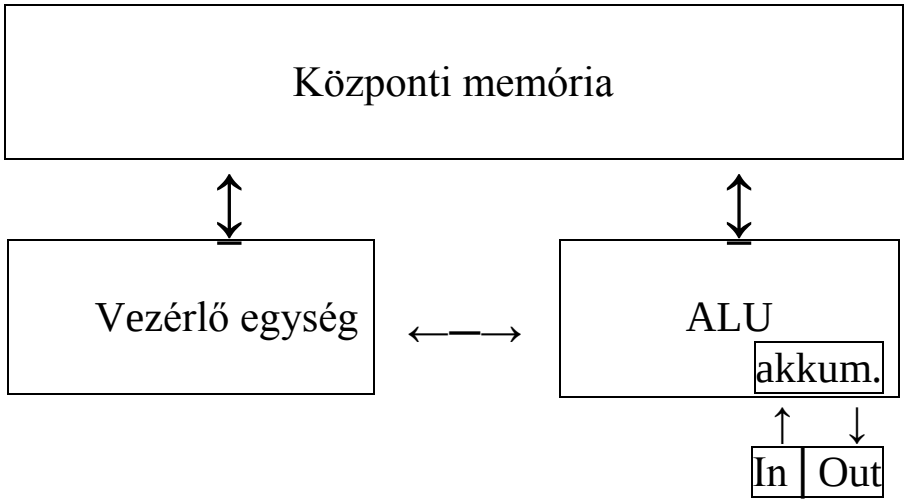
- **Norbert Wiener (1940)**
  Univerzális Elektronikus Digitális Számítógép
  - Aritmetikai egység numerikus
  - A kapcsolások elektroncsövek útján
  - A műveletek kettes számrendszerben
  - Automatikus végrehajtás (műveletsor, logikai döntés a gépben)
  - Adatok a gépben (törlés, hívás)

Elektronikus gépek

- **COLOSSUS (Turing, 1943):** titkosírások megfejtése, megsemmísítik, 30 évre titkosítva.
  Az első elektronikus számítógép.

- **ENIAC** (Electronic Numerical Integrator and Computer – J.P. Mauchley, J.W. Eckert, H.H. Goldstine, 1943):
  18000 elektroncső, csövek élettartama 2500 óra
  140 KWh energia, 30 tonna, 30 méteres terem,
  10-es számrendszer, 20 darab 10 decimális jegyes regiszter. 10 cső egy decimális számjegyhez!
  Dugaszolással programozható. Programtárolás nem volt!
  Output: nyomtató (kártyára lyukasztva)

Neumann János megismerte az **ENIAC**-ot, és új gépet tervezett (**IAS**): bináris aritmetika, tárolt program.



Neumann elvű gép:

Tárolt programú elektronikus számítógép, amely az adatok és a program tárolására közös memóriát használ. (Neumann-Goldstine jelentés, 1945).

Harvard típusú számítógép

Külön memóriát használ az adatok és külön memóriát a program számára.

EDSAC (**Wilkes**, Cambridge, 1949)

az első Neumann elvű működő gép
2-es számrendszer
4096 szavas memória. 40 bites szavak: előjeles egész, vagy két utasítás.
Tárolt program: 8 bites utasításkód, 12 bites cím.
Akkumulátor.
Nem volt lebegőpontos aritmetika! Első tárolt programú szgép.

EDVAC (1949 Eckert és Mauchley), de elvérzett a projekt. Később ebből a vállalkozásból lett az **UNISYS.**

Eckert és Mauchley sikertelenül próbálják találmánynak elfogadtatni.

A számítógép diadalútja (XX. század második fele)

Számítógép generációk:

1. Jelfogók, elektroncsövek (1955-ig, vagy más csoportosítás szerint 1958-ig): 5-10000 művelet/sec
2. 1955-65 (vagy más csoportosítás szerint (1958-63)

   Tranzisztorok (a tranzisztor feltalálása: 1948. Bell Laboratórium, John Bardeen, Walter Brattain és William Shockley, 1956: fizikai Nobel-díj)
3. Integrált áramkörök, 1965-80 (vagy 1963-71)

   Robert Noyce, 1958. több tucat tranzisztor egyetlen csipen

   1 millió műv./sec.
4. VLSI: nagyon széles skálájú integráció, 1980-tól (vagy 1972-től)

   10, 100, …, több millió tranzisztor egyetlen csipen

   (1971: az első mikroprocesszor egyetlen csipen)

1960 PDP-1 (DEC) első miniszámítógépek (50 eladott)

1963 számítógépes egér szabadalmaztatása

1963 B5000 (Burroughs) első magasszintű programozási nyelvet bizt. (Algol60)

1965 PDP-5 (DEC) miniszámítógép (50 000 eladott)

1968 Az Intel alapítása

1969 Arpanet, Amerikai Védelmi Minisztérium

1973 első komputeres játék (Pong)

1974 Microsoft alapítása

1975 Cray-1, az első szuperkomputer

1976 Apple megalakítása

1978 VAX (DEC) első 32 bites szuperszámítógép

1981 IBM személyi számítógép, MS-DOS op. rendszer

1982 Commodore 64

1982 Compaq első IBM-kompatibilis gépe

1983 Macintosh

1985 CD-ROM (Philips, SONY)

1985 MIPS (MIPS) az első RISC gép

1985 Windows

1987 SPARC (Sun) az első Sparc-alapú munkaállomás

1990 RS6000 (IBM) az első szuperskalár számítógép

1993 eső Pentium-chip

1997 IBM Deep Blue legyőzi Kaszparovot

1999 100 millió személyi számítógép vásárlása

Császármorzsa

Keverj össze 25 dkg grízt 1 mokkás kanál sóval, 4 evőkanál cukorral és egy csomag vaníliás cukorral!

Adj hozzá két evőkanál olajat és két tojást, jól dolgozd el!

Folyamatos keverés közben adj hozzá apránként fél liter tejet!

Adj hozzá egy bögre előre beáztatott mazsolát!

Süsd ki 3 evőkanál olajon!

Ez egy program.

De ki tudja végrehajtani?

A **digitális számítógép** olyan gép, amely a neki szóló utasítások alapján az emberek számára problémákat old meg.

Azt az utasítássorozatot, amely leírja, hogyan oldjunk meg egy feladatot, programnak nevezünk.

A legtöbb gépi utasítás ritkán bonyolultabb, mint

- Adj össze két számot!
- Ellenőrizz egy számot, vajon nulla-e!
- Egy adatot másolj a számítógép memóriájában egyik helyről a másikra!

Egy számítógép utasításainak együttese egy olyan nyelvet alkot, amelyen az ember a számítógéppel képes kommunikálni.

Az ilyen nyelvet gépi nyelvnek nevezzük.

Egyszerűbb gépi nyelv
↓
Egyszerűbb elektronika
↓
Olcsóbb gép
↓
Az ember számára nehézkes

Legyen $\mathbf{L_0}$ egy gépi nyelv, és $\mathbf{L_1}$ egy az ember számára kényelmesebb nyelv. Hogy hajtható végre az $\mathbf{L_1}$ nyelven írt program?

Kellene olyan gép, amelynek gépi nyelve az $\mathbf{L_1}$ nyelv.

Más megoldás: fordítás és értelmezés.

Fordítás: Először az $\mathbf{L_1}$ nyelvű program minden utasítását helyettesítjük az $\mathbf{L_0}$ nyelv utasításainak sorozatával. Az így nyert program teljes egészében az $\mathbf{L_0}$ utasításaibó áll.- Ezután az eredeti $\mathbf{L_1}$ nyelvű program helyett a számítógép ezt az $\mathbf{L_0}$ nyelvű programot hajtja végre.

Értelmezés: Az $\mathbf{L_1}$ nyelvű program következő utasítását elemezzük, és a vele ekvivalens $\mathbf{L_0}$ nyelvű utasítássorozatot azonnal végrehajtatjuk a számítógéppel.

A fordítás és az értelmezés is elvégezhető az $\mathbf{L_0}$ nyelvű számítógéppel. Olyan, mintha lenne olyan gépünk, amely végre tudja hajtani az $\mathbf{L_1}$ nyelven írt programot: **virtuális gép**.

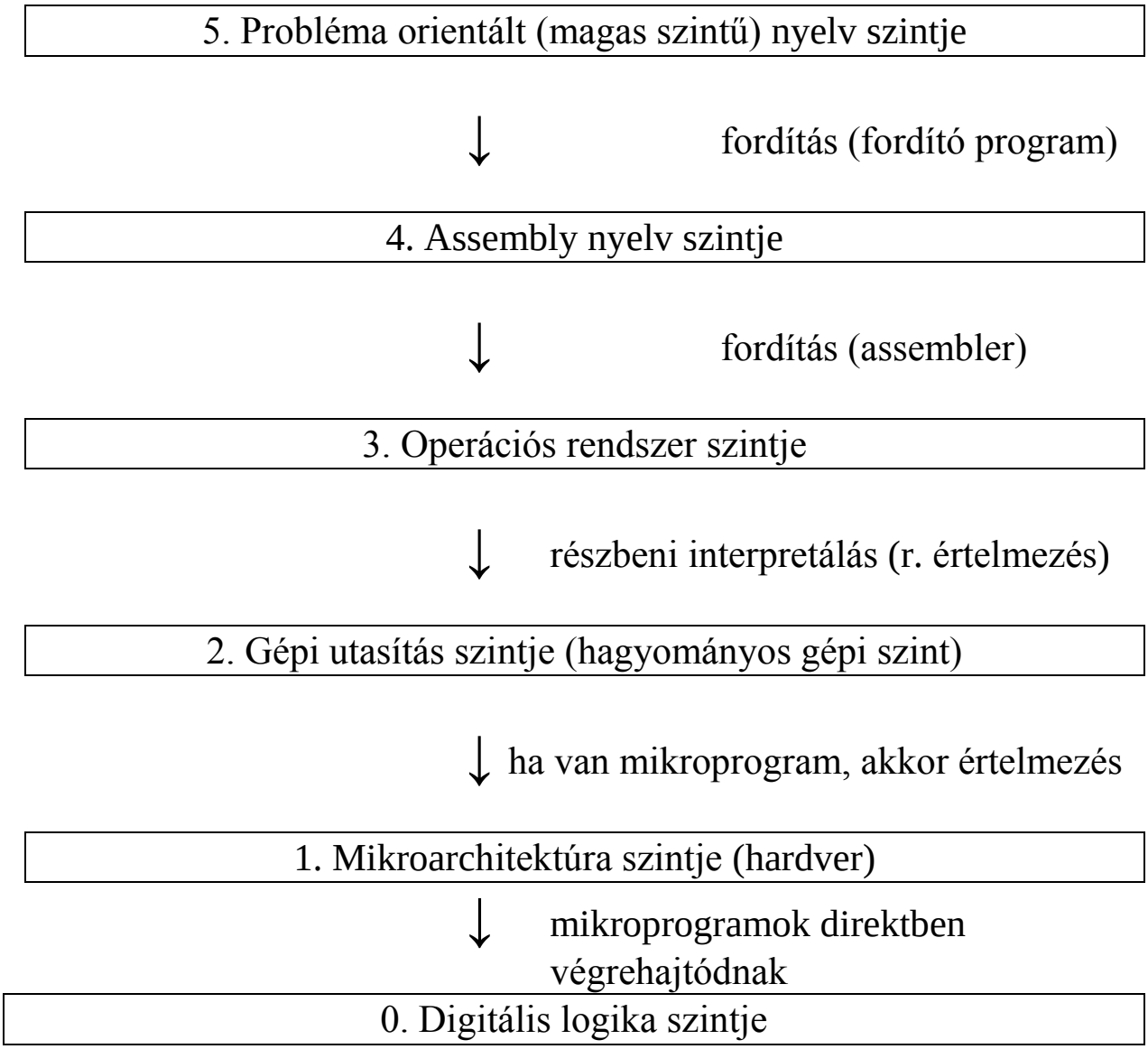
A gépi és az ember számára kényelmes nyelv között oly nagy az eltérés, hogy annak áthidalásához nyelvek és virtuális számítógépek hierarchiája alakult ki:

a strukturált számítógép-felépítés.

Alapgondolat: a gépet tekinthetjük úgy, mint egymásra épülő szintek rendszere (virtuális gépek), minden ilyenhez tartozik egy programozási nyelv.

Bonyolultabb nyelvek: fordítás vagy értelmezés.

Gépi, nyelvi szintek (1.2. ábra)



0: **digitális logika** szintje: logikai **kapu** (gate). ***AND***, ***OR***, **… műveletek.**

A kapuk bemenete(i): egy vagy több 0 vagy

Kimenet: ezen végzett művelet eredménye.

Néhány kapu összerakása:

→ 1 bit memória (0 vagy 1 tárolására)

→ több bites memória, regiszter (adattárolók,

16, 32, 64 bites csoportok)

1: **Mikroarchitektúra** szintje: mikroutasítások, mikroprogram szintje

nem minden gépen létezik, de a gépi utasítások végrehajtását

gyakran mikroprogram végzi, ekkor ez a szint **értelmezi** a 2.

szintet.

Egyetlen gépi kódú utasítás elemi vezérlési lépésekre bomlik.

Pl. a szorzást összeadásra és léptetésre vezeti vissza.

• Regiszterek, aritmetikai-logikai egység – **ALU**

• Adatfolyam - **adatút**

2: **Gépi utasítás** szintje (tényleges gépi utasítások)

utasításkészlet,

itt dől el a kompatibilitás kérdése.

3: **Operációs rendszer** szintje: speciális kiegészítők, az op.

rendszernek szóló utasítások (memóriakezelés, védelem, …)

Általában **értelmezés**. A szint utasításait

– az operációs rendszer,

– vagy közvetlenül a 2. szint hajtja végre.

Az eddigi szintek programjai hosszú számsorozatok

(természetesen ma már szimbolikusan készülnek)

--------- Eddig: rendszerprogramozók területe ---------

4: **Assembly nyelv** szintje, szimbolikus leírás

5: **Probléma orientált nyelv** szintje:

Pascal, C, C++, …, adatbázis kezelők, …

Ezek tényleges nyelvek, fordítás (fordító program = compiler)

Szintek között kapcsolat

A magasabb szinten írt program alacsonyabb szintre képezhető.

a) Fordítással (compiler): végrehajtás előtt az egész programot átalakítjuk az alacsonyabb szintre.

(Pl. …•pas →…•exe)

b) Értelmezéssel (interpretálás, **interpreter** végzi)., Utasításonként alakítja át és hajtja végre a programot (Pl. BASIC).

1, 2, 3 szint: általában interpretálás

4, 5 szint: általában fordítás

(Megjegyzés: a valódi végrehajtás mindig a legalsó szinten történik.)

Gépi utasítás szintje

Az utasítások a memóriában vannak tárolva.

addr) **command dest, source1, source2, next**

addr: az utasítás címe a memóriában

command (utasítás): az utasítás kódja

dest (cél): itt képződik az eredmény

source1 (forrás1): a művelet 1. operandusa

source2 (forrás2): a művelet 2. operandusa

next: a következő végrehajtandó utasítás címe. Ez legtöbbször az utasítás utáni első rekesz címe, ezért általában nem kell megadni (**implicit operandus)**, csak akkor, ha más utasítással folytatódik a program (ugró utasítás).

cím) **add dest, source1, source2**

hatására **dest** fölveszi a **source1** + **source2** értéket.

Ilyenkor természetesen elvész **dest** régi értéke.

További implicit operandusok:

Sokszor egyszerűsítik az utasításokat, pl.:

cím) **add op1, op2**

hatására **op1** fölveszi az **op1** + **op2** értéket. További

egyszerűsítés: cím) **add op**

hatására **A** fölveszi az **A** + **op** értéket, ahol **A** egy

kitüntetett regiszter (**accumulator**).

Történetileg a gépek kialakulása:

- Első gépek: mikroprogramozás szintje hiányzott,

  Kezdetben két szint:

  - utasítások: kevés és primitív gépi szintű utasítás (hagyományos gépi szint)

  - digitális logika (komplikált digitális szint, nem megbízható)

- **Mikroprogram** (hardver bővítése programozással),

  mikroprogramozás

  1951 – első mikroprogramok (Wilkes)

  Gyorsan elterjedt.

  Csúcs: hatvanas, hetvenes évek (általánossá vált);

  nagyon sok új utasítás (egyre komplikáltabb gépi utasítások,

  pl. *, / , …, ciklusszervezés, megszakítások)

  Később ezek az utasítások hardverrel is megvalósíthatókká

  váltak, és úgy gyorsabbak lettek.

  Folyamatosan változó határok

- Más irányzat: RISC (Reduction Instruction Set Compiler):

  visszaegyszerűsíteni a gépi utasításokat, rátenni a terhet a compiler-ekre

Operációs rendszerek

A hatvanas években készültek először.

Miért volt szükséges?

Példa: egy FORTRAN program futtatásának lépései

(miután sikerült hajnali 3 és 5 között a géphez jutni)

1. A „programkönyvtár egy fiókjából” előkeresni a FORTRAN-compilert (kártyacsomag), betenni a kártyaolvasóba, START gombot megnyomni.
2. A FORTRAN nyelvű felhasználói programját betenni a kártyaolvasóba, megnyomni a CONTINUE gombot.
3. Mikor a számítógép leáll, újraolvasni a FORTRAN felhasználói programot. Számos compiler többszöri újraolvasást igényelt.
4. A programozó ideges, mert a complier hibát talált, kijavítja és kezdi elölről. Ha a program helyes, akkor a complier kiírja a lefordított gépi kódot kártyákra.
5. A gépi kódú programot beteszi a szubrutin könyvtárcsomaggal a kártyaolvasóba és beolvassa őket.
6. A program elkezd futni. Általában nem működik, vagy épp váratlanul leáll. Ha szerencsés, akkor kitalálja és kijavítja a hibát és kezdi újra az 1. pontnál. Ha nem szerencsés, kinyomtatja a memóriát és hazamegy tanulmányozni.

Operációs rendszer: automatizálja az operációs munkát, mindig a gépben van.

Supervisor, rendszerhívások, kötegelt (batch) feldolgozás,
közvetlen telefonos összeköttetés (remote terminálok),
időosztás (time sharing).

Technológiai fejlődés

• **Moore törvény** (1965):

Az egy lapkán elhelyezhető elemek száma másfél évenként duplázódik. Azt várják, hogy 2020-ig teljesülni fog.

Minden más területen (lemezek, adatátvitel, ...) hasonló sebességű a fejlődés.

A szoftverek mérete, bonyolultsága is követi ezt:

• **Nathan első törvénye**:

A szoftver gáz: kitölti a rendelkezésére álló teret. A szoftver egy állandó igényt jelent a gyorsabb processzorokra, a nagyobb memóriára és több kapacitásra.
Például: a szövegszerkesztők az 1980-as években több tíz kB memóriát igényeltek, ma több tíz MB-ot, a jövőben lehet, hog több tíz GB-ot.

• A népszerűsítő irodalom kedvenc hasonlata szerint, ha az autóipar az utóbbi hetven évben úgy haladt volna, mint a számítástechnika, egy Rolls-Royce-t 20 $-ért lehetne kapni, motorja gyufafej nagyságú lenne, sebessége 100 000 km/h és egymillió kilométeren 3 liter benzint fogyasztana. (Vámos Tibor, 1981.)

A mai (2005) számítógéptípusok választéka

Típus	Ár (US$)	Felhasználható például
Nagyszámítógép	5 000 000	Időjárás előrejelzés, …
Munkaállomás-gyűjtemény (COW)	50 000-500 000	Tanszéki mini-szuperszámítógép
Szerver	5 000	Hálózati szerver
Személyi számítógépek	500	Asztali/hordozható
Játékok	50	Videójátékok
Mikrovezérlők	5	Órák, autók, eszközök
Eldobható	0.5	Eldobható: üdvözlőlapok belsejében lévó csipek, **RFID** (Radio Frequency IDentification)

Az Intel processzorai a Pentumiokig

Lapka	Dátum	MHz	Tranz.	Mem.	Megjegyzés
I-4004	1971/4	0.108	2300	640	Első egylapkás mikroproc
I-8008	1972/4	0.108	3500	16 KB	Első 8 bites mikroproc.
I-8080	1974/4	2	6000	64 KB	Első általános célú mikroproc
I-8086	1978/6	5-10	29000	1 MB	Első 16 bites mikroproc.
I-8088	1979/6	5-8	29000	1 MB	Az IBM PC processzora
I-80286	1982/6	8-12	134000	16 MB	Memória védelem
I-80386	1985/10	16-33	275000	4 GB	Első 32 bites mikroproc.
I-80486	1989/4	25-100	1.2M	4 GB	8 KB beépített gyorsítótár
Pentium	1993/5	60-233	3.1M	4 GB	Két csővezeték, MMX
Pentium Pro	1995/3	150-200	5.5M	4 GB	Két szintű beépített gyorsítótár
Pentium II	1997/5	233-400	7.5M	4 GB	Pentium Pro + MMX utasításkészlet
P. III	1999/2	650-1400	9.5M	4 GB	SSE utasítások 3D grafikához
P.4	2000/11	1300-3800	42 M	4 GB	Hyperthreading + több SSE

MMX (*Pentium with MMX Trechnology*), 1997

- SIMD utasításkészlet (Single Instruction, Multiple Data)
- 8 darab 64 bites MMX regisztert használ

SSE (*Streaming SIMD Extensions*)

- 70 új utasítás, többségük egyszeres pontosságú lebegőpontos (ALU),
- XMM regiszterek (8 darab 128 bit széles)
- de cache (L1/L2 gyorsítótár-kezelési) utasítások is

SSE2 (*Streaming SIMD Extensions 2*)

- A Pentium 4 első verziójához (2004).
- Kiterjeszti a korábbi SSE utasításkészletét, 144 új utasítást ad hozzá.
- Dupla pontosságú lebegőpontos adatokon

Pentium III (1999): 1 GFLOPS (!)

Pentium 4 – NETBURST architektúra

2006-2008 Dual Core

2008-2012 Asztali gépek Laptop (hány magos)

Core i3	**2**	**2**
Core i5	**2-4**	**2**
Core i7	**4-6**	**2-4**

Fixpontos számábrázolás

Pl.: előjeles kétjegyű decimális számok:

- Ábrázolási tartomány: [-99, +99].
- Pontosság (két „szomszédos” szám különbsége): 1.
- Maximális hiba: (az ábrázolási tartományba eső) tetszőleges valós szám és a hozzá legközelebb lévő ábrázolható szám különbsége: 1/2.

Számolási pontatlanságok:

a = 70, b = 40, c = - 30 esetén

a + (b + c) = 80, (a+b) + c = -20.

Túlcsordulás

Helyértékes ábrázolás

Pl.: $521{,}2510 = 5 * 10^2 + 2 * 10^1 + 1 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$.

Általában (q alapú számrendszer esetén):

$$a_n\, a_{n-1} \ldots a_0, b_1 b_2 \ldots b_m =$$

$$= a_n * q^n + a_{n-1} * q^{n-1} + \ldots + a_0 + b_1 * q^{-1} + b_2 * q^{-2} + \ldots + b_m * q^{-m}$$

$$0 \leq a_i, b_j < q$$

Átszámolás számrendszerek között

B: Bináris, O: Oktális, D: Decimális H: Hexadecimális

B	O	D	H	B	O	D	H
0	0	0	0	1000	10	8	8
1	1	1	1	1001	11	9	9
10	2	2	2	1010	12	10	A
11	3	3	3	1011	13	11	B
100	4	4	4	1100	14	12	C
101	5	5	5	1101	15	13	D
110	6	6	6	1110	16	14	E
111	7	7	7	1111	17	15	F

Pl. 23,37510 átszámítása kettes számrendszerbe.

Egész rész osztással: Tört rész szorzással:

/2	marad	egész	tört *2	törtrésze
			.375	
23	1	0	.75	
1	1	1	.5	
5	1	1	.0	
2	0			
1	1			

10111_2 $0{,}011_2$

$$23{,}375_{10} = 10111{,}\ 011_2.$$

Véges tizedes tört nem biztos, hogy binárisan is véges!

Példa bináris összeadásra:

1. összeadandó: 0 1 0 1 1 0 1 0 (= 90_{10})

2. összeadandó: 0 1 1 1 1 1 0 0 (=124_{10})

Átvitel: 1 1 1 1 0 0 0

Eredmény: 1 1 0 1 0 1 1 0 (=214_{10})

Szorzás: bináris számok aritmetikájának megfelelően; eltolás+összeadás

```
          1 1 0 0 1  (= 25₁₀)
            1 0 1 1  (= 11₁₀)
-------------------
          1 1 0 0 1
        1 1 0 0 1
      1 1 0 0 1
-------------------
  1 0 0 0 1 0 0 1 1  (=275₁₀)
```

Osztás:

```
1 0 0 1 : 100 = 10,01
1 0 0
    0 1 0 0
      1 0 0
      0 0 0
```

Átszámítás 10-es számrendszerbe

q alapú számrendszerből legegyszerűbben a Horner elrendezéssel alakíthatunk át számokat:

$a_n*q^n + a_{n-1}*q_{n-1} + \ldots + a_0 + b_1*q^{-1} + b_2*q^{-2} + \ldots + b_m*q_{-m} =$
$(\ldots(a_n*q + a_{n-1}) * q + \ldots + a_1)* q + a_0 +$
$+(\ldots(b_m/q + b_{m-1})/q + \ldots + b_1)/q$

A számítógép kettes számrendszerben dolgozik, 10-es számrendszerből a Horner elrendezéssel alakítja át a számokat.
A formulában a_i -t, b_j -t és q –t kettes számrendszerben kell megadni.

Kettes számrendszerből 10-es számrendszerbe 10-zel való ismételt osztással állítja elő az egész részt, és 10-zel való ismételt szorzással állítja elő a tört részt – hasonlóan ahhoz, ahogy korábban bemutattuk a 10-es számrendszerből 2-esbe való átszámítást.

Bit: egy bináris számjegy, vagy olyan áramkör, amely egy bináris számjegy ábrázolására alkalmas.

Bájt (Byte): 8 bites egység, 8 bites szám.

Fixpontos számábrázolás

Előjeles fixpontos számok:

$2^8 = 256$ különböző 8 bites szám lehetséges.

Melyek jelentsenek negatív értékeket?

Előjeles számok szokásos ábrázolásai:

- előjeles abszolút érték,
- egyes komplemens,
- kettes komplemens,
- többletes.

Előjeles abszolút érték (egyenes kód):

előjel és abszolút érték,

az első bit (balról) az előjel: 0: +, 1: -

Pl.: $+25_{10} = 00011001_2$, $-25_{10} = 10011001_2$.
+ -

Jellemzők (8 bites szám esetén):

- a legkisebb szám -127, a legnagyobb 127,
- a nulla kétféleképpen ábrázolható.

Egyes komplemens kód (inverz kód):

Az első bit az előjel, 0: pozitív, 1: negatív. (Azaz negatív számnál az előjel helyiértékén 1-es.)

Egy szám -1-szerese (negáltja) úgy kapható meg, hogy a szám minden bitjét negáljuk (ellenkezőjére változtatjuk). (számjegyek megcserélése (0↔1)

Pl.: $+25_{10} = 00011001_2$,
$- 25_{10} = 11100110_2$.

Jellemzők (8 bites szám esetén):

- a legkisebb szám -127, a legnagyobb 127,
- a nulla kétféleképpen ábrázolható.

Kettes komplemens:

Az első bit az előjel, 0: pozitív, 1: negatív. (negatív szám: előjel =1).

Egy pozitív szám negáltja úgy kapható meg, hogy az egyes komplemenshez egyet hozzáadunk. (Azaz helyiértékenként felcseréljük a bitket és a legalacsonyabb helyiértékhez + 1).

Pl.: $+25_{10} = 00011001_2$,
$- 25_{10} = 11100110_2$ egyes komplemens,
$- 25_{10} = 11100111_2$ kettes komplemens.

Jellemzők (8 bites szám esetén):

- a legkisebb szám -128, a legnagyobb 127,
- a nulla egyértelműen ábrázolható.

Többletes: a szám és a többlet összegét ábrázoljuk előjel nélkül (ez már pozitív!). **m** bites szám esetén a többlet általában $\mathbf{2^{m-1}}$ vagy $\mathbf{2^{m-1} - 1}$.

Pl.: $+25_{10} = 10011001_2$, 128-többletes ábrázolás

$- 25_{10} = 01100111_2$, 128-25=103

Jellemzők (128 többlet esetén):
• a legkisebb szám -128, a legnagyobb 127,
• a nulla egyértelműen ábrázolható.

Megjegyzés: a **2m-1** többletes ábrázolás azonos a kettes komplemenssel fordított előjellel.

Használata: a lebegőpontos számok kitevő-részénél.

Lebegőpontos számok:

előjel karakterisztika törtrész

Sok ekvivalens ábrázolási lehetőség, a leggyakrabban a

törtrész első számjegye az adott szám első nullától

különböző számjegye (normalizált alak).

Példa: $254 = 0{,}0254 \times 10^4 = \mathbf{0{,}254 \times 10^3}$ (= $\mathbf{2{,}54 \times 10^2}$).

Megjegyzések:

• A nulla ábrázolásához külön megállapodásra van szükség (általában csupa nulla számjegyből áll).

• A lebegőpontos ábrázolásoknál is meghatározható a legkisebb és a legnagyobb ábrázolható szám, továbbá a legkisebb és legnagyobb hiba.

Lebegőpontos összeadás (kivonás)

- karakterisztikák egyeztetése
- mantisszákat összeadjuk
- karakterisztika = közös karakterisztika

Pld. X = 1 0 1 , 1 0 1 1

Bináris alakjai:

0 , 1 0 1 1 0 1 1 · 2 11
0 , 0 1 0 1 1 0 1 1 · 2 100
0 , 0 0 1 0 1 1 0 1 1 · 2 101 stb.

X = 1 0 1 1 0 1 1 1 1 5 11/16 = 91/16
Y = 1 1 1 0 1 0 0 1 0 3 5/8 = 58/16
149/16

0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0

Közös karakterisztika

0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1

Összeg:

1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

0	0	0	0	0	0	1	1

Túlcsordulás!

0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0

0	0	0	0	0	1	0	0

↑

9 5/16

Kivonás: mantisszák komplemens kódjainak összeadása.
Szorzás: szorzat előjelének előállítása, karakterisztikákat összeadjuk, mantisszákat összeszorozzuk, a végeredményt normalizáljuk.

BCD ábrázolás (Binary Coded Decimal representation): minden decimális számjegyet négy biten ábrázolunk.

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Negatív számok BCD ábrázolása: 9 vagy 10 komplemens kóddal.
Pl.: $+301_{10}$ = 0000 0011 0000 0001,
-301_{10} = 1001 0110 1001 1000 (9 komplemens),
-301_{10} = 1001 0110 1001 1001 (10 komplemens).

Digitális logikai szint

Digitális áramkör: két érték – általában

0-1 Volt között az egyik (pl. 0, hamis),
2-5 Volt között a másik (1, igaz).
Más feszültségeket nem engednek meg.

Kapu (gate): kétértékű jelek valamilyen függvényét tudja meghatározni.

Kapcsolási idő néhány ns (nanoszekundum = 10^{-9} s)

Műveletek logikai megvalósítása

Logikai algebra alapelveit **George Boole** dolgozta ki.

Boole-algebra: ítéletekkel végzett műveletek összessége.
(Ítéletkalkulus.)
Egy ítélet lehet igaz (1) vagy hamis (0).
Olyan algebra, amelynek változói és függvényei csak a 0, 1 értéket veszik fel,

Alapműveletek:

- ***ÉS*** (konjunkció, szorzás),
- ***VAGY*** (diszjunkció, összeadás),
- ***NEM*** (negáció, tagadás).

Igazságtábla: olyan táblázat, amely a változók összes lehetséges értéke mellett megadja a függvény vagy kifejezés értékét.

Definíció: Akkor mondjuk, hogy két Boole-függvény **ekvivalens**, ha az összes lehetséges bemenetre a két függvény azonos kimenetet ad.

Két Boole-függvény ekvivalenciája könnyen ellenőrizhető az igazság táblájuk alapján.

NAND (= NEM-ÉS) **és** ***NOR (= NEM-VAGY)*** **előnye:** teljesség (Sheffer-féle művelet, Peirce-féle művelet.) Ezeket szokás univerzális logikai műveleteknek is nevezni.

NEM(A) = NAND(A,A) = NOR(A,A)

ÉS(A,B) = AB = NAND(NAND(A,B))

VAGY(A,B) = A+B = NOR(NOR(A,B))

Alapvető digitális logikai áramkörök

Integrált áramkör (**IC**, Integrated Circuit, chip, lapka) 5 x 5 mm^2 szilícium darab kerámia vagy műanyag lapon (tokban), lábakkal (pins).
Négy alaptípus:

- **SSI** (Small Scale Integrated 1-10 kapu),
- **MSI** (Medium Scale ..., 10-100 kapu),
- **LSI** (Large Scale..., 100-100 000 kapu),
- **VLSI** (Very Large Scale ..., > 100 000 kapu).

Memória szervezése

Elvárás: szavak címezhetősége (olvasás, írás választása)

CPU feladata: a memóriában tárolt program végrehajtása.

• vezérlőegység, feladata:
– a program utasításainak beolvasása,
– az **ALU**, a regiszterek vezérlése,

• aritmetikai-logikai egység (**ALU**),
feladata: az utasítások végrehajtása,

• regiszter készlet, feladata:részeredmények, vezérlő információk tárolása. A legfontosabb regiszterek:

– utasításszámláló (Program Counter): **PC.** A CPU ezen regisztere tartalmazza az operatív tárból lehívandó soron következő utasítás címét.
– utasításregiszter (Instruction Register): **IR.**

CPU (Central Processing Unit) feladatai (1 Neumann-ciklus)

• a végrehajtandó utasítás betöltése,

• a betöltött utasítás típusának megállapítása,

• az ezt követő utasítás címének megállapítása,

• ha kell, az operandus(ok) helyének megállapítása,

• ha kell, az operandus(ok) betöltése,

• az utasítás végrehajtása,

• ha kell, az eredmény helyének megállapítása,

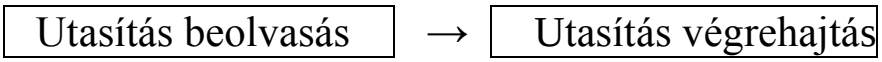
• ha kell, az eredmény tárolása,

• az egész ciklus újra kezdése.

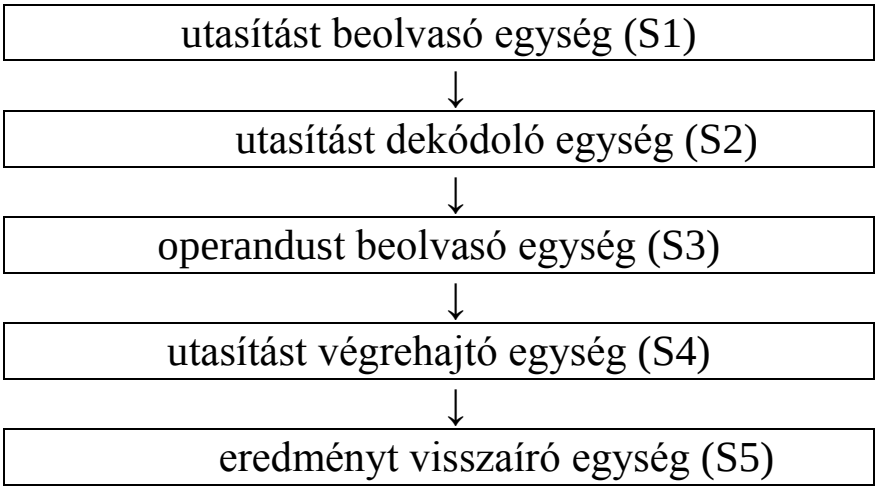
Párhuzamosítás: utasítás vagy processzor szintű.

Utasítás szintű: szállítószalag, csővezeték (pipelining).

Kezdetben:



Minden fázist külön hardver hajt végre ezek párhuzamosan működhetnek (szerelő csarnok).



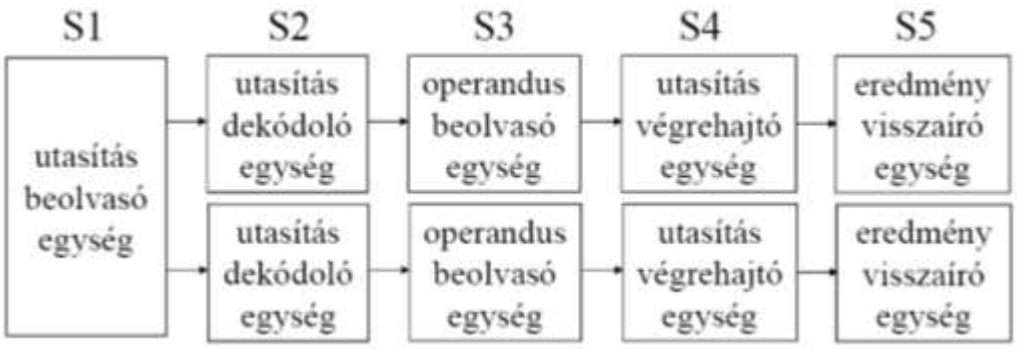
	A végrehajtás alatt lévő utasítás sorszáma									
S1:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**
S2:		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**
S3:			**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**
S4:				**1**	**2**	**3**	**4**	**5**	**6**	**…**
S5:					**1**	**2**	**3**	**4**	**5**	**…**
Idő:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**

- **Késleltetés** (latency): mennyi ideig tart egy utasítás.

- **Áteresztőképesség** (processor bandwidth): hány **MIPS** (Million Instruction Per Second) a sebesség.


utasítást beolvasó egység (S1)
utasítást dekódoló egység (S2)
operandust beolvasó egység (S3)
utasítást végrehajtó egység (S4)
eredményt visszaíró egység (S5)
utasítást dekódoló egység (S2)
operandust beolvasó egység (S3)
utasítást végrehajtó egység (S4)
eredményt visszaíró egység (S5)

Több szállítószalagos CPU



Két szállítószalag (2.5. ábra):

• Két végrehajtó egység, de közös regiszterek,
• A két szállítószalag lehet különböző is (**Pentium**):

Fő – ez többet tud, elsőbbséget élvez – és mellék.

Bonyolult szabályok a párhuzamos végrehajthatóságra
(fordítók vagy hardver).

RAM (Random Access Memory)

• **Statikus RAM** (**SRAM**)
Amíg áram alatt van, tartja a tartalmát.
Elérési idő: néhány nsec (cache-nek jók).

• **Dinamikus RAM** (**DRAM**): minden bit egy tranzisztor és egy kondenzátor: néhány msec-onként frissíteni kell, de nagyobb adatsűrűség érhető el. Elérési idő: néhány tíz nsec (főmemóriák).

• **SDRAM** (Synchronous **DRAM**). A központi óra vezérli. Blokkos átvitel.

Újabban: **DDR** (Double Data Rate). Az órajel föl- és lefutó élénél is van adatátvitel.

ROM (Read-Only Memory)

ROM: gyárilag kialakított tartalom.

PROM (Programmable **ROM**)**:** a tartalom biztosítékok kiégetésével alakul ki**.**

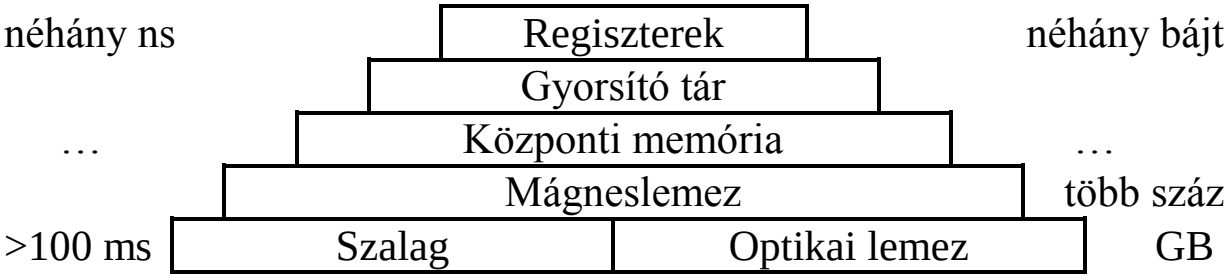
EPROM (**Erasable PROM**): a biztosítékok speciális fénnyel kiolvaszthatók és „kijavíthatók”.

EEPROM: elektromos impulzusokkal**.**

Flash memória: az EEPROM memória speciális típusa.
Törlés és újraírás csak blokkonként. Kb. 100 000 használat után „elkopnak”. Ilyen van a legtöbb MP3 lejátszóban, digitális fényképezőgépben. A **pendrive** (USB-flash-tároló, USB-kulcs, tollmeghajtó) egy USB-csatlakozóval egybeépített flash memória**.** A pendrive tehát egy *elektronikus* típusú tárolóeszköz (amely félvezető memóriát alkalmaz az adatok tárolására).

Memória hierarchia

Elérési idő **Kapacitás**



A diszkeket (mágneslemezek) szokás másodlagos háttértáraknak is nevezni.

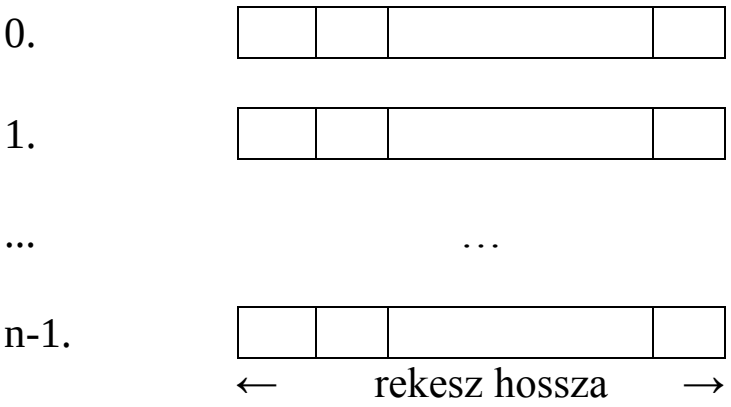
A szalagokat, CD-ket pedig harmadlagos háttértáraknak.

Központi memória (**2.9. ábra**)

A programok és adatok tárolására szolgál.

Bit: a memória alapegysége, egy 0-t vagy 1-et tartalmazhat.

Memória rekesz (cella): több bit együttese. Minden rekesz ugyanannyi bitből áll. Minden rekeszhez hozzá van rendelve egy szám, a **rekesz címe**. Egy rekeszre a címével hivatkozhatunk. A rekesz a legkisebb címezhető egység.



n: a rekeszek száma

A rekesz hossza manapság legtöbbször 8 bit (1 bájt).

Ezek a **bájtszervezésű gépek** (pl. PC).

Vannak **szószervezésű** (szó=word) gépek is, itt egy rekesz 8,12,16,… bit (DEC PDP-8, IBM1130, DEC PDP-15,…).

Bájtsorrend

A legtöbb processzor több egymás utáni bájttal (=szóval) is tud dolgozni.

A legmagasabb helyértékű bájt a szóban a

legalacsonyabb címen:	legmagasabb címen:
nagy (big) endian	**kis (little) endian**
MSBfirst (SPARC)	**LSBfirst (Pentium)**

Most/**L**east **S**ignificant **B**yte **first**

Ha egy 32 bites szó bájtjainak értéke rendre: *a, b, c, d*, akkor a szó értéke:

$a*256^3+b*256^2+c*256+d$ $a+b*256+c*256^2+d*256^3$

Virtuális memória és kezelése, lapozás: részletesen lásd Operációs rendszerek előadás. Egy gép virtuális memóriája a memória kiterjesztése háttértáron.
Ha gépünkben bővítjük az operatív tárat, akkor ugyanaz a programhalmaz gyorsabban fog futni, mert kevesebb lapcserét kell végeznie a virtuális memóriakezelőnek.

Gyorsító tár

A processzorok mindig gyorsabbak a memóriáknál.

A **CPU** lapkára integrálható memória gyors, de kicsi.

Feloldási lehetőség: a központi memória egy kis részét (gyorsító tár) a **CPU** lapkára helyezni: Amikor egy utasításnak adatra van szüksége, akkor először itt keresi, ha nincs itt, akkor a központi memóriában.

Lokalitási elv: Ha egy hivatkozás a memória ***A*** címére történik, akkor a következő valószínűleg valahol ***A*** közelében lesz (ciklus, mátrix manipulálás, …).

Ha ***A*** nincs a gyorsító tárban, akkor az ***A***-t tartalmazó (adott méretű) blokk (gyorsító sor - cache line) kerül beolvasásra a memóriából a gyorsító tárba.

Találati arány (***h***): az összes hivatkozás mekkora hányada szolgálható ki a gyorsító tárból.

Hiba arány: *1-h*.
Ha a gyorsító tár elérési ideje: ***c***, a memória elérési ideje: ***m***, akkor az **átlagos elérési idő** = ***c + (1- h) m***.

A gyorsító tár mérete: nagyobb tár – drágább.

A gyorsító sor mérete: nagyobb sor, a hivatkozott cím nagyobb környezete lesz a gyorsító tárban –nagyobb a sor betöltési ideje is. Ugyanakkora tárban kevesebb gyorsító sor fér el.

Osztott (külön utasítás és adat) **gyorsító tár előnyei:**

- Egyik szállítószalag végzi az utasítás, másik az operandus előolvasást.

- Az utasítás gyorsító tárat sohasem kell visszaírni (az utasítások nem módosulnak).

Egyesített gyorsító tár: nem lehetséges párhuzamosítás.

Hierarchia:

- elsődleges, a **CPU** lapkán,
- másodlagos, a **CPU**-val egy tokban,
- külön tokban.

LRU (Least Recently Used) algoritmus:

gyorsító sor betöltése előtt a legrégebben használt bejegyzés kerül ki a gyorsító tárból.

ASCII (American Standard Code for Information Interchanges), **alap**: 7 bites: vezérlőkarakterek, az angol ábécé kis és nagy betűi, szimbólumok.

Kiterjesztett (extended): 8 bites. Latin-1 kód: 8 bites. **IS 8859**: kódlap, **IS 8859-2:** magyar betűk is (ő és ű is).

UNICODE (**IS 10646**), 16 bites: kódpozíciók (code point). Általában egy nyelv jelei egymás után vannak – a rendezés könnyű.

- Kínai, japán, koreai: fonetikus szimbólumok, Han ideogramok (20992 jel, nincsenek szótár szerint rendezve). ... Japán íráshoz kevés (> 50000 kanji jel van).

- Új jelek? Braille nincs benne.

UTF-8: változó hosszú karakterkészlet (Universal Transformation Format)

- A 0-val kezdődő, 7 bittel folytatódó jelek a standard 7 bites ASCII karakterkészlet elemei
- 11-gyel kezdődőek több 8 bites szekvencia kezdetét, ahány 1-es van annyi bájtból áll összesen a szekvencia, majd egy 0-s, majd maradék bitek
- 10-val kezdődőek a folytatás, majd 6 bites maradékok.
- A maradékok összeolvasva az adott karakter Unicode kódolása.

Pl. 110yyyyy 10110011 vagy 1110zzzz 10yyyyyy 10xxxxxx

Pl. ó betú Unicode kódja 243,
00000000 00000000 00000000 11110011
UTF-8 kódja bináris 11000011 10110011

Mágneslemezes tárolók, diszkek

Céljuk: másodlagos tárolás (fájl-rendszer, virtuális memória).

Mágnesezettség változáson alapulnak: nem felejtenek kikapcsolva.

A mágneses jelrögzítés két fizikai törvénye
– változó áram mágneses mezőt hoz létre, ez mágnesezhető anyag mágnesezettségét megváltoztathatja (jelrögzítés);
– változó mágneses térben vezetőben áram indukálódik (kiolvasás alapja).

Lemezoldalak - író/olvasó (I/O) fejek

I/O fej: vékony légrés választja el a lemeztől.

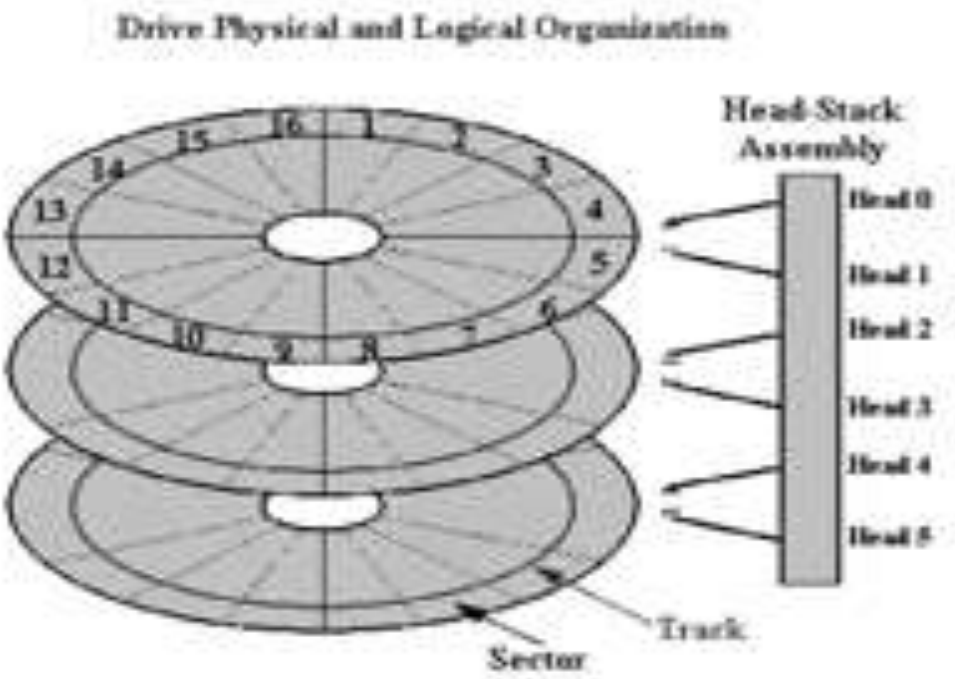
Sávok (track, 5000-10000 sáv/cm**)** - egy sáv egy koncentrikus kör egy oldalon, adott fejállásnál (adott fejpozíción, sugáron)

Szektor (tipikusan 512B, 50.000-100.000 bit/cm)**: egy sávon körcikk** pl.: fejléc + 4096 bit (= 512B) adat + hibajavító kód (Hamming vagy **Reed-Solomon**).
Címük.
Szektor rés: a szektorok közötti hézag, hogy az írás ne rontsa el a szomszédos szektort.

Formázott és **formázatlan kapacitás**.

Winchester lemez (**IBM**), légmentesen lezárt. Kezdetben 30 MB fix + 30 MB cserélhető. Az átmérő régen 50 cm, mostanában 3 – 12 cm közötti, sőt, kisebb is lehet.
Lemezegység közös tengelyen több lemez, a ma szokásosak azonos szögsebességgel forognak.

Cilinder: több oldal egymásfeletti sávjai, egy fejállással elérhető.



A szektorok címei

• Lemezoldal + sáv + szektor címhármasok (head + cyl + sec).
• Egydimenziós logikai címek (LBA) alakíthatók ki, ha
– az oldalak adott sorrendben beszámozottak,
– a sávok is adott sorrendben számozottak.
• A címhármasból(ba) le(vissza)képezhető az egydimenziós logikai cím. Ezt a leképzést végezheti a kontroller!
• „Fentről” a diszk így 0-n közötti szektorokból (blokkokból) „látszik”.

Keresési idő (seek time): fej mozgatása a megfelelő sávra **(**sáv/cilinder keresés). 5-10 ms. Közelebbinél kisebb. Meghatározó.

Forgási késleltetés (rotation latency): míg a szektor elfordul a fej alatt. Átlagosan egy fél fordulat ideje, 3-6 ms (60-180 fordulat/sec). rpm: rotation per minute (fordulat per perc). Pl. 7200 rpm = 120 fordulat/sec.

Átviteli sebesség: 20-40 MB/sec.

Maximális ↔ átlagos

Írás sűrűség:

- Régen: belül maximális, kifelé egyre kisebb (forgás szög alapján).
- Jelenleg: 10-30 zóna, a külső zónákban több szektor van egy sávon.

Lemezvezérlő: vezérli a hardvert, nyilvántartja és átcímzi a hibás sávokat.
Szoftver parancsokat hajt végre: karmozgatás,
READ, **WRITE**, **FORMAT**, **…** utasítások.
További feladatai: hiba felismerés/javítás,
soros – párhuzamos és
párhuzamos – soros átalakítás.

Hajlékony (floppy) lemez: szerviz célokra
(karbantartási információk tárolására) találták ki. Az I/O fej hozzáér a lemezhez: gyorsan kopik, ezért leáll, ha éppen nincs feladata. Kb. 0,5s, míg a lemez fölpörög.

Lemezvezérlés

PC-ken kezdetben **CPU** regiszterekbe töltött fej, cilinder, szektor-címek alapján a **BIOS** (Basic Input Output System) vezérelt.
Seagate lemezegység: 20 bites szektor cím. 4 fej (4 bit), 306 cilinder (10 bit) és sávonként 17 db 512 bájtos szektor (6 bit).
Később kevés lett 10 bit a cilinder címzésére.
IDE (Integrated Drive Electronics, max. 504 MB): a meghajtóba integrált vezérlő. Seagate kompatibilis!
„Hazudnak” a **BIOS**-nak.
A címet a vezérlő fej-cilinder-szektor címre fordítja.

EIDE (Extended IDE): **LBA** (logikai blokk címzés -Logical Block Addressing). Cím: 0 .. 2^{28}-1. Maximum 128 GB
ATA-3 (**AT A**ttachment, **AT** kiegészítő), majd
ATAPI-4 (ATA Packet **I**nterface,
ATA-csomaginterfész**)** 33 MB/s
ATAPI-5 66 MB/s
ATAPI-6 100 MB/s, 48 bites szektor cím
ATAPI-7 A korábbi 80 vezetékes szalagkábel helyett 7 vezetékes kerek kábelt alkalmaz (PCI express): jobb a légáramlás.
Kezdetben 150 MB/s soros átvitel, ami várhatóan hamarosan 1,5 GB/s fölé emelkedik.
5 V helyett 0.5 V: kisebb energiafogyasztás.

SCSI (Small Computer System Interface) **lemezek:**
sokkal gyorsabb átvitelt biztosít, drágábbak is.
SCSI: sín, vezérlő + maximum 7 SCSI eszköz csatolható, 15 a wide SCSI-nál.
Eszköz lehet lemez, nyomtató, CD, szkenner,…
A sín „átmegy” az eszközökön: az eszközöknek van egy bemenő és egy kimenő csatlakozója.
Minden eszköznek 0-7 (15) közötti azonosítója van.
Egyszerre több eszköz is aktív lehet (**EIDE**: csak egy).
SCSI-1: 5MHz, 5 Mbyte/s,
SCSI-2: 10MHz, 10-20 MB/s
Fast20, Ultra: 20 MHz, 20-40 Mbyte/s
Fast40, Ultra-2: 40 MHZ, 40-80 Mbyte/sec

Kieg.:
A winchestert partícionálással több logikai meghajtóra oszthatjuk fel. Minden partíciónak van saját „állománykiosztási” táblája (FAT vagy egyéb).
RAID technológia: redundáns tárolási rendszer, az adatok elosztása vagy replikálása több, fizikailag független (ált. kevéssé költséges) merevlemez-egységen. (Részletesen lásd. Számítógép-rendszerek üzemeltetése gyakorlaton.)
(Egy hálózati munkahely működéséhez a háttértár, mint hardverelem nem feltétlenül szükséges, processzor, operatív memória, hálózati interfész igen.)

Optikai lemezek: optikai technológia, lézer fény …

Az adattárolás „sávja” itt „spirál” …

CD: 1980, Philips, Sony: **Red Book**.

• Üveg mesterlemez: írás nagy energiájú lézerrel, üreg (pit, Ø=8μ, ¼λ mély) – szint (land).

• A mesterlemezről negatív öntőforma készül.

• A negatív öntőformába olvadt polikarbonát gyantát öntenek.

• Megszilárdulás után tükröző alumínium réteget visznek rá.

• Ezt védő lakk réteggel vonják be és erre nyomtatják a címkét.

Olvasás kis energiájú infravörös lézerrel (λ=0,78μ)

Az üregből visszavert fény λ/2 –vel hosszabb utat tesz meg, mint az üreg pereméről visszavert, ezért gyengíteni fogják egymást.

Belűről induló 22188 fordulatú kb. 5,6 km hosszú spirál 35 mm-es sávban, kb. 600 menet/mm (az adattárolás belülről kifelé történik).

A jel sűrűség a spirál mentén állandó.

A fordulatszám 530 és 200 fordulat/perc között változik, hogy a kerületi sebesség állandó legyen (120 cm/s).

Ábrázolás:

1: üreg – szint és szint – üreg átmenet,
0: átmenet hiánya.

CD-ROM : 1984, **Yellow Book**.

Több szintű hibajavítás: 650 MB

ECC: Error Correction Code (Reed-Solomon)

Forgási sebesség: 1-szeres (75 szektor/s) – 32-szeres.

Keresési idő: több 100 msec, sebesség < 5 MB/sec.

1986: **Green Book**, multimédiás alkalmazásokhoz.

CD-R (írható CD – CD Recordable,):

1989: **Orange Book**.

Spirál: 0,6 μm széles vájat mutatja, ezen egy 22,05 kHz frekvenciájú 0,03 μm amplitúdójú szinusz hullám szolgál a pontos forgási frekvencia ellenőrzésére.

Alumínium helyett arany, üreg helyett sötét pont.

Az eredetileg átlátszó festéket a nagyobb energiára kapcsolt lézer sötétre változtatja.

Felírás több részletben történhet, az egyszerre felírt szektorokat **CD-ROM sáv**nak (track) nevezzük. Minden sávot megszakítás nélkül, folyamatosan kell felírni!

Trükkök az illegális másolat készítés nehezítésére:pl. szándékosan hibás ECC-k.
CD-RW (újraírható CD – CD-ReWritable): három különböző energiájú lézer (törlő, író, olvasó).

Viszonylag drága.

DVD (Digital Versatile Disk):

• precízebb mechanika,

• kisebb üreg: 0.4 μ (0.8 μ helyett),

• szorosabb spirál: 0.74 μ (1.6 μ helyett),

• vörös lézer: λ=0.65 μ (0.78 μ helyett),

Ezek együtt nagyobb jelsűrűséget engednek meg.
Kapacitás: 4.7 Gbyte (133 perces video elfér rajta). Kétoldalas kétrétegű: 17 GB.

A lézer fókuszálásával választják ki a kívánt réteget.

Az alsó réteg kapacitása kisebb.

Blu-Ray Kék lézert használ a DVD-ben használt piros lézer helyett.

Egyoldalas: 25 GB

Kétoldalas: 50 GB
Átviteli sebesség: 4,5 MB/s

Arra számítanak, hogy le fogja váltani a CD-ROM-ot és a DVD-t.

Egér (mice, mouse, **2.33. ábra**): az egér mozgatása
egy mutató mozgását váltja ki a képernyőn.

• **Mechanikus (golyós):** gumi golyó, két tengely körüli forgásra bontva a golyó gördülése, potenciométerek.

• **Optikai: LED** (Light Emitting Diode), rácsozott „asztal”, fényérzékelő. Fényvisszaverődés a raszteres egérpadról, számlálható a fényimpulzus szám.

• **Optomechanikus:** gumi golyó, résekkel ellátott tárcsák, LED, fényérzékelő.

Működése: bizonyos időnként (pl. 0,1 sec) vagy esemény hatására 3 adatos (általában 3 bájtos) üzenetet küld a soros vonalon a számítógépnek:

x, y irányú elmozdulás + az egér gombok állapota.

Nyomtatók

Mátrixnyomtató: 7-24 tű, olcsó, lassú, zajos, több példányos nyomtatás (pénztárgépek, …).

Egy soron többször is végigmehet az írófej, egy picit változtatva a pozíción: vastagított betűk.

Tintasugaras nyomtató: - olcsó, lassú,

1200-4800 dpi.

dpi = dot per inch (pont / 2.54 cm).

Piezoelektromos. Piezoelektromos hatás: Feszültség hatására bizonyos kristályok bizonyos irányban összehúzódnak/kitágulnak. Hő vezérlésű (bubblejet, festékbuborékos): A fúvókát hevítik/hűtik.

Lézernyomtató: a hengert feltöltik 1000 voltra, lézerrel modulálják (ahol fény éri a hengert, ott elveszti a töltését), a töltött részre rátapad a festék, ezt a papírra égetik. Saját CPU, memória.

Nagy a memória igénye, pl. egy A4-es 1200*1200 dpi képen 115 millió pixel van.

A nyomtató színkezelésének elve a CMYK.

Terminál: billentyűzet (keyboard) + monitor.

Billentyűzet: megszakítás a billentyű leütésekor és felengedésekor, a többit a megszakítás kezelő végzi.

Monitor:

CRT (Cathode Ray Tube): soronként állítja össze a képet (raszteres).
• Elektron ágyú: elektronokat bocsát ki.
• Eltérítő tekercsek: vízszintes és függőleges.
• Rács: szabályozza a képernyőt érő elektronok mennyiségét.

Színes monitorban 3 elektron ágyú.

LCD (Liquid Crystal Display – folyadék kristályos) monitor:
Bizonyos kristályok elektromos tér hatására fénytörési tulajdonságaikat változtatják (kristálysíkonként elfordulnak), ezzel „szűrőként" viselkednek.
Raszteres grafika megvalósítható: sorokra-oszlopokra bontott képpontok kristályai „gerjeszthetők".

TN (csavart molekula elrendeződéses – Twisted Nematic) megjelenítő:

• a megvilágító fényt a hátsó polárszűrő vízszintesen polarizálja,

• a folyadékkristály függőlegesbe forgatja a polaritást,

• az első polárszűrő csak a függőlegesen polarizált fényt engedi át.

Feszültség hatására a forgatás csökken vagy elmarad, következésképpen csökken a fényerő.

• Passzív (vízszintes és függőleges elektródák).
• Aktív mártix display (pixelenként kapcsolóelem, Thin Film Transistor), drágább, de lényegesen jobb képet ad (**TFT** megjelenítők).

Képátló: A monitor egyik ellentétes sarkától a másikig terjedő távolság, *col*-ban (2,54 cm) mérik.

Kontraszt: A részletgazdagságot jellemző tulajdonság (250–1000 :1)

Válaszidő: Az az idő, amely alatt a monitor reagál a az utasításra. *Milliszekundumban* mérik (ms). A mai modern háromdimenziós játékoknál 12 ms feletti reakcióidejű monitor utánhúzást eredményez. (A válaszidő fogalma csak LCD monitorokra vonatkozik.)

Fényerő: A monitor fényességét jellemzi. (Milyen fényes az elektronok felvillanása (CRT), milyen erős, fényes a háttérvilágítás (LCD).)

Maximális felbontás: Maximálisan mekkora felbontásra állítható.

Megjeleníthető színek száma: Megjeleníthető színárnyalatok száma. Általában 16,7 millió színt tud megjeleníteni egy monitor, de gyakran „csak” 16,2 milliót

Látószög: Az a paraméter, mely megadja, hogy a monitor milyen szögből látható. Általában két adattal jellemzik, az első a horizontális (szélesség), második a vertikális (magasság) adat. Például: H:160°/ V:150°

Video RAM-ok

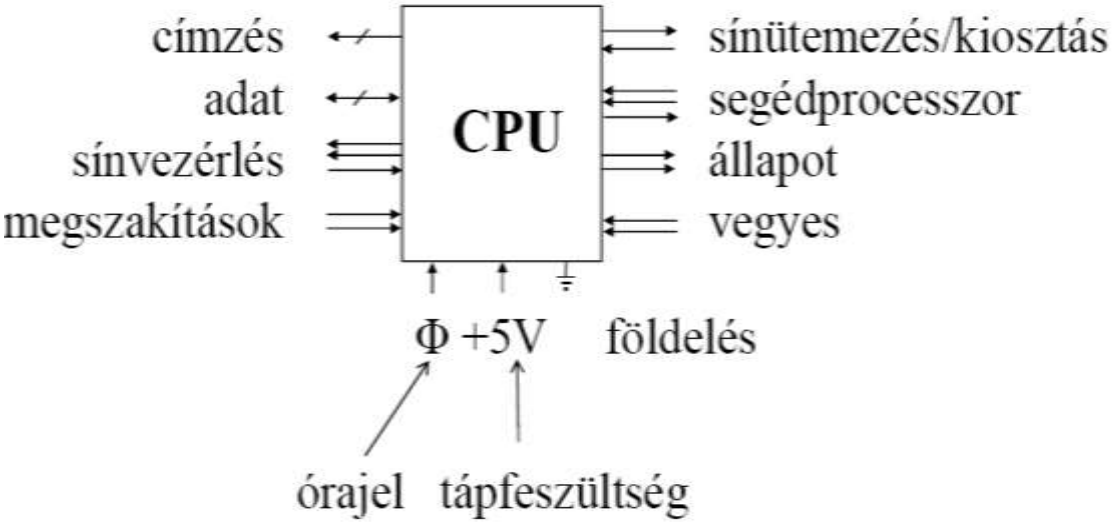
A megjelenítők másodpercenként 60-100 alkalommal frissítik a képernyőt a videomemóriából, ami a videokártyán van. Több képernyőnyi tartalom.

Általában pixelenként 3 bájt (RGB).

A képernyő kiszolgálásához nagy sávszélesség kell: korábban PCI sín (127,2 MB/s),
Pentium II-től AGP (Accelerated Graphics Port) sín 252 MB/s.
Újabb verziók 2-, 4-, 8-szoros sávszélességet tudnak.

Színpaletta (indexelt színelőállítás): 256 elem, mind 3 bájt (RGB),
a pixelekhez csak az elem indexét tárolják.

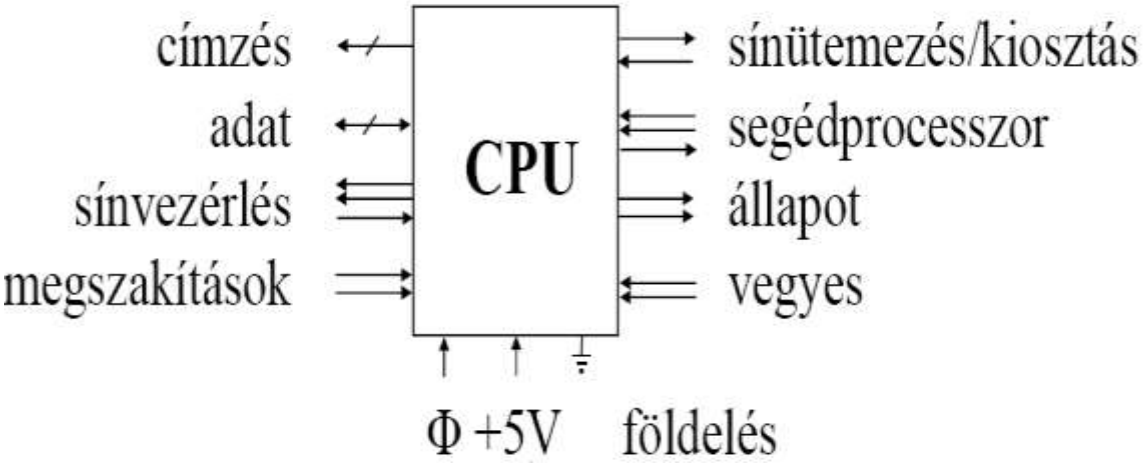
Pl.: -100x200-as méretű 256 színt tartalmazó kép megjelenítéséhez 20000 bájt videomemóriára van szükség, mert 256 szín ábrázolható 1 bájton. (Kell még a palettainformáció, de az minimális.)
- 1280 x 768 pixel felbontású, 65536 színt tartalmazó kép megjeleníthető, ha videomemóriánk kapacitása 2 MB, de 1280 x 1024 felbontásút nem.
Szerverek, pl. fájlszerver esetén kevésbé fontos a nagy kapacitású videomemória (nagy kapacitású operatív tár, nagy kapacitású háttértár, nagysebességű hálózati csatoló fontosabb).



Általában egyetlen lapkán van. **Láb**akon keresztül kommunikál a többi egységgel (**3.34. ábra**). Lábkiosztás: pinout. (Lábkészlet.)

Lábak (**pins**) három típusa: **cím adat vezérlés** Ezek párhuzamos vezetékeken, az un. **sín**en keresztül kapcsolódnak a **memória**, az **I/O** egységek hasonló lábaihoz.

Lényeges a cím- és adatlábak száma (**3.34. ábra**):
• Ha *m* címláb van, akkor 2^m memóriarekesz érhető el (tipikus *m* = 16, 20, 32, 64).
• Ha *n* adatláb van, akkor egyszerre *n* bit olvasható illetve írható (tipikus *n* = 8, 16, 32, 36, 64).

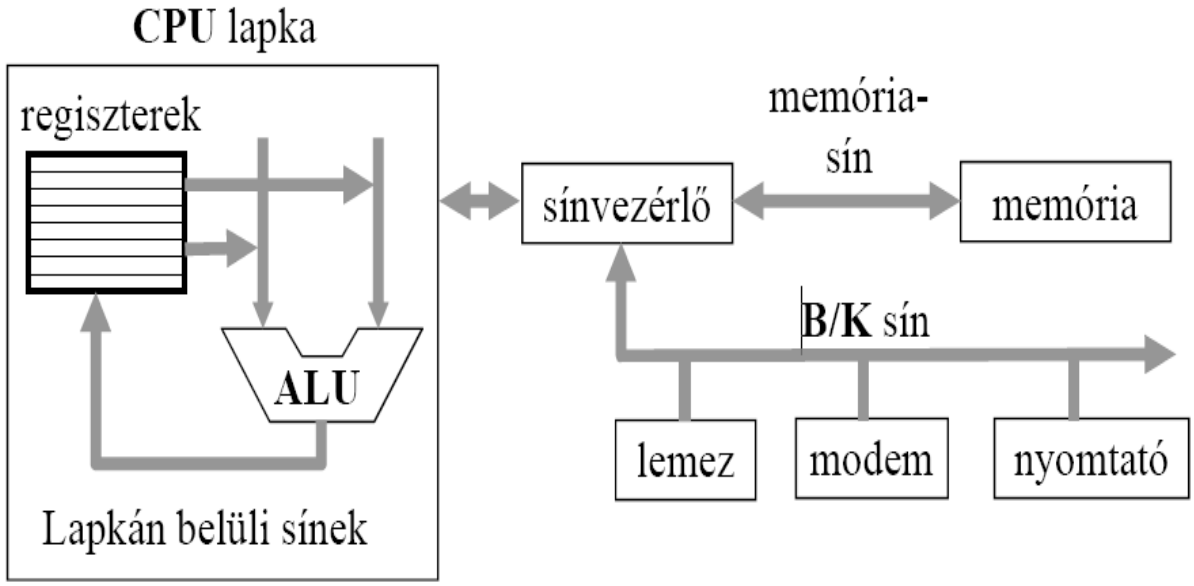


Óra, áram (3.3 v. 5V), föld, továbbá **vezérlőlábak**:
• sín vezérlés (bus control): mit csináljon a sín
• megszakítások,
• sín kiosztás (ütemezés, egyeztetés – bus arbitration): kinek dolgozzon a sín,
• segéd processzor vezérlése, jelzései,
• állapot,
• egyebek.

Pl. **utasítás betöltése**:

• A **CPU** kéri a sín használat jogát.
• Az utasítás címét a cím lábakra teszi,
• vezérlő vonalon informálja a memóriát, hogy olvasni szeretne,
• a memória a kért szót az adat vonalakra teszi, kész jelzést tesz egy vezérlő vonalra,
• a **CPU** végrehajtáshoz átveszi az utasítást.

Sín (bus): Korai személyi számítógépeknél egyetlen (külső) rendszersín, manapság legalább kettő van: egy belső és egy külső (**I/O**), **3.35. ábra**.



Sínprotokoll: a sín működésének + a csatlakozások mechanikai, íelektronikus definíciója

Mesterek (**masters**): aktív (kezdeményező) berendezések (**CPU**, lemezvezérlő).

Szolgák (**slaves**): passzív (végrehajtó) berendezések (lemez vezérlő, **CPU**), **3.35. ábra**.
Ez a szereposztás tranzakciónként eltérő lehet.

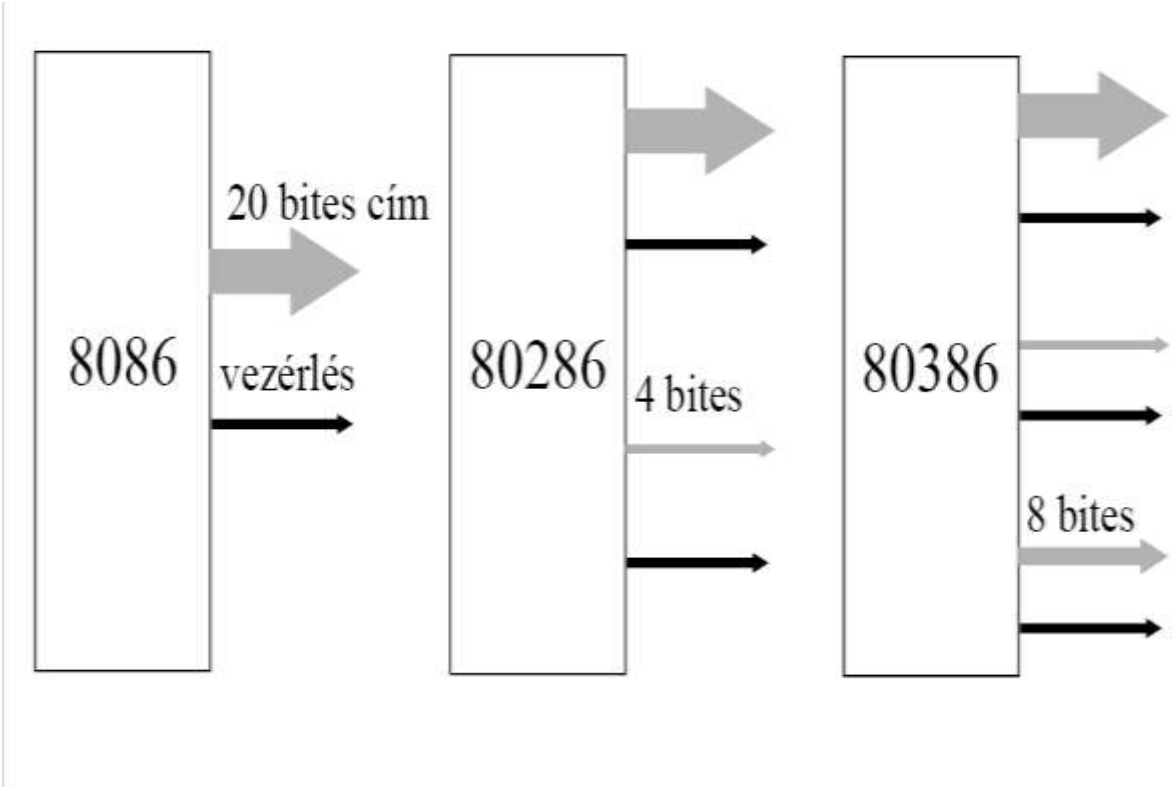
Mester	Szolga	Példa
CPU	Memória	Utasítások és adatok betöltése
CPU	I/O eszközök	Adatátvitel kezdeményezése
CPU	Segédprocesszor	**CPU** felkínálja az utasítást
I/O	Memória	DMA (Direct Memory Access; közvetlen memóriaelérés)
Segédprocesszor	**CPU**	Segédprocesszor kéri az operandusokat

A memória sohasem lehet mester!

A sínhez kapcsolódó lapkák lényegében erősítők.
Mester – **sín vezérlő** (**bus driver**) – sín.
Sín – **sín vevő** (**bus receiver**) – szolga.
Mester–szolgáknál: **sín adó-vevő** (**bus transceiver**).
A csatlakozás gyakran **tri-state device** vagy **open collector** – **wired-OR** segítségével történik.

Sávszélesség: továbbítható bitek száma) sec.
Sávszélesség növelése:
Gyorsítás: probléma a sín aszimmetria (skew), kompatibilitás.
Sínszélesség: több vezeték → drágább, kompatibilitás.

Sínszélesség (pl. **IBM PC**: **3.37., 3.51. ábra**).



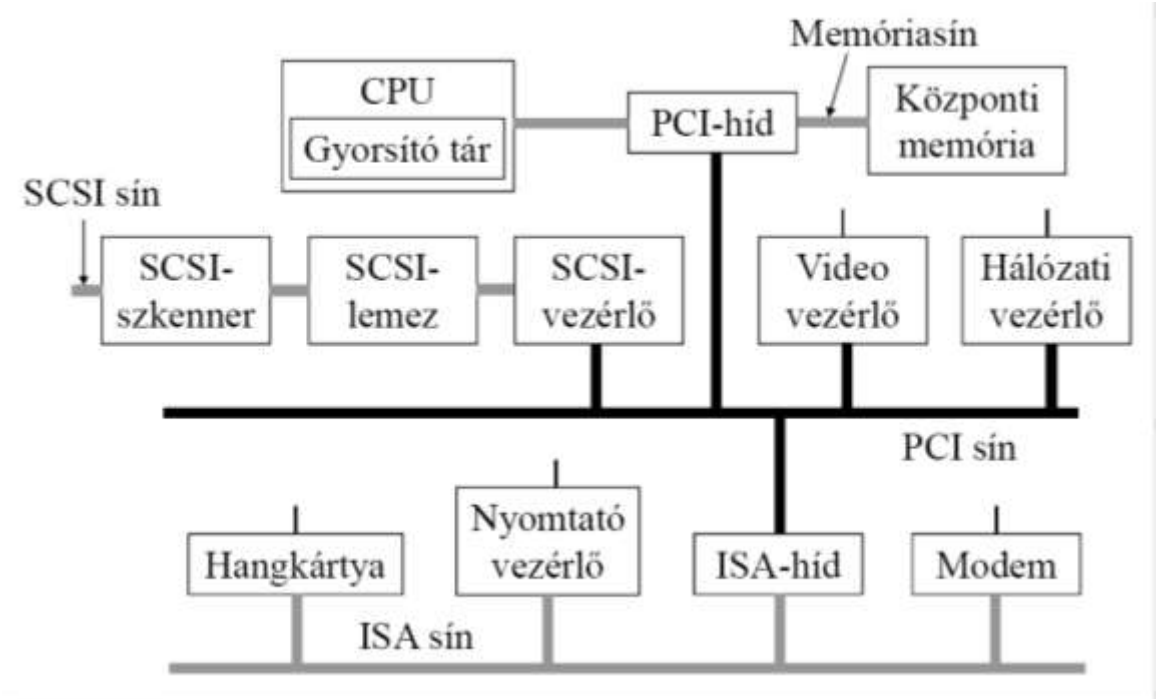
3.37. ábra. A cím szélességének növekedése az elmúlt időszakban

Alaplap (motherboard, parentboard, **3.51. ábra**)
Rajta van a **CPU**, sín(ek), ezen illesztő helyek (slots) a memória és a **beviteli/kiviteli (Input/Output – I/O)** eszközök számára (**3.51., 2.28. ábra**).

I/O eszköz: maga az eszköz + vezérlő (controller) külön kártyán vagy az alaplapon (**2.29. ábra**).

Gyorsabb CPU gyorsabb sínt igényel!

Kívánság: PC cseréjénél megmaradhasson a régi perifériák egy része: az új gépben is kell a régi sín!
Sínek szabványosítása.
Egy gépen belül több sín is használható: **2.30. ábra**.



2.30. ábra. *Egy tipikus modern PC PCI, SCSI és ISA sínnel*

Sokszorozott (multiplexed) **sín:** pl. először a cím van a sínen, majd az adat (ugyanazokon a vezetékeken).

Ilyenkor a sín szélessége lényegesen csökken (olcsóbb, kevesebb láb szükséges a sínhez valócsatlakozáshoz), csökken a sáv szélesség is, de nem olyan mértékben.

Általában bonyolultabb a sín protokoll.

Sínek időzítése

Szinkron sín: 5 – 100 MHz-es órajel van a sín egy vezetékén. Minden síntevékenység az órajelhez van igazítva.

Minden sínművelet a ciklusidő (sín ciklus) egész számú többszöröséig tart: pl. 2.1 ciklusidő helyett 3 ciklusidő kell. A leglassabb eszközhöz kell a sín sebességét igazítani, a gyors eszköz is lassan fog működni.

Aszinkron sín:

Minden eseményt egy előző esemény okoz!

Ugyanazon a sínen gyors és lassú mester - szolga pár is lehet.

Sínütemezés (kiosztás)

Ha egyszerre többen is igénylik a sínt (**CPU, I/O** vezérlő), akkor a **sínütemező** (bus arbiter) dönt.

- Centralizált: láncolás (daisy chaining), (**3.40. (a) ábra**): gyakorlatilag az eszközökhöz egy prioritást rendel, hogy mennyire van az eszköz az ütemezőhöz közel.

- Decentralizált: - pl. 16 prioritású: 16 eszközhöz 16 kérés vonal, minden eszköz minden kérés vonalat figyel, tudja, hogy a saját kérése volt-e a legmagasabb prioritású.

Sín műveletek

Az eddigiek **közönséges sín műveletek** voltak.

Blokkos átvitel (3.42. ábra): A kezdő memória címen kívül az adatsínre kell tenni a mozgatandó adatok számát. Esetleges várakozó ciklusok után ciklusonként egy adat mozgatása történik.

Megszakítás kezelés: más tantárgyban tárgyaljuk részletesen.

Több processzoros rendszerekben:
olvasás – módosítás – visszaírás ciklus: szemafor.

Példák sínekre

Az első **IBM PC** (**3.37. ábra**) 62 vonalas (vezeték, line), 20 címnek, 8 adatnak + **DMA**, megszakítás …

PC/AT szinkron sín (**3.51. ábra**): további 36 vezeték (címnek összesen 24, adatnak 16, …).

Microchannel (**IBM OS/2** gépekhez), szabadalmak

ISA (Industry Standard Architecture) lényegében 8.33 MHz-es **PC/AT** sín (sávszélesség: 16.7 MB/s).

EISA (Extended **ISA**) 32 bitesre bővített **ISA** (sávszélesség: 33.3 MB/s).

Színes TV-hez 135 MB/s sávszélesség kellene (1024*768 pixel, 3 bájt*2, 30 kép/sec).
lemez → memória → képernyő

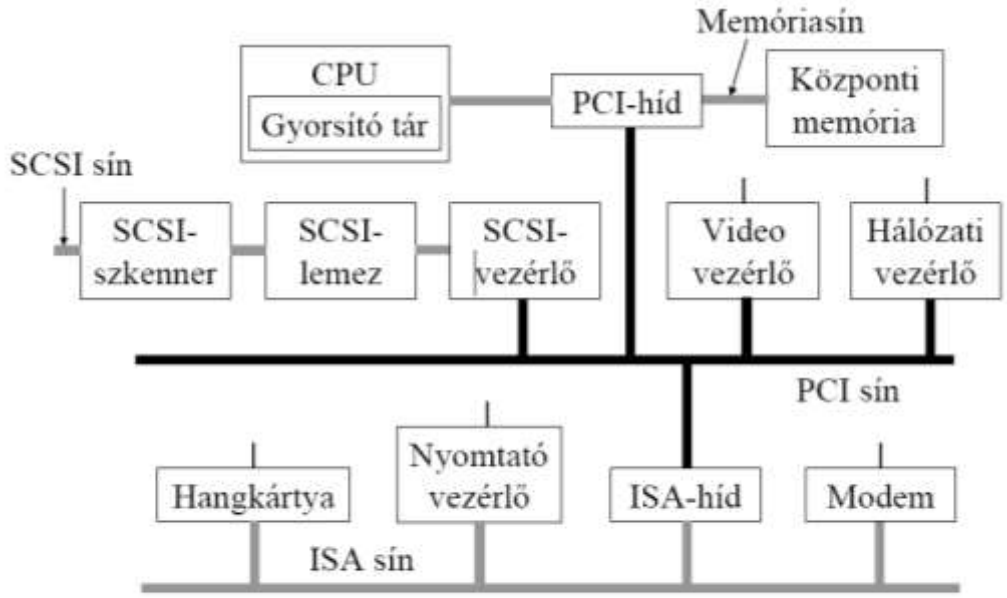
PCI (Peripheral Component Interconnect): 32 bites adat átvitel (33,3 MHz, sávszélesség: 133 MB/s) szabadon felhasználható licensz.

Multiplexelt cím- és adatkivezetések.

Új változatai: 64 bites adat, 66 MHz, 528 MB/s.
Problémák:
• a memóriához lassú,
• nem kompatíbilis az **ISA** bővítőkártyákkal.

Megoldás (**3.52.** vagy **2.30. ábra**): több sín

Belső sín, **PCI** híd, **PCI** sín, **ISA** híd, **ISA** sín.



2.30. ábra. *Egy tipikus PC PCI, SCSI és ISA sínnel 2000 körül*

PCI sín ütemezés **(3.54. ábra)**. A **PCI** sín centrális ütemezőt használ.

Általános soros sín (USB)

Universal Serial Bus

Igény: bármikor könnyen, azonos módon lehessen sokféle perifériát kapcsolni a géphez, akár a gép működése közben, hardver ismeretek nélkül:
– ne kelljen kikapcsolni a gépet,
– ne kelljen szétszedni a gépet,
– ne kelljen újra boot-olni,
– ne kelljen áramellátásról gondoskodni,
–…
Plug 'n Play (csatlakoztasd és működik) perifériák.

USB (Universal Serial Bus - általános soros sín):

Négy vezeték: adatok (2), tápfeszültség (1), föld (1).

USB 1.0 1,5 Mbps (billentyűzet, egér, …),
USB 1.1 12 Mbps (nyomtató, fényképezőgép, …),
USB 2.0 480 Mbps (DVD lejátszó, …) a maximális adatátviteli sebesség.

A központi elosztó (**root hub**) 1 ms-onként üzenetekkel (**frame, 3.54. ábra**) kommunikál az eszközökkel.

A frissen csatlakoztatott eszköz címe 0.
Ha a központi elosztó tudja fogadni az eszközt, akkor egyedi címet (1-127) ad neki (**konfigurálja**)

Frame – keret

Egy vagy több csomagból áll. Az egyes csomagok haladhatnak a központból az eszközök felé vagy fordítva. A haladási irány egy kereten belül is változhat.
Az első csomag mindig **SOF**:
Start Of Frame – keret kezdet, szinkronizálja az eszközöket.

A keret lehet

• **Control** – vezérlő: Eszközkonfigurálás, Parancs, Állapot lekérdezés.
• **Isochronous** – izoszinkron: valós idejű eszközök használják, pl. telefon. Hiba esetén nem kell ismételni az üzenetet.
• **Bulk** – csoportos: nagy tömegű adat átvitelére szolgál.
• **Interrupt** – megszakítás: Az **USB** nem támogatja a megszakítást, helyette pl. 50 ms-enként lekérdezhető az eszköz állapota.

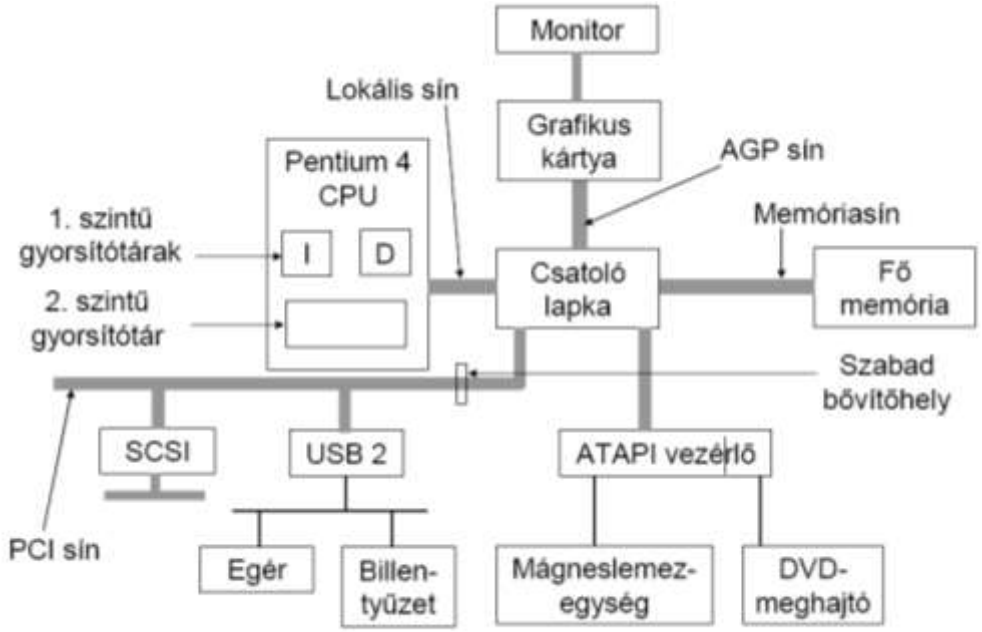
A csomag lehet

• **Token** – parancs (központ küldi az eszköznek): (**SOF.**
IN – be: adatokat kér az eszköztől.
Az **IN** parancsban meg lehet adni, melyik eszköztől milyen adatokra van szükség.
OUT – ki: adatok fogadására kéri az eszközt.
SETUP – beállítás: konfigurálja az eszközt.

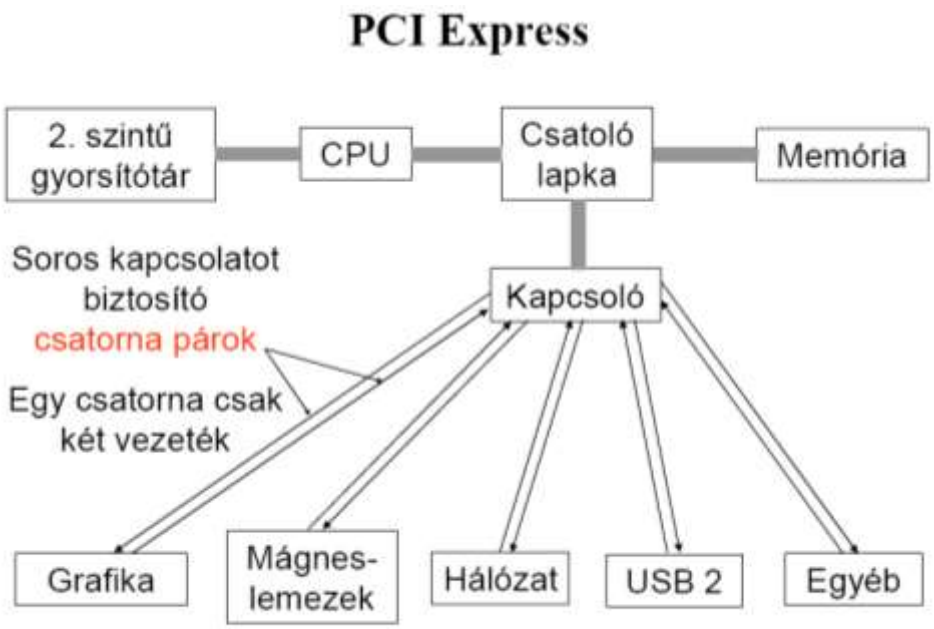
• **Data** – adat: 64 bájt információ mozgatása akármelyik irányban. A **Data** csomag részei:
- **SYN**: 8 bit szinkronizáció,
- **PID**: a csomag típusa (8 bites),
- **PAYLOAD**: hasznos adat,
- **CRC**: Cyclic Redundancy Code – ciklikus redundancia kód (16 bit az adatátvitel helyességének ellenőrzésére).

• **Handshake** – kézfogás:
- **ACK**: az előző adatcsomagot hibátlanul vettem,
- **NAK: CRC** hibát észleltem,
- **STALL**: kérem, várjon, el vagyok foglalva.

• **Special** – speciális.



3.53. ábra. *Egy modern Pentium 4 rendszer sín struktúrája*



3.57. ábra. *Egy tipikus PCI Express rendszer vázlata*

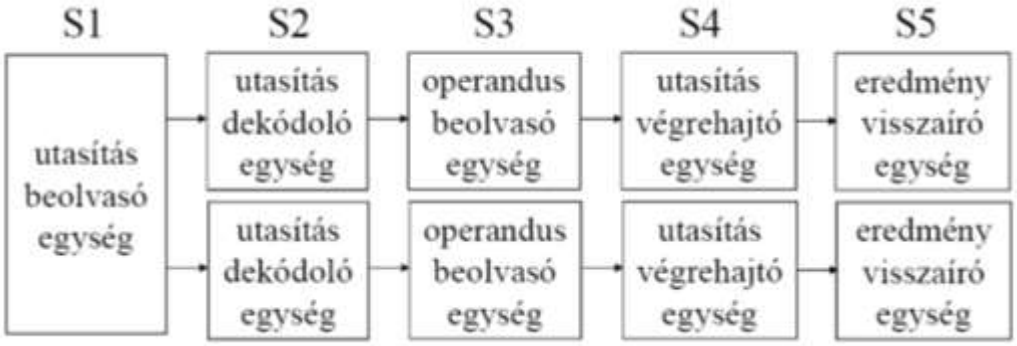
PCI Express (PCIe) Egy kapcsolón keresztül érik el (point-to-point, síntopológia) a sínt (minden eszköz úgy látja, mintha saját külön sínnel rendelkezne). A kapcsoló gondoskodik a P-P kapcsolatok létrehozásáról és a vezérli a sín adatforgalmát. A csatorna (link) duál szimplex.

Hagyományos sín	**PCI Express**
Több leágazású sín	Központosított kapcsoló
Széles, párhuzamos sín 32 vagy 64 bites	Keskeny, közvetlen soros kapcsolat
Bonyolult mester – szolga kapcsolat	Kicsi, csomagkapcsolt hálózat
	CRC kód: nagyobb megbízhatóság
	A csatlakozó kábel > 50 cm lehet
	Az eszköz kapcsoló is lehet
	Meleg csatlakoztatási lehetőség
	Kisebb csatlakozók: kisebb gép
	Nem kell nagy bővítőkártyával csatlakozni a sínhez
	Egy csatorna hasznos sávszélessége minimum 2 Gbps, de bíznak benne, hogy hamarosan 10 Gbps

Párhuzamos számítógépek osztályozása

1. SISD (Neumann). Klasszikus, szekvenciális, egy utasításfolyama, egy adatfolyama van, és egyszerre egy műveletet végez el.

De: Több szállítószalagos CPU (emlékeztető)



Két szállítószalag (**2.5. ábra**):

• Két végrehajtó egység, de közös regiszterek,
• A két szállítószalag lehet különböző is (**Pentium**):

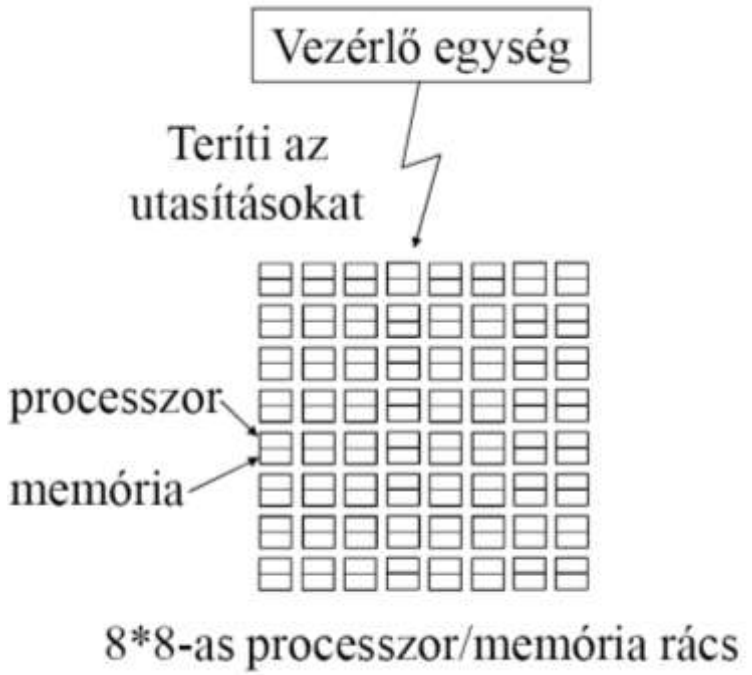
Fő – ez többet tud, elsőbbséget élvez – és mellék.

Bonyolult szabályok a párhuzamos végrehajthatóságra
(fordítók vagy hardver).

Processzor szintű párhuzamosítás

2. SIMD számítógépek: Single Instruction Multiple Data stream (egy utasításfolyam, több adatfolyam).
Egyetlen vezérlőegységgel rendelkeznek, ami egyszerre több utasítást ad ki, de több ALU-juk van, amik a kiadott utasítást több adathalmazon párhuzamosan végzik el. **Szokásos adatszerkezeteik a vektorok és tömbök.**
Nagy **tömegű számítást igénylő tudományos és mérnöki munkákra,**

- **Tömb (array) processzor** (**2.7. ábra**)



8*8-as processzor/memória rács

Sok azonos processzor (**ILLIAC IV**: (4*)8*8, 1972.), mindnek saját memóriája. Vezérlő processzor adja ki a feladatot
Mindegyik processzor ugyanazt csinálja, de a saját adatain.
Már nem divatos (drága, nehéz kihasználni).

- **Vektor processzorok:** kereskedelmi forgalomban sikeresebbek voltak. Cray gépei (Cray-1 vektor szuperszámítógép 1976, C90, T90 évtizedeken keresztül vezető szerpet töltöttek be.) Pl. ehhez hasonló feladatokkal jár egy számolásigényes feladat:
  For (i = 0 ; i < n ; i++) a[i] = b[i] + c[i];
  Vektorregisztereket használnak.
  A vektorregiszter több hagyományos regiszterből áll. Gyors szállítószalag gondoskodik a regiszterek feltöltéséről, szintén gyors

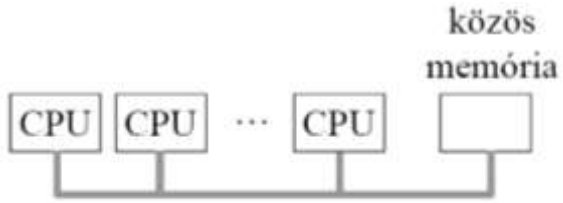
szállítószalag továbbítja a regiszterek tartamát az aritmetikai egységbe, pl. a vektor regiszterek összeadásához Az eredmények szintén vektor regiszterbe kerülnek.
Jól kombinálhatók hagyományos processzorokkal (csővezetékes feldolgozással).

3. MISD: több utasítás dolgozik ugyanazon az adatrészen. Nem meggyőző, hogy létezik-e ilyen gép? Egyesek a csővezetékes gépeket ide sorolják.

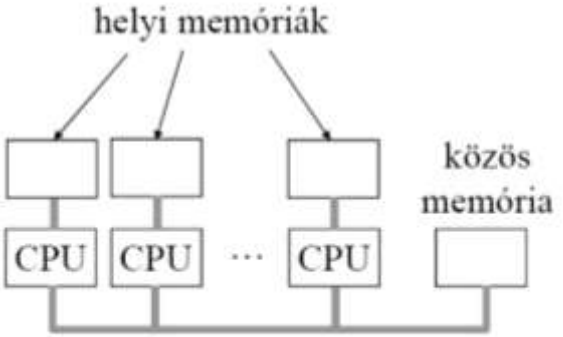
4. MIMD (Multiple Instruction Multiple Data stream).

MIMD: több független CPU egysége, amelyek egy nagy rendszer részeiként működnek. Multiprocesszorok és multi-számítógépek ilyenek.

- **Multiprocesszorok = közös memóriájú rendszer (megosztott memóriájú endszer)**



A közös memória megkönnyíti a feladat megosztását.
• Csak közös memória.
Nagyon terheli a memóriasínt.



2.8. ábra

• Lokális memória is van.

Sok (>64) processzoros rendszert nehéz építeni a közös memória miatt. Pl. a Sequent NUMA-Q architektúra 256 processzorsos, de nem egységes memória elérésű.

- **Multi-számítógépek**

Üzenetátadásos multi-számítógépek.

Nincs közös memória: A **CPU**-k üzenetekkel tartják egymással a kapcsolatot. Néhány μs üzenet idő.

Topológia: 2-3 dimenziós hálók (rács), fák, gyűrűk, teljes gráf.

Közel 10 000-es rendszer is van.

Ide tartoznak a COW rendszerek is. (munkaállomások klasztere).

Kommunikáció: PVM, MPI. (Szoftverek.)

Teljesítménynövelés (Mikroarcitektúra szinten)

- **A gyorsítótár**

- **Elágazásjövendölés:** a modern számítógépek magas szinten vannak csővezetékkel ellátva (7-10).

  Legkorábban a dekódoló veheti észre, hogy ugró utasítást kell végrehajtani, de addigra a következő utasítás már a csővezetékben van! (4.40 ábra).
  Eltolás rés (delay slot): Az ugró utasítás utáni pozíció. Az ugró utasítás végrehajtásakor ez az utasítás már a csővezetékben van!
  A feltétel nélküli elágazást (ugró utasítást) követő utasítás végrehajtódik vagy
  • **Pentium 4**: bonyolult hardver gondoskodik a csővezeték helyreállításáról

A feltételes elágazások még rosszabbak. A korai csővezetékes gépek 3-4 ciklusra bedugultak.

Sok gép megjövendöli, hogy egy ugrást végre kell hajtani vagy sem.

Egy triviális jóslás:

• a visszafelé irányulót végre kell hajtani (ilyen van a ciklusok végén),

• az előre irányulót nem (jobb, mint a semmi).

Feltételes elágazás esetén a gép tovább futhat a jövendölt ágon,

• amíg nem ír regiszterbe vagy

• csak „firkáló” regiszterekbe ír.

Ha a jóslat bejött, akkor minden rendben, ha nem, akkor sincs baj (visszaforgatható).

Bonyolult, sok könyvelés.

Több feltételes elágazás egymás után!

Dinamikus elágazás jövendölés

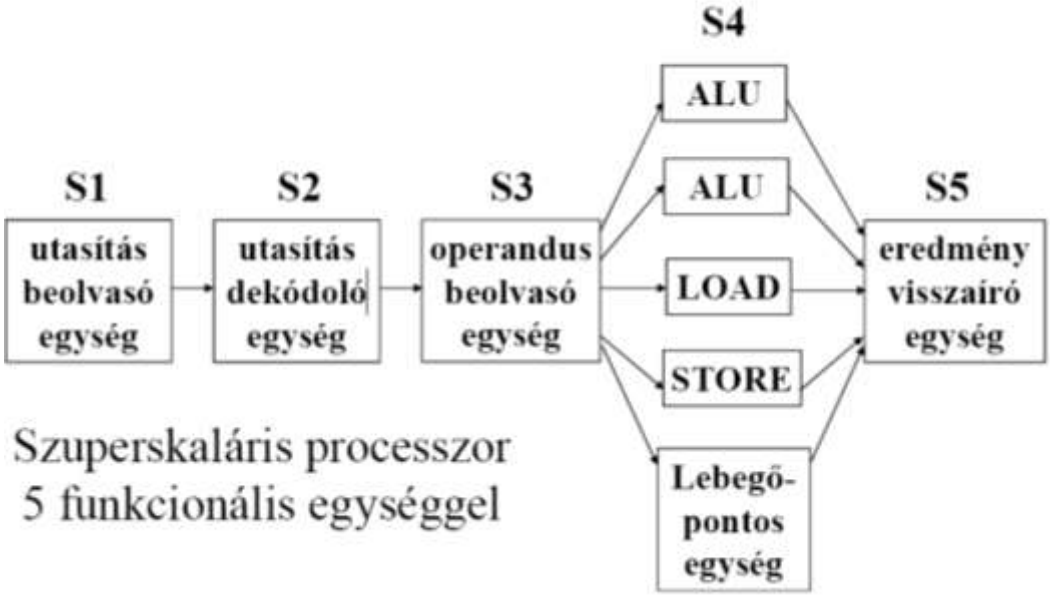
Elágazás előzmények tábla (**4.41. ábra**), minden feltételes elágazásra egy bejegyzés (hasonló jellegű, mint a gyorsítótár).

- **Sorrendtől eltérő végrehajtás**

Néhány CPU megengedi, hogy a függő utasításokat átugorjuk (függő utasítás: amikor pl. egy utasításnak egy olyan értékre van szüksége, amit egy korábbi utasítás számít ki, akkor annak végrehajtását meg kell várnia).

Néhány gép bizonyos utasításokat átugorva függőben hagy, előbb későbbi utasításokat hajt végre, és később tér vissza a függőben hagyott utasítások végrehajtására (**~4.44. ábra**).

Szuperskaláris architektúrák (2. 6. ábra)



Szuperskaláris architektúra esetén a dekódoló egység az utasításokat mikroutasításokra darabolhatja.
Legegyszerűbb, ha a mikroutasítások végrehajtási sorrendje megegyezik a betöltés sorrendjével, de ez nem mindig optimális.

Függőségek: nem olvashatjuk (azt a regisztert), aminek az írása még nem fejeződött be (**RAW** – Read After Write), és nem írhatjuk felül, amit korábbi utasítás olvasni (**WAR** – Write After Read) vagy írni akar (**WAW** - Write After Write).

Annak nincs akadálya, hogy onnan olvassunk, ahonnan egy korábbi olvasás még nem fejeződött be, tehát az olvasás utáni olvasás nem okoz függőséget. Nincs **RAR** (Read After Read) függőség.

Pentium 4 (2000. november)

Felülről kompatibilis az **I8088**, **…**, **Pentium III**-mal.
29.000, …, 42 → 55 M tranzisztor, 1,5 → 3,2 GHz,
63-82W, 478 láb (**3. 44. ábra**), 32 bites gép, 64 bites adat sín.
NetBurst architektúra. (NetBurst csővezeték.)
2 fixpontos **ALU** Mindkét **ALU** kétszeres órajel sebességgel fut.

Többszálúság (hyperthreding):

Többszálúság (hyperthreding, 8.7. ábra)

(a)	**A1**	**A2**			**A3**	**A4**	**A5**			**A6**	**A7**	**A8**
(b)	**B1**			**B2**			**B3**	**B4**	**B5**	**B6**	**B7**	**B8**
(c)	**C1**	**C2**	**C3**	**C4**			**C5**	**C6**			**C7**	**C8**

Óraciklus →

Az (a), (b) és (c) processzus külön futtatva az üres téglalapoknál várakozni kényszerül a memóriához fordulások miatt.
A többszálúság többszörözött regiszter készlet és némi szervező hardver hozzáadásával valósítható meg:

EGYÜTT	**A1**	**B1**	**C1**	**A2**	**B2**	**C2**	**A3**	**B3**	**C3**	**A4**	**B4**	**C4**

Pentium 4

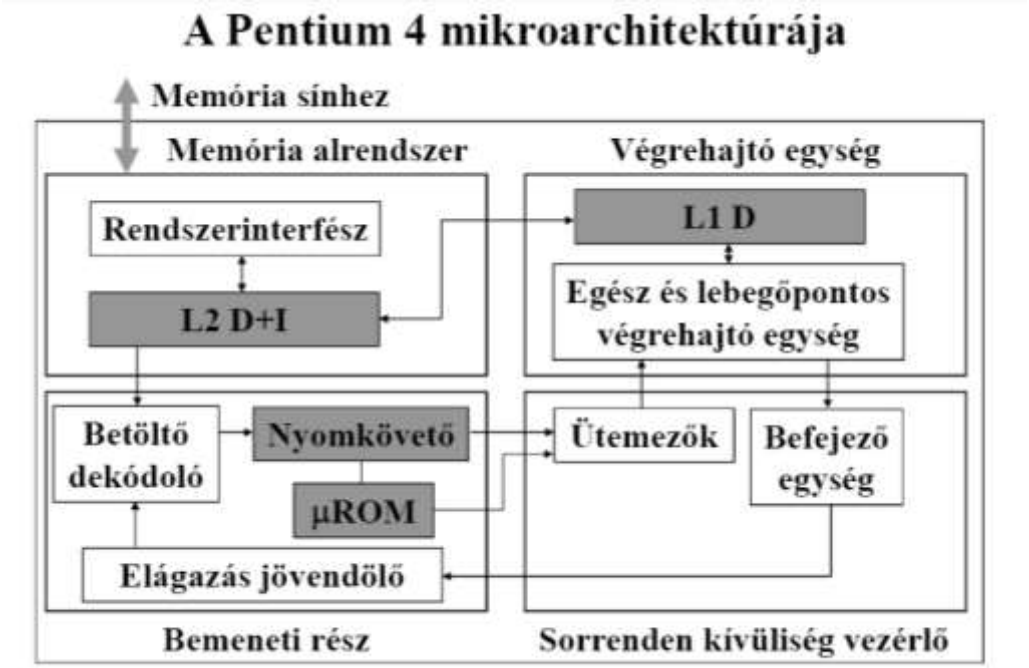
Gépi utasítások → **RISC** szerű mikroutasítások, több mikroutasítás futhat egyszerre: szuperskaláris gép, megengedi a sorrenden kívüli végrehajtást is.

2-3 szintű belső gyorsító tár.

Pentium 4

L1: **8 KB** adat
8 KB utasítás + nyomkövető tár akár 12000 dekódolt mikroutasítás tárolására.
L2: **256 KB – 512 kB - 1 MB** (első, második, harmadik generációs P4). egyesített. Előre betöltő egység.
Az Extrem Edition-ban **2 MB** (közös) **L3** is van.

Multiprocesszoros rendszerekhez:.
Szimatolás – snoop
Minden processzor figyeli a sínen a többi processzor memóriához fordulásait (szimatol). Ha valamelyik processzor olyan adatot kér, amely bent van a gyorsító tárában, akkor a gyorsító tárából megadja a kért adatot és letiltja a memóriához fordulást.



4.46. ábra. A Pentium 4 blokkdiagramja

NetBurst csővezeték:

- **A Pentium 4 bemeneti rész (4.46 ábra)**

  L2-ből betölti és dekódolja a programnak megfelelő sorrendben az utasításokat. Az utasításokat **RISC** szerű mikroműveletek sorozatára bontja. A dekódolt mikro-műveletek a **Nyomkövető**be kerülnek (nem kell újra dekódolni).

  Elágazás jövendölés.

  A bemeneti rész az utasításokat **L2**-ből kapja. A dekódoló egység az utasításokat **L2**-ből kapott utasításokat dekódolja, **RISC** szerű mikroműveletekre bontja, a nyomkövető gyorsító tárban tárolja (akár 12 K mikroműveletet) a programnak megfelelő sorrendben. 6 mikroműveletet csoportosít minden nyomkövető sorban.

Feltételes elágazásnál az utolsó **4 K** elágazást tartalmazó **L1 BTB**-ből (Branch Target Buffer – elágazási cél puffer) kikeresi a jövendölt címet, és onnan folytatja a dekódolást. Ha az elágazás nem szerepel **L1 BTB**-ben, akkor statikus jövendölés történik: visszafelé ugrást végre kell hajtani, előre ugrást nem.

- **Sorrenden kívüliség vezérlő (4.46 ábra)**

  Az utasítások a programnak megfelelő sorrendben kerülnek az ütemezőbe, eltérő sorrendben kezdődhet a végrehajtásuk (esetleg regiszter átnevezéssel), de a pontos megszakítás követelménye miatt az előírt sorrendben fejeződnek be.
  ALU ütemezők, memória sor.
  Két regisztergyűjtő, mindkettő 128 regisztert tartalmaz, időben változik, hogy melyikben mi van.

- **2 ALU**

- **Befejező egység** feladata, hogy az utasítások a programnak megfelelő sorrendben fejeződjenek be.

A **Pentium 4-nek** nagyon sok elődje van (kompatibilitás!), a fontosabbak:
• **4004**: 4 bites,
• **8080**: 8 bites,
• **8086**, **8088**: 16 bites, 8 bites adat sín.
• **80286**: 24 bites (nem lineáris) címtartomány (16 K darab 64 KB-os szegmens).
• **80386**: **IA-32** architektúra, az Intel első 32 bites gépe, lényegében az összes későbbi is ezt használja.
• **Pentium II** –től **MMX** utasítások.

A Pentium 4 üzemmódjai

- real (valós): az összes **8088** utáni fejlesztést kikapcsolja (valódi **8088**-ként viselkedik).
Hibánál a gép egyszerűen összeomlik, lefagy.

- virtuális 8086: a **8088**-as programok védett módban futnak (ha **WINDOWS**-ból indítjuk az **MS-DOS**-t, és ha abban hiba történik, akkor nem fagy le hanem visszaadja a vezérlést a **WINDOWS**-nak).

- védett: valódi **Pentium 4**. 4 védelmi szint (**PSW**):
0: kernelmód (operációs r.), **1**, **2**: ritkán használt,
3: felhasználói mód.

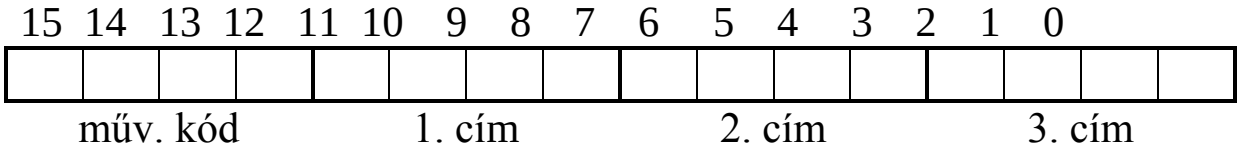
PSW (Program Status Word): az eredmény negatív, nulla, ... mód, prioritásszint, megszakítás-állapot, ...

Utasításformák, utasításhossz (5.10-11. ábra).

Műveleti kód			
Műveleti kód		cím	
Műv. kód	cím1	cím2	
M.k	. cím1	cím2	cím3

A műveleti kód kiterjesztése

k bites műveleti kód esetén 2^k különböző utasítás lehet, ***n*** bites címrésznél 2^n memória címezhető, fix utasítás hossz esetén egyik csak a másik rovására növelhető (**5.12. ábra**).



Lehetőségek:
• fix utasításhossz: rövidebb kód mellett hosszabb operandus rész,
• minimális átlagos utasításhossz: a gyakori kódok rövidek, a ritkán használtak hosszabbak.

Az **1111** kódot nem használtuk ki 3 címes utasításnak **(menekülő kód**), és ez lehetővé teszi, hogy további – igaz, nem 3 címes – utasításokat adjunk meg.
1111 1110 és **1111 1111** is menekülő kód.

Minden utasítás tartalmaz műveleti kódot. Ezen kívül tartalmazhat az operandusokra, eredményre vonatkozó információt.

Utasítás típusok:

• Regiszter - memória típusú utasítások: a regiszterek és a memória közötti adatforgalom (betöltés, tárolás). Ilyenkor egy regiszter és egy memória cím megadása szükséges a címrészen.

• Regiszter - regiszter típusú utasítások: összeadás, kivonás,…
Az eredmény is regiszterben keletkezik.
Ilyenkor három regiszter megadása szükséges a címrészen.

• …

Címzési módszerek

Három cím: *cél = forrás1 + forrás2*.

A memória sok rekeszt tartalmaz, de csak kevés regiszter van. Egy regiszter néhány bittel címezhető. Regiszterek használata rövidíti a címeket, de nyújtja a programot, ha az operandus csak egyszer kell.

A legtöbb operandust többször használjuk.

Implicit operandusok:
• **Két cím**: ***regiszter2*** = *regiszter2 + forrás1*.
• **Egy cím**: ***akkumulátor*** = ***akkumulátor*** + *forrás1*.
• **Nulla cím**: verem, pl. az **IJVM IADD** utasítása.

Operandus megadás

• **Közvetlen operandus** (immediate operand): Az operandus megadása az utasításban (**5.17. ábra**)

MOV	R1	#4

• **Direkt címzés** (direct addressing): A memóriacím megadása a címrészen. Az utasítás mindig ugyanazt a címet használja. Az operandus értéke változhat, de a címe nem (fordításkor ismert kell legyen!).

• **Regisztercímzés** (register addressing): Mint a direkt címzés, csak nem memóriát, hanem regisztert címez.

• **Regiszter-indirekt címzés** (register indirect addressing):
A címrészen valamelyik regisztert adjuk meg, de a megadott regiszter nem az operandust tartalmazza, hanem azt a **memóriacímet**, amely az operandust tartalmazza (**mutató - pointer**).
Rövidebb, és a regiszter értékének módosításával a cím változtatható.
Pl.: A 100 szóból álló **A** tömb elemeinek összeadása

két címes gépen (egy elem 4 bájt), ~ **5.18. ábra.**

```
     MOV  R1, #0          ; gyűjtsük az eredményt R1-ben
                          ; kezdetben ez legyen 0.
     MOV  R2, #A          ; R2-be töltjük az A tömb címét
     MOV  R3, #A + 400 ; a tömb utáni első cím
C:   ADD  R1, ( R2)       ; regiszter-indirekt címzés a tömb
                          ; aktuális elemének elérésére
     ADD  R2, #4          ; R2  tartalmát növeljük 4-gyel
     CMP  R2, R3          ; végeztünk?
     BLT C                ; ugrás a C címkéhez, ha nem
     . . .                ; kész az összegzés
```

• **Indexelt címzés** (indexed addressing): Egy eltolási érték (offset) és egy (index) regiszter tartalmának összege lesz az operandus címe, **5.19-20. ábra**.

```
     MOV  R1, #0      ; gyűjtsük az eredményt R1-ben,
                      ; kezdetben ez legyen 0.
     MOV  R2, #0      ; az index kezdő értéke
     MOV  R3, #400    ; a tömb mögé mutató index
C:   ADD  R1 A(R2)    ; regiszter-indirekt címzés a tömb
                      ; aktuális elemének elérésére
     ADD  R2, #4      ; R2 tartalmát növeljük 4-gyel
     CMP  R2, R3      ; végeztünk?
     BLT  C           ; ugrás a C címkéhez, ha nem
     . . .            ; kész az összegzés
```

• **Bázisindex címzés** (based-indexed addressing): Egy eltolási érték (offset) és két (egy bázis és egy index) regiszter tartalmának összege lesz az operandus címe. Ha **R5 A** címét tartalmazza, akkor

```
C: ADD R1, A(R2)        helyett a
C: ADD R1 (R2+R5)
```

utasítás is írható. Ez a módszer előnyös, ha nem csak az **A** tömb elemeit szeretnénk összegezni.

Veremcímzés (stack addressing): Az operandus a verem tetején van. Nem kell operandust megadni az utasításban.

Fordított Lengyel Jelölés
(Postfix Polish Notation - Lukasiewicz)

Postfix jelölés: a kifejezéseket olyan formában adjuk meg, hogy az első operandus után a másodikat,
majd ezután adjuk meg a műveleti jelet:

infix: *x + y*, **postfix**: *x y +*.

Előnyei: nem kell zárójel, sem precedenciaszabályok.
Jól alkalmazható verem címzés esetén.

Infix jelölés konvertálása postfix-re: algoritmusok vannak rá (pl. Dijkstra)

Pl. (**5.24. ábra**):

(8 + 2 * 5)/(1 + 3 * 2 – 4) // infix

8 2 5 * + 1 3 2 * + 4 – / // postfix

Olvassuk a formulát balról jobbra!

Ha a következő jel
• **operandus**: rakjuk a verembe,
• **műveleti jel**: hajtsuk végre a műveletet (a verem tetején van a jobb, alatta a bal operandus!).

Ortogonalitási elv: Jó architektúrában a műveleti kódok és a címzési módszerek (majdnem) szabadon párosíthatók. Utasításformáknál a műveleti kód, cél, forrás mellé ezek is (pl. eltolás) odakerülnek.

Utasítástípusok
• Adatmozgató (másoló) utasítások.
• Diadikus: **+, -, *, /, AND, OR, XOR, …**
• Monadikus: léptetés, forgatás, **INC, NEG, …**
• Összehasonlítás, feltételes elágazás:
• Eljáráshívás. Visszatérési cím: rögzített helyre (rossz),
az eljárás első szavába (jobb), verembe (rekurzív eljárásokhoz is jó).
• Ciklusszervezés (**5.30. ábra**): számláló
• Input/output (**5.31-33. ábra**):
- programozott **I/O**: tevékeny várakozás, **5.32. ábra**,
- megszakítás vezérelt **I/O**,
- **DMA I/O** (**5.33. ábra**)

Pl. **Megszakítás**
A (program) megszakítás azt jelenti, hogy az éppen futó program végrehajtása átmenetileg megszakad – a processzor állapota megőrződik, hogy a program egy későbbi időpontban folytatódhasson – és a processzor egy másik program, az úgynevezett **megszakítás kezelő** végrehajtását kezdi meg.
Miután a megszakítás kezelő elvégezte munkáját, gondoskodik a processzor megszakításkori állapotának visszaállításáról, és visszaadja a vezérlést a megszakított programnak.
Megszakítás vezérelt I/O
Bizonyos előkészületek után az eszköz megkapja a feladatát (pl. az első karakter kiírását), és a program futása folytatódik.
Ha az eszköz elkészült a feladatával, akkor beállítja a „Megszakítás engedélyezett” bitet. Ez megszakítás-kérést eredményez. A megszakítás bekövetkezésekor ez a bit törlődik.
A megszakítás-kezelő újabb feladatot ad az eszköznek (a következő karakter kiírását) mindaddig, amíg a teljes feladat el nem készült, és visszatér a megszakításból.
Az **I/O** művelet végrehajtása közben a központi egység más feladatot végezhet.

DMA (Direct Memory Access, **5.33. ábra**).
Pl. egy tömb átvitele során sok megszakítás lenne.
Jobb megoldást kínál a **CPU**-nál lényegesen egyszerűbb **DMA**.
A **DMA** önállóan végzi az eszköz figyelését és az adatok mozgatását.

A szoftver fogalma
(Emlékeztető)

A program fogalma

Szoftver: számítógépes programok és dokumentumok összefoglaló neve.

Szoftver verziója: adott szoftver azonosítója, amely jelzi egy szoftver fejlesztésének egymásutániságát.

Interaktív program: végrehajtás közben kommunikál a felhasználóval.

Programvégrehajtás:

Single task rendszerekben a felhasználó egy időben csak egy programot tud futtatni (ilyen volt pl. az MS DOS), míg a *multitask rendszerekben* lehetőség van arra, hogy a felhasználó egyszerre több programot is futtasson.

Egyszerre több program is futhat (végrehajtás alatt lehet) – *multitasking.*

Egy program befejezése előtt is átkerülhet a vezérlés egy másik programra – *megszakítás*.

Több egyszerre indítandó program közül az indul először, amelynek nagyobb a *prioritása.*

Grafikus felületű (GUI) multitasking esetén egy program *előtérben*, míg a többi *háttérben* fut.

A szoftverek fogalmi csoportosítása

- Alkalmazói programok
  Szövegszerkesztők
  Táblázatkezelők
  Adatbázis-kezelők
  Grafikai szerkesztők
  Integrált adatkezelői rendszerek
  Prezentációkészítők, pl. MS PowerPoint, Corel Presentations

- Fejlesztői környezetek

- Segédprogramok

  Víruskezelők
  Fájlkezelők
  Tömörítők
  Fordítóprogramok
  Értelmező programok (*interpreterek)*

- Rendszerprogramok

Operációs rendszer Az operációs rendszer egyik fontos feladata, hogy kapcsolatot teremt a gép és a felhasználó között.

Csoportosítás a szoftverek beszerzési és felhasználási jogosultsága alapján:

- A program használója fizet a használati jogért. Az ilyen programok módosítása, másnak továbbadása bűncselekménynek számít!
- *shareware* program, amelyet ingyen ki lehet próbálni egy meghatározott ideig, esetleg korlátozott funkciókkal. Az idő lejárta után, illetve ha a felhasználó a teljes szolgáltatási kört szeretné felhasználni, akkor ki kell fizetnie a szoftver árát, és ezzel megszerezheti a teljes használati jogot, és a teljes programot ezután kaphatja meg.
- *freeware* a készítője által szabadon hozzáférhetővé tett nyílt kódú szoftverek előnyeit a biztonság és a megbízhatóság területén
  - ✓ A *freeware* programokkal rokon, jogilag pontosan leszabályozott szoftvertípus a *Szabad Szoftver Alapítvány* (Free Software Foundation) és a GNU által létrehozott *General Public Licence* hatálya alá tartozó, úgynevezett *nyílt vagy szabad forráskódú szoftver*, amelyet szabadon lehet terjeszteni, a forráskódját szabadon meg lehet változtatni, aminek eredményeként az esetleges hibái könnyen kijavíthatók, így az ilyen szoftverek rendkívül gyorsan fejlődnek. Pl. Linux.

A legális és illegális szoftver között csak törvényi oldalról van különbség.

Cookie: a cookie egy szöveges fájl, amit egy weblaphoz köthetünk. Ezzel azonosítják a webüzemeltetők a visszatérő felhasználókat. Információkat és beállításokat tartalmazhat.

Számítógépvírusok:

A vírus ártó szándékkal írt program, amelyek a számítógép működésének vagy tárolt adatainak rongálására, megsemmisítésére lettek létrehozva, és amelyeket támadók terjesztenek, E-mailben fájlletöltés és fájlátvitel során, valamint az azonnali üzenetküldő (pl. Skype, Messenger) programokkal kerülhetnek át másik számítógépre.

Fajtái:

- Fájlvírusok: *.com, *.exe fájlokat fertőznek.
- Boot vírusok: akkor fertőznek, ha fertőzött lemezről indítjuk a gépet.
- Új típusú vírusok: kevésbé használt lemezterületre bújnak, onnan a segédprogram indításakor indítják tevékenységüket.
- Mutáló vírusok: minden fertőzés után változtatnak a víruskódon.
- Többlakú vírusok: a vírus több helyre is befészkelte magát például: .com, .exe fájlok mellett a partíciós táblát vagy a boot szektort is fertőzik.
- Dropperek: olyan programok, amelyek valamilyen hasznos program mellett (mellékesen) vírusokat is tartalmaznak. (Virusscan néven is futott ilyen vírus.)
- Vírusgyártó automaták: olyan program, amellyel szint bárki képes vírust gyártani.
- Trójai programok: a program nem az, aminek mutatja magát.
- ANSI bombák: az ANSI.SYS fájlt fertőzik. Átprogramozzák a billentyűzetet. (ENTER = format C)
- Makró vírusok: a Word és Excel programok dokumentumainál használt makrókat fertőzik meg.

Férgek: a féreg önmagát sokszorozó program, ami a hálózatokra veszélyes. A féreg a hálózatot használja, hogy átmásolja magát más hálózatokon lévő állomásokra, gyakran a felhasználó beavatkozása nélkül. A számítógépen is saját programjukat másolják tovább, így foglalják el a memóriát és az erőforrásokat.