



TÁMOP-4.1.1.C-12/1/KONV-2012-0004

SZÁMÍTÓGÉP ARCHITEKTÚRÁK DIGITÁLIS TANANYAG

Nagy Antal, Tanács Attila

Szegedi Tudományegyetem
Informatikai Tanszékcsoport

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

BEVEZETÉS

Strukturált számítógép-felépítés,
mérföldkövek a számítógépek felépítésében

Digitális számítógép

- **Utasításokat hajt végre**

- **PROGRAM**

- Utasítássorozat, amely megadja az adott problémát megoldó algoritmust
- Akár egy recept
 - Hozzávalók
 - Az étel elkészítésének a lépései

- **Elektronikus áramkörök**

- Egyszerű utasítások, korlátozott halmaza
 - Ezeket az utasításokat kell használni
 - Nem bonyolultak
 - Adj össze két számot
 - Hasonlítsd össze a számot nullával
 - ...



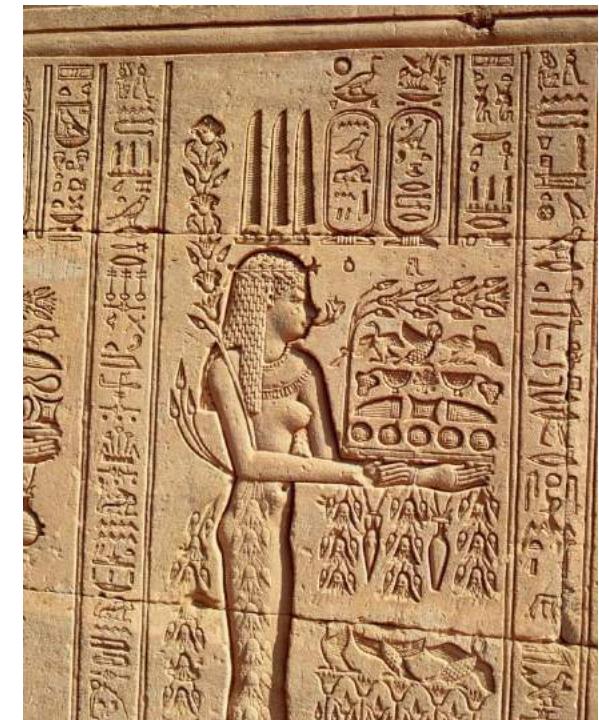
Digitális számítógép

• Gépi nyelv

- Adott számítógép utasításainak halmaza
- A felhasználó a számítógéppel tud kommunikálni
- Elemi utasítások
 - A lehető legegyszerűbbek
 - Elektronikus áramkörök
 - Bonyolultság
 - Ár
 - A felhasználók számára
 - Nehézkes, bonyolult

• Absztraktiós szintek

- Strukturálás
- Szintek egymásra épülése
 - Architektúra = építészet



Strukturált számítógép-felépítése

- **Nyelvek, szintek és virtuális gépek**
 - A felhasználó számára kényelmesebb utasításhalmaz biztosítása
 - Az új utasítások egy nyelvet alkotnak (Formális nyelvek)
 - Nevezzük L_1 -nek
 - A gépi nyelvet L_0 -val jelöljük
 - Kérdés: hogyan hajtja végre a számítógép az L_1 nyelven írt programokat?
 - Csak az L_0 nyelven leírt programokat képes végrehajtani



Nyelvek, szintek és virtuális gépek

Fordítás

- Az L_1 nyelvű program minden utasítását helyettesítjük
 - L_0 nyelv utasításainak ekvivalens sorozatával
 - Új L_0 nyelvű program
 - Csak L_0 nyelv utasításaiból áll
 - L_1 nyelvű program helyett az új programot hajtja végre a számítógép
- **Fordító**
 - Compiler/assembler

„Szakfordító”

Értelmezés

- L_0 nyelvű program
 - Bemenete az L_1 nyelvű program
 - minden utasítást elemez
 - Ekvivalens, L_0 nyelvű utasítássorozat
 - Azonnal végrehajtja
- Értelmező
 - Interpreter

„Szinkrontolmács”

Nyelvek, szintek és virtuális gépek

Fordítás

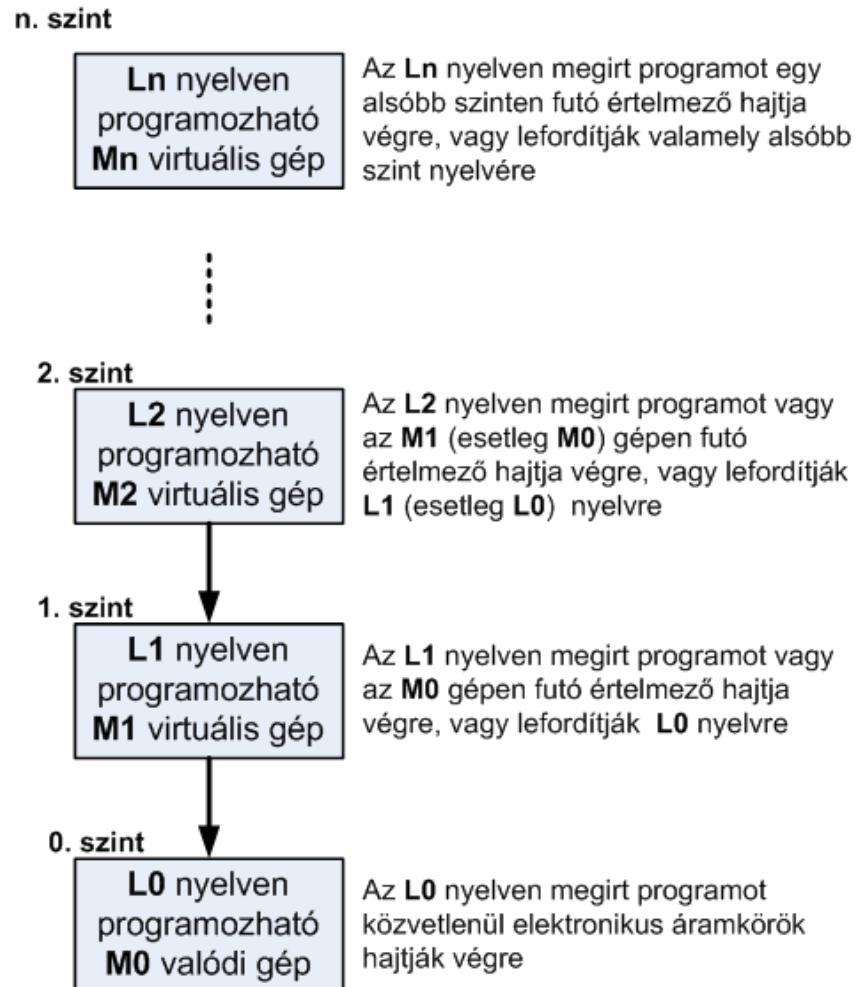
- Az egész L_1 programot átírjuk L_0 utasításaiból álló sorozattá
 - L_1 programot eldobjuk
 - L_0 -t hajtja végre a számítógép

Értelmezés

- Nem keletkezik lefordított program
- A számítógépet az értelmező vezérli
 - L_1 nyelvű program alapján

Nyelvek, szintek és virtuális gépek

- **M_0 virtuális gép**
 - Gépi nyelve L_0
- **M_1 virtuális gép**
 - Gépi nyelve L_1
- **Fordítás/értelmezés**
 - Programokat írhatunk virtuális gépekre
 - Az L_0 és L_1 nem különbözhet nagyon
 - Még nem értük el a célunkat
 - M_2 virtuális gép
 - Gépi nyelve L_2
 - ...
- **Nyelvek sorozata**
 - Megelőzőnél kényelmesebb használat



Nyelvek, szintek és virtuális gépek

- **Nyelvek és virtuális gépek közötti kapcsolat**

- minden gépnél van saját gépi nyelv
 - Gép által végrehajtható utasítások együttese
- Virtuális gép elektronikus áramkörökből való megépítése
 - Drága lehet
 - Nem lehetetlen
- Költséghatékonyaság
 - Mi hajtja végre az utasítást?
 - Fordítás/értelmezés
 - Elektronika teljesíti



Korszerű többszintű számítógépek

5. Problémaorientált nyelv szintje
fordítás (fordítóprogram)
 4. Assembly nyelv szintje
fordítás (assembler)
 3. Operációs rendszer gép szintje
részben értelmezés (op. rendszer)
 2. Utasításrendszer-architektúra szintje
értelmezés (mikroprogram) vagy
közvetlen végrehajtás (elektronikus áramkörök)
 1. Mikroarchitektúra szintje
hardver
 0. Digitális logika
-
- 1. Eszközsint
elektronikai tervezés (tranzisztorok)

Korszerű többszintű számítógépek

0. Digitális logika szintje

- Kapuk
 - Analóg alkatrészek
 - Pl. tranzisztorok
 - Egy vagy több bemenete van
 - 0 vagy 1 értéket reprezentáló jelek
 - Kimenetként egyszerű függvényeket számolnak ki
 - Pl. AND vagy OR
- Kapukból előállítható egy 1 bites memória
 - 16-os, 32-es vagy 64-es csoportokból regiszterek
 - Bináris szám tárolása adott értelmezési tartományon

1. Mikroarchitektúra szintje

- Aritmetikai-logikai egység
 - 8-32 elemű, lokális memóriakészletként használt regiszterkészlet
 - Egyszerű matematikai műveletek elvégzése
 - Kapcsolódó regiszterek
- Adatút
 - Adatok áramlása
 - Regiszterek kiválasztása
 - Művelet elvégzése (ALU)
 - Eredmény tárolása valamelyik regiszterben
 - Vezérlés
 - Mikroprogram (értelmező)
 - Közvetlenül a hardver

Korszerű többszintű számítógépek

2. Utasításrendszer-architektúra szintje

- Instruction Set Architecture (ISA)
- Gépi nyelv referencia kézikönyv
 - Utasításrendszer leírása
 - Mikroprogram
 - Hardver-végrehajtó áramkör

3. Operációs rendszer gép szintje

- Kevert szint
- Utasítások többsége az ISA szinten is megvan
- Új utasítások, szolgáltatások
 - Memóriaszervezés
 - Több program futtatása
 - Kvázi párhuzamosság
 - Örökölt ISA szintű utasítások végrehajtása ISA szinten

Korszerű többszintű számítógépek

4. Assembler nyelv szintje

- Fordító
- Alsóbb szintek gépi nyelvei numerikusak
 - Számsorozatok
- Felhasználó számára érthető
 - Szavak, rövidítések
- Alsóbb szintekhez tartozó nyelvek szimbolikus formája
 - 1., 2. vagy 3. szintekre való program írás

5. Problémaorientált nyelv szintje

- Magas szintű nyelvek
 - C, C++, JAVA, LISP, Prolog, ...
- Fordítóprogramok
 - 3-as vagy 4-es szintű nyelvekre fordítanak
- Értelmezők
 - Pl. Java ISA szerű nyelvre fordítják
 - JAVA bájtkódra
 - Értelmező hajtja végre

Korszerű többszintű számítógépek

- Egy szint architektúrája
 - Adattípusok
 - Műveletek
 - Szolgáltatások
- Látható tulajdonságok egységbe foglalása
- Számítógépes rendszerelemek tervezése
 - Számítógépes architektúra
 - Számítógépek felépítése



A többszintű számítógépek fejlődése

- Hardver
 - Elektronikus áramkörök
 - Memória
 - Bemeneti és kimeneti eszközök
- Szoftver
 - Algoritmusok
 - Programok
 - Algoritmusok számítógépes reprezentációja
 - Utasítássorozat



A többszintű számítógépek fejlődése

*A hardver és a szoftver
logikailag ekvivalens*

A mikroprogramozás feltalálása

- 1940-es években 2 szintű gépek
 - ISA
 - Programok írása
 - Digitális logika
 - Programok végrehajtása
 - Bonyolult, nehezen áttekinthető áramkörök
 - Megbízhatatlanok
- 1951 - Maurice Wilkes
 - Beépített értelmező
 - Mikroprogram
 - ISA szintű programok végrehajtása
 - A hardver kis utasításkészletű programot hajt végre az ISA szintű programok helyett
 - Vákuumcsövek számának csökkenése
 - Megbízhatóság növelése
- 1970-re uralkodó elv
 - ISA-szint értelmezése mikroprogrammal
 - NEM közvetlen elektronikus megvalósítással

Az operációs rendszer feltalálása

- A korai években a számítógépet a programozó kezelte
 - Adott időbeosztás alapján
 - Lyukkártya, ceruza
 - Ismernie kellett a számítógép működését
 - Nem voltak kihasználva a számítógépek
 - Gyakran várakoztak/álltak
- 1960: a gépkezelői feladatok automatizálása
 - Operációs rendszer
 - Vezérlő kártyák csatolása a programokhoz
 - Fortran Monitor System (IBM 709)
 - Új virtuális gép fejlesztésének első lépése is
 - Új utasítások, szolgáltatások és tulajdonságok kerültek az ISA szint felé

Az operációs rendszer feltalálása

- Új utasítások
 - Operációs rendszeri makrutasítások
 - Felügyelő/supervisor hívások
 - Mai néven rendszerhívás
- Kötegelt rendszerek
- Időosztásos rendszerek
 - Multiuser – multitasking
- ISA szinten nem létező utasítások és tulajdonságok az érdekesek

Szolgáltatások átterelése mikroprogram szintjére

- Új utasítások létrehozása a mikroprogramok bővítésével
 - Hardver bővítése programozással
- Meglévő utasításokkal is megvalósíthatóak lettek volna
 - INC \Leftrightarrow ADD
- Nagyobb sebességet értek el
 - INC kicsit gyorsabb volt, mint az ADD
- Új utasítások
 - Egészek szorzása/osztása
 - Lebegőpontos aritmetika
 - Eljárást hívó/visszatérő
 - Ciklusokat gyorsító
 - Karaktersorozatokat kezelő

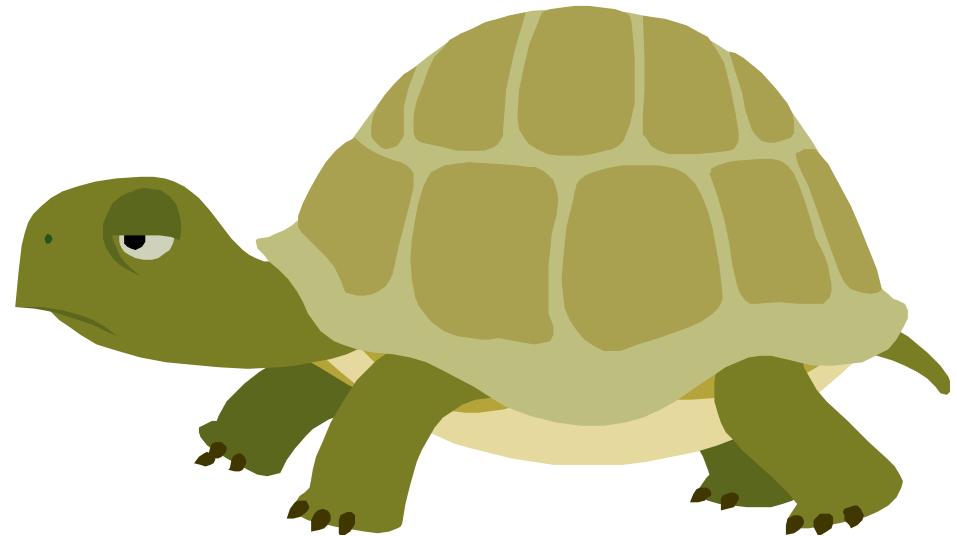
Szolgáltatások átterelése mikroprogram szintjére

- Új tulajdonságok
 - Tömbökkel való számítások felgyorsítása
 - Programok memóriában való áthelyezése
 - Megszakítások rendszere
 - Jelzések a bemeneti és kimeneti műveletek befejezéséről
 - Program felfüggesztése – másik program folytatása
 - Speciális utasítások
 - Hang-, video-, multimédia-fájlok feldolgozására



Mikroprogramok száműzése

- A mikroprogramok meghíztak
 - Lassabbak lettek
- Gépek felgyorsítása
 - Utasításkészlet csökkentése
 - Megmaradó utasítások közvetlen végrehajtása
- Hardver fejlődése
 - Pl. videokártyák



Mérföldkövek a számítógépek felépítésében

Év	Név	Készítette	Megjegyzés
1834	Analitikus gép	Babbage	Első kísérlet digitális számítógép építésére
1931	Z1	Zusse	Első jelfogós számológép
1943	COLOSSUS	Brit kormány	Első elektronikus számítógép
1944	Mark I	Aiken	Első amerikai általános célú számítógép
1946	ENIAC I	Eckert/Mauchley	Mai számítógépek történetének kezdete
1949	EDSAC	Wilkes	Első tárolt programú számítógép
1951	Whirlwind	M.I.T.	Első valós idejű számítógép
1952	IAS	Neumann	Legtöbb mai gépeknél alkalmazott felépítés
1960	PDP-1	DEC	Első miniszámítógép
1961	1401	IBM	Vállalati kisgép
1962	7094	IBM	Tudományos számításokra alkalmazták
1963	B5000	Borroughs	Magas szintű nyelvre tervezett gép
1964	360	IBM	Számítógépcsalád sorozat

Mérföldkövek a számítógépek felépítésében

Év	Név	Készítette	Megjegyzés
1964	6600	CDC	Első tudományos szuperszámítógép
1965	PDP-8	DEC	Első miniszámítógép 50 000 eladott példánnnyal
1970	PDP-11	DEC	70-es évek uralkodó miniszámítógépe
1974	8080	Intel	Első általános célú egylapkás 8 bites számítógép
1974	CRAY-1	Cray	Első vektoros szuperszámítógép
1978	VAX	DEC	Első 32 bites szuper-miniszámítógép
1981	IBM PC	IBM	Személyi számítógépek korszakának elindítója
1981	Osborne-1	Osborne	Első hordozható számítógép
1983	Lisa	Apple	Első grafikus felhasználói felületű személyi szg.
1985	386	Intel	Pentium vonal első 32 bites előfutára
1985	MIPS	MIPS	Első piaci RISC-munkaállomás

Mérföldkövek a számítógépek felépítésében

Év	Név	Készítette	Megjegyzés
1987	SPARC	Sun	Első SPARC alapú RISC munkaállomás
1990	RS6000	IBM	Első szuperskaláris számítógép
1992	ALPHA	DEC	Első 64 bites személyi számítógép
1993	Newton	Apple	Első kézi számítógép
2001	POWER4	IBM	Első két-magos multiprocesszor

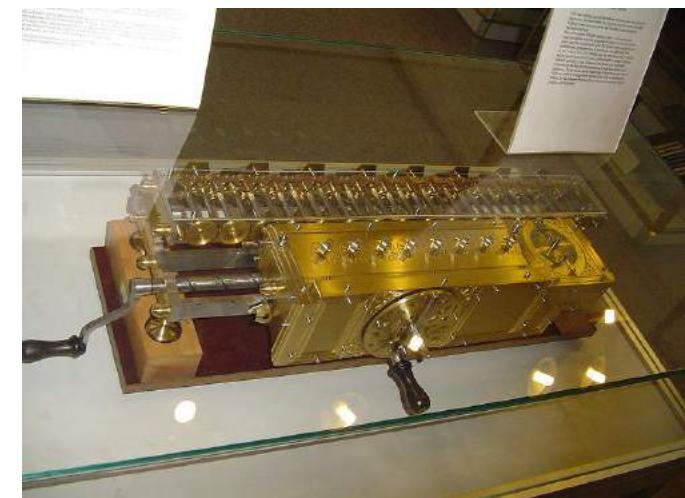
Nulladik generáció: mechanikus számológépek (1642 – 1945)

- Blaise Pascal
 - 1642
 - Összead és kivon
 - Fogaskerekek
 - Kézi forgatókar



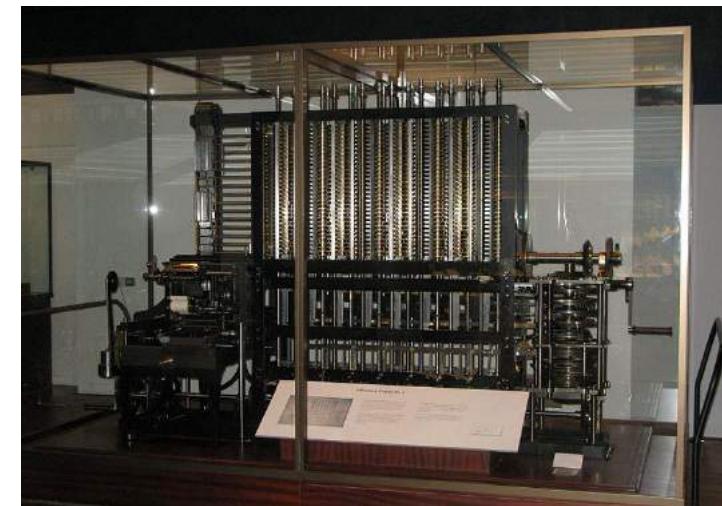
Forrás: Wikipedia

- Gottfried Wilhelm von Leibniz
 - Kb. 30 év múlva
 - Osztani és szorozni is tud
- 150 évig nem történik semmi



Nulladik generáció: mechanikus számológépek (1642 – 1945)

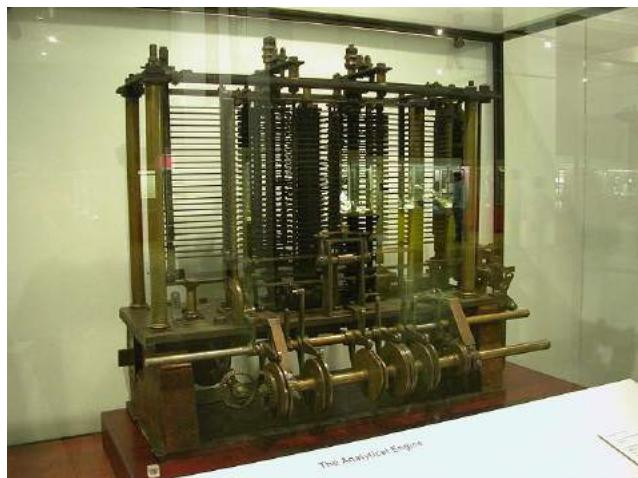
- Charles Babbage
 - 1834
 - Differencia gép
 - Összeadás, kivonás, kimenő adatok rézbevonatú lemezbe lyukasztotta acél formanyomóval
 - Polinomokra alkalmazott véges differenciák módszerének végrehajtására



Forrás: Wikipedia

Nulladik generáció: mechanikus számológépek (1642 – 1945)

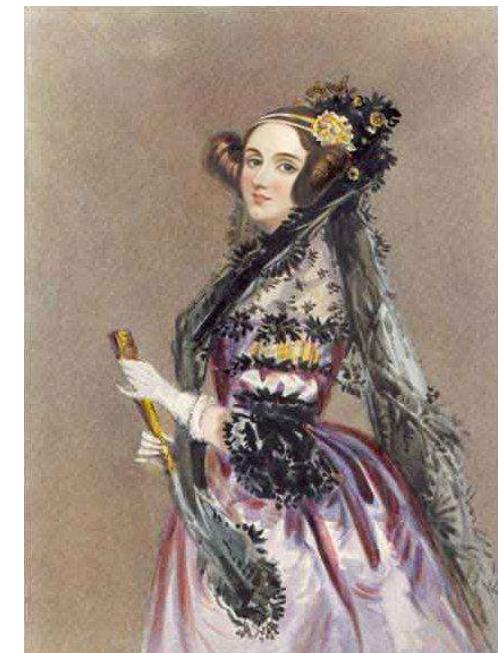
- Charles Babbage
 - Analitikus gép
 - Tároló (memória), malom (számolóegység), bemeneti (lyukkártya-olvasó) és kimeneti (lyukkártya és nyomtatott papír) rész
 - Általános célú
 - A kor technológiai szintjén nem volt biztosítható a megbízható működés



Forrás: Wikipedia



Az első programozó



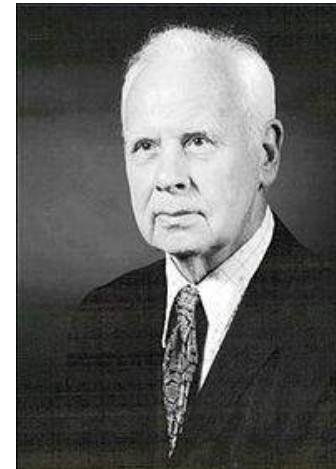
Ada Augusta Lovelace
Lord Byron lánya

Nulladik generáció: mechanikus számológépek (1642 – 1945)

- Konrad Zuse
 - 1930-as évek végén
 - Elektromágneses jelfogók
 - Z1, Z2, Z3 és Z4
 - Berlin bombázásakor megsemmisülnek
- John Atanasoff
 - Bináris aritmetika
 - Kondenzátorok kisülése
 - Memória frissítés
 - DRAM hasonló elven működik
 - Nem készül működőképes változat
- George Stibitz
 - 1940 bemutatta a működőképes gépet
 - Egyszerűbb volt Atanasoffénál



Forrás: Wikipedia

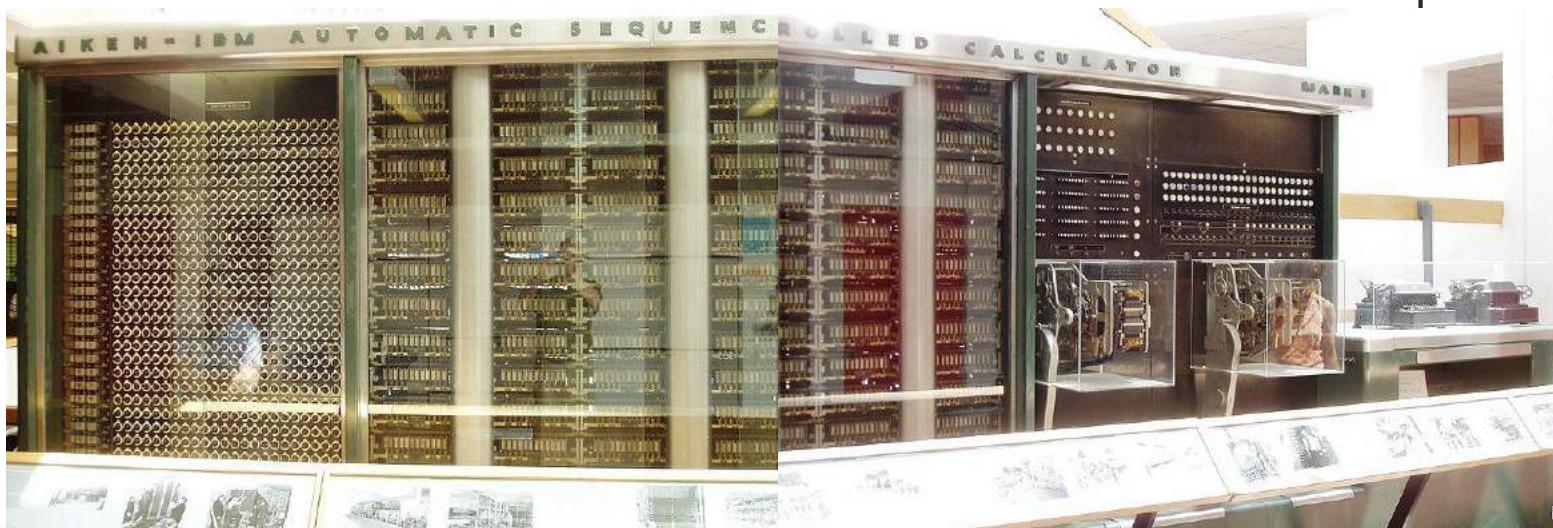


Nulladik generáció: mechanikus számológépek (1642 – 1945)

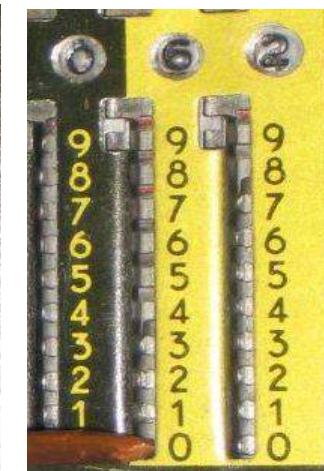
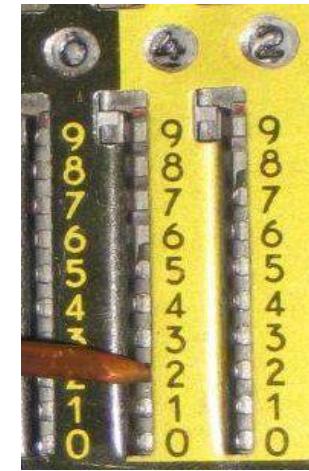
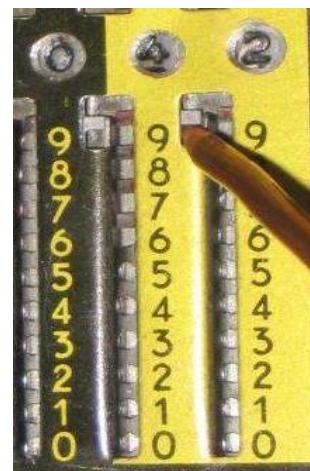
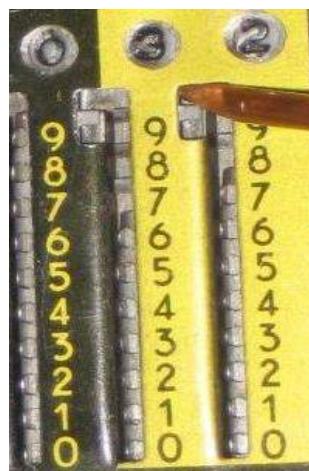
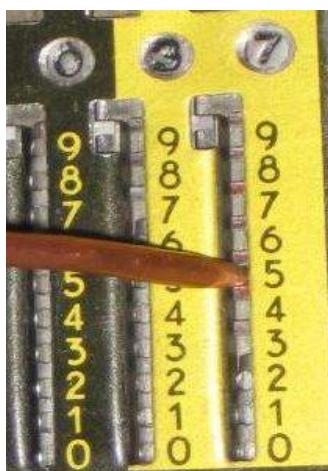
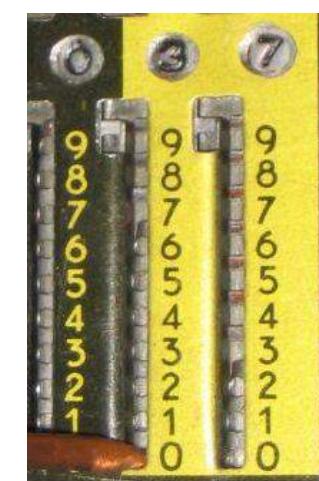
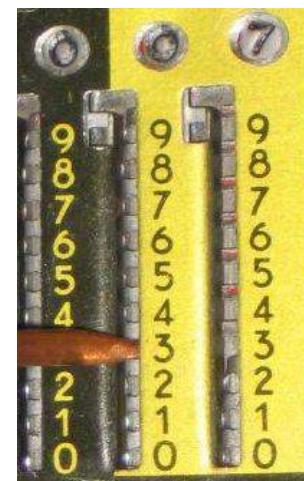
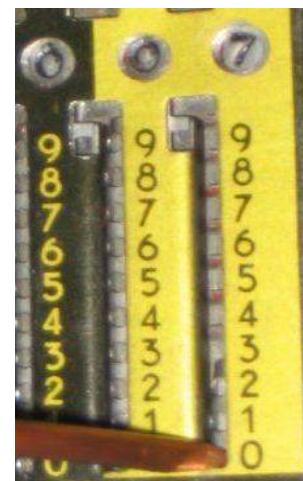
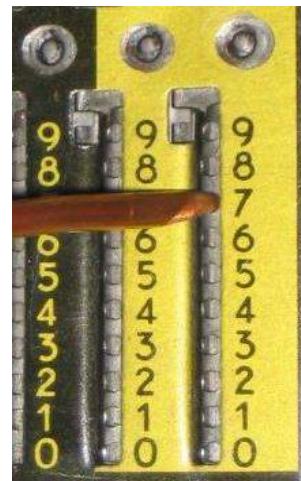
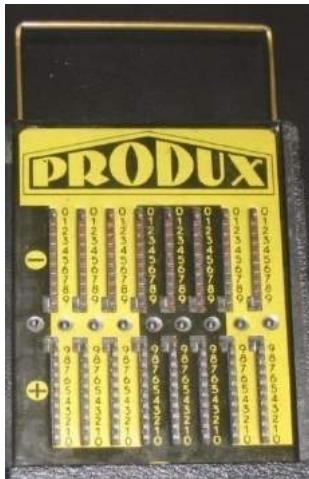
- Howard H. Aiken
 - 1944 Mark I
 - ~ Babbage gépe jelfogókból megépítve
 - 72 db 23 decimális jegyű szó
 - 1 utasítás 6 mp
 - Lyukszalag
 - Mark II



Forrás: Wikipedia

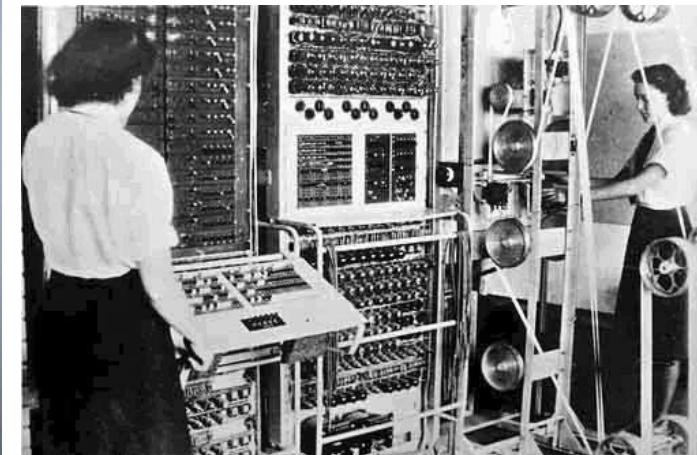


Nulladik generáció: mechanikus számológépek (1642 – 1945)



Első generáció: vákuumcsövek (1945 – 1955)

- II. világháború
 - Kódolt utasítások
 - ENIGMA
 - Gyors számításokra
 - 1943
 - COLOSSUS
 - Alan Turing
 - 30 évre titkosították



Forrás: Wikipedia

- Irányzéktáblázatok
 - Megbízható nők százai kézi számológépekkel
 - John Mauchley és doktorandusza J. Presper Eckert
 - 1943 **Electronic Numerical Integrator And Computer** (ENIAC)
 - 18 ezer vákuumcső, 1 500 jelfogó, súlya 30 tonna, 140 kilowatt
 - 20 db 10 jegyű decimális szám tárolására alkalmas regiszter
 - Programozása: 6 000 darab többállású kapcsoló
 - 1946-ig nem fejezték be
 - Nyári egyetem

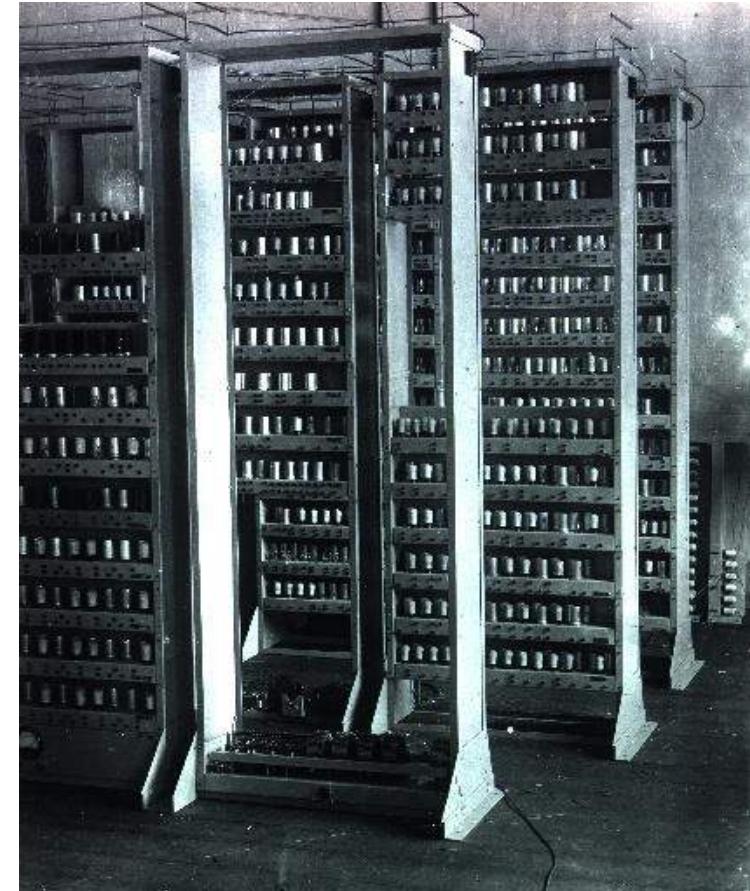
ENIAC



Forrás: Wikipedia

Első generáció: vákuumcsövek (1945 – 1955)

- Maurice Wilkes
 - 1949
 - EDSAC
 - Electronic Delay Storage Automatic Calculator
 - John von Neumann
 - First Draft of a Report on the **EDVAC**
 - Első működőképes
- Rand Corporation
 - JOHNIAC
- University Illinois
 - ILLIAC
- Los Alamos Laboratory
 - MANIAC
- Weizmann Institute
 - WEIZAC



Forrás: Wikipedia

Első generáció: vákuumcsövek (1945 – 1955)

- Eckert és Mauchley
 - Electronic Discrete Variable Automatic Computer
 - A projekt elvérzett

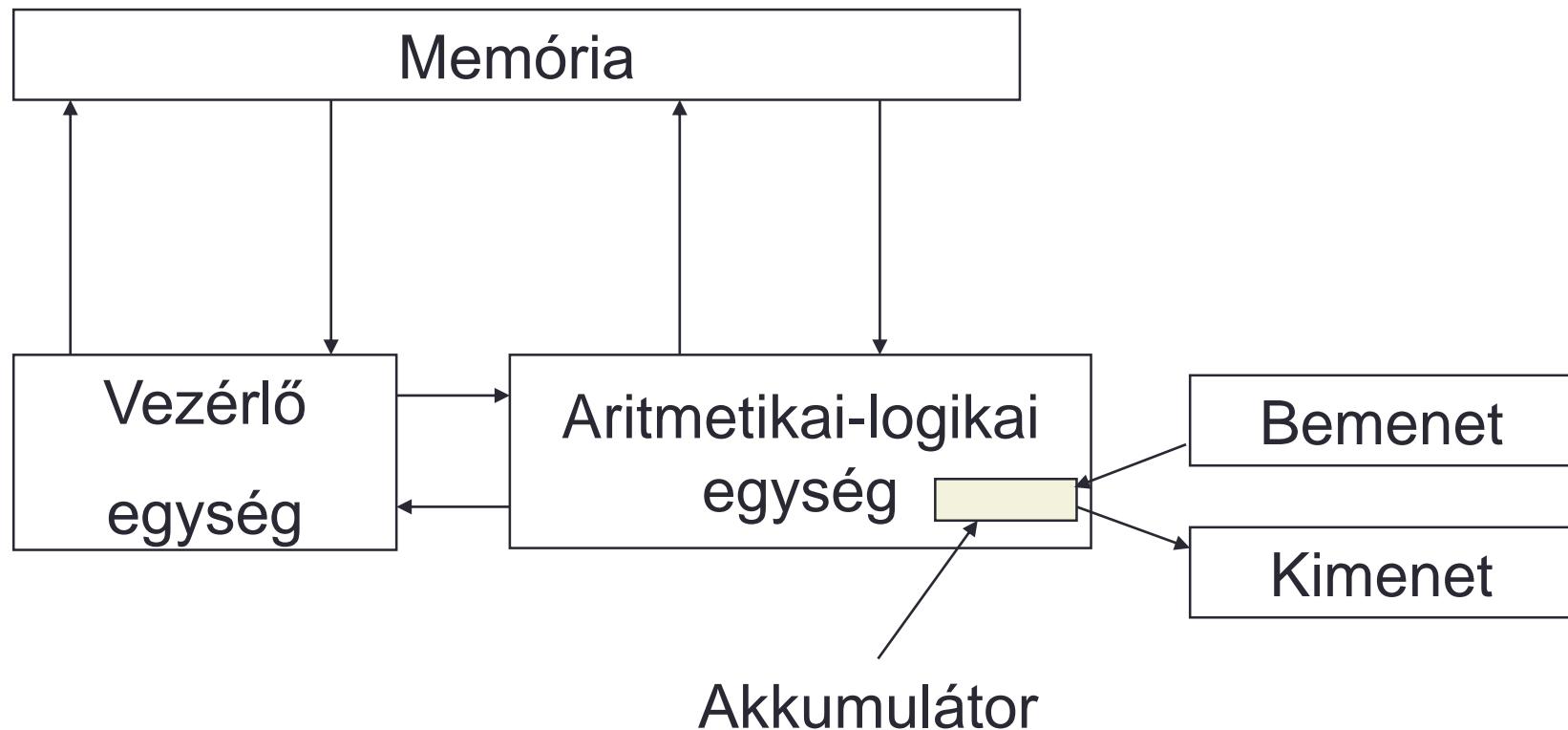
Forrás: Wikipedia

- Neumann János
 - ENIAC projekt egyik résztvevője
 - Saját EDVAC-változat
 - IAS gép
 - Program tárolása digitális formában a memóriában
 - Tárolt program
 - Párhuzamos bináris aritmetika
 - ENIAC: 1 bekapcsolt 9 kikapcsolt vákuumcső
 - Atanasoff



Első generáció: vákuumcsövek (1945 – 1955)

- Neumann-gép



Első generáció: vákuumcsövek (1945 – 1955)

- 1953
 - IBM 701
- 1958
 - Legnagyobb első generációs
 - IBM 709
- 1963
 - Szeged
 - M3

Forrás: Wikipedia



Második generáció: tranzisztorok (1955 – 1965)

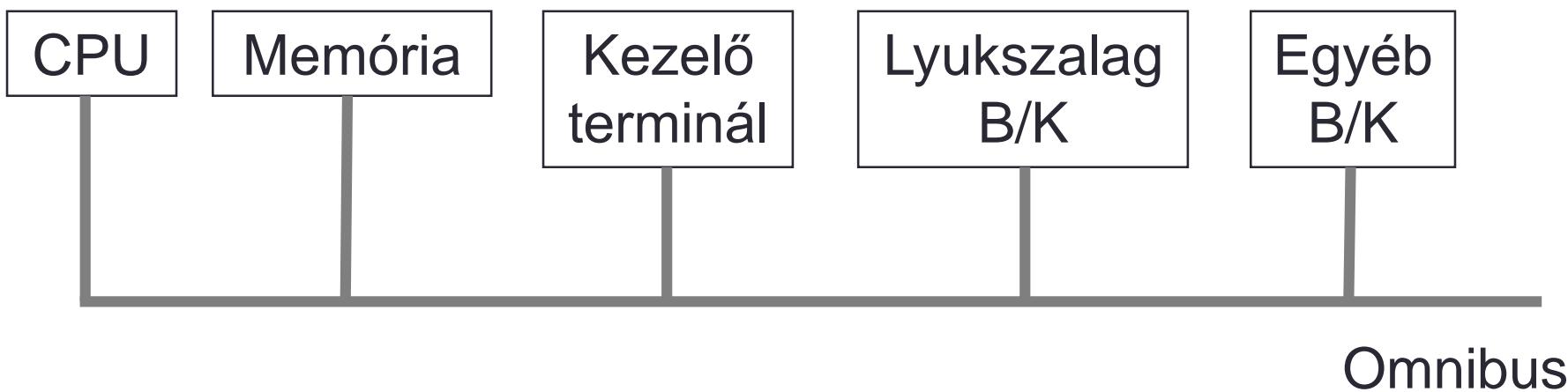
- 1948
 - Tranzisztorok feltalálása
 - 1956 fizikai Nobel-díj
- DEC PDP-1
 - 1961
 - 4096 db 18 bites szó
 - 200 000 utasítás mp-ként
 - Viszonylag olcsó volt
 - 120 000 \$
 - Teljesítménye fele volt a leggyorsabb gépnak
 - IBM 7090
 - Több millió dollár



Forrás: Wikipedia

Második generáció: tranzisztorok (1955 – 1965)

- DEC PDP-8
 - 12 bites
 - Omnibus sín
 - Többszörös vezeték
 - Különböző egységek összekapcsolása
 - 16 000 \$
 - 50 000 eladott példány



Második generáció: tranzisztorok (1955 – 1965)

- Magyarországon
 - KFKI, TPAI
- Tudományos számítások
 - IBM 7090, 7094
- Üzleti adatok tárolása
 - IBM 1401
 - Olcsóbb
 - Közelítő sebesség IBM 7094
 - Mágnesszalag olvasás/írás
 - Lyukkártya olvasás/lyukasztás
 - Nyomtatás



Forrás: <http://hampage.hu/tpa/tpai.html>

Második generáció: tranzisztorok (1955 – 1965)

- 1964
 - Control Data Corporation (CDC) 6600
 - Egy nagyságrenddel gyorsabb, mint a 7094
 - Nagyfokú párhuzamosság
 - Funkcionális egységek
 - Több kisebb segédszámítógép beépítése
 - Feladatszervezés
 - Bemenet/kimenet kezelése

Forrás: Wikipedia



Második generáció: tranzisztorok (1955 – 1965)

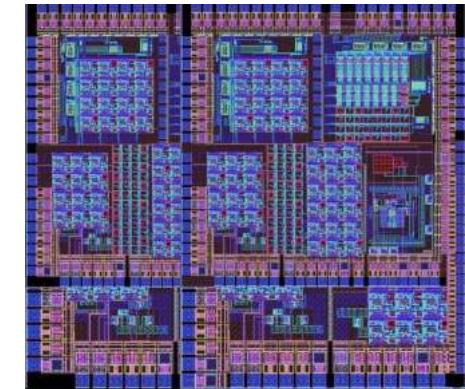
- Seymour Cray
 - CDC 6600 tervezője
 - Gyors számítógépek tervezése
 - Szuperszámítógépek
 - 6600, 7600
 - Cray-1



Forrás: Wikipedia

Harmadik generáció: integrált áramkörök (1965 – 1980)

- Robert Noyce
 - 1958 – szilikon integrált áramkör
 - Egyetlen lapkán több tucat tranzisztor elhelyezése
 - Kisebb, gyorsabb és olcsóbb számítógépek építése
- IBM – 1964
 - 7094 és 1401 probléma
 - Nem voltak kompatibilisek
 - Két különálló programozási részleg
 - Egyetlen termék
 - System/360
 - Különböző modellek (gépcsalád)
 - 20, 30, ..., 75
 - Multiprogramozás
 - Egyidejűleg több program van a gép memoriájában



Forrás: Wikipedia



Harmadik generáció: integrált áramkörök (1965 – 1980)

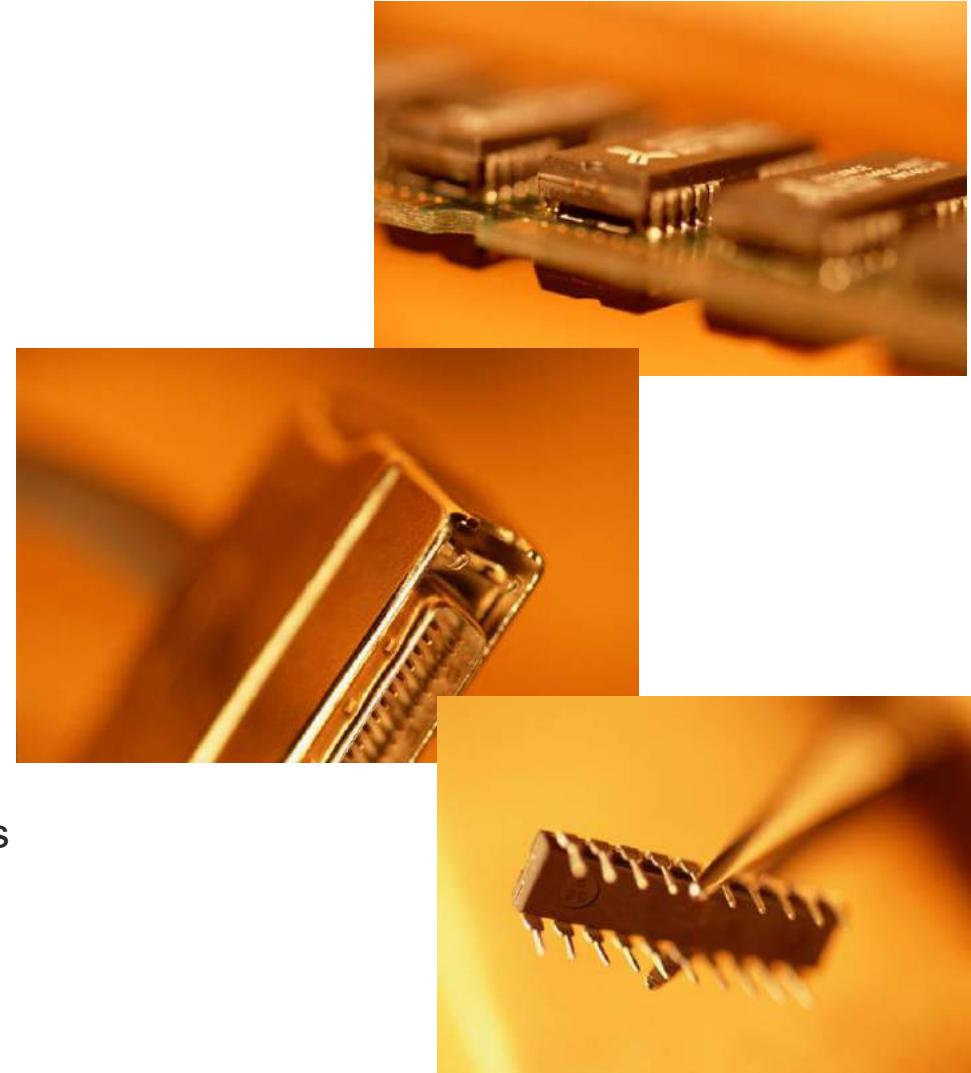
Tulajdonság	Model 30	Model 40	Model 50	Model 65
Relatív teljesítmény	1	3,5	10	21
Ciklus idő (ns)	1000	625	500	250
Maximális memória (KB)	64	256	256	512
Ciklusonként elérhető bájt	1	2	4	16
Adatcsatornák max. száma	3	3	4	6

Harmadik generáció: integrált áramkörök (1965 – 1980)

- System/360
 - Multiprogramozás
 - Egyidejűleg több program van a gép memóriájában
 - Más gépeket tudott emulálni bináris szinten
 - A kisebbek a 1401-est, a nagyok a 7094-est
 - Mikroprogramozott utasításrendszerek
 - 360-as saját
 - 1401-es
 - 7094-es
 - Párhuzamos bináris és soros decimális aritmetika probléma
 - Bináris aritmetika 16 db 32 bites regiszter
 - A memória bájt szervezésű maradt (1401-eséhez hasonlóan)
 - 2^{24} bájtos címtartomány
 - 1980 memória korlát
 - Kompatibilitás feladása
 - 2^{32} bájtos memóriaméret 32 bites címzési módszer

Negyedik generáció: Magas integráltságú áramkörök (1980 -)

- Very Large Scale Integration
- Több millió tranzisztor elhelyezése egy lapkán
- Kisebb és gyorsabb számítógépek
- Számítóközpont
 - Nagy gépek üzemeltetéséhez
- Miniszámítógépek
 - Árak csökkenése
 - Személyi számítógép
 - Alkatrész
 - Egy nyomtatott áramköri kártya
 - Zacskónyi lapka (Intel 8080)
 - Kábelek, tápegységek és 8 inches floppy meghajtó
 - CP/M
 - Gary Kildall



Negyedik generáció: Magas integráltságú áramkörök (1980 -)

- Személyi számítógépek
 - Steve Jobs, Steve Wozniak
 - Apple, Apple II
- Philip Estridge
 - IBM alkatrészekből
 - Intel 8088
 - Personal Computer (PC)
 - 1981-ben kezdték el a gyártást
 - A terveket nem szabadalmazták
 - 49\$-os könyvben leközölték
 - IBM PC klónok
 - Lényegesen olcsóbban



Forrás: Wikipedia



Negyedik generáció: Magas integráltságú áramkörök (1980 -)

- Commodore, Apple, Amiga és Atari
 - Nem Intel CPU
 - IBM PC-ipar súlya összeroppantotta őket
- Apple Lisa
 - Elbukott – túl drága volt
- Apple Macintosh
 - GUI
- Microsoft Corporation
 - MS-DOS
 - OS/2
 - Windows

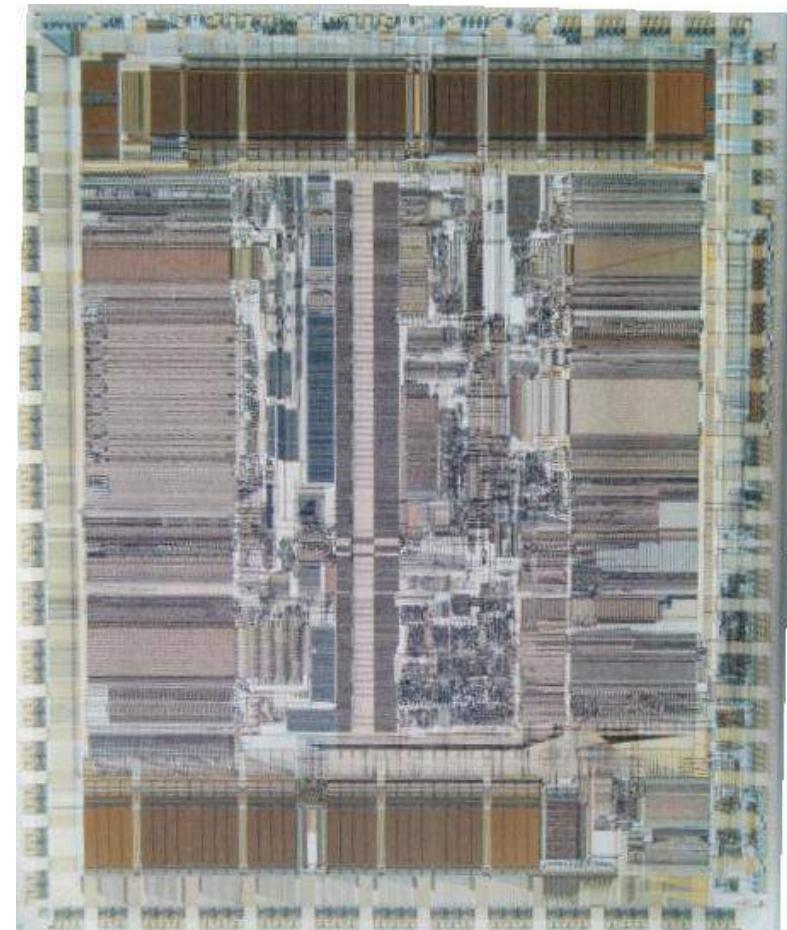


Forrás: Wikipedia



Negyedik generáció: Magas integráltságú áramkörök (1980 -)

- 1985
 - Intel 386 – az első „pentium”
- 1980-as évek közepe
 - RISC
- 1992
 - DEC 64 bites Alpha
 - RISC



Forrás: Wikipedia

Ötödik generáció: láthatatlan számítógépek

- 1981
 - Japán kormány 500 millió dollár
 - Kudarc
- 1993
 - Apple Newton
 - Hordozható
 - Kézírás bevitelre
 - Personal Digital Assistant
- 2007
 - Modern okostelefonok megjelenése
- Láthatatlan számítógépek
 - Órák, bankkártya, háztartási gépek, ...
 - Beágyazott számítógépek
 - mindenütt jelenlévő számítástechnika



Forrás: Wikipedia

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Technológiai és gazdasági mozgatórugók

Számítógépcsaládok

Neumann-elvű gép

Processzorok

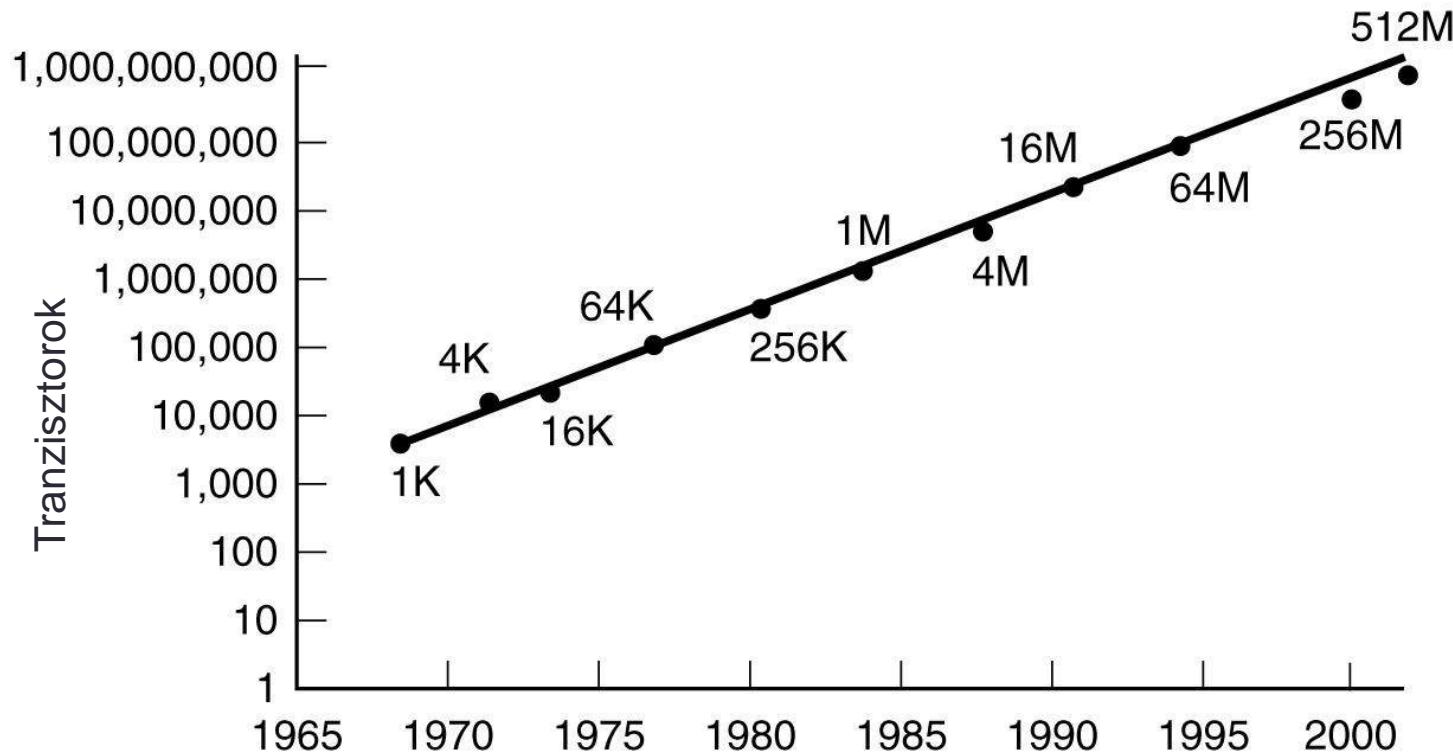
Korszerű számítógépek tervezési elvei

Nagy Antal, Tanács Attila

Technológiai és gazdasági mozgatórugók

- A gyártók egyre több tranzisztorot képesek egy lapkára integrálni
- Gordon Moore
 - Intel egyik alapítója
 - „*hogy ha a repülőgép-technológia olyan gyorsan fejlődött volna, mint a számítógép-technológia, egy repülőgép 500\$-ba kerülne, 20 perc alatt kerülné meg a földet 5 gallon (kb. 18,5 liter) üzemanyagot fogyasztva, és akkora lenne, mint egy cipős doboz*”
 - Három évente új memória áramkör generációk jelennek meg
 - minden generáció megnégyszerezte az előző memóriakapacitását
 - Állandó ütemben nőtt az elhelyezhető tranzisztorok száma
 - Moore-szabály
 - 18 havonta megkétszereződik a tranzisztorok száma
 - Három évtizede teljesül

Technológiai és gazdasági mozgatórugók



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Technológiai és gazdasági mozgatórugók

- Moore szabálya
 - Nem törvény
 - Még egy évtizedig érvényben marad
 - Fizikai korlátok
- Nathan Myhrvold első törvénye
 - „A szoftver gáz, amely kitölti a rendelkezésre álló teret”

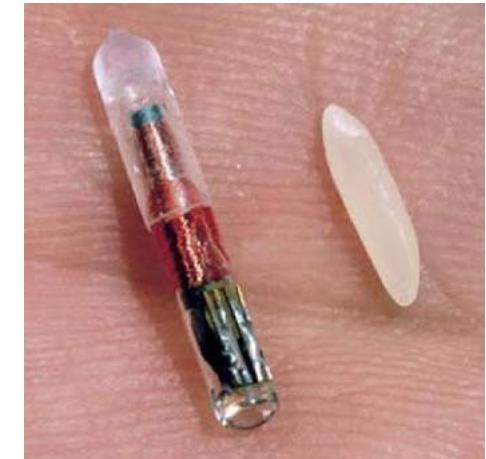


Számítógépek termékskálája

Típus	Ár (US \$)	Felhasználható például
Eldobható	0,5	Üdvözlőlapok, RFID (Radio Frequency IDentification), NFC (Near Field Communication)
Mikrovezérlő	5	Órák, autók, eszközök
Játék	50	Videojátékok
Személyi számítógép	500	Asztali/hordozható
Szerver	5 000	Hálózati szerver
Munkaállomás-gyűjtemény (COW)	50 000-500 000	Tanszéki mini-szuperszámítógép
Nagyszámítógép	5 000 000	Időjárás előrejelzés, ...

Eldobható számítógépek

- **Rádiófrekvenciás azonosító (RFID)**
 - Áramforrás nélkül
 - Rádióadóvevő
 - Egy beépített 128 bites szám
 - Külső antennáról jövő impulzus hatására visszasugározza a számot
 - Felhasználás
 - Önkiszolgáló bolt
 - Kis állatok azonosítása
 - Útdíj elektronikus beszedése
 - <http://www.rfid.org/>
- Adatvédelmi kérdések!
 - Richard Stallman
 - GNU General Public License



Forrás: Wikipedia



Eldobható számítógépek

- **NFC (Near Field Communication)**

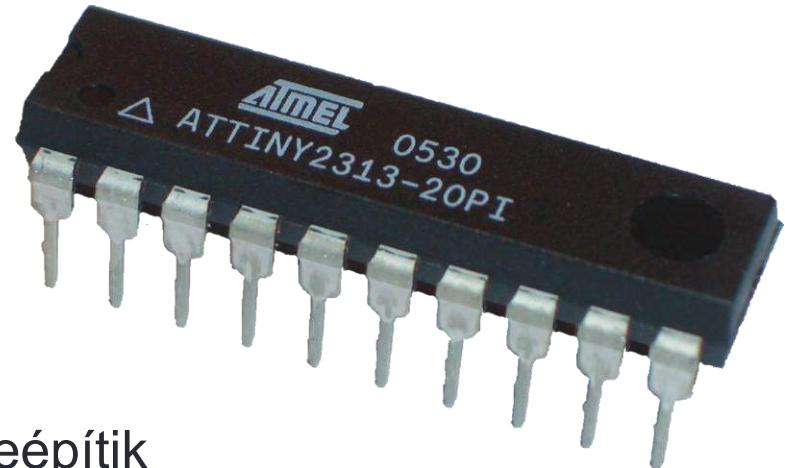


- RFID továbbfejlesztett változata
- Tárgyak „felcímkézése” (NFC Tag)
 - Passzív
 - Szöveg, webcím, kontakt információ, okostelefon alkalmazásindítás, ...
- Fizetési rendszerek
 - Bérletek, utazási díjak, jegyek, ...
 - Google Wallet, Apple Pay, PayPass, ...
- Okostelefon integráció
 - Gyors adatátvitel készülékek összeérintésével
 - NFC Tag-ek segítségével alkalmazás, berendezés indítható
 - Otthoni Wifi, TV, ...

Mikrovezérlők

- **Teljes értékű számítógépek**

- Processzor
- Memória
- I/O egységek
- Gyakran a szoftvert már gyártáskor beépítik
 - Csak olvasható memória



- **Általános és speciális célú mikrovezérlők**

- Kis méretű közönséges számítógépek
- Konkrét alkalmazás
- 4, 8, 16 és 32 bites
- Valós időben működnek
- Méret, súly, áramfelvétel és egyéb korlátok figyelembevétele tervezéskor

Mikrovezérlők

- Beágyazott számítógépek ~ mikrovezérlők
- Felhasználásuk
 - Háztartási berendezések
 - Rádiós óra, mosógép, szárítógép, ...
 - Kommunikációs eszközök
 - Mobiltelefon, fax, személyi hívó (pager - beeper)
 - Számítógép-perifériák
 - Nyomtató, lapolvasó, modem, ...
 - Szórakoztató elektronikai cikkek
 - Hifi berendezés, mp3 lejátszó
 - Képpel kapcsolatos berendezések
 - Tv, digitális kamera, objektívek, ...
 - Orvosi berendezések
 - CT, MRI, szívmonitor, digitális hőmérő
 - Katonai fegyverrendszerök
 - Robotrepülőgép, torpedó, ...
 - Kereskedelmi
 - Ital automaták, ATM, pénztárgép, ...
 - Játékok
 - Beszélő baba, játék konzol, rádióvezérelt járművek



Forrás: Wikipedia

Mikrovezérlők

- Arduino beágyazott vezérlő platform
 - Olcsóbb legyen, mint egy nagy pizza extra feltéttel
 - Massimo Banzi és David Cuartielles
 - Ivrea, Olaszország
 - Egy teljes Arduino rendszer kevesebb, mint 20\$-ért
 - Nyílt forrású hardver
 - Szabadon elérhetőek a tervezetek
 - Meg lehet építeni (el is lehet adni)
 - Atmel AVR 8-bit RISC mikrovezérlő
 - Alapvető I/O támogatást is tartalmaz
 - Wiring programozási nyelv
 - Aktív fejlesztő közösség
 - <http://www.arduino.cc>

Miniatűr PC-k

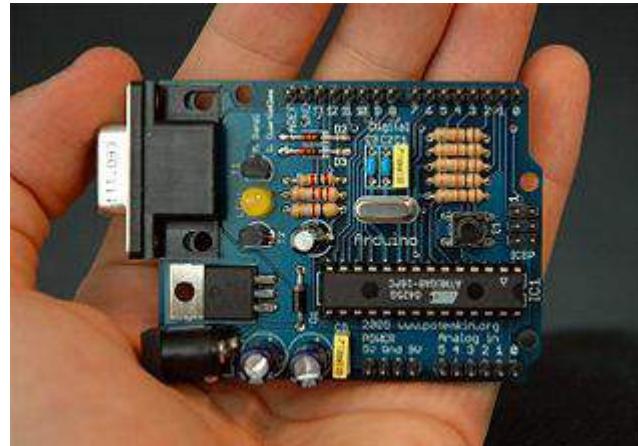
- **Kis méret, PC-szintű tudás**

- Teljes rendszer (CPU, B/K vezérlő, memória, ...)
- Külső perifériák nélkül
 - De ráköthető USB billentyűzet, pendrive; csatlakozik TV-re, monitorra

- **Raspberry Pi**

- 800 MHz processzor, 512 MB RAM
- Olcsó! \$35

- **Arduino**



Okostelefon, tablet

Személyi titkári funkciók

- **Menedzserkalkulátorok**
 - EPOC, Apple Newton
 - Monokróm kijelző
 - Digitális titkári feladatok
 - Naptár, kontaktlista, teendők
- **PDA-k**
 - Palm OS, Windows Mobile
 - Asztali PC adatok szinkronja
 - Nyitás a multimédia felé
 - Nincs telefon funkció

Telefónia

- „**Buta”telefonok**
 - Hangátvitel, SMS
- **Kezdeti okostelefonok**
 - Nokia Communicator
- **PDA + telefónia**
 - RIM Blackberry, Palm, Symbian
- **Modern okostelefonok**
 - **Apple iOS, Android, Bada, WebOS, Windows Phone**
 - Rendszerint ARM CPU

Játékgépek

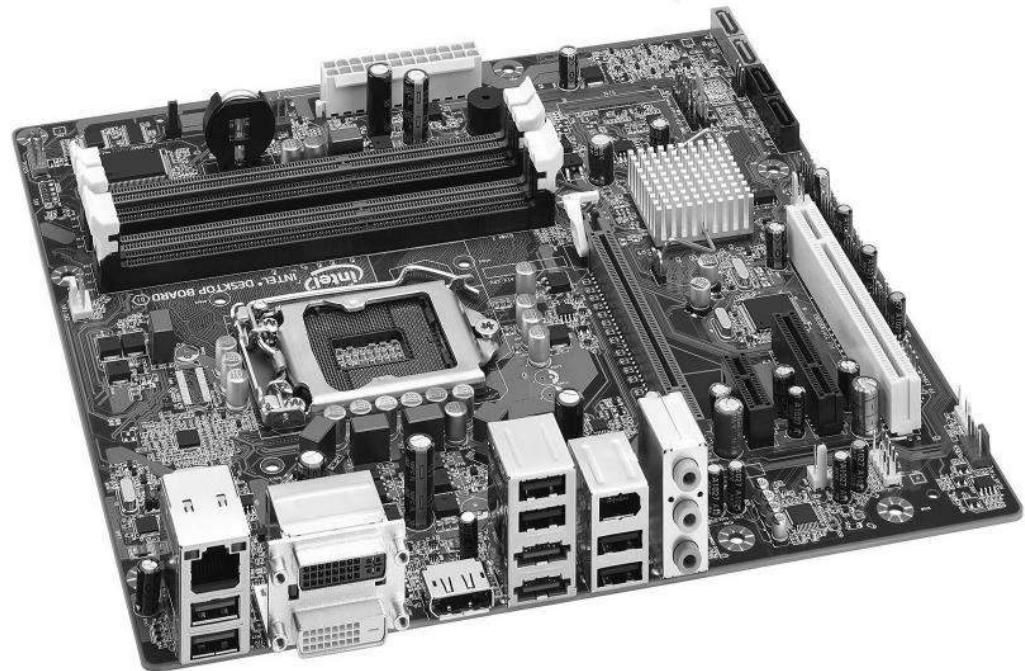
- Sony PS3
 - Cell mikroprocesszor
 - 512 MB RAM
 - 550-MHz módosított Nvidia
 - Blu-ray lejátszó
- Microsoft XBOX
 - IBM 3 magos Power CPU
 - 512 MB RAM
 - 500-MHz módosított ATI
 - DVD lejátszó+háttértároló



Személyi számítógép

- Asztali, hordozható
 - Desktop
 - Laptop
 - Notebook
 - Netbook
- Alaplap
 - Motherboard

Forrás: Tanenbaum



Intel DQ67SW alaplap.
© 2011 Intel Corporation.

Kiszolgálók és munkaállomások

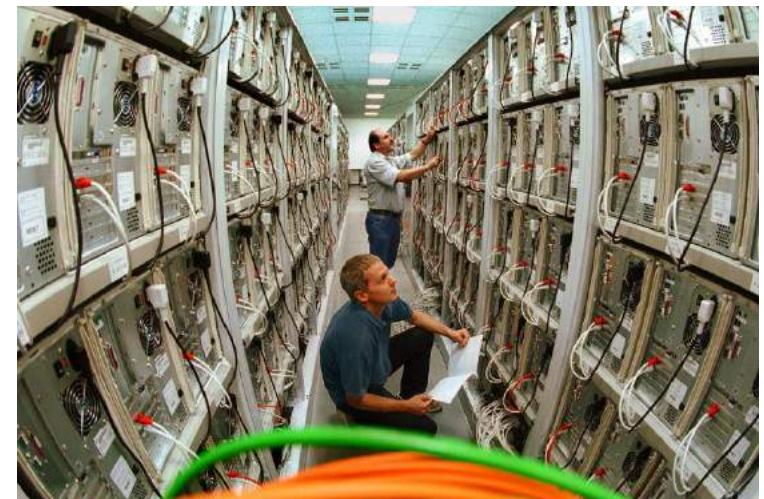
Kiszolgálók

- Erősebb személyi számítógépek vagy munkaállomások
 - Hálózati kiszolgáló (szerver)



Munkaállomások

- Munkaállomás klaszterek
- Gigabites hálózaton összekötött személyi számítógépek



Forrás: Wikipedia

Nagyszámítógépek

- Tovább üzemeltették a régi nagy gépeket
 - Régi szoftverek
 - 2000 év probléma
 - Két számjegy hozzácsapása az évszámokhoz
 - 9999 december 31. éjfél?
 - 8000 éves COBOL programok megadják magukat
- Internet
 - Nagyszámú elektronikus üzleti tranzakció
- Rendkívül gyors CPU
- Sok gigabájt központi memória
- Nagyon gyors háttértárolók és hálózat
- Feladatok
 - Nagy mennyiségű tudományos és mérnöki számítások
 - Gyógyszerek szintetizálása
 - Légmozgás modellezése
 - ...

Szegedi szuperszámítógép

A Szegedi Tudományegyetemen egy Hewlett-Packard gyártmányú szuperszámítógépet helyeztünk el. A HP a piac három vezető szuperszámítógép gyártója közül az egyik.

A Szegeden üzembe helyezett gép egy ún. fat-node cluster amely az igen kifinomult, **CP4000BL típusú blade** technológiát tartalmazza. A gépben a legkorszerűbb **AMD Opteron 6174** típusú **12 magos Magny Cours processzorok** találhatók, a számítási csomópontokban lévő **magok száma összesen 2112**. A belső kommunikációt a nagy teljesítményű, redundáns, mesh topológiájú Infiniband hálózat biztosítja. A gépben **5,6 Tbyte memória** található, és tartozik hozzá egy **0.25 Pbyte méretű háttértár** és high-end vizualizációs alrendszer is. A gép számítási teljesítménye meghaladja a **14 Tflops** értéket, és különlegesen jó energiahatékonyiségi mutatókkal rendelkezik.

E szuperszámítógép különlegessége, hogy nagyon hatékonyan futtathatók rajta a vegyes párhuzamos programozási paradigmákat (üzenetküldéses + osztottmemóriás) alkalmazó algoritmusok is, hiszen olyan clusterről van szó, amelynek minden csomópontja egy igen erős, 48 magos SMP számítógép. Ez az újszerű, egyszerre MPI-t és OpenMP-t is alkalmazó programozási technika különlegesen hatékony alkalmazások készítését teszi lehetővé. A gép operációs rendszere RedHat Linux, és el van látva az optimalizált parallel fejlesztési és futtatási környezetekkel.

Számítógépcsaládok

- x86
 - Személyi számítógépek
 - Szerver rendszerek
- Szerverek
 - UltraSPARC III
- Beágyazott rendszerek
 - 8051
 - AVR
 - Nagyon alacsony költségű mikrovezárló
 - Beágyazott számítási alkalmazások
 - Pl. kocsik, televíziók stb. vezérlésére
- ARM
 - Mobil piac
 - Okostelefonok
 - Tabletek
 - Okostévék, médialejátszók

x86 architektúra

- Robert Noyce, Gordon Moore és Arthur Rock
 - 1968 – Intel Corporation
 - Memória lapok készítése
 - 1970
 - 4004-es
 - 4 bites
 - 2300 tranzisztoros egylapkás CPU
 - 8008-as
 - 8 bites fejlesztése
 - 16 KB
 - 1972
 - Nagy volt a kereslet rá
 - 8080-as
 - 64 KB
 - Millió szám adtak el

x86 architektúra

Lapka	Dátum	MHz	Tranz.	Mem.	Megjegyzés
I-4004	1971/4	0.108	2300	640 B	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
P. Pro	1995/3	150-200	5.5M	4 GB	Két szintű beépített gyorsítótár
P. II	1997/5	233-400	7.5M	4 GB	Pentium Pro + MMX

x86 architektúra

Lapka	Dátum	MHz	Tranz.	Mem.	Megjegyzés
P. III	1999/2	650-1400	9.5M	4 GB	SSE utasítások 3D grafikához
P. 4	2000/11	1300-3800	42M	4 GB	Hyperthreading + több SSE
Core Duo	2006/1	1600-3200	152M	2 GB	Két mag egyetlen tokban
Core	2006/7	1200-3200	410M	64GB	64 bites négy magos architektúra
Core i7	2011/1	1100-3300	1160M	24GB	Integrált grafikus processzor

x86 architektúra

- Az összes Intel-lapka felülről kompatibilis elődjével egészen 8086-ig
 - A Pentium 4 a 8086-os programjait módosítás nélkül tudja futtatni
 - Apránkénti bővítgetés
 - Nem elegáns a Pentium 4 felépítése
- Magas fokú integráltság
 - Hő kibocsátás
 - 3,6 GHz-es Pentium 4 fogyasztása 115 Watt
 - Több CPU egy lapkán közös memória terüettel (gyorsítótár)
 - Kevesebb energiát fogyasztana

UltraSPARC III

- 1970-es évek
 - Népszerű Unix rendszerek
 - Nem futottak személyi számítógépeken
 - Túlterhelt gépek
 - Andy Bechtolsheim
 - Saját Unix munkaállomás megépítése
 - Meglévő elemekből
 - SUN-1 (Stanford University Network)
 - Vinod Khosla
 - Scott McNealy gyártás irányítás
 - Bill Joy szoftver készítés
 - 1982 - Sun Microsystems
- SUN-1
 - Motorola 68020 CPU
 - Nagyobb teljesítményű volt, mint a PC
 - Hálózatban működnek
 - Ethernet-csatolókártya
 - TCP/IP
 - ARPANET
 - Internet elődje
- Saját processzor
 - RISC II mintájára
 - California at Berkley
 - Scalable Processor Architecture (SPARC)
 - SUN-4 alapja

UltraSPARC III

- Átadta a gyártási jogokat
- SPARC nyílt architektúra
- Az első SPARC 36 MHz-es valódi 32 bites
 - IU (Integer Unit) processzor három utasítás típussal, 55 utasítás
 - Lebegőpontos egység 14 utasítás
- 1995 valódi 64 bites architektúra
 - 64 címbit
 - 64 bites regiszterkészlet
 - UltraSPARC I
 - Binárisan kompatibilis volt a többi 32 bites SPARC gépekkel
 - Visual Instruction Set (VIS)
 - Multimédiás alkalmazások
 - UltraSPARC II, UltraSPARC III, UltraSPARC IV (két UltraSPARC III)

8051

- Beágyazott rendszerek
- 1976
 - 8 bites 8080
 - Szükség volt
 - 8080-as CPU lapkára
 - Egy vagy több memórialapkák
 - Egy vagy több B/K lapka
- 8748
 - 8080-hoz hasonló processzor
 - 1 KB olvasható memória
 - 64 bájtos író-olvasható memória
 - 8 bites időzítő
 - 27 I/O vonal
- 8051
 - 60 000 tranzisztor
 - Sokkal gyorsabb CPU
 - 4 KB olvasható memória
 - 128 bájt írható-olvasható
 - 32 I/O vonal
 - Egy soros port
 - Két 16 bites időzítő
 - Olcsó
 - Évente 8 milliárdot adnak el belőle
- 8751, 8752
 - Programozható
 - Debug-golás
 - Kiégetés
 - Ultraibolya sugárzással

8051

Lapka	Program-memória	Memória típus	RAM	Időzítők	Megszakítások
8031	0 KB		128	2	5
8051	4 KB	ROM	128	2	5
8751	8 KB	EPROM	128	2	5
8032	0 KB		256	3	6
8052	8 KB	ROM	256	3	6
8752	8 KB	EPROM	256	3	6

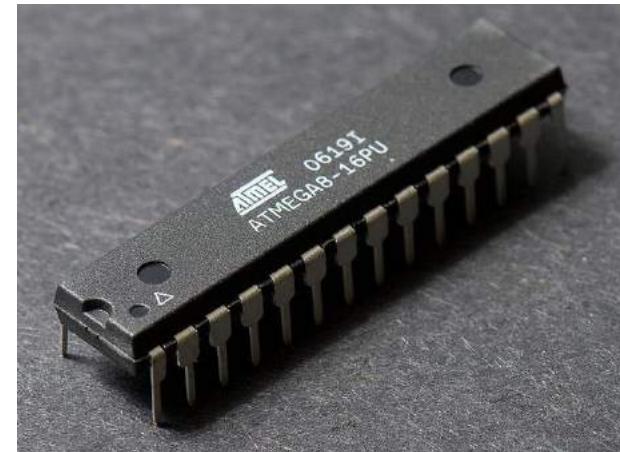
AVR architektúra

- Nagyon alacsony árfekvésű beágyazott rendszerekben használják
- 1996 - Norwegian Institute of Technology
 - Alf-Egil Bogen és Vegard Wollan
 - 8 bites RISK CPU-t terveztek
 - (A)lf és (V)egard (R)ISC
- Atmel megvásárolta
 - Finomították a terveket
 - 1997 - AT90S1200
 - Intel 8051-es lábkiosztás
- A lelke a nyílt forrású Arduino beágyazott vezérlő platformnak

AVR architektúra

Forrás: Wikipedia

- Három fajta mikrokontroller lett megvalósítva
- Flash memória
 - programozható, nem felejtő
 - Programkód és adat
- EEPROM
 - Nem felejtő
 - Futási időben változtatható programmal
 - Felhasználói konfigurációk tárolása
- RAM
 - Program változók tárolása program futása során
 - felejtő



Lap	Flash	EEPROM	RAM	Lábak	Tulajdonságok
tinyAVR	0,5-16 KB	0-512 B	32-512 B	6-32	Kicsi, digitális I/O, analóg bemenet
megaAVR	8-256 KB	0,5-4 KB	0,25-8 KB	28-100	Sok periféria, analóg kimenet
AVR XMEGA	16-256 KB	1-4 KB	2-16 KB	44-100	Titkosítás gyorsítás, USB I/O

AVR architektúra

- Mikrokontrollerek sikere
- Pl. Atmel megaAVR-168 perifériák
 - 3 timer
 - 2 8-bites és 1 16-bites
 - Valósidejű óra oszcillátorral
 - Hat impulzus szélességű modulációs csatorna
 - Fény intenzitás vagy motor sebesség vezérlésre
 - Nyolc analóg-digitál konverziós csatorna
 - Feszültségszintek olvasására
 - Univerzális soros fogadó/küldő
 - I2C soros interfész
 - Programozható watchdog timer
 - Analóg összehasonlító két feszültség összehasonlítására
 - ...

ARM architektúra

- Korai 80-as évek
 - Acorn Computer
 - Elkezdtek dolgozni a második gépükön
 - IBM PC versenytársa
 - A korábbi 8-bites processzor nem lett volna elég a 16-bites 8086-os legyőzésére
 - Saját processzort terveztek
 - Berkeley RISC által inspirált
 - Acorn RISC Machine (ARM)
- Átkeresztelték később
 - Advanced RISC Machine
 - Mikor az ARM levált az Acorn-ról
- 1985-re fejezték be a tervezést
 - 32-bites utasítás és adatok
 - 26-bites címterület
 - VLSI technológiával gyártották

ARM architektúra

- Az első ARM architektúra (ARM2)
 - Acorn Archimedes személyi számítógépben jelent meg
 - 2 MIPS
 - 899 angol font
 - Népszerű volt (UK, Írország, Ausztrália, Új-Zéland)
- Apple Newton projekt
 - Kézi számítógép
 - 1993 - ARM 610
- 1990-es évek
 - DEC együttműködés
 - Nagy sebességű, alacsony fogyasztású ARM fejlesztése
 - StrongARM
 - 233 MHz
 - 1 watt

ARM architektúra

- Az ARM nem gyárt mikroprocesszorokat
 - Terveket és ARM-alapú fejlesztő eszközöket és könyvtárakat (lib) készítenek
 - Ezeket licenszálják rendszer tervezőknek, és lapka gyártóknak
 - Pl. Samsung Galaxy Tab
 - Androidos táblagép CPU-ja egy ARM alapú processzor
 - Két ARM Cortex-A9 processzor és egy Nvidia GeForce grafikus feldolgozó egység
 - A Tegra 2 magokat az ARM tervezte
 - Nvidia integrálta
 - Taiwan Semiconductor Manufacturing Company (TSMC) gyártja
- 2011 október
 - 64-bites ARM
- 2011 január Nvidia Denver projekt
 - ARM alapú processzor szerverekbe
 - Több 64 bites ARM processzor és egy általános célú GPU
 - Szerver farmok hűtési problémájának megoldására

Mértékegységek

Hatványkitevő	Előtag
10^{-3}	milli
10^{-6}	micro
10^{-9}	nano
10^{-12}	pico
10^{-15}	femto
10^{-18}	atto
10^{-21}	zepto
10^{-24}	yocto

Hatványkitevő	Előtag
10^3	Kilo
10^6	Mega
10^9	Giga
10^{12}	Tera
10^{15}	Peta
10^{18}	Exa
10^{21}	Zetta
10^{24}	Yotta

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Mértékegységek

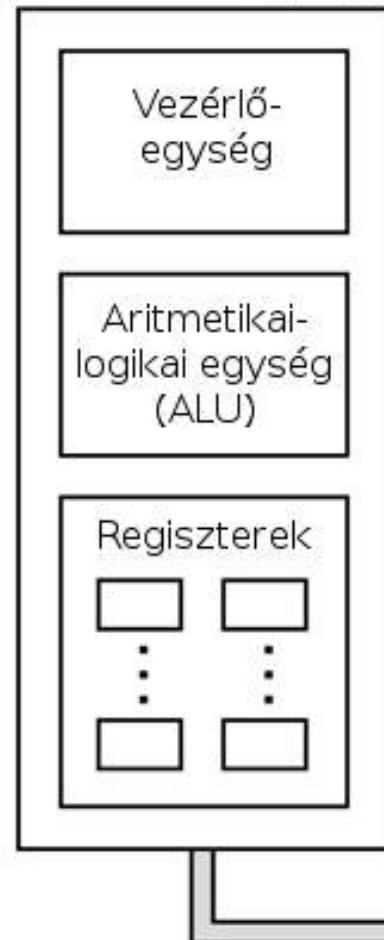
- **Memória méretére vonatkozóan**
 - 2 egész kitevős hatványai, így
 - $1 \text{ KB} = (1 \text{ KiB}) = 2^{10} = 1024 \text{ bájt}$
 - $1 \text{ MB} = (1 \text{ MiB}) = 2^{20} = 1073741824 \text{ bájt}$
 - ...
- A **háttértárak** (merevlemezek, memóriakártyák) **tárolókapacitását** viszont a hagyományos módon (10 hatványai) adják meg!
 - $1 \text{ kB} = 1000 \text{ bájt}$
 - $1 \text{ MB} = 1000^2 \text{ bájt}$
- **Adatátvitelnél** szintén 10-es alapú
 - $1 \text{ Mbps} = 10^6 \text{ bit átvitele másodpercenként}$

Számítógéprendszerek felépítése

- Főbb komponensek legfontosabb feladatai
 - MOST
 - Neumann-elvű gép legfontosabb egységeinek áttekintése
 - CPU vázlatos működése
 - Tervezési elvek
 - KÉSŐBB
 - Fizikai felépítés
 - Tényleges megvalósítási lehetőségek
 - Egyes megvalósítások részletes jellemzői

Neumann-elvű gép sematikus váza

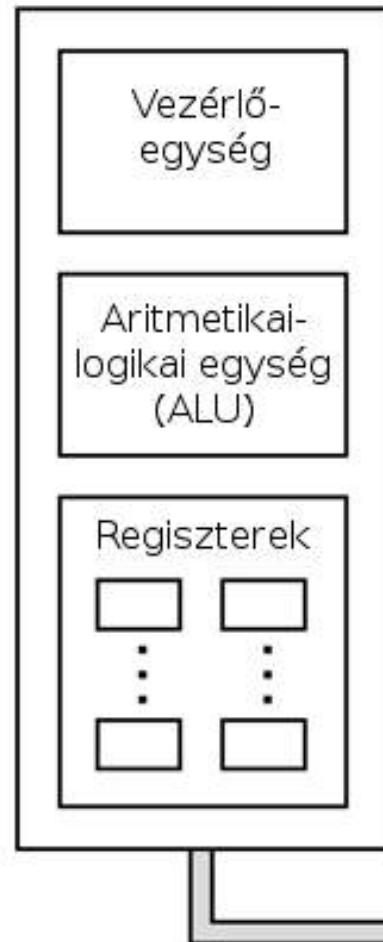
Központi vezérlőegység (CPU)



*Tárolt programú digitális számítógép
alapvető komponensei, kapcsolatai*

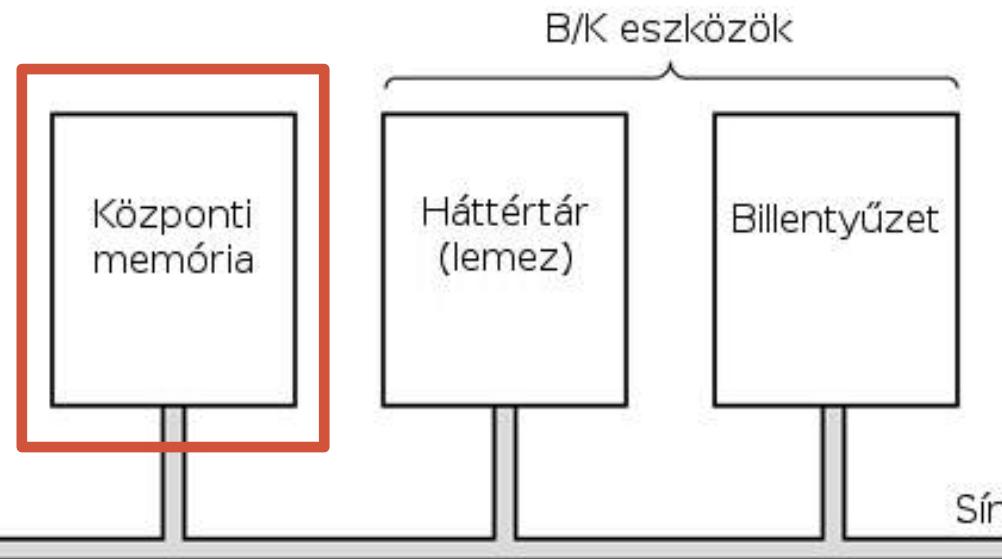
Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



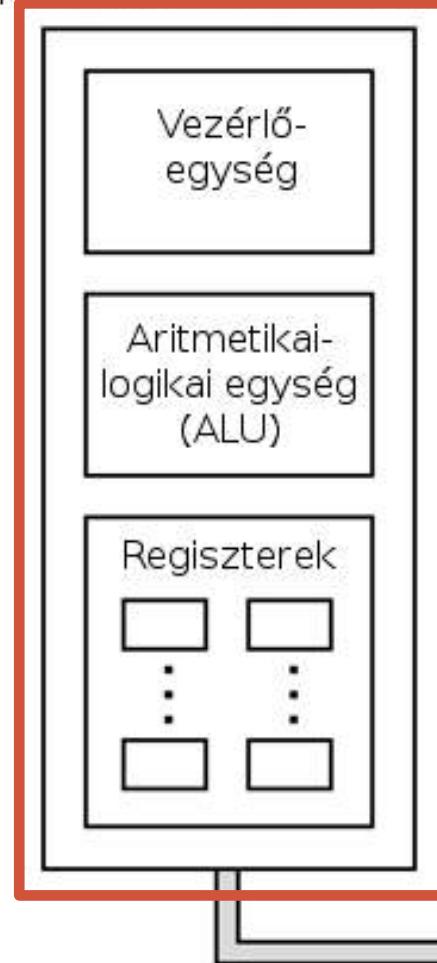
Központi memória

- Program kódja és adatai
- Számokként tárolva



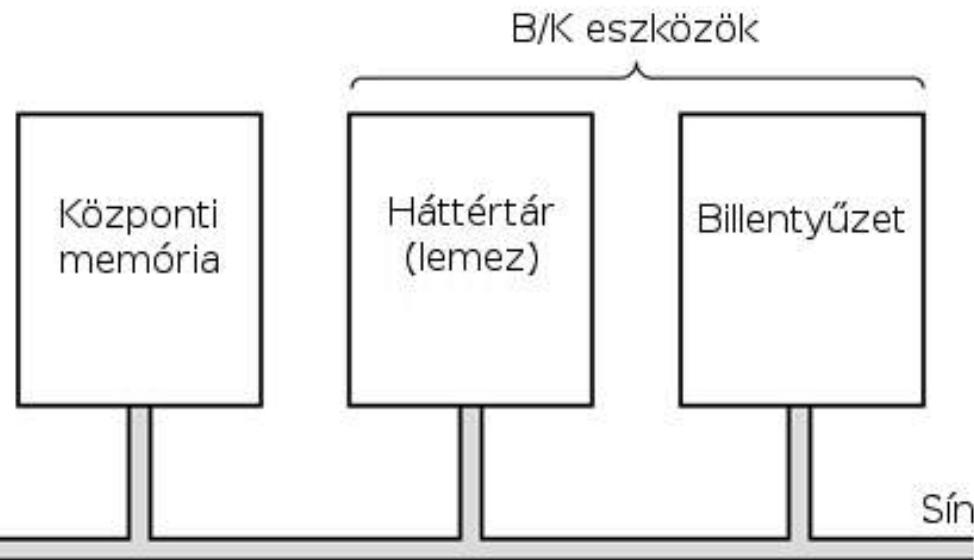
Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



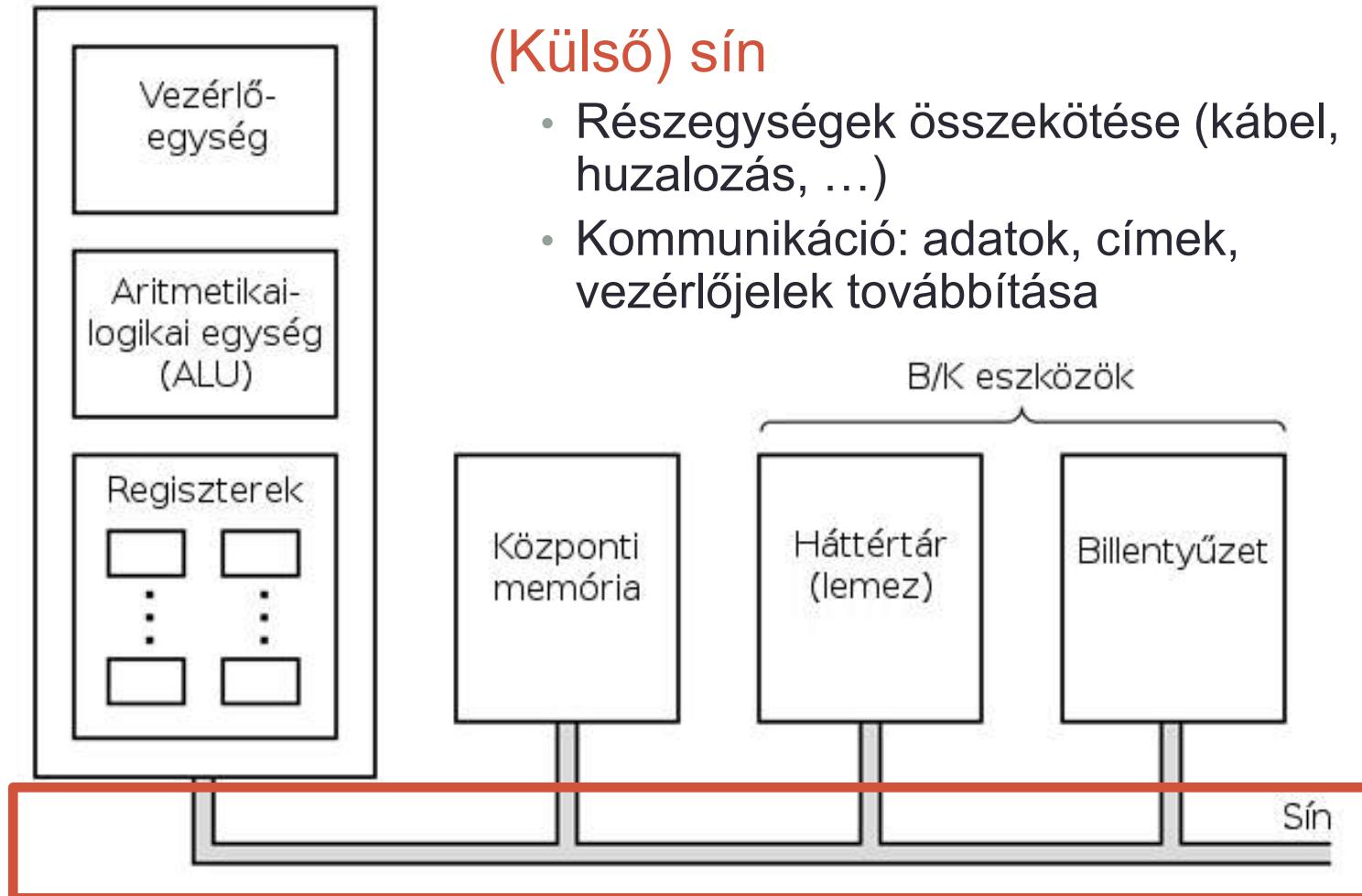
Központi feldolgozóegység (CPU)

- A központi memoriában tárolt program utasításainak beolvasása és végrehajtása



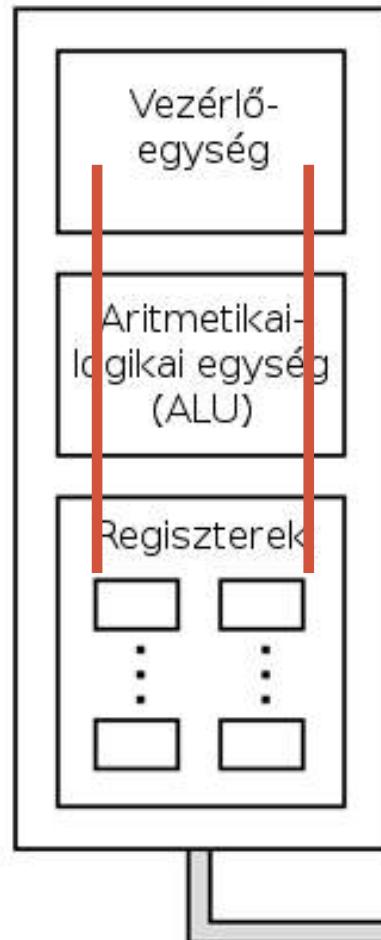
Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)

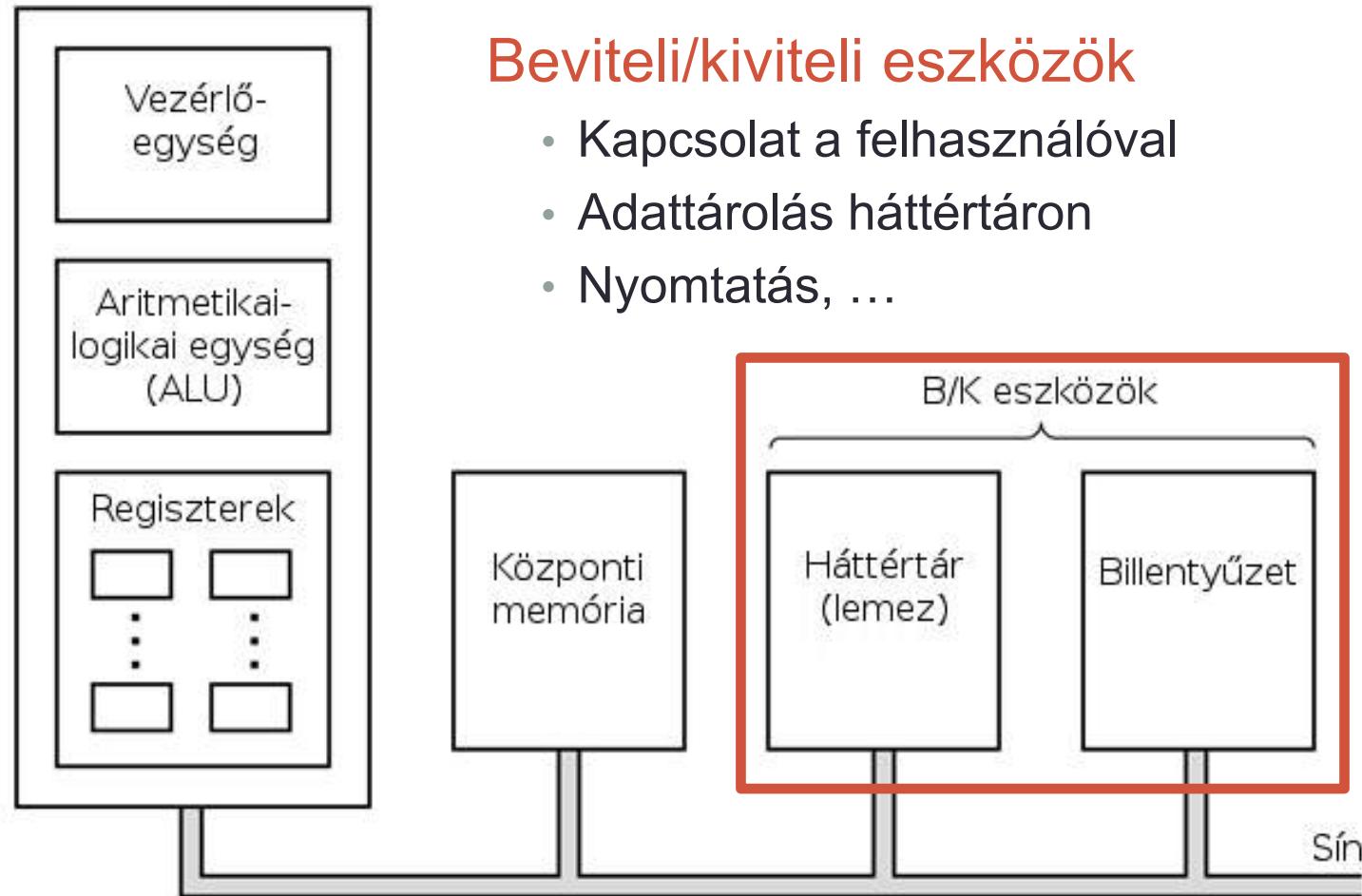


Belső sín

- CPU részegységei közötti kommunikáció (vezérlőegység, ALU, regiszterek)

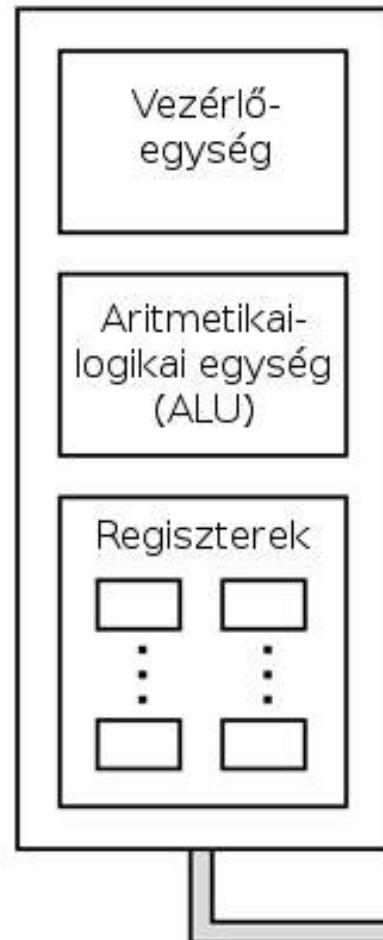
Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



Működést biztosító járulékos eszközök

- Gépház
- Tápellátás, ...

B/K eszközök

Sín

Processzorok (CPU-k)

Központi feldolgozóegység (CPU)

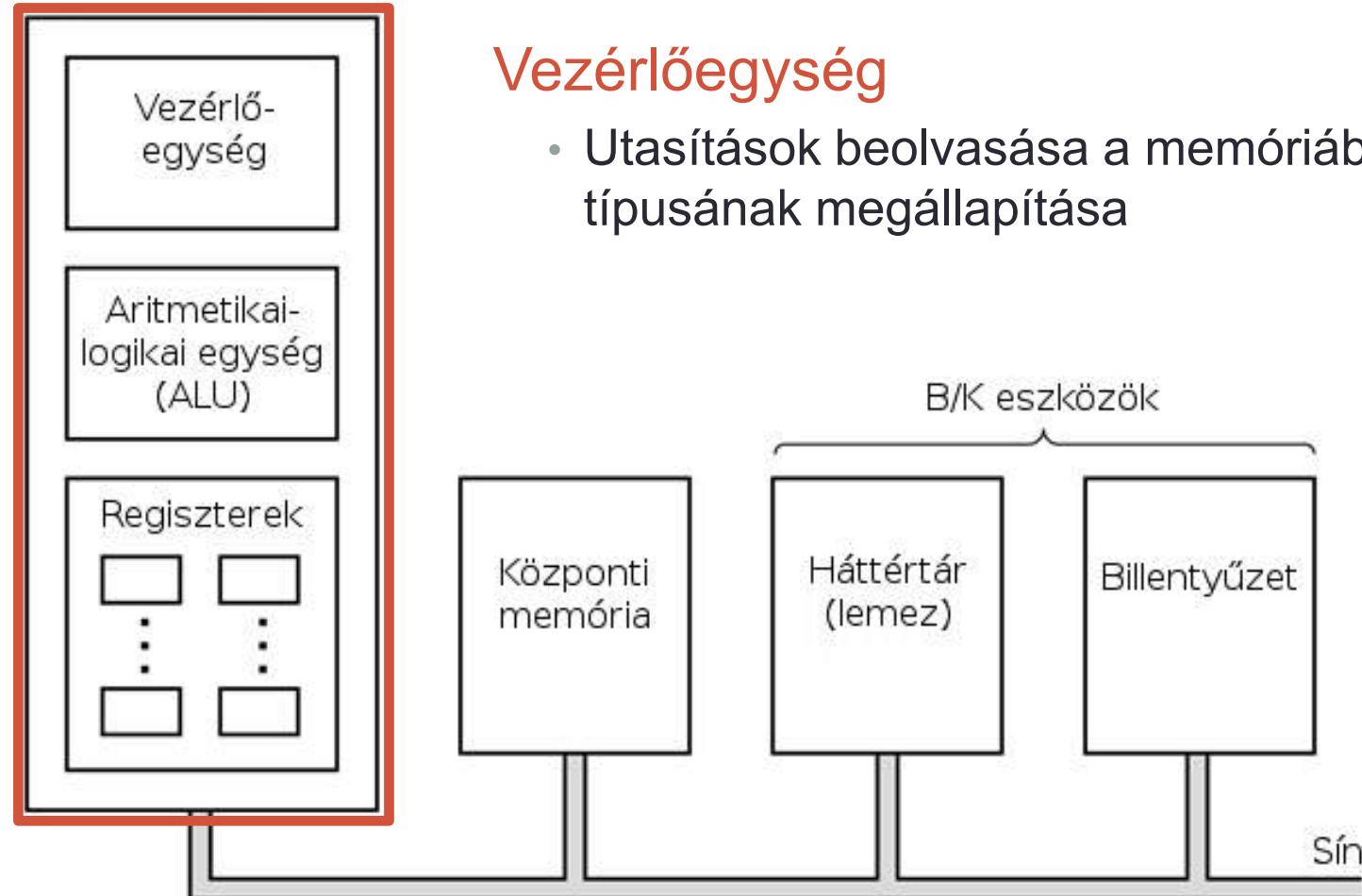


Központi feldolgozóegység (CPU)

- A központi memoriában tárolt program utasításainak beolvasása és végrehajtása

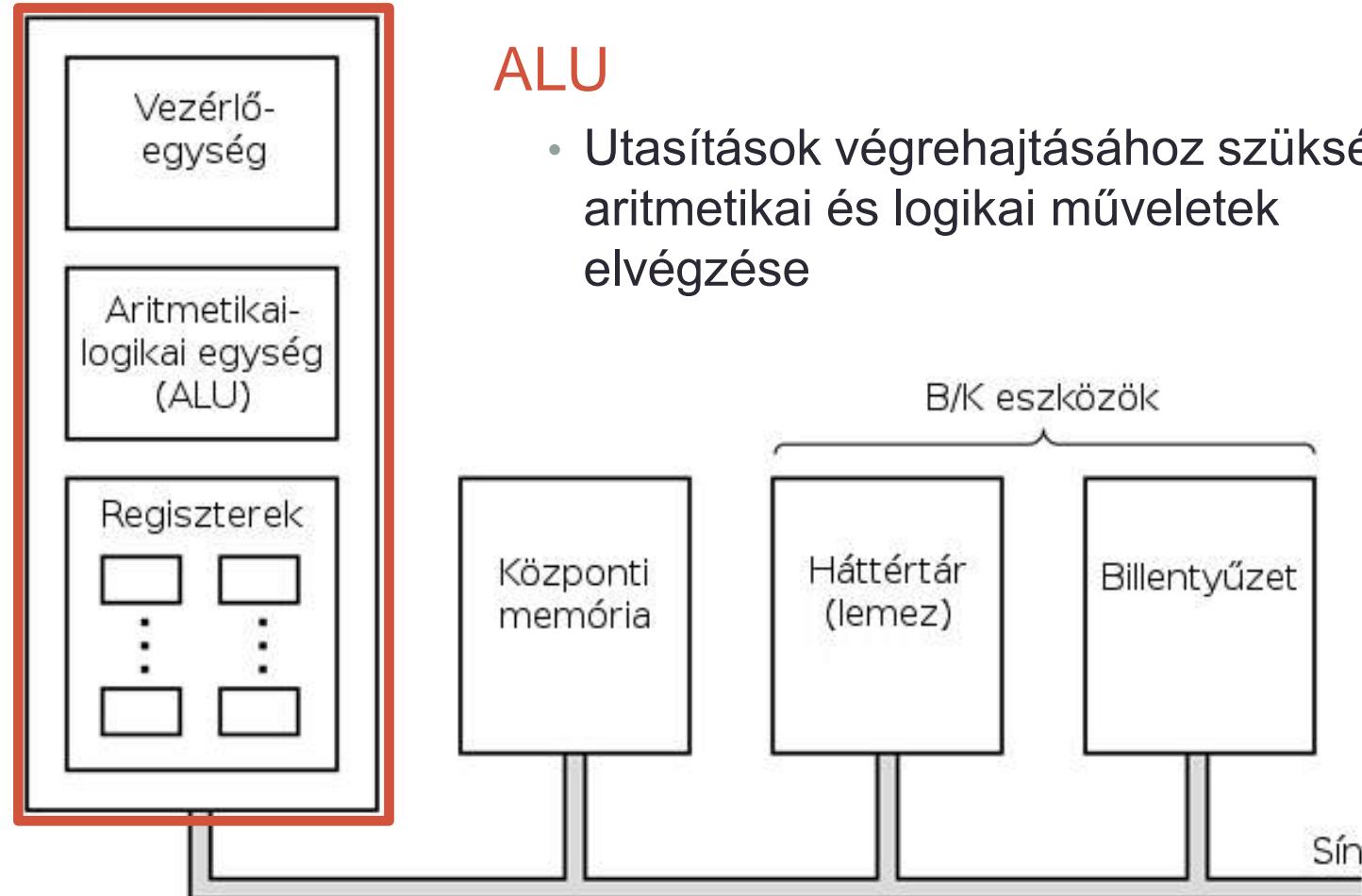
Processzorok (CPU-k)

Központi feldolgozóegység (CPU)



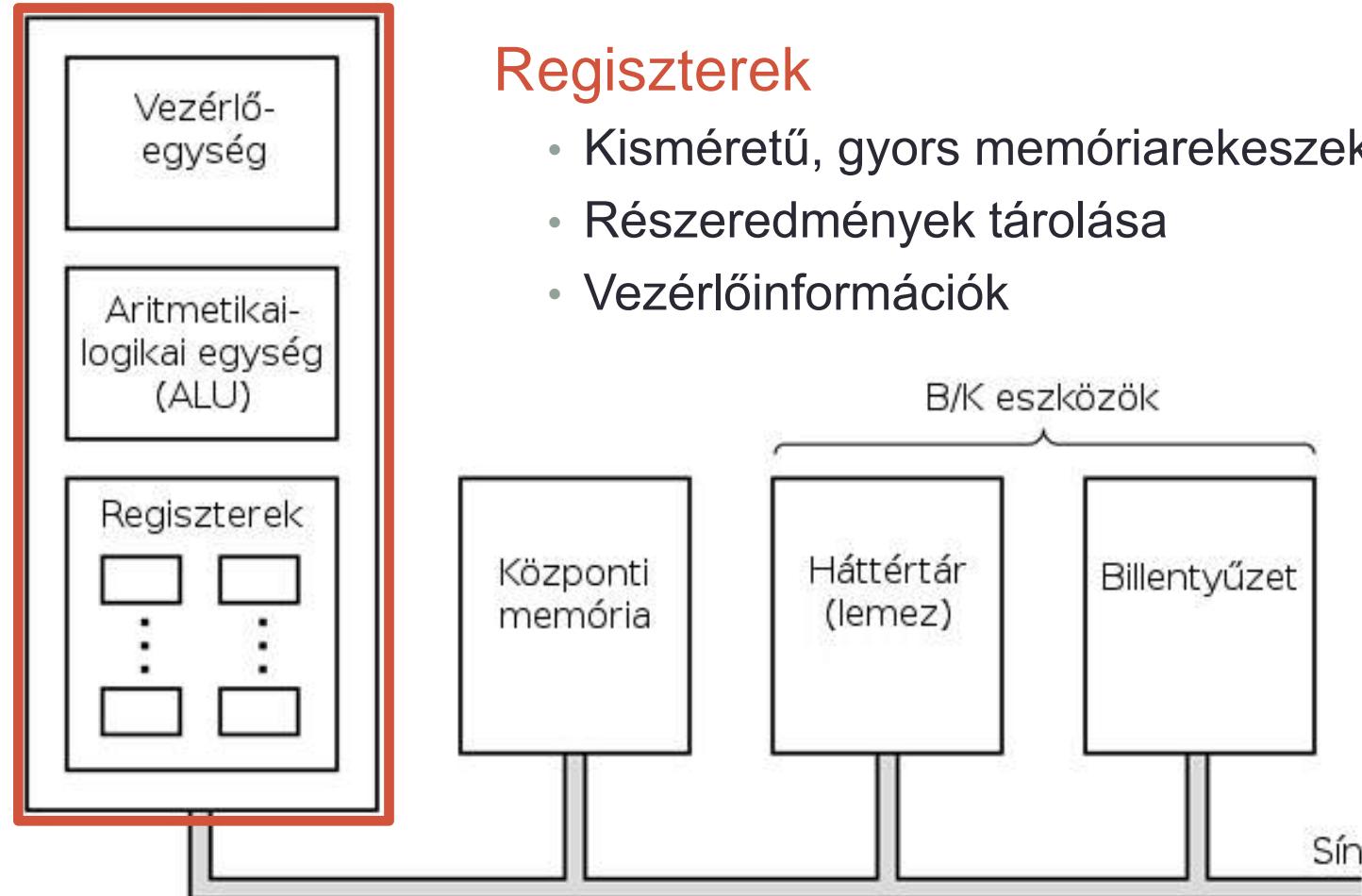
Processzorok (CPU-k)

Központi feldolgozóegység (CPU)

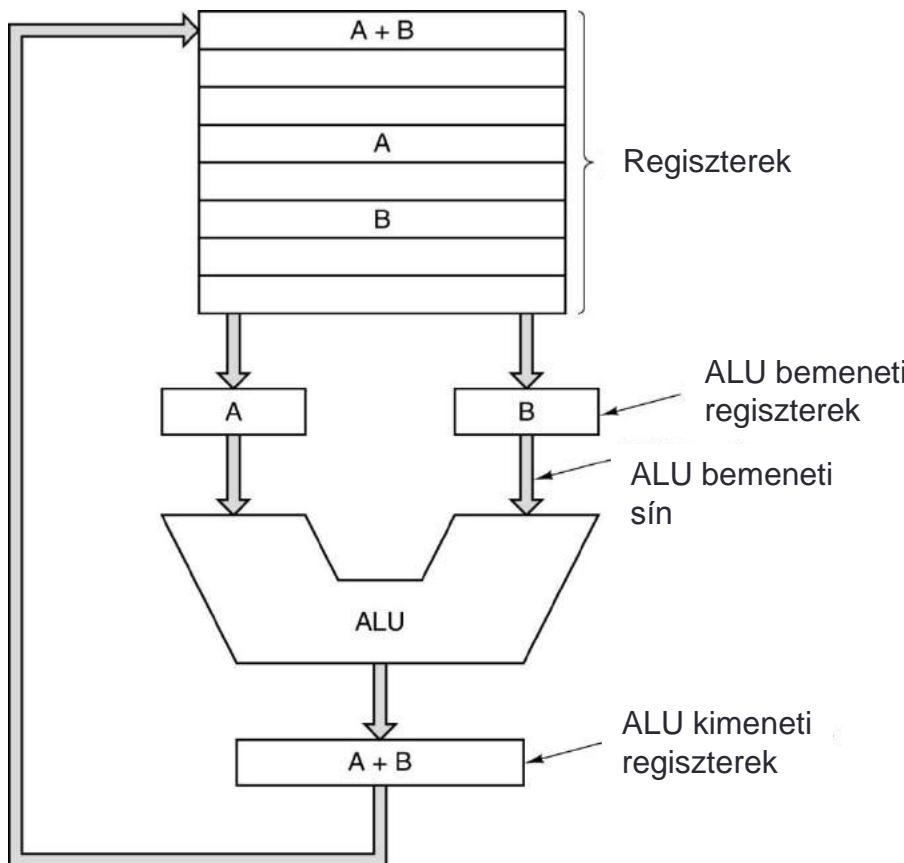


Processzorok (CPU-k)

Központi feldolgozóegység (CPU)



Adatút



- Példa

- Összeadás elvégzése ALU-val
- Legtöbb utasítás
 - Regiszter-memória
 - Regiszter-regiszter
- Adatciklus
 - Két operandus ALU-n való átfutása és az eredmény regiszterben tárolása

Utasítás-végrehajtás

- **Betöltő-dekódoló-végrehajtó ciklus**

1. Soron következő utasítás beolvasása a memóriából az utasításregiszterbe az utasításszámláló regiszter mutatta helyről
2. Utasításszámláló beállítása a következő címre
3. A beolvasott utasítás típusának meghatározása
4. Ha az utasítás memóriára hivatkozik, annak lokalizálása
5. Ha szükséges, adat beolvasása a CPU egy regiszterébe
6. Az utasítás végrehajtása
7. Vissza az 1. pontra

- **Probléma**

- A memória olvasása lassú, az utasítás és az adatok beolvasása közben a CPU többi része kihasználatlan

Utasítás-végrehajtás

- **Gyorsítási lehetőségek**

- Órajel frekvenciájának emelése (korlátozott)
- Utasításszintű párhuzamosság
 - Csővezeték
 - Szuperskaláris architektúrák
- Processzorszintű párhuzamosság
 - Tömbszámítógépek
 - Multiprocesszorok
 - Multiszámítógépek

- **Definíciók**

- Késleltetés: utasítás végrehajtásának időigénye
- Áteresztőképesség: MIPS (millió utasítás mp-enként)

Utasításszintű párhuzamosság

- Előolvasási puffer

- Az utasítás végrehajtásának vége előtt megkezdődik a következő utasítás beolvasása egy dedikált regiszterkészletbe (1959)

- Csővezeték

- Utasítás-végrehajtás több részre osztása

- Példa

- a) 5-fázisú csővezeték

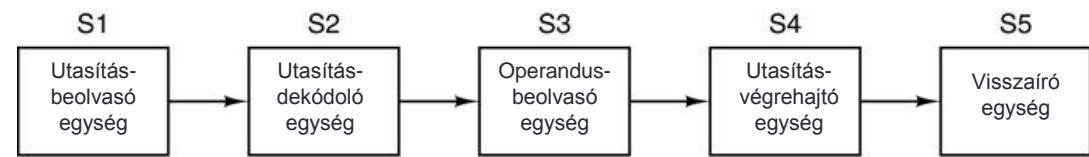
- b) Végrehajtás lépései
9 órajel ciklus alatt

- Tfh.

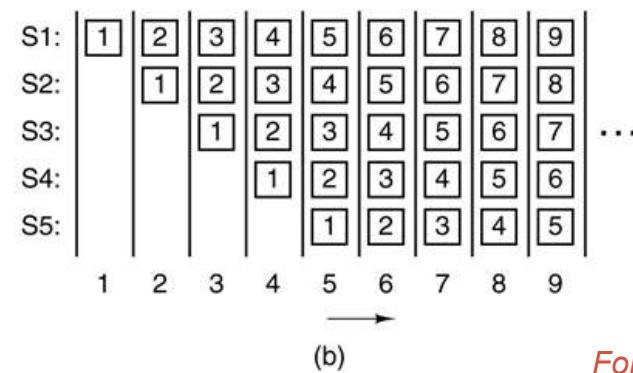
- Órajel: 2 ns

- 1 utasítás végrehajtása: 10 ns

- Áteresztőképesség: 500 MIPS
(100 MIPS helyett!)



(a)

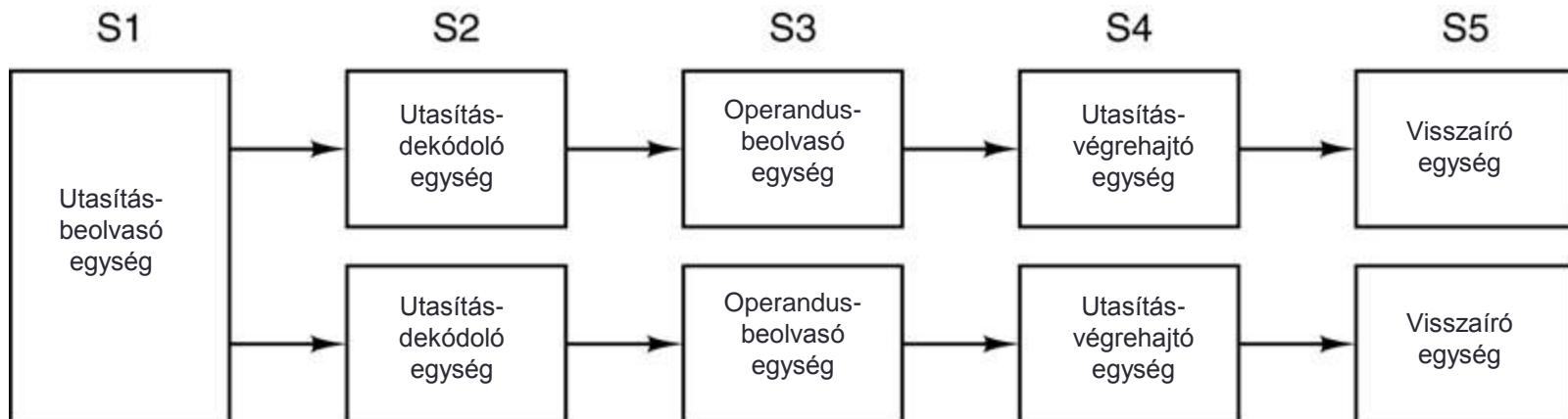


(b)

Utasításszintű párhuzamosság

- **Párhuzamos csővezetékek**

- Közös utasítás-beolvasó egységgel
- A csővezetékek saját ALU-val rendelkeznek
 - Párhuzamos végrehajtás, ha nincs erőforrás-használat ütközés!
- Általában 2 vagy 4 csővezeték

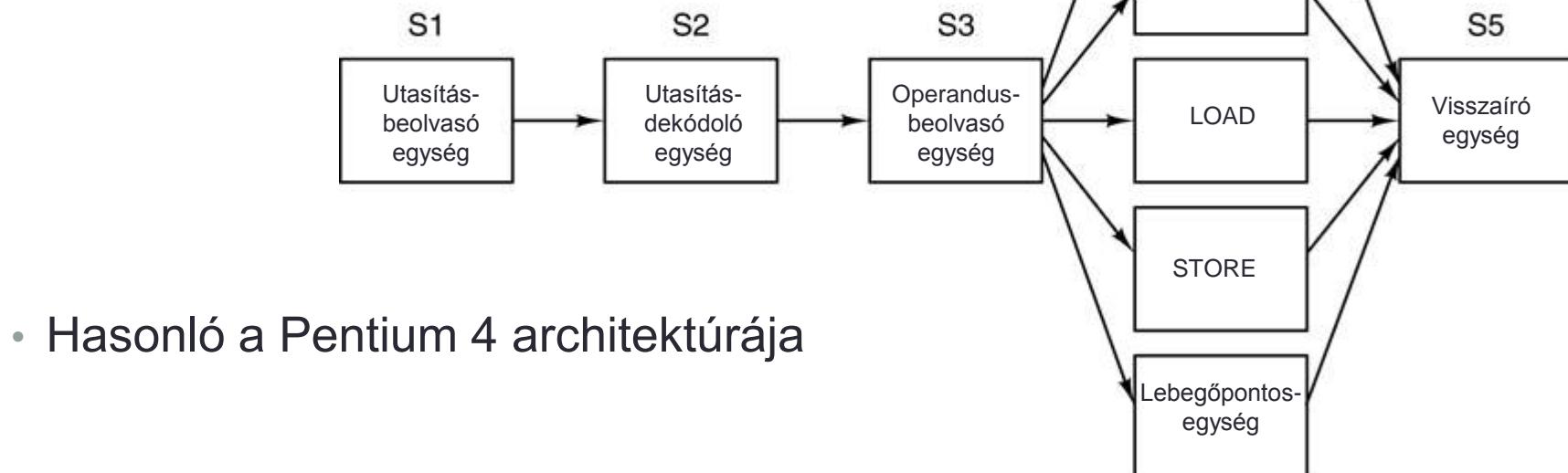


- Pentium hasonlót alkalmaz
 - Fő csővezeték: tetszőleges Pentium utasítás
 - Második csővezeték: csak egész műveletek

Utasításszintű párhuzamosság

- Szuperskaláris architektúrák

- Egy csővezeték, de több funkcionális egységgel
- Feltételezzük
 - S1-S3 fázis sokkal gyorsabb, mint S4
 - Funkcionális egységek ismétlődhetnek
 - Pl. több ALU is lehet



- Hasonló a Pentium 4 architektúrája

Processzorszintű párhuzamosság

- **Tömbszámítógépek**
 - Ugyanazon műveletek elvégzése különböző adatokon → párhuzamosítás
- **Multiprocesszorok (szorosan kapcsolt CPU-k)**
 - Több CPU, közös memória → együttműködés vezérlése szükséges
 - Sínrendszer
 - 1 közösen használt (lassíthat)
 - Emellett a CPU-k akár saját lokális memóriával is rendelkezhetnek
 - Jellemzően max. pár száz CPU-t építenek össze
- **Multiszámítógépek (lazán kapcsolt CPU-k)**
 - Nincs közös sín, processzor-kommunikáció üzenetküldéssel
 - Általában nincs minden gép összekötve egymással (pl. fa-struktúra)
 - Több ezer gép is összeköthető

RISC és CISC

- **RISC** (Reduced Instruction Set Computer)
 - Csökkentett utasításkészletű számítógép
 - Csak olyan utasítások legyenek, amelyek az adatút egyszeri bejárásával végrehajthatók
 - Tipikusan kb. 50 utasítás
- **CISC** (Complex Instruction Set Computer)
 - Összetett utasításkészletű számítógép
 - Sok utasítás (akár több száz), mikroprogram interpretálással
 - Lassabb végrehajtás
- **Intel**
 - A kezdeti CISC felépítésbe integráltak egy RISC magot (80486-tól) a leggyakoribb utasításoknak

Korszerű számítógépek tervezési elvei

- minden utasítást közvetlenül a hardver hajtson végre
 - A gyakran használtakat mindenképpen
 - Interpretált mikrutasítások elkerülése
- Maximalizálni az utasítások kiadási ütemét
 - Párhuzamos utasításkiadásra törekedni
- Az utasítások könnyen dekódolhatók legyenek
 - Kevés mezőből álljanak, szabályosak, egyforma hosszúak legyenek, ...
- Csak a betöltő és a tároló utasítások hivatkozzanak a memóriára
 - Egyszerűbb utasításforma, párhuzamosítást segíti
- Sok regiszter legyen
 - Számítások során ne kelljen a lassú memóriába írni

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Központi memória (Neumann-elvű gép)

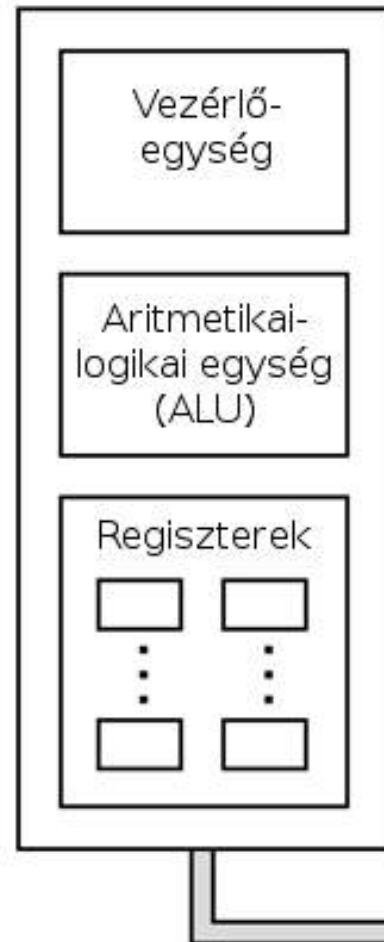
Számrendszerek, átváltások

Adatábrázolás, hibajavító kódolások

Tanács Attila

Neumann-elvű gép sematikus váza

Központi feldolgozóegység (CPU)



*Tárolt programú digitális számítógép
alapvető komponensei, kapcsolatai*
Központi memória

- Program kódja és adatai
- Számokként tárolva

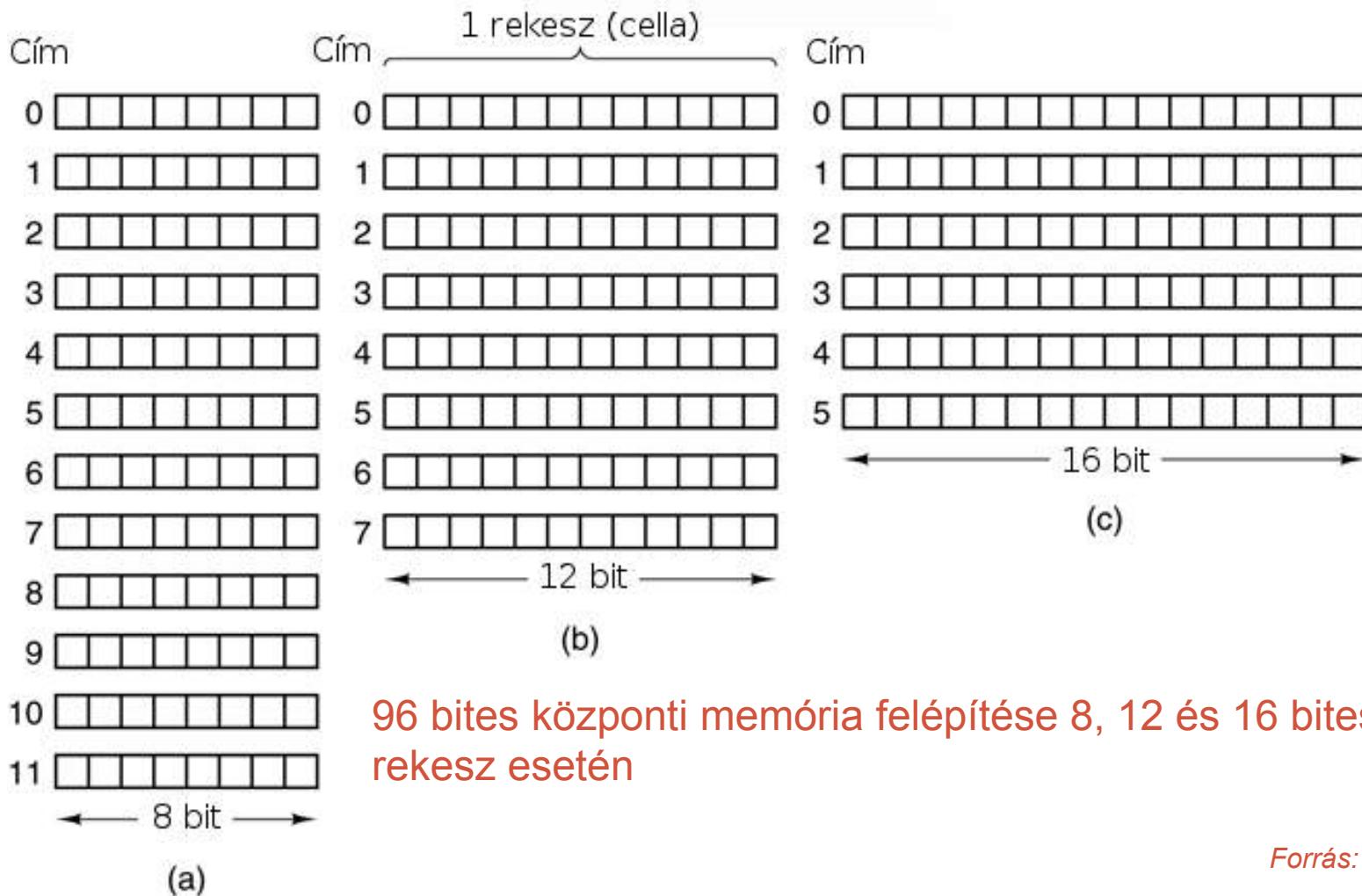
Központi memória

- Tárolás alapja a ***bit***
 - Bináris 0 vagy 1 érték
 - *Van áram vagy nincs áram*
- Legkisebb címezhető egység a ***rekesz (cella)***
 - Memóriában egymást követő bitek
 - Leggyakoribb: **8 bites** rekesz, a **bájt**
 - Lehetséges más csoportosítás is!
 - Újabb gépeknél a bájt a szabványos
- A memória rekeszek sorozata
- A rekeszeket a **címükkel** érhetjük el
 - Számos címzési mód!
 - ~1 dimenziós tömb
 - *Szegmens:offset* címzés

Forrás: Tanenbaum

Számítógép	Bitek/cella
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Eletrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

Központi memória



Forrás: Tanenbaum

Központi memória

- **További csoportosítások**
 - **Szó:** hardver által egységként kezelt bájtok
 - Sokszor a számítógép regiszterszélességének megfelelő bájtok összessége
 - 4 bájt 32-bites rendszeren, 8 bájt 64-bitesen, ...
 - Sok utasítás teljes szavakkal dolgozik
 - **Paragrafus:** 16 bájt együttese, 16-tal osztható címen kezdődően
 - ...
- **Igazítás**
 - Általában címezhetünk tetszőleges bájton kezdődő szavakat,
 - de szóhatáron (négyel, nyolccal, ... osztható címen) kezdődők elérése **gyorsabb lehet.**
 - A fordítóprogramok általában képesek adatokat szó- vagy akár paragrafushatárra igazítani. (Így kihasználatlan memóriaterületek keletkezhetnek.)

Központi memória

- Rekeszek értékeinek értelmezése
 - *Adat*
 - Szám (egész, lebegőpontos, BCD, ...)
 - Karakterkód (ASCII, UTF, ...)
 - *Gépi kód*
 - A számok utasításokat és azok operandusait jelentik
 - A számítógép csak ezeket tudja végrehajtani!
 - Magas szintű nyelveket fordítani (vagy értelmezni) kell!
- Rekesz műveletek
 - Rekesz értékének olvasása
 - Rekesz értékének módosítása (*felülírása*)
 - Lehetnek csak olvasható rekeszek is!
 - Kódterület felülírása önmódosító kódot eredményez! Veszélyes lehet!
 - A modern operációs rendszerek védi a kódterületet felülírás ellen!

Tárgykód (Intel Borland C fordító)

Fájl pozíció

Tárgykód

The screenshot shows three windows of the Lister debugger:

- Left Window (Assembly View):** Shows assembly code for a function named `?live1@32`. The code includes instructions like `push ebp`, `mov ebp, esp`, `mov eax, 5`, and `add edx, eax`. A red box highlights the first few lines of the assembly code.
- Middle Window (Memory Dump View):** Shows memory dump in hex format. A red box highlights the first few bytes of memory.
- Right Window (File Contents View):** Shows the file contents of `sum.c`. It includes source code, preprocessor directives, and assembly code. A red box highlights the assembly code section.

Fájl tartalma:
kód, adat és járulékos (pl. szerkesztőnek szóló) információk.

Középen hexaszámokként, jobb oszlopban karakterkódokként.

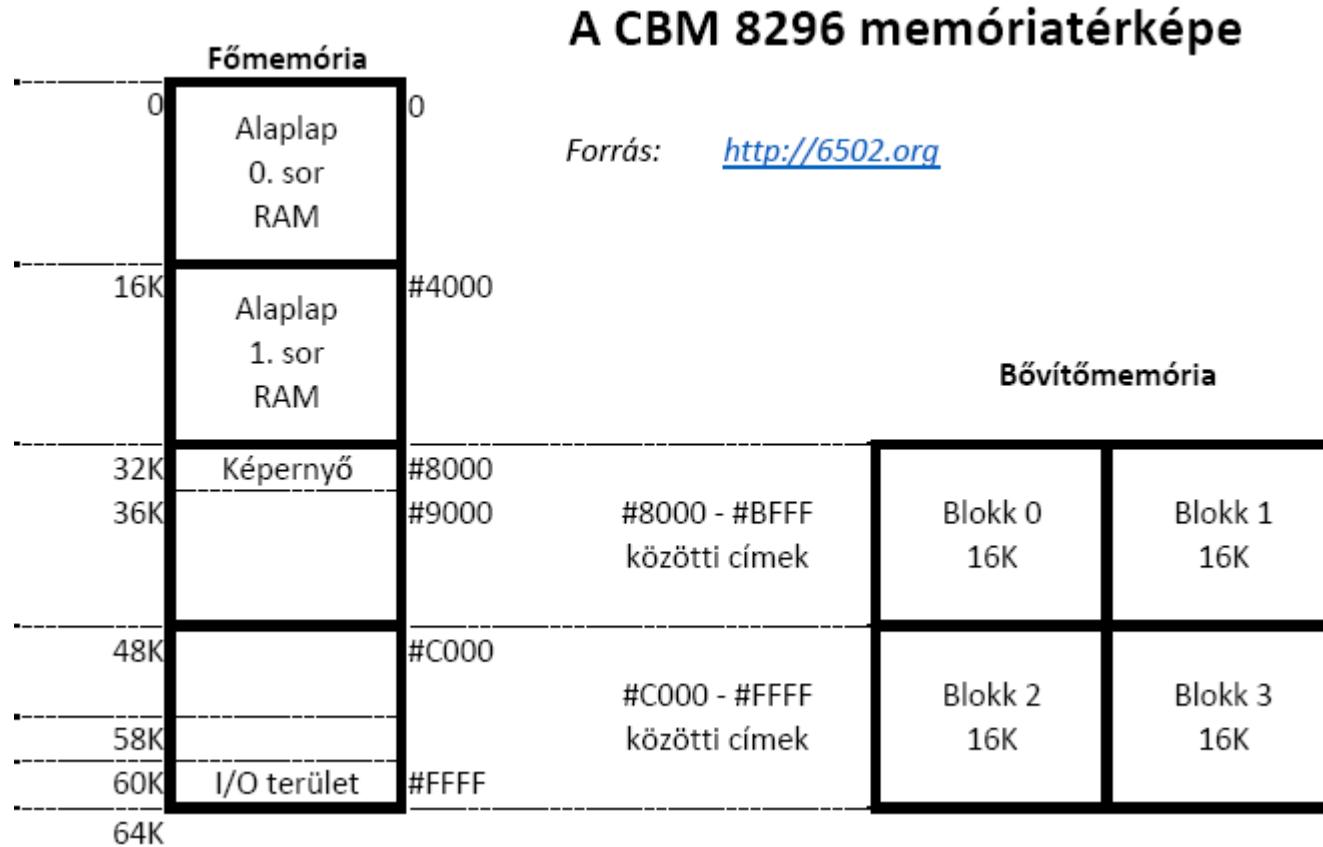
A példát nem kell megtanulni!

Központi memória

- **Felosztása**
 - Nagy része a programok számára szabadon felhasználható
 - Bizonyos címterületek a hardverrel való kapcsolattartásra lehetnek fenntartva
 - Pl. kijelző, merevlemez, külső meghajtók
 - Bizonyos címek meghatározhatják egyes címterületek tartalmát
 - Pl. RAM (írható/olvasható) vagy ROM (olvasható) legyen ott elérhető
 - Átlapolási lehetőség a címtartományban
- **Mérete**
 - Korábban: pár kilobájt, megabájt
 - Pl. A Commodore 64 gép 64 kilobájtot ért el, ez megfelel egy 256x256 méretű szürkeárnyalatos kép mátrixának!
 - Manapság: több gigabájt

Központi memória

- Felosztása
 - Memória átlapolás példa CBM 8296 esetén

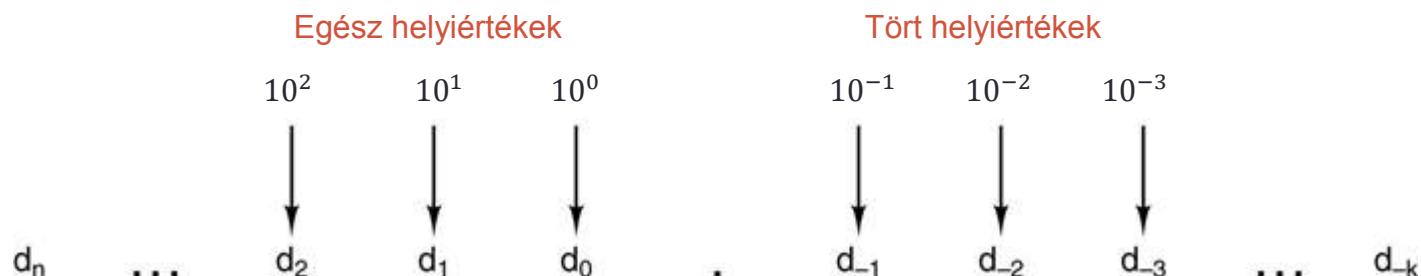


Kis kitérő

- Cél
 - Számok, karakterek digitális gépi reprezentációja
 - Egész, lebegőpontos, negatív, ...
- Eszköz
 - Számrendszer ismerete
 - Véges pontosságú számok fogalma
 - IEEE 754-es lebegőpontos szabvány

Számrendszer

- Tízes számrendszer (*decimális*)
 - Hétköznapi életben ezt használjuk leggyakrabban
 - 0-9 számjegyek (10 darab)
 - Hatványkifejezés alapszáma (*radix*) a 10



$$\text{számérték} = \sum_{i=-k}^n d_i \cdot 10^i$$

$$1 \cdot 100 + 5 \cdot 10 + 2 \cdot 1 + \\ 3 \cdot 0,1 + 9 \cdot 0,01 + 6 \cdot 0,001$$

Például: **1 5 2 . 3 9 6**

Számrendszerök

- **Kettes számrendszer (bináris)**
 - Kétféle számjegy: **0** és **1** (radix: 2)
 - Megfeleltethető a biteknek, emiatt a digitális számítógép tkp. ezt használja
- **Tizenhatos számrendszer (hexadecimális)**
 - 16-féle számjegy: **0–9** és **A–F** között (radix: 16)
 - Egy 4 számjegyű bináris szám egy darab 16-osbeli ábrázolható
 - **Bináris tömörebb leírási módja!**
- **Nyolcas számrendszer (oktális)**
 - Nyolcféle számjegy: **0–7** között (radix: 8)
 - Egy 3 számjegyű bináris szám egy darab nyolcas számrendszerbelivel leírható

Két informatikus barkochbázik:

- Gondoltam egy számra!
- Egy?
- Nem.
- Akkor nulla!

$$d_n \dots d_2 \ d_1 \ d_0 \ . \ d_{-1} \ d_{-2} \dots d_{-k}$$

$$\text{decimális_számérték} = \sum_{i=-k}^n d_i \cdot \text{radix}^i$$

Számrendszerk példa

Bináris

$$\begin{array}{ccccccccccccc}
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
 & 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
 & 1024 + 512 + 256 + 128 + 64 + 0 + 16 + 0 + 0 + 0 + 0 + 0 + 1
 \end{array}$$

Oktális

$$\begin{array}{cccc}
 & 3 & 7 & 2 & 1 \\
 & 3 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 \\
 & 1536 + 448 + 16 + 1
 \end{array}$$

Decimális

$$\begin{array}{cccc}
 & 2 & 0 & 0 & 1 \\
 & 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 \\
 & 2000 + 0 + 0 + 1
 \end{array}$$

Hexadecimális

$$\begin{array}{cccc}
 & 7 & D & 1 & . \\
 & 7 \times 16^2 + 13 \times 16^1 + 1 \times 16^0 \\
 & 1792 + 208 + 1
 \end{array}$$

A 2001 egész szám többféle számrendszerben kifejezve

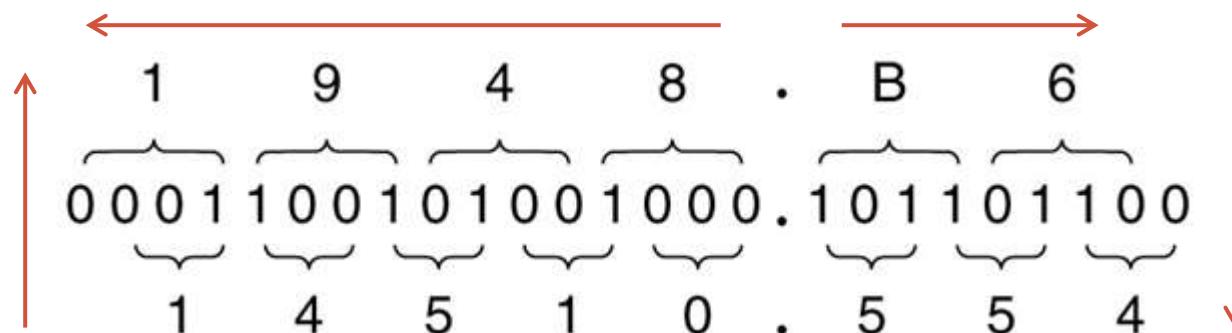
Számrendszerk példa

Szám	Bináris	Oktális	Hexa
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	3	3
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E

15	1111	17	F
16	10000	20	10
20	10100	24	14
30	11110	36	1E
40	101000	50	28
50	110010	62	32
60	111100	74	3C
70	1000110	106	46
80	1010000	120	50
90	1011010	132	5A
100	11001000	144	64
1000	1111101000	1750	3E8
2989	101110101101	5655	BAD

Átváltások számrendszer között

- Bináris-oktális, bináris-hexadecimális átváltás
 - Bináris számjegyek négyes csoportja 1 hexadecimális számjegy
 - Bináris számjegyek hármás csoportja 1 oktális számjegy
 - Csoportosítás a tizedesponttól indul
 - Egész értékeknél balra, tört értékeknél jobbra
 - Vezető és záró 0 számjegyek szükségesek lehetnek



- Oktális-hexadecimális átváltás
 - Pi. a bináris számrendszer felhasználásával

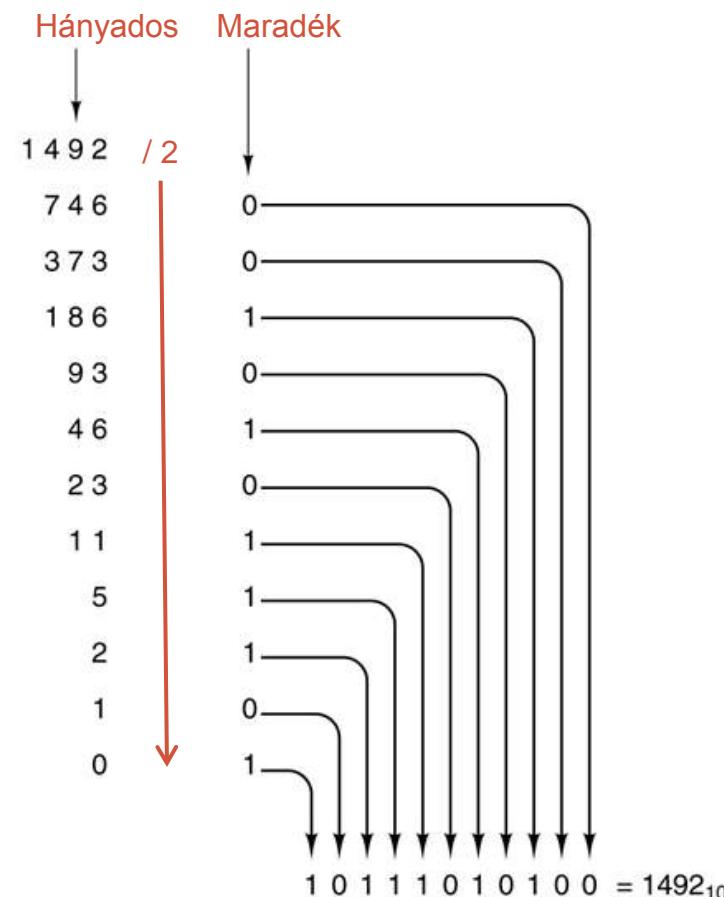
Átváltások számrendszer között

- Egész érték decimálisból binárisba

- A decimális értéket osztjuk 2-vel
- Feljegyezzük
 - a hányados egész részét
 - és a maradékot
- Ismételjük, míg a hányados 0 nem lesz
- A maradék értékek adják a bináris számjegyeket
 - Az első osztás a legkisebb helyiértékű számjegyet adja

- Más számrendszerbe

- A számrendszer alapszámával végezzük az osztást



Átváltások számrendszer között

- Tört érték decimálisból binárisba
 1. A decimális értéket szorozzuk 2-vel
 2. Ha 1, akkor 1-es számjegyet írunk és kész
 3. Ha 1-nél nagyobb, akkor 1-es számjegyet írunk, kivonunk 1-et, és folytatjuk az 1. lépéssel
 4. Ha 1-nél kisebb, akkor 0-s számjegyet írunk és folytatjuk az 1. lépéssel
 - Az első szorzás a legnagyobb helyiértékű (2^{-1}) számjegyet adja
 - Nem biztos, hogy véges lesz!

Átváltások számrendszer között

- Tört átváltás decimálisból binárisba (példák)

Helyérték	$0,3125 \text{ (10)} = 0,0101 \text{ (2)}$	$0,26 \text{ (10)} = 0,0100001\dots$
$2^{-1} = 0,5$	$0,625 = 2 * 0,3125 \rightarrow 0$	$0,52 = 2 * 0,26 \rightarrow 0$
$2^{-2} = 0,25$	$0,25 = 2 * 0,625 - 1 = 1,25 - 1 \rightarrow 1$	$0,04 = 2 * 0,52 - 1 = 1,04 - 1 \rightarrow 1$
$2^{-3} = 0,125$	$0,5 = 2 * 0,25 \rightarrow 0$	$0,08 \rightarrow 0$
$2^{-4} = 0,0625$	$0 = 2 * 0,5 - 1 = 1 - 1 \rightarrow 1$	$0,16 \rightarrow 0$
$2^{-5} = 0,03125$		$0,32 \rightarrow 0$
$2^{-6} = 0,015625$		$0,64 \rightarrow 0$
$2^{-7} = 0,0078125$		$0,28 = 1,28 - 1 \rightarrow 1$
...	Véges!	...

Átváltások számrendszer között

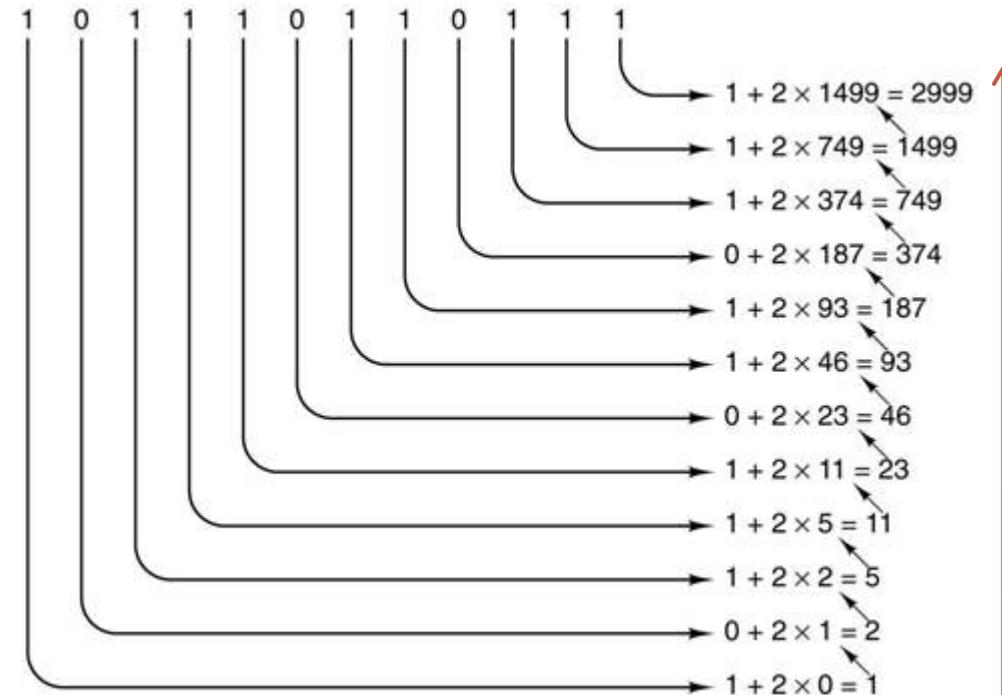
- Binárisból decimálisba
 - A $\sum_{-k}^n d_i \cdot 2^i$ számítási mód sok felesleges számítást tartalmaz
 - A hatványértékek számítása összevonható
 - Optimális: Horner-elrendezés

Forrás: Tanenbaum

Példán keresztül:

101110110111 bináris szám
átváltása decimálisba (2999)

1. Legmagasabb helyiértékű bináris számjegytől indulunk
2. Megszorozzuk 2-vel és hozzáadjuk a következő bináris számjegy értékét
3. A 2. lépést ismételjük, míg vannak számjegyek



Véges pontosságú számábrázolás

- **Digitális ábrázolás**
 - A központi memória véges számú rekeszből áll
 - A hatékony munkához: tárolás rögzített számú bájton
 - Csak adott értéktartományt tudnak reprezentálni
 - Számok véges pontossággal ábrázolhatók
- **Véges pontosságú számok jellemzői**
 - Új problémák
 - Túlcordulás
 - művelet eredménye nagyobb a legnagyobb ábrázolhatónál
 - Alulcsordulás
 - művelet eredménye kisebb a legkisebb ábrázolhatónál
 - Példa: három számjeggyel ábrázolható egész számok: 000, ..., 999
 - $600 + 600 = 1200$ (nem ábrázolható, túl nagy)
 - $003 - 005 = -2$ (nem ábrázolható, túl kicsi)
 - $007 / 002 = 3.5$ (nem ábrázolható, nem egész)
 - $a + (b - c)$ nem lesz egyenlő $(a + b) - c$ értékével,
ha pl. $a = 700$, $b = 400$, $c = 300$ és az alul- és túlcordulást nem kezeljük

Egész számok ábrázolása

- Nem negatív egész számok
 - Bitek helyiértékének megfelelően
 - 1 bájtos tárolás: 0 - 255 (2^8) közötti értékek
 - 2 bájtos tárolás: 0 - 65,535 (2^{16}) közötti értékek
 - 4 bájtos tárolás: 0 - 4,294,967,295 (2^{32}) közötti értékek
- Binárisan kódolt decimális (BCD)
 - 4 (pakolt) vagy 8 bit (pakolatlan) egy decimális számjegyet jelöl
 - 100 darab (4-bites reprezentáció) vagy 10 darab (8-bites) különböző szám ábrázolható így
 - Egyes bájt értékek nem hordoznak jelentést (pl. 1011, pazarló)
 - Példák
 - 19 pakolt BCD kódolással: 0001 1001 = 19H (decimális 25)
 - 73 pakolt BCD kódolással: 0111 0011 = 73H (decimális 115)
 - 199 pakolt BCD kódolással: 0000 0001 1001 1001 = 199H (dec. 409)

Számábrázolás

- Előjeles (negatív és nem negatív) egész számok
 - Előjeles abszolút érték
 - Legfelső bit: előjel (0 pozitív, 1 negatív)
 - További bitek: szám abszolút értéke
 - Probléma: létezik +0 és -0 nulla is!
 - Példák
 - **00000001:** +1, **00000000:** +0, **10000000:** -0, **11111111:** -127, **01111111:** +127
 - Egyes komplementens
 - Legfelső bit: előjel (0 pozitív, 1 negatív)
 - További bitek: szám értéke, de negatív esetben mindegyik ellentett értékkel (0 és 1 számjegyeket felcseréljük)
 - +0 és -0 probléma itt is fennáll!
 - 00000001: +1, 11111110: -1, 00000000: +0, 11111111: -0, 01111111: +127

Számábrázolás

- Előjeles (negatív és nem negatív) egész számok
 - Kettes komplement
 - Képzése: egyes komplement + 1
 - Ha átvitel jelenik meg a bal szélső bitnél, azt eldobjuk
 - Csak egyféllel 0 érték reprezentáció! (Ez végre jó!)
 - Nem szimmetrikus ábrázolási tartomány: -128 és +127 között
 - 2^{m-1} többletes
 - m bites szám esetén
 - $m = 8$ esetén 128 többletes
 - A többletes értékét hozzáadjuk az ábrázolandó számhoz
 - Negatív számok: kisebbek a többletesnél
 - 0 értéke: többletes értéke
 - Pozitív számok: nagyobbak a többletesnél
 - Azonos a kettes komplementessel fordított előjelbittel
 - Használata: lebegőpontos számok kitevő részénél

Számábrázolás

N decimális	N bináris	-N előjeles absz.ért.	-N előjeles 1-es komplemens	-N előjeles 2-es komplemens	-N 128 többletes
1	00000001	10000001	11111110	11111111	01111111
2	00000010	10000010	11111101	11111110	01111110
3	00000011	10000011	11111100	11111101	01111101
4	00000100	10000100	11111011	11111100	01111100
5	00000101	10000101	11111010	11111011	01111011
6	00000110	10000110	11111001	11111010	01111010
7	00000111	10000111	11111000	11111001	01111001
8	00001000	10001000	11110111	11111000	01111000
9	00001001	10001001	11110110	11110111	01110111
10	00001010	10001010	11110101	11110110	01110110

Számábrázolás

N decimális	N bináris	-N előjeles absz.ért.	-N előjeles 1-es komplemens	-N előjeles 2-es komplemens	-N 128 többletes
20	00010100	10010100	11101011	11101100	01101100
30	00011110	10011110	11100001	11100010	01100010
40	00101000	10101000	11010111	11011000	01011000
50	00110010	10110010	11001101	11001110	01001110
60	00111100	10111100	11000011	11000100	01000100
70	01000110	11000110	10111001	10111010	00111010
80	01010000	11010000	10101111	10110000	00110000
90	01011010	11011010	10100101	10100110	00100110
100	01100100	11100100	10011011	10011100	00011100
127	01111111	11111111	10000000	10000001	00000001
128	Nem ábrázolható	Nem ábrázolható	Nem ábrázolható	10000000	00000000

Bináris aritmetika

- Bináris számok összeadása
 - Decimális számrendszerbeli összeadással analóg módon

Első tag	0	0	1	1
Második tag	0	1	0	1
Összeg	0	1	1	0
Átvitel	0	0	0	1

Példa

- Jobbról balra haladunk
- Átvitelt figyelembe vesszük
- Utolsó átvitel kezelése
 - 1-es komplementsnél hozzáadjuk a jobb szélső bithez
 - 2-es komplementsnél eldobjuk

	1-es komplement	2-es komplement
$ \begin{array}{r} 10 \\ + (-3) \\ \hline \end{array} $	$ \begin{array}{r} 00001010 \\ 11111100 \\ \hline \end{array} $	$ \begin{array}{r} 00001010 \\ 11111101 \\ \hline \end{array} $
$ \begin{array}{r} +7 \\ \hline \end{array} $	$ \begin{array}{r} 1\ 00000110 \\ \searrow \\ \hline \end{array} $	$ \begin{array}{r} 1\ 00000111 \\ \downarrow \\ \hline \end{array} $
	átvitel hozzáadása jobbról	átvitel eldobása
		00000111

Bináris aritmetika

- Bináris számok összeadása
 - Ha a két tag ...
 - ... ellentétes előjelű, akkor nem lesz túlcsordulás.
 - ... egyező előjelű, de az eredmény más előjelű, akkor túlcsordulás következett be és az eredmény rossz!
 - Legtöbb esetben megtartásra kerül az előjelbitnél bekövetkezett átvitel
 - Pl. Speciális flag regiszter bitben
 - Így kezelhető az átviteli probléma
 - Jellemzően csak alacsony szinten (gépi kód, Assembly) érhető el, magasszintű programozási nyelvekben nem!

Lebegőpontos számok

- Tört számok reprezentációja
 - $n = f \cdot radix^e$
 - f: törtrész (*fraction*) vagy mantissa
 - e: kitevő vagy exponens
 - radix: alkalmazott számrendszer
 - Számítógépen jellemzően 2-essel dolgoznak 10-es helyett!
- Többféle reprezentáció lehetséges
 - $3,14 = 0,314 \cdot 10^1 = 3,14 \cdot 10^0$
 - $0,000001 = 0,1 \cdot 10^{-5} = 1,0 \cdot 10^{-6}$
- Normalizálás: megegyezés alapján; pl. a tizedesvessző mellett jobbra (vagy balra) legyen az első nem 0 számjegy
 - Példák (2-es alapú hatvány)
 - $432 = 01010100|000000000011011$ (előjel, 64-többletes kitevő: $84-64=20$, normalizálatlan törtrész, $2^{20} \cdot (2^{-12} + 2^{-13} + 2^{-15} + 2^{-16}) = 432$)
 - $432 = 01001001|1101100000000000$ (előjel, 64-többletes kitevő: $73-64=9$, normalizált törtrész, $2^9 \cdot (2^{-1} + 2^{-2} + 2^{-4} + 2^{-5}) = 432$)
 - Normalizáláskor 11 helytel a törtrészt, amit a kitevőnél kompenzáltunk

Lebegőpontos számok

- Tört számok reprezentációja
 - *Nagyságrend*
 - Kitevő jegyeinek száma határozza meg
 - *Pontosság*
 - Törtrész jegyeinek száma határozza meg
 - A valós számhalmaz csak egy véges részét lehet reprezentálni!
 - Mivel két valós szám között végtelen sok további valós szám található
 - Összehasonlító műveletek esetén a különbségük nagyságát nézzük
 - Ha kisebb egy megadott értéknél, akkor egyenlőnek vesszük
 - Az ábrázolási pontosság nem egyenletes
 - Pl. nagyobb abszolútértékű számok esetén pontatlanabb

Lebegőpontos számok

- IEEE 754-es lebegőpontos szabvány (1985)

- Kettes hatványalapot használ
- Egyszeres pontosság (32 biten)
 - 1 előjelbit, 8 bites kitevőrész (127-többletes ábrázolással), 23 bites törtrész
 - Decimális kiterjedés: kb. 10^{-38} -tól 10^{38} -ig
- Dupla pontosság (64 biten)
 - 1 előjelbit, 11 bites kitevőrész (1023-többletes ábrázolással), 52 bites törtrész
 - Decimális kiterjedés: kb. 10^{-308} -tól 10^{308} -ig
- Kiterjesztett pontosság (80 bit)
 - Lebegőpontos aritmetikai egység belsejében a kerekítési hibák csökkentésére

`float` (C, Java, ...)

`double` (C, Java, ...)

Lebegőpontos számok

- Numerikus típusok

- Normalizált

\pm	$0 < \text{kitevőrész} < \text{max}$	Bitminta
-------	--------------------------------------	----------

- Nem normalizált

\pm	0	Nem 0 bitminta
-------	---	----------------

- Nulla

\pm	0	0
-------	---	---

- Végtelen

\pm	111...1	0
-------	---------	---

- Nem szám

\pm	111...1	Nem 0 bitminta
-------	---------	----------------

Előjel

Kitevőrész

Szignifikáns (~törtrész)

Egyeszeres pontosság

- Normalizált számok

- 1 előjelbit, 8 bites kitevőrész (127 többletes), 23 bites törtrész
- Ha $0 < \text{kitevőrész} < 255$, akkor a szám *normalizált*
- A *normalizált* szám első jegye (tizedesvesszötől balra) 1, így az nincs ábrázolva!
 - Törtrész helyett ezt *szignifikánsnak* hívjuk, hogy ne keverjük a fogalmakat
- Ábrázolt szám: $\pm(1 + \text{szignifikáns}) \cdot 2^{\text{kitevő}}$, ahol $\text{kitevő} = \text{kitevőrész} - 127$
- Példák

+1,0:	$0011\ 1111\ 1000\ ... \ 0000_2$	$= 3F80\ 0000_{16}$
+0,5:	$0011\ 1111\ 0000\ ... \ 0000_2$	$= 3F00\ 0000_{16}$
-1,5:	$1011\ 1111\ 1100\ ... \ 0000_2$	$= BFC0\ 0000_{16}$

± kitevő- 1. szigni-
 rész fikáns

Egyeszeres pontosság

- További példák

+15,0: 0100 0001 0111 ... 0000₂ = 4170 0000₁₆

- Előjel: pozitív

- Kitevő: 130 – 127 = 3

- Szignifikáns: 1 + 0,5 + 0,25 + 0,125 = 1,875

- Eredmény: 1,875 · 2³ = 15

-15315,0: 1100 0110 0110 1111 0100 1100 0000 0000

- Előjel: negatív

- Kitevő: 140 – 127 = 13

- Szignifikáns: 1 + 0,5 + 0,25 + 0,0625 + 0,03125 + 0,015625 + 0,0078125 + 0,001953125 + 0,000244140625 + 0,0001220703125

- Eredmény: -1,8695068359375 · 2¹³ = -15315

Egyeszeres pontosság

- **Nem normalizált számok**

- Ha a *kitevőrész* = 0
 - A *normalizálatlan* szám első jegye 0, így az nincs ábrázolva!
 - Ábrázolt szám: $\pm(0 + \text{törtrész}) \cdot 2^{\text{kitevő}}$, ahol *kitevő* = –126 rögzítetten!
 - Segítségével kezelhető az alulcsordulás: a *normalizáltnál* kisebb értékeket reprezentálhatunk
 - Legkisebb *normalizált* szám: $1,0 \cdot 2^{-126}$
 - Legnagyobb *nem normalizált* szám: $0,9999999 \cdot 2^{-126}$
 - Legkisebb *nem normalizált* szám: $1,0 \cdot 2^{-149}$
 - A tizedesvesszötől tovább jobbra tudjuk tolni a törtrészt, amit a normalizált forma nem engedett

Egyszeres pontosság

- Végtelen és érvénytelen szám (`NaN`) ábrázolására
 - Ha a *kitevőrész* = 255
 - Végtelen: csupa 0 törtrész bitminta
 - Érvénytelen szám: csupa 0-tól eltérő törtrész bitminta
 - `NaN`: *Not a Number*
 - Aritmetika
 - Szám / 0 = végtelen
 - Szám / végtelen = 0
 - Végtelen + bármi = végtelen
 - 0 / 0 = `NaN`
 - ...

Bájtsorrend

- **Mostanra ismerjük**
 - Központi memória rekeszes felépítését
 - Bájtos rekesz-felépítést feltételezünk a továbbiakban
 - Számok bájtos reprezentációját
- **Kérdés**
 - Ezek a reprezentációk hogyan jelennek meg a memóriában?
- **(Meglepő) válasz**
 - **A több bájton reprezentált értékek memóriabeli megjelenése architektúra függő!** Számít:
 - Processzor regiszterszélessége (16, 32, vagy 64 bites)
 - Magas („big”) vagy alacsony („little”) helyiértékek bájtjai szerepelnek-e az alacsonyabb memóriacímeken
 - Nagy endián (big endian) rendszerek: pl. SPARC, IBM nagygépek
 - Kis endián (little endian) rendszerek: pl. Intel processzorok

Bájtsorrend

- Példa

```
typedef struct {
    int darabszam;
    short min;
    short max;
    float atlag;
    float szoras;
} statisztika;

...
statisztika stat1;

stat1.darabszam = 10;
stat1.min = -20;
stat1.max = 120;
stat1.atlag = 43.2;
stat1.szoras = 5.3;
```

Memóriatartalom

- 32 bites *kis endián* rendszeren
0a 00 00 00 ec ff 78 00 cd cc 2c 42 9a 99 a9 40
- 32 bites *nagy endián* rendszeren
00 00 00 0a ff ec 00 78 42 2c cc cd 40 a9 99 9a

Probléma

- Eltérő rendszerek közötti bináris adatátvitel nem kompatibilis!

Megoldási lehetőségek

- Szöveges kódolás
 - (pl. XML, JSON)
- Bináris kódolás
 - 2 bájtos kód elhelyezése az adat elején, amiből az endián típus meghatározható (pl. ,1' short értéke)
 - Ha szükséges, a bájtokat a memóriában fel kell cserélni beolvasás után

Szöveg kódolása

- **ASCII (1963)**

- A bájtok értékei karaktereket jelentenek
 - Latin ábécé betűi + írásjelek + vezérlőkódok
- Eredetileg az alsó 7 bit használatos
 - 0-31: vezérlő karakterek (pl. új sor, ESC, szöveg vége)
 - 32: szóköz
 - 33-47: írásjelek
 - 48-57: számjegyek
 - 58-64: nagybetűk
 - 91-96: írásjelek
 - 97-122: kisbetűk
 - 123-126: írásjelek
 - 127: DEL

ASCII Code Chart															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
~	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Ábra forrása: Wikipedia

Szöveg kódolása

- **Kiterjesztett ASCII**

- 128-255 közötti értékek is definiáltak (1963)
 - pl. nemzeti karakterek, szimbólumok
- Probléma
 - Sok hiányzó karakter (pl. magyar Ő, Ś)
- Megoldás (nem tökéletes)
 - ISO 8859 kódolási szabvány (1987)
 - A különféle kiterjesztett karakterkészletek definiálására
 - ISO 8859-1, ... , ISO8859-16
 - ISO 8859-1: nyugat-európai karakterkészlet
 - ISO 8859-2: nyugat- és kelet-európai karakterkészlet (pl. magyar is)
 - ISO 8859-3: nyugat- és dél-európai karakterkészlet (pl. török, máltai + eszperantó)
 - ...
 - Kínai, japán, karaktereket ebbe is lehetetlen belezsúfolni

OEM Extended ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	ã	ß
9	é	æ	Œ	ô	ö	ð	û	ù	ÿ	ö	Ü	¢	£	¥	₹	ƒ
A	á	í	ó	ú	ñ	Ñ	œ	œ	ç	œ	¬	%	¤	฿	«	»
B	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
C	↳	↳	↳	↳	↳	↳	↳	↳	↳	↳	↳	↳	↳	=	♫	±
D	¤	⌐	π	¤	€	ƒ	π		÷	„	„	„	„	■	■	■
E	¤	฿	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪
F	≡	±	≥	≤	ƒ	J	÷	≈	°	-	-	√	n	z	█	

Ábra forrása: cplusplus.com

Szöveg kódolása

- **UTF (Unicode) és ISO/IEC 10646 szabvány (1989-1990)**
 - Maximálisan 2,147,483,648 különböző karakter leírására
 - 128 csoport x 256 sík x 256 sor x 256 cella
 - **UTF-8**
 - Változó hosszúságú (elvileg 1-6, valójában 1-4 bájtos) tárolás
 - Alsó hét bit: ASCII kódtábla
 - További kódok
 - Vezető bajt definiálja a követő bajtok számát (legfelső bitek 1 értékeivel)
 - A követő bajtok felső bitértékei rögzítettek (10)
 - U+10FFFF korlát (2003)
 - Max. 4 bájtos az UTF-8
 - 1,112,064 karakter
 - Népszerű kódolás

Bits	Last code point	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+007F	0xxxxxxxx					
11	U+07FF	110xxxxx	10xxxxxx				
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
21	U+1FFFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+3FFFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Szöveg kódolása

- **UTF (Unicode) és ISO/IEC 10646 szabvány (1989-1990)**
 - **UTF-16**
 - Rögzített 16 bites (2 bájtos) tárolás
 - 0 és hexa 10FFFF közötti érvényes kódok
 - 1,112,064 karakter kódolására képes
 - Bájtok sorrendje függ az architektúrától!
 - Kis- vagy nagy-endián
 - Opcionális BOM (bájt sorrend jelölő - *Byte order mark*) a szöveg elején
 - U+FEFF érték
 - Ebből megállapítható a bájtsorrend
 - Windows 2000 óta a Windows API használja
 - **UTF-32**
 - Rögzített 32 bites (4 bájtos) tárolás
 - 0 és hexa 7FFFFFFF közötti érvényes kódok (2,147,483,648 lehetséges kód)
 - Bájtsorrend itt is problémás!
 - Szövegek helyett inkább csak 1-1 karakter leírására használják

Hibadetektáló és -javító kódok

- A memória hibázhat
 - A rekeszben tárolt érték nem programozottan megváltozik
 - Pl. Áramlökés, hibás hardver hatására
- Hatása
 - Adat területen: számítási hiba keletkezhet
 - Kód területen: hibás kód keletkezhet, ha végrehajtódi, „lefagyhat” a program/rendszer
- Kivédése
 - Redundancia bevezetésével
 - *Detekció (felismerés)*
 - Bizonyos hibákat a rendszer „észrevesz”
 - Hibaüzenettel megállíthatja a program futását
 - *Hibajavítás*
 - Bizonyos hibák esetén a legvalószínűbb helyes bájtérték helyreállítható

Hibadetektáló és -javító kódok

- **Egyszerű példa: Paritásbit alkalmazása**
 - minden bájthoz plusz 1 bitet (paritásbit) hozzáveszünk
 - így a memória 1/9-ed része nem tárol érdemi adatot
 - A paritásbit a bájt értékben szereplő 1 értékek párosságát mutatja
 - másként: biztosítja, hogy a 9 biten pl. minden páros számú 1 érték legyen (lehetne páratlan is, ez megegyezés kérdése)
 - Detekció
 - Ha a paritásbit értéke nem megfelelő, akkor memóriahiba fordul elő
 - Példák
 - 00000000 **0**, 00001000 **1**, 11010011 **1** (érvényes kódok)
 - 000**1**0000 **0**, 000**1**1000 **1**, 11**10**0011 **0** (érvénytelen kódok)
 - 00**11**0000 **0**, 000**10**0000 **1**, **00000111** **1** (érvényes kódok)
 - Alkalmazhatósága
 - **Csak páratlan számú bitibát tud detektálni** (tehát nem minden!)
 - **Javításra nem képes**

Hibadetektáló és -javító kódok

- Definíciók

- m adat bit, r ellenőrző bit esetén $n = m + r$ bites a **kódszó**
- Hamming-távolság
 - Két bináris szám eltérő számjegyeinek száma
 - 1000101
 - 1011001 Hamming-távolságuk 3

- Detekciós és javító képesség

- 2^n kódszó közül csak 2^m darab érvényes
 - Összes érvényes kódszó listája: meghatározható az ellenőrző bitek számítási algoritmusának ismeretében
 - Összes kód Hamming-távolsága: az előző listában szereplő kódok közötti legközelebbi Hamming-távolság
- d egyszeres bithiba felismeréséhez $d + 1$ távolságú kódolás kell
- d egyszeres hiba javításához $2 \cdot d + 1$ távolságú kódolás kell
 - Hiba esetén: a legközelebbi Hamming-távolságú érvényeset vesszük

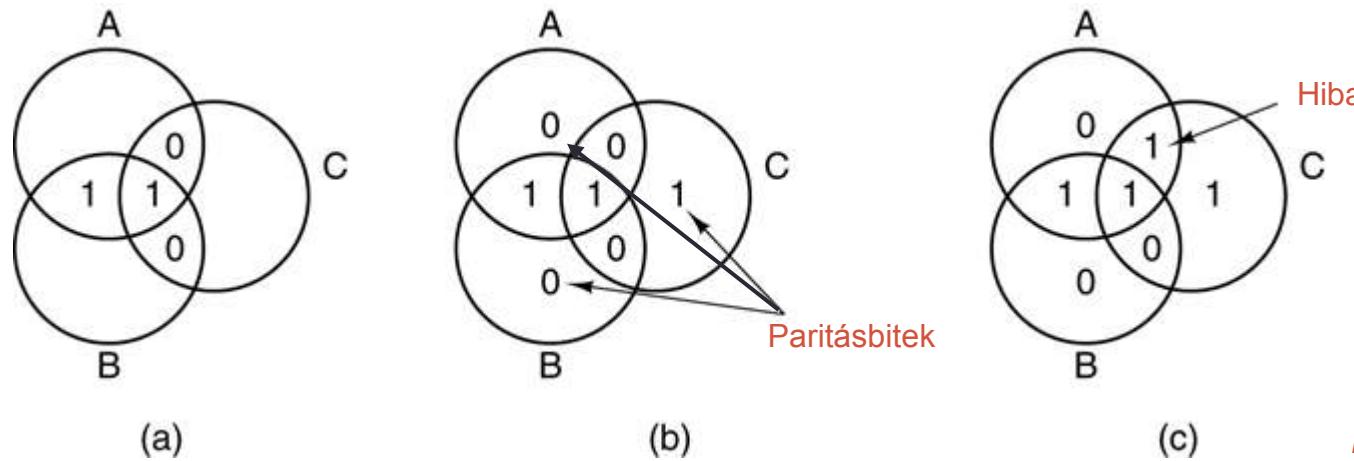
Hibadetektáló és -javító kódok

- Kódolás egyszeres bithiba *javítására*
 - m adat bit, r ellenőrző bit, $n = m + r$ bites kódszó
 - Alsó határ egyszeres bithiba javításához: $(m + r + 1) \leq 2^r$
 - (Részleteket *Id. a Tanenbaum könyvben*)
 - Nagyobb m érték esetén rövidebb kód, de időigényesebb ellenőrzés!

Szó hossza	Ellenőrző bitek	Teljes hossz	Hozzáadott bitek százaléka
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

Hibadetektáló és -javító kódok

- Hamming-algoritmus 4 bites szóra



Forrás: Tanenbaum

- 1100 érték kódolása
- Paritásbitek számítása (A, B, C körökben páros számú 1 érték legyen)
- Egyszeres hiba megjelenése: A és C körökben elrontja a paritást. Javítás: ezen körök metszetében bit inverzió

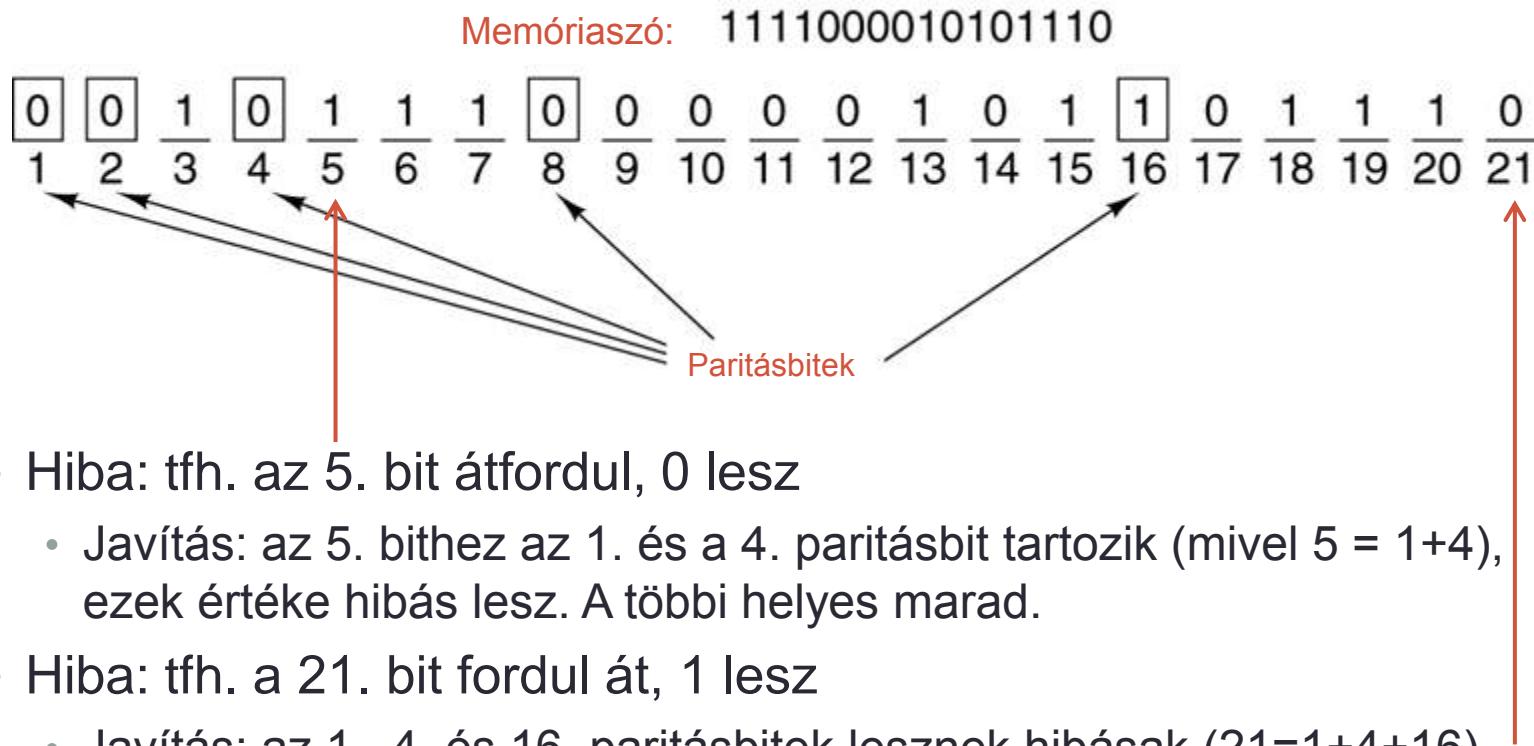
Hibadetektáló és -javító kódok

- **Hamming-algoritmus**

- m adat bit, r ellenőrző bit, $n = m + r$ bites kódszó
- Sorszámozási konvenció
 - Biteket 1-gyel kezdődően sorszámozzuk, a legnagyobb helyértékű az 1-es
- Paritásbitek
 - Amelyek sorszáma 2 egész kitevős hatványa ($1, 2, 4, 8, 16, \dots$)
 - minden paritásbit meghatározott bitpozíciókat ellenőriz
 - A b . bitet azok a b_1, b_2, \dots, b_j paritásbitek ellenőrzik, amelyekre $b_1 + b_2 + \dots + b_j = b$
 - Értéke: biztosítja, hogy az ellenőrzött pozíciókon az 1-esek száma páros legyen
- Használata
 - Kiszámítjuk az össze paritásbitet, ha mindegyik helyes, akkor nincs javítható hiba
 - A hibás bit sorszámát a hibás paritásbitek sorszámainak összege adja
 - Ezt a bitet ellentettre állítjuk

Hibadetektáló és -javító kódok

- Hamming-algoritmus példa
 - 16 bites szó (+5 ellenőrző paritásbit)



ECC és nem-ECC memóriák

- Megvásárolható PC RAM modulok
 - **Nem-ECC**
 - Nincs hibaellenőrzés
 - Sem paritás, sem korrekció!
 - Olcsóbb
 - A mai memóriamodulok már ritkán hibálnak, ezért ez a gyakoribb
 - **ECC (error correcting code)**
 - Hibadetektálás és javítás
 - 1 bithiba javítása és 2 bithiba detektálása 64 bites szavanként
 - Szerverekbe és kritikus alkalmazásokat kiszolgáló számítógépekbe
 - Alaplapi támogatás szükséges

www.arukereso.hu

[HP 8GB \(2x4GB\) DDR2 667MHz 466440-B21](#)

667 MHz, 8 GB, DDR2



HP (466440-B21) HP 8GB Fully Buffered DIMM PC2-5300 2x4GB Low Power DDR2 Memory Kit Így is ismerheted: 8 GB 2 x 4 GB DDR 2 667 MHz 466440 B 21, 8GB2x4GBDDR2667MHz466440B21, 8GB (2x4GB)...

138 003 Ft

[irány a bolt >](#)

[sénetic.hu](#)

[Összehasonlítás](#)

[Kingston 32GB \(4x8GB\) DDR3 1600MHz D1G72K111K4](#)

1600 MHz, 32 GB, DDR3



KIT kiszerelésben, 4 darabos csomagban kaphatóak Gyors, 32GB kapacitású memóriamodul. ECC regisztrált, integrált hőelterelő. Kifejezetten Szerver alaplapokba ajánlott. DDR3-as technológia...

134 300 Ft

[irány a bolt >](#)

[bluechip.hu](#)

[Összehasonlítás](#)

[Corsair 32GB \(4x8GB\) DDR3 2133MHz CMZ32GX3M4A2133C10](#)

2133 MHz, 32 GB, DDR3



Kit kiszerelés, azaz 4db memóriamodul! Dual Channel: 4 csatornás memóriabusz, a lehető leggyorsabb adatfelvételrész érdekeiben! A Corsair Vengeance DDR3 memóriamodulokat a tuningolókat szem...

130 000 Ft

[irány a bolt >](#)

[bluechip.hu](#)

[Összehasonlítás](#)

[Fujitsu 16GB DDR3 1066MHz S26361-F3604-L516](#)

1066 MHz, 16 GB, DDR3



Fujitsu S26361-F3604-L516 RAM Module - 16 GB - DDR3 SDRAM - 1066 MHz DDR3-1066/PC3-8500 - Registered Így is ismerheted: 16 GB DDR 3 1066 MHz S 26361 F 3604 L 516, 16GBDDR31066MHzS26361F3604L516,...

127 688 Ft

[irány a bolt >](#)

[webshop.a4team.hu](#)

[Összehasonlítás](#)

KIT kiszerelésben, 4 darabos csomagban kaphatóak Gyors, 32GB kapacitású memóriamodul. ECC regisztrált, integrált hőelterelő. Kifejezetten Szerver alaplapokba ajánlott. DDR3-as technológia. 1600MHz frekvencia sebességgel rendelkezik. minden Kingston szerver memória a legújabb vezető gyártási technológia eredménye. Garancia a minőségre. Stabilitás, gyorsaság, 100%-os kompatibilitás jellemzi. A Kingston a lehető legjobb forrás, ha rendszer specifikus szerver memóriát...

[Kingston 32GB \(4x8GB\) DDR3 1600MHz D1G72K111K4 teljes leírás >](#)

SZÁMÍTÓGÉPES ARCHITEKTÚRÁK

A digitális logika szintje I.

Nagy Antal

Kapuk és Boole-algebra

• Digitális áramkör

- Két logikai érték van jelen
 - 0: 0 és 1 volt között
 - 1: 2 és 5 volt között
- Az intervallumokon kívüli feszültségértékek nem megengedettek

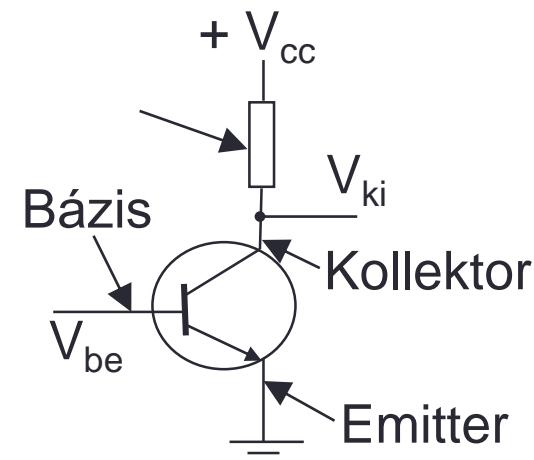
• Kapuk

- Kétértékű jelek különböző függvényei
- Belső működés a -1 szinthez kapcsolódik

• Tranzisztor

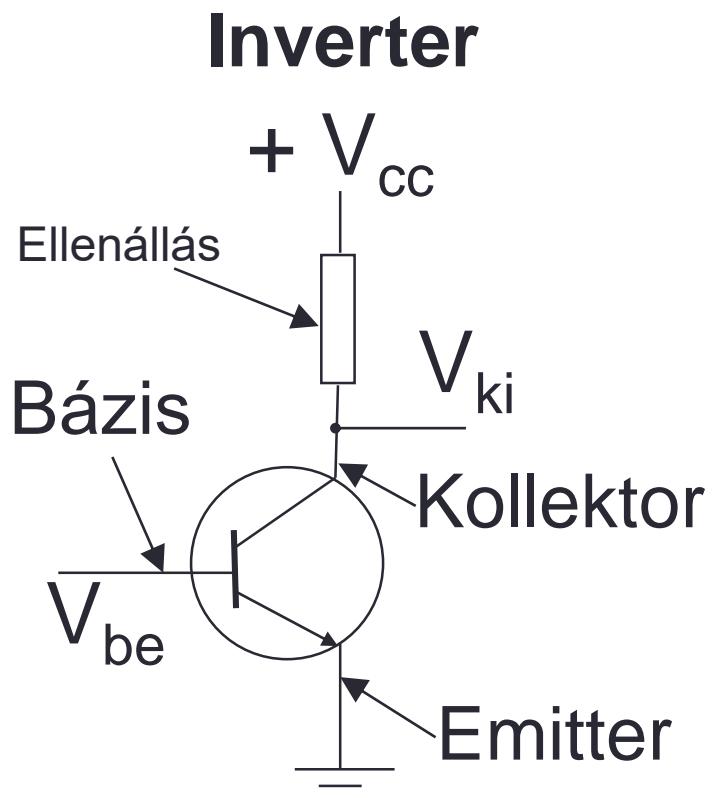
- Kapcsolatok
 - Kollektor
 - Bázis
 - Emitter
- Kapcsolási idő
 - 1 nanoszekundum

Forrás: Wikipedia

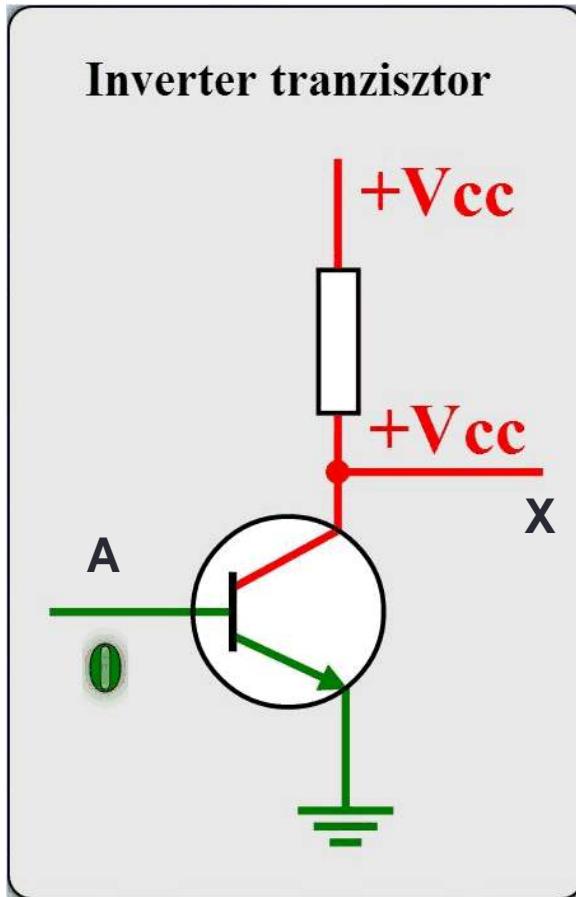


Tranzisztor

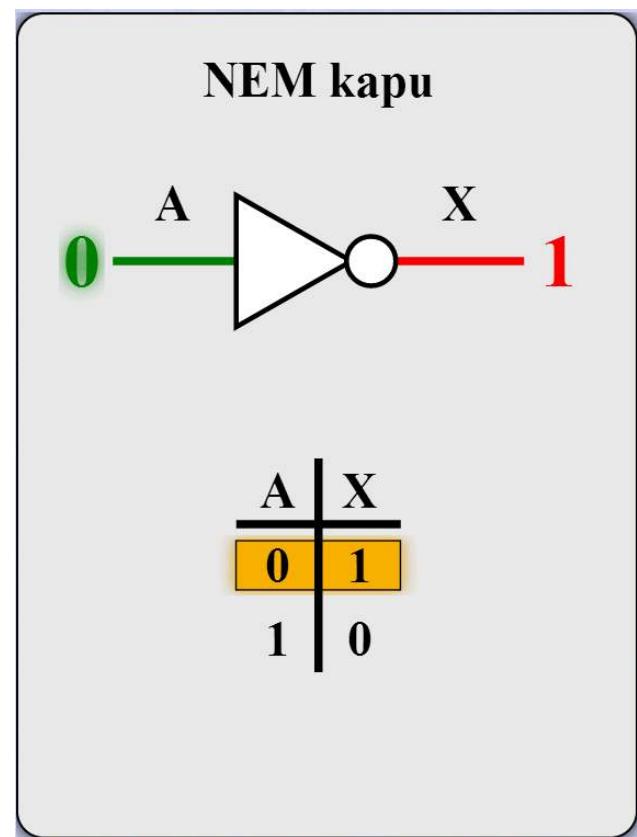
- **Zárt**
 - V_{be} bizonyos szint alatt van
 - Végtelen ellenállás
 - $V_{ki} \approx V_{cc}$
- **Nyitott**
 - V_{be} meghaladja a kritikus értéket
 - Vezeték
 - V_{ki} -t lehúzza a földhöz
 - Lecsökkenti a 0 voltra
- Ha V_{be} alacsony, akkor V_{ki} magas és fordítva



Inverter

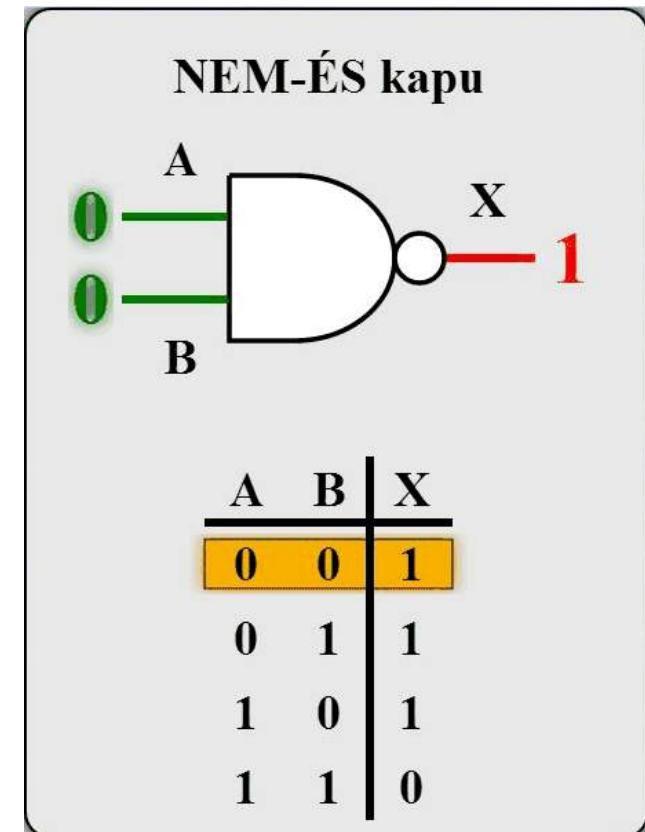
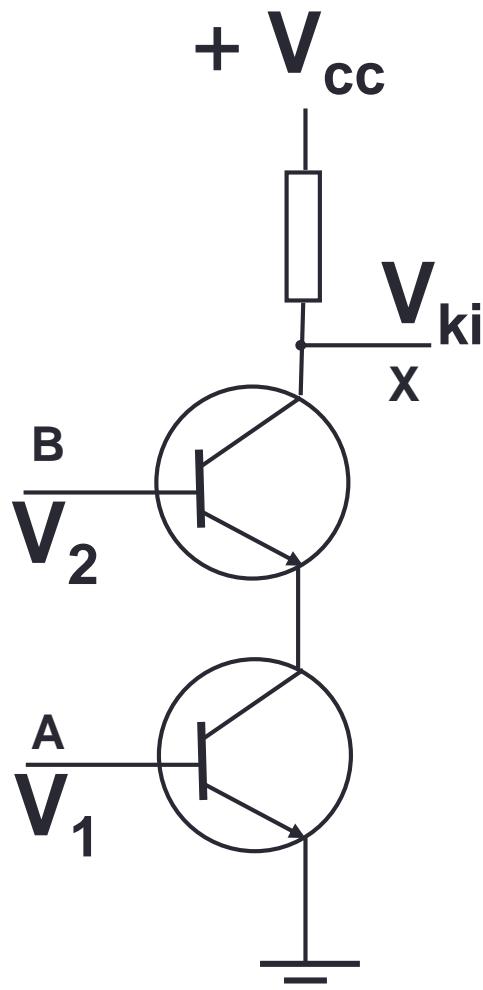


Igazság
tábla



Forrás: Tanenbaum, Gulyás Dénes

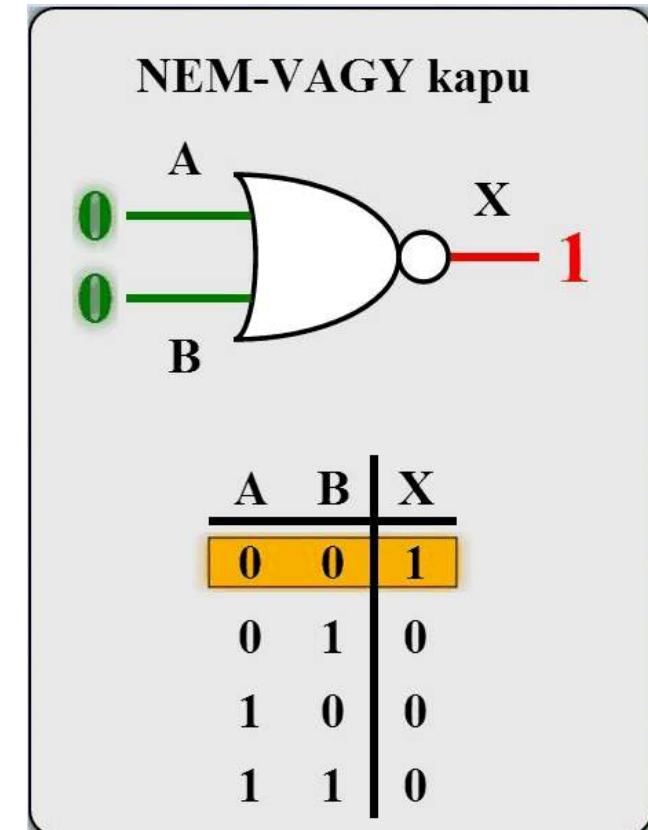
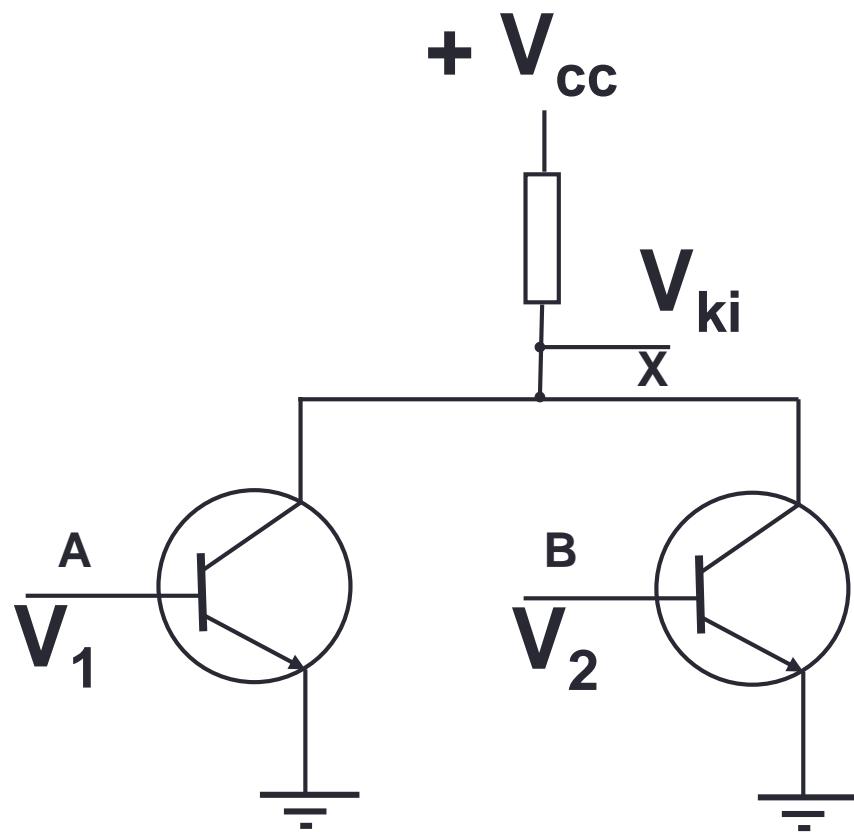
NEM-ÉS (NAND) kapu



Forrás: Tanenbaum, Gulyás Dénes

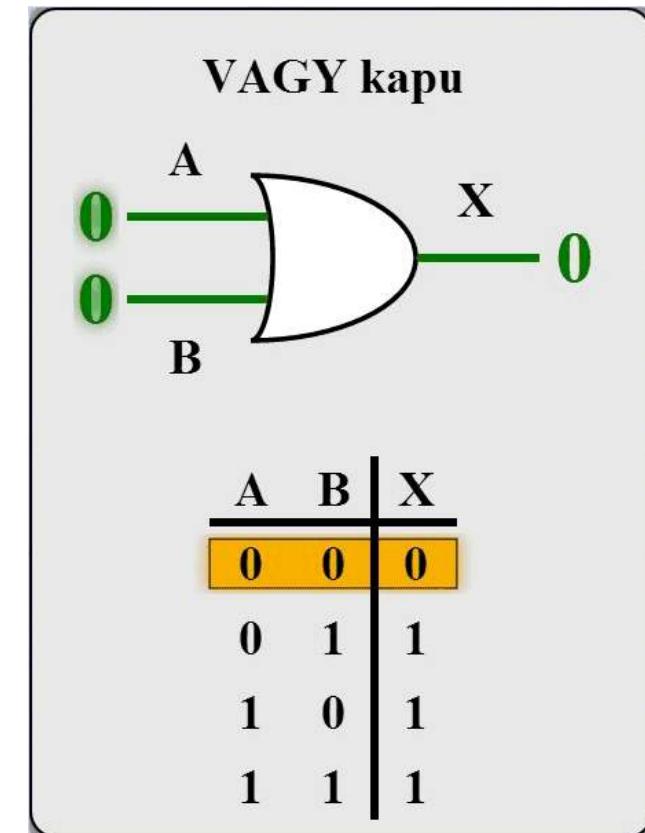
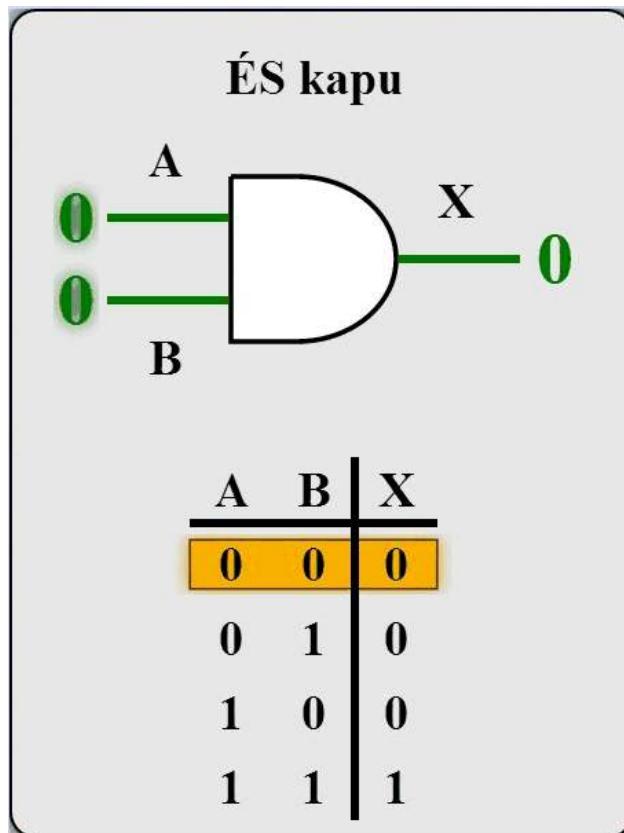
Két tranzisztor

NEM-VAGY (NOR) kapu



Forrás: Tanenbaum, Gulyás Dénes

ÉS (AND), VAGY (OR) kapuk



Forrás: Tanenbaum, Gulyás Dénes

Forrás: Tanenbaum, Gulyás Dénes

három tranzisztor

Boole-algebra

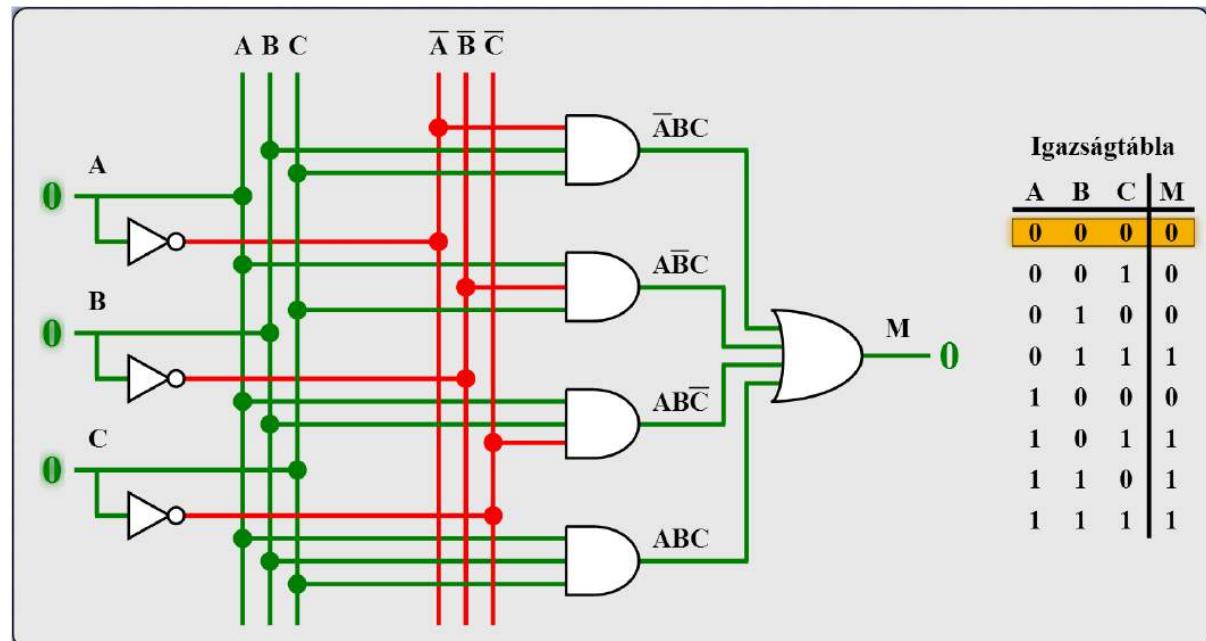
- **Kapuk kombinációjából felépíthető áramkörök leírására**
 - 0 és 1 értékű változók és függvények
 - Kapcsoló algebra
- **Boole-függvények értelmezése**
 - Csak 0 és 1 értékek lehetségesek
 - Egy vagy több bemeneti változó
 - Egy kimeneti érték
 - Függvény definiálása
 - Lehetséges bemenő értékekre megadjuk a kimeneti értéket
 - n bemenő változó esetén 2^n lehetséges kombinációra kell megadni a kimenő értékeket
 - Igazságtáblázat
 - Egy sor megadja adott kombinációjú bemeneti értékek esetén a függvény kimeneti értékét

Boole-algebra

- Többségi logikai függvény

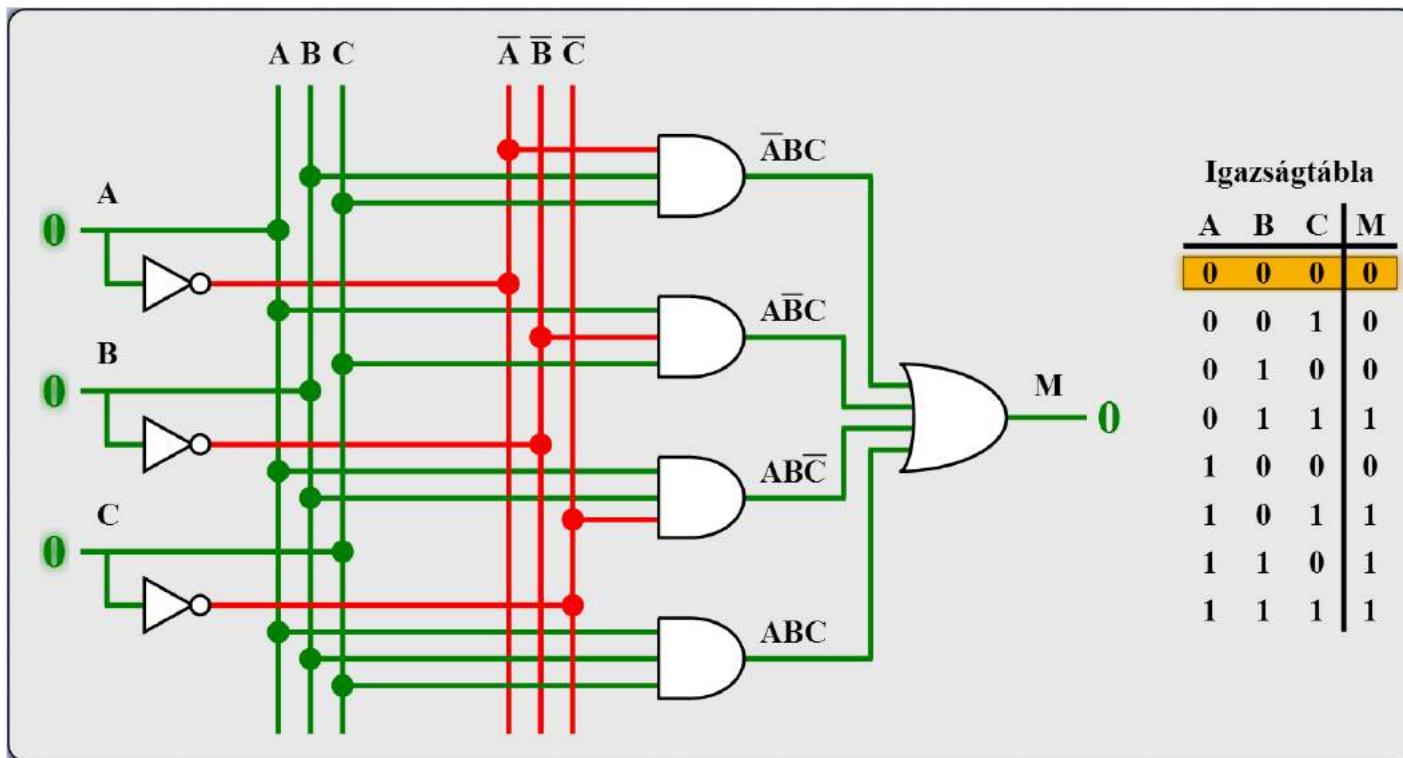
- $M = f(A, B, C)$
- Tetszőleges Boole-függvény megadható igazságtáblázattal
- Változók számának növekedése

Forrás: Tanenbaum, Gulyás Dénes



Boole-algebra

- Más jelölés alkalmazása
 - Diszjunktív normálforma: $M = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$



Boole-algebra

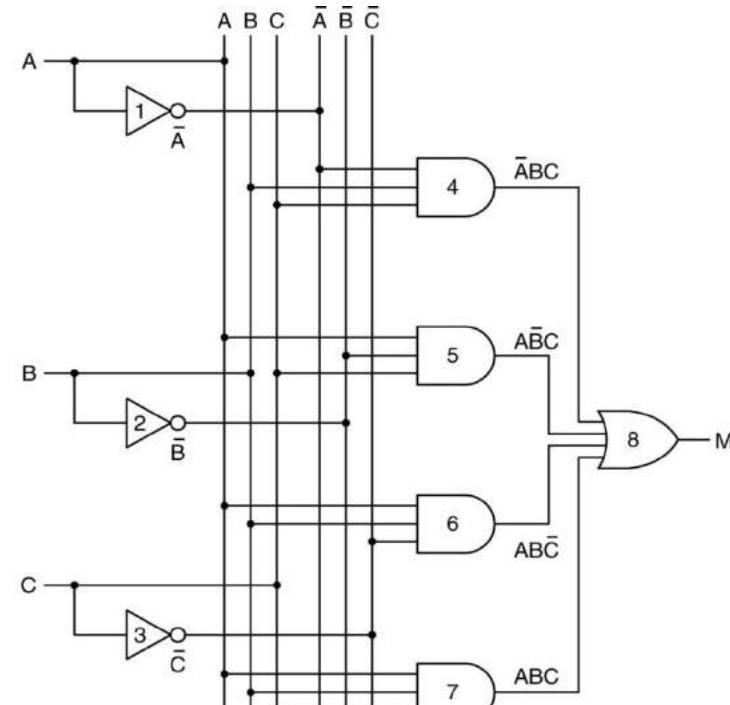
- **n változós függvény**
 - Legfeljebb 2^n darab n változós logikai szorzat logikai összege
 - A Boole-függvény megvalósítása szabványos kapuk segítségével
- **Az absztrakt Boole-függvény és az elektronikus áramkörrel való megvalósítás között KÜLÖNBSÉG van**
 - Boole-függvény
 - Változók
 - A, B és C
 - Boole-műveletek
 - ÉS, VAGY és NEM
 - Igazságtáblázat vagy Boole-kifejezések
 - Elektronikus áramkörrel megvalósítható
 - Gyakran különböző módon
 - A változókat elektromos jelek, a műveleteket pedig kapuk reprezentálják

Boole-függvények megvalósítása

- Írjuk fel a függvény igazságtáblázatát
- NEM kapuk biztosítása a bemenet komplementensének előállításához
- ÉS kapuk elhelyezése minden olyan sorhoz, amelynek az eredményoszlopában 1-es van
- ÉS kapuk összekapcsolása megfelelő bemenettel
- Az összes ÉS kapu kimenetének bekötése egy VAGY kapuba

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)



(b)

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

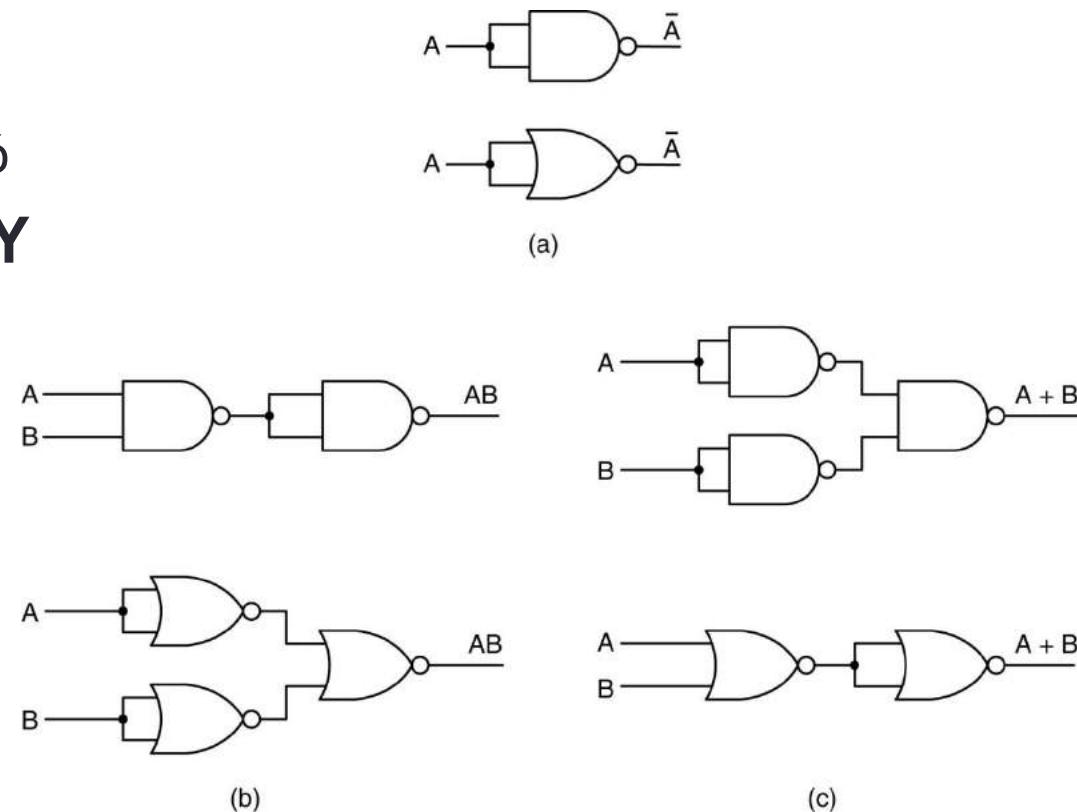
Boole-függvények megvalósítása

- Kényelmesebb egy fajta kapuval megvalósítani az áramköröket**

- Nem optimális áramkörök
- De minden megvalósítható

- NEM-ÉS és NEM-VAGY**

- Teljesek
 - Bármely Boole-fv. kiszámítható bármelyik kizárolagos felhasználásával
- Más kapuk nem rendelkeznek ezzel a tulajdonsággal



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

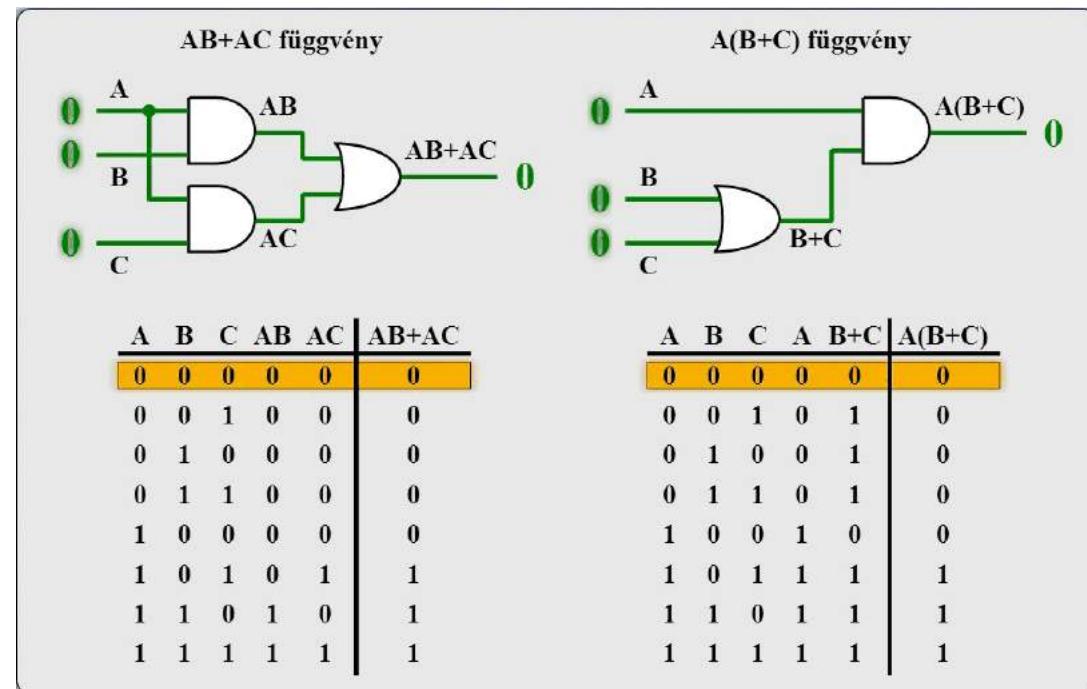
Áramköri ekvivalencia

- Kapuk számának a csökkentése fontos probléma**

- Alkatrész ára
- Nyomtatott áramköri lap nagysága
- Áramfogyasztás

- Találni kell egy olyan áramkört**

- Ugyanazt a függvényt számolja ki, mint az eredeti
- Kevesebb kapuból áll
- Boole-algebra fontos eszköz



Áramköri ekvivalencia

- Boole-algebra legfontosabb azonosságai

Név	ÉS forma	VAGY forma	Igazságtábla:
Identitásszabály	$1A=A$	$0+A=A$	$\begin{array}{c cc} 1 & A & 1A \\ \hline 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$
Nullszabály	$0A=0$	$1+A=1$	
Idempotens szabály	$AA=A$	$A+A=A$	
Inverz szabály	$A\bar{A}=0$	$A+\bar{A}=1$	
Kommutatív szabály	$AB=BA$	$A+B=B+A$	
Asszociatív szabály	$AB(C)=A(BC)$	$(A+B)+C=A+(B+C)$	
Disztribúciós szabály	$A+BC=(A+B)(A+C)$	$A(B+C)=AB+AC$	
Abszorpciós szabály	$A(A+B)=A$	$A+AB=A$	
De Morgan-szabály	$\overline{AB}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}\overline{B}$	

Forrás: Tanenbaum, Gulyás Dénes

Áramköri ekvivalencia

- Minden szabálynak két formája van**

- Egymásnak duálisai
 - Az ÉS és VAGY valamint a 0 és 1 egyidejű cseréjével előállítható a másik

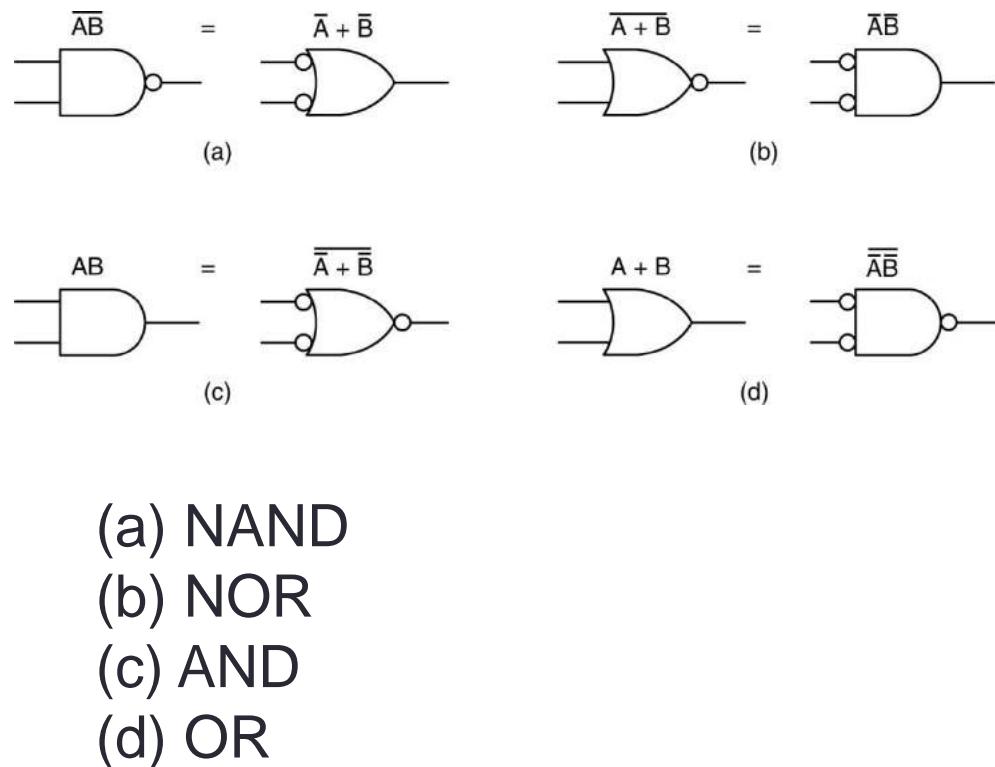
- A szabályok az igazságtáblázatukkal bizonyíthatóak**

- De-Morgan szabály kiterjesztése**

- $$\overline{ABC} = \bar{A} + \bar{B} + \bar{C}$$

- Alternatív jelölés**

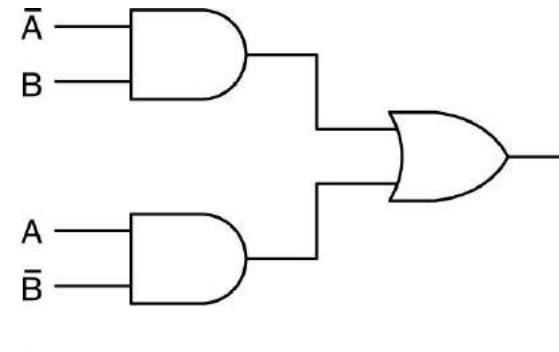
- Pl. (a) esetén egy VAGY kapu invertált bemenettel egy NEM-ÉS kapuval ekvivalens



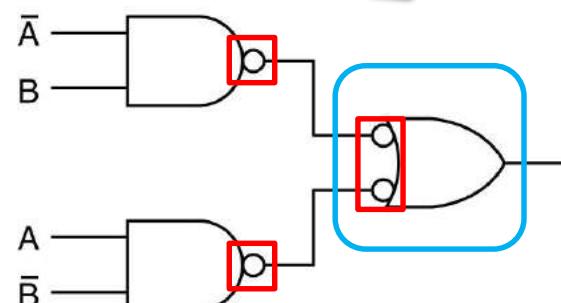
Áramköri ekvivalencia

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

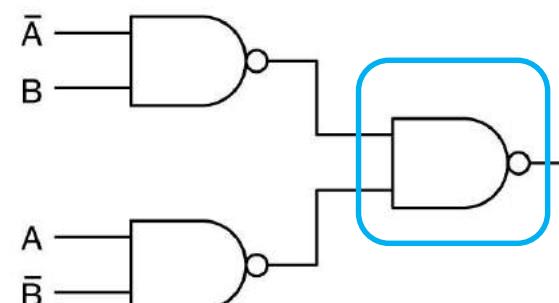
(a)



(b)

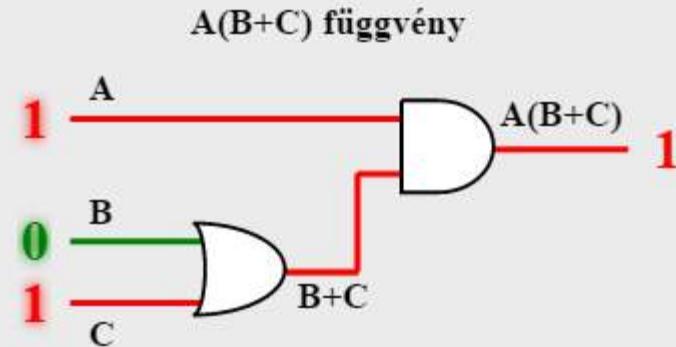
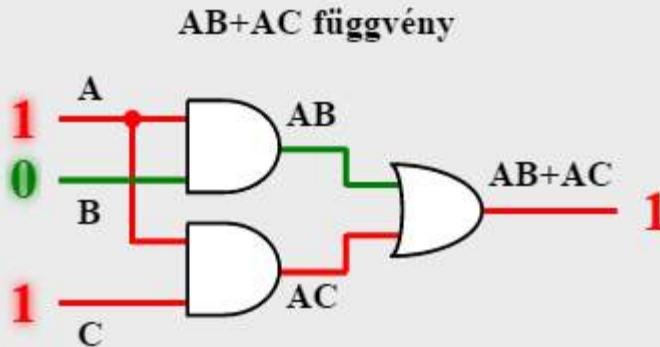


(c)



(d)

Áramköri ekvivalencia



A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

A	B	C	A	B+C	A(B+C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Áramköri ekvivalencia

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

A	B	F
0 ^V	0 ^V	0 ^V
0 ^V	5 ^V	0 ^V
5 ^V	0 ^V	0 ^V
5 ^V	5 ^V	5 ^V

(a)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

(b)

A	B	F
1	1	1
1	0	1
0	1	1
0	0	0

(c)

(a) Elektronikus karakterisztika

(b) **Pozitív logika**

1: igaz ~ magas

0: hamis ~ alacsony

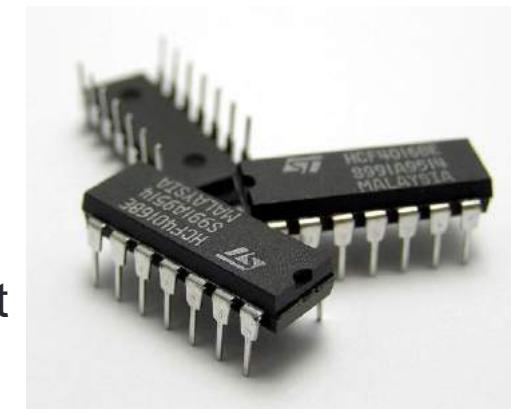
(c) Negatív logika

Alapvető digitális logikai áramkörök

- Kevés áramkört alakítanak ki egyedi kapuk használatával
 - Inkább építőblokkokat használnak
 - Számos kaput tartalmaznak
- Integrált áramkörök
 - Integrated Circuits (IC)
 - Derékszögű műanyag vagy kerámia tok
 - Lábak
 - Behelyezhető egy foglalatba vagy forrasztható
 - minden lab egy kapu bemenete vagy kimenete
 - Áram
 - Föld
 - Kívül kétsoros lábazást és a belső integrált áramkört együtt Dual Inline Packages-nek (DIP) neveznek



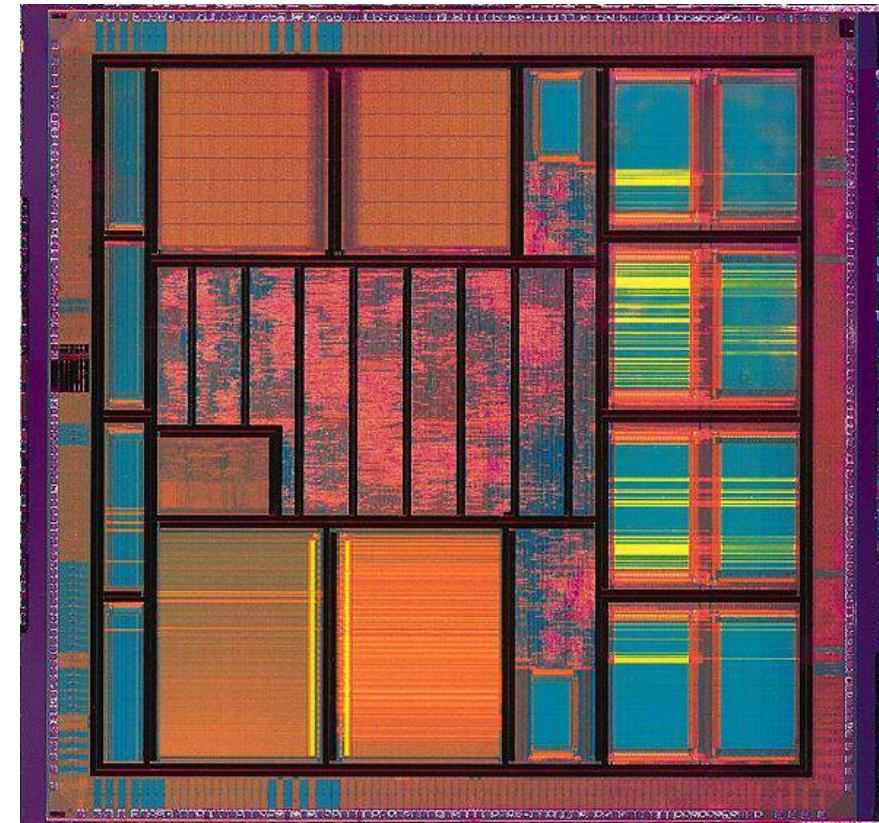
Forrás: Wikipedia



Integrált áramkörök

- **Osztályozás**

- **SSI** (Small Scale Integrated)
 - kis integráltságú
 - 1-10 kapu
- **MSI** (Medium Scale Integrated) – közepes integráltságú
 - 10-100 kapu
- **LSI** (Large Scale Integrated)
 - nagy integráltságú
 - 100-100 000 kapu
- **VLSI** (Very Large Scale Integrated) – nagyon nagy integráltságú
 - > 100 000 kapu



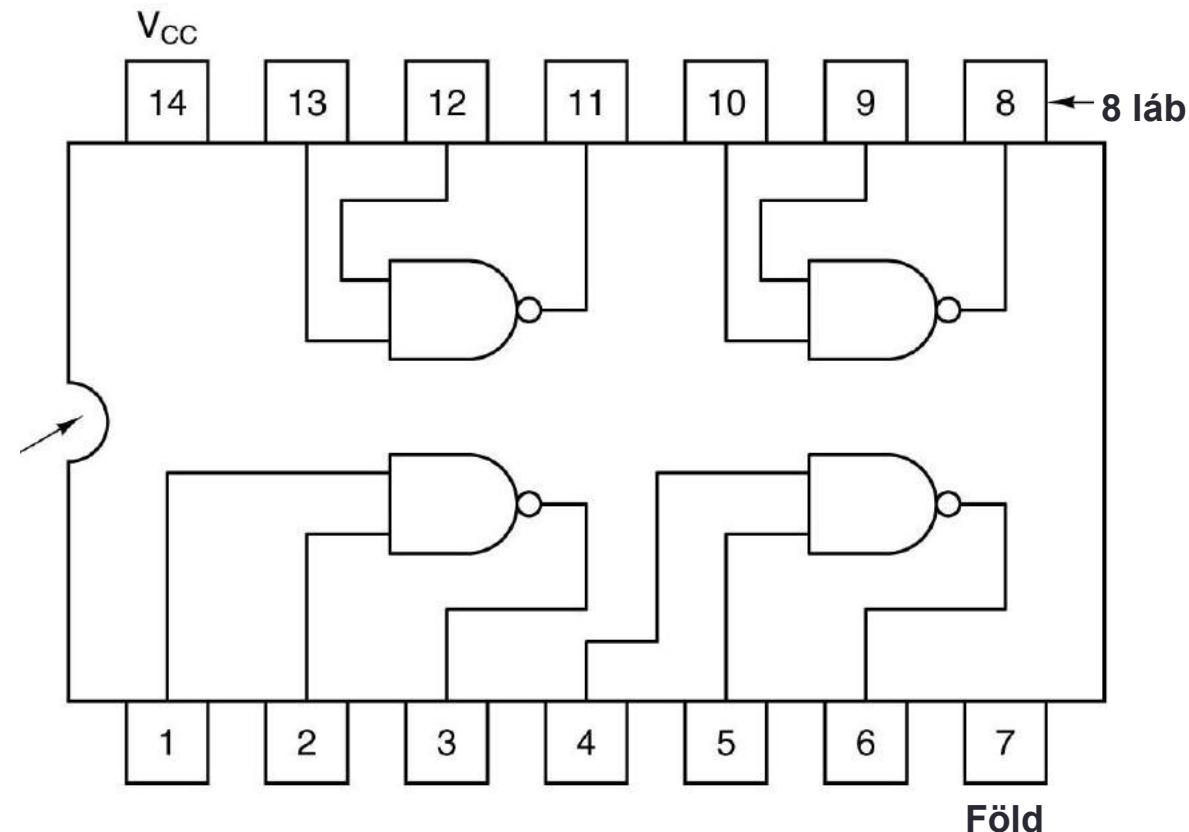
Forrás: Wikipedia

Integrált áramkörök

- SSI lapka
 - 2-6 független kapu

- 4 NEM-ÉS kapu
- 12 láb
 - 2 bemenet
 - 1 kimenet
- Áram
 - (V_{CC} feszültség)
- Föld
 - (GND)

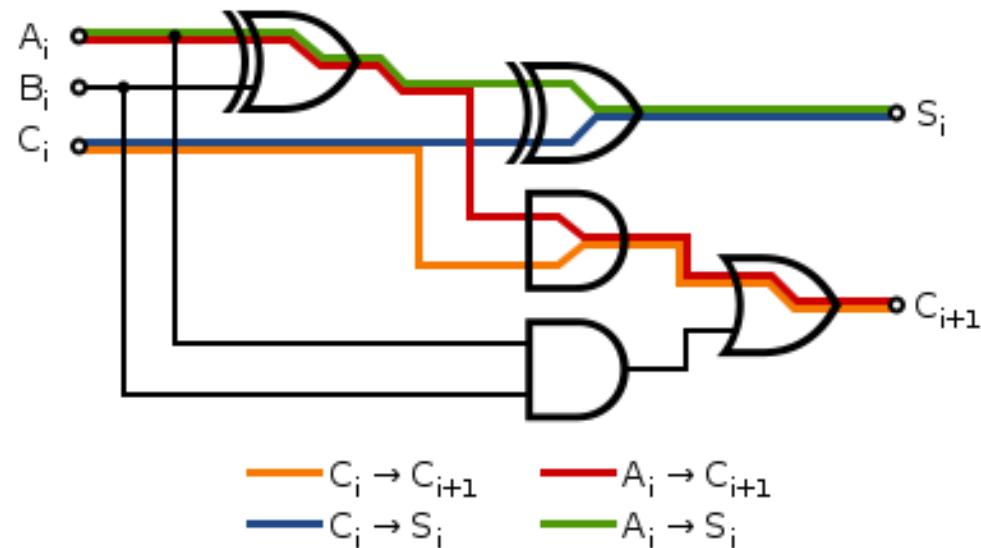
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Integrált áramkörök

- Ideális kapu
 - A kimenet azonnal előáll, amint a bemenetet alkalmazzuk
- Valóság
 - A lapkáknak véges kapukésleltetésük van
 - Tartalmazza a jel terjedését a lapkán
 - Kapcsolási idő
 - 1-10 ns
- Manapság kb. 10 millió tranzisztor van egy lapkán
 - 5 millió NEM-ÉS kapu
 - 15 000 002 lábra lenne szükség
 - lábtávolság
- Nagy kapu/láb arányú áramkörök
 - Korlátozott számú kapcsolat

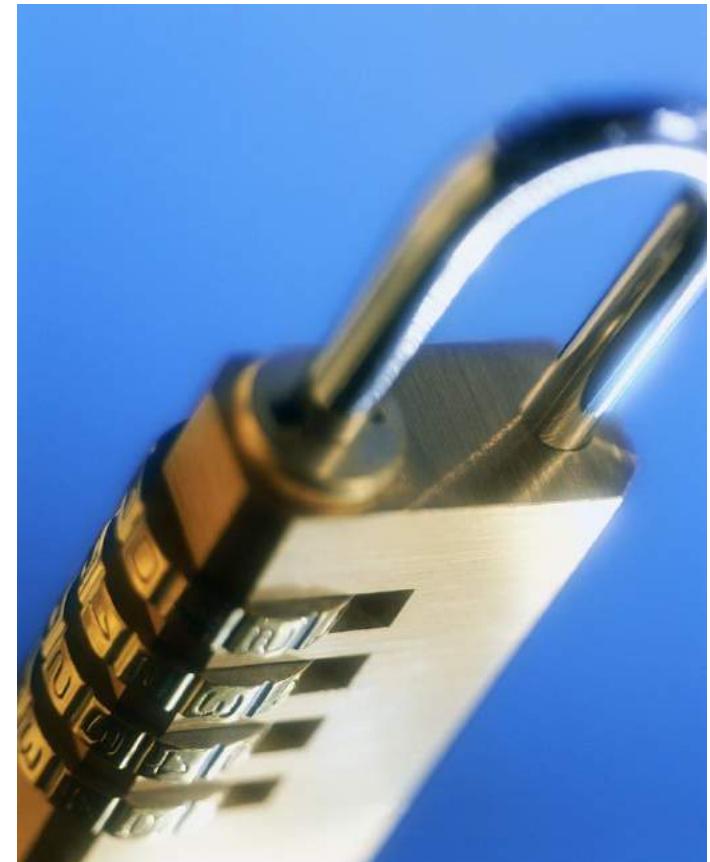
Terjedési késleltetés



Forrás: Wikipedia

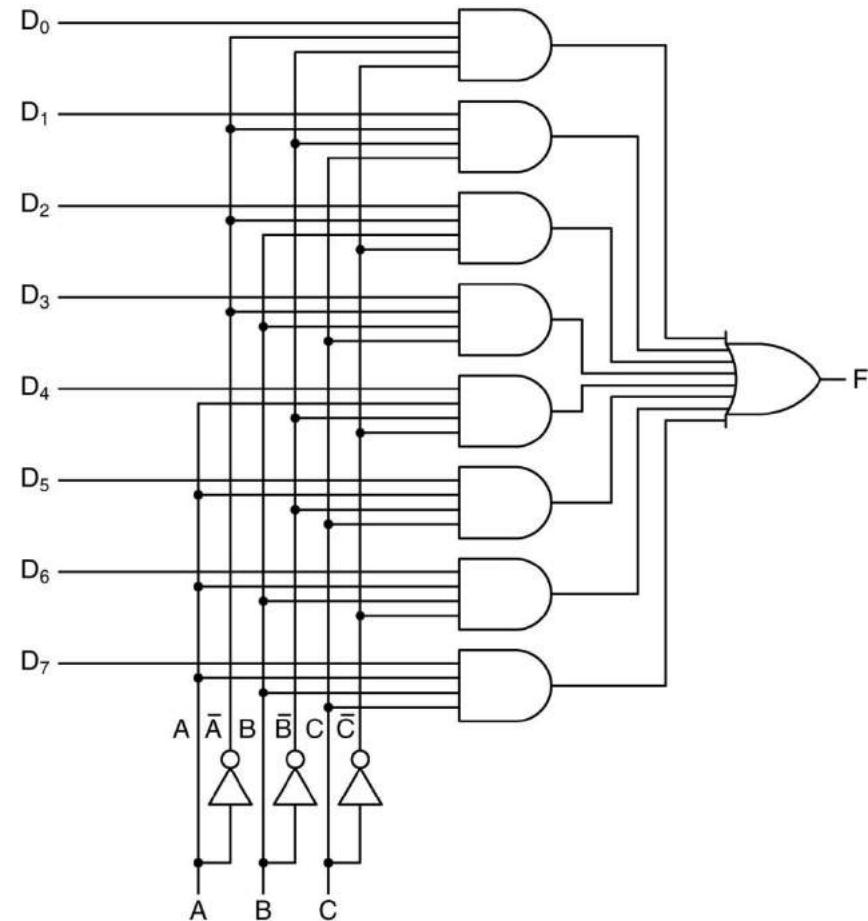
Kombinációs áramkörök

- Sok alkalmazás megkívánja, hogy egy áramkör
 - Többszörös bemenet
 - Többszörös kimenet
 - A kimenetet a bemenetek határozzák meg
- Nem minden áramkörnek van ilyen tulajdonsága
 - Pl. memória elemek



Multiplexerek

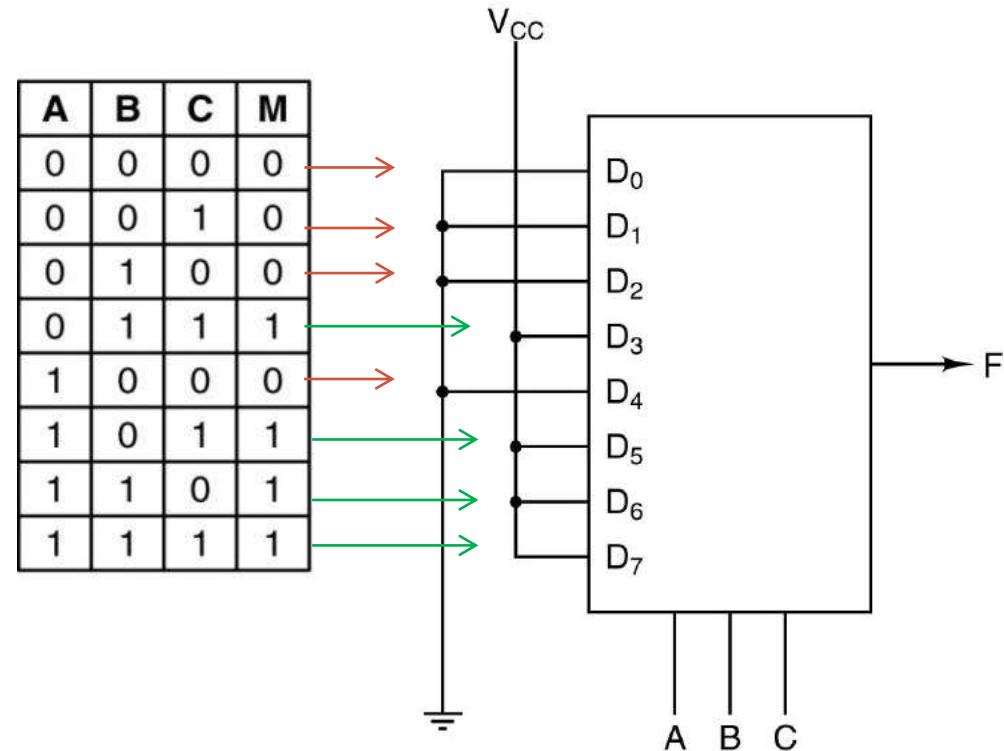
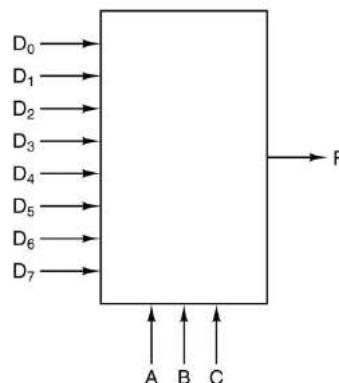
- 2^n adatbemenet
- 1 adatkimenet
- n vezérlőbemenet
- A kiválasztott adatbemenet
 - Kimenetre irányított vagy „kapuzott”
- Áram és föld lábakkal
 - 14 lábas tokban helyezhető el



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

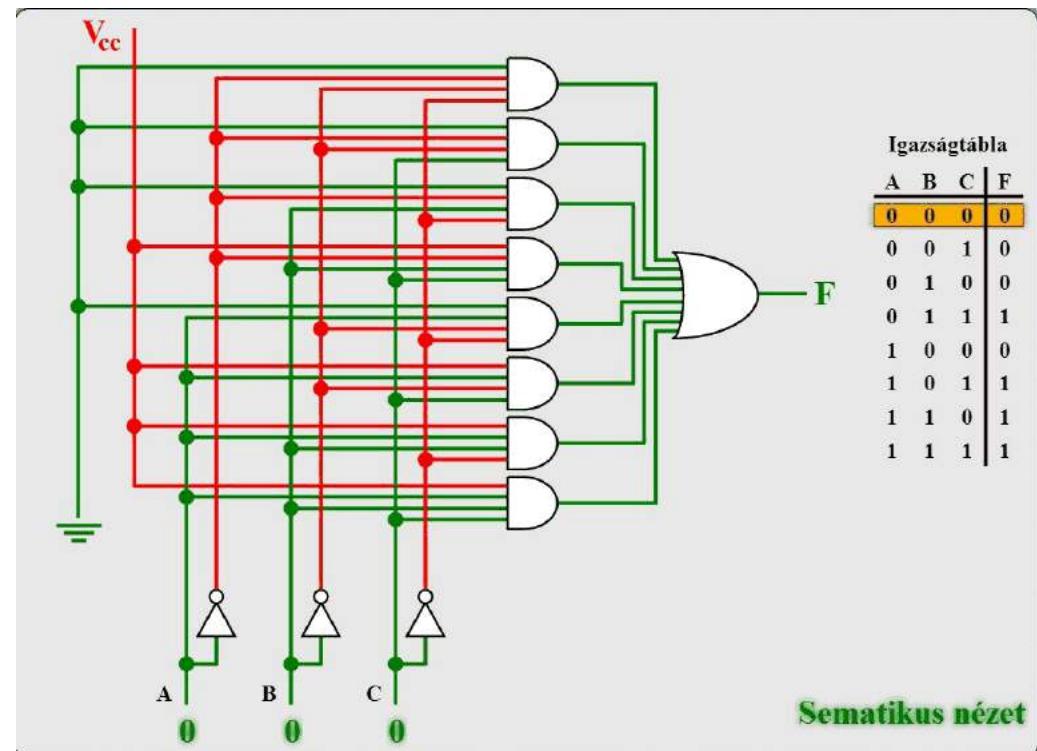
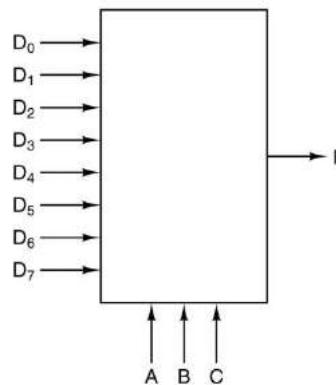
Multiplexerek

- Többségi függvény megvalósítása multiplexerrel
 - Bármelyik háromváltozós igazságtábla implementálható



Multiplexerek

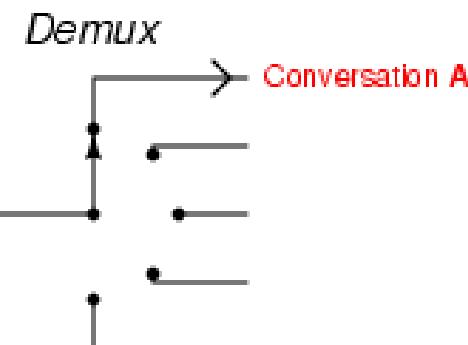
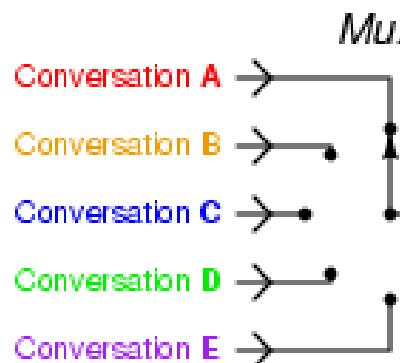
- Többségi függvény megvalósítása multiplexerrel
 - Bármelyik háromváltozós igazságtábla implementálható



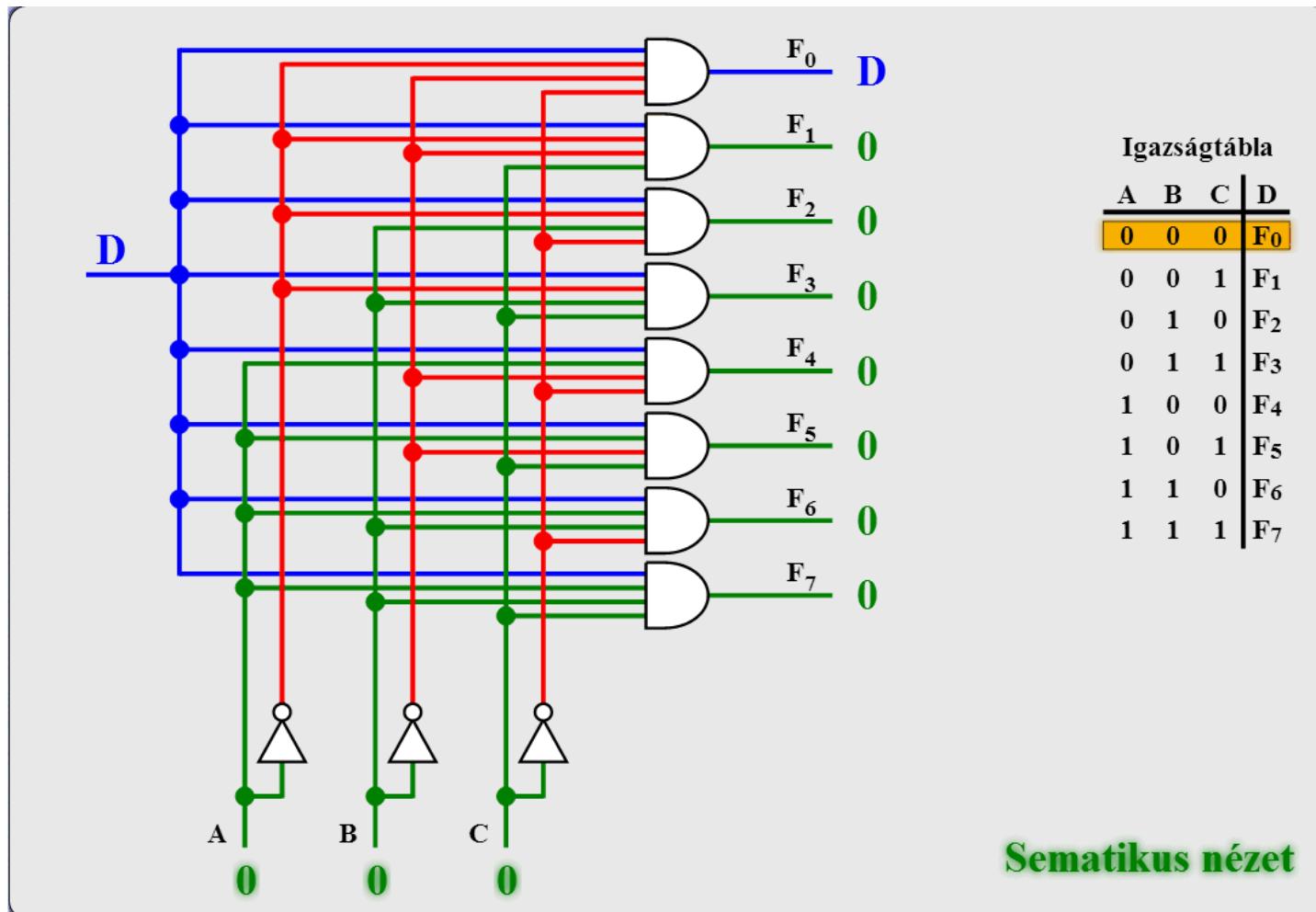
Forrás: Tanenbaum, Gulyás Dénes

Multiplexerek – demultiplexerek

- 000-tól 111-ig
 - Sorban lépegetés
 - 8 bitet egymás után küldhetjük a kimenetre
 - Párhuzamos soros konverzió
 - Billentyűkód átküldése telefonvonalon
- A multiplexer fordítottja
 - Demultiplexer
 - Egy egyedi jel bemenőjel megjelenítése 2^n kimenet valamelyikén, n vezérlővonaltól függően

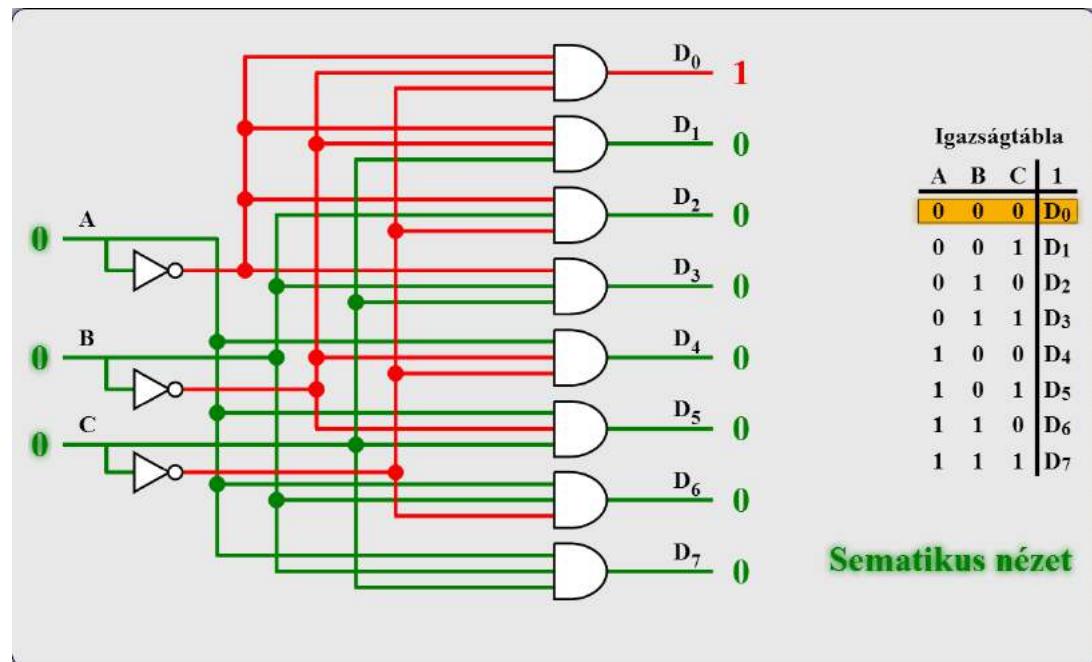


Demultiplexerek



Dekódolók

- Bemenet egy n bites szám
- Egyet kiválaszt a 2^n kimenet közül és 1-re állítja
 - Példa $n = 3$
- 8 db 1 MB-os lapka
 - Memóriacím használata
 - Felső 3 bit
 - Az adott lapka kiválasztása
 - Különböző bemenet kombinációval minden egyik lap kiválasztható

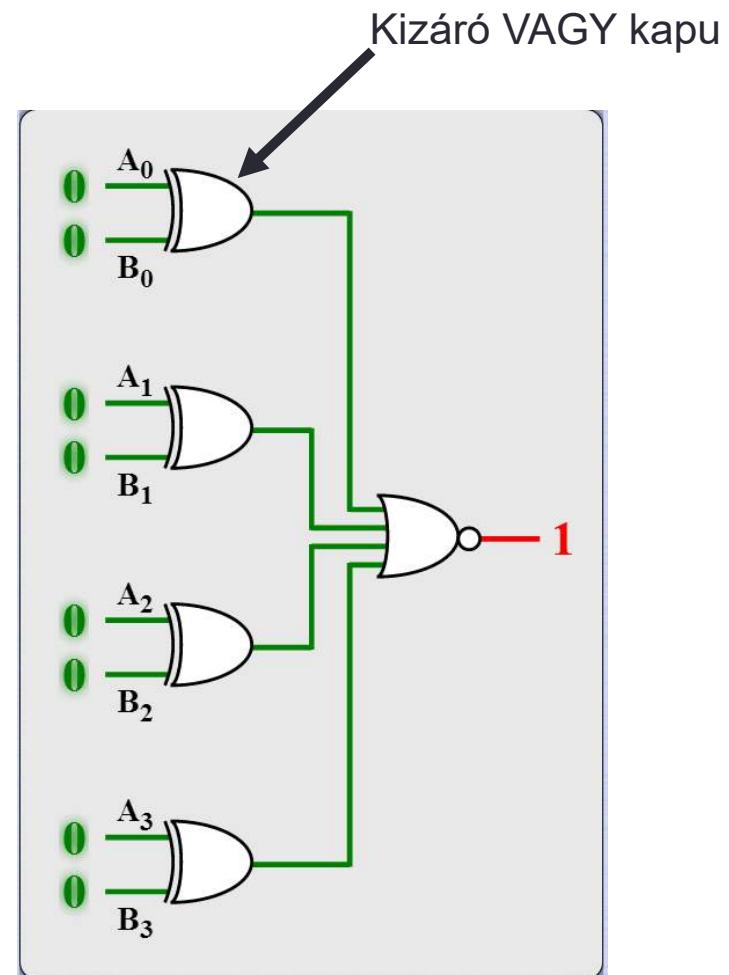


Forrás: Tanenbaum, Gulyás Dénes

Összehasonlítók

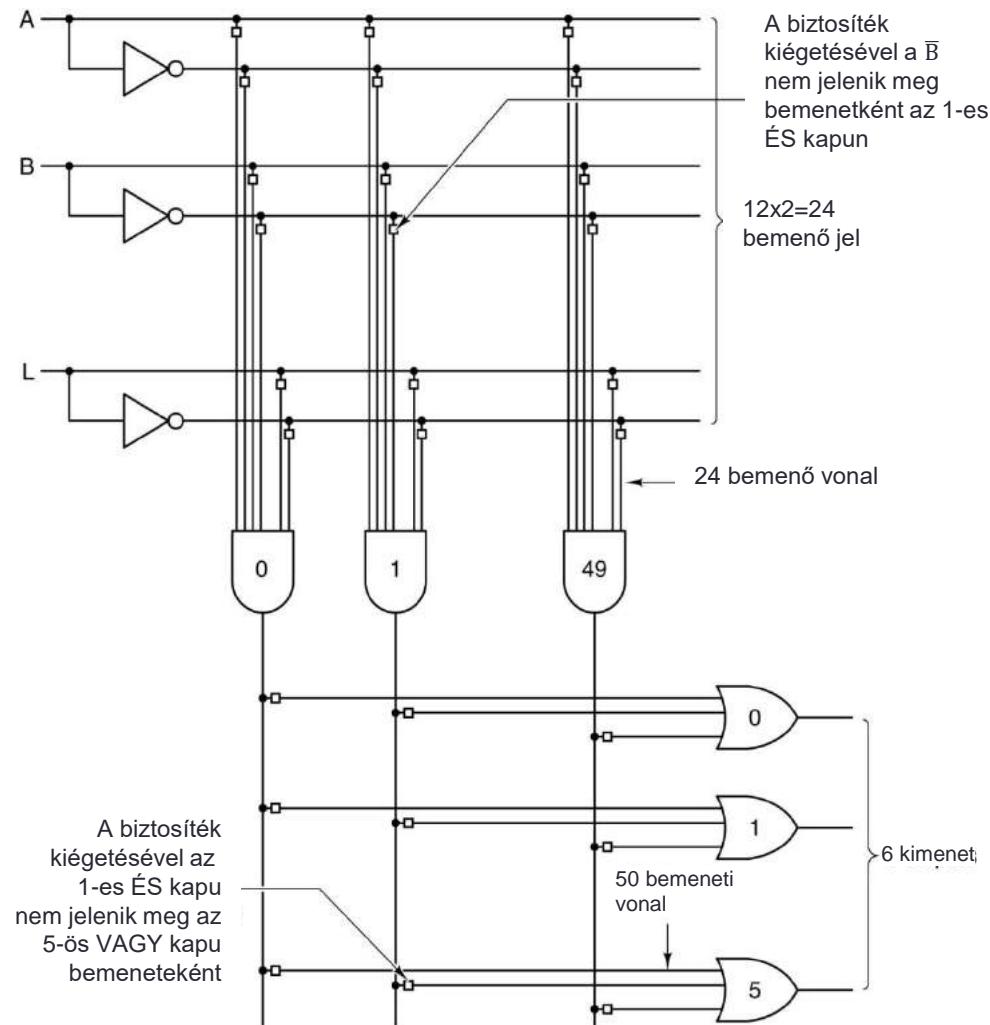
- Két bemenet összehasonlítása
 - Bitenkénti összehasonlítás

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Programozható logikai tömb (PLA)

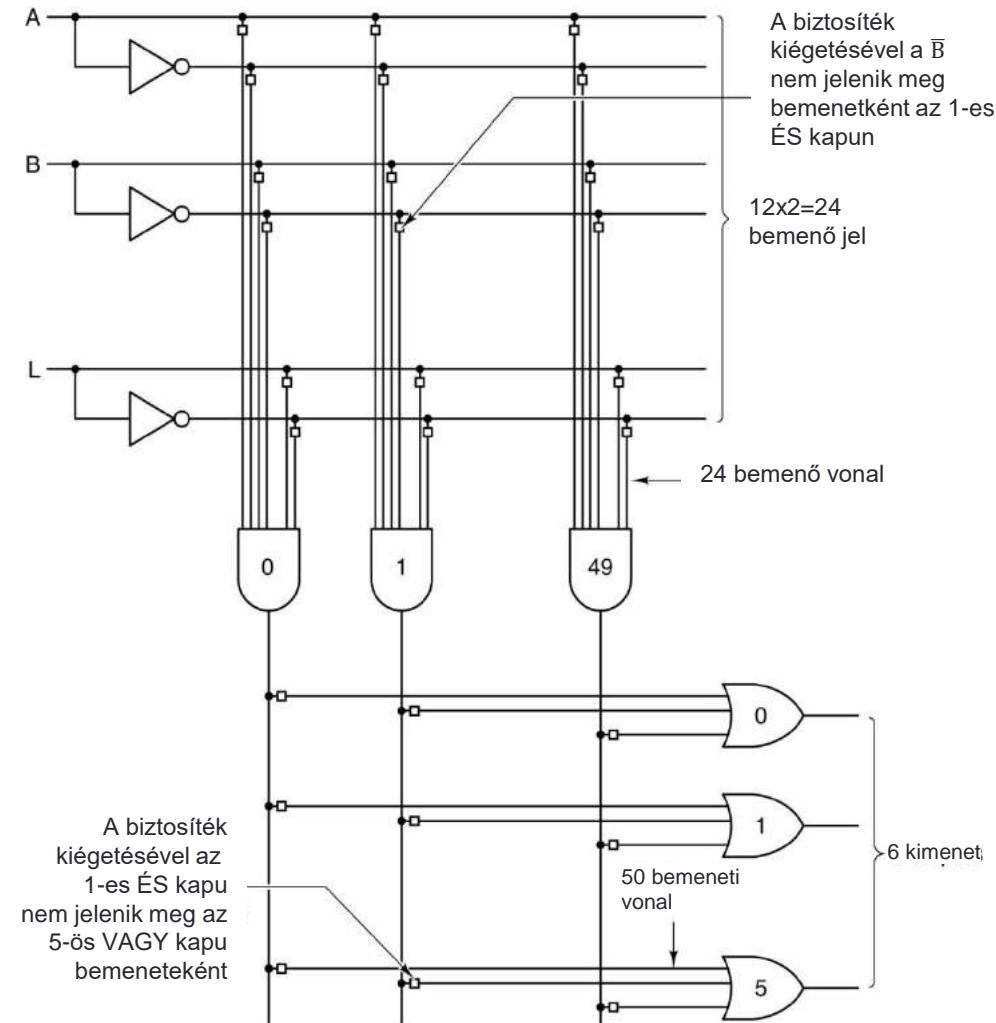
- **Nagyon általános lap**
 - 12 változó bemenet
 - Komplemensek \rightarrow 24 bemenet
 - 6 kimenet
- **50 ÉS kapu**
 - 24×50 bites mátrix
 - Melyik bemenőjel melyik ÉS kapura kapcsolódik
- **6 VAGY kapu**
 - 50 bemenő jel
 - ÉS kapuk kimenete
 - 50×6 bites mátrix
 - Melyik bemenőjel melyik VAGY kapura kapcsolódik
- **12 bemenő, 6 kimenő, tápfeszültség és föld**
 - 20 lába van összesen



Programozható logikai tömb (PLA)

• Programozás

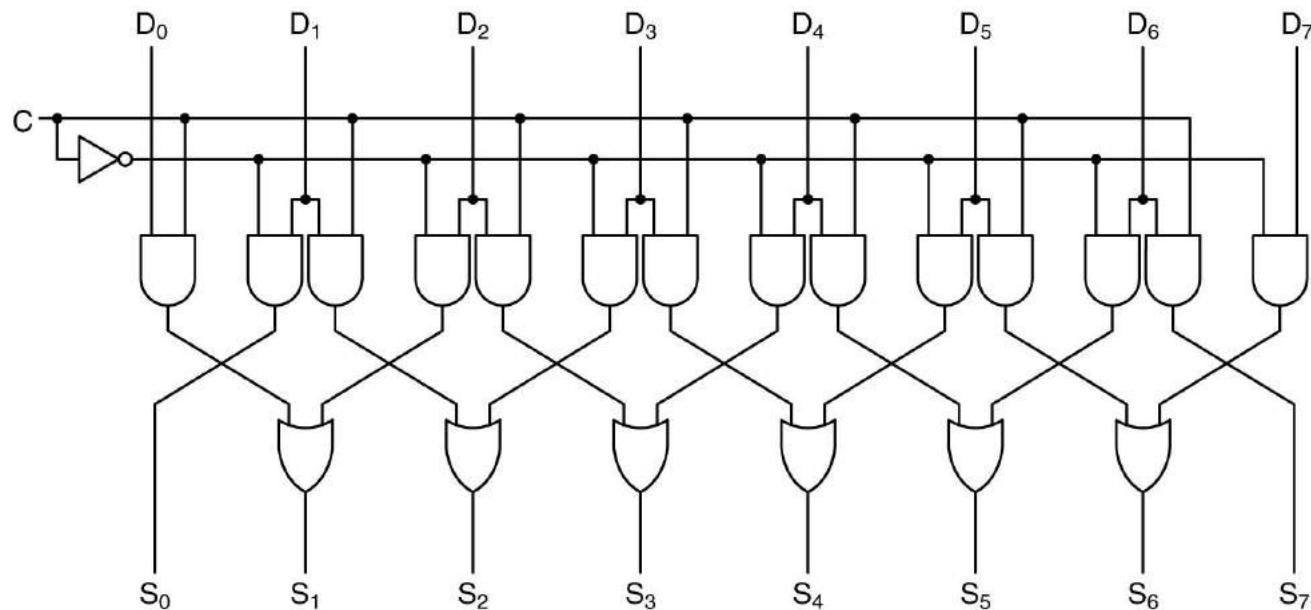
- Kezdetben (pl. vásárláskor) az összes olvadó biztosíték sértetlen
- Nagy árammal a lapkában kiégetik a kiválasztott biztosítékokat



Aritmetikai áramkörök

- **Léptető**

- 8 bemenet
 - D_0, D_1, \dots
- 8 kimenet
 - S_1, S_2, \dots
- C vezérlővonal
 - 1 jobbra
 - 0 balra

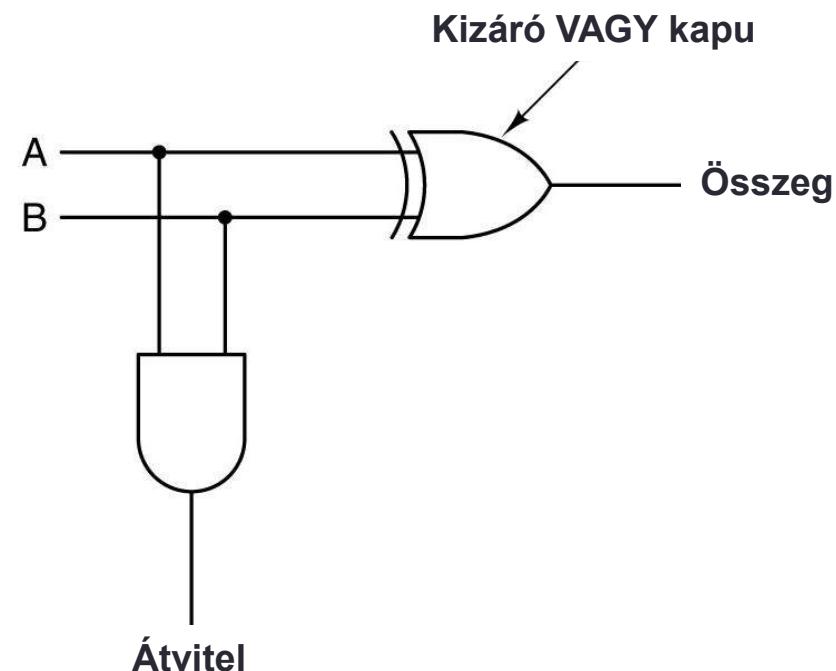


Összeadók

- 1 bites egészek összeadásának igazságtáblázata
- Fél összeadó
 - Átvitel (carry)
 - Balra lévő pozícióba
 - Nem kezeli a jobbról jövő átvitelt

A	B	Összeg	Átvitel
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

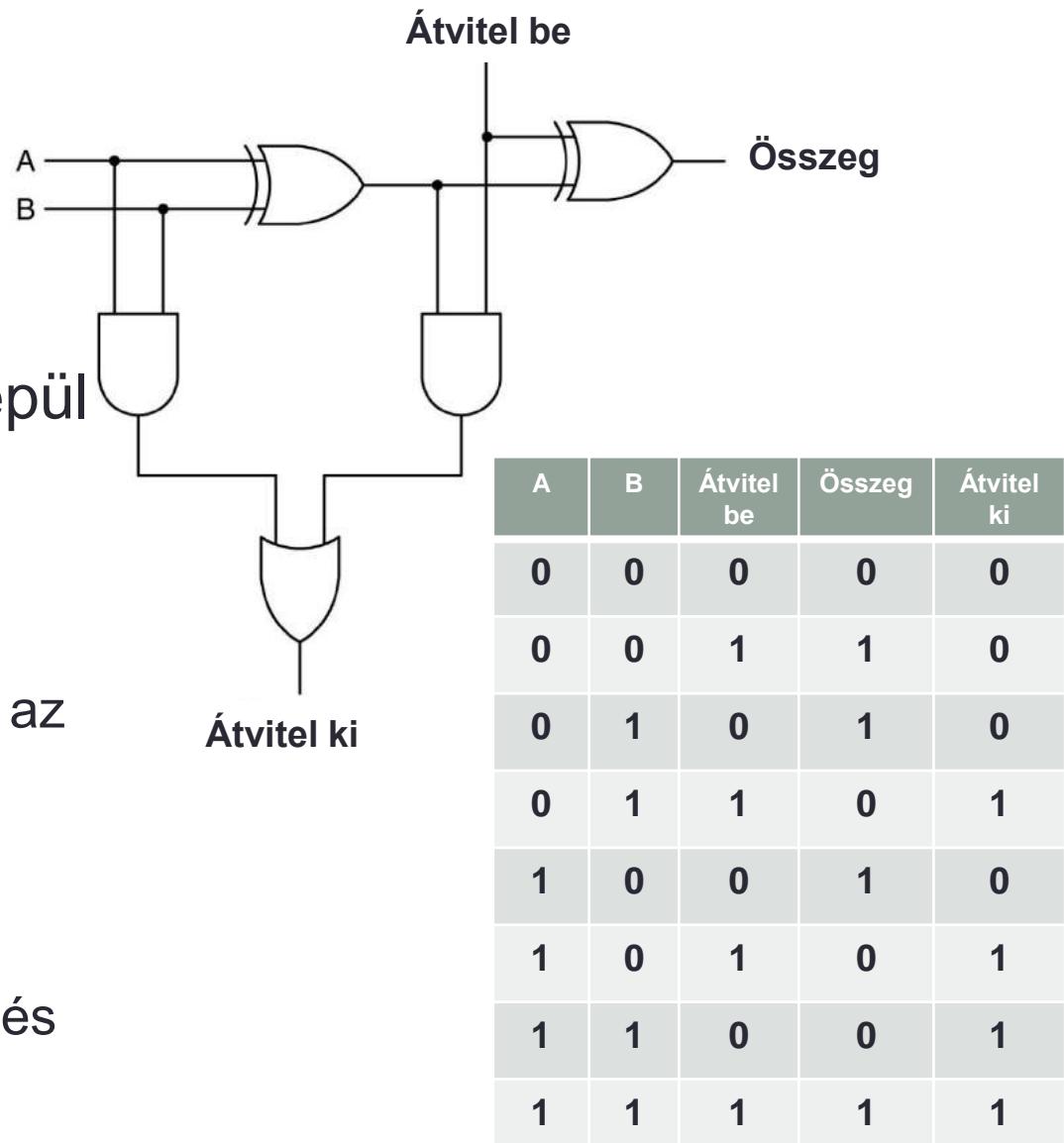
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Összeadók

- Teljes összeadó

- Két fél összeadóból épül fel
- Az összeg==1
 - Ha az A, B és átvitel be bemeneteken páratlan az 1-sek száma
- Átvitel ki == 1, ha
 - A és B is 1
 - Pontosan az egyikük 1 és az Átvitel be szintén 1

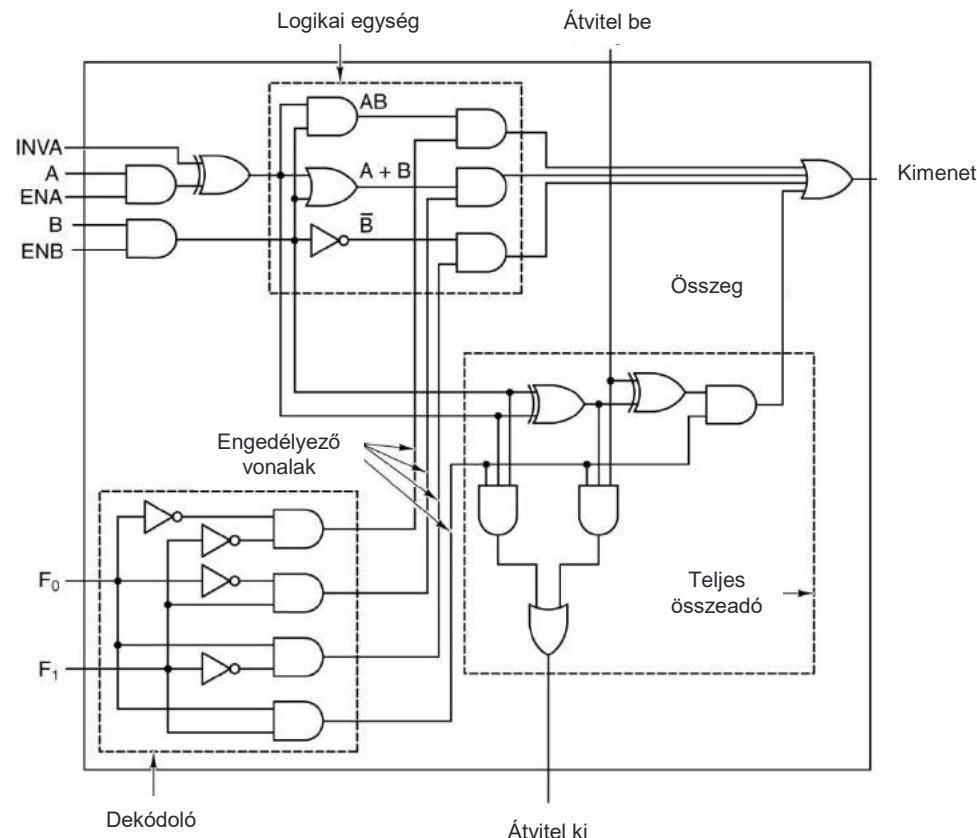


Összeadók

- 16 bites szó összeadása
 - 16 teljes összeadó
 - Az Átvitel ki bitet a bal oldali szomszéd Átvitel be ágához kötjük
 - Átvitelt tovább terjesztő összeadó
 - Legrosszabb eset
 - 111 ... 111 + 1
 - Addig nem kapjuk meg az eredményt, amíg az átvitel nem halad végig
- Vannak olyan összeadók, amelyeknél nincs ilyen késleltetés
 - 32 bites összeadó szétbontása
 - 16 bites alsó és 16 bites felső rész
 - Nem áll rendelkezésre az átvitel a felső 16 bites rész számára
 - Alsó rész, és felső rész U0, U1
 - U0, U1 párhuzamos végrehajtása
 - Alsó rész átvitele kijelöli a felső részt
 - Átvitelkiválasztó összeadó

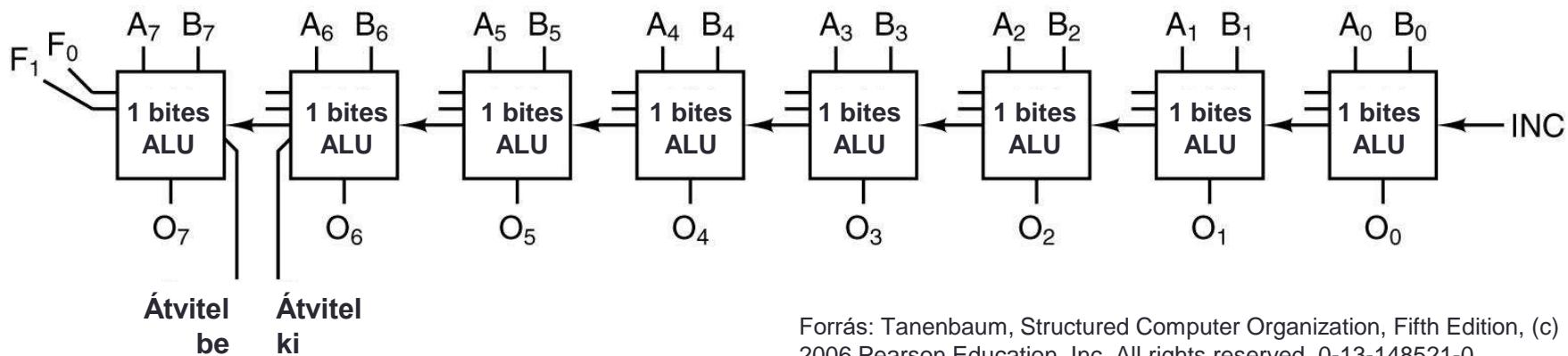
Aritmetikai-logikai egység

- Áramkör, amely végrehajtja az ÉS, VAGY műveleteket és két gépi szót összead
 - n bites szavak
 - n azonos áramkört tartalmaz
- 1 bites ALU
 - 2 bites dekóder
 - F_0, F_1 vezérlőjelek
 - 4 engedélyező vonal pontosan egy kerül kiválasztásra
 - Logika
 - A ÉS B, A VAGY B és NEM B
 - A és B engedélyezése
 - ENA, ENB
 - NEM A
 - INVA
 - Teljes összeadó
 - Több ilyen áramkör
 - Teljes szóhosszúságú művelet



Aritmetikai-logikai egység

- Bitszelet áramkörökből 8 bites ALU
 - Az invertáló jelek nincsenek feltüntetve

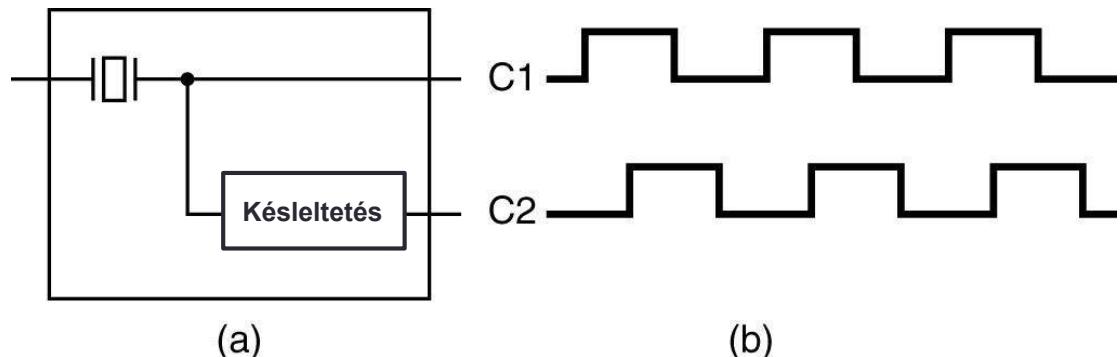


Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

- INC bemenet
 - Az eredmény eggyel nő
 - $A + 1, A + B + 1$

Órák

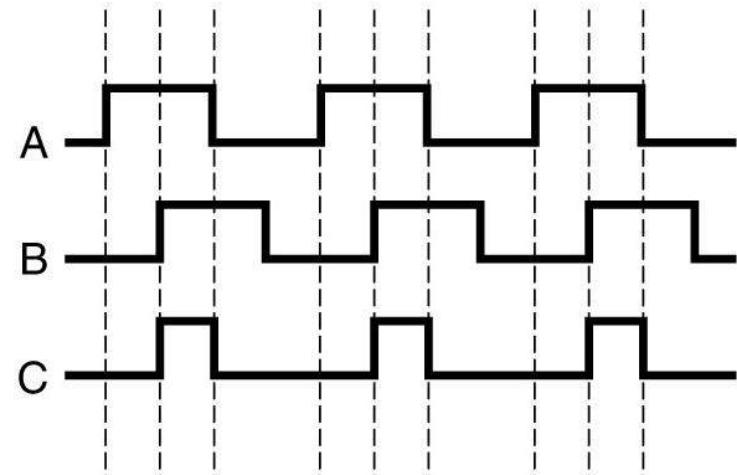
- Események sorrendje fontos
- Áramkör
 - Meghatározott szélességű pulzusok
 - Két pulzus közötti időintervallum
 - Ciklusidő
 - Pulzus frekvencia
 - 1 – 500 MHz
 - Órajel
 - 1000 – 2 ns
- Finomabb felbontás
 - Órajel megcsapolása
 - Késleltető áramkör
 - Másodlagos órajel
 - Fázis eltolással keletkezik



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

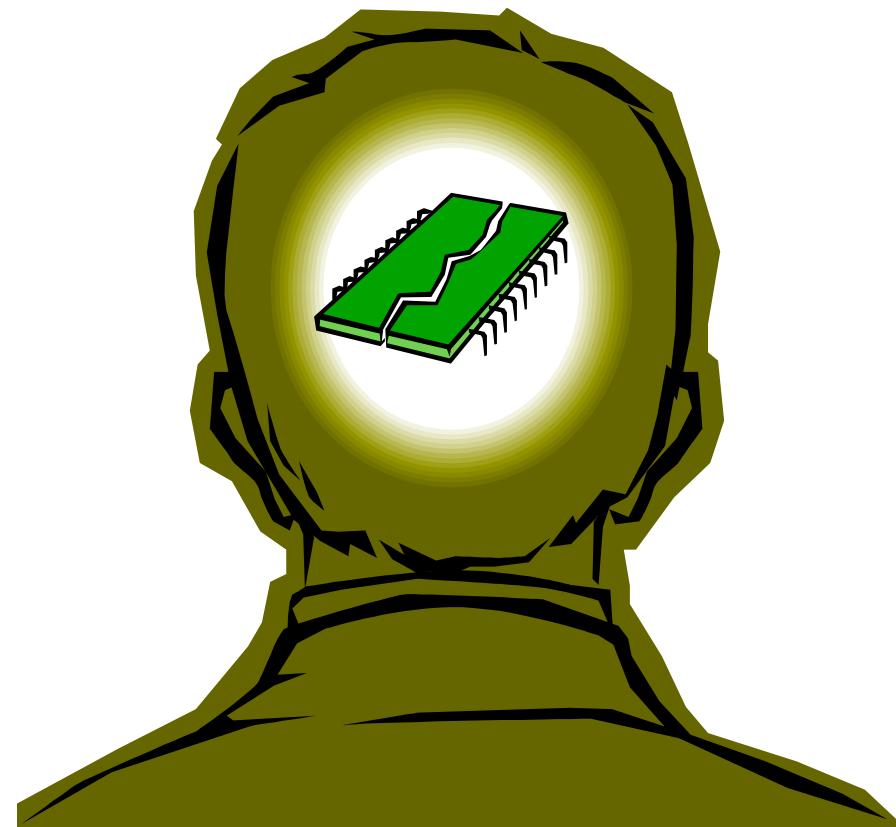
Órák

- Felfutó és lefutó élek
 - Időzítési intervallum
 - Magas érték esetén bekövetkező események
 - Szimmetrikusak
 - Magas állapotban eltöltött idő megegyezik az alacsony állapotban eltöltött idővel
- Aszimmetrikus óra
 - Alapóra
 - Késleltető
 - ÉS-selni az eredeti jellel



Memória

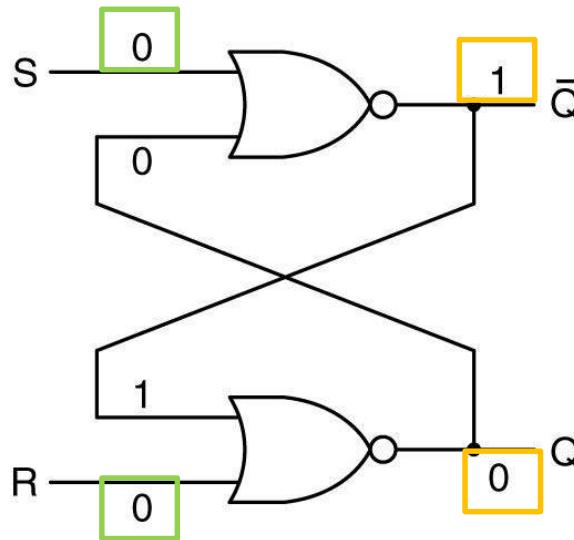
- Lényeges eleme a számítógépnek
 - Végrehajtandó utasítások és adatok tárolására



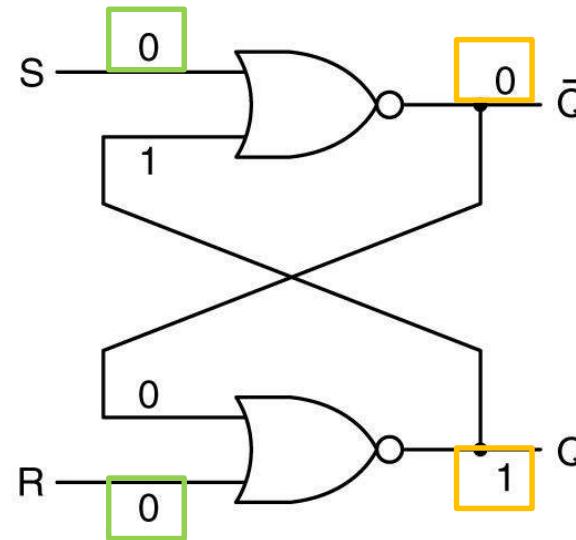
Tárolók

- 1 bites memória
- Visszaemlékezik az előző bemeneti értékre
- NEM-VAGY kapuk
 - NEM-ÉS kapuból is építhető
 - Set Reset latch (SR-tároló)

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Stabil, állandó állapot ($Q=0$)

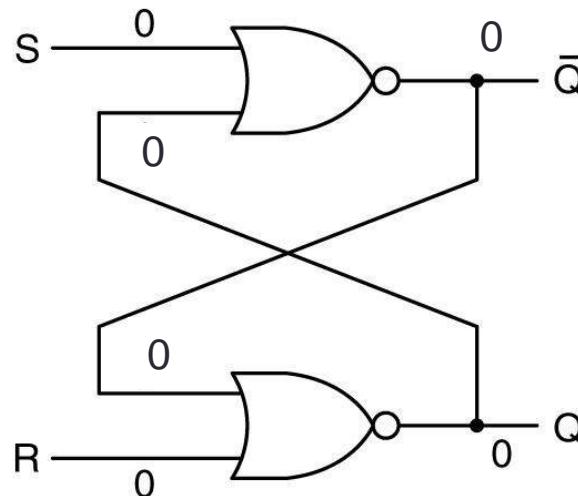


Stabil, állandó állapot ($Q=1$)

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Tárolók

- Nem stabil
 - Mindkét kimenet 0
 - Mindkét bemenet 0
 - Kimenet 1-re változik
- Lehetetlen
 - Mindkét kimenet 1 legyen
 - 0 és 1 bemenetre
 - Kimenet 0-ra változik



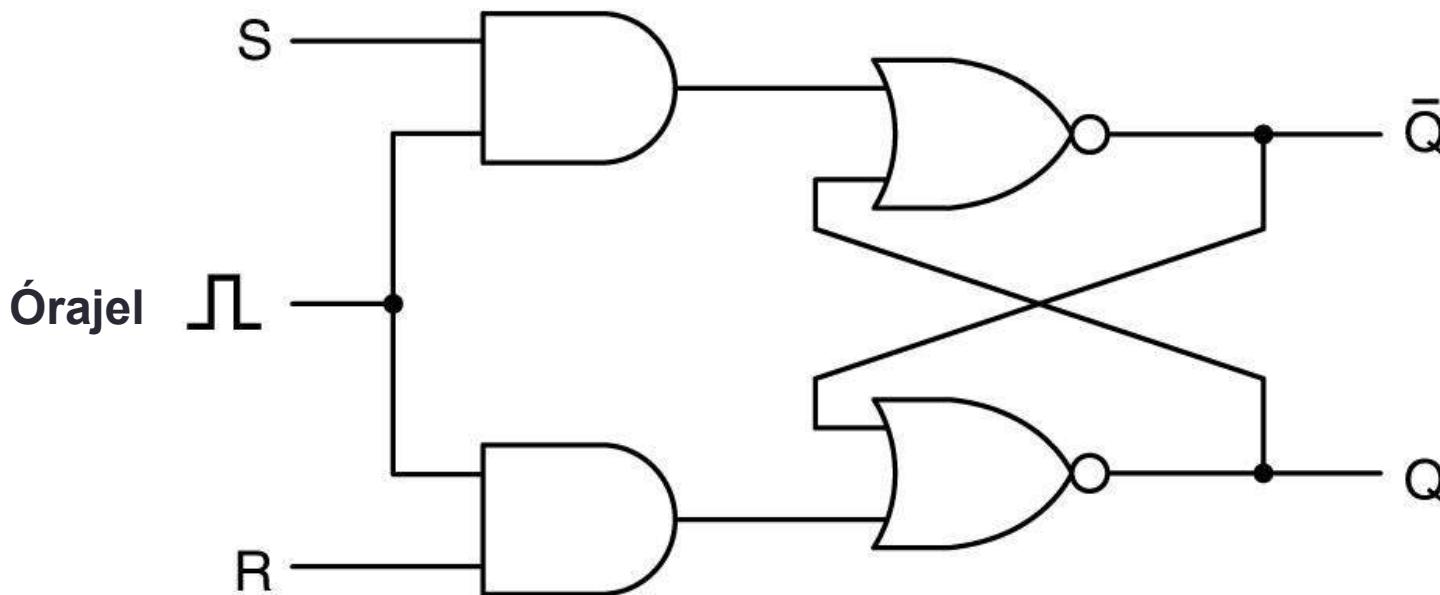
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Tárolók

- Bemenet hatása a tároló állapotára
 - $S=1, Q=0$
 - S beállításával a tároló 1-es állapotba kerül
 - $R=1$ nincs hatása, amikor 0-s állapotban van
 - S 1-re állításnak nincs hatása, amikor $Q=1$
 - R beállítása a tárolót 0-s állapotba viszi
 - Az áramkör emlékszik, hogy S vagy R jelet kapott utoljára

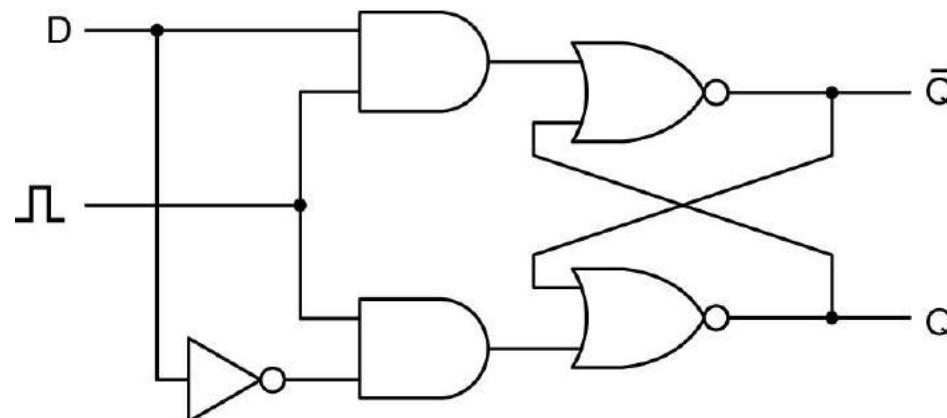
Időzített SR-tároló

- Csak bizonyos meghatározott időpontban történjen az állapotváltozás
 - 0-s órajel hatására a tároló nem változtatja az állapotát
 - Amikor az órajel 1
 - Az ÉS kapuk hatása megszűnik
 - Érvényes kapujel



Időzített D-tárolók

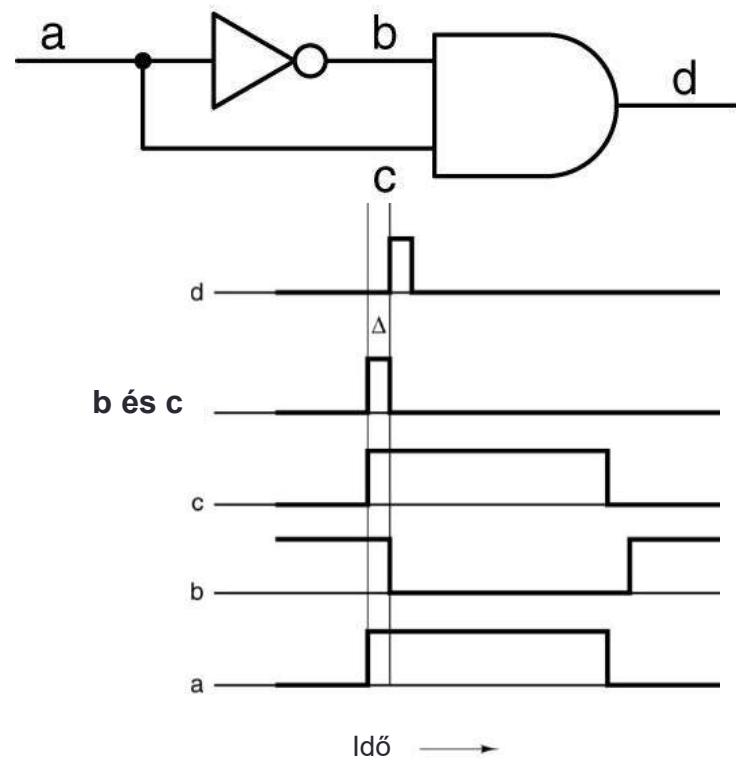
- $S=R=1$ által okozott bizonytalanság
 - Stabil állapot $Q = \bar{Q} = 0$
 - Két bemenet 0-ba megy át
 - A lassabb nyer
 - Amelyik később megy át nullába
- D bemenet
- A tárol érték mindig elérhető Q-ban



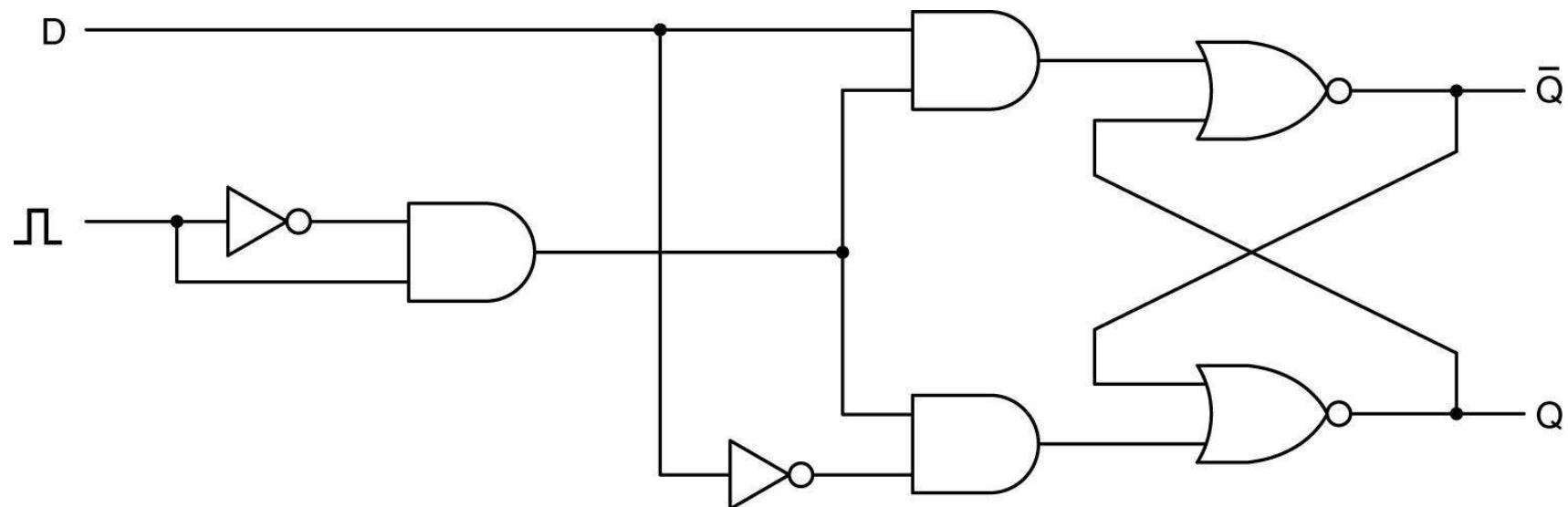
Flip-flopok

- Meghatározott időpillanatban mintavétel
 - Bizonyos vonalon lévő értékekből
 - Tárolja azt az értéket
- Flip-flop, billenőkör
 - Órajel 1-es állásánál
 - Nem fordul elő állapotváltozás
 - Órajel átmegy 0-ból 1-be vagy 1-ből 0-ra
 - Az órajel hossza nem érdekes
 - A flip-flop élezérelt
 - Míg a tároló szintvezérelt
 - Tervezés megközelítése
 - Nagyon rövid impulzus létrehozása az órajel felmenő élénél
 - Ezt tápláljuk be egy D-tárolóba
- ÉS kapu kimenete mindig 0!?
 - Inverter nem nulla késleltetési ideje

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

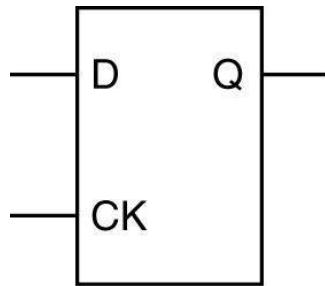


D-flip-flop



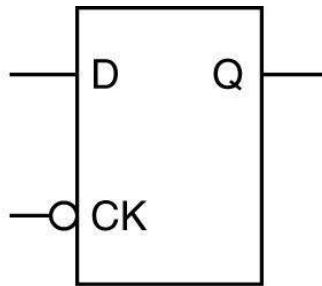
D-flip-flop

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

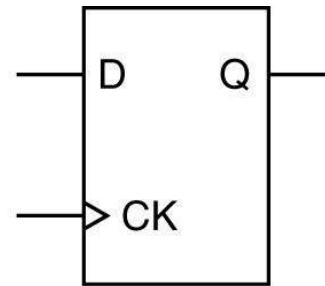


(a)

D tárolók

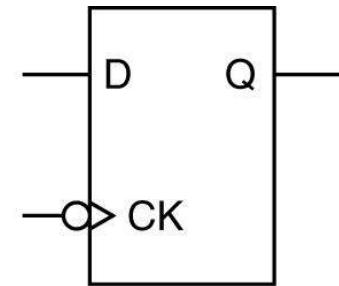


(b)



(c)

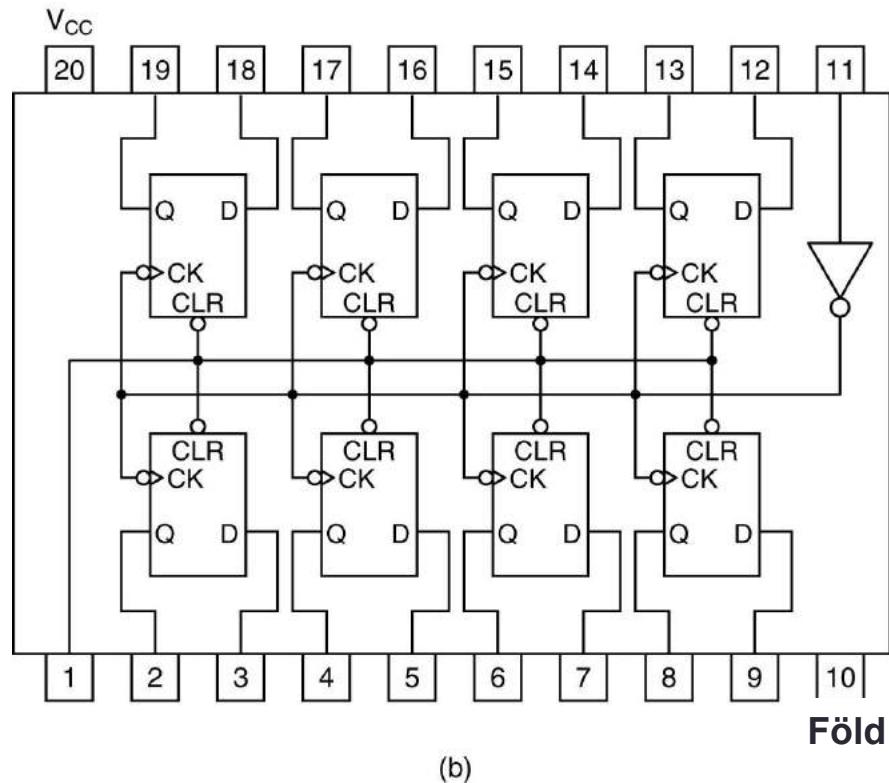
Flip-flopok (d)



- (a) $CK = 1$ esetén írja be D-t Q-ba
- (b) $CK = 0$ esetén írja be D-t Q-ba
- (c) CK emelkedő élnél
- (d) CK lefelé menő élnél

Regiszterek

- Felfutó élnél tárolnak
- Nyolc törlőjel csoportosítása
- Óravonalak csoportosítása
- Bemeneti invererek erősítőként szolgálnak
- 8 bites regiszter
- Két ilyen lapka párhuzamosan használva
 - 16 bites regiszter

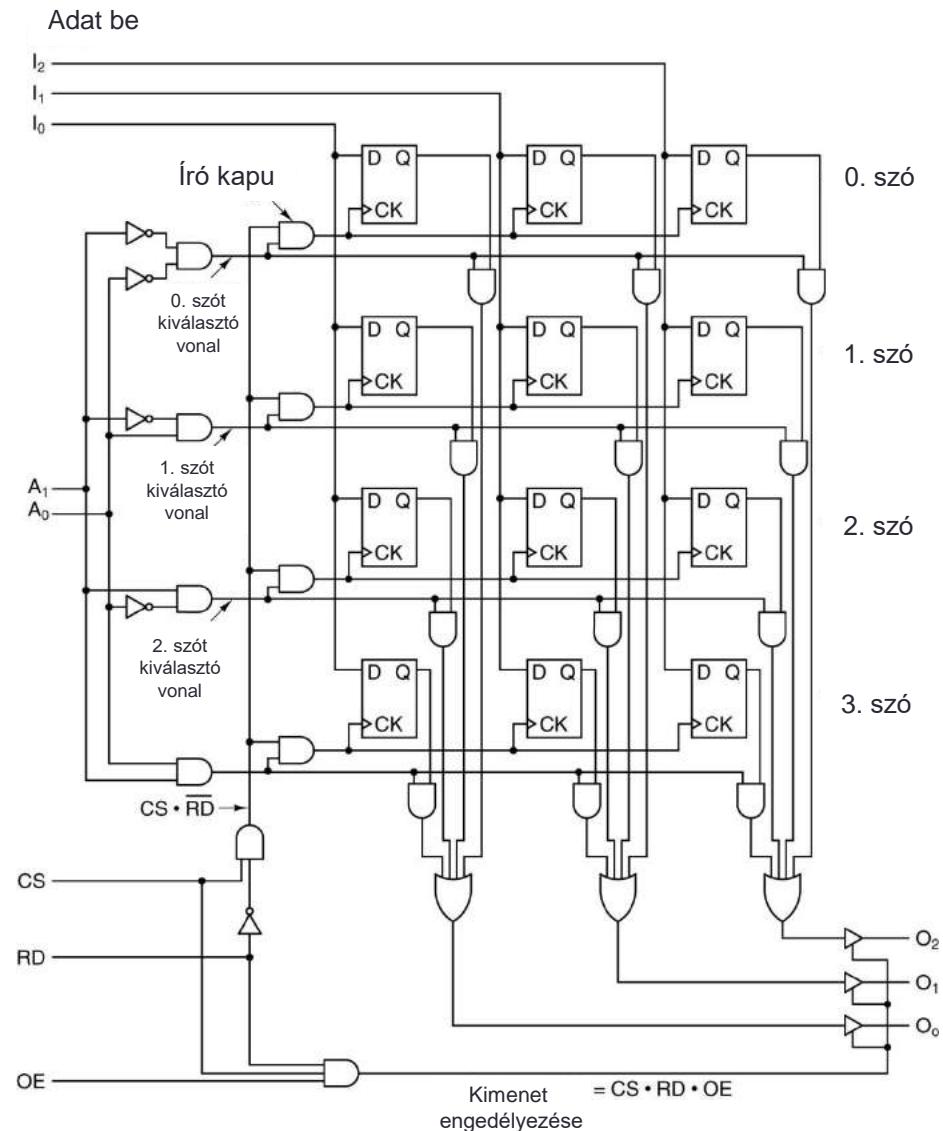


Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Memóriaszervezés

- Egyedi szavak megcímzése
 - Másfajta szervezés szükséges, mint az előzőekben
- 4 db 3 bites szóból álló memória
 - minden művelet teljes 3 bites szavakat olvas vagy ír
 - Kevesebb láb
 - Kiterjeszthető nagy memoriákra

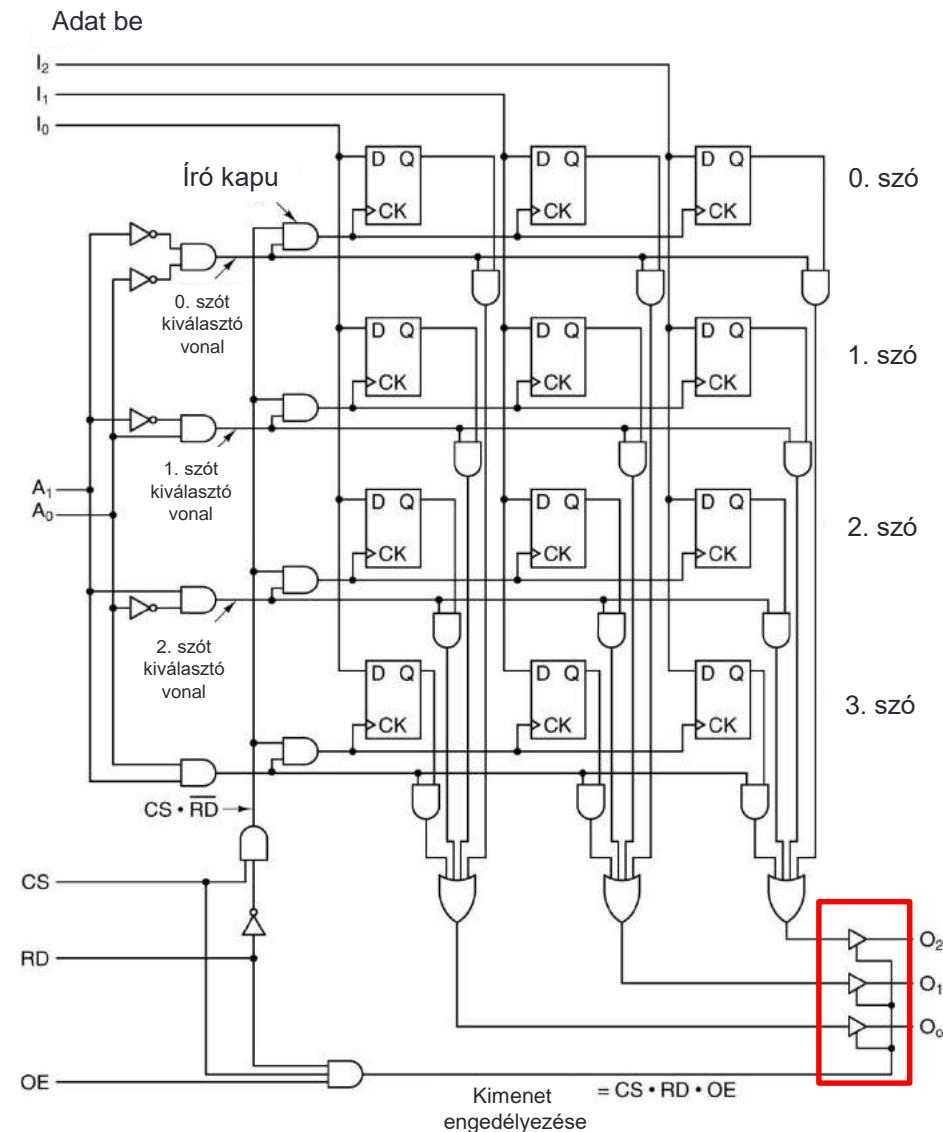
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Memóriaszervezés

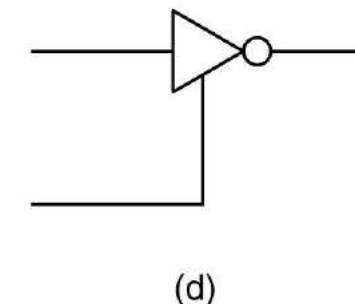
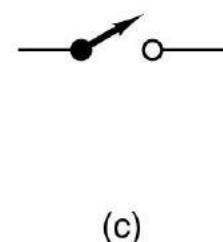
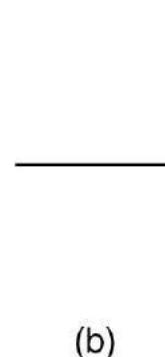
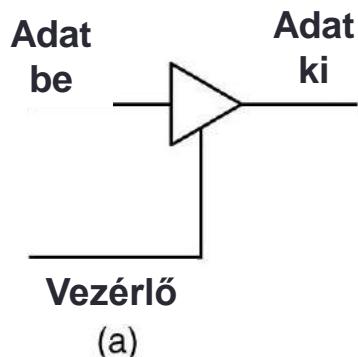
- 8 bemenő vonal
 - 3 adat I_0, I_1, I_2
 - 2 cím A_0, A_1
 - 3 vezérlő
 - CS: Chip Select
 - RD: Read
 - OE: Output Enable
- 3 kimenő vonal
 - O_0, O_1, O_2
- 14 lábas tokozás
 - Tokkal-vonóval együtt
 - Áram, föld

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



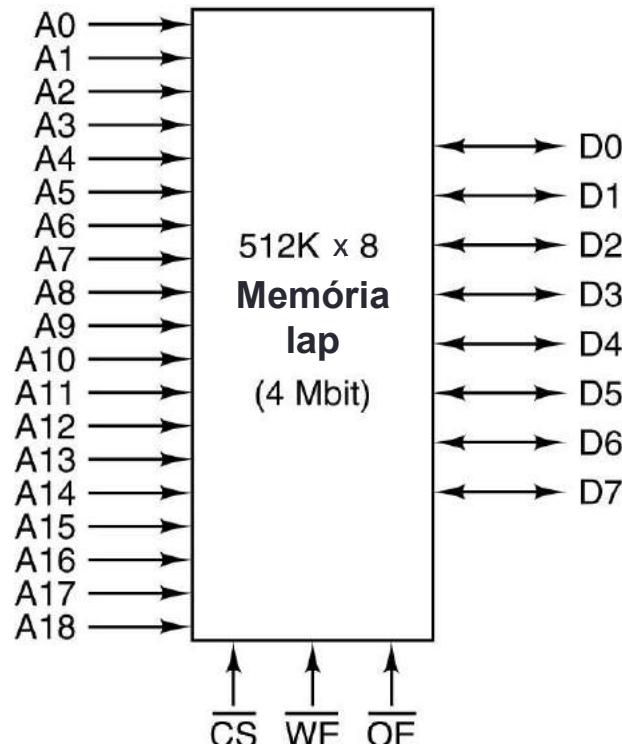
Memóriaszervezés

- Elektronikus kapcsoló
 - Olvasás esetén VAGY kapuk összekötése a kimeneti vonalakkal
 - Írás esetén teljesen leválaszt
- Nem invertáló puffer
- Háromállapotú eszköz
 - 0, 1 vagy semmi (nyitott áramkör)

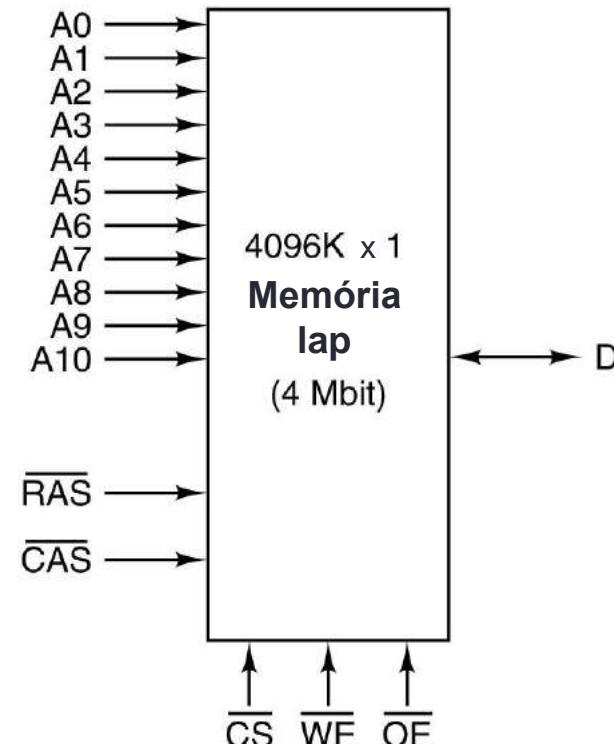


Memórialapkák

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



(a)



(b)

- 19 cím, 8 adat vonal

- 11 cím, 1 adat vonal
- RAS – sorcím
- CAS – oszlopcím

RAM-ok és ROM-ok

Típus	Kategória	Törlés	Bájt-változtatás	Felejtés	Tipikus használat
SRAM	Olvasás/írás	Elektromos	Igen	Igen	2-es szintű gyorsítótár
DRAM	Olvasás/írás	Elektromos	Igen	Igen	Fő memória
SDRAM	Olvasás/írás	Elektromos	Igen	Igen	Főmemória
ROM	Csak olvasás	Nem lehet	Nem	Nem	Nagy berendezések
PROM	Főleg olvasás	Nem lehet	Nem	Nem	Apró berendezések
EPROM	Főleg olvasás	UV-fény	Nem	Nem	Prototípusok
EEPROM	Főleg olvasás	Elektromos	Igen	Nem	Prototípusok
Flash	Olvasás/írás	Elektromos	Nem	Nem	Digitális kamera

Hibajegyzék

- A. S. Tanenbaum: Számítógép-Architektúrák
2. átdolgozott, bővített kiadás
- Hibák
 - 173-ik oldal
 - ... mert nem tudja, hogy mi lesz az Átvitel be a **felső** 16 bit összeadása után.
 - ... mert nem tudja, hogy mi lesz az Átvitel be a **alsó** 16 bit összeadása után.
 - 173-ik oldal
 - Megjegyezzük, hogy A+B egy aritmetikai összeadást, és nem Boole **ÉS**-t jelent.
 - Megjegyezzük, hogy A+B egy aritmetikai összeadást, és nem Boole **VAGY**-t jelent.
 - 177-ik oldal utolsó bekezdés
 - A bemenet a felső kapunál így 1 és 0, és a **$Q-t$** kimenetet 0-ba állítja.
 - A bemenet a felső kapunál így 1 és 0, és a **\bar{Q}** kimenetet 0-ba állítja.
 - 182-ik oldal
 - ... ahol nemcsak **\bar{Q}** és az **Törlő** vonal hiányzik ...
 - ... ahol nemcsak **\bar{Q}** és az **Előre beállító** vonal hiányzik ...

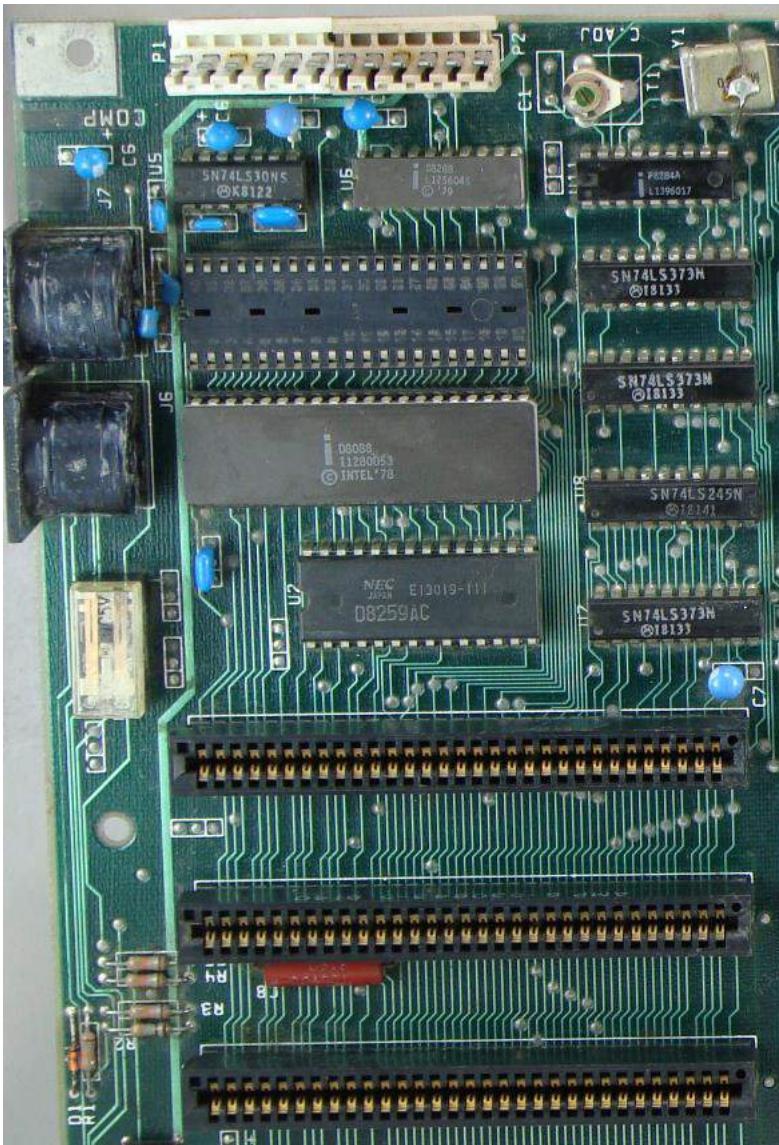
SZÁMÍTÓGÉP ARCHITEKTÚRÁK

A digitális logika szintje II.
CPU lapkák, sínek, B/K lapkák

Tanács Attila

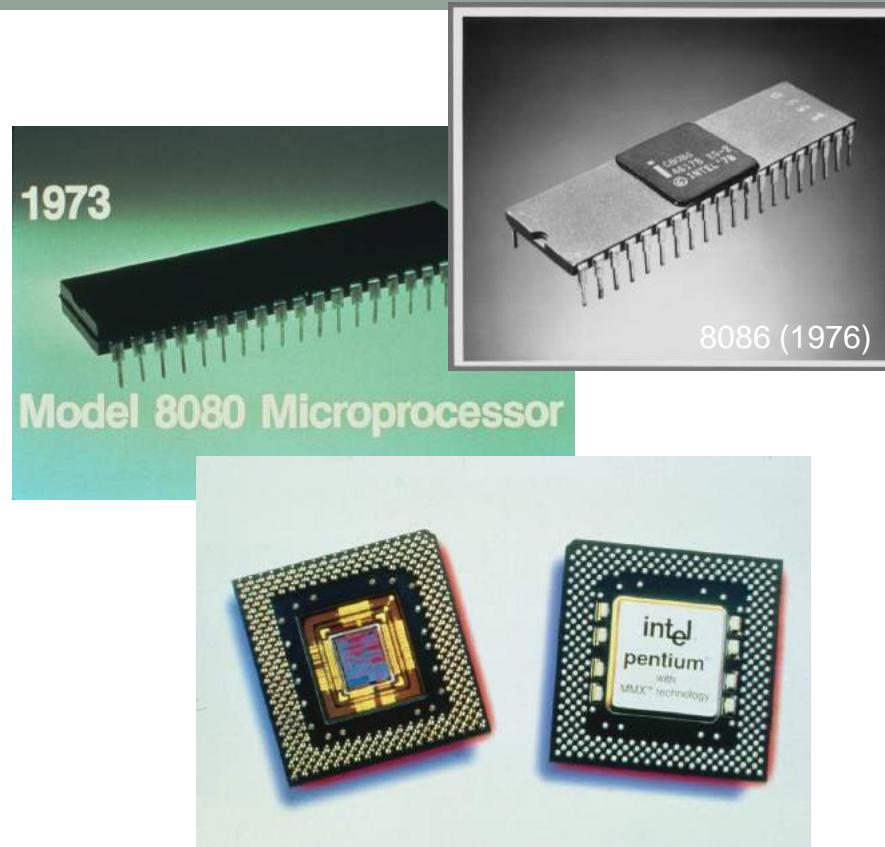
Témakörök

- **CPU lapkák**
 - Lábak
- **Számítógépes sínek**
 - Szélesség, időzítés, sínütemezés, sínműveletek
- **Példák CPU lapkára**
 - Pentium 4, UltraSPARC III, 8051
 - Intel Core i7, TI OMAP4430, ATmega168
- **Példák sínekre**
 - IBM PC, ISA, VESA, PCI, AGP, PCI Express, USB
- **Kapcsolat a perifériákkal, interfészek**
 - B/K lapkák

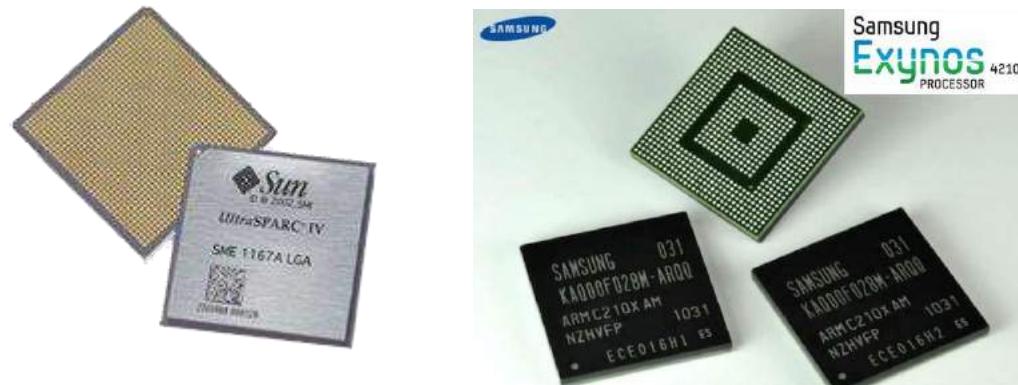


CPU lapkák

- Általában egyetlen lapka
- Lábkészlet (*pins*)
 - Kommunikáció a többi egységgel
 - Kimeneti jelek, bementi jelek, minden kettő
 - Cím, adat, vezérlés
- Kapcsolódás a memóriához, B/K eszközökhöz, ...
- Jeleket küld a lábakon
- Párhuzamos huzalok: sín



Képek forrása: [Intel Museum](#)



Kép forrása: [WikiMedia Commons](#)

Kép forrása: [Samsung](#)

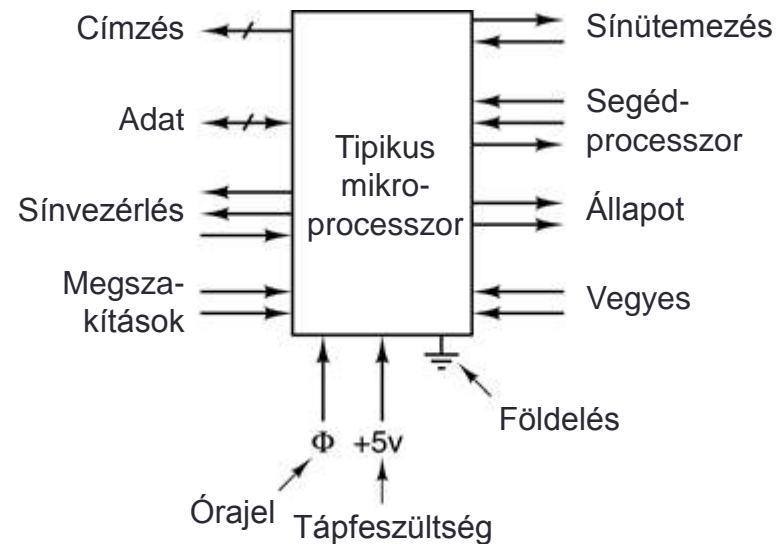
CPU lapkák

- **Címlábak száma**

- m címlábbal 2^m memóriahely címezhető
- Szokásos értékek: 16, 20, 32, 64

- **Adatlábak száma**

- n adatlábbal n bites szavak olvashatók/írhatók egy művelettel
- Szokásos értékek: 8, 16, 32, 36, 64, 128
- Több adatlábbal gyorsabb az olvasás/írás, de drágább a gyártás!



Ábra magyarázat

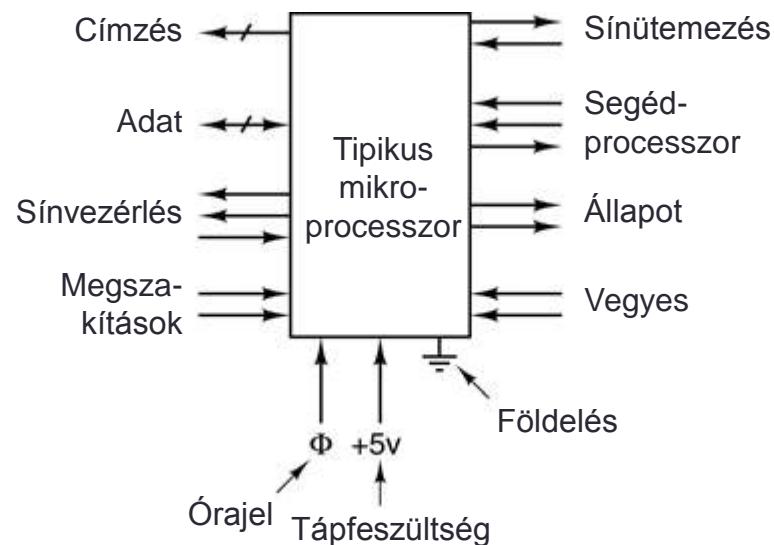
Nyilak a kimenetnek, bemenetnek megfelelően.
A rövid átlós vonalak a többszörös lábakat jelzik.

Forrás: Tanenbaum

CPU lapkák

- **Vezérlőlábak**

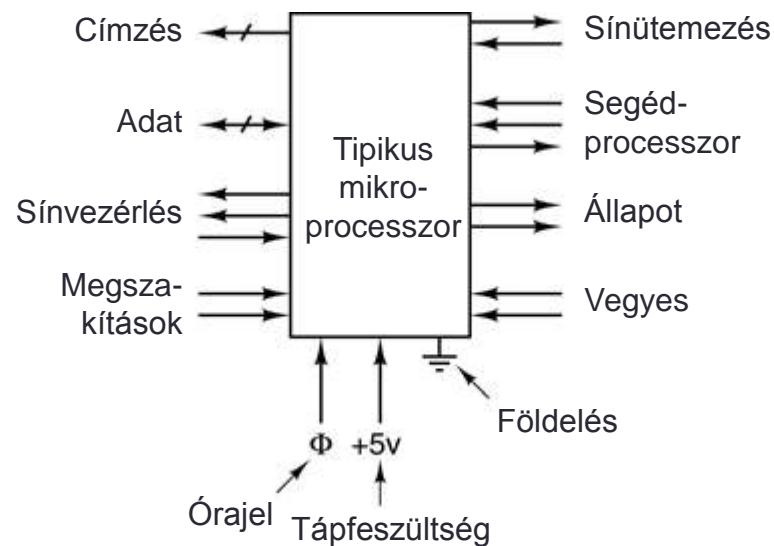
- Áramellátás (+3,3 vagy 5 volt), földelés, órajel
- Sínvezérlés
 - Mit akar tenni a CPU
- Megszakítások
 - Pl. a lassú B/K eszköz jelzi a feladat elvégeztét, CPU ezt nyugtázhatja
- Sínütemezés/kiosztás
 - Forgalomirányítás: csak 1 eszköz használja a sínt
- Segédprocesszor jelei
- Állapot
- Vegyes
 - Alaphelyzetbe állítás, kompatibilitás



Forrás: Tanenbaum

CPU lapkák

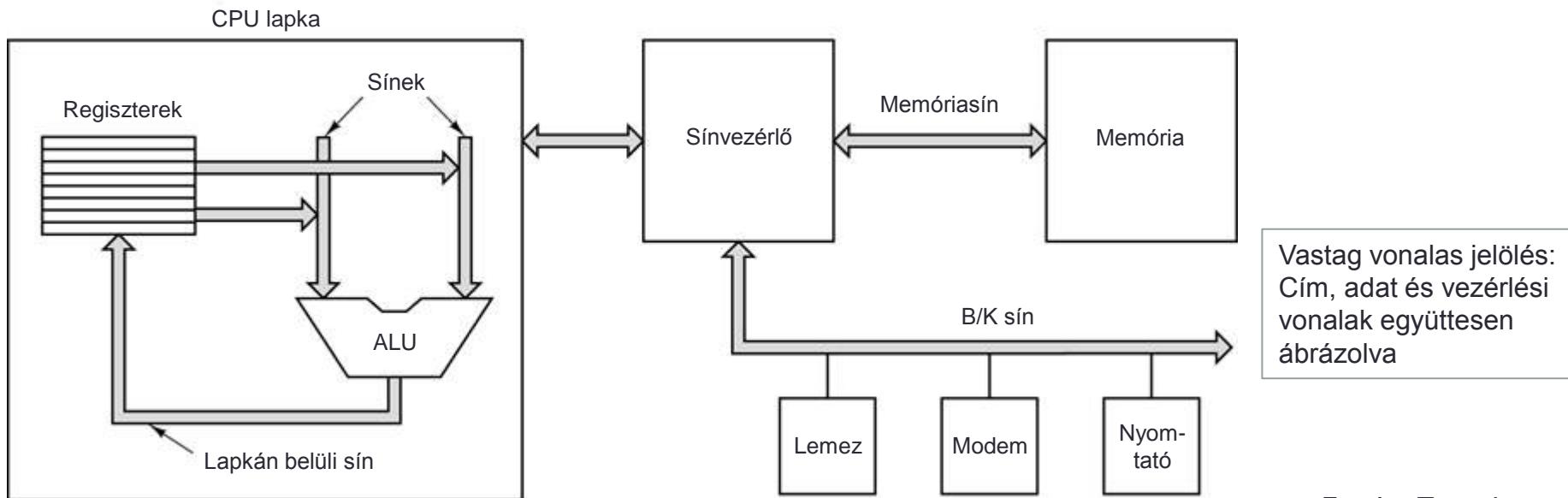
- Példa: utasítás betöltése digitális logikai szintjén (vázlat)
 - Memóriacím beállítása a címlábakon (CPU)
 - Memória informálása az olvasási igényről: vezérlővonalakon (CPU)
 - Kért szó CPU adatlábakra helyezése (memória)
 - Vezérlőjel beállítása a válaszról (memória)
 - Szó elfogadása, utasítás végrehajtása (CPU)



Forrás: Tanenbaum

Sínek (Bus)

- Eszközök közötti közös elektronikus pálya
- **Belső sín**
 - CPU egységein belül
- **Külső sín**
 - CPU és memória, B/K eszközök, tárprocesszorok, ... között



Sínek

- **Sínprotokoll**

- Belső sín szabadon tervezhető
- Külső sín esetén: **szabványok!**
 - Harmadik fél által tervezett kártyák is használhatók legyenek
 - Mechanikai és elektronikus előírások: ISA, EISA, PCI Express, USB, ...

- **Aktív/passzív berendezések**

- Mester (aktív): átvitelt tud kezdeményezni (CPU, lemezvezérlő, ...)
- Szolga (passzív): kérésekre vár (memória, CPU, lemezvezérlő, ...)
 - A memória csak szolga lehet!
- A szerepek tranzakciónként változhatnak

Mester	Szolga	Példa
CPU	Memória	Utasítások és adatok betöltése
CPU	B/K eszköz	Adatátvitel kezdeményezése
CPU	Segédprocesszor	Utasítás átadása a segédprocesszornak
Segédprocesszor	CPU	Operandusok átvétele a CPU-tól
B/K	Memória	Direkt memóriaelérés (DMA)

Sínek

- **Kapcsolódások**

- Sok berendezés vagy hosszú sín esetén digitális erősítés szükséges
 - Sínvezérlő: sínmester lapkák esetén
 - Sínvevő: szolga kapcsolatok
 - Sínadóvevő: mesterként és szolgaként is működő eszközök

- **Sínkapcsolat-típusok**

- Hárromállapotú: mester, szolga, lekapcsolódás a sínről
- Nyílt gyűjtők: huzalozott-VAGY elrendezés

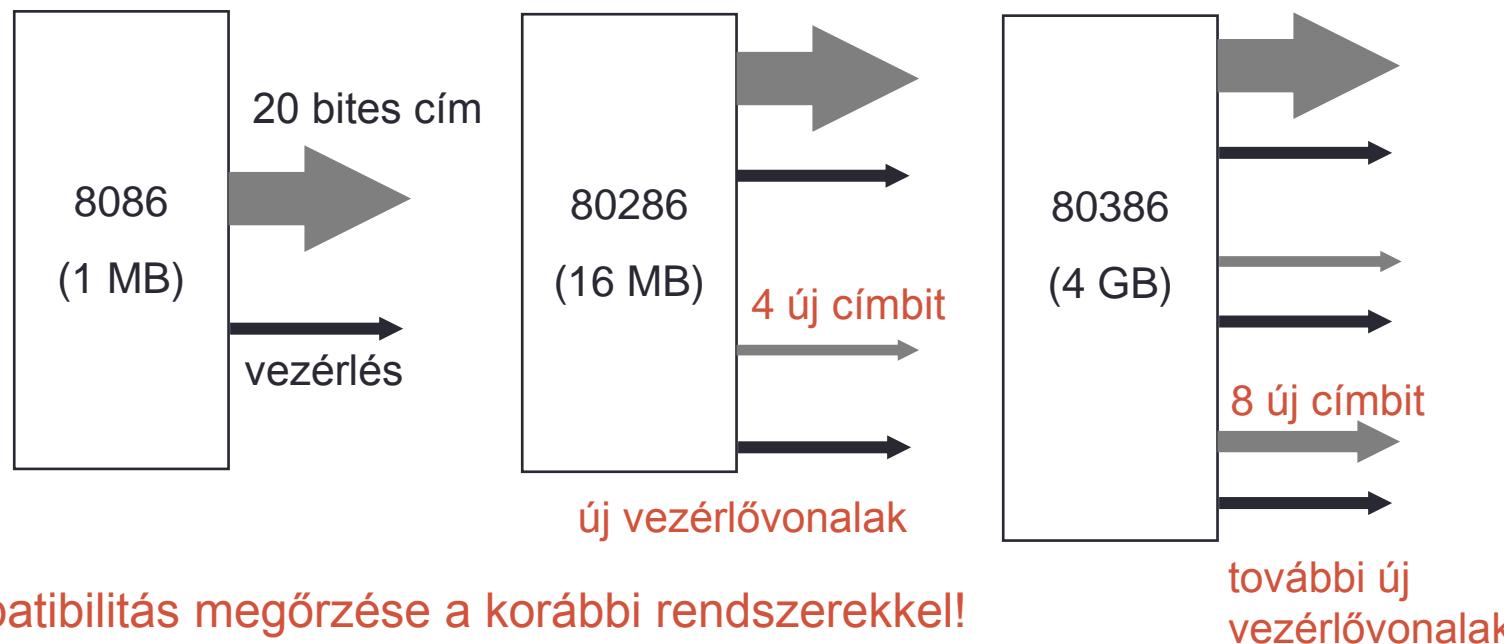
- **Sín vonalak**

- Cím-, adat- és vezérlővonalak
- Dekódoló lapka: ha ezek eltérnek a CPU lábkiosztásától

Sínszélesség

- **Címvezetékek**

- Több sínvezeték → nagyobb fizikai hely, drágább, kompatibilitási probléma korábbi rendszerekkel
- Csak az éppen elegendő számú sínvezeték: olcsóbb, de később nem bővíthető



Sínszélesség

• Adatvezetékek

- Sín savszélessége = sínszélesség x sín sebessége
 - Pl. 32 bites (4 bájt), 33,3 MHz esetén ~133 MB/mp
- Szélesség növelése (szokásos)
 - Kompatibilitás megőrzése esetén komplex konstrukció
- Sínciklus idejének csökkentése (nehézkes)
 - Sín aszimmetria probléma (egyes vezetékeken más a jelek terjedési sebessége)
 - Kompatibilitási probléma: korábbi lassabb eszközök nem működnek

• Multiplexelt sín

- Címek és adatok ugyanazokat a vezetékeket használják, megfelelően ütemezve
- Lassabb rendszer: nem lehet egyszerre címet és adatot beállítani a sínen
- Bonyolultabb: összetettebb sínprotokoll szükséges
- De olcsóbb: kisebb sínszélesség elegendő

Sínek időzítése

• Szinkron sín

- 5 és 100 MHz közötti frekvenciájú négyszögjel vezérlí (sínciklus)
- Síntevékenységek ennek egész számú többszöröséig tartanak
 - Tétlen várakozás előfordulhat
- A leglassabb eszközökhöz kell a sín sebességét igazítani
- Későbbi gyorsabb eszközök előnyei nem használhatók ki
 - Hiába tudja feleannyi idő alatt szolgáltatni az adatot a memória, ha korábban rögzítésre került a memóriaolvasás cikluseideje
- Ez a gyakoribb technika, könnyebb megépíteni

• Aszinkron sín

- Nincs órajel-generátor
- Változó hosszúságú sínciklusok
- A ciklusok váltásához szinkronizálás szükséges (protokoll)

Szinkron sín időzítése

- **Síntevékenységek**

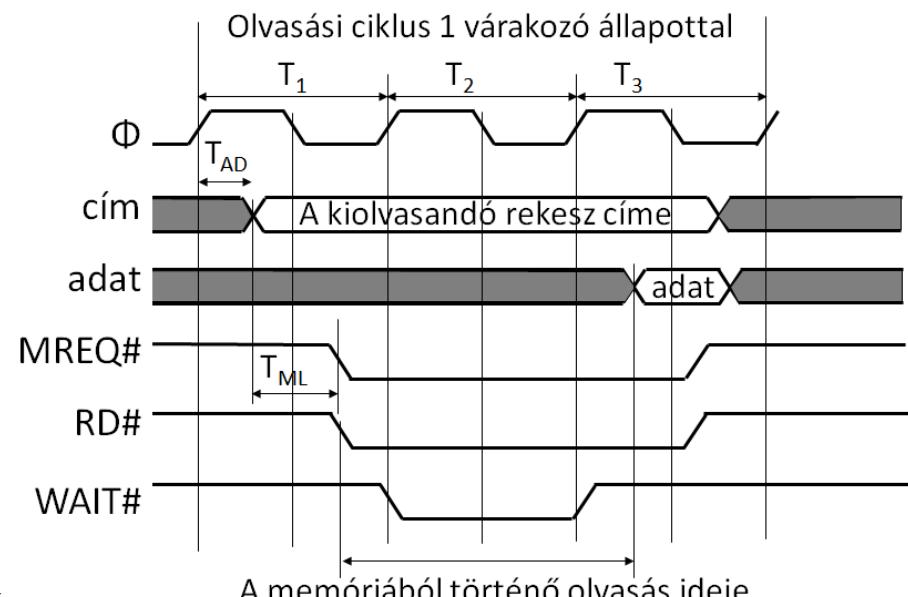
- Cím megadása, vezérlőjelek beállítása (MREQ#, RD#, WAIT#), ...
- Az egyes tevékenységekhez minimális vagy maximális időtartam megadása szükséges

Példa:

- Sínciklus 10 ns
- 1 jel változásának ideje: 1 ns
- T_{AD} (cím megérkezése a sínre): max. 4 ns
- T_{ML} (cím a sínen van MREQ# előtt): min. 2 ns
- ...
- Olvasás memóriából: 15 ns

Szó olvasása a memóriából: 3 ciklus alatt

- 1. ciklus: cím a sínre, MREQ#, RD#
- 2. ciklus: WAIT# beállítása, mert 1 ciklus alatt nem végez a memóriaolvasás ($15 \text{ ns} > 10 \text{ ns}$)
- 3. ciklus: WAIT negálása, mert ebben a ciklusban meglesz az adat, adat megjelenik, T3 lefutó élén CPU olvas, majd MREQ és RD negálás



Forrás: Tanenbaum

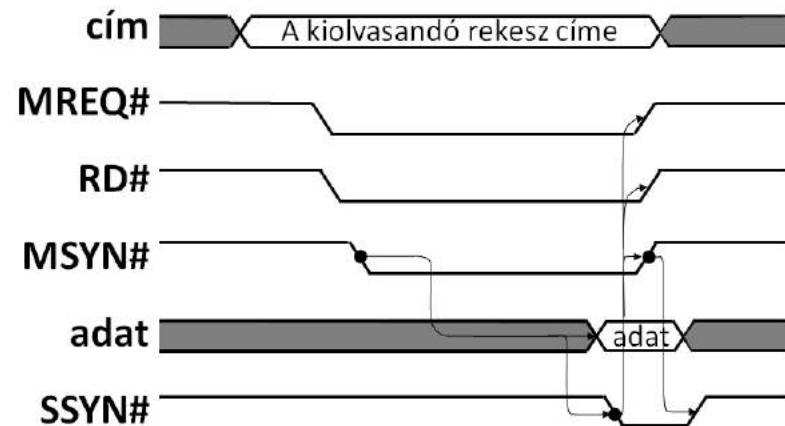
Aszinkron sín időzítése

- **Síntevékenységek**

- Nincs időzítés, minden eseményt egy előző esemény okoz
- WAIT helyett itt MSYN# (mester kérés) és SSYN# (szolga kész)
- Ugyanazon sínen gyors és lassú mester-szolga pár is lehet

„Teljes kézfogás” példa memóriaolvasásra

- SSYN negált állapotában indulhat csak!
- Cím beállítás (sínmester)
- MREQ és RD jelek (sínmester)
- **MSYN jel** (mesterszinkronizáció)
- Szolga elvégzi a feladatot
- **SSYN jel** (szolgaszinkronizáció)
- Adatok tárolása (sínmester)
- Címvezetékek, MREQ, RD, **MSYN jel** negálása (mester)
- **SSYN jel negálása** MSYN változásának hatására (szolga)



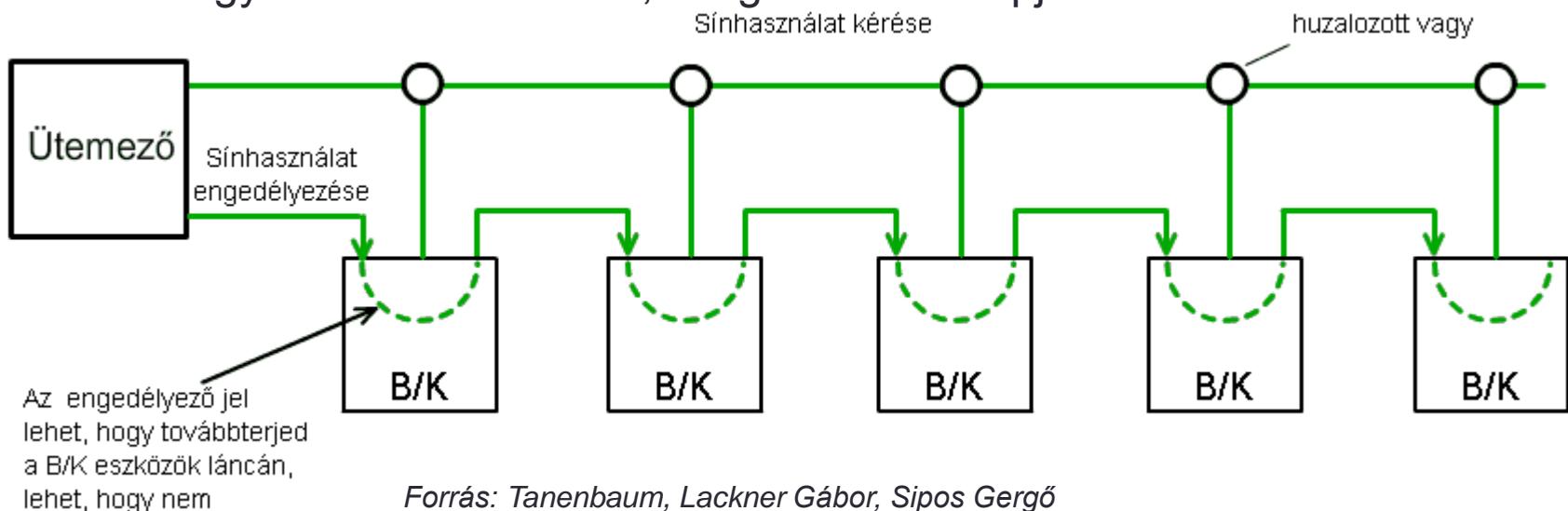
Sínütemezés

- **Feladat**
 - Sínhez mesterként hozzáférés szabályozására
- **Centralizált**
 - Dedikált ütemező lapka (CPU-ba integrálva vagy külön) és
 - eszközök sorba kötve + huzalozott-VAGY sín a kérésekhez
- **Decentralizált**
 - minden eszköz összekapcsolva egymással, vagy
 - sorba kötött eszközök, ütemező lapka nélkül

Centralizált sínütemezés

- **Egyszintű láncolás (daisy-chaining)**

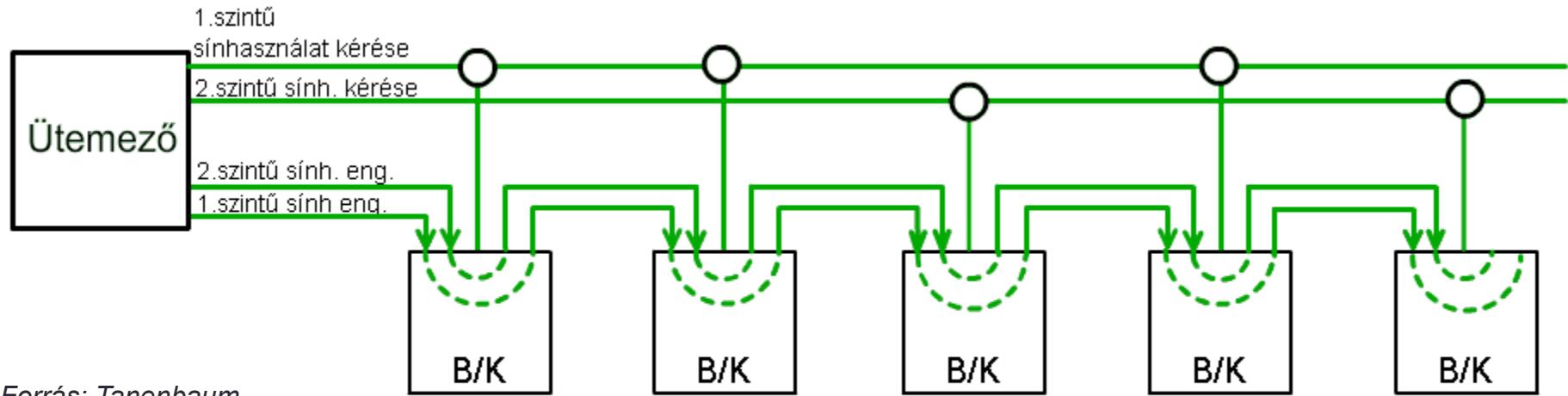
- B/K eszköz jelzi az igényét a huzalozott-VAGY vonalon
- Az ütemező az első eszköznek ad engedélyező jelet
 - Csak a kérést látja, hogy ki kérte, azt nem
- Ha nem Ő kérte a hozzáférést, akkor továbbadja a szomszédnak
- Így az ütemezőhöz legközelebbi nagyobb prioritást élvez
 - Ha egyszerre többen kérlik, a legközelebbi kapja



Centralizált sínütemezés

- **Többszintű láncolás**

- Különböző prioritási szintek kezelésére
 - Az eszközök ismerik a prioritásukat
 - Ha van kérés a magasabb prioritású vonalon, az kapja
 - Azonos prioritási szinten láncolás
- Példában kettő, de a gyakorlatban 4, 8 vagy 16 szint is lehet
- A legidőkritikusabb eszközök magasabb prioritású vonalon
- Pl. B/K eszköz magasabb, a CPU általában alacsonyabb prioritású



Forrás: Tanenbaum,
Lackner Gábor, Sipos Gergő

Decentralizált sínütemezés

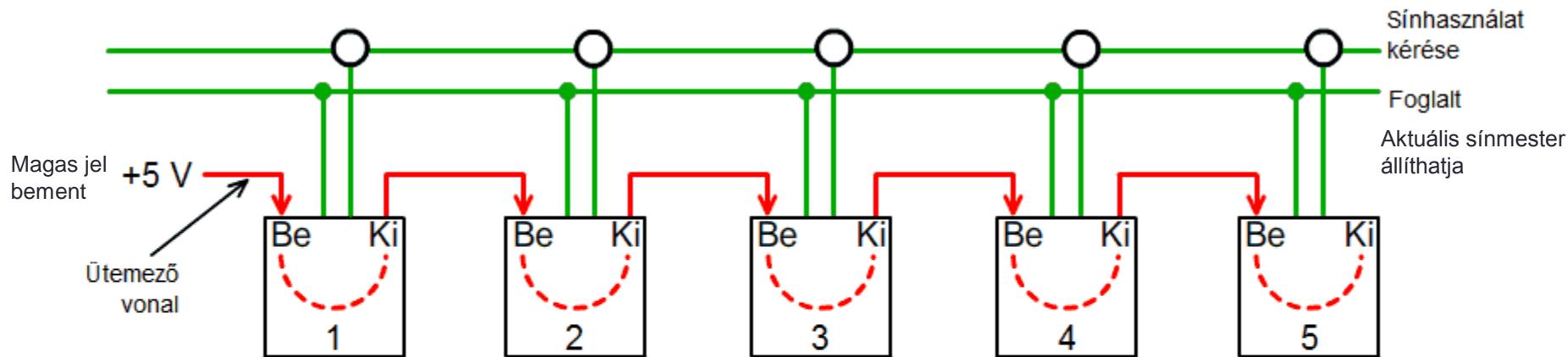
- **Teljes összekötés**

- minden eszköz mindegyikkel összekötve
- Sínhasználati kérés mindenikhez befut
- A prioritásaiak ismeretében tudják ki kapja a sínt
- Több sínt igényel, az eszközök száma kötött

Decentralizált sínütemezés

- **Decentralizált láncolás**

- Tetszőleges számú eszköz láncba fűzve
- Az ütemező jel végigfut a láncon, ezen változtathatnak az eszközök
 - Ha *nem foglalt* és *Be* magas, akkor kérheti a sínt (*Ki* alacsonyra, *foglalt be*)
 - Ha *Be* alacsony, akkor *Ki* alacsonyra állítódik és nem kaphatja meg a sínt
- Egy teljes kör végigfutása után derül ki, hogy ki kapja a sínt
 - Akinél a *Be* magas és a *Ki* alacsony



Sínműveletek

- **Egy szó továbbítása a sínen**
 - Korábbi példákban
- **Blokkátvitel**
 - Nagyobb mennyiségű adat optimálisabb továbbítására
 - A cím mellett a mozgatandó adatok száma is a sínre kerül
- **Többprocesszoros rendszerek**
 - Szemafor használat a közös elérésű területek kezelésére
 - *Olvasás-módosítás-visszaírás ciklus*: nem kell elengednie a sínt, amíg nem végez, hogy más CPU ne zavarja

Sínműveletek

- **Megszakítások**

- Lassú B/K eszközök esetén a CPU kiadja a kérést és nem vár a válaszra
- A B/K eszköz megszakítási kérelmen keresztül jelzi, hogy végzett a feladattal
- A megszakítási kérelem jelzéséhez szükség van a sínre
- Kezelni kell az egyidejű kérelmeket (sínütemezés)
 - Prioritások beállíthatók
 - CPU vagy dedikált megszakításvezérlő lapka ütemez
 - Pl. Intel 8259A

Sínműveletek

- **Intel 8259A lapka**

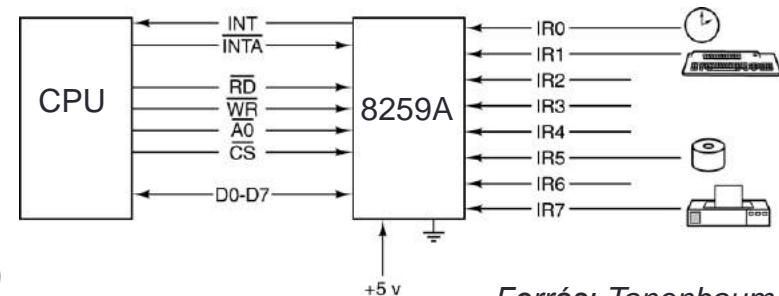
- Maximum 8 B/K lapka kezelhető
 - De sorba köthetők (pl. 8 lábra 1-1 8259A: 64 eszköz)

- **Működése**

- Megszakítási kérelem az IRx vonalon
- INT vonal beállítása (8259A)
- INTA visszajelzés (CPU)
- Bemenet sorszáma adatsínre (8259A)
- Megszakítás kezelése (CPU): hívandó eljárás kezdőcíme megszakításvektor táblából, végrehajtás
- Ha kész, a CPU egy lapka regiszter írásával ezt jelzi
 - INT negálódik, vagy ha van várakozó megszakítás, az aktiválódik

- **Lapka regiszterek**

- RD (olvasás), WR (írás), CS (lapkaválasztás)
- Egyes megszakítások letilthatók
- Sorba kötés vezérelhető

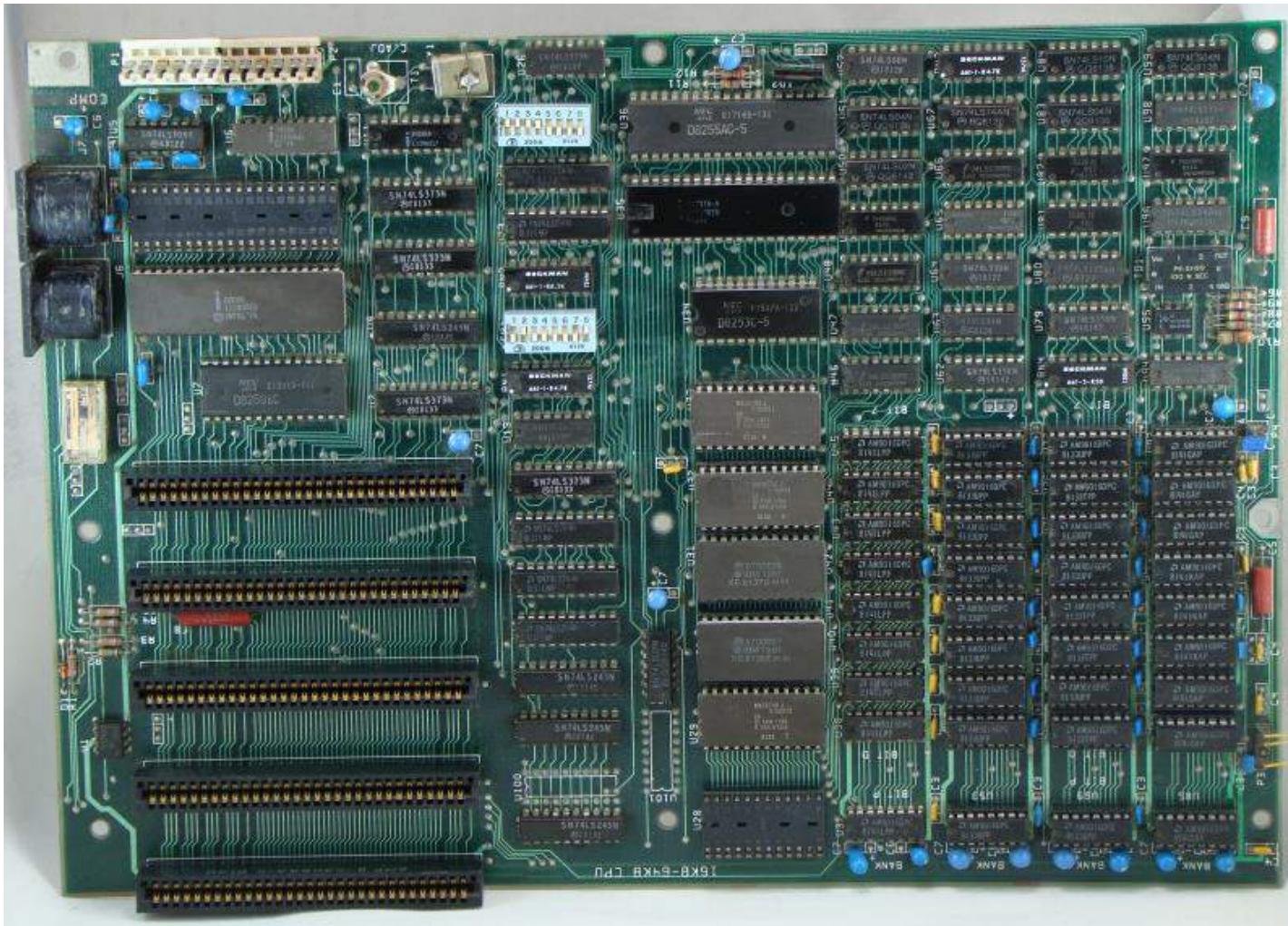


Forrás: Tanenbaum

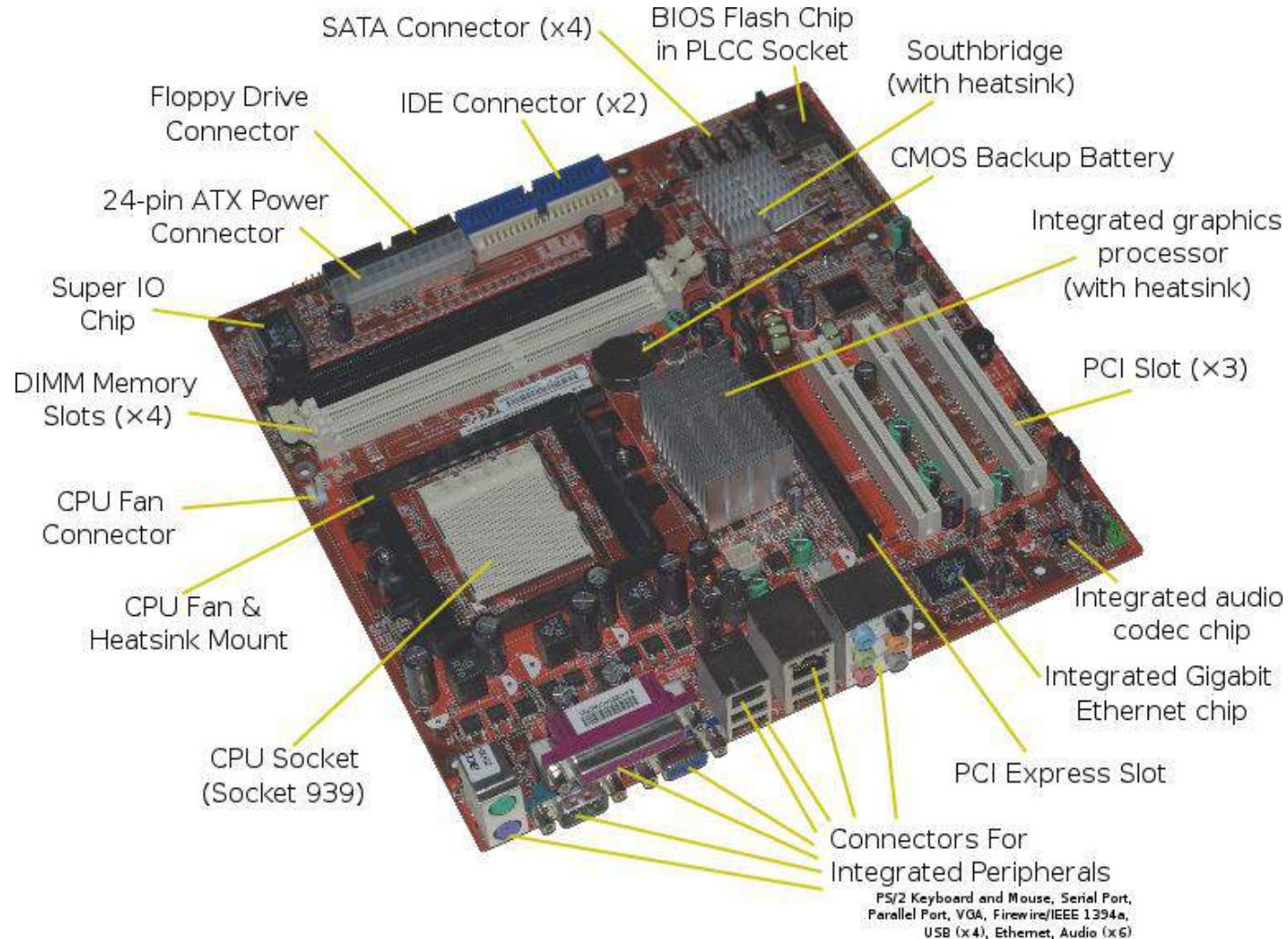
Alaplapok (motherboard)

- **Tartalmazzák**
 - CPU-t
 - Síneket
 - Illesztőhelyek bővítőkártyáknak
 - Központi memóriát
 - ...
- **Beviteli/kiviteli eszközök**
 - Külön kártyán
 - Alaplapra integrálva
- **Moduláris felépítés**
 - A külső eszközök később külön cserélhetők, fejleszthetők
 - A lapra integrált eszközök csak alaplap cserével

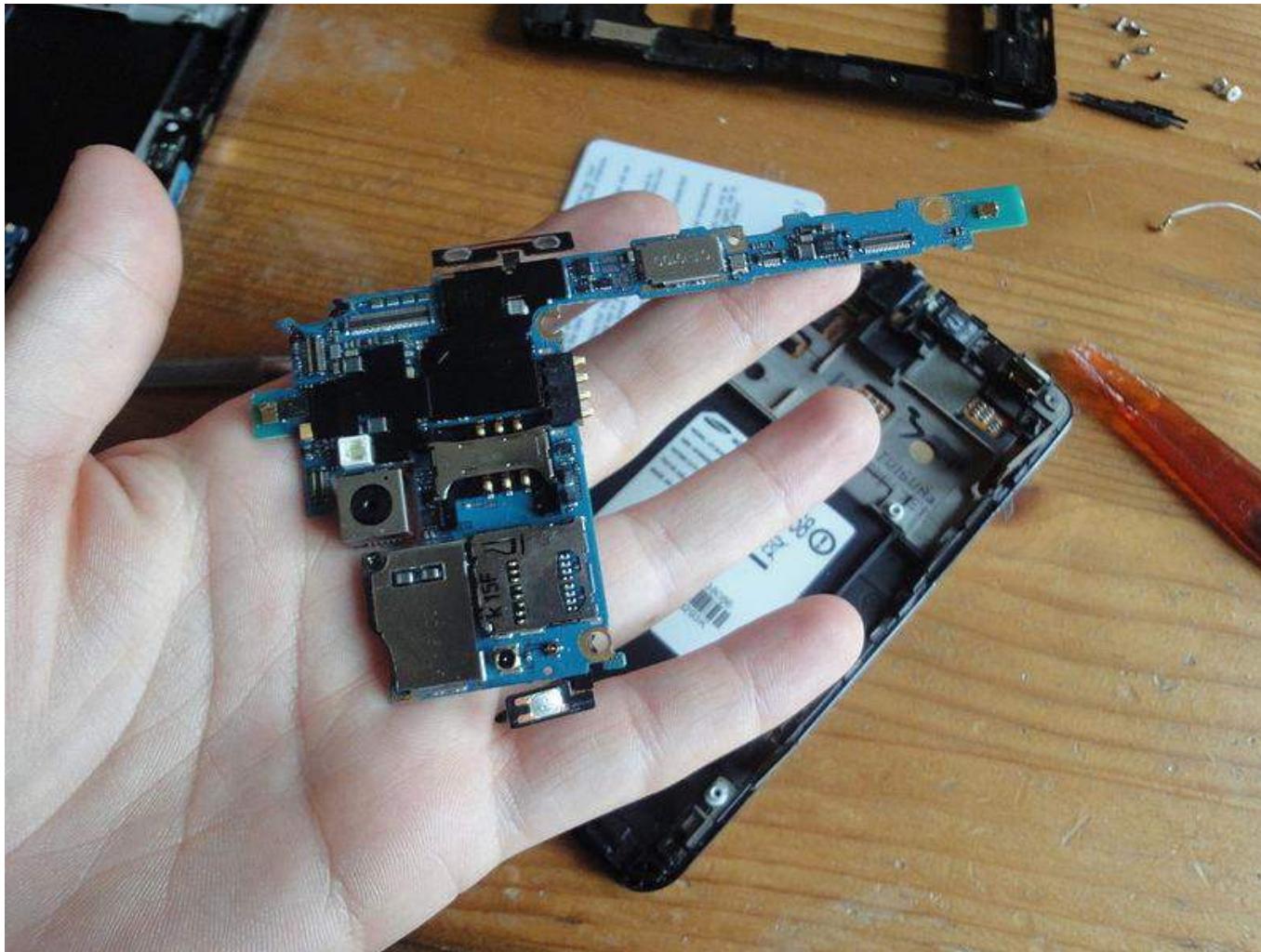
IBM PC alaplap (1970-es évek vége)



Acer E360 Socket 939 (Foxconn)



Samsung Galaxy SII okostelefon (2011)



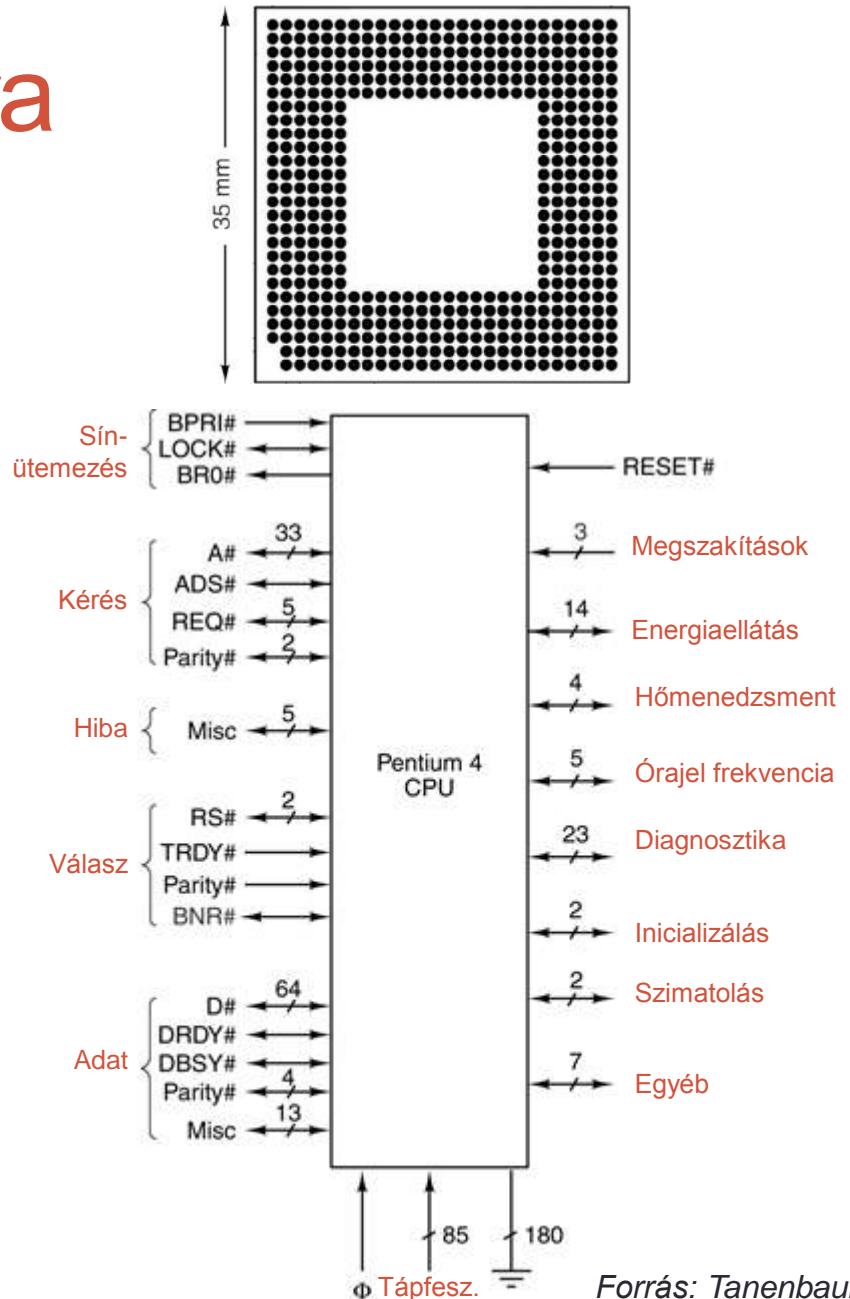
Példák CPU lapkára

- **Intel 8086 (1978)**

- Kb. 29 ezer tranzisztor
- 5 MHz-es működés
- 16 bites regiszterek és külső adatsín
- 20 bites memóriacímzés (1 MB)

- **Intel Pentium 4 (2000)**

- 42 – 55 millió tranzisztor
- 1,5 – 3,2 GHz
- 0,18 – 0,09 mikron vezetékszélesség
- 60 – 80 Watt fogyasztás!
 - Hőelvezetés szükséges
- 478 láb
 - 26x26 elrendezés ($14 \times 14 + 2$ hiányzik)
 - 198 jel, 85 tápf., 180 föld
 - 15 tartalék
- Két szinkron elsődleges sín
- 5 állapot: aktívtól mély alvásig
 - Felébresztés külső jelre
- 32 bites CISC mag, de 64 bites adatátvitel, 8-cal osztható kezdőcímről
- 33 címvezeték (36 bit, de az alsó három 0 értékű), 64 GB címezhető



Forrás: Tanenbaum

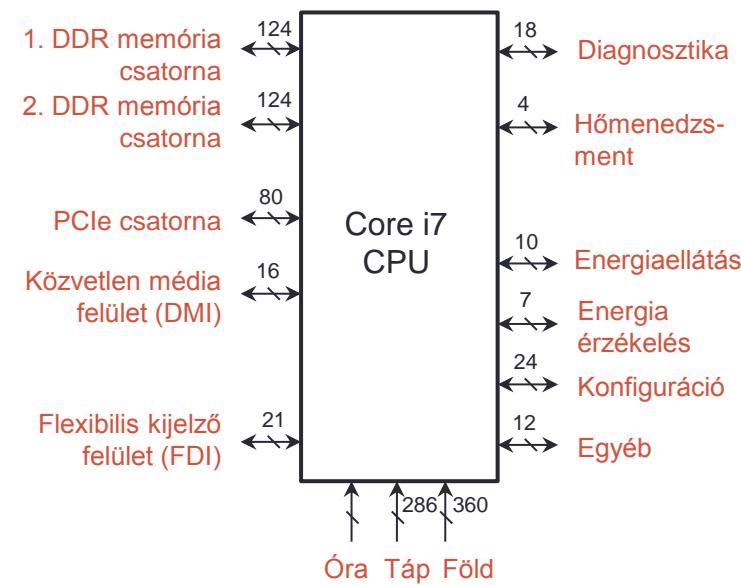
Kiegészítő anyag!

Példák CPU lapkára

- **Pentium Core processzorok (2008-)**
 - i3: alsó-, i5: közép-, i7: felsőkategória
 - Nahalem, Sandy Bridge, Ivy Bridge architektúrák
 - 2, 4, 6, ... processzormag, magonként hiperszásnak
 - Akár 8088 ISA kompatibilitás...
 - CISC + integrált RISC mag
- **Pentium Core i7 (Nahalem, Sandy Bridge)**
 - 731 – 1160 millió tranzisztor
 - 45 – 32 nanométer vezetékszélesség
 - 3,2 – 3,5 GHz
 - 17 – 150 Watt fogyasztás!
 - Hőelvezetés szükséges
 - 1155 láb
 - 40x40 elrendezés (17x25 + 20 hiányzik)
 - Ebből 447 jel, 286 táp, 360 földelés, 62 fenntartott
 - 64 bites processzormagok
 - 4-széles szuperskalár gép
 - Két elsődleges külső szinkron sín
 - DDR3 memoriához, PCI Express perifériákhoz
 - 5 processzor állapot: teljesen aktívtól mély alvásig
 - Felébresztés külső jelre



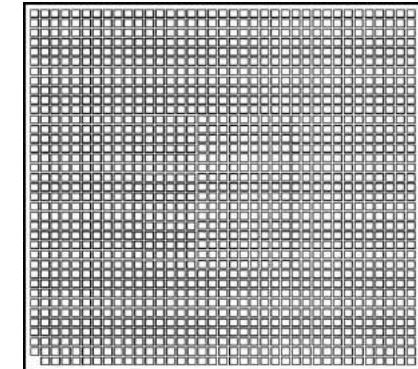
Intel Core i7 lábkiosztás
Forrás: [TechReport](#)



Forrás: Tanenbaum

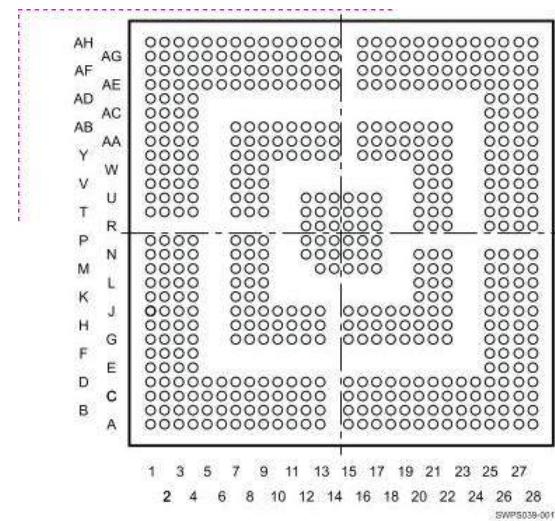
Példák CPU lapkára

- **SUN UltraSPARC III (2000)**
 - 29 millió tranzisztor
 - 600 MHz – 1,2 GHz
 - 0,18 – 0,15 mikron
 - 50 Watt fogyasztás
 - 1368 láb, 37x37 elrendezés – 1 láb
 - 6 belső csővezeték
 - 64 bites RISC mag, 128 bites adatátvitel
 - 43 bites címsín (8 TB címezhető)
 - 150 MHz sín (2,4 GB/mp > PCI!)



Példák CPU lapkára

- **Texas Instruments OMAP4430** (2011)
 - System-on-a-chip (SoC): integrált vezérlők + perifériák = komplett rendszer
 - Két **ARM A9** mag
 - 2-széles szuperskalár mikroarchitektúra
 - 2 utasítás dekódolása és végrehajtása órajelciklusonként – jellemzően nincs kihasználva
 - **POWERVR SGX540** grafikus processzor
 - **ISP** processzor
 - képalkotáshoz, képfeldolgozáshoz
 - **IVA3** videó processzor
 - kódolás/dekódolás
 - 1 GHz, 45 nanométer vezetékszélesség
 - A használaton kívüli processzorok kikapcsolása
 - 547 láb (gömb alakú csatlakozó)



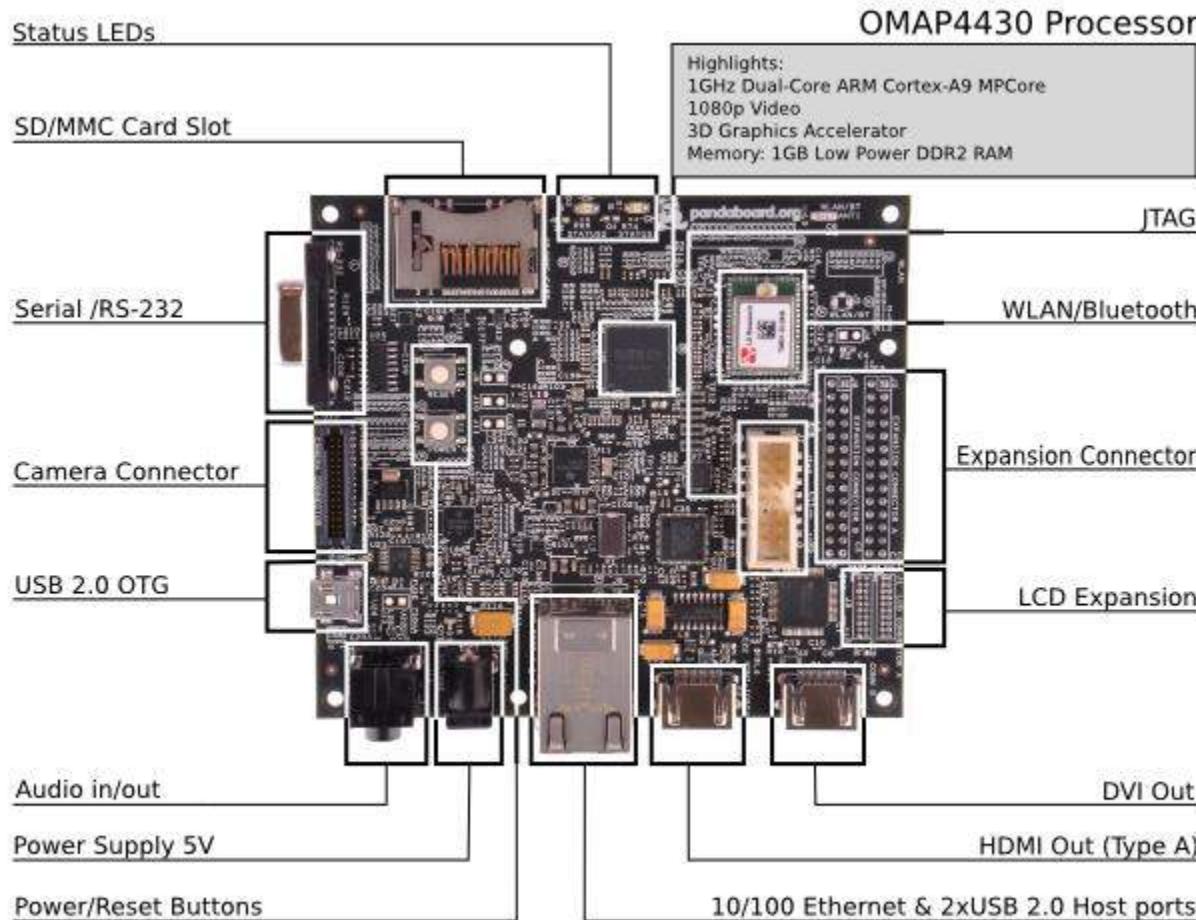
Forrás: [DataSheet Zone](#)

Példák CPU lapkáakra

- **Texas Instruments OMAP4430 (2011)**
 - Mobil és beágyazott eszközökben alkalmazzák (okostelefonok, táblagépek)
 - (Viszonylag) nagy számítási kapacitás, multimédia
 - Nagyon alacsony energiafogyasztás
 - 600 mW maximális fogyasztás (Core i7 250-ed része)
 - Alvó módban 100 µW
 - Dinamikus feszültségskálázás: alacsonyabb feszültségfelvétel esetén kisebb teljesítmény
 - **ARM utasításkészlet (RISC)**

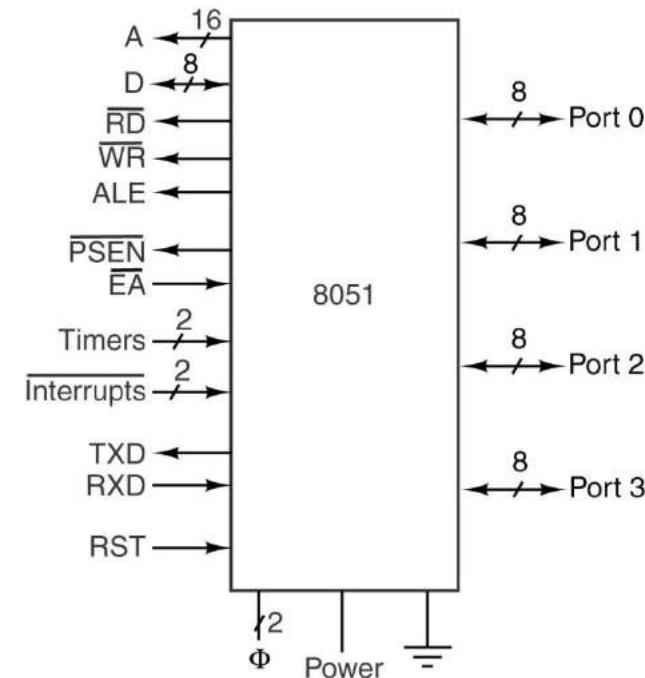
Példák CPU lapkáakra

- Texas Instruments OMAP4430 PandaBoard lapkán



Példák CPU lapkákra

- **8051 mikrovezérlő (1980-)**
 - Beágyazott rendszerekben
 - 40 lábú tokozás
 - 4 KB ROM, 128 bájt beépített RAM
 - Duál 16 címvezeték
 - 2x64 KB memória címezhető
 - 8 bites adatsín (adatátvitel bájtonként)
 - **32 B/K vonal**
 - Nyomógombhoz, kapcsolóhoz, LED-hez, ...
 - CISC, Harvard architektúra



Példák CPU lapkáakra

- **Atmel ATmega168 mikrovezérlő (1997-)**
 - Beágyazott rendszerek, kis számításigényű feladatok
 - Nagyon olcsó (~1 USD)
 - **28 lábú tokozás**
 - Nincsenek cím- és adatvonalak!
 - SRAM és Flash memória a processzorba integrálva
 - Kapcsolódás külső eszközökhöz
 - 23 digitális B/K vonal (B, D, C kapuk)
 - Szoftveresen konfigurálható bementként is kimenetként is
 - 6 vonal (PC0-PC5) analóg vonalként is beállítható
- **Memória**
 - Maximum 16 KB belső Flash (felejtő, ritkán változó tartalomra: programkód)
 - Maximum 1 KB EEPROM (rendszer konfigurációs adatok)
 - Maximum 1 KB SRAM (temporális változók tárolása)
- **AVR utasításkészlet**
 - 131 utasítás, 16 bit hosszúságúak
 - RISC elv, Harvard architektúra
 - 8 bites processzor, 8 bites regiszterek

PC6	1	28	PC5
PD0	2	27	PC4
PD1	3	26	PC3
PD2	4	25	PC2
PD3	5	24	PC1
PD4	6	23	PC0
VCC	7	22	GND
GND	8	21	AREF
PB6	9	20	AVCC
PB7	10	19	PB5
PD5	11	18	PB4
PD6	12	17	PB3
PD7	13	16	PB2
PB0	14	15	PB1



Példák sínekre

• IBM PC sín

- 8088 alapú rendszerekben
 - 4,77 MHz
 - 62 jelvezeték
 - 20 cím + 8 adat + memória írás/olvasás, B/K írás/olvasás, megszakításkezelés, DMA, ...
 - PC alaplapjára maratva
 - Csatlakozók, bővítőkártyák
 - Fül, 31 csatlakozó vezetéksávval
 - Sok, harmadik féltől származó kártya!

• 80286 tervezések

- 16 MB címezhető memória, 16 bites adat
 - A régi sínnel ez nem érhető el
 - **Kompatibilitás:** a régi bővítőkártyák használhatók maradjanak!
 - **Kibővített új sín: ISA**

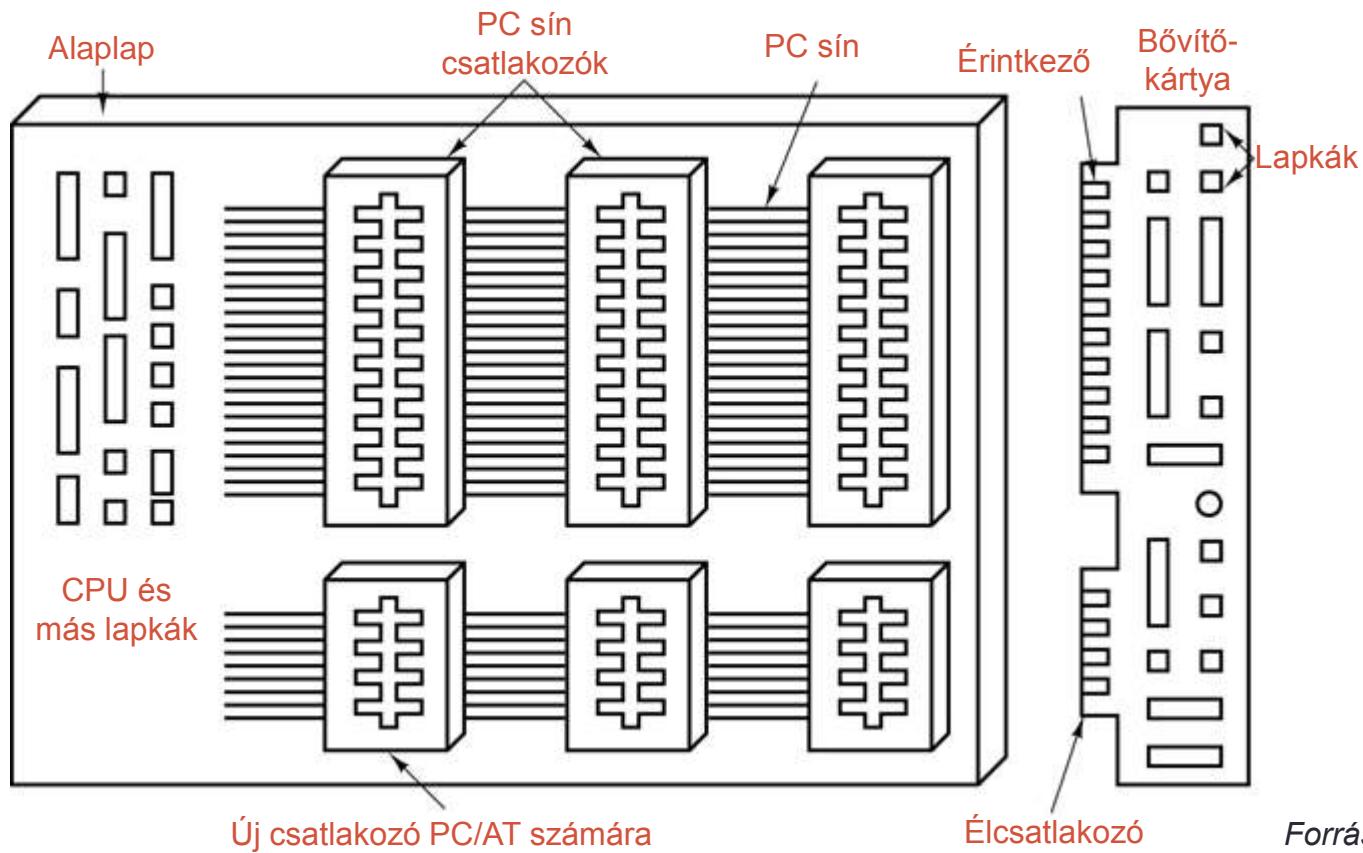


ISA sín (1981)

- Kibővített IBM PC sín



Forrás: [WikiMedia Commons](#)



- 36 vezetékes második élcsatlakozó
- Régi kártyák is használhatók
- 8,33 MHz
- Később: 32 bites bővítés (EISA sín)
- ISA csatlakozók sokáig megmaradtak az alaplapokon (pl. külön sínvezérlővel csatlakozva az új sínekre)

Forrás: [Tanenbaum](#)

VESA lokális sín és PCI sín

- **Grafikus felhasználói felületek megjelenése**

- (80386,) 80486: 1990-es évek eleje
 - Nagy mennyiségű adat mozgatása szükséges a grafikus hardver és a rendszer többi része között!
 - Egy 1024x768 RGB képernyőn 2,25 MB adat van. 30 Hz frissítés mellett ez 67,5 MB/mp átvitelt igényel
 - DVD lejátszás: képi adat háttértárról, dekódolás, átvitel videókártyára
 - 16 MB/mp kevés! (ISA)

- **VESA lokális sín**

- 32 bites, 33 MHz: 127 MB/mp
- Több hátrányos tulajdonsága miatt visszaszorul
 - Plug-and-Play és sínvezérlő képesség hiánya
 - Erős 80486 függés



Photo: Miha Ulanov, [WikiMedia Commons](#)

- **PCI sín**

- 32 bites, 33 MHz: 127 MB/mp
 - Plug-and-Play és sínvezérlés támogatás
 - Ez terjedt el
 - PCI 2.2: 64 bites, 66 MHz: 528 MB/mp
- ISA kompatibilitás megőrzése
 - Hidak: különféle sínrendszerek összekötésére

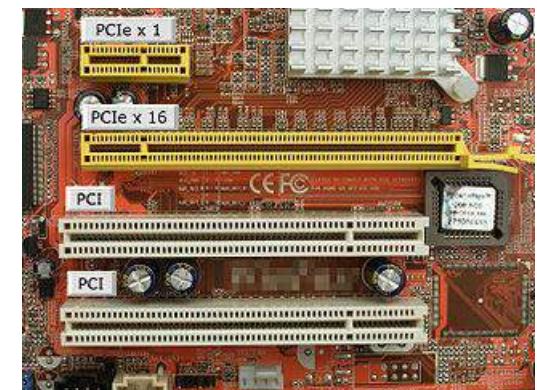
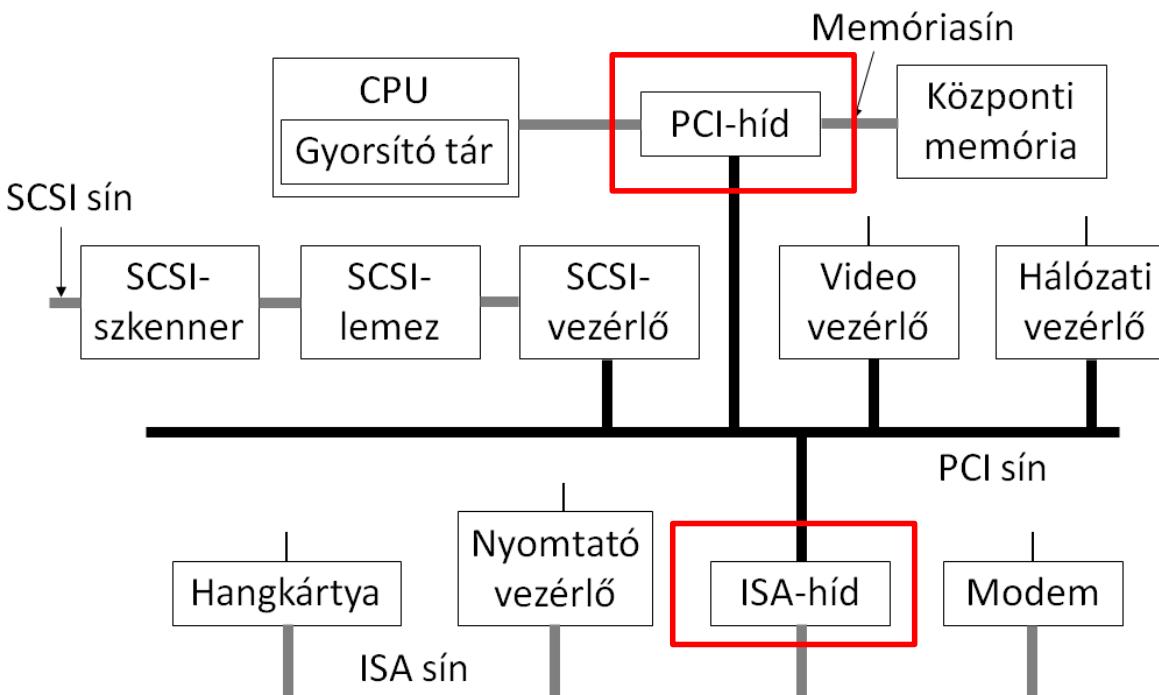


Photo: Rainer Knäpper, [Free Art License](#)

Hidak

- PCI, SCSI, és ISA sínek egy korai Pentium rendszerben



Főmemória gyorsan elérhető

- Saját sínen

PCI: gyors eszközök számára

- Multiplexelt, szinkron sín
- SCSI-lemezek
- Grafikus kártyák
- ...

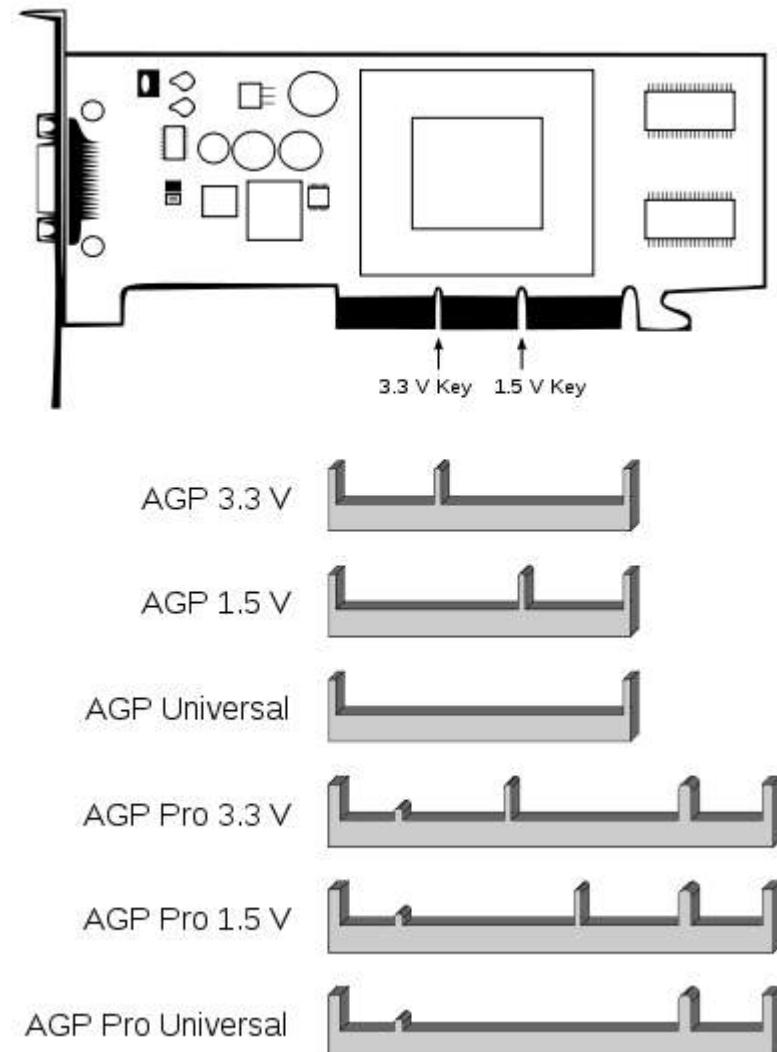
ISA: régi és lassabb eszközök

Többféle PCI változat

- Tápfeszültség
- Sínszélesség
- Frekvencia

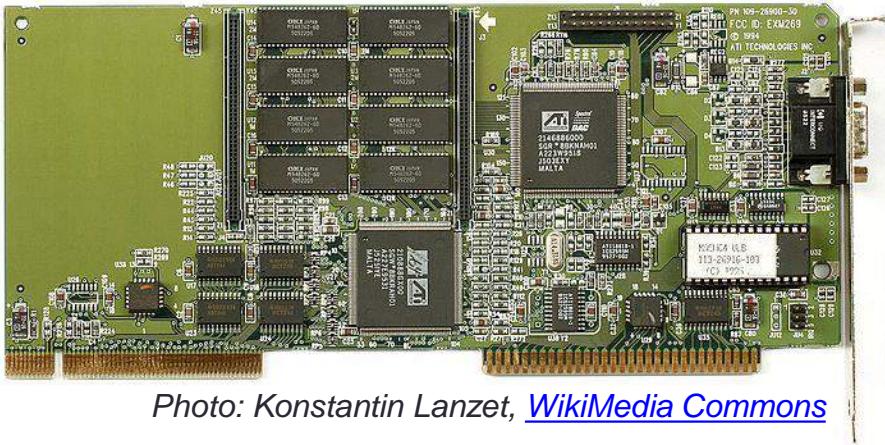
AGP sín

- **1990-es évek vége**
 - ISA sín idejétmúlt, megszűnik
 - Monitorok felbontása tovább nőtt: 1600x1200
 - 3D (játék)programok
- **Dedikált sín a grafikus kártya vezérlésére**
 - Gyorsított Grafikus Port (*Accelerated Graphics Port*)
 - AGP 1x: 264 MB/mp sávszélesség
 - Kisebb, mint a PCI, de csak a grafikus hardveré
 - ...
 - AGP 8x: 2,1 GB/mp



Néhány bővítőkártya

ATI MACH64 VLB grafikus kártya



Nvidia Gforce Quadro grafikus kártya (AGP)



8 bites ISA, 16 bites ISA és EISA csatlakozók

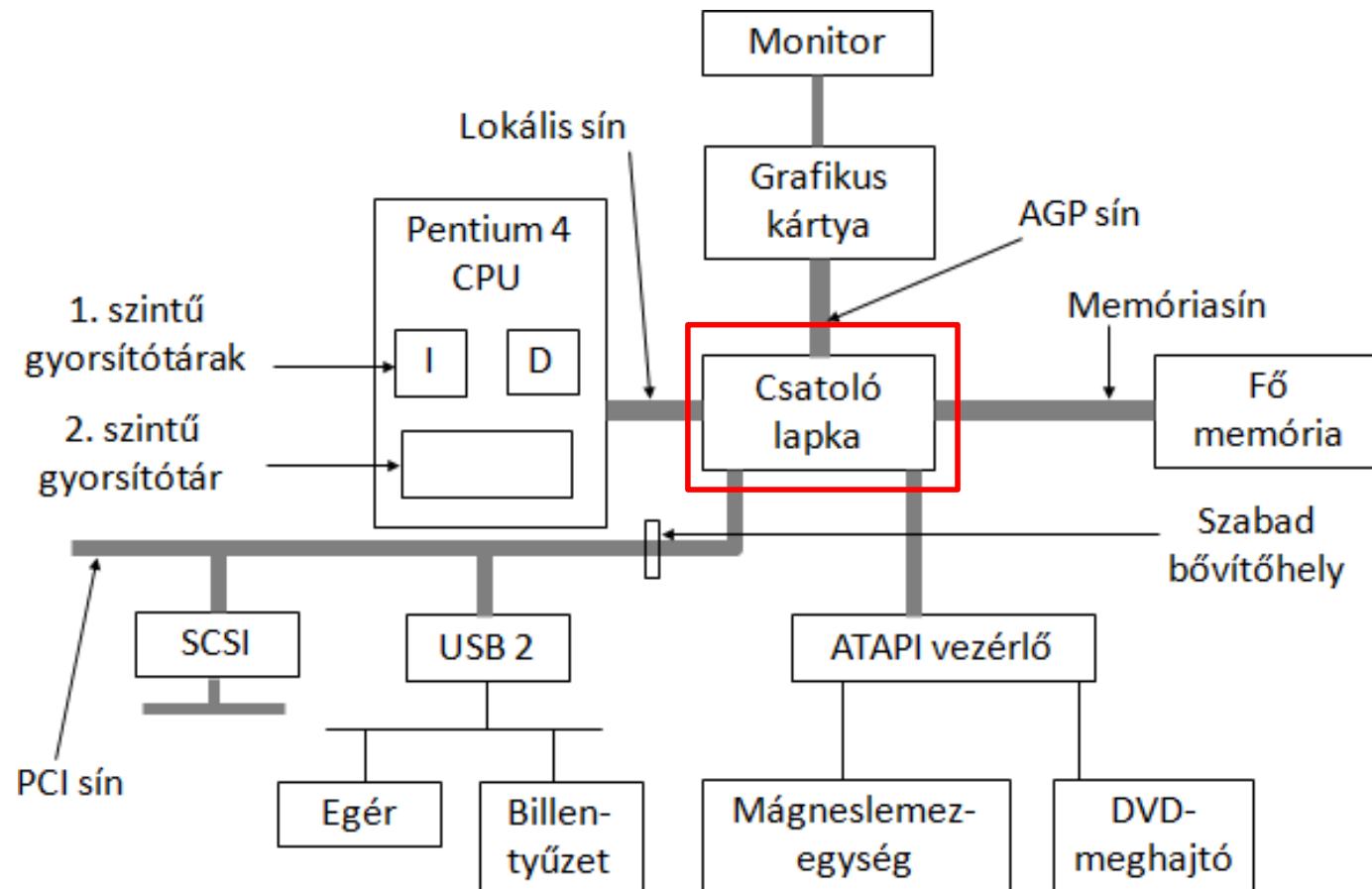


PCI csatlakozós SCSI adapter



Pentium 4 rendszer sín struktúra

- Hidak helyett csatoló lapka alkalmazása



PCI Express (PCIe)

- **PCI problémák**
 - Gyors eszközök kezeléséhez lassú
 - A gyors eszközök új síneket kapnak (pl. AGP)
 - Nem folytatható a végtelenségig
 - A vezérlést a csatoló lapka végzi
 - Nagy méretű bővítőkártyák
 - Laptopokba, okostelefonokba nem helyezhetők
 - Eszközök helyileg nem elkülöníthetők
- **Megoldás**
 - Többféle megoldási javaslat
 - PCI Express (Intel, 2004)
 - Párhuzamos sínek helyett nagysebességű közvetlen soros kapcsolatok
 - Szakít a korábbi sín-tradícióval (ISA/EISA/PCI)
 - A PCI megnevezést a „bejáratott” elnevezés miatt tartották csak meg

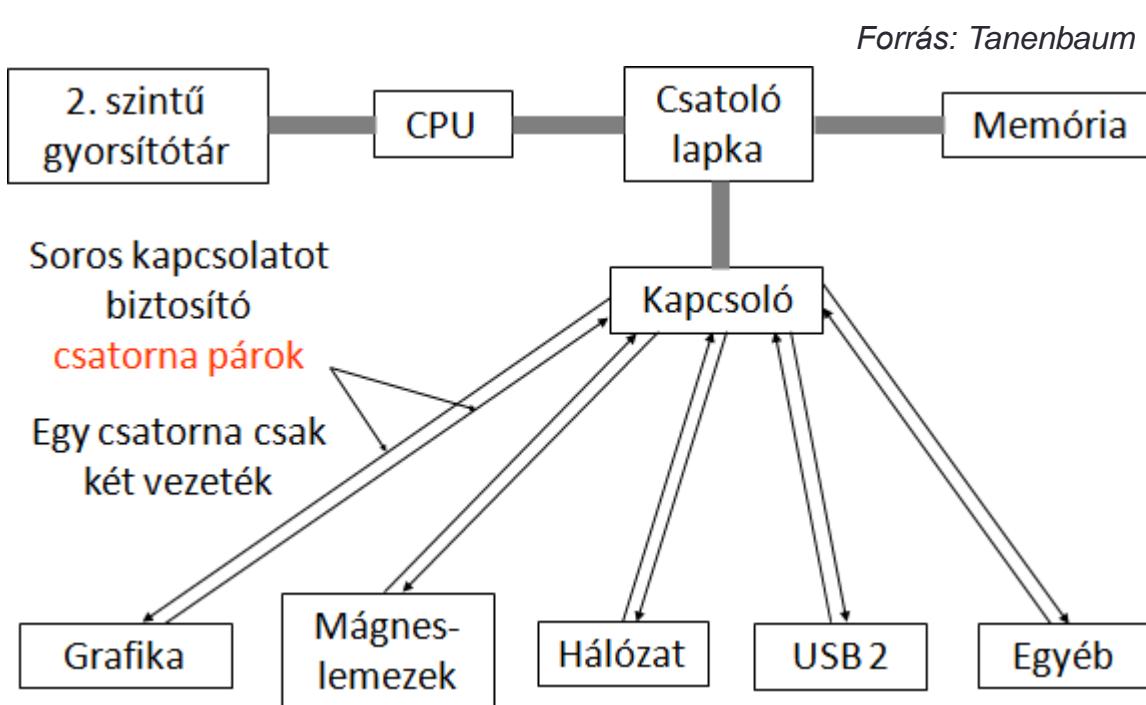
PCI Express (PCIe)

- **Tulajdonságok**

- A CPU, a memória és a gyorsítótár hagyományos módon csatlakozik a csatoló lapkához

- **Újdonság**

- Kapcsoló és a csatornapárok
- Adatcsomag küldése
 - hálózatok világából



PCI Express (PCIe)

Hagyományos sín	PCI Express
Több leágazású sín	Központosított kapcsoló
Széles, párhuzamos sín	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 nagyméretű bővítőkártya
- Eszközök csoportosíthatók (pl. merevlemez a monitorba)

Univerzális soros sín (USB, 1993-96)

- **ISA, PCI probléma**

- Az eszközök beszerelése körülményes
 - Számítógépet ki kell kapcsolni
 - Gépházat fel kell nyitni (garancia?)
 - Kártya kapcsolóit és vezetékáthidalásait („jumper”) manuálisan beállítani
 - Konfliktusok elkerülésére figyelve
 - Plug-and-Play kártyák ezt maguktól megoldják
 - Korlátosított számú kártya helyezhető be (pl. 3-8 darab)
- A sín feleslegesen gyors a lassú perifériák kezeléséhez
 - Billentyűzet, egér, ...

- **Igény**

- Bármikor könnyen, azonos módon lehessen sokféle perifériát kapcsolni a géphez, akár a gép működése közben, hardver ismeretek nélkül → **USB**

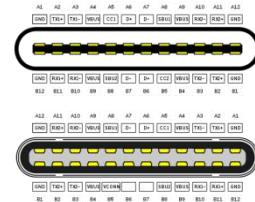
USB

- **Topológia**

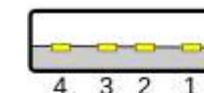
- Rendszersínre csatlakozó központi csomópont (root hub)
- Fa topológia
 - Központi csomóponthoz eszközök vagy újabb csomópontok csatlakoznak

- **Vezetékek (A és B típus esetén)**

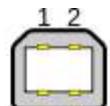
- 4 darab: 1 tápfeszültség, 2 adat, 1 föld
- Jeltovábbítás: 0 feszültségátmenet, 1 annak hiánya
- Kétféle csatlakozó a vezetékek végein
 - Ne legyen két hub összekapcsolható véletlenül
- Normál, mini, mikro csatlakozóméret
 - Mini és mikro esetén 5 vezeték
 - 1-3.: mint normál esetén
 - 4.: hoszt vagy szolga kapcsolat jelzésére
 - 5.: föld



Type C



Type A



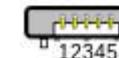
Type B



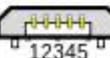
Mini-A



Mini-B



Micro-A



Micro-B

USB

- **Sebesség**

- USB 1.0 (1996): 1,5 MB/mp
- USB 1.1 (1998): 12 MB/mp
- USB 2.0 (2000): 35 MB/mp
- USB 3.0 (2008): 60 MB/mp

- **Új eszköz csatlakoztatásakor**

- *A központi csomópont*
 - Érzékeli az eseményt
 - Megszakítást kezdeményez
- *Operációs rendszer*
 - Lekérdezi: milyen eszköz, mekkora a sávszélesség igénye
 - Ha kielégíthető, 1-127 közötti azonosítót rendel hozzá
 - Azonosító és más paraméterek a B/K eszköz konfigurációs regisztereibe töltése



USB működési elve

- **Működési elve**
 - 1 ms-ként üzenetváltási keretet küld a központi csomópont
- **Keret típusok**
 - **Vezérlési keret**
 - Eszközök konfigurálása, parancsok küldése, állapot lekérdezése
 - **Izoszinkron keret**
 - Valós idejű eszközök (pl. hangátvitel)
 - Kieső adat (hiba) esetén nincs szükség annak pótlására
 - **Tömeges adat keret**
 - Nagy tömegű adat átvitele
 - Valós idejű átvitel nem szükséges (pl. nyomtató)
 - **Megszakítási keret**
 - Hagyományos értelemben vett megszakításokat nem támogatja az USB
 - Helyette pl. 50 ms-ként lekérdezi a billentyűzet állapotát

USB keret, csomagok

- **Keret felépítése**

- Egy vagy több csomagot tartalmaz (akár minden két irányú)

- **Csomag típusok**

- ***Token csomag***

- Rendszer irányítása (csomóponttól az eszközök felé)
 - SOF: Start of Frame (keretkezdet), minden ez az első (akár egyetlen)
 - IN: adatok kérése az eszköztől
 - OUT: adatok az eszköz számára
 - SETUP: eszköz konfigurálására

- ***Data csomag***

- 64 bájtnyi adat küldése (tetszőleges irányban)
 - SYN: szinkronizációs mező (8 bit)
 - PID: csomagtípus (8 bit)
 - PAYLOAD: hasznos adat
 - CRC: ellenőrző kód

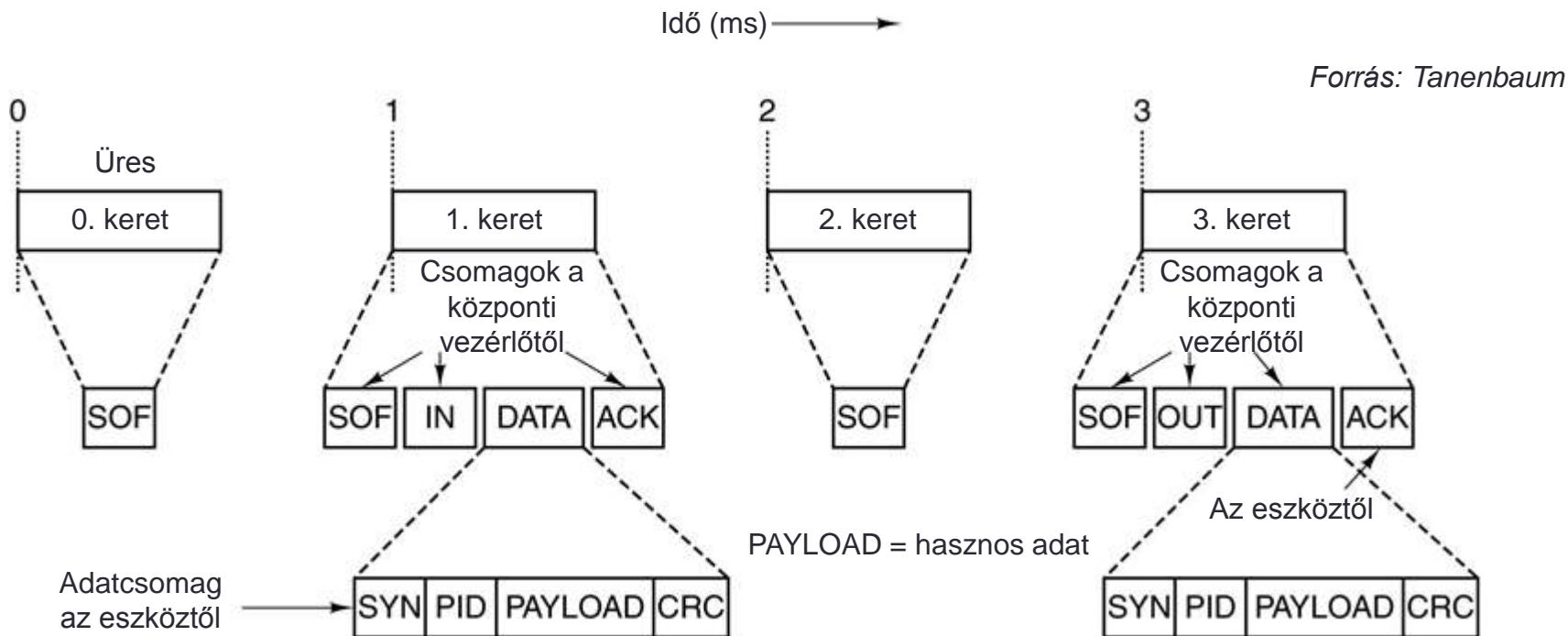
- ***Speciális csomag***

USB csomagok

- **Csomag típusok (folytatás)**

- **Kézfogás csomag**

- ACK: csomag rendben megérkezett
- NAK: CRC hiba az átvitel során → újraküldés (kivéve izoszinkront)
- STALL: foglalt jelzés, várjon



Kapcsolat a perifériákkal

- **Bemeneti/kimeneti lapkák (B/K lapkák)**
 - **UART** (Universal Asynchronous Receiver and Transmitter)
 - egy bájtot tud olvasni az adatsínről
 - sorosan továbbítja az eszközhöz (vagy fordítva)
 - programmal konfigurálható (belsı regiszterének beállításával)
 - 5-8 bit szélesség
 - 50 bps – 19200 bps sebesség
 - Paritás ellenőrzéssel (páros, páratlan, nincs)
 - **USART** (Universal Synchronous Asynchronous Receiver and Transmitter)
 - Szinkron és aszinkron működés is lehetséges
 - **PIO** (Parallel Input/Output)
 - Párhuzamos működés
 - 24 vonal
 - TTL kompatibilis eszközökhöz (billentyüzet, nyomtató, ...)

SZÁMÍTÓGÉPES ARCHITEKTÚRÁK

A mikroarchitektúra szintje I.

Nagy Antal

A mikroarchitektúra szintje

- Digitális logika feletti szint
- Feladata
 - A felette lévő ISA-szint (utasításrendszer-architektúra szintje) megvalósítása
- Függ
 - Az ISA szint megvalósításától
 - A számítógép ár- és teljesítmény-célkitűzéseitől
- Sok ISA olyan, egyszerű utasításokkal rendelkezik, amelyek egyetlen ciklusidő alatt végrehajthatók
 - RISC
- Összetettebb ISA-k egyetlen utasítás végrehajtásához több ciklust igényelhetnek
 - Pentium-4, CISC

A mikroarchitektúra szintje

- Egy utasítás végrehajtása
 - Operandusok helyének meghatározása a memóriában
 - Operandusok memóriából való kiolvasása
 - Eredmény eltárolása a memóriába
- Utasításokon belüli műveletek sorba állítása
 - Vezérlés másfajta megközelítése

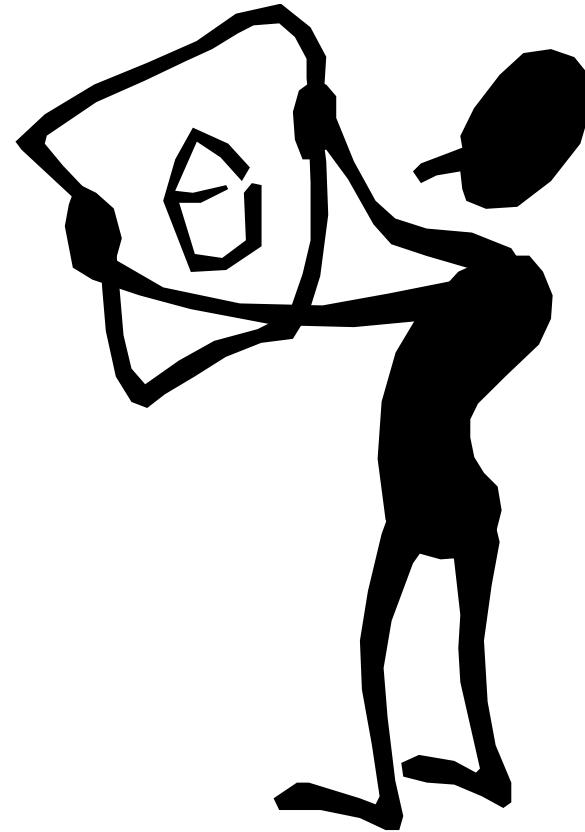


Mikroarchitektúra példa

- Nincs általános tervezési alapelv
- ISA-példa
 - Egész utasításokat tartalmazó Java virtuális gép (IJVM)
- Mikroprogramot fog tartalmazni a mikroarchitektúránk
 - ROM-ban tároljuk
 - IJVM utasítások
 - Betöltése
 - Dekódolása
 - Végrehajtása
 - Kis mikroprogramra van szükség
 - Hatékonyan vezérli az egyes kapukat az aktuális hardverben
 - Csak az előző fejezetben leírt alapelemekből áll

Mikroarchitektúra példa

- Tervezés, mint programozási probléma
 - ISA-szint minden egyes utasítása egy függvény, amit a főprogram hív meg
 - A főprogram egy végtelen ciklus
 - Meghatározza a meghívandó függvényt
 - Meghívja azt
 - Kezdi újra az egészet
- Mikroprogram
 - Változók egy halmaza
 - Számítógép állapota
 - minden függvény el tudja érni
 - Megváltoztatja
 - Pl. utasítás számlálót (PC)
 - A következő végrehajtandó függvényre mutat



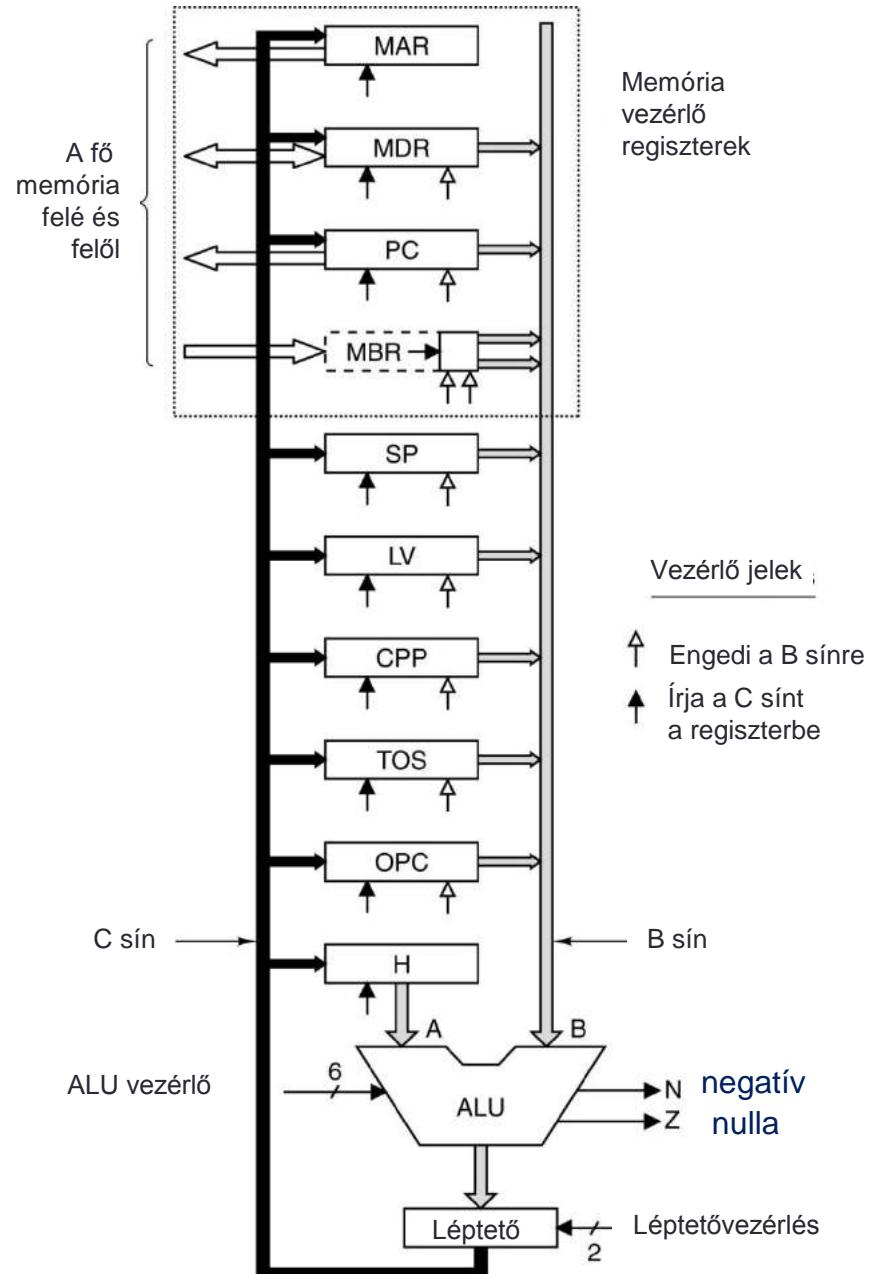
Mikroarchitektúra példa

- Az IJVM utasításai rövidek
 - Egy vagy két mező
 - Speciális szereppel
 - Első mező
 - Műveleti kód
 - Azonosítja az utasítást
 - További mezők
 - Operandusok
 - Lokális változók
- Betölt-végrehajt ciklus
 - Alapja az ISA-k megvalósításának



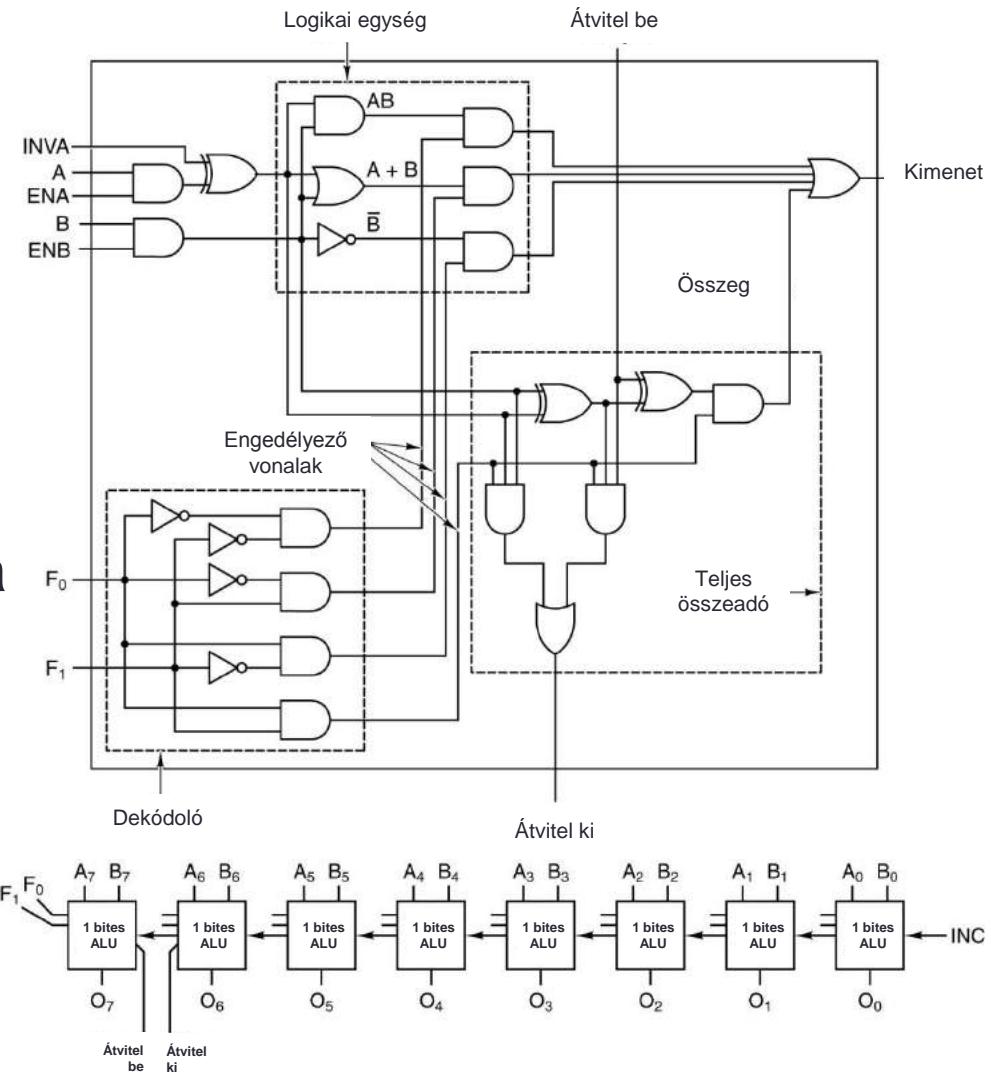
Adatút/Data Path

- Tartalmazza az ALU-t
 - Bemeneteivel és kimeneteivel együtt
- Sok 32 bites regisztert tartalmaz
 - PC, SP, MDR
 - Ezek a regiszterek **csak** a mikroarchitektúra szintjén érhetők el a mikroprogrammal
 - Egyesek az ISA-szintű architektúra ugyanilyen nevű regiszterével egyezik meg
 - A regiszterek tartalmát a B sínre lehet irányítani
 - ALU kimenete a léptetőbe vezet
 - Ezután a C sínre vezet
 - Értéke egy időben több regiszterbe írható



Adatút

- Az ALU 6-os címkéjű nyila a hat vezérlővonal
 - F0 és F1 meghatározza a műveletet
 - ENA és ENB engedélyezi a bemenetet
 - INVA invertálja az A bemenetet
 - INC a legalacsonyabb helyértékű biten az „átvitel be” bemenetet
- Az ALU-t vezérlő jelek nem mindegyike van kihasználva



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

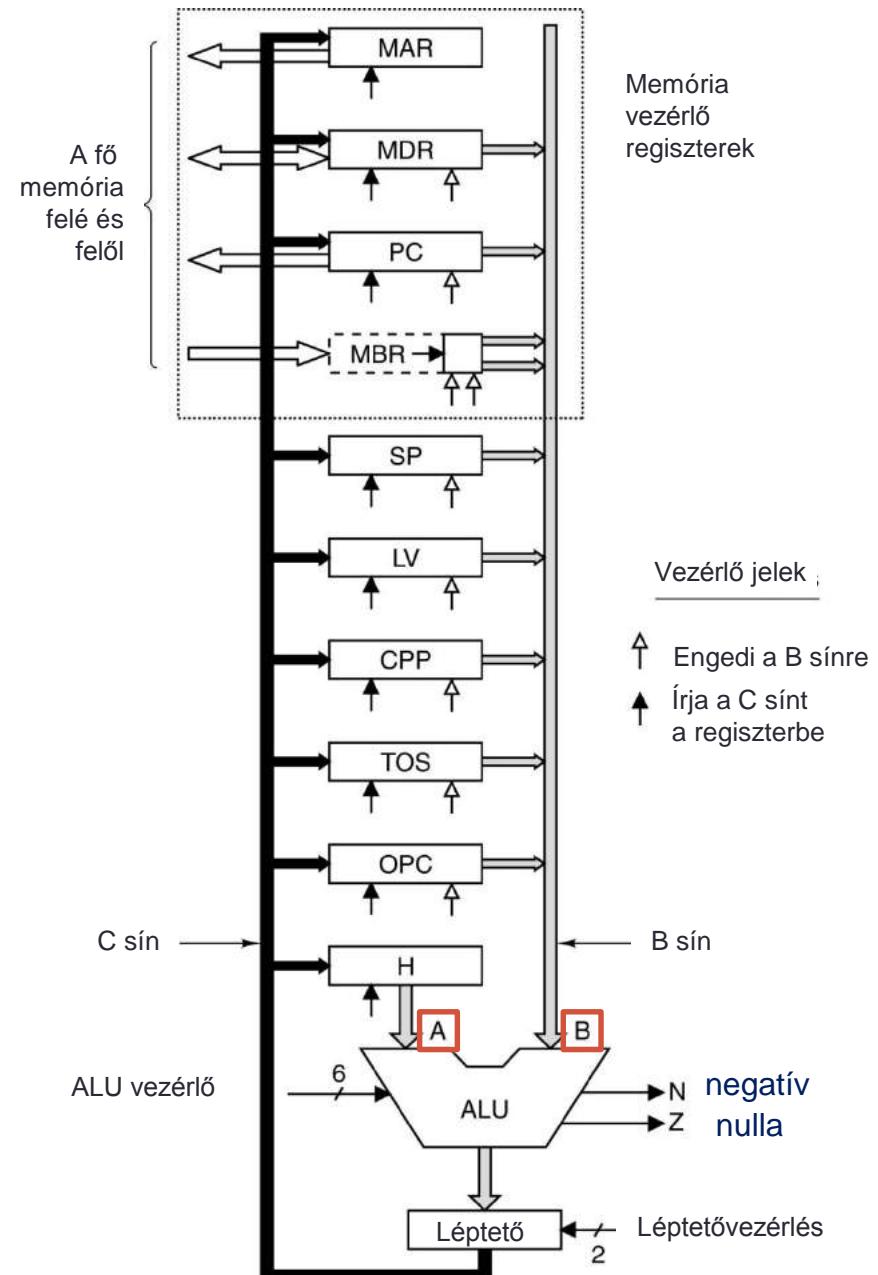
Adatút

- Néhány érdekesebb kombináció
 - Több lehetőség van ugyanannak az eredménynek az eléréséhez
 - $\neg A$ az A kettes komplementjét jelenti

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	\bar{A}
1	0	1	1	0	0	\bar{B}
1	1	1	1	0	0	$A + B$
1	1	1	1	0	1	$A + B + 1$
1	1	1	0	0	1	$A + 1$
1	1	0	1	0	1	$B + 1$
1	1	1	1	1	1	$B - A$
1	1	0	1	1	0	$B - 1$
1	1	1	0	1	1	$\neg A$
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	$\neg 1$

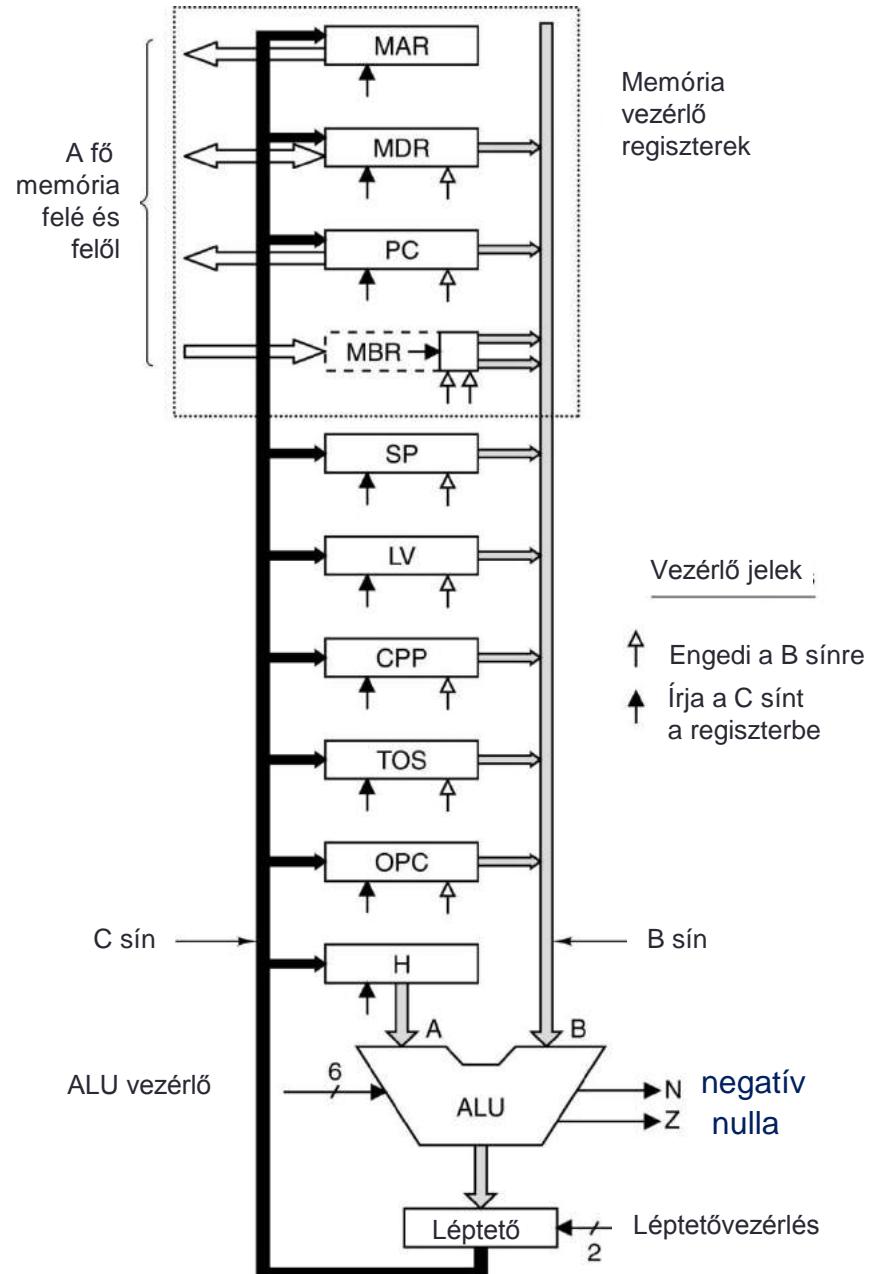
Adatút

- Az ALU-nak két adatbemenetre van szüksége
 - A és B
 - A bal oldali bemenet (A) a H (Holding) tartó regiszterhez kapcsolódik
 - A jobb oldali bemenet a pedig a B sínhez
 - Képes betölteni a kilenc forrás bármelyikét



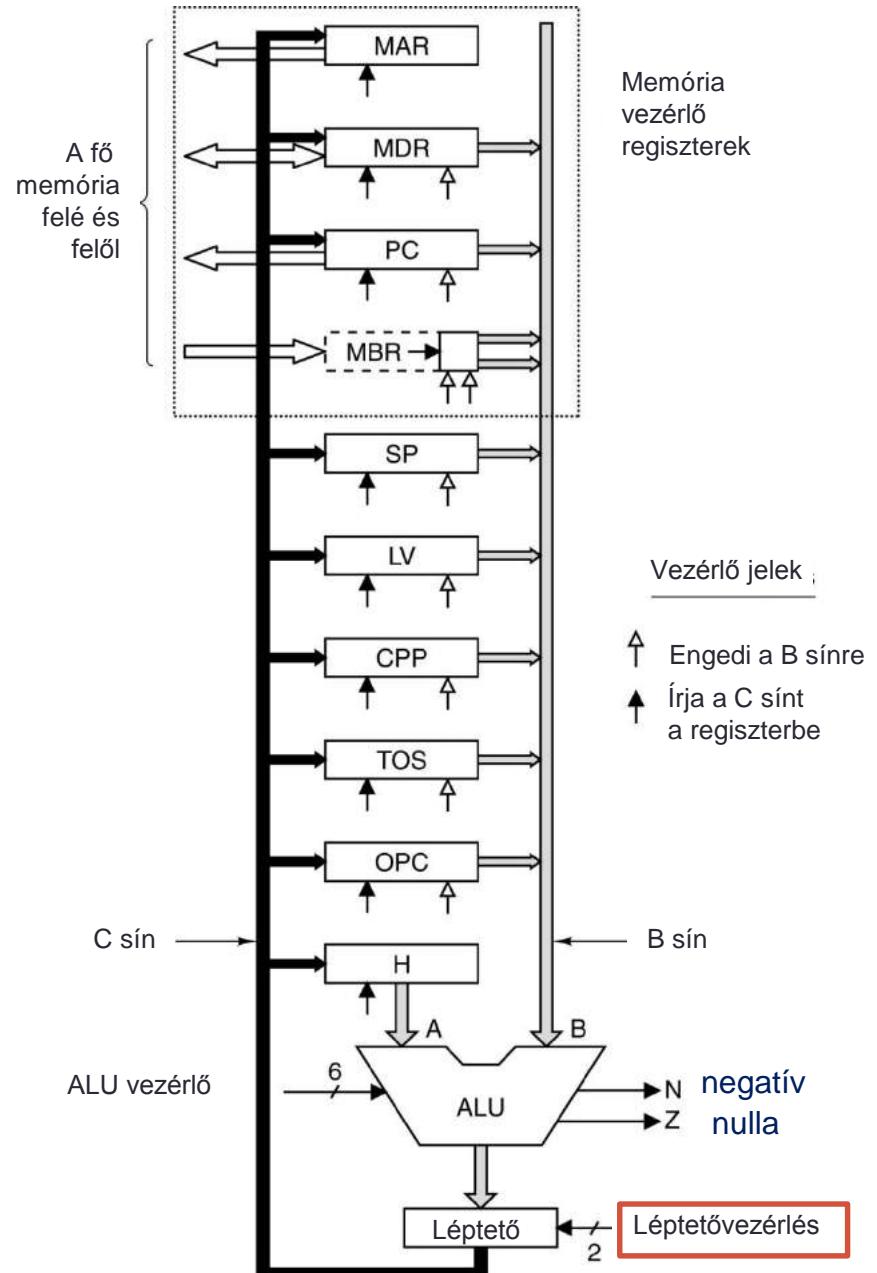
Adatút

- H feltöltése
 - Választunk egy ALU tevékenységet
 - A jobb oldali bemenetet a B sínről csak átengedi
 - ALU bemenetek összeadása negált ENA-val
 - A baloldali bemenet 0 lesz
 - Nullát adva a B sín értékéhez
 - A léptetőn módosítás nélkül tovább küldve H-ban tároljuk



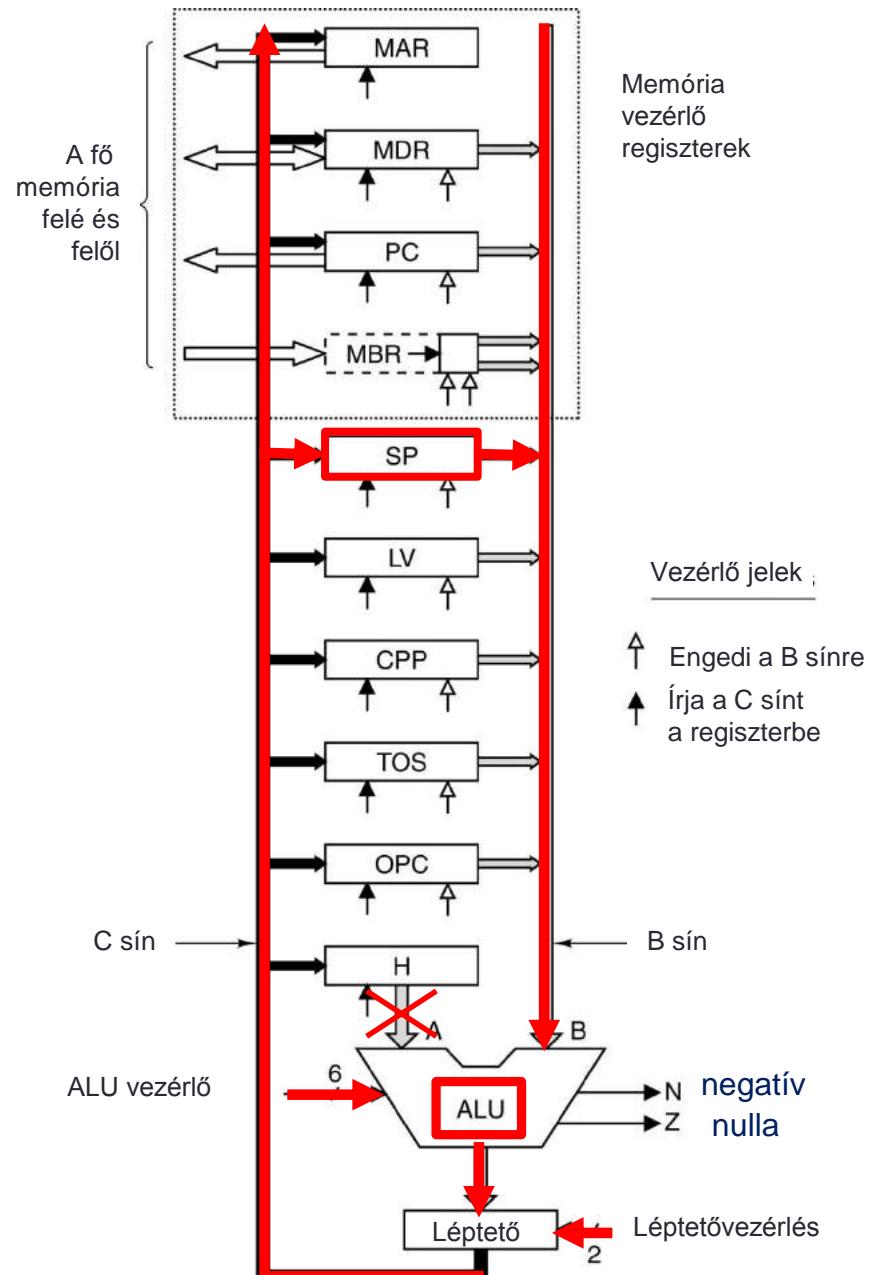
Adatút

- ALU kimenetének irányítása (léptető)
 - SLL8
 - Logikai balra léptetés 8 bittel
 - Nullával tölti fel az alacsony helyértékű biteket
 - SRA1
 - Aritmetikai jobbra léptetés 1 bittel
 - A legmagasabb helyértékű bitet nem változtatja meg



Adatút

- Ugyanazt a regisztert egy cikluson belül olvasni és írni is lehet
 - SP-t B sínre rakjuk
 - Kiválasztjuk SP-t
 - Tiltjuk az ALU bal oldali bemenetét
 - Engedélyezzük az INC-t
 - Az eredményt SP-be tároljuk
 - SP értékét eggyel növeltük



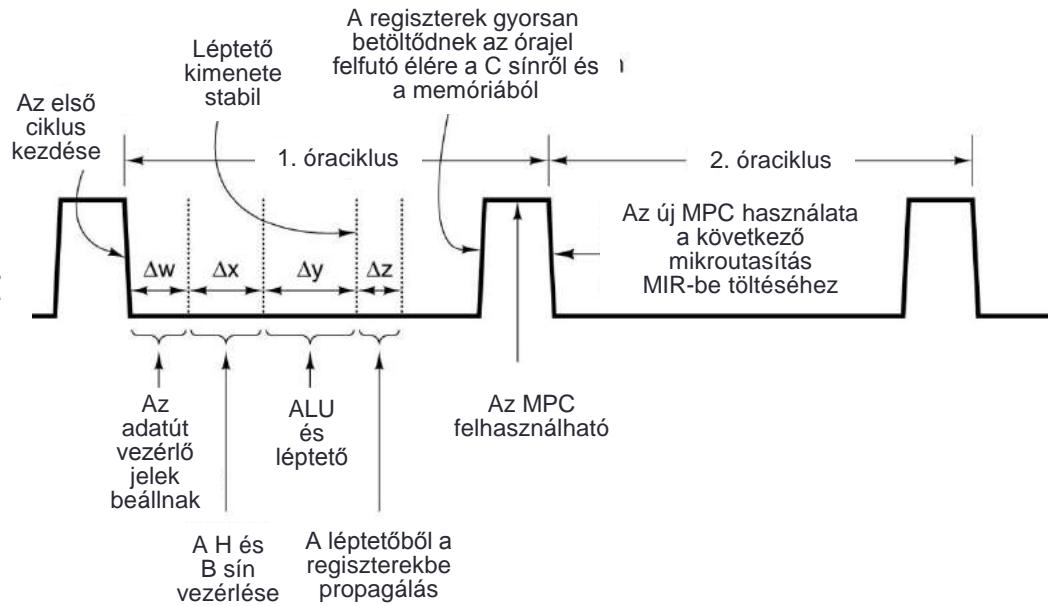
Adatút

- Az olvasás és írás egy cikluson belül különböző időpillanatban hajtódnak végre
 - Amikor kiválasztódik, hogy melyik legyen az ALU jobb oldali bemenete
 - B sínre kerül az értéke
 - Ott marad majdnem a ciklus végéig
 - Az ALU elvégzi a feladatát
 - Az eredmény a léptetőn keresztül a C sínre küldi
 - Amikor az ALU és a léptető kimenete stabil
 - Egy órajel hatására a C sín tartalma egy vagy több regiszterben eltárolódik
- Az adatút pontos időzítése lehetővé teszi egy regiszter egy cikluson belüli olvasását és írását

Adatút időzítése

- Minden óracikluskor egy rövid impulzus keletkezik
 - Származhat a fő órától
- A lefutó élen beállnak a kapukat irányító bitek
 - Δw idő alatt bekövetkezik
- Kiválasztódik a regiszter és a B sínrre kapcsolódik
 - Δx idő műlva lesz az értéke stabil
- Az ALU és a léptető végrehajtja a műveletet az érvényes adatokon
 - Δy idő műlva a kimenetek stabilak
- Δz idő után az eredmények tovább terjednek a C sín mentén a regiszterekhez
 - A felfutó élnél a regiszterek feltöltődnek
 - C sínről
 - Memoriából

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



MPC és MIR később

Adatút időzítése

- A betöltésnek élvezéreltnek és gyorsnak kell lennie
- Az impulzus felfutó élénél
 - A regiszter és a B sín kapcsolata megszakad
 - Felkészül a következő ciklusra
- Meghatározott terjedési időt figyelembe kell venni
- A tervezés megköveteli
 - Szigorú időzítést
 - Hosszú óraciklust
 - Az ALU-n keresztfüggő minimális terjedési idő ismeretét
 - C sínről a regiszterbe való gyors betöltést
- **Helyes működés!!!**
 - Tervezés

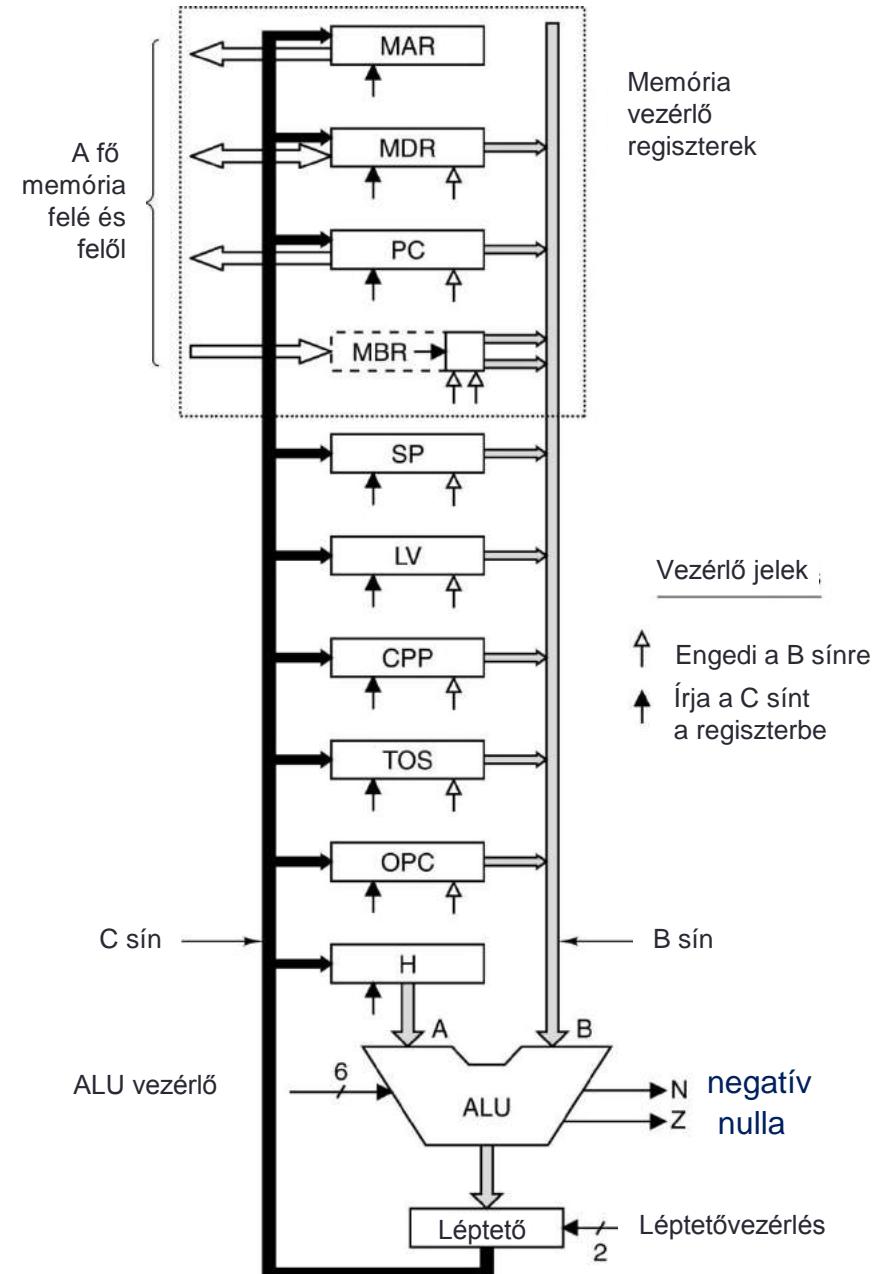


Adatút időzítése

- Adatút-ciklus másik vizsgálata
 - Rész ciklusokra való darabolás
 - A vezérlőjelek beállnak (Δw)
 - A regiszterek a B sínre töltődnek (Δx)
 - Az ALU és a léptető működik (Δy)
 - Az eredmények a C sín mentén visszakerülnek a regiszterekhez (Δz)
 - A következő óraciklus felfutó élénél az eredmények eltárolódnak a regiszterekben
- A legjobb az, ha a részciklusok értelemszerűek
 - Nincs óra- vagy egyéb meghatározott jel, ami megmondja, hogy mikor mi működjön
 - Az ALU és a léptető mindenkorban működik
 - A bemenetek bizonytalanok $\Delta w + \Delta x$ ideig
 - A kimenetek bizonytalanok $\Delta w + \Delta x + \Delta y$ ideig
 - Az egyedüli meghatározott jel az órajel lefutó és felfutó éle
 - Az adatút ciklusát indítja el illetve feltölti a regisztereket a C sínről

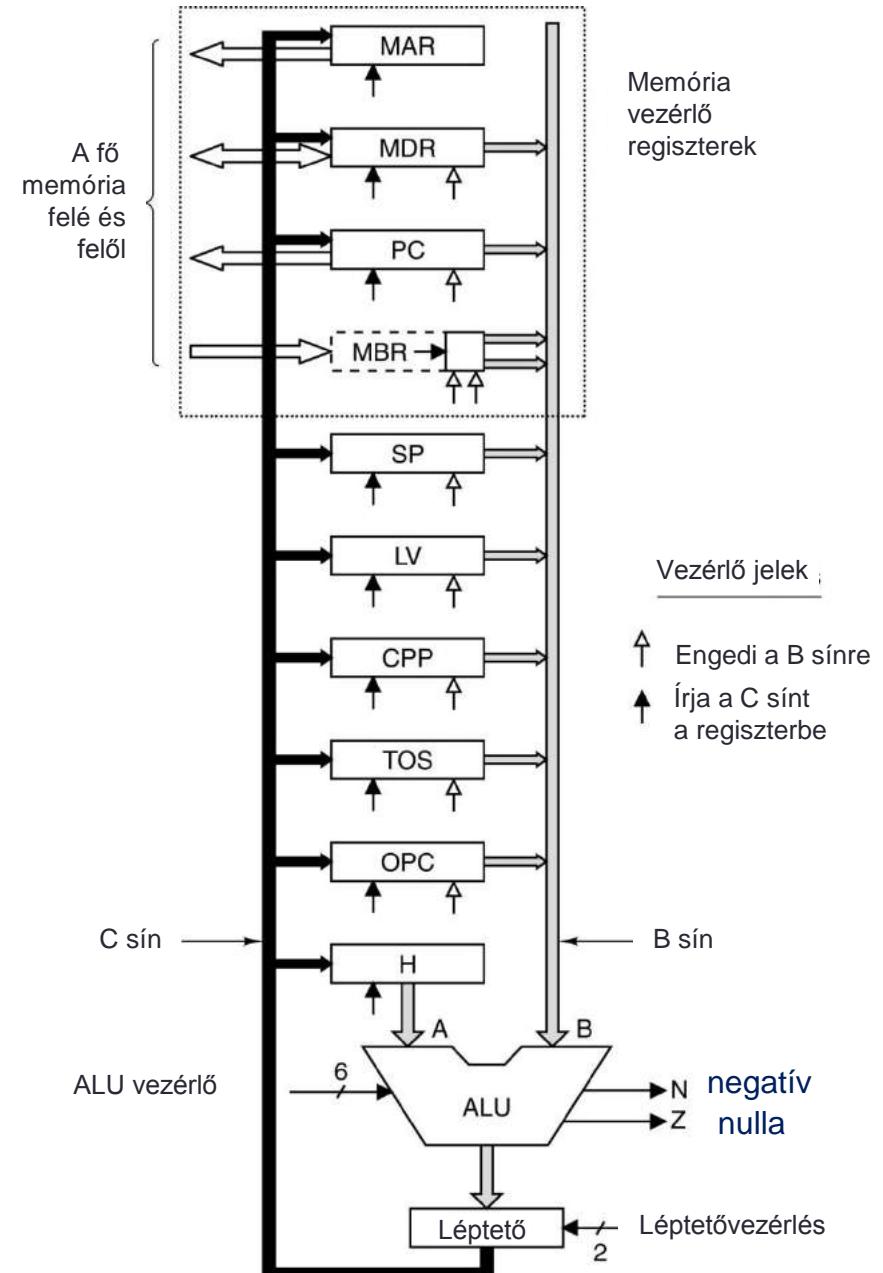
A memóriaművelet

- Memóriával való kommunikáció
 - 32 bites szócímzésű memóriaport
 - Memory Address Register (memóriacím-regiszter) vezérli
 - Memory Data Register (memóriaadat-regiszter)
 - Olvasás/írás
 - 8 bites bájt címzésű memóriaport
 - PC vezérli
 - Memory Buffer Register (memóriapuffer-regiszter)
 - Csak olvasni tud



A memóriaművelet

- A regisztereket egy vagy két vezérlőjel irányítja
 - Üres nyíl: a regiszter tartalmát engedi a B sínrre
 - Teli nyíl: C sínről írja a regisztert
- Memóriaolvasás vagy – írás
 - Megfelelő regiszterek feltöltése
 - Olvasó- vagy írójel küldése a memória felé
 - Nincs az ábrán feltüntetve

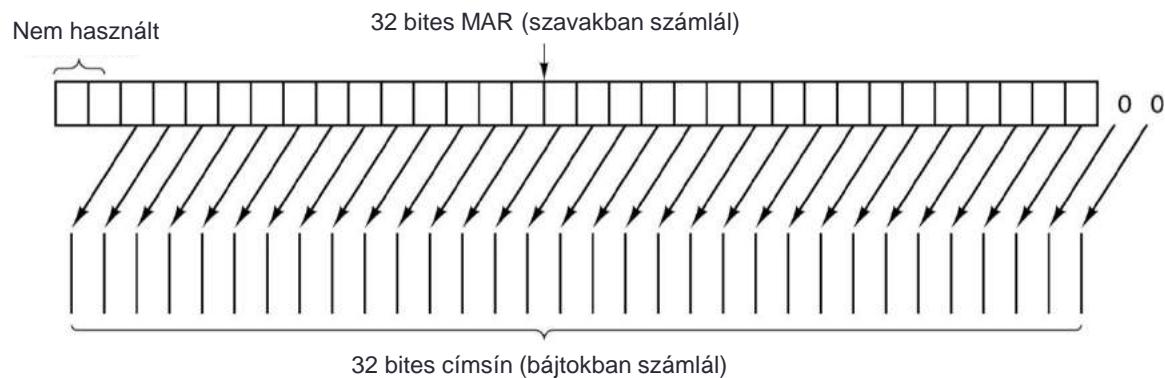


A memóriaművelet

- MAR
 - Szócímeket tartalmaz
 - Egymást követő szavakra hivatkozik
 - A kiolvasáskor az eredményt az MDR-be rakja
- PC
 - Bájtcímeket tartalmaz
 - Egymást követő bajtokra hivatkozik
 - MBR alsó 8 bitjére teszi
- A MAR \Leftrightarrow PC
 - A memória két különböző részére való hivatkozás
 - Program Counter
- MAR/MDR
 - ISA-szintű adatszavak olvasása és írása
- PC/MBR
 - ISA-szintű bajt folyamból álló végrehajtható program olvasása
 - A többi regiszter szócímeket használ

A memóriaművelet

- Csak egy fajta fizikai memória van
 - Bájt-szervezésű
- MAR szavakban (4 bájt) számlál
 - Amikor a MAR a címsínre kerül
 - Nem képződik le a 32 címvonalra közvetlenül
 - 0. bit – címsín 2. vonala
 - 1. bit – címsín 3. vonala
 - ...
 - A legfelső 2 bitet eldobjuk
 - 2^{32} feletti címek

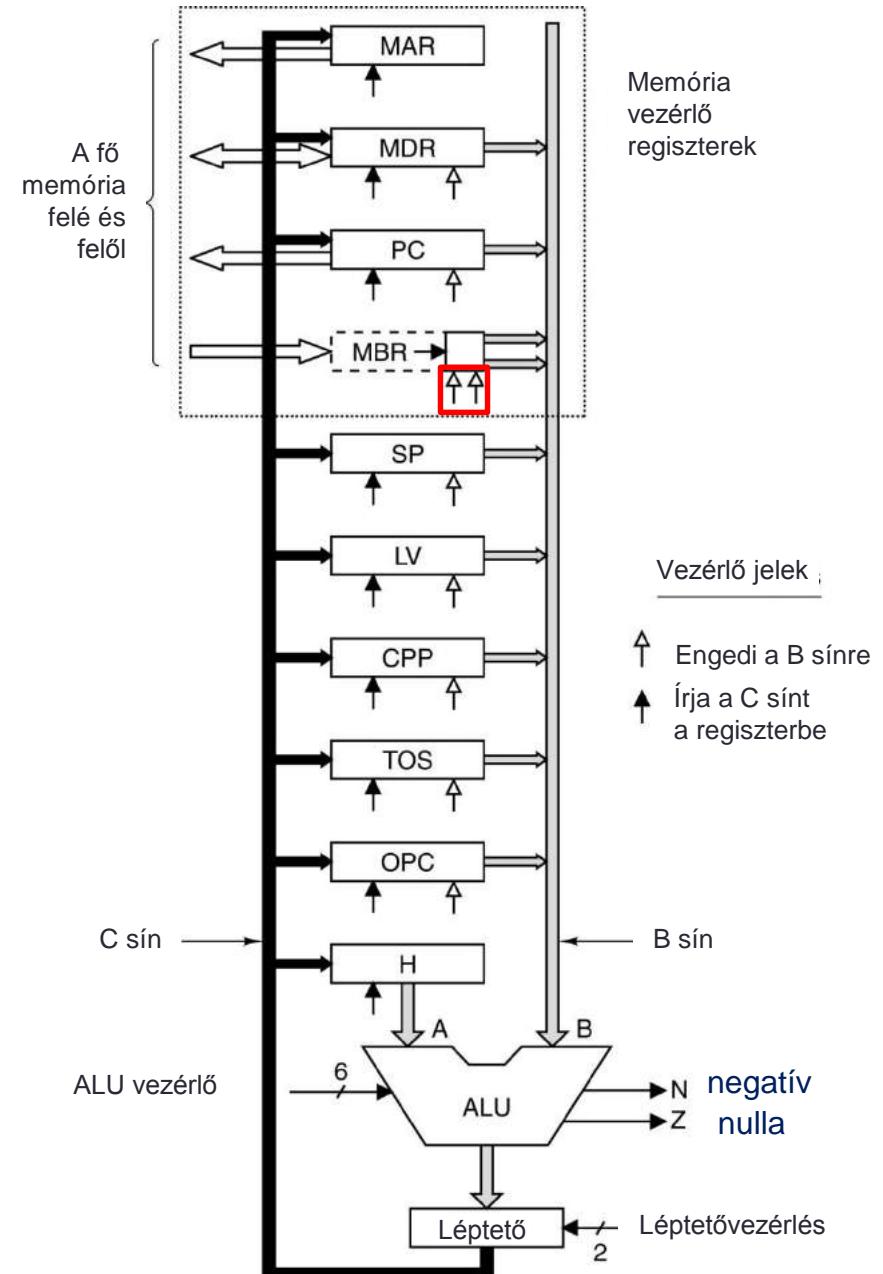


Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

MAR = 1, akkor a sínrre 4 kerül
 MAR = 2, akkor a sínrre 8 kerül

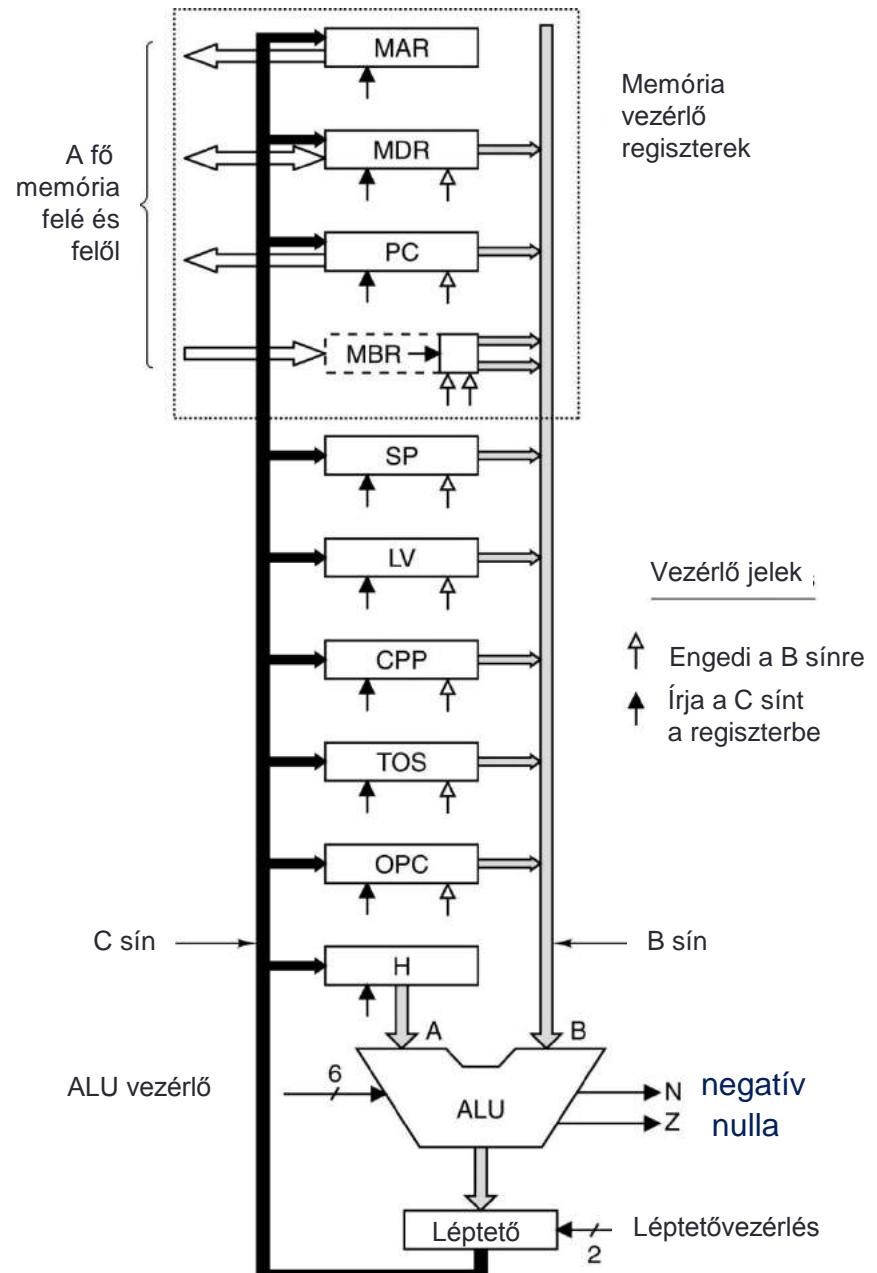
A memória művelet

- 8 bites memóriaporton keresztüli olvasás
 - MBR
 - Kapuzható (másolható) a B sínre
 - Előjel nélkül
 - Alsó 8 bit és többi 24 biten 0-k
 - Táblázatok indexelése
 - Előjelesen
 - 32 bites szóval való konvertálás
 - -128 és 127
 - Előjel bit másolása a felső 24 bitre
 - Előjel kiterjesztés
 - Két vezérlőjel



Mikroutasítások

- Az adatút vezérléséhez 29 jelre van szükség
 - 9 jel vezérli az adatírást a C sínről a regiszterekbe
 - 9 jel vezérli a regiszterek engedélyezését a B sínrre az ALU bemenet számára
 - 8 jel vezérli az ALU és a léptető tevékenységeit
 - 2 jel (nem látható) mutatja a memóriaolvasását/írást MAR/MDR-n keresztül
 - 1 jel (nem látható) mutatja a memóriabetöltést PC/MBR-n keresztül



Mikroutasítások

- A 29 vezérlőjel érteleke határozza meg az adatút egy ciklusának a műveleteit
 - Regiszterek értékének a B sínre való másolása
 - Csak egy regiszter
 - Jelek ALU-n és a léptetőn keresztsüli továbbítása
 - Az eredmény C sínre való vitéle
 - A C sínen lévő eredmény írása
 - megfelelő regiszterbe vagy regiszterekbe
- Ezután, ha a memóriaolvasás jel be van állítva
 - A memóriaművelet az adatútciklus végén kezdődik
 - MAR vagy PC feltöltődnek
 - Az adat a következő ciklus legvégén lesz MBR-ben vagy MDR-ben
 - Csak a rákövetkező ciklusban használható fel!!
 - Mindaddig, amíg az új értékek nem töltődnek be, addig a régi értékek használhatóak



Mikroutasítások

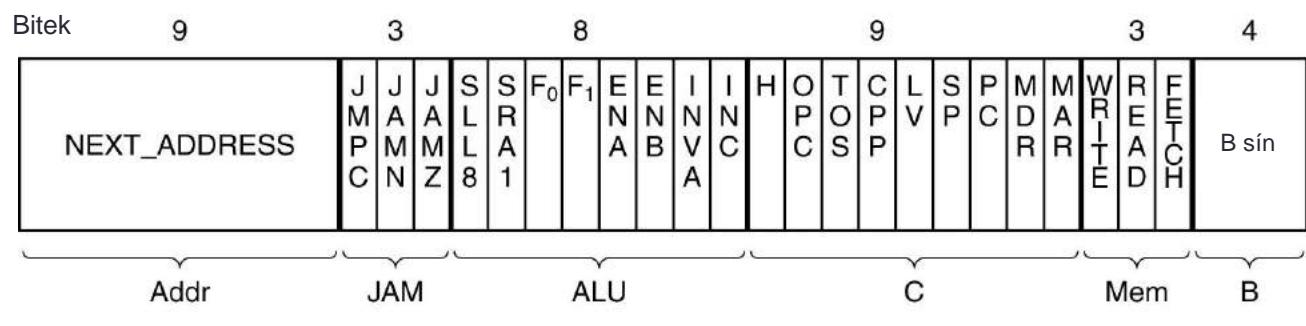
- Bitek számának a csökkentése
 - Lehetséges források kiválasztása és B sínre való küldése
 - Kilenc lehetséges regiszter van
 - 4 biten megoldható a B sín vezérlése
 - Dekóderrel előállíthatjuk a 16 vezérlőjelet
 - 7-et nem használunk ($16-9 = 7$)
 - Egy regiszter „felszabadításával” 3 bit elegendő lenne a feladat végrehajtásához
- 24 bittel vezéreljük az adatutat
 - Csak egy ciklusra vezéri
 - Vezérlés második része
 - Mit akarunk csinálni a következő ciklusban?

Mikroutasítások

- Formátum létrehozása a végrehajtandó műveletek leírására

- 24 vezérlő bit
- NEXT_ADDRESS
 - Lehetséges következő mikroutasítás címe
- JAM
 - A következő mikroutasítás hogyan választódik ki

- 36 jel – 36 bites mikroutasítás
 - Sorrend

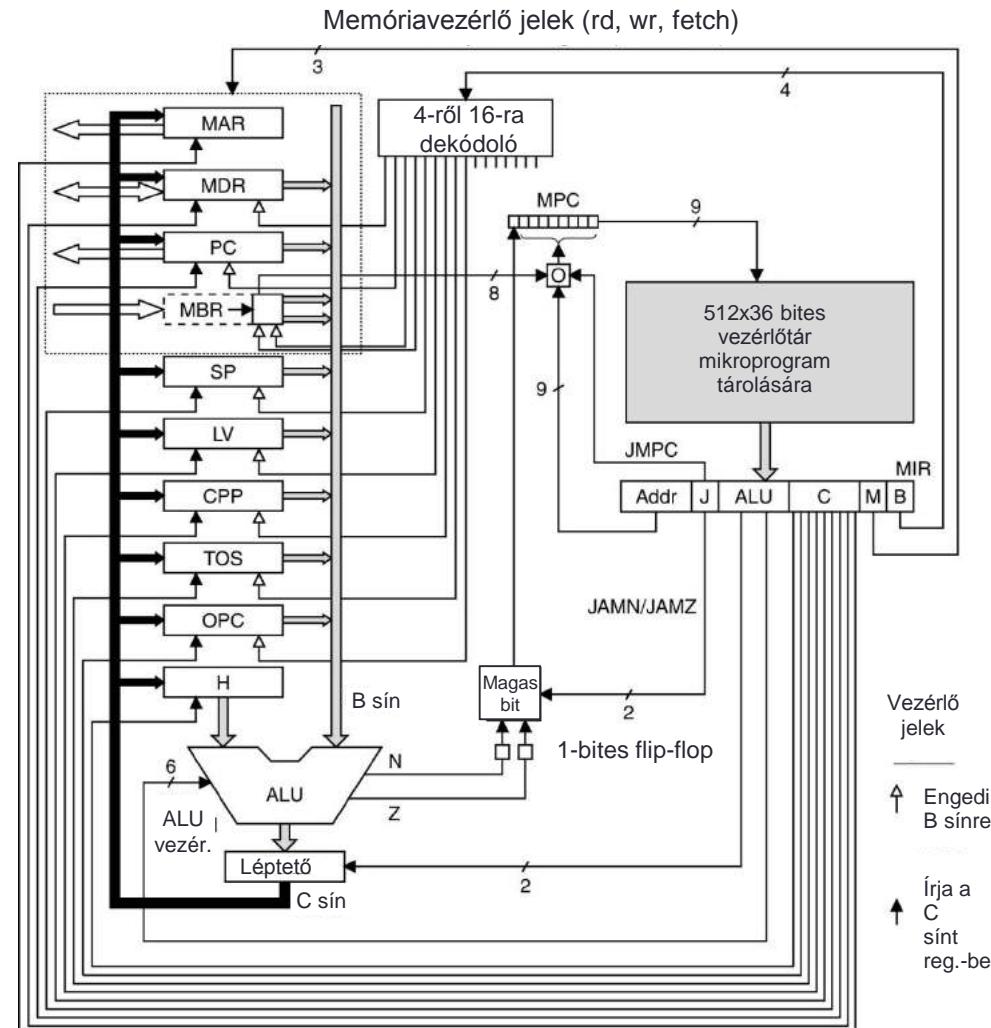


B sínsre küldhető regiszterek

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 Nincs

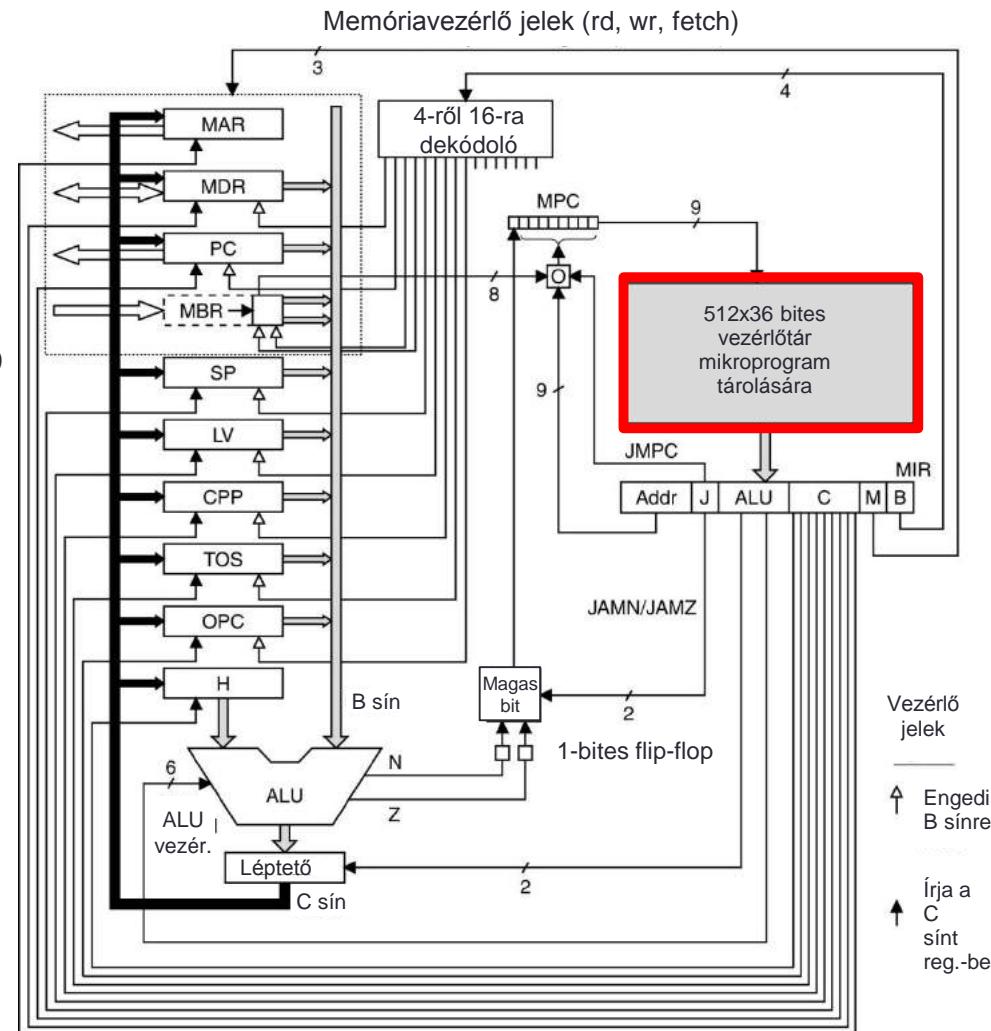
Mikroutasítás- vezérlés: Mic-1

- Vezérlőjelek engedélyezése egy cikluson belül
 - Sorba állító
 - ISA-utasítás végrehajtásához szükséges műveletek sorozatán való végigléptetésért felelős
 - Információ előállítása
 - Rendszerben lévő összes vezérlőjel értékét
 - Következő végrehajtandó mikroutasítás címét



Mikroutasítás- vezérlés: Mic-1

- Adatút és vezérlőrész
 - Vezérlőtár
 - Megvalósítása
 - Mikroprogramot tartalmazó memória
 - Nem főtár
 - Logikai kapuk halmaza
 - Memória, amely mikroutasításokat tartalmaz
 - 512 szót tartalmaz
 - minden szó 36 bites mikroutasításból áll

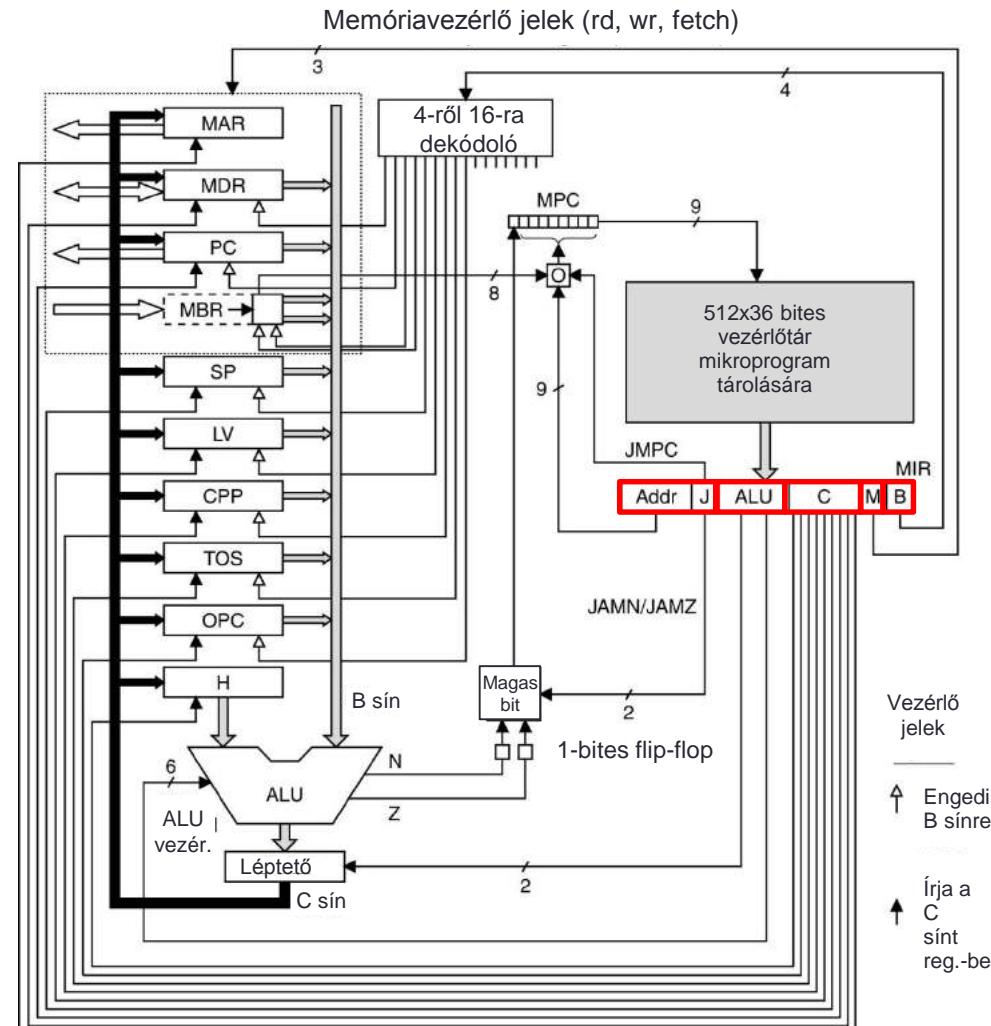


Mikroutasítás- vezérlés: Mic-1

- A főmemóriában lévő utasítások végrehajtása címsorrendben történik
 - Kivéve az elágazásokat
- A mikroutasítások végrehajtása nem címsorrendben történik
 - minden mikroutasítás előírja, hogy mi fogja követni
- A vezérlőtár csak olvasható memória
 - Nincs szüksége olvasó- és írójelekre
- Memóriacím-regiszter
 - MPC (MicroProgram Counter?)
 - Mikroutasítás számláló???
- Memóriadat-regiszter
 - MIR (MicroInstruction Register)
 - Tartalmazza az érvényes mikroutasítást
 - Bitjei meghajtják az adatutat működtető vezérlőjeleket

Mikroutasítás- vezérlés: Mic-1

- ADDR és J (JAM) csoport
 - Vezérli a következő mikroutasítás kiválasztását
 - ALU 8 bit
 - Kiválasztja az ALU tevékenységét és irányítják a léptetőt
 - C bitek
 - ALU és léptető kimeneti eredményének C sínről való betöltése regiszterekbe
 - M bitek
 - Memóriaműveletek vezérlése
 - B bitek
 - Dekóder
 - Meghatározza, hogy melyik regiszter kerüljön a B sínre



Mikroutasítás- vezérlés: Mic-1

Működés

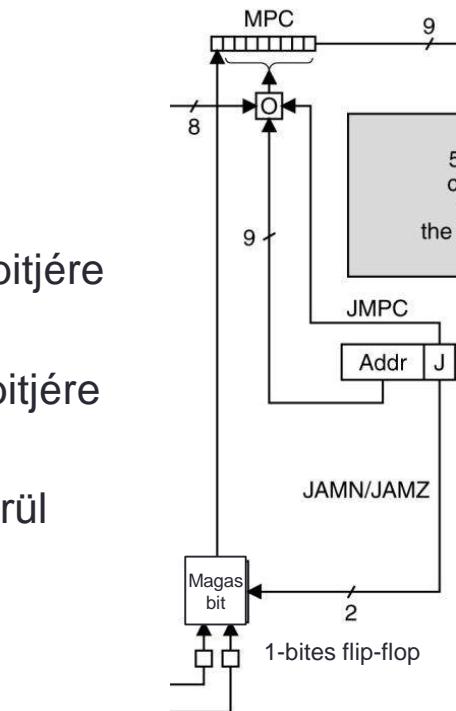
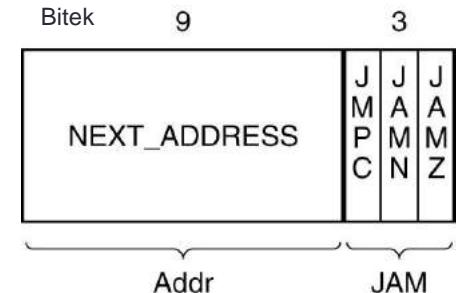
1. Az óraciklus kezdetén
 - A MIR feltöltődik az MPC által mutatott vezérlőtárbeli szóval
 - Δw idő alatt
2. A mikroutasítás MIR-be kerülése után
 - Különböző jelek kerülnek az adatútra
 - Egy regiszter kikerül a B sínre
 - Az ALU megtudja, hogy milyen műveletet kell végrehajtani
 - $\Delta w + \Delta x$ idő után az ALU bemenetei stabilak lesznek
3. Az ALU, N, Z és a léptető kimenete stabil
 - N és Z értéke két 1-bites flip-flopban tárolódnak el
 - Az ALU kimenete a léptetőbe töltődik
 - $\Delta w + \Delta x + \Delta y$
4. A léptető kimenete a C sínen keresztül eléri a regisztereket
 - A regiszterek és az N és Z flip-floppok betöltéséből áll
 - Az óra emelkedő éle után befejeződik
 - minden eredmény el van mentve
 - Előző memóriaművelet eredményei rendelkezésre állnak
 - MPC fel van töltve

Mikroutasítás- vezérlés: Mic-1

Működés

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

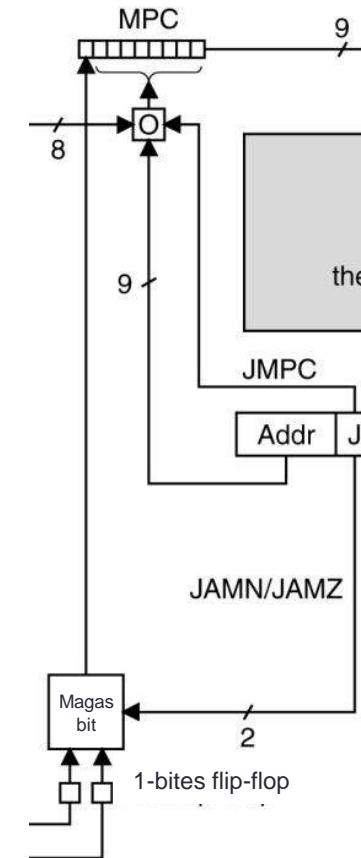
- Párhuzamosan meg kell határozni, hogy melyik mikroutasítást kell végrehajtani ezután
- A MIR feltöltődése és stabilizálódása után
 - A 9 bites NEXT_ADDRESS az MPC-be másolódik
 - Közben megvizsgálja a JAM értékét
 - 000 esetén nem kell csinálni semmit
 - JAMN = 1 esetén
 - Az N flip-flop OR kapcsolattal kerül az MPC legmagasabb bitjére
 - JAMZ = 1 esetén
 - Az Z flip-flop OR kapcsolattal kerül az MPC legmagasabb bitjére
 - JAMN = JAMZ = 1
 - Mindkettő OR kapcsolattal az MPC legmagasabb bitjére kerül
 - Az N és Z flip-floppra szükség van
 - Emelkedő órajelnél a B sín már nincs vezérelve
 - Az ALU kimenetei nem biztos, hogy helyesek



Mikroutasítás- vezérlés: Mic-1

Működés

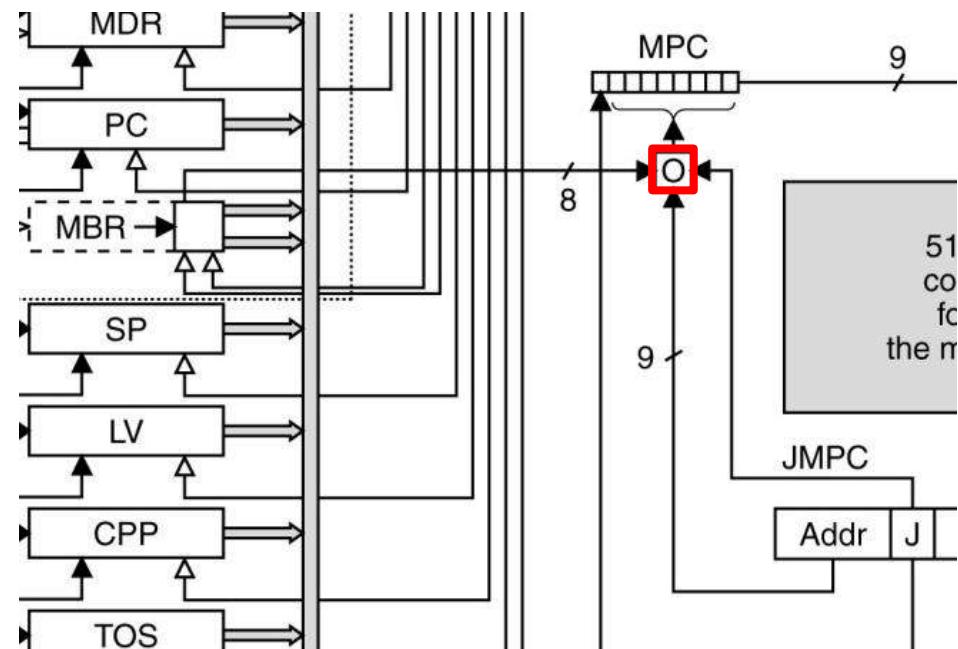
- A magas bit kiszámítása
 - $F = (JAMZ \text{ AND } Z) \text{ OR } (JAMN \text{ AND } N) \text{ OR } \text{NEXT_ADDRESS}[8]$
- Amikor JAMN vagy JAMZ értéke 1, akkor a lehetséges címek
 - NEXT_ADDRESS
 - NEXT_ADDRESS OR 0x100
 - Ha NEXT_ADDRESS <= 0xFF



Mikroutasítás- vezérlés: Mic-1

Működés

- **JMPC = 1 esetén**
 - MBR 8 bitjével bitenként OR műveletet hajt végre a NEXT_ADDRESS alsó 8 bitjével
 - Az eredmény az MPC-be kerül
 - A NEXT_ADDRESS értéke ilyenkor általában
 - 0x000 vagy 0x100
- **JMPC = 0 esetén**
 - Csak átereszi NEXT_ADDRESS értékét
- Több utas elágazás (ugrás) hatékony megvalósításánál fontos



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Mikroutasítás- vezérlés: Mic-1

Működés

- Az MBR egy műveleti kódot tartalmaz tipikusan
 - A JMPC használatával egyértelműen ki tudjuk választani a következő mikroutasítást
 - Hasznos, amikor egy betöltött műveleti kódtól függő tevékenységet egy másiknak kell követnie
 - Gyors és közvetlen ugrást lehet végrehajtani

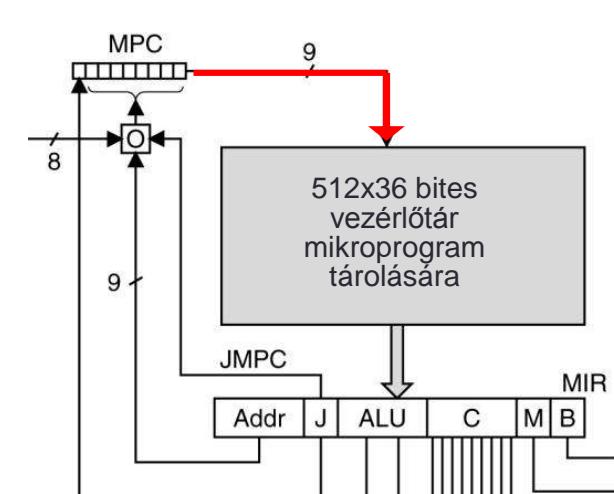
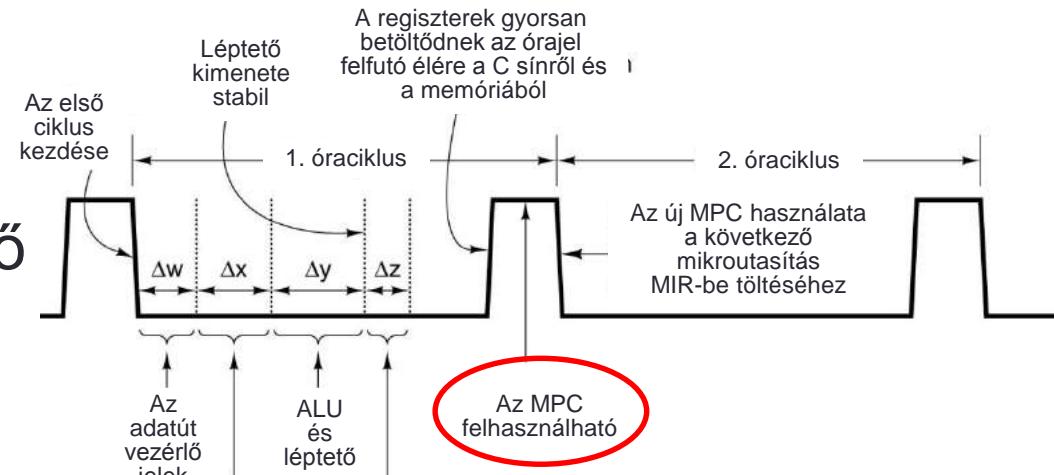


Mikroutasítás- vezérlés: Mic-1

Működés

- Az időzítés fontos
 - 31-es fólia
- Amikor az MBR elérhető
 - Az MPC betöltődik a következő mikroutasítás előkészítéséhez
 - MPC megkapja az értékét
 - Addig nem töltődik be, amíg azok a regiszterek nincsenek betöltve, amelyektől függ
 - MBR, N és Z
- Az órajel lefutásával az MPC megcímzi a vezérlőtárat
 - Új ciklus kezdődhet

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



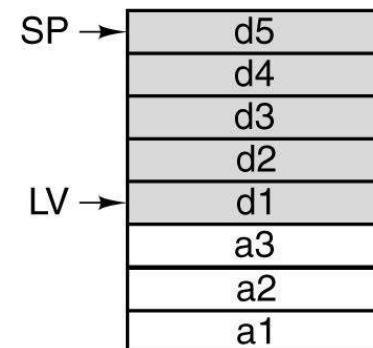
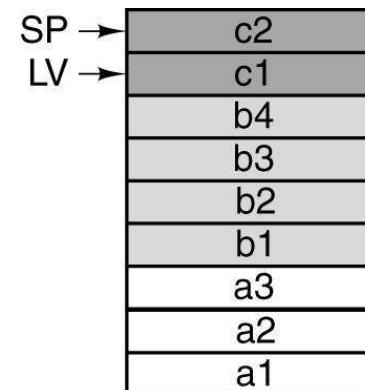
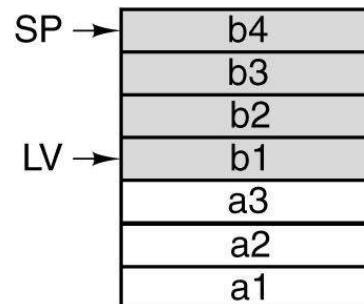
ISA-példa: az IJVM

- Vermek

- Lokális változók tárolásának a problémája
 - Eljárások, metódusok
 - Elérhetőek az adott érvényességi körön belül
 - Megszűnik az elérhetőségük, amikor pl. kilépünk az eljárásból
- Az eljárást önmagát is hívhatja (rekurzió)
 - Nem adhatunk abszolút memóriacímeket
- LV (Local Variable) regiszter a lokális változók kiindulási pontjára mutat
 - Hivatkozás az eltolás mértékének megadásával LV-től
- SP (Stack Pointer) adott eljárás lokális változói közül a legmagasabb szóra mutat
- Lokális változó mező
 - LV és SP valamint az azok között lévő adatstruktúra

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

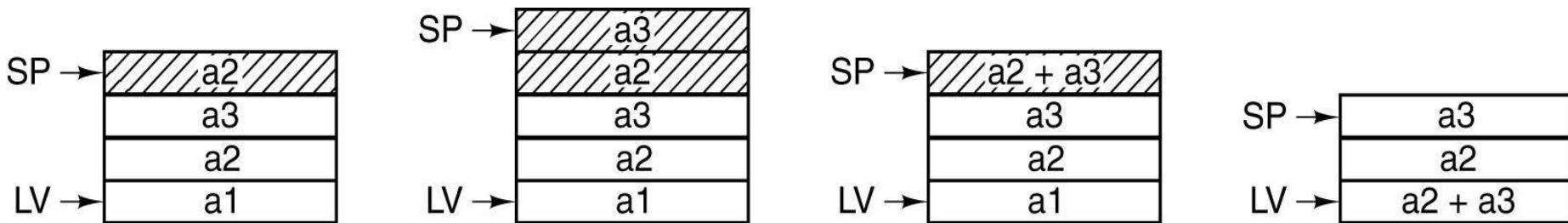
SP →	a3	108
	a2	104
LV →	a1	100



Vermek

- Használhatók egy aritmetikai művelet operandusainak a tárolására is
 - Operandus verem

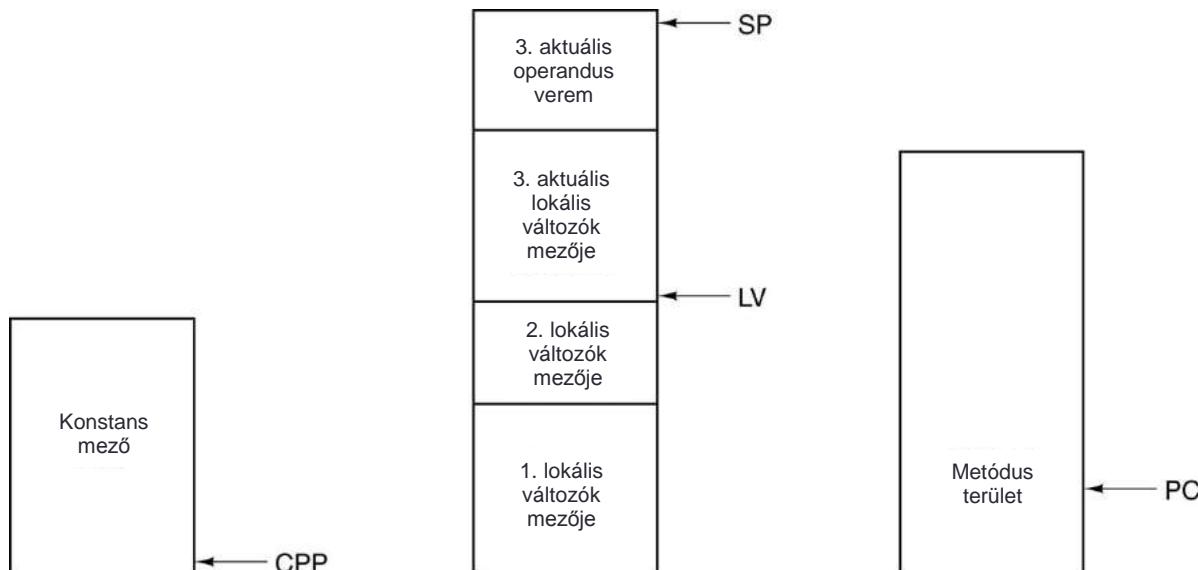
Pl. $a1 = a2 + a3$



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

Az IJVM memóriamodellje

- A memória két módon látható
 - 4 294 967 296 bájtos tömb (4 GB)
 - 1 073 741 824 szavas tömb
- Nincsenek közvetlenül látható abszolút memóriacímek
- Indexeléssel érheti el a memóriát



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

Az IJVM memóriamodellje

Konstans mező

- IJVM program nem írhatja
- Betöltés a program memóriába kerülésével
- CPP (Constant Pool Pointer)
 - Első szavának a címe

Lokális változók mezője

- Metódus hívásakor egy terület foglalódik le
- Elején az argumentumok
- Nincs benne az operandusverem
- LV
 - Első címhely tárolása

Az IJVM memóriamodellje

Operandusverem

- A verem mező nem lehet túl nagy
- Lokális változók mezője fölött van lefoglal
- SP
 - A verem legfelső szavának a címe
 - Változik

Metódus mező

- A programot tartalmazza
- „szöveg”
- PC (Program Counter)
 - Utasítás számláló
- Bájttömbként kezeljük

IJVM utasításkészlete (részlet)

Hex	Mnemonic	Jelentés
0x10	BIPUSH byte	Betesz a <i>byte</i> -ot a verembe
0xA7	GOTO offset	Feltétel nélküli ugrás <i>offset</i> -re
0x60	IADD	Kivesz a veremből két szót, az összegüket a verembe teszi
0x99	IFEQ offset	Kivesz a veremből egy szót, ha 0, akkor <i>offset</i> -re ugrik
0x9F	IF_ICMPEQ offset	Kivesz a veremből két szót, ha egyenlők, akkor <i>offset</i> -re ugrik
0x15	ILOAD varnum	Betesz <i>varnum</i> -ot a verembe
0x36	ISTORE varnum	Kivesz a veremből egy szót, és eltárolja <i>varnum</i> -ba
0x64	ISUB	Kivesz a veremből két szót, a különbségüket a verembe teszi
0x00	NOP	Nem csinál semmit
0x5F	SWAP	A verem két felső szavát megcseréli

Java fordítása IJVM-re

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

i = j + k;	1	ILOAD j	// i = j + k	0x15 0x02
if (i == 3)	2	ILOAD k		0x15 0x03
k = 0;	3	IADD		0x60
else	4	ISTORE i		0x36 0x01
j = j - 1;	5	ILOAD i	// if (i == 3)	0x15 0x01
	6	BIPUSH 3		0x10 0x03
	7	IF_ICMPEQ L1		0x9F 0x00 0x0D
	8	ILOAD j	// j = j - 1	0x15 0x02
	9	BIPUSH 1		0x10 0x01
	10	ISUB		0x64
	11	ISTORE j		0x36 0x02
	12	GOTO L2		0xA7 0x00 0x07
	13	L1: BIPUSH 0	// k = 0	0x10 0x00
	14	ISTORE k		0x36 0x03
	15	L2:		

Példa a megvalósításra

- Hogy néz ki az a program, ami a mikroarchitektúrán fut és értelmezi a makroarchitektúrát?
- Hogyan működik?
- Mikroutasítások és jelölésrendszer
 - A vezérlőtárat le lehetne írni binárisan
 - Szükség van egy jelölésrendszerre
 - Pl. címek kiválasztása
 - Egyetlen sor
 - Tevékenység meghatározása, ami egy óraciklusban történik
 - Ciklusról-ciklusra való vezérlés
 - Hack-elés
 - Sok dolog rejtte marad
 - Gyorsítás

Mikroutasítások és jelölésrendszer

- Művelet
 - SP értékét növelni akarjuk
 - egy olvasást kezdeményezünk
 - A következő utasítás a 122-es helyen lévő vezérlő tárbeli utasítás legyen
 - ReadRegister=SP, ALU=INC, WSP, Read, NEXT_ADDRESS=122
 - WSP: írunk az SP regiszterbe
 - Összhatás
 - $SP = SP + 1$; rd
- Micro Assembly Language (MAL)
 - minden ciklusban egy regiszter írható
 - MDR = SP
 - SP-t bemásolja MDR-be

Mikroutasítások és jelölésrendszer

- $MDR = H + SP$
 - $MDR = SP + H$
- Megengedett műveletek
 - Ki lehet egészíteni
 - $<<8$
 - Egy bájttal való balra léptetés
- SOURCE
 - MDR, PC, MBR, MBRU, SP, LV, CPP, TOS, OPC
 - B sínen
- DEST
 - MAR, MDR, PC, SP, LV, CPP, TOS, OPC, H
 - C sínen

DEST = H
DEST = SOURCE
DEST = \bar{H}
DEST = $\overline{\text{SOURCE}}$
DEST = $H + \text{SOURCE}$
DEST = $H + \text{SOURCE} + 1$
DEST = $H + 1$
DEST = $\text{SOURCE} + 1$
DEST = $\text{SOURCE} - H$
DEST = $\text{SOURCE} - 1$
DEST = $-H$
DEST = $H \text{ AND } \text{SOURCE}$
DEST = $H \text{ OR } \text{SOURCE}$
DEST = 0
DEST = 1
DEST = -1

Mikroutasítások és jelölésrendszer

- Egy ciklus alatt nem végrehajtható ezért nem megengedett
 - Pl. $MDR = SP + MDR$ vagy $H = H - MDR$
- Többszörös tárolás
 - $SP = MDR = SP + 1$
- Memória írás olvasás
 - rd, rw: MAR/MDR
 - fetch: PC/MBR
- Példa, hogyan ne csináljuk
 - $MAR = SP; rd$
 - Átad egy értéket a memóriából az MDR-be a második utasítás végén
 - $MDR = H$
 - Betesz egy értéket az MDR-be
 - Az eredmény definiálatlan lesz

Mikroutasítások és jelölésrendszer

- Címkére ugrás
 - Következő cím meghatározása
 - Egymás után írt sorok, egymás után lesznek végrehajtva
 - goto label
- Elágazás
 - Valami egyenlő-e 0-val?
 - TOS = TOS
 - Z bitje az ALU-nak
 - if (Z) goto L1; else goto L2
 - Z = TOS; if (Z) goto L1; else goto L2
- JMPC bit használata
 - goto (MBR OR value)

IJVM megvalósítása Mic-1 felhasználásával (részlet)

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

Címke	Műveletek	Megjegyzések
Main1	PC=PC+1;fetch; goto (MBR)	MBR-ben a műveleti kód; veszi a következő bájtot, elágazás
nop1	goto Main1	Semmit nem csinál
iadd1	MAR=SP=SP-1;rd	Beolvassa a verem teteje alatti szót
iadd2	H=TOS	H=verem teteje
iadd3	MDR=TOS=MDR+H; wr; goto Main1	Összeadja a két felső szót; beírja a verem tetejére
isub1	MAR=SP=SP-1;rd	Beolvassa a verem teteje alatti szót
...

IJVM megvalósítása Mic-1 felhasználásával

- Hogyan dolgozik az értelmező
 - MBR = 0x60

0x60	IADD	Kivesz a veremből két szót, az összegüket a verembe teszi
------	------	---

- Fő ciklus
 - PC = PC + 1; fetch; goto (MBR)
 - Növeljük a PC-t
 1. A műveleti kódot követő első bájt címét fogja tartalmazni
 2. Következő bájt betöltése MBR-be
 3. MBR-ben lévő címkére ugrunk
 - MAR = SP = SP – 1; rd
 - Verem teteje alatti szót beolvassa
 - H = TOS
 - H tartalmazz a verem tetejét
 - MDR = TOS = MDR + H; wr goto Main1
 - iadd1, iadd2, iadd3
 - MPC megkapja az előző utasítás NEXT_ADDRESS mezőjéből

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Mikroarchitektúra szintje II.
Gyorsítási lehetőségek

Tanács Attila

Mikroarchitektúra szint tervezése

- **MIC-1**

- Fiktív mikroarchitektúra
- „Közepesen” egyszerű, „közepesen” gyors
- Kb. 5000 tranzisztorból felépíthető lenne (memória nélkül)

- **Sebesség vagy ár**

- A részegységek gyorsabb vagy lassabb működésre tervezhetők
- Különböző költséggel

- **Gyorsítási lehetőségek**

- Csökkenteni egy utasítás végrehajtásához szükséges óraciklusok számát
 - Külön áramkör a PC regiszter növelésére (ne az ALU végezze)
- Egyszerűbb felépítés a rövidebb óraciklushoz
- Utasítások átlapololt végrehajtása
 - Ne legyenek üres ALU ciklusok (memóriabeolvasásra várva)
 - Utasításbetöltő áramkörök leválasztása a fő adatútról
 - ...

Végrehajtási út hosszának csökkentése

- **Értelmező ciklus és a mikrokód összefűzése**
 - Meddő ciklusok kihasználása
 - Példa a POP műveleten keresztül

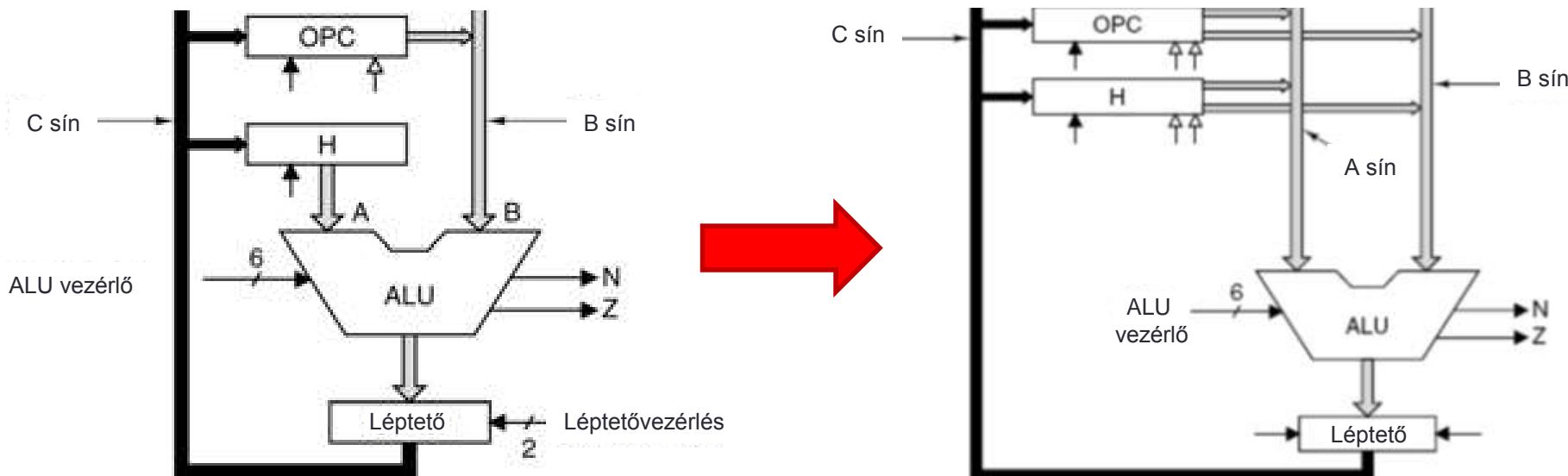
Címke	Műveletek	Megjegyzések
pop1	$\text{MAR} = \text{SP} = \text{SP} - 1; \text{rd}$	Verem teteje alatti szó olvasása
pop2		Várakozás a beolvasásra
pop3	$\text{TOS} = \text{MDR}; \text{goto Main1}$	Beolvasott szó a TOS-ba
Main1	$\text{PC} = \text{PC} + 1; \text{fetch}; \text{goto } (\text{MBR})$	Következő utasítás végrehajtására ugrik

Címke	Műveletek	Megjegyzések
pop1	$\text{MAR} = \text{SP} = \text{SP} - 1; \text{rd}$	Verem teteje alatti szó olvasása
Main1.pop	$\text{PC} = \text{PC} + 1; \text{fetch}$	Következő utasítás beolvasása, míg a memóriaolvasás zajlik
pop3	$\text{TOS} = \text{MDR}; \text{goto } (\text{MBR})$	Beolvasott szó a TOS-ba

Végrehajtási út hosszának csökkentése

- Többsínes architektúra

- Háromsínes példa
- Az ALU A regisztere közvetlenül kaphat értéket, nem csak H-ból
- Tetszőleges regiszterek összeadhatók egy cikluson belül
- Nem kell külön adatútciklus a H-ba töltéshez



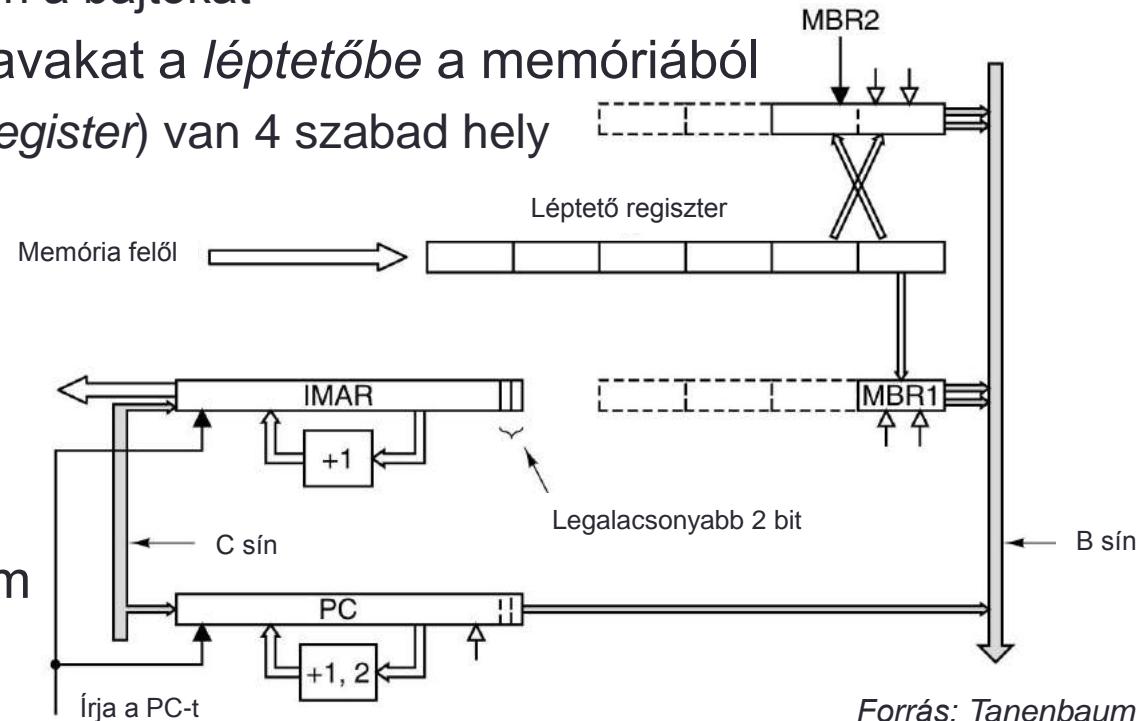
Végrehajtási út hosszának csökkentése

- **Utasításbetöltő egység**
 - IFU – Instruction Fetch Unit
 - Tehermentesítsük az ALU-t
- **IFU feladatai**
 - PC növelése
 - Teljes összeadó helyett elég egy növelő áramkör
 - Bájtok betöltése a memóriából, mielőtt felhasználásra kerülnek
 - 8 és 16 bites operandusok előkészítése (összerakása)
- **Működési módok**
 - Mindig legyen elérhető a következő 8 és 16 bites adat (műveleti kód értelmezés nélkül)
 - Műveleti kódok értelmezése, a megfelelő számú bájt beolvasása, regiszterben tárolása

Utasításbetöltő egység (IFU)

- **Újdonságok**

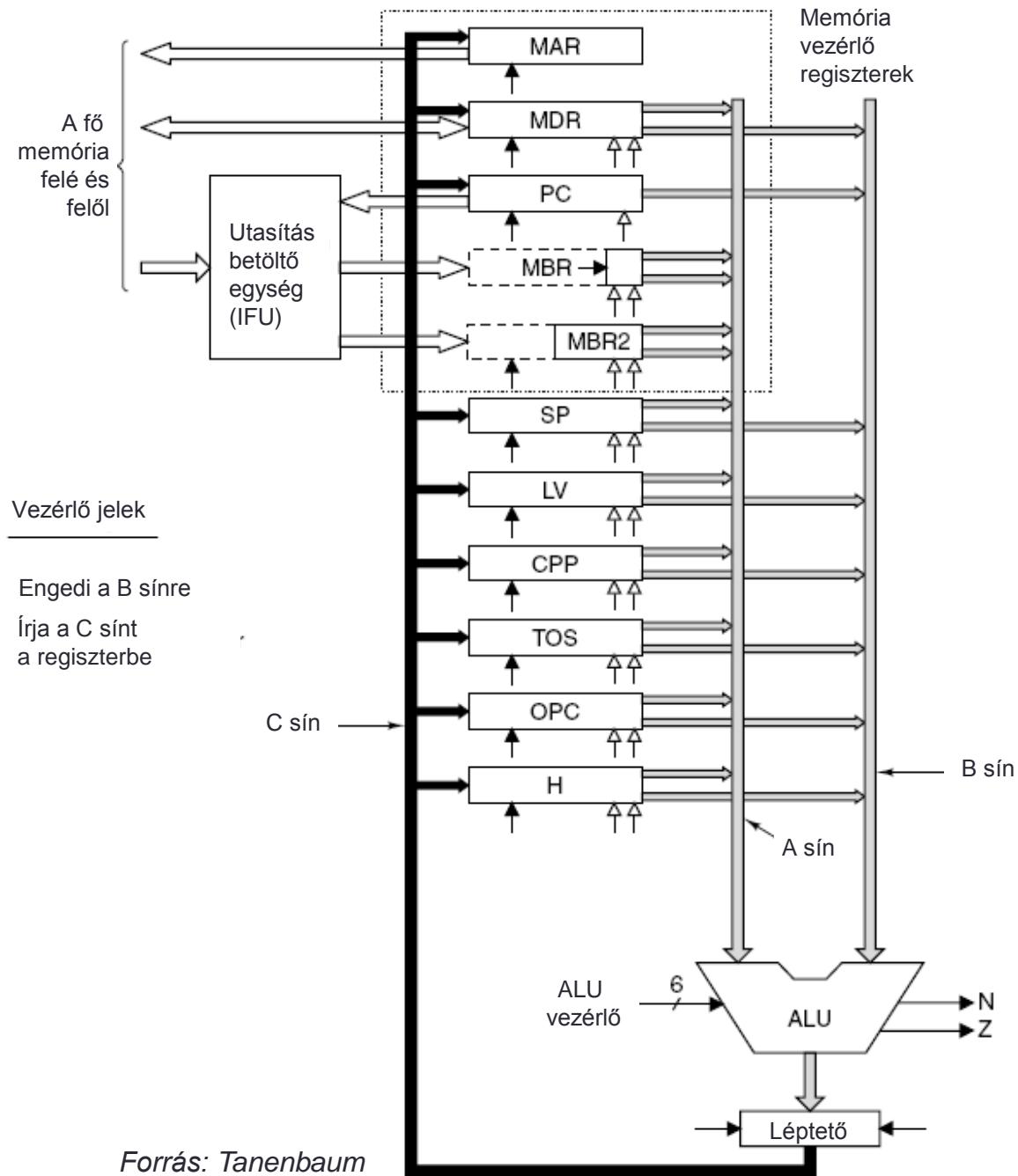
- MBR helyett MBR1 (8 bites) és MBR2 (16 bites)
 - Soron következő bájt(ok), közvetlenül elérhető minden kettő
- MBR1 és MBR2 olvasását érzékeli
 - Jobbra tolja a léptetőben a bájtokat
- Előre betölt 4 bájtos szavakat a léptetőbe a memóriából
 - Ha a léptetőben (*shift register*) van 4 szabad hely
 - Blokkolás
 - Ha MBR2 kellene, de még olvas a memóriából
- PC módosulásakor frissítés
- IMAR: saját memóriacím regiszter



Forrás: Tanenbaum

MIC-2

- Az eddig tárgyalt gyorsítások alkalmazása
- Csökken az utasítások végrehajtásához szükséges mikroutasítás-szám
 - Utasításonként különböző mértékben
- Gyorsabb gép, valamivel magasabb áron



Csővonalas megvalósítás (MIC-3)

- **Adatút felosztása**

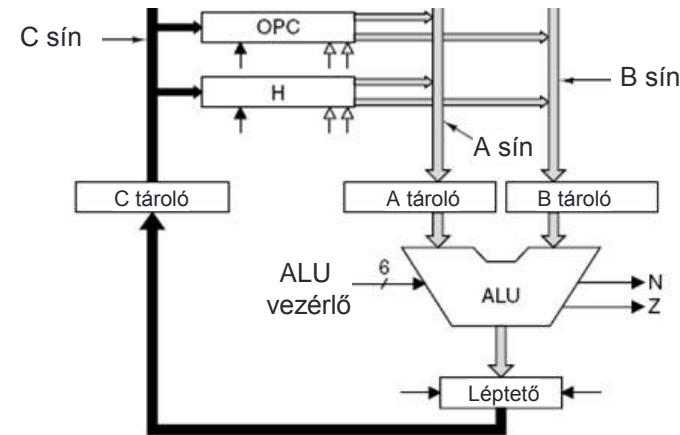
- Párhuzamosan működni tudó részekre
- Sínenként 1-1 tároló közbeiktatásával

- **Adatútciklus időigénye korábban**

- Regiszterek A és B sínre
- ALU és a léptető regiszter munkája
- Eredmények regiszterekbe írása

- **Felosztás után**

- Gyorsíthatjuk az órát, mert a maximális késleltetés kisebb
- minden ciklus alatt az adatút minden része dolgozhat
 - **Mikrolépések:** A és B betöltése; művelet elvégzése (ALU); C feltöltése és regiszterekbe írás
 - **Függőségek kialakulhatnak!**



Forrás: Tanenbaum

Csővonalas megvalósítás (MIC-3)

- **SWAP utasítás példa**

- Verem tetején lévő két érték megcserélése
- MIC-2 architektúrán **6 ciklus**

Címke	Műveletek	Megjegyzések
Swap1	$\text{MAR} = \text{SP} - 1; \text{rd}$	Verem teteje alatti szó olvasása MDR-be
Swap2	$\text{MAR} = \text{SP}$	Cím a verem tetejére áll
Swap3	$\text{H} = \text{MDR}; \text{wr}$	MDR mentése, szó írása a verem tetejére
Swap4	$\text{MDR} = \text{TOS}$	Előző TOS az MDR-be kerül
Swap5	$\text{MAR} = \text{SP} - 1; \text{wr}$	Előző TOS a verem 2. helyére
Swap6	$\text{TOS} = \text{H}; \text{goto (MBR1)}$	TOS frissítése az új értékre; ugrás

Csővonalas megvalósítás (MIC-3)

- **SWAP utasítás mikrolépései MIC-3-on**

- 11 ciklus, de 1 ciklus sokkal rövidebb, mint MIC-2-n
- Párhuzamos működés
- Mikrolépés sorozat:

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3	MAR = C; rd	C = B				
4	MDR = Mem	MAR = C				
5			B = MDR			
6			C = B	B = TOS		
7			H = C; wr	C = B	B = SP	
8			Mem = MDR	MDR = C	C = B – 1	B = H
9					MAR = C; wr	C = B
10					Mem = MDR	TOS = C
11						goto (MBR1)

Csővonalas megvalósítás (MIC-3)

- **1. ciklus**

- Swap1 mikroutasítás végrehajtása elindul
 - 1. mikrolépés

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						

Csővonalas megvalósítás (MIC-3)

- **2. ciklus**

- Swap1 második mikrolépése
- Swap2 első mikrolépése

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3						
4						
5						
6						
7						
8						
9						
10						
11						

Csővonalas megvalósítás (MIC-3)

- **3. ciklus**

- Swap1 és Swap2 folytatása
- Swap3 nem indítható!
 - Meg kell várni MDR-t az első utasítás eredményeként! (RAW függőség)

Read After Write



	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3	MAR = C; rd	C = B	Elakadás!			
4						
5						
6						
7						
8						
9						
10						
11						

Csővonalas megvalósítás (MIC-3)

- **4. ciklus**

- Swap1 és Swap2 véget ér
- MDR felhasználhatóvá válik

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3	MAR = C; rd	C = B	Elakadás!			
4	MDR = Mem	MAR = C	Elakadás!			
5						
6						
7						
8						
9						
10						
11						

Csővonalas megvalósítás (MIC-3)

- 5. ciklus
 - Swap3 elindul

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3	MAR = C; rd	C = B				
4	MDR = Mem	MAR = C				
5			B = MDR			
6						
7						
8						
9						
10						
11						

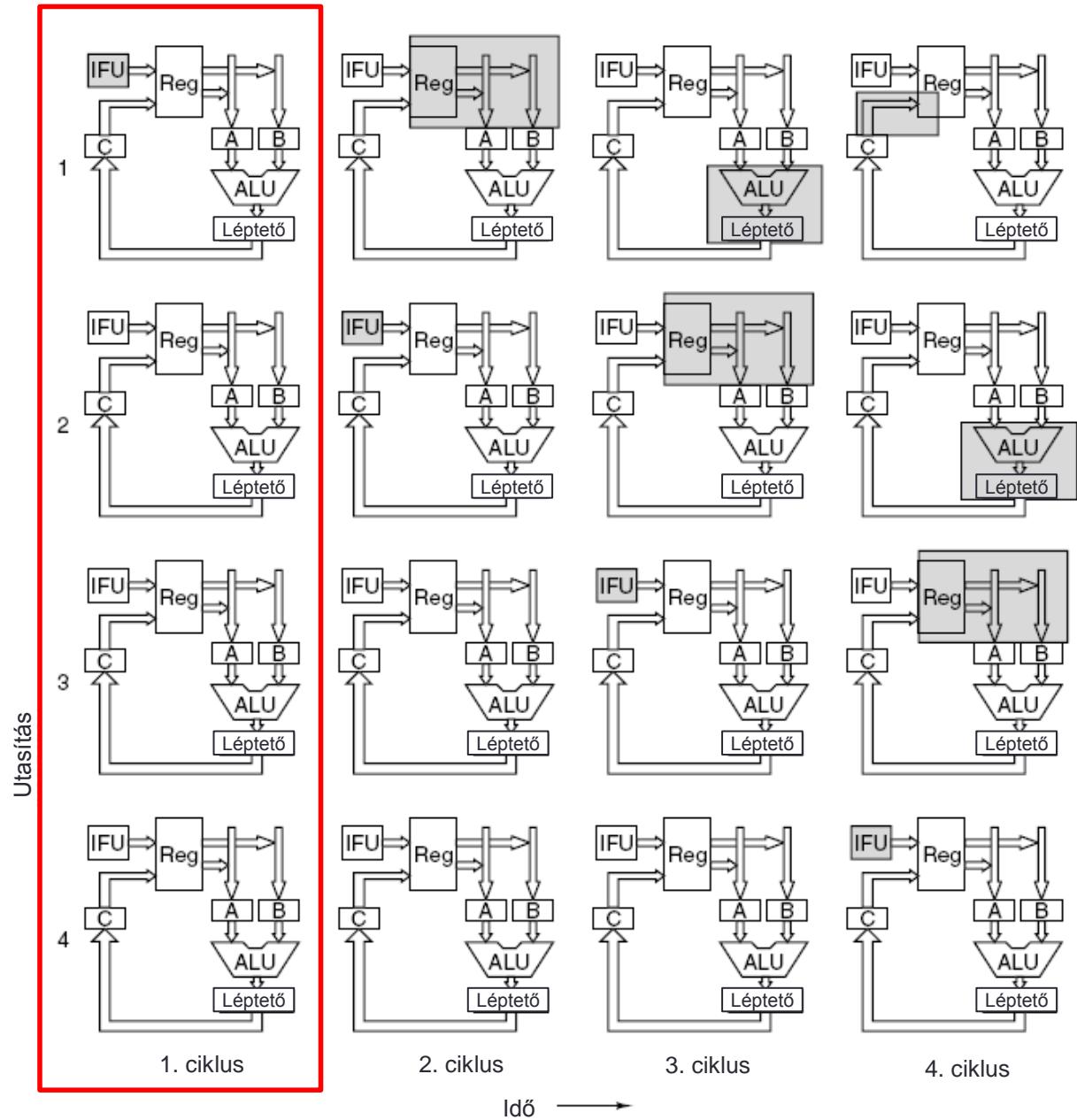
Csővonalas megvalósítás (MIC-3)

- **SWAP utasítás mikrolépései MIC-3-on**
 - Teljes kód

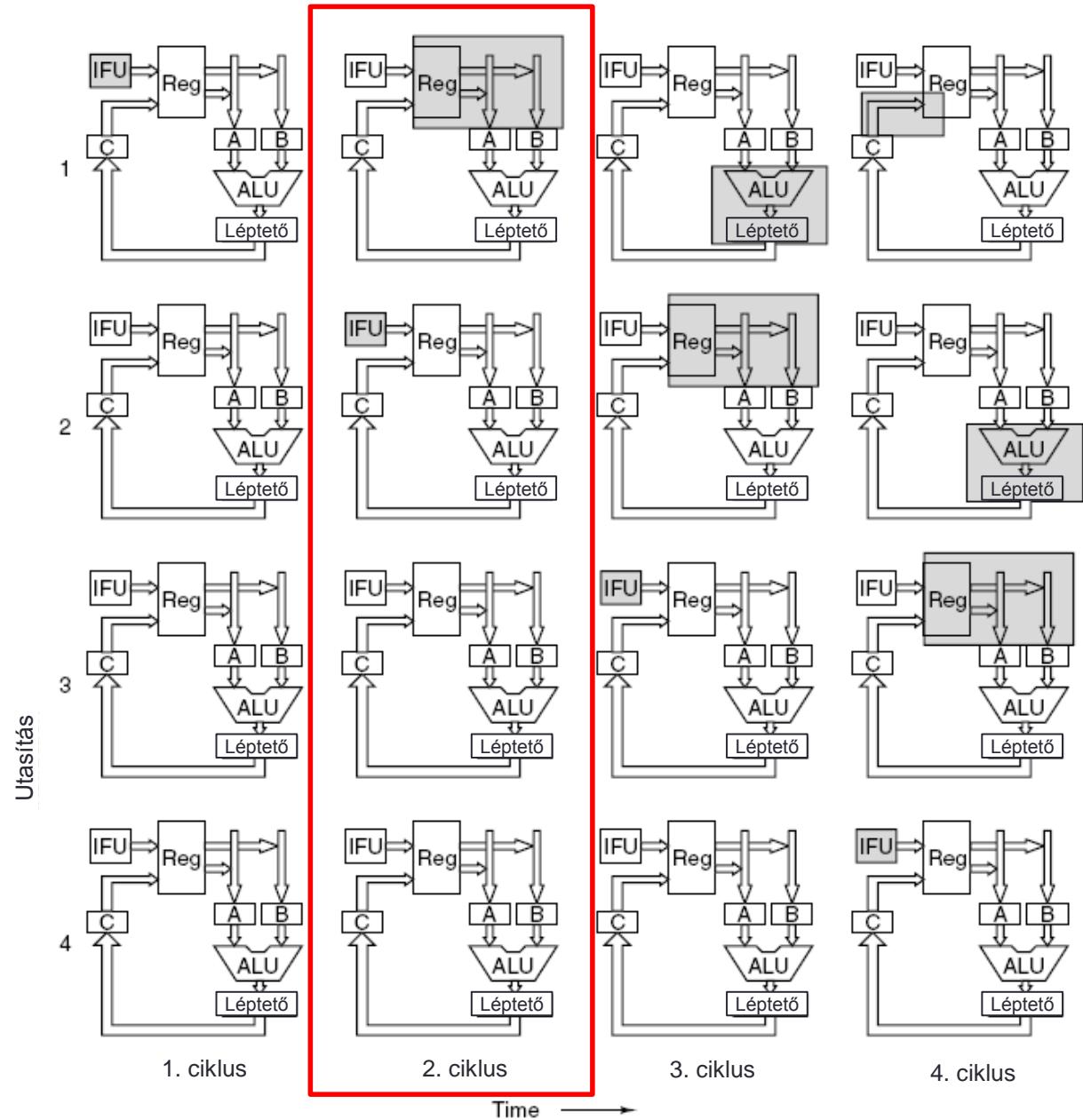
	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Ciklus	MAR = SP – 1; rd	MAR = SP	H = MDR; wr	MDR = TOS	MAR = SP – 1; rd	TOS = H; goto (MBR1)
1	B = SP					
2	C = B – 1	B = SP				
3	MAR = C; rd	C = B				
4	MDR = Mem	MAR = C				
5			B = MDR			
6			C = B	B = TOS		
7			H = C; wr	C = B	B = SP	
8			Mem = MDR	MDR = C	C = B – 1	B = H
9					MAR = C; wr	C = B
10					Mem = MDR	TOS = C
11						goto (MBR1)

- IFU betölti az első utasítást

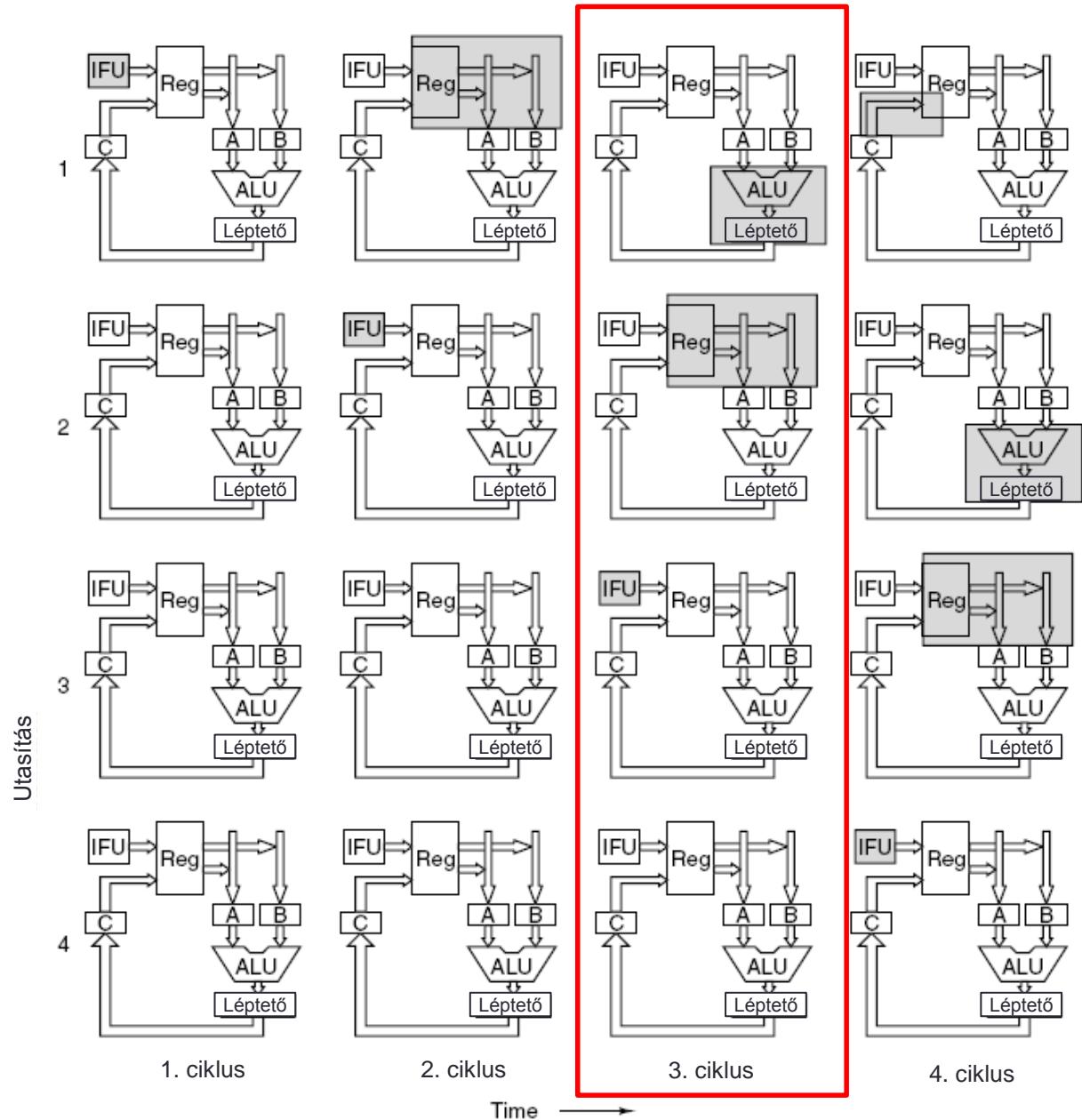
Megjegyzés: ez egy általános csővezeték végrehajtást mutat, ezért nem jelenik meg a végrehajtás során az előző példa 3. utasításának blokkolódása!



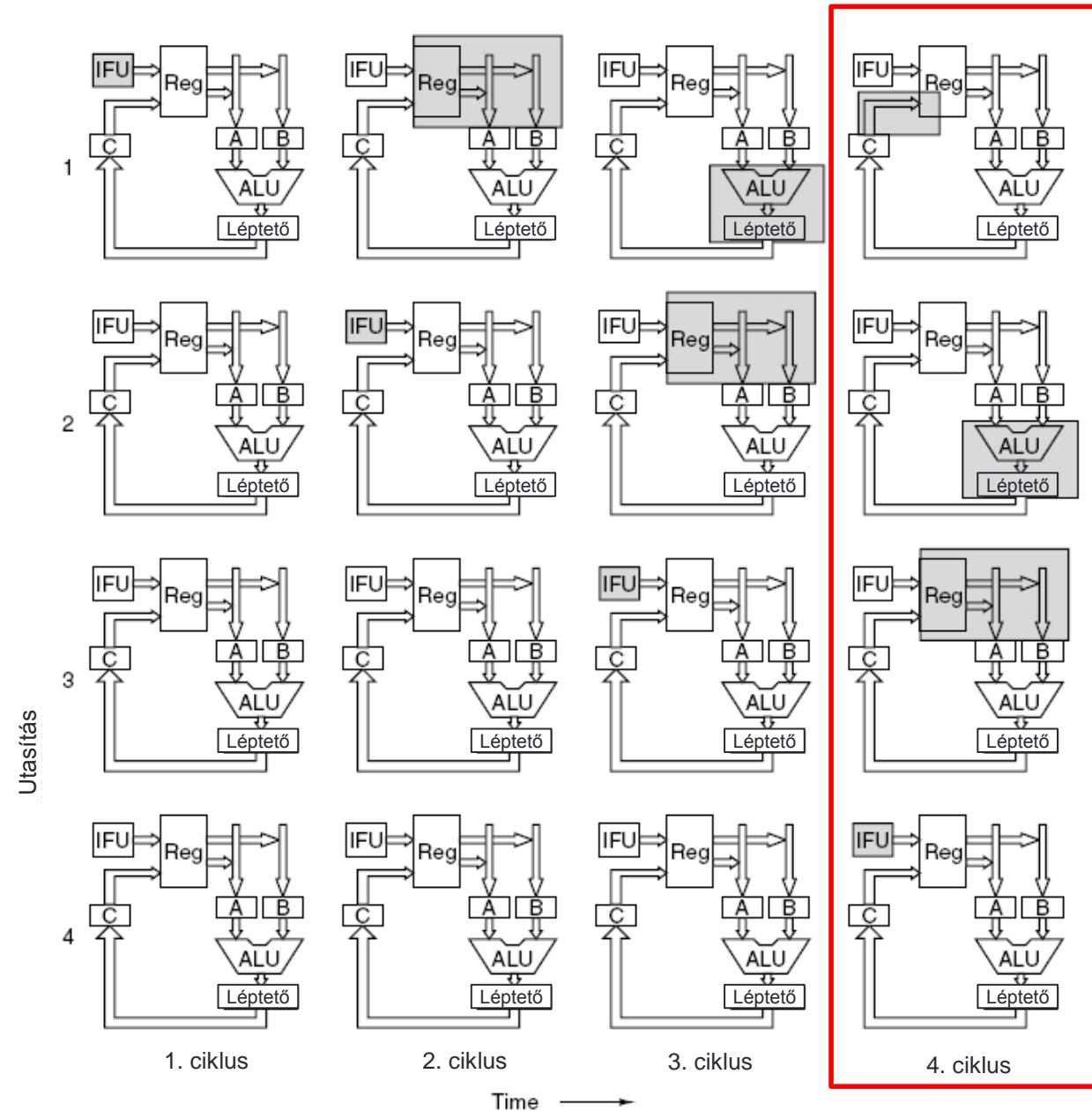
- Az 1. utasítás által kért regiszterek A és B tárolóba töltődnek
- IFU betölti a 2. utasítást



- Az 1. utasítás az ALU-t használja
- A és B tároló a 2. utasításnak megfelelően töltődik fel
- IFU betölti a 3. utasítást



- Az 1. utasítás eredménye tárolódik
- A 2. utasítás használja az ALU-t
- A 3. utasításnak megfelelő regiszterek A és B tárolókba kerülnek
- IFU betölti a 4. utasítást



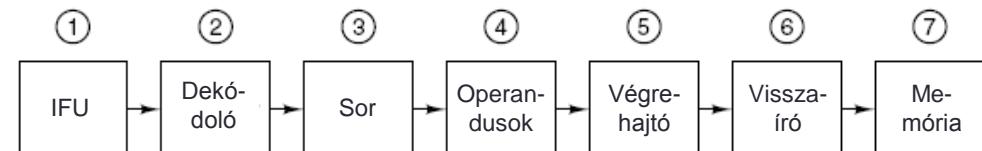
Hétszakaszú csővezeték (MIC-4)

- **Korábbi MIC architektúrák**

- Nem értelmezik a bájtfolyamot
- Nem tudják, hogy a következő bájt utasítás vagy operandus
- Nem ismerik az utasítás méretét

- **MIC-4 újdonságok**

- Dekódoló egység
 - Bájtfolyam dekódolása utasításokként
 - Átalakítja mikroműveletek sorozatává egy ROM-táblázat alapján
- Sor
 - Végrehajtáshoz sorba állítja a mikroműveleteket
- MIR1, ..., MIR4
 - 4-7. szakaszok irányítására



Teljesítmény növelése

- **Megvalósítás javítása**

- Pl. új CPU, memória beépítése az architektúra módosítása nélkül
- Módosítás nélkül futnak a régi programok
- Nagyobb órajel
- CPU teljesítményének javítása
 - Gyorsítótár
 - Elágazásjövendölés
 - Sorrendtől eltérő végrehajtás regiszterátnevezéssel
 - Feltételezett végrehajtás

- **Architektúra tökéletesítése**

- Új utasítások, regiszterek
 - A régiek megtartásával a kompatibilitás megmarad
 - A szoftverek újrafordíthatók az újdonságok kihasználására

Gyorsítótár (Cache)

• Probléma

- A központi memória elérése lassú!
 - Pedig onnan töltődnek be az utasítások és az adatok!
- Gyors elérésű memória rendkívül drága!
 - És nem is érdemes a fizikai mérete miatt nagyon a CPU-ra integrálni
 - Kisebb méretű alkalmazása azért még kifizetődő
 - Jól kell tudni használni!

• Lokalitási elv

- *Térbeli lokalitás*
 - Az elkövetkezendő utasítások végrehajtásakor valószínűleg el kell érni a jelenlegi memóriacím után következő bájtokat
- *Időbeli lokalitás*
 - A nemrégiben elért memóriahelyeket esetleg újra el kell érni

• Ötlet

- Tároljuk ezeket a bájtokat gyors elérésű memóriában!
- Ha szükség van rájuk, onnan töltsük be, ne a központi memóriából

Gyorsítótár

- **Fogalmak**

- **Találati arány:** h

- Az összes hivatkozás mekkora hányada szolgálható ki a gyorsítótárból

- **Hiba arány:** $1-h$

- Ha a gyorsítótár elérési ideje: c , a memória elérési ideje: m , akkor
 $\text{átlagos elérési idő} = c + (1-h) \cdot m$

- Szerencsétlen esetben lassabb lehet, mint a közvetlen memóriacímzés!

- **Gyorsítósor**

- Rögzített méretű blokkok a gyorsítótárban (pl. 4-64 bájt)

- Teljes blokk mozgatásra kerül

- Nagyobb méretű sor esetén

- a hivatkozott cím nagyobb környezete lesz a gyorsítótárban, viszont
 - nagyobb a sor betöltési ideje is,
 - ugyanakkora tárban kevesebb gyorsítósor fér el

A gyorsítótár-elérés mindenképpen megtörténik + ha nem szolgálható ki onnan a kérés, akkor még a memória is dolgozik!

Gyorsítótár

- **Szétválasztott gyorsítótár**
 - Külön utasításoknak és adatoknak
 - Egyik szállítószalag végzi az utasítás, másik az operandus előolvasást
 - Az utasítás gyorsító tárat sohasem kell visszaírni
 - Az utasítások nem módosulnak
- **Egyesített gyorsítótár**
 - Nem lehetséges párhuzamosítás
- **Hierarchia**
 - Elsődleges gyorsítótár (L1), a CPU lapkán (KB nagyságrendű),
 - Másodlagos gyorsítótár (L2), a CPU-val egy tokban (~MB),
 - További szintek: külön tokban (néhány MB SRAM).

Gyorsítótár

- **Direkt leképezésű gyorsítótarak**
 - Adott számú bejegyzéssel rendelkező táblázat
 - Pl. 2048 darab
 - Bejegyzésenként rögzített méretű gyorsítósor
 - Pl. 32 bájt
 - A memóriacímből egyértelműen számítható, melyik bejegyzésnél kell keresni
 - Ha nincs ott, akkor nincs a gyorsítótárban
 - Jól meghatározható címtartományok ugyanarra a bejegyzésre kerülnek
 - Ütközés előfordulhat!
 - Gyakran alkalmazott elv

Direkt leképezésű gyorsítótár

- **Bejegyzés részei** (2048 bejegyzéses, 32 bájtos gyorsítósor esetén)
 - *Valid bit*: érvényes-e az adat (kezdetben minden egyik hamis)
 - *TAG*: 16 bites egyedi érték. Adat származási helyét azonosítja.
 - *DATA*: 32 bájt, a központi memóriában lévő adat másolata

Bejegyzés	Valid	TAG	DATA	Hozzáartozó címtartomány (TAG alapján)
2047				
...
3				96-127, 65632-65663, 131168-131199, ...
2	2			64-95, 65600-65631, 131136-131167, ...
1				32-63, 65568-65599, 131103-131135, ...
0				0-31, 65536-65567, 131072-131103, ...

- 32 bites virtuális cím



Direkt leképezésű gyorsítótár

- **Címzés menete**

- CPU beállítja a memóriacímet (32 bites virtuális cím)
- Hardver kiveszi a 11 **LINE** bitet, a táblázat indexelésére
 - $2^{11} = 2048$
- Táblázat bejegyzés **Valid** bitjének ellenőrzése
 - *Ha hamis*
 - Nincs a gyorsítótárban → központi memória címzés kell; gyorsítósor a gyorsítótárba (**DATA**), **TAG** és **Valid** kitöltése; adat szolgáltatása
 - *Ha igaz*
 - **TAG** mezők összehasonlítása
 - *Ha különböznek*
 - Más van a gyorsítótárban → központi memória címzés kell; ha változott az adattartalom, akkor visszaírás, majd betöltés, mint fent, a *hamis* ágon volt
 - *Ha egyeznek*
 - **WORD** és **BYTE** által megadott szó vagy bájt elérése

Példa #1 a működésre

- C példaprogram

```
int array_A[16], array_B[16], array_C[16];  
...  
int idx;  
for(idx = 0, idx < 16; idx++)  
{  
    array_C[idx] = array_A[idx] + array_B[idx];  
}
```

- Feltesszük, hogy az `int` típus 32 bites egész értéket jelöl.
- Így a tömbök 64 bájt méretű memóriaterületet foglalnak.
- Vegyük észre, hogy ha a memóriában bájtfolytonosan helyezkednek el, akkor tömbönként 2-2 gyorsítósor foglalnak el, a gyorsítótárban egymás utáni sorokban.

Példa #1 a működésre

- **Lépések**

- Feltesszük, hogy kezdetben üres a gyorsítótár.
- **idx = 0**

- **array_A[0]** elérése

- Mivel nincs a gyorsítótárban, ezért az első 32 bájtjának (8 tömb értékének) beolvasása a **0** indexű sorba.
- **0** indexű szó **0.** bájttól beolvasásra

- **array_B[0]** elérése

- Mivel nincs a gyorsítótárban, ezért az első 32 bájtjának (8 tömb értékének) beolvasása a **2** indexű sorba.
- **0** indexű szó **0.** bájttól beolvasásra

- **array_C[0]** elérése

- Mivel nincs a gyorsítótárban, ezért az első 32 bájtjának (8 tömb értékének) beolvasása a **4** indexű sorba.
- Összeg írása a **0** indexű szó **0.** bájtjától

Bejegyzés	Valid	Tag	Data	Hozzá tartozó címtartomány (Tag alapján)
2047				
...
3				96-127,
2				64-95, array_B[0-7]
1				32-63,
0				0-31, array_A[0-7]

Példa #1 a működésre

- Lépések

- `idx = 1`

- `array_A[1]` elérése

- Elérhető a gyorsítótárban: `1` indexű szó `0.` bájttól beolvasásra

- `array_B[1]` elérése

- Elérhető a gyorsítótárban: `1` indexű szó `0.` bájttól beolvasásra

- `array_C[1]` elérése

- Elérhető a gyorsítótárban: Összeg írása az `1` indexű szó `0.` bájtjától

- Nagyobb `idx` értékek esetén

- `idx = 0, ..., 7` kiszolgálható a `0, 2` és `4` sorokból

- `idx = 8` esetén a következő 32 bájtok beolvasónak az `1, 3` és `5` gyorsítósorokba

- `idx = 9, ..., 15` esetén már minden bájt tömb teljes egészében a gyorsítótárban van, onnan kiszolgálható

Bejegyzés	Valid	Tag	Data	Hozzá tartozó címtartomány (Tag alapján)
2047				
...
3				96-127, <code>array_B[8-15]</code>
2				64-95, <code>array_B[0-7]</code>
1				32-63, <code>array_A[8-15]</code>
0				0-31, <code>array_A[0-7]</code>

Példa #2 a működésre

- C példaprogram

```
int array_A[16384], array_B[16384], array_C[16384];  
...  
int idx;  
for(idx = 0, idx < 16384; idx++)  
{  
    array_C[idx] = array_A[idx] + array_B[idx];  
}
```

- Feltesszük, hogy az `int` típus 32 bites egész értéket jelöl.
- Így a tömbök 64 kB méretű memóriaterületet foglalnak.
- Vegyük észre, hogy ha a memóriában bájt folytonosan helyezkednek el, akkor ugyanarra a gyorsítósorra kerülnek az azonos tömbindexű elemek, ami folyamatos ütközést fog okozni!

Példa #2 a működésre

- **Lépések**

- Feltesszük, hogy kezdetben üres a gyorsítótár.
- **`idx = 0`**

- **`array_A[0]` elérése (0 memóriacím index)**
 - Mivel nincs a gyorsítótárban, ezért az első 32 bájtjának (8 tömb értékének) beolvasása a 0 indexű sorba.
 - 0 indexű szó 0. bájttól beolvasásra
- **`array_B[0]` elérése (65536 memóriacím index)**
 - Van adat a gyorsítótárban, de az az `array_A` első 32 bájtja (**TAG** alapján ütközés van) ezért az első 32 bájtjának (8 tömb értékének) beolvasása a 0 indexű sorba, felülírva `array_A` értékeit!
 - 0 indexű szó 0. bájttól beolvasásra
- **`array_C[0]` elérése (131072 memóriacím index)**
 - Van adat a gyorsítótárban, de az az `array_B` első 32 bájtja (**TAG** alapján ütközés van) ezért az első 32 bájtjának (8 tömb értékének) beolvasása a 0 indexű sorba, felülírva `array_B` értékeit!
 - Összeg írása a 0 indexű szó 0. bájtjától

- **Nagyobb `idx` értékek**

- **Mindig ütközés van, folyamatos gyorsítósor olvasás kell!**
- **`array_A` olvasáskor ráadásul ki kell írni a memóriába a korábbi `array_C` értékeket!**

Bejegyzés	Valid	Tag	Data	Hozzáartozó címtartomány (Tag alapján)
2047				
...
3				96-127, 65632-65663, 131168-131199, ...
2				64-95, 65600-65631, 131136-131167, ...
1				32-63, 65568-65599, 131103-131135, ...
0				0-31, 65536-65567, 131072-131103, ...

Gyorsítótár

- **Direkt leképezésű problémái**

- 1 bejegyzéshez a címtartomány több része tartozik
- Ezek közül csak 1 lehet a gyorsítótárban
- Ha másikra történik hivatkozás, cserálni kell
 - Gyakori csere gyenge teljesítményhez vezet!
 - Pl. Fordítóprogramok optimalizálhatják az adatok memóriacímét
 - Szétválasztott gyorsítótár esetén kisebb probléma

- **Halmazkezelésű (csoportasszociatív) gyorsítótár**

- Két vagy több gyorsítósor is tartozhat 1 táblázat bejegyzéshez
 - Új feladat a bejegyzéshalmaz nyilvántartása
- n utas halmazkezelésű gyorsítótár: n darab lehetséges bejegyzés
- Bonyolultabb áramkört igényel, de megéri
 - Lehet pl. 2 vagy 4 utas
 - Akár 2048 utas is (nem feltétlenül célszerű)

A korábbi Példa #2 problémára már egy 3 utas gyorsítótár is megoldást ad.

Halmazkezelésű gyorsítótár

- **Új probléma**
 - Ha egy bejegyzéshez betelt minden gyorsítósor, melyiket kell eldobni?
- **Megoldási lehetőség**
 - Legrégebben használt eldobása (LRU – Least Recently Used)
 - Rendezett lista nyilvántartása szükséges minden bejegyzéshalmazhoz

Gyorsítótárak írási problémája

- **Mi történjen, ha memóriába íráskor az adat a gyorsítótárban van?**
 - Eldobhatjuk a bejegyzést és az adatot a főmemóriába írhatjuk, vagy
 - Frissíthetjük a gyorsítótárban (ez a jellemzőbb)
- **Ha a gyorsítótárban frissítünk, mikor kerüljön ki a főmemóriába?**
 - **Írásáteresztés:** az adat azonnal kiírásra kerül a főmemóriába is
 - Egyszerűbb, megbízhatóbb, de nagyobb írásforgalom
 - **Késleltetett írás:** akkor kerül a főmemóriába, ha a gyorsítósor eldobásra kerül
- **Mi történjen, ha memóriába íráskor az adat nincs a gyorsítótárban?**
 - **Írunk csak a főmemóriába**
 - Írásáteresztő stratégiakor jellemző
 - **Írásallokálás:** töltsük be az adatot a gyorsítótárba
 - Késleltetett írás stratégia esetén jellemző
 - Csak akkor előnyös, ha ismételt írások történnek egy soron belül

Elágazásjövendölés

- **Csővezeték architektúra problémái**

- Lineáris kóddal dolgoznak a legjobban
- A gyakorlatban sok *feltételes* vagy *feltétel nélküli* elágazás van!

<pre>if(i == 0) k = 1; else k = 2;</pre>	<pre>CMP i, 0 BNE Else Then: MOV k, 1 BR Next Else: MOV k, 2 Next: ...</pre>
--	--

C nyelvű programrészlet

Assembly kód

- **Mire kiderül, hogy hova kell ugrani, a rákövetkező utasítás végrehajtása már megkezdődött a csővezetéken!**
 - Ezt hogyan tudjuk „visszavonni” vagy „hatástalanítani”?
 - Meg tudjuk-e tippelni egy feltételes elágazásnál, hogy melyik teljesül?

Elágazásjövendölés

- **Feltétel nélküli ugrás problémája**
 - Tudjuk hova kell ugrani, de a cím csak a csővezeték 2. szakaszában derül ki!
 - Dekódolás
 - A rákövetkező utasítás végrehajtása ekkor már elkezdődött!
 - UltraSPARC III
 - **Végrehajtódik a rákövetkező utasítás!!!**
 - **Eltolási rés:** elágazás utáni tartomány, amely végrehajtódik (leggyakrabban NOP kerül ide)
 - Pentium 4
 - Bonyolult módon, de kikerüli a végrehajtást

Elágazásjövendölés

- **Feltételes ugrás problémája**

- Itt is létrejönnek eltolási rések, és ráadásul
- a feltétel kiértékelése a csővezeték egy még későbbi szakaszán történik meg. Ha ezt kivárjuk, **bedugulás** következik be.
- **Jövendölés:** próbáljuk kitalálni, melyik ág következik be, és azt kezdjük el végrehajtani

- **Mi történjen jó jövendölés esetén?**

- Szuper! Nincs tennivaló!

- **Mi történjen rossz jövendölés esetén?**

- Engedjük az utasítások végrehajtását, amíg nem változtatják meg a gép állapotát
 - Pl. amíg nem ír regisztert/memóriát
 - Vagy használunk **rejtett firkáló** regisztert
 - Csak akkor kerül az érték az „igaziba”, ha ez a jó ág
- Jegyezzük fel a módosuló regiszterek értékét, hogy visszaállítható legyen, ha rossz a jövendölés

Jövendölési stratégiák

- **Egyszerű stratégia**

- minden *visszafelé történő* feltételes elágazás bekövetkezik
 - pl. ciklus végi visszaugrások
- Egyetlen *előreirányuló* feltételes elágazás sem következik be
 - Jó a tipp pl. hibakezelő ágak esetén (ha hiba van, átugrunk egy kódrészt)

- **Előnye**

- Egyszerűen alkalmazható

- **Hátránya**

- Nem túl jó, de jobb mint semmi

Jövendölési stratégiák

- **Dinamikus elágazásjövendölés**

- Feljegyezzük, hogy adott memóriacímen milyen irányba történt elágazás legutóbb, és ezt vesszük
- Előzménytábla: gyorsítótárhoz hasonló felépítés
 - Valid bit, elágazási cím, igaz/hamis legutóbb
 - n -utas megvalósítás is megoldható
- Kiegészíthető úgy, hogy csak a második más irányú ág után változzon
 - Ciklusok esetén kilépés után és újból belépéskor hasznos

- **Előnye**

- Futási időben működik
- Alkalmazkodik a program viselkedéséhez

- **Hátránya**

- Speciális és drága hardver szükséges

Jövendölési stratégiák

- **Statikus elágazásjövendölés**
 - Fordítóprogram ad segítséget
 - Pl.: `for(i = 0; i < 1000000; i++)` esetén a ciklus végi ugrást szinte mindenkorán végre kell hajtani
 - UltraSPARC III: **jövendölő elágazó utasítások**
 - Az utasítás kódja tartalmazza a jövendölést, amit a fordítóprogram tölt ki
 - „ugrás, ha a feltétel teljesül – és várhatóan teljesülni fog”
 - „ugrás, ha a feltétel teljesül – de várhatóan nem fog teljesülni”
 - Megfigyelésen alapuló stratégia
 - A programot futtatjuk
 - Közben figyeljük az elágazások viselkedését
 - Ezt betápláljuk a fordítóprogramba, ami előállítja a jövendölő elágazó utasításokat

Sorrendtől eltérő végrehajtás

- **Modern CPU-k**

- Csővezetékesek és szuperskalárisak
- Egy ciklusban akár több utasítás végrehajtását is elindíthatják
 - Akár 4-6 utasítás is indulhat!
- Elakadás következik be, ha egy utasítás egy megelőző eredményére vár (függő utasítás; lásd 13. dia példáját!)
 - **RAW** (Read After Write): olyan adatot olvasnánk, amit egy előző még ír
 - **WAR** (Write After Read): olyan helyre írnánk, amelyről egy előző még olvas
 - **WAW** (Write After Write): olyan helyre írnánk, amelyet egy előző még ír

Sorrendtől eltérő végrehajtás

- Ha alkalmasak egy ciklusban több utasítás indítására, akkor **célszerű átugrani a függő utasításokat** (rákövetkezőkkel folytatni)
 - Lineáris (elágazás nélküli) kódblokkon belül működik
 - **Vigyázni kell!**
 - Ugyanazt az eredményt kell szolgáltatni, mint a sorrendben történő végrehajtásnak!
 - Függőségeket kezelní!
- **Sorrend szerint befejezés megkövetelése**
 - Fontos lehet pl. a megszakítások kezelése miatt
 - Nehéz lenne visszaállítani a megszakítás bekövetkezte előtti állapotot
 - „**Pontos megszakítás**” tulajdonság

Regiszterátnevezés

- **Bizonyos WAR és WAW függőségek kiküszöbölése**
 - Függőség esetén az eredetileg kért regiszter helyett egy titkos regiszter kerül felhasználásra
 - Akár több tucat titkos regisztere is lehet a CPU-nak!
 - Táblázatot tart nyilván a leképezések nyomon követésére
 - A függőség megszűnése után
 - az eredmény bekerülhet az eredetileg megnevezett regiszterbe, vagy
 - a további utasítások erre a titkos regiszterre vonatkozhatnak
- **RAW függőség így nem kiküszöbölhető**
 - Mindenképpen meg kell várni az előző író utasítás befejeződését, mert az szolgáltatja az olvasandó adatot

Feltételezett végrehajtás

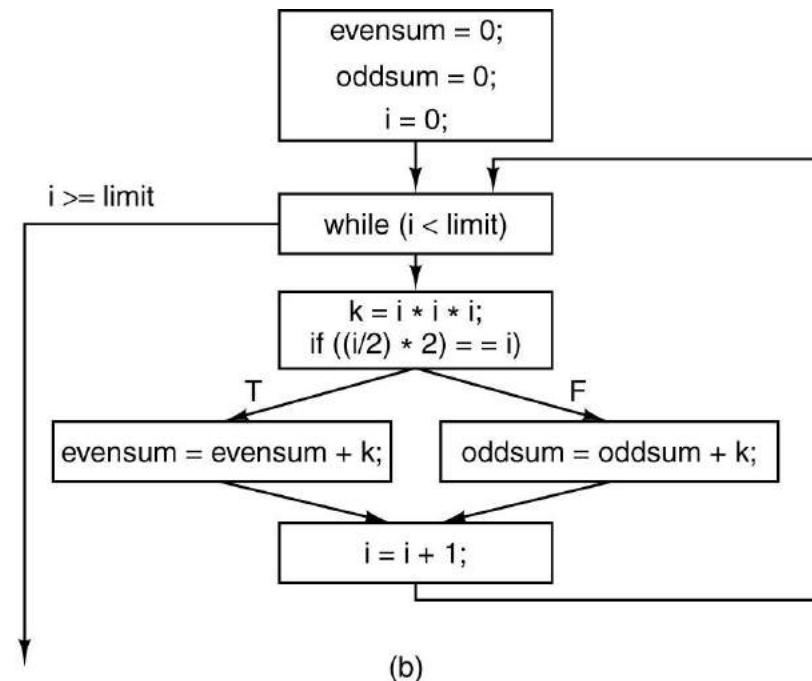
- **Programok**

- Alapblokkok: lineáris kódsorozat, vezérlési szerkezet nélkül
- Ezeket összekötő vezérlési szerkezetek
- **Végrehajtunk olyan kódot, amiről még nem tudjuk, hogy szükség lesz-e rá**

```

evenum = 0;
oddsum = 0;
i = 0;
while (i < limit) {
    k = i * i * i;
    if (((i/2) * 2) == i)
        evenum = evenum + k;
    else
        oddsum = oddsum + k;
    i = i + 1;
}
    
```

(a)



Feltételezett végrehajtás

- Megengedjük az alapblokkok határain átnyúló végrehajtás-sorrendi átrendezéseket
 - *Emelésnek* nevezzük
 - Fordítóprogram közreműködése is kell
- Felmerülő kérdések
 - Nem lehetnek visszavonhatatlan eredmények
 - Pl. regiszter átnevezések használatával
 - Mi történik, ha a feltételesen végrehajtott utasítás kivételt okoz?
 - Mi történik, ha gyorsítótáriányt okoz?
 - Spekulatív betöltő utasítás: ha elérhető a gyorsítótárban, akkor betölt, egyenként feladja
 - Mi történik, ha csapdát vált ki?
 - `if (x>0) z=y/x; // nem lehet (ne) 0-val való osztás`
 - Mérgezés-bit: csak akkor okoz csapdát, ha erre az ágra kerül a vezérlés

Mikroarchitektúra példák

- Csak érdekesség, nem kell megtanulni!

	Pentium 4	UltraSPARC III cu	8051
Típus	Asztali PC	Asztali PC	Beágyazott rendszerek
Utasításrendszer	CISC, kevés regiszter, változó hosszúságú utasítások, de belső csővezetékes RISC mag	RISC, a gépi utasítások mikroműveletek, általában 4 bájtos utasítások	Utasítások általában 1 órajelciklus alatt végrehajtódnak, speciális célú 8 és 16 bites regiszterek
Gyorsítótár	L2: belső, 1 MB, 8 utas, halmazkezelésű, visszaíró, 128 bájtos gyorsítósor L1 adat: 4 utas, halmazkezelésű, 64 bájtos gyorsítósor, írásáteresztő L2-be L1 kód: 12K mikroművelet	L2: külső, 1, 2, 8 MB L1 adat: 4 utas halmazkezelésű, 32 bájtos gyorsítósor L1 kód: 4 utas halmazkezelésű, 2-2 KB előre betöltő és tároló gyorsítótárok	Nincs
Utasításbetöltő	L2 gyorsítótárból dekódol CISC-ből RISC mikroműveletekre	Akár 4 utasítás előkészítése ciklusonként	
Végrehajtás	Sorrend szerinti kiosztás, sorrendtől eltérő végrehajtás, sorrend szerinti befejezés, regiszterátnevezés	Sorrend szerinti kiosztás, sorrendtől eltérő végrehajtás, sorrend szerinti befejezés; regiszterátnevezés	Sorrend szerinti kiosztás, sorrend szerinti végrehajtás, sorrend szerinti befejezés
Elágazásjövendölés	L1 gyorsítótárból 4096 utolsó elágazás tárolása, Ha nincs benne, egyszerű jövendölést használ	Van. A feltételes utasítás utáni utasítást mindenképpen végrehajtja!	Nincs
Egyéb	MIC-4-hez hasonló		Nincs csővezeték MIC-1-szerű

Mikroarchitektúra példák

- Csak érdekesség, nem kell megtanulni!

	Intel Core i7	TI OMAP4430	ATmega168
Típus	Asztali PC	Mobil eszközök, táblagépek	Beágyazott rendszerek
Utasításrendszer	CISC, kevés regiszter, változó hosszúságú utasítások, de belső csővezetékes RISC mag	RISC, a gépi utasítások mikroműveletek, általában 4 bájtosak, új grafikus és multimédiás utasítások	Utasítások általában 1 órajelciklus alatt végrehajtónak, speciális célú 8 bites regiszterek
Gyorsítótár	L3: közös 1-20 MB, 12 utas, halmazkezelésű, 64 bájtos sor, L2: magonkénti belső, 256 KB, 8 utas, halmazkezelésű, visszaíró, 64 bájtos gyorsítósor L1: 32 KB, 8 utas, halmazkezelésű, 64 bájtos gyorsítósor, írásáteresztő L2-be	L2: 1 MB L1 adat: 32KB, 4 utas halmazkezelésű, 32 bájtos sor L1 kód: 32 KB, 4 utas halmazkezelésű, 32 bájtos sor kis méretű ciklusoknak külön gyorsítótár	Nincs
Utasításbetöltő	Mikroutasítások gyorsítótárazása a dekódolás után (L0 tár)	Akár 4 utasítás előkészítése ciklusonként	Csővezetékes betöltés (2 lépéses)
Végrehajtás	Sorrend szerinti kiosztás, sorrendtől eltérhető végrehajtás, sorrend szerinti befejezés, regiszterátnevezés	11 elemű csővezeték; sorrendtől eltérhető végrehajtás, sorrend szerinti befejezés, regiszterátnevezés	Sorrend szerinti kiosztás, sorrend szerinti végrehajtás, sorrend szerinti befejezés
Elágazásjövendölés	Nagyon összetett, de algoritmusa nem publikus	4K méretű elágazásjövendölő tábla, 1K méretű címtábla	Nincs
Egyéb	MIC-4-hez hasonló		MIC-1-szerű

Hibalista

- Tanenbaum könyv (2. magyar nyelvű kiadás)
 - 4.26 ábra (Háromsínes kód ILOAD végrehajtására)
 - iload1 lépésben még nem áll rendelkezésre MBRU!

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Utasításrendszer szintje (ISA)

Tanács Attila

Architektúra szintek

5. Probléma orientált nyelv szintje
fordítás (fordító program)
4. **Assembly nyelv szintje**
fordítás (assembler)
3. Operációs rendszer szintje
részben értelmezés (operációs rendszer)
2. **Gépi utasítás szintje (ISA)**
*ha van mikroprogram, akkor értelmezés
(Kompatibilitás!)*
1. Mikroarchitektúra szintje
hardver
0. Digitális logika szintje

Fogalmak

- **Gépi kód**

- Numerikus gépi nyelv
- Az utasítások és az operandusok számok
- 1 utasítás 1 vagy több bájton kódolódik
- Elemi műveletek végrehajtására

- **Assembly nyelv**

- A numerikus gépi nyelv szimbolikus formája
- **Mnemonik**
 - emlékeztető kód a numerikus utasítások helyett
- Szimbolikus nevek és címek
- Makrók
- Feltételes fordítás
- ...

0013	RESETA	EQU	%00010011	*****
0011	CTLREG	EQU	%00010001	
C003 86 13	INITA	LDA A	#RESETA	RESET ACIA
C005 B7 80 04		STA A	ACIA	SET 8 BITS AND 2 STOP
C008 86 11		LDA A	#CTLREG	
C00A B7 80 04		STA A	ACIA	
C00D 7E C0 F1	JMP	SIGNON		GO TO START OF MONITOR

Cím Gépi kód Szimbólumok + mnemonikok

Megjegyzés

Fogalmak

- **Assembler**
 - A fordító, amely assembly nyelvről gépi kódra fordít
- **Disassembler**
 - Gépi kódból mnemonik kód listázása
 - Vigyázni kell, hogy a listázás kezdőcíme valóban utasításhatáron legyen!
 - Ha szimbólumlista elérhető, akkor Assembly-szerű lista kapható

Assembly nyelv

- **Jellemzők**

- minden utasításnak egyetlen gépi utasítás felel meg
- Architektúránként különböző Assembly nyelv!
 - Pl. Intel, UltraSPARC, RISC-alapú architektúrák
 - Nincs a magasszintű nyelveknél tapasztalható portabilitás!

- **Hátrányok**

- Nehézkes, időigényes
- Sok hibalehetőség
- Hosszadalmasabb hibakeresés, karbantartás

- **Előnyök**

- Hatékonyúság
- A hardver teljes elérhetősége
 - Bizonyos regiszterek magasszintű nyelvekből nem használhatók
 - Hardver közvetlen elérése

Fordítás, szerkesztés

- **Fordító (Compiler)**

- *Forrásnyelvből célnyelvre alakít, pl.:*
 - C++, C -> tárgykód (.o/.obj)
 - **Assembly -> tárgykód (.o/.obj)**
 - Java -> class fájl
 - C -> Assembly (.asm)
 - FORTRAN -> C

- **Szerkesztő (Linker)**

- Tárgykódok összeszerkesztése *futtatható állománnyá*
 - Külső hivatkozások feloldása
 - Virtuális címek feloldása
 - Címterek összefésülése (relokáció)
 - ...

Assembly program fordítása

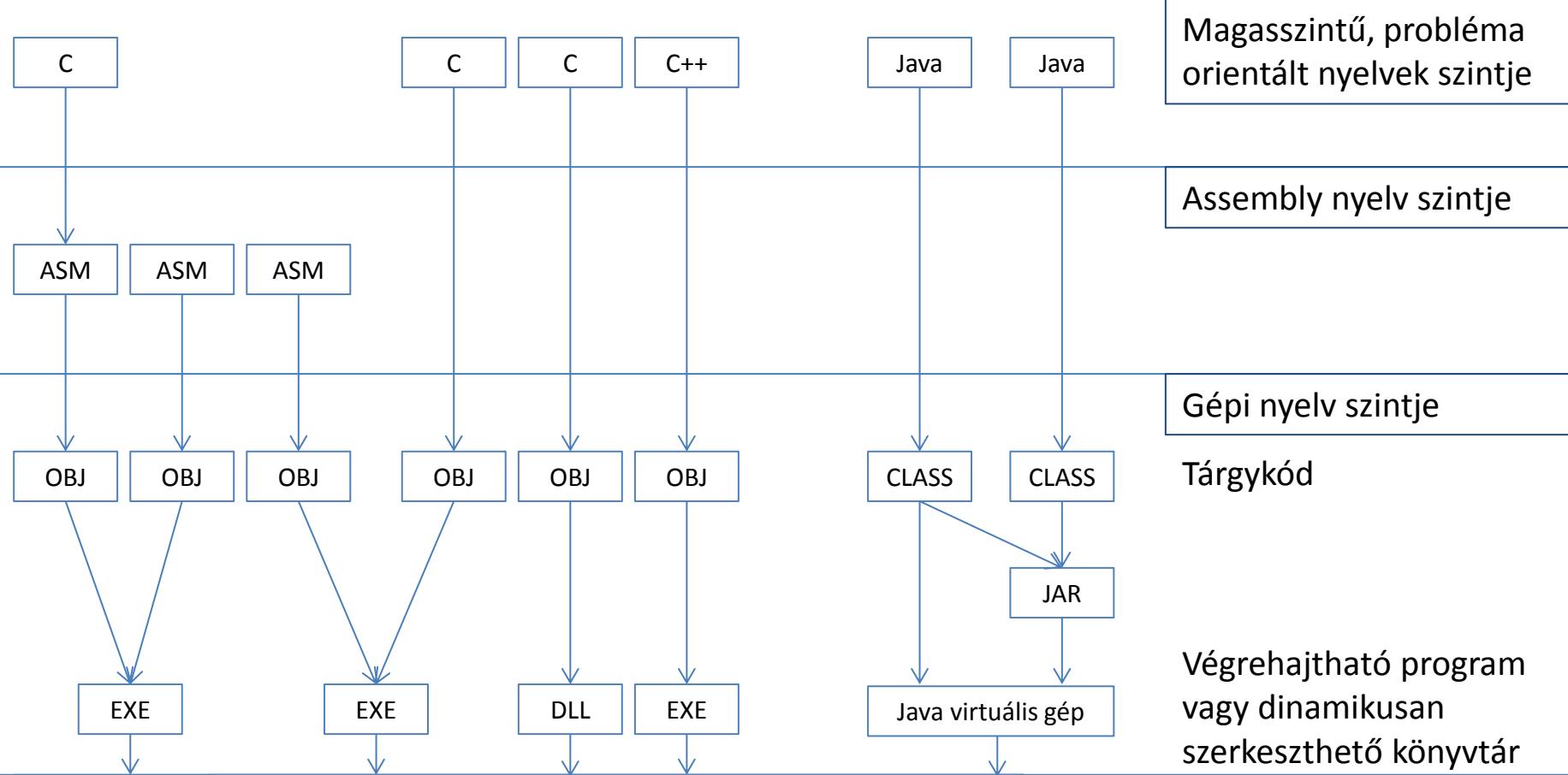
- **Jellemzően kétmenetes fordítók (*compiler*)**
 - **Forráskódból tárgykód készítése**
 - Tárgykód: lefordított utasítások + szerkesztőnek szóló információk
 - **Első menet**
 - Szimbólumtábla felépítése
 - Címkék, változók, literálok, makrók, ... feljegyzése
 - Utasítás-helyszámláló segítségével
 - **Második menet**
 - Tárgykód elkészítése
 - Kimeneti fordítási listák készítése
 - Szerkesztőnek szóló utasítások tárgykódba helyezése

Assembly program fordítása

- **Szerkesztés (*linker*)**
 - Tárgykódok futtatható kóddá szerkesztése
 - Az azonos nevű és osztályú szegmens szeletek egymáshoz illesztése a szegmens szeletek definíciójában megadott módon,
 - egy csoportba sorolt szegmensek egymás után helyezése,
 - relokáció elvégzése,
 - külső hivatkozások feloldása.
 - Típusok
 - Statikus
 - A tárgykód a futtatható állományba kerül
 - Dinamikus
 - Nagyméretű programokban bizonyos eljárások csak nagyon ritkán szükségesek. Ezeket nem kell statikusan a programhoz szerkeszteni. A szerkesztés futási időben történik meg (operációs rendszer).
 - .dll (dynamic link library), .so (shard object - UNIX) fájlok

Fordítás, szerkesztés

- **Fájlformátumok, kiterjesztések**
 - **Tárgykód**
 - .obj: DOS, Windows
 - .o: Unix, Linux, Cygwin, ...
 - Fordító-specifikus formátum!
 - Azonos kiterjesztés még nem jelenti azt, hogy összeszerkeszthetők!
 - **Végrehajtható (futtatható)**
 - .com: kis méretű DOS alkalmazások
 - .exe: DOS, Windows formátum (többféle szerkezet!)
 - .dll: dinamikusan szerkeszthető (Windows)
 - Unix-alapú rendszereknél nincs futtatható kiterjesztés, azt a fájl jogosultsága mondja meg
 - .so: dinamikusan szerkeszthető (Unix, Linux, ...)
 - .a: statikusan szerkeszthető tárgykód gyűjtemény (Unix, Linux...)



Operációs rendszer

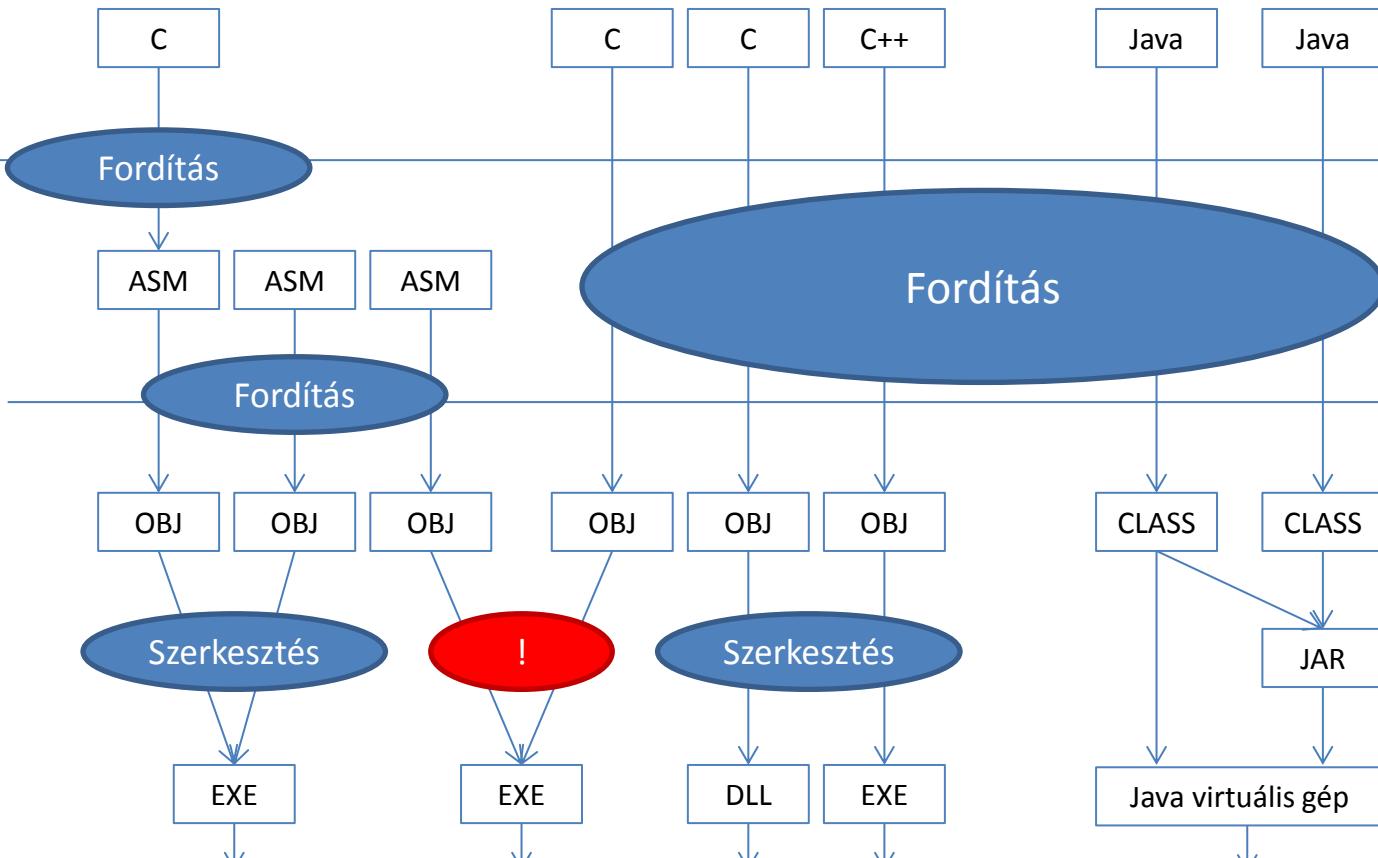
Processzusok ütemezése, erőforrások kezelése, fájlkezelés, hardver absztrakciós szint, ...

BIOS

Meghajtó-programok

Hardver

Az ábrát nem kell megtanulni!



Különböző forrásnyelvekből származó tárgykódok akkor szerkeszthetők össze, amennyiben megegyező formátumúak, valamint egyeznek a hívási konvenciók, paraméterátadások, a memóriamodell, ...

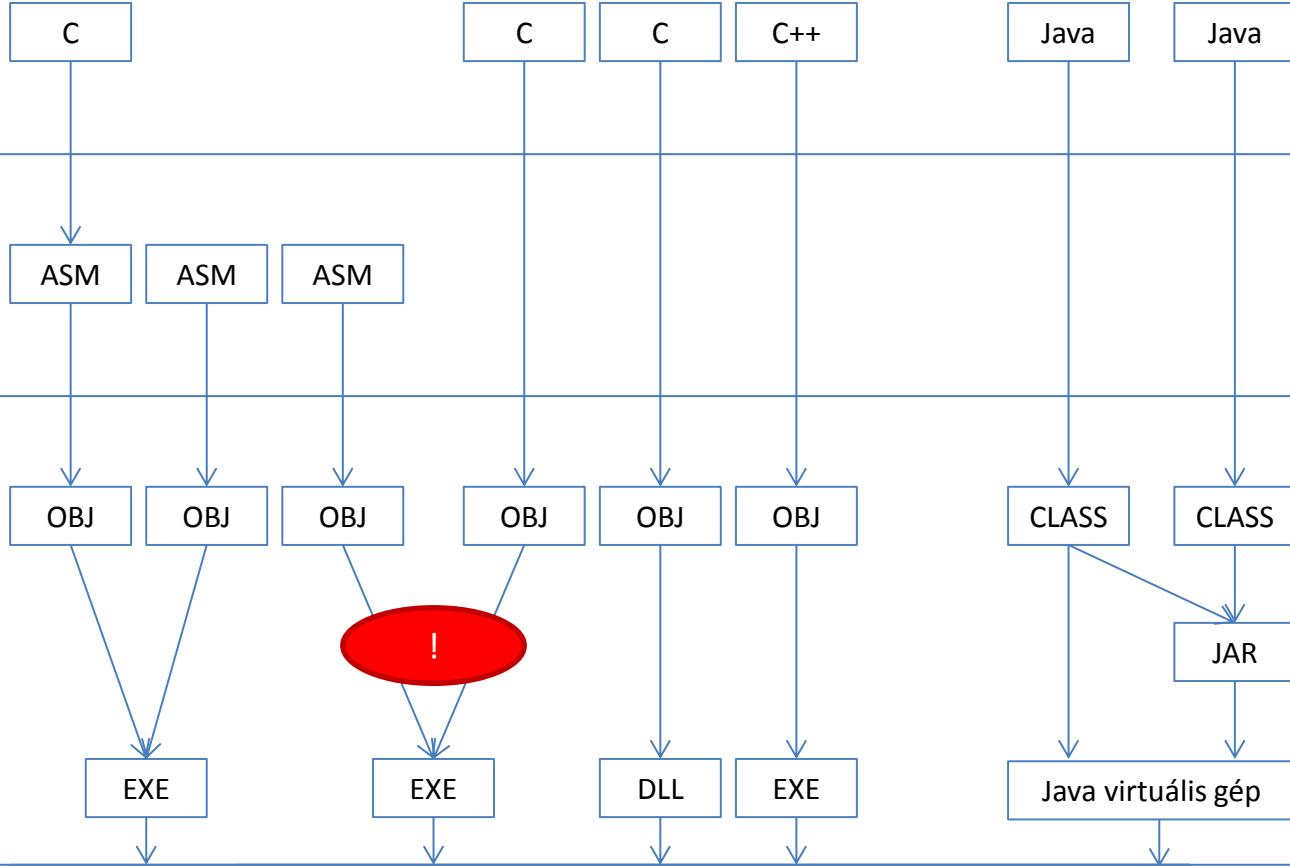
Operációs rendszer

Processzusok ütemezése, erőforrások kezelése, fájlkezelés, hardver absztrakciós szint, ...

BIOS

Meghajtó-programok

Hardver



Assembly:
hardverközeli és/vagy
hatékony implementációt
igénylő részek

Magas szintű nyelv:
GUI, vezérlés, Assembly
betétek hívása

Operációs rendszer

Processzusok ütemezése, erőforrások kezelése, fájlkezelés, hardver
absztrakciós szint, ...

BIOS

Meghajtó-programok

Hardver

Az ábrát nem kell megtanulni!

C → Assembly fordítás példa

- C forráskód

```
#include <stdio.h>
int main( int argc, char *argv[] )
{
    int a, b, sum;
    a = 5;
    b = 7;
    sum = a + b;
    printf( "Osszeg: %d\n", sum );

    return 0;
}
```

A példát nem kell megtanulni!

Generált Assembly kód (Intel Ubuntu Linux)

```

.file  "sum.c"
.section .rodata
.LC0:
.string "Osszeg: %d\n"
.text
.globl main
.type  main, @Function
main:
    pushl %ebp
    movl %esp, %ebp
    andl $-16, %esp
    subl $32, %esp
    movl $5, 28(%esp)
    movl $7, 24(%esp)
    movl 24(%esp), %eax
    movl 28(%esp), %edx
    leal (%edx,%eax), %eax
    movl %eax, 20(%esp)
    movl $.LC0, %eax
    movl 20(%esp), %edx
    movl %edx, 4(%esp)
    movl %eax, (%esp)
    call printf
    movl $0, %eax
    leave
    ret
.size  main, .-main
.ident "GCC: (Ubuntu/Linaro 4.5.2-8ubuntu4) 4.5.2"
.section .note.GNU-stack,"",@progbits

```

A példát nem kell megtanulni!

Generált Assembly kód (SUN Sparc)

```

.file  "sum.c"
gcc2_compiled.:
.section ".rodata"
.align 8
.LL0:
.asciz "Osszeg: %d\n"
.section ".text"
.align 4
.global main
.type  main,#function
.proc  04
main:
    !#PROLOGUE# 0
    save %sp, -128, %sp
    !#PROLOGUE# 1
    st %i0, [%fp+68]
    st %i1, [%fp+72]
    mov %0, %0
    st %0, [%fp-20]
    mov %0, %0
    st %0, [%fp-24]
    ld [%fp-20], %0
    ld [%fp-24], %01
    add %0, %01, %0
    st %0, [%fp-28]
    sethi %hi(.LL0), %01
    or %01, %lo(.LL0), %0
    ld [%fp-28], %01
    call printf, 0
    nop
    mov 0, %i0
    b .LL2
    nop
.LL2:
    ret
    restore
.LLfe1:
.size  main,.LLfe1-main
.ident "GCC: (GNU) 2.95.3 20010315 (release)"

```

Tárgykód (Intel Borland C fordító)

The screenshot shows the Lister debugger interface with two windows. The left window displays assembly code with addresses and opcodes. The right window shows memory dump data. A red box highlights the assembly code area, and another red box highlights the memory dump area. A large brown arrow points from the assembly code area to the memory dump area, indicating the relationship between the two.

```

    push    ebp
    mov     ebp,esp
    mov     eax,5
    mov     edx,7
    add     edx,eax
    mov     eax,edx
    mov     eax,sum
    push    eax
    push    offset s@_printf
    call    _printf
    add    esp,8
    xor    eax,eax
    pop    ebp
    ret

```

Fájl pozíció

A Példát nem kell megtanulni!

Fájl tartalma:
kód, adat és járulékos (pl. szerkesztőnek szóló) információk.

Középen hexaszámokként, jobb oszlopban karakterkódokként.

ISA szint

- **Összekötő kapocs a szoftver és a hardver között**
 - Több magasszintű nyelven legyen programozható a számítógép
 - Közvetlenül az ISA-szint hajtható végre
 - Szükséges egy ISA-szintre történő fordítás
- **ISA-szint tervezése**
 - Hardver oldalról: mi valósítható meg (költség)hatékonyan
 - Szoftver oldalról: milyen utasításokra lenne szükség a magasszintű nyelvekről fordításkor
 - Hosszú időre szóló terv kell!
- **Kompatibilitás**
 - Kulcsfontosságú!
 - Nem lehet pár havonta/évente a szoftverrendszeret újratervezni
 - Visszafelé kompatibilitás
 - Az új futtatni tudja a régit, de fordítva nem (kell) teljesül(jön)
 - Új mikroarchitektúra szint is bevezethető így

ISA szint

- **Fő komponensei**
 - Regiszterkészlet
 - Memóriamodell
 - Felhasználható utasítások és adattípusok
- **Nem része**
 - Mikroarchitektúra tényleges megvalósítása
 - Csővezetékes, szuperskaláris, ...
 - De ezek az ismeretek felhasználhatók optimális gépi kódú programok készítéséhez/fordításához
 - Pl. egész és lebegőpontos ALU-val rendelkező szuperskaláris gép egyszerre végre tud hajtani két ilyen műveletet: érdemes az utasítások sorrendjét így megadni a kódban

ISA szint

- **Védelem nélküli rendszerek**

- minden utasítás végrehajtható a felhasználói programokban, és
- „szabadon garázdálkodhat” (akár az operációs rendszer kódját is felülírhatja)
- Pl. DOS

- **Védett üzemmódok**

- CPU-szintű hardveres védelem (utasítás-végrehajtás, kódterület)

- **Felhasználói mód**

- Bizonyos utasítások végrehajtásának tiltása
 - Pl. gyorsítótár manipulálása

- **Kernelmód**

- Operációs rendszer futtatásához használt
- Nincs korlátozás

Memóriamodellek

- **Korábbi előadásanyagban tárgyalásra került**
 - Cellák, cellaméret
 - Bájtok, szavak
 - Bájtok értékeinek értelmezése
 - Adat, kód
 - Igazítás

Memóriamodellek

- **Címtartomány**
 - **Lineáris cím**
 - Bájtok sorszámozása 0-tól
 - Bájt címe az indexe
 - **Szegmentált cím**
 - BÁZISCÍM : ELTOLÁS (Szegmens : Offszet) alakban
 - Szegmensen belüli relatív címek használhatók
 - Könnyebb relokáció
 - A szegmens áthelyezhető a memória más részére a címzések és vezérlésátadások módosítása nélkül
 - 16 bites architektúrák esetén használatos (pl. Intel 8086)

ISA regiszterek

- **Nagyon gyors elérésű memóriarekeszek**
 - Végrehajtás vezérlésére
 - Aritmetikai, logikai műveletek elvégzésére
 - Eredmények ideiglenes tárolására
- **Mikroarchitektúra kapcsolat**
 - Nem minden mikroarchitektúra regiszter jelenik meg az ISA szinten!
 - Pl. nincs TOS, MAR, ...
 - De általában látható a PC, SP, ...
 - Az ISA szint regiszterkészlete elérhető a mikroarchitektúra szinten
 - Mivel ott kerülnek megvalósításra

ISA regiszterek

- **Speciális célú**

- Utasításszámláló, veremmutató, flag regiszter, ...
- Csak rögzített feladatra használhatók!
- Csak speciális utasításokkal módosíthatók

- **Általános célú**

- Gyakran használt adatok tárolására, gyors elérésére
- Felcserélhető regiszterek
 - Teljesen ekvivalensek
- Alkalmazási konvenciók
 - Könyvtári alkalmazásokban rögzített szerep miatt (paraméterátadás)
 - Célszerű betartani és alkalmazni
- Speciális szereppel rendelkezők
 - Pl. szorzás eredménye mindig 1 megnevezett regiszterbe kerül
 - Nem felülbírálható viselkedés

ISA regiszterek

- **Jelzők vezérlőregisztere**

- Flag vagy PSW (Program Status Word) regiszter
- Bitek, mint feltételkódok
 - minden ALU művelet beállítja őket
 - más utasítások is állíthatják
 - Elsősorban **feltételes elágazásoknál** használatosak
- Tipikus feltételkódok
 - N (negative): beállítva, ha az eredmény negatív volt
 - Z (zero): beállítva, ha az eredmény nulla volt
 - V (oVerflow): beállítva, ha volt túlcsordulás
 - C (Carry): beállítva, ha volt átvitel
 - ...

ISA utasítástípusok

- **Adatmozgató**
 - Memória és/vagy regiszterek között
 - Nem minden kombináció érhető el
 - Pl. általában legalább 1 operandus regiszter kell legyen
- **Aritmetikai és logikai**
 - Összeadás, kivonás, szorzás, osztás
 - Logikai műveletek, bitléptetés, bitforgatás
- **Összehasonlító**
 - FLAG biteket állítanak
- **Vezérlésátadó**
 - Feltételes és feltétel nélküli
 - Ugró és eljáráshívó
 - Megszakításkezelő

Rendszerek áttekintése

- **Pentium 4 és Core i7 (Intel 80x86 család)**
 - Erőteljes visszafelé kompatibilitás
 - Összetett utasításrendszer
 - Elavult részek
- **UltraSPARC III és TI OMAP4430**
 - Betöltő/tároló, RISC architektúra
 - Fix méretű utasítások
- **Intel 8051 és ATmega168 mikrovezérlők**
 - Egyetlen üzemmód, egyszerű memóriamodell, egyszerű utasítások

Intel 80x86 processzorcsalád

A táblázatot nem kell megtanulni!

Lapka	Dátum	MHz	Tranz.	Mem.	Megjegyzés
I-4004	1971/4	0.108	2300	640	Első egylapkás mikroprocesszor
I-8008	1972/4	0.108	3500	16 KB	Első 8 bites mikroprocesszor
I-8080	1974/4	2	6000	64 KB	Első általános célú mikroprocesszor
I-8086	1978/6	5-10	29000	1 MB	Első 16 bites mikroprocesszor
I-8088	1979/6	5-8	29000	1 MB	Az IBM PC processzora
I-80286	1982/6	8-12	134000	16 MB	Védett üzemmód
I-80386	1985/10	16-33	275000	4 GB	Első 32 bites mikroprocesszor, virtuális 8086 üzemmód
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
P. Pro	1995/3	150-200	5.5M	4 GB	Két szintű beépített gyorsítótár
P. II	1997/5	233-400	7.5M	4 GB	Pentium Pro + MMX
P. III	1999/2	650-1400	9.5M	4 GB	SSE utasítások 3D grafikához
P. 4	2000/11	1300-3800	42M	4 GB	Hyperthreading + több SSE
Core Duo	2006/1	1600-3200	152M	2 GB	Két mag egy lapkán
Core	2006/7	1200-3200	410M	64 GB	64 bites, négymagos architektúra
Core i7	2011/1	1100-3300	1160M	24 GB	Integrált grafikus processzor

P4 és Core i7 32 bites üzemmódjai

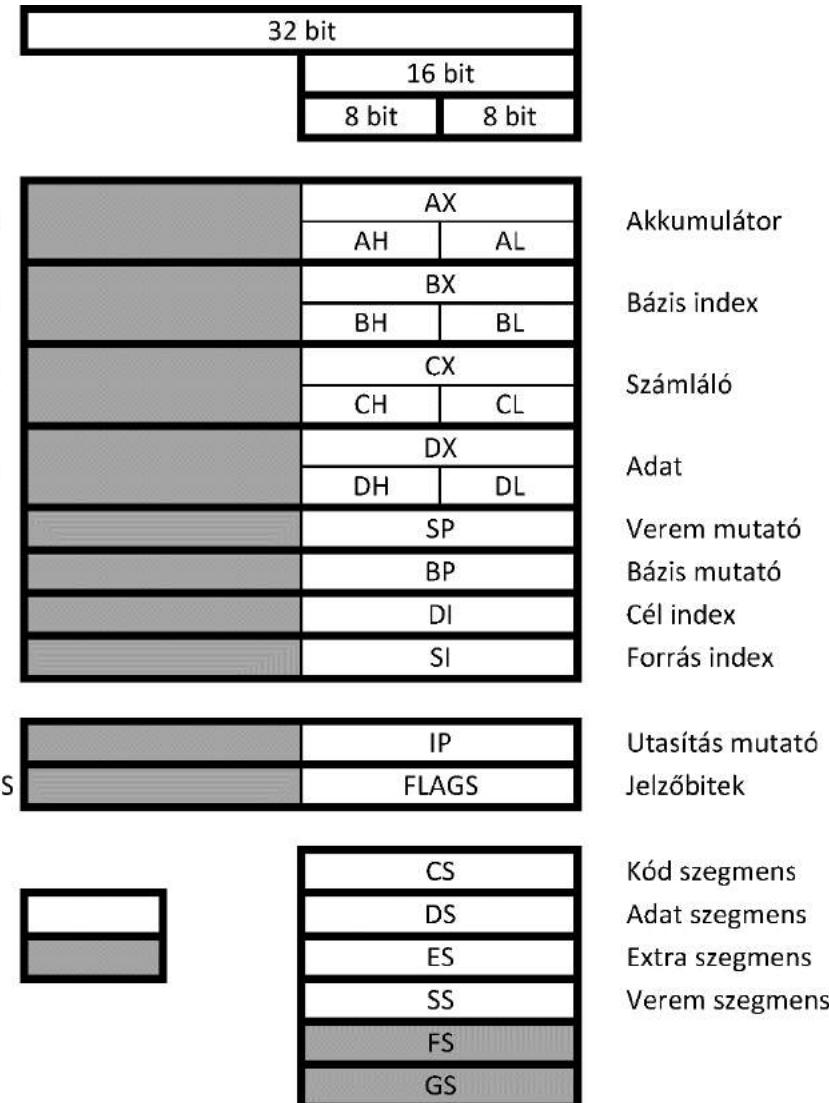
- **Valós mód**
 - 8086/8088-ként működik
 - Max. 1 MB memória címezhető
- **Védett mód**
 - A CPU lehetőségeinek teljes kihasználása
 - Négy privilegizált szint (PSW bitek vezérlik)
 - 0-s szint: kernel mód (operációs rendszer)
 - 1-es, 2-es szint: ritkán használt
 - 3-as szint: felhasználói programok
- **Virtuális 8086 mód**
 - Párhuzamosan több valós módú virtuális gép futhat
 - Védett módú környezet vezérli
 - Ha „lefagy” a virtuális gép, visszatérhetünk az indító környezethez

Core i7 64 bites üzemmódjai

Üzemmód		Szükséges OS	Alkalmazás újrafordítás	Alapbeállítások (bit)		Regiszter kiterjesztés	Ált. célú reg. méret	
				Cím	Operandus méret			
Long	64-bit	64-bites	Igen	64	32	Igen	64	
	Kompatibilitási		Nem	32	32	Nem	32	
				16	16		16	
Legacy	Védett mód	32-bites	Nem	32	32	Nem	32	
	Virtuális 8086			16	16		16	
	Valós mód	16-bites		16	16			

P4 és Core i7 32 bites regiszterei

- **Általános célú regiszterek**
 - 32 bitesek
 - EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
 - Alsó 16 bit, és annak alsó/felső bájtja külön elérhető
 - EAX; AX; AH; AL
- **Speciális célú regiszterek**
 - 32 bitesek
 - EIP, EFLAGS (PSW)
- **Szegmens regiszterek**
 - 16 bitesek
 - CS, SS, DS, ES, FS, GS



P4 és Core i7 32 bites regiszterei

• Általános célú regiszterek

- EAX: Akkumulátor
Egyedi szerep szorzásnál és osztásnál
- EBX: Bázis index
- ECX: Számláló
Egyedi szerep bitléptető, sztring és ismétléses műveleteknél
- EDX: Adat
Egyedi szerep szorzásnál és osztásnál
- EBP: Bázis mutató
Egyedi szerep verem indexelt címzésére
- ESI: Forrás index
Egyedi szerep sztring műveleteknél
- EDI: Cél index
Egyedi szerep sztring műveleteknél
- ESP: Veremmutató
Verem műveletek állítják. Erős speciális jellege miatt általános célra nem célszerű használni.

P4 és Core i7 32 bites regiszterei

- **Speciális célú regiszterek**

- EIP: Utasítás számláló (Csak vezérlésátadással írható felül)
- EFLAGS: CPU állapotát jelzőbitek összessége
 - C (Carry): Átvitel előjel nélküli műveleteknél
 - P (Parity): Az eredmény alsó 8 bitjének paritása
 - A (Aux Carry): Átvitel a 3. és 4. bitek között (BCD számoknál)
 - Z (Zero): 1 (igaz), ha az eredmény 0, különben 0 (hamis)
 - S (Sign): Az eredmény legmagasabb helyiértékű bitje (előjel)
 - T (Trap): 1: debug mód, 0: automatikus
 - I (Interrupt): 1: maszkolható megszakítás engedélyezve, 0: tiltva
 - D (Direction): Sztring műveletek iránya 1: csökkenő, 0: növekvő
 - O (Overflow): Előjeles túlcsordulás

P4 és Core i7 32 bites regiszterei

• Szegmens regiszterek

- CS (Code Segment): kód szegmens
 - Csak ugró utasítással módosítható
- DS (Data Segment): adat szegmens
- ES (Extra Segment): (másodlagos) adat szegmens
- SS (Stack Segment): verem szegmens
- FS, GS: segéd szegmens regiszterek

• Használatuk

- **Védett üzemmódban** (ez a jellemző)
 - Szelektor index a szegmens leíró táblára
 - Tábla tartalma: szegmens fizikai kezdőcíme, mérete, hozzáférési jogosultságok
 - Szegmensen belül 32 bites címtartomány
 - Több operációs rendszer csak 1 lineáris címtartományt engedélyez
 - UNIX, Windows, ...

P4 és Core i7 32 bites regiszterei

- **Szegmensregiszterek használata** Nem kell megtanulni!
- **16 bites 8086 üzemmódban (elavult!)**
 - 20 bites címbusz, de csak 16 bites regiszterek
 - 1 regiszter nem elegendő a címzéshez, használunk kettőt
 - SZEGMENS : OFFSZET alakban írjuk
 - Fizikai cím előállítása: a SZEGMENS értéke 16-tal (10H) szorzódik / 4 bittel balra tolódik / hexa 0 íródik utána és hozzáadódik az OFFSZET értéke
 - Vagyis
 - Egy szegmens mérete maximálisan 64 KB lehet
 - Egy fizikai címnek több szegmentált címe létezik!
 - Túlcordulás kezelés nincs
 - FFFF:0010 az első (0.) memóriapozíciót jelenti!
 - Az egyes szegmensek részben vagy teljesen átfedőek lehetnek!
- **Kis endián tárolás**

SZEGMENS :	1234H
OFFSZET :	0012H

12340H	* 10H
+ 0012H	+ OF.

12352H	

Core i7 64 bites regiszterei

Általános célú regisztrek (GPR-ek)		64 bites média és lebegőpontos reg.		128 bites média regiszterek	
EAX	RAX		MMX0/FPRO		XMM0
EBX	RBX		MMX1/FPR1		XMM1
ECX	RCX		MMX2/FPR2		XMM2
EDX	RDX		MMX3/FPR3		XMM3
EBP	RBP		MMX4/FPR4		XMM4
ESI	RSI		MMX5/FPR5		XMM5
EDI	RDI		MMX6/FPR6		XMM6
ESP	RSP		MMX7/FPR7		XMM7
R8	63	0			XMM8
R9					XMM9
R10	0 EFLAGS		RFLAGS		XMM10
R11	63	0			XMM11
R12					XMM12
R13	EIP		RIP		XMM13
R14	63	0			XMM14
R15					XMM15
63	0				0
Örökölt x86 regiszterek, minden üzemmódban elérhetők		E = „Extended”			
64-bites módban támogatott, kiterjesztett regiszterek		R = „Really extended”			

E = „Extended”
R = „Really extended”

UltraSPARC III ISA-szintje

- **Jellemzői**

- Az egyik első RISC architektúra leszármazottja (SPARC 1987)
- 64 bites működés

- **Memóriaszerkezet**

- 2^{64} bajt méretű lineáris tömb, jelenleg 44 bites címezhető tartomány
- Nagy endián bájtsorrend
 - PSW megfelelő bitjével kis endiánra is állítható

- **Regiszterek**

- 32 darab 64 bites általános célú regiszter (R0 – R31), konvenciók:
 - R0 (G0): hardveres 0 érték, minden tárolás eredménytelen
 - R1-R7 (G1-G7): Globális változók
 - R8-R13 (O0-O5): Hívott eljárás paraméterei
 - R14 (SP): Veremmutató
 - R15 (O7): Ideiglenes regiszter
 - R16-R23 (L0-L7): Aktuális eljárás lokális változói
 - R24-R29 (I0-I5): Bejövő paraméterek
 - R30 (FP): Aktuális veremkeret-mutató
 - R31 (I7): Aktuális eljárás visszatérési címe
- 32 darab lebegőpontos regiszter

UltraSPARC III

- **Regiszterablak**

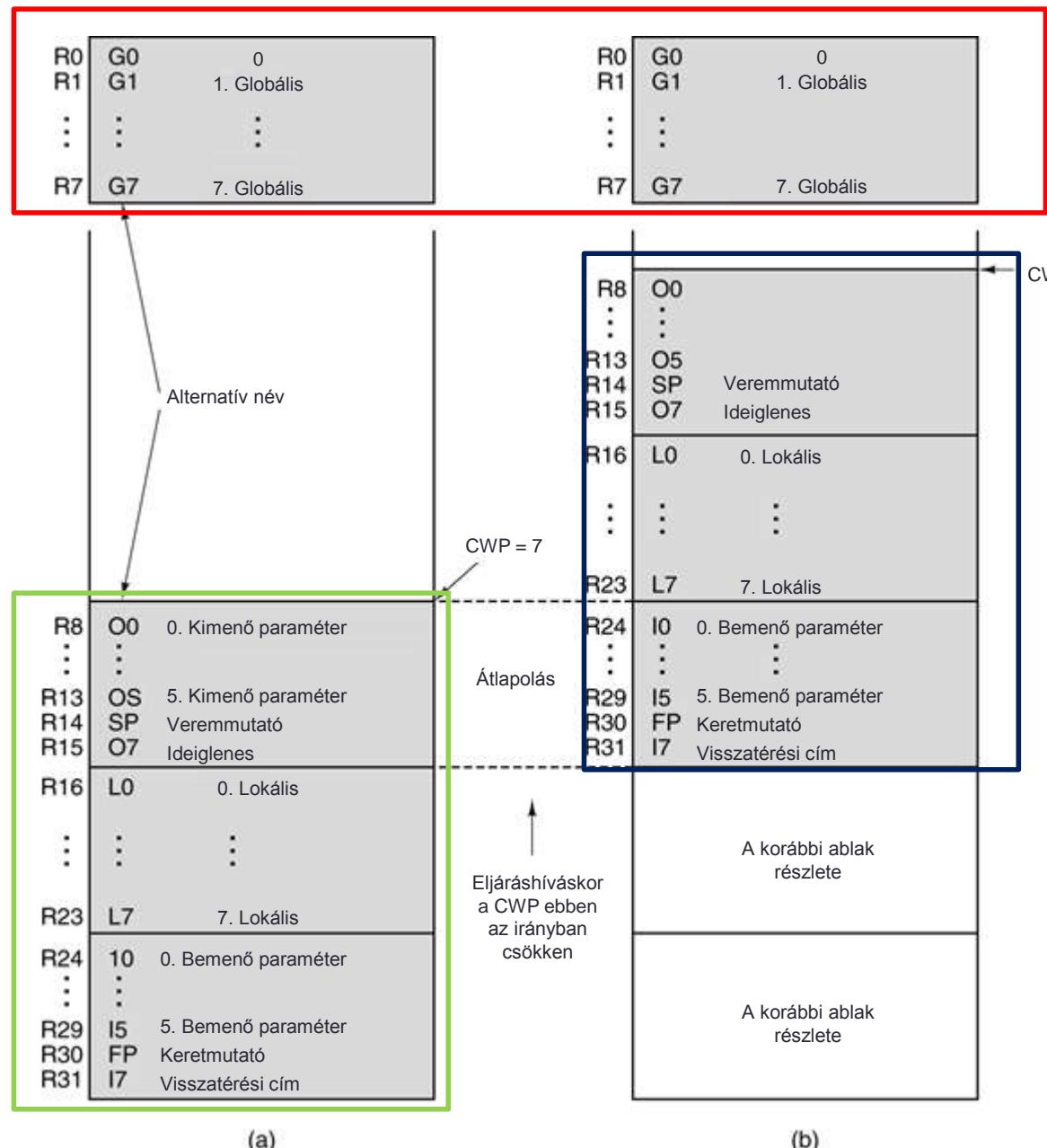
- Ténylegesen 32-nél több regiszter van
- De csak 32 látszik egyszerre
- CWP regiszter: aktuálisan mely regiszterek használhatók
 - Eljáráshíváskor veremszerű működés: új készlet nyílik
 - A globális regiszterek az új készletben is megtalálhatók lesznek

- **Probléma**

- Eljáráshívások egymásba ágyazásával elfogynak a regiszterek
- Ekkor a betelt készletet ki kell írni a memóriába
- Eljárásból visszatéréskor vissza kell olvasni
- Csak nem túl mély egymásba ágyazottság esetén hatékony

Minden ablakban látható

Hívás előtti ablak



UltraSPARC III

- **Betöltő/tároló architektúra**

- Memória írás/olvasás csak regiszterből/regiszterbe történhet
- A többi utasítás csak regiszter vagy beépített adat operandussal rendelkezhet
- Az eredmény mindenkorán regiszterbe kerül

TI OMAP4430 (ARMv7) ISA-szintje

- **Jellemzői**

- Mobil eszközökben (okostelefonok)
- ARM Cortex A9 processzor
- ARMv7 ISA
- 32 bites, RISC
- Tiszta ISA
 - Betöltő-tároló architektúra

- **Memóriaszerkezet**

- 2^{32} bájt méretű lineáris tömb a címtartomány
 - Kezd kevés lenni még mobil eszközökben is...
 - (ARMv8: már 64 bites architektúra)
- Kétféle endián üzemmód lehetőség
 - Rendszerindításkor a rendszer memória blokk tartalma határozza meg
 - Ez az információ mindenkorban kis-endián módban tárolódik

TI OMAP4430 (ARMv7)

- **Regiszterek**

- Program Status Register (PSR): jelzőbitek
- Általános célú: 16 darab 32 bites regiszter

Regiszter	Alternatív név	Funkció
R0 – R3	A1 – A4	Eljáráshívás paraméterei
R4 – R11	V1 – V8	Aktuális eljárás lokális változói
R12	IP	Eljárások közötti hívás regisztere: Ha $\pm 2^{25}$ bájt távolság nál messzebbre kell ugrani, ez tartalmazza a címet (32 bites utasításhosszba nem fér bele 32 bites cím az opkód mellé...)
R13	SP	Veremmutató
R14	LR	Link regiszter (aktuális függvény visszatérési címe)
R15	PC	Programszámláló (aritmetikai művelet operandusa is lehet!)

- Lebegőpontos: 32 darab 32 bites regiszter
 - 16 darab 64 bitesként is elérhetők! (Egyeszeres és dupla pontosság)

Intel 8051 ISA-szintje

- **Beágyazott rendszerekben**
 - Egyetlen üzemmód, nincs védelmi hardver
 - Egyszerre 1 program fut
- **Memóriamodell**
 - 64 KB címtartomány programnak (ROM)
 - 64 KB címtartomány adatoknak (RAM)
- **Lehetséges megvalósítások**
 - Többféle lehet
 - 4 KB ROM, 128 bájt (!) RAM, lapkára integrálva (gyors elérés!)
 - 64 KB külső ROM, 64 KB külső RAM (lassabb, de nagyobb)
 - 64 KB külső RAM a programnak és az adatoknak
 - ...

Intel 8051

- **Általános regiszterkészlet**

- 4 elkülönülő regiszterkészlet
- Készletenként 8 darab 8 bites regiszter (R0 – R7)
 - PSW 2 bites mezője választja ki az aktuális készletet
- Gyors megszakításkezeléshez hasznos!
 - Elegendő a készletváltás, nem kell menteni és visszatölteni az értékeket

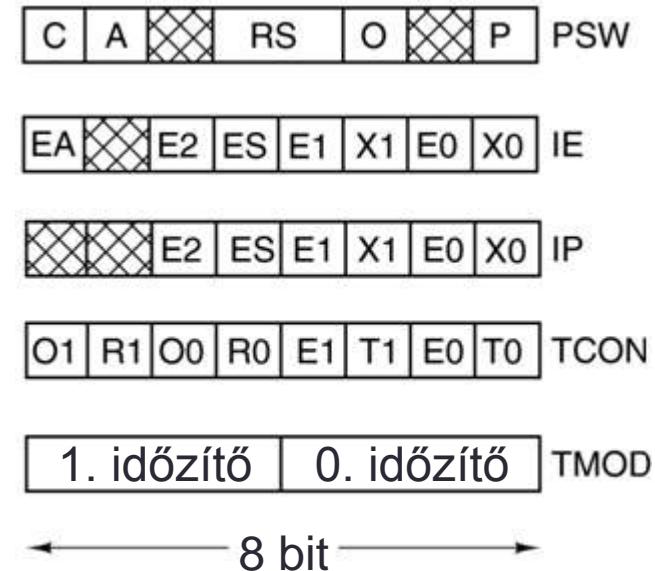
- **Memoriába ágyazottak**

- Memória első $8 \times 4 = 32$ bájtja
 - Memória 0. bájtja: 0. készlet R0 regisztere
 - Memória írása/olvasása a regiszter írással/olvasással egyezik meg!
- Memória 32-47 című bájtai: bit-címezhető memóriamező
 - 0 – 127 bitcím-tartomány
 - Logikai értékek optimális reprezentációja

Intel 8051

- Speciális regiszterek

- PSW
 - C, A, O, P: mint korábban
 - RS: regiszterkészlet választó
- IE
 - Megszakítások kollektív (EA) és egyedi (E.,X.) engedélyezése és tiltása
- IP
 - Megszakítások prioritási sorrendje
 - Alacsony/magas: alacsony megszakítható magassal, fordítva nem
- TCON
 - Fő időzítők kezelése
- TMOD
 - Időzítők üzemmódja



Forrás: Tanenbaum

Nem kell megtanulni!

Intel 8051

- **További regiszterek**
 - 128-255 címtartományban
 - Akkumulátor: aritmetikai műveletekhez (240-es címen)
 - B/K portok
 - Áramellátással, soros port vezérléssel kapcsolatos regiszterek

AVR ISA

- **Jellemzői**
 - Alacsony számítási igényű és energiafelhasználású beágyazott rendszerekben
 - ATmega48, ATmega88, ATmega168, ...
 - Egyfélé üzemmód, nincs védett mód, egy program fut
- **Memóriamodell (ATmega168)**
 - 16 KB program memória (Flash memória)
 - Rendszerbetöltő szekció
 - Csak innen futó kód módosíthatja a Flash memóriát (írhat be programot)
 - A gyártó digitális aláírással láthatja el a kódot
 - Alkalmazás szekció
 - 1 KB adat memória (SRAM memória)
 - Különálló címtartománnyal!

AVR ISA

- **Regiszterek**

- 32 darab 8 bites általános célú regiszter (R0 – R31)
 - Memoriába ágyazottak
 - 0. adatbájt R0 regiszterrel egyenértékű, ...
- Speciális célú regiszterek
 - Státusz regiszter (SREG): jelzőbitek
 - Megszakítás engedélyezés; fél-átvitel; előjel; túlcsordulás; negatív; nulla; átvitel
 - Verem mutató (SP): 16 bites; 80 – 81. memóriacímen

- **Adat memória**

- 0 – 31 bájtok: R0 – R31 regiszterek
- 32 – 95 bájtok: 64 darab B/K eszköz regiszter
- 96 – 1023: adatterület

ISA-szint adattípusai

- **Hardver mit támogat**
 - **Numerikus**
 - Egész, lebegőpontos
 - Többféle kódolással (BCD, kettes komplement, IEEE 754, ...)
 - **Nem numerikus**
 - Logikai értékek
 - 0 (hamis) és 1 (igaz)
 - Jellemzően egész bájton tárolva
 - Karakterláncok (sztringek)
 - ASCII, UTF, ...
- **Szoftveres megvalósítás**
 - Ha nincs hardver támogatás
 - Komplex számok, sztringként leírt hosszú egész és tört számok, ...
 - Szabadon választható reprezentáció

ISA-szint adattípusai

Típus	1 bit	8 bit	16 bit	32 bit	64 bit	128 bit
Bit	8					
Előjeles egész		P,C,U,A,8,M	P,C,U,A	P,C,U,A	C,U	
Előjel nélküli egész		P,C,U,A,M	P,C,U,A,M	P,C,U,A	C,U	
BCD		P,C				
Lebegőpontos				P,C,U,A	P,C,U,A	U

- **Jelölés** A táblázatot nem kell megtanulni!
 - P (Pentium 4), C (Core i7), U (UltraSPARC III), 8 (8051), A (ARMv7), M (ATmega168)
- **Előjeles egész**
 - Kettes komplement ábrázolás
- **Lebegőpontos**
 - IEEE 754 szabvány szerint
- **BCD**
 - Csak az Intel processzorok esetén támogatott

ISA-szint utasításformátumai

- **Utasítás tartalma**

- Műveleti kód: mit kell csinálni
- Operandusok: min kell elvégezni
 - Memóriacím, regiszter(ek), kódba épített adat
- Prefix: utasítás hatásának esetleges módosítására
 - Sztringművelet ismétlésére, sín zárolására, ...

- **Utasítások hossza**

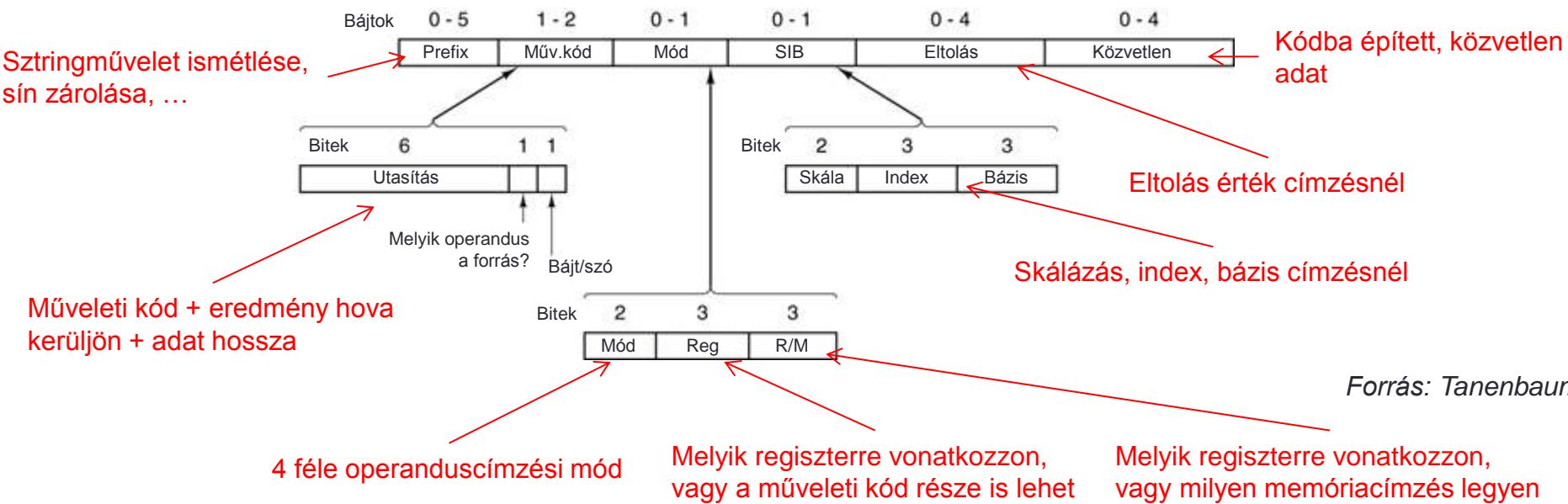
- Egyező: rögtön ismert a következő utasítás címe (kivéve ugrások esetén), de pazarlóbb a memóriával
- Változó: lassabb dekódolás
- Műveleti kód kiterjesztése
 - Változó számú bit írja le a kódot és az operandusokat
 - De egyértelműen dekódolhatónak kell lennie

P4 és Core i7 utasításformátumai

- **Jellemzői**

- Bonyolult, szabálytalan: 1-17 bájt közötti méret
- Hatfélé, változó hosszúságú mező, több opcionális
- Régi ISA „foltozása” a kompatibilitás miatt
- Sok, komplex utasítás
- Nem lehet minden operandus memóriahivatkozás, legalább 1 regiszter kell

Az ábrát nem kell megtanulni!



UltraSPARC III utasításformátumai

- **Jellemzői**

- minden utasítás 32 bites és szóhatárra igazított
- Aritmetikai művelet: két bemenő regiszter, egy eredmény; vagy egyik regiszter helyett 13 bites előjeles konstans
- Kevesebb, egyszerűbb művelet

Az ábrát nem kell megtanulni!

Format	2	5	6	5	1	8	5	3 Register	Három regiszter
1a		DEST	OPCODE	SRC1	0	FP-OP	SRC2		
1b		DEST	OPCODE	SRC1	1	IMMEDIATE CONSTANT		Immediate	-4096 és 4095 közötti érték
2	2	5	3		22			SETHI	A SETHI utasításnak 22 bites operandus kell
3	2	1	4	3		22		BRANCH	Feltételes elágazás
4	2			30				CALL	Eljárás hívása: a 30 bites érték négyzere a cél

Forrás: Tanenbaum

TI OMAP4430 utasításformátumai

- **Jellemzői**

- 16 és 32 bit hosszúságú, egyszerű utasítások
 - Pl. aritmetikai: két forrásregiszter, 1 célregiszter
 - 16 bites esetben „Thumb ISA”: csak két regiszter adható meg, bemenetként csak az első 8 regiszter adható meg
- Memóriában igazított helyen találhatók
- Predikált (feltételes) utasítások
 - Eredmény csak a feltétel teljesülése esetén tárolódik
- Ugrás operandusa csak 24 bites!
 - Csak négygyel osztható címre ugorhatunk → $\pm 2^{25}$ távolság érhető el
 - Nagyobb távolság esetén IP regisztert kell használni

TI OMAP4430 utasításformátumai

- 32 bites utasításformátumok

A táblázatot nem kell megtanulni!

31	2827	1615	87	0	Instruction type
Cond	0 0 I	Opcode S	Rn	Rd	Data processing / PSR Transfer
Cond	0 0 0 0 0 0	A S	Rd	Rn	Multiply
Cond	0 0 0 0 1	U A S	RdHi	RdLo	Long Multiply
Cond	0 0 0 1 0	B 0 0	Rn	Rd	0 0 0 0 1 0 0 1 Rm Swap
Cond	0 1 P U B W L		Rn	Rd	Offset Load/Store Byte/Word
Cond	1 0 0 P U S W L		Rn		Register List Load/Store Multiple
Cond	0 0 0 P U 1 W L		Rn	Rd	Offset1 1 S H 1 Offset2 Halfword transfer: Immediate offset
Cond	0 0 0 P U 0 W L		Rn	Rd	0 0 0 0 1 S H 1 Rm Halfword transfer: Register offset
Cond	1 0 1 L				Offset Branch
Cond	0 0 0 1	0 0 1 0	1 1 1 1	1 1 1 1	1 1 1 1 0 0 0 1 Rn Branch Exchange
Cond	1 1 0 P U N W L		Rn	CRd	CPNum Offset Coprocessor data transfer
Cond	1 1 1 0	Op1	CRn	CRd	CPNum Op2 0 CRm Coprocessor data operation
Cond	1 1 1 0	Op1 L	CRn	Rd	CPNum Op2 1 CRm Coprocessor register transfer
Cond	1 1 1 1				SWI Number Software interrupt

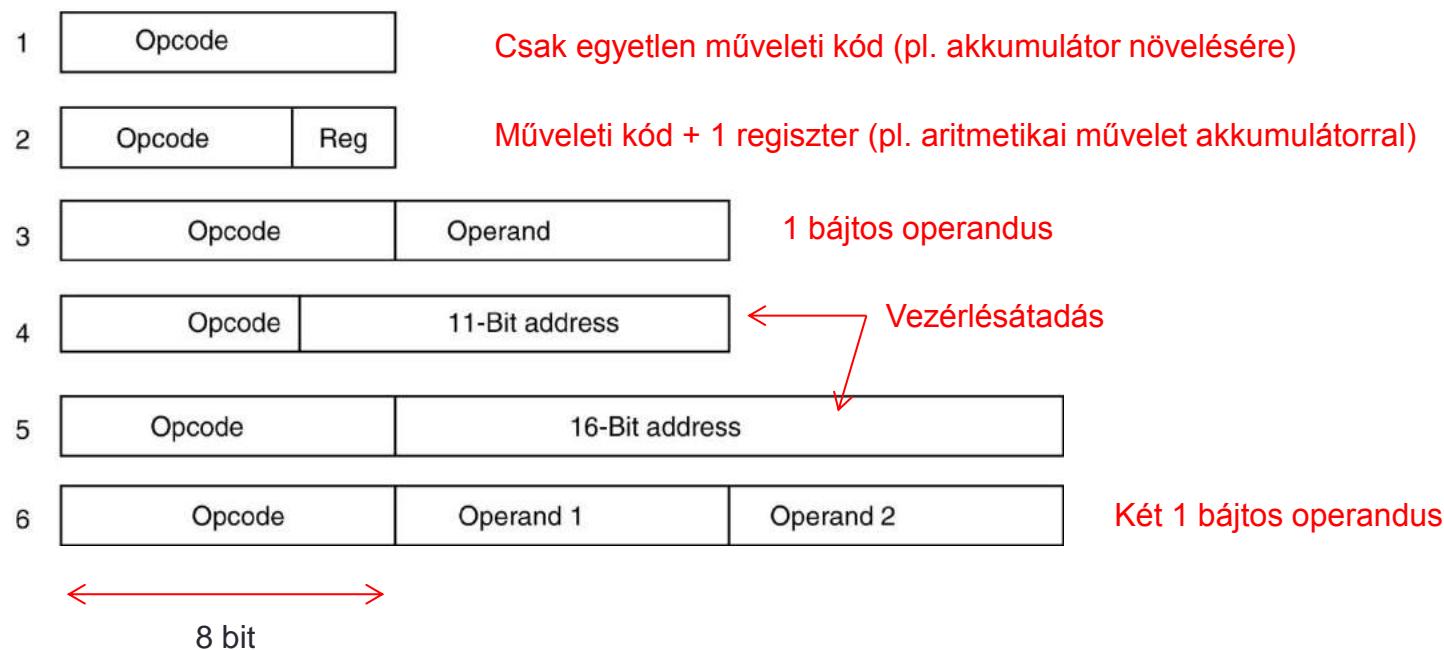
Intel 8051 utasításformátumai

- **Jellemzői**

- Hatfélé egyszerű utasításforma

Format

Az ábrát nem kell megtanulni!



Forrás: Tanenbaum

ATmega168 AVR utasításformátumai

- **6 utasításformátum; 5 darab 16 bites és 1 darab 32 bites**
 - 1.: opkód + két regiszter
 - eredmény az egyik regiszterbe
 - 2.: opkód + regiszter
 - eredmény ugyanebbe a regiszterbe (negálás, inkrementálás)
 - 3.: opkód + 8 bites közvetlen (utasításba épített) adat + 1 regiszter
 - R16 – R31 közötti regiszter lehet
 - Csak 4 utasítás használhatja (2 bit marad opkód azonosításra)
 - 4.: opkód + 6 bites előjel nélküli közvetlen operandus
 - Betöltő/tároló utasítások; opkód határozza meg a használandó regisztert
 - 5.: opkód + 12 bites közvetlen eltolási érték
 - Ugró utasítás, PC relatív érték
 - 6.: opkód + 22 bites közvetlen eltolási érték
 - Ugró utasítás, PC relatív érték

Címzési módok

- **Feldolgozandó adat helyének megadása**
 - Operandus megadása közvetlenül az utasításban
 - Memóriabeli operandus teljes címének megadásával
 - Az operandus egy regiszter
 - ...
- Részletesebb tárgyalás az Assembly programozás kurzuson!

Vezérlési folyamat

- **Utasítások végrehajtási sorrendje**
 - **Szekvenciális vezérlés és elágazás**
 - Egymást követő utasítások végrehajtása
 - **Eljárások**
 - Programok strukturáltságához fontos
 - Eljárásprológus
 - Eljárás híváskor lejátszódó folyamat
 - Pl. veremkeret mentése, visszatérési cím a verembe, ...
 - Eljárássepilógus
 - Eljárásból való visszatréskor lejátszódó folyamat
 - visszatérési cím beállítása, ...
 - **Rekurzív eljárás**
 - Önmagát hívja
 - Verem szükséges a megvalósításához

Vezérlési folyamat

- **Utasítások végrehajtási sorrendje**

- **Csapdák**

- Program által előidézett fontos, de csak ritkán bekövetkező események
- Egész túlcsordulás, 0-val való osztás, lebegőpontos alul- vagy túlcsordulás
- Bekövetkeztekor a csapdakezelőre adódik a vezérlés
 - Nem kell minden utasítás után a hibalehetőségeket ellenőrizni

- **Megszakítások**

- B/K eszköz általi vezérlési folyamat módosítás
- Megszakításkezelő
 - Bekövetkeztekor ide adódik a vezérlés, végeztével a program fut tovább
- Átlátszóság
 - A számítógép ugyanabba az állapotba kerül a megszakításkezelő után, mint előtte volt
- Megszakítható megszakítások
 - Prioritási szintek bevezetése

Pentium 4 utasításai

Adatmozgatás	
MOV DST , SRC	SRC másolása DST-be
PUSH SRC	SRC verembe helyezése
POP DST	Verem teteje DST-be
XCHG DS1 ,DS2	DS1 és DS2 felcserélése
LEA DST , SRC	SRC offszetcíme DST-be
CMOVcc DST , SRC	Feltételes adatmozgatás

Csak érdekesség,
nem kell megtanulni!

Aritmetikai	
ADD DST , SRC	SRC hozzáadása DST-hez
SUB DST , SRC	SRC kivonása DST-ből
MUL SRC	EAX előjel nélküli szorzása SRC-vel
IMUL SRC	EAX előjeles szorzása SRC-vel
DIV SRC	EDX:EAX előjel nélküli osztása SRC-vel
IDIV SRC	EDX:EAX előjeles osztása SRC-vel
ADC DST , SRC	SRC és carry hozzáadása DST-hez
SBB DST , SRC	SRC és carry kivonása DST-ből
INC DST	DST 1-el növelése
DEC DST	DST 1-el csökkentése
NEG DST	DST negálása

Pentium 4 utasításai

Binary coded decimal

DAA	Decimal adjust
DAS	Decimal adjust for subtraction
AAA	ASCII adjust for addition
AAS	ASCII adjust for subtraction
AAM	ASCII adjust for multiplication
AAD	ASCII adjust for division

Boolean

AND DST,SRC	Boolean AND SRC into DST
OR DST,SRC	Boolean OR SRC into DST
XOR DST,SRC	Boolean Exclusive OR SRC to DST
NOT DST	Replace DST with 1's complement

Shift/rotate

SAL/SAR DST,#	Shift DST left/right # bits
SHL/SHR DST,#	Logical shift DST left/right # bits
ROL/ROR DST,#	Rotate DST left/right # bits
RCL/RCR DST,#	Rotate DST through carry # bits

Test/compare

TEST SRC1,SRC2	Boolean AND operands, set flags
CMP SRC1,SRC2	Set flags based on SRC1 - SRC2

Transfer of control

JMP ADDR	Jump to ADDR
Jxx ADDR	Conditional jumps based on flags
CALL ADDR	Call procedure at ADDR
RET	Return from procedure
IRET	Return from interrupt
LOOPxx	Loop until condition met
INT n	Initiate a software interrupt
INTO	Interrupt if overflow bit is set

Strings

LODS	Load string
STOS	Store string
MOVS	Move string
CMPS	Compare two strings
SCAS	Scan Strings

Csak érdekesség, nem kell megtanulni!

Pentium 4 utasításai

Condition codes

STC	Set carry bit in EFLAGS register
CLC	Clear carry bit in EFLAGS register
CMC	Complement carry bit in EFLAGS
STD	Set direction bit in EFLAGS register
CLD	Clear direction bit in EFLAGS reg
STI	Set interrupt bit in EFLAGS register
CLI	Clear interrupt bit in EFLAGS reg
PUSHFD	Push EFLAGS register onto stack
POPFD	Pop EFLAGS register from stack
LAHF	Load AH from EFLAGS register
SAHF	Store AH in EFLAGS register

Miscellaneous

SWAP DST	Change endianness of DST
CWQ	Extend EAX to EDX:EAX for division
CWDE	Extend 16-bit number in AX to EAX
ENTER SIZE,LV	Create stack frame with SIZE bytes
LEAVE	Undo stack frame built by ENTER
NOP	No operation
HLT	Halt
IN AL,PORT	Input a byte from PORT to AL
OUT PORT,AL	Output a byte from AL to PORT
WAIT	Wait for an interrupt

SRC = source

DST = destination

= shift/rotate count

LV = # locals

UltraSPARC III utasításai

Loads

LDSB ADDR,DST	Load signed byte (8 bits)
LDUB ADDR,DST	Load unsigned byte (8 bits)
LDSH ADDR,DST	Load signed halfword (16 bits)
LDUH ADDR,DST	Load unsigned halfword (16)
LDSW ADDR,DST	Load signed word (32 bits)
LDUW ADDR,DST	Load unsigned word (32 bits)
LDX ADDR,DST	Load extended (64-bits)

Stores

STB SRC,ADDR	Store byte (8 bits)
STH SRC,ADDR	Store halfword (16 bits)
STW SRC,ADDR	Store word (32 bits)
STX SRC,ADDR	Store extended (64 bits)

Arithmetic

ADD R1,S2,DST	Add
ADDCC " "	Add and set icc
ADDC " "	Add with carry
ADDCCC " "	Add with carry and set icc
SUB R1,S2,DST	Subtract
SUBCC " "	Subtract and set icc
SUBC " "	Subtract with carry
SUBCCC " "	Subtract with carry and set icc
MULX R1,S2,DST	Multiply
SDIVX R1,S2,DST	Signed divide
UDIVX R1,S2,DST	Unsigned divide
TADCC R1,S2,DST	Tagged add

Csak érdekesség, nem kell megtanulni!

UltraSPARC III utasításai

Shifts/rotates

SLL R1,S2,DST	Shift left logical (32 bits)
SLLEX R1,S2,DST	Shift left logical extended (64)
SRL R1,S2,DST	Shift right logical (32 bits)
SRLX R1,S2,DST	Shift right logical extended (64)
SRA R1,S2,DST	Shift right arithmetic (32 bits)
SRAX R1,S2,DST	Shift right arithmetic ext. (64)

Miscellaneous

SETHI CON,DST	Set bits 10 to 31
MOVcc CC,S2,DST	Move on condition
MOVr R1,S2,DST	Move on register
NOP	No operation
POPC S1,DST	Population count
RDCCR V,DST	Read condition code register
WRCCR R1,S2,V	Write condition code register
RDPC V,DST	Read program counter

Boolean

AND R1,S2,DST	Boolean AND
ANDCC “	Boolean AND and set icc
ANDN “	Boolean NAND
ANDNCC “	Boolean NAND and set icc
OR R1,S2,DST	Boolean OR
ORCC “	Boolean OR and set icc
ORN “	Boolean NOR
ORNCC “	Boolean NOR and set icc
XOR R1,S2,DST	Boolean XOR
XORCC “	Boolean XOR and set icc
XNOR “	Boolean EXCLUSIVE NOR
XNORCC “	Boolean EXCL. NOR and set icc

Csak érdekesség, nem kell megtanulni!

UltraSPARC III utasításai

Transfer of control

BPcc ADDR	Branch with prediction
BPr SRC,ADDR	Branch on register
CALL ADDR	Call procedure
RETURN ADDR	Return from procedure
JMPL ADDR,DST	Jump and link
SAVE R1,S2,DST	Advance register windows
RESTORE “	Restore register windows
Tcc CC,TRAP#	Trap on condition
PREFETCH FCN	Prefetch data from memory
LDSTUB ADDR,R	Atomic load/store
MEMBAR MASK	Memory barrier

SRC = source register

DST = destination register

R1 = source register

S2 = source: register or immediate

ADDR = memory address

TRAP# = trap number

FCN = function code

MASK = operation type

CON = constant

V = register designator

CC = condition code set

R = destination register

cc = condition

r = LZ,LEZ,Z,NZ,GZ,GEZ

UltraSPARC III utasításai

Utasítás	Megoldás
MOV SRC , DST	OR SRC G0-al, tárolás DST-ben
CMP SRC1 , SRC2	SRC2 kivonása (SUBCC) SRC1-ből, tárolás G0-ban
TST SRC	ORCC SRC és G0 között, eredmény G0-ba
NOT DST	XNOR DST és G0 között
NEG DST	DST kivonása G0-ból, eredmény DST-be
INC DST	DST-hez 1 hozzáadása (ADD)
DEC DST	DST-ből 1 kivonása (SUB)
CLR DST	G0 vagyolása G0-al, eredmény DST-be
NOP	G0 felső bitjeinek nullázása (SETHI)
RET	JMPL %I7+8,%G0

Csak érdekesség, nem kell megtanulni!

8051 utasításai

Inst.	Description	ACC	Reg	Dir	@R	#	C	Bit
MOV	Move <i>src</i> to ACC		X	X	X	X		
MOV	Move <i>src</i> to register	X		X		X		
MOV	Move <i>src</i> to memory	X	X	X	X	X		
MOV	Move <i>src</i> to indirect RAM	X		X		X		
MOV	Move 16-bit constant to DPTR							
MOVC	Move code to ACC offset from DPTR							
MOVC	Move code to ACC offset from PC							
MOVX	Move external RAM byte to ACC				X			
MOVX	Move ext. RAM byte to ACC @DPTR							
MOVX	Move to ext. RAM byte from ACC				X			
MOVX	Move to ext. RAM byte from ACC @DPTR							
PUSH	Push <i>src</i> byte to stack				X			
POP	Pop stack byte to dst				X			
XCH	Exchange ACC and dst	X		X	X			
XCHD	Exchange low-order digit ACC and dst			X				
SWAP	Swap nibbles of dst	X						
ADD	Add <i>src</i> to ACC		X	X	X	X		

Csak érdekesség, nem kell megtanulni!

8051 utasításai

Inst.	Description	ACC	Reg	Dir	@R	#	C	Bit
ADDC	Add <i>src</i> to ACC with carry		x	x	x	x		
SUBB	Subtract <i>src</i> from ACC with borrow		x	x	x	x		
INC	Increment dst	x	x	x	x			
DEC	Decrement dst	x	x	x	x			
INC	DPTR							
MUL	Multiply							
DIV	Divide							
DA	Decimal adjust dst	x						
ANL	AND <i>src</i> to ACC		x	x	x	x		
ANL	AND ACC to dst			x				
ANL	AND immediate to dst			x				
ORL	OR <i>src</i> to ACC		x	x	x	x		
ORL	OR ACC to dst			x				
ORL	OR immediate to dst			x				

Csak érdekesség, nem kell megtanulni!

8051 utasításai

Inst.	Description	ACC	Reg	Dir	@R	#	C	Bit
XRL	XOR src to ACC		x	x	x	x		
XRL	XOR ACC to dst			x				
XRL	XOR immediate to dst			x				
CLR	Clear dst	x						
CPL	Complement dst	x						
RL	Rotate dst left	x						
RLC	Rotate dst left through carry	x						
RR	Rotate dst right	x						
RRC	Rotate dst right through carry	x						

Csak érdekesség, nem kell megtanulni!

8051 utasításai

Inst.	Description	ACC	Reg	Dir	@R	#	C	Bit
CLR	Clear bit						×	×
SETB	Set bit						×	×
CPL	Complement bit						×	×
ANL	AND <i>src</i> to carry							×
ANL	AND complement of <i>src</i> to carry							×
ORL	OR <i>src</i> to carry							×
ORL	OR complement of <i>src</i> to carry							×
MOV	Move <i>src</i> to carry							×
MOV	Move carry to <i>src</i>							×
JV	Jump relative if carry set							
JNC	Jump relative if carry not set							
JB	Jump relative if direct bit set							×
JNB	Jump relative if direct bit not set							×
JBC	Jump rel. if direct bit set and carry clear							×

Csak érdekesség, nem kell megtanulni!

8051 utasításai

Inst.	Description	ACC	Reg	Dir	@R	#	C	Bit
ACALL	Call subroutine (11-bit addr)							
LCALL	Call subroutine (16-bit addr)							
RET	Return from subroutine							
RETI	Return from interrupt							
SJMP	Short relative jump (8-bit addr)							
AJMP	Absolute jump (11-bit addr)							
LJMP	Absolute jump (16-bit addr)							
JMP	Jump indirect rel. to DPR+ACC							
JZ	Jump if ACC is zero							
JNZ	Jump if ACC is nonzero							
CJNE	Comp. <i>src</i> to ACC, jump unequal				x		x	
CJNE	Comp. <i>src</i> to immediate, jump unequal		x		x			
DJNZ	Decrement dst and jump nonzero							
NOP	No operation							

Csak érdekesség, nem kell megtanulni!

Assembly példaprogramok

- $N = I + J$ megvalósítása
- Pentium 4

Címke	Műv. kód	Operandusok ; Megjegyzések
	FORMULA: MOV	EAX,I ; EAX regiszter = I
	ADD	EAX,J ; EAX regiszter = I + J
	MOV	N,EAX ; N = I + J
I	DD	3 ; 4 bájtos terület, értéke 3
J	DD	4 ; 4 bájtos terület, értéke 4
N	DD	0 ; 4 bájtos terület, értéke 0

- Motorola 680x0

Címke	Műv. kód	Operandusok ; Megjegyzések
	FORMULA MOVE.L	I,D0 ; D0 regiszter = I
	ADD.L	J,D0 ; D0 regiszter = I + J
	MOVE.L	D0,N ; N = I + J
I	DC.L	3 ; 4 bájtos terület, értéke 3
J	DC.L	4 ; 4 bájtos terület, értéke 4
N	DC.L	0 ; 4 bájtos terület, értéke 0

Csak érdekesség, nem kell megtanulni!

Assembly példaprogramok

- $N = I + J$ megvalósítása
- UltraSPARC III

Címke	Műv. kód	Operandusok	; Megjegyzések
FORMULA:	SETHI	%HI(I), %R1	; R1 = I címének felső bitjei
	LD	[%R1+%LO(I)], %R1	; R1 = I
	SETHI	%HI(J), %R2	; R2 = J címének felső bitjei
	LD	[%R2+%LO(J)], %R2	; R2 = J
	NOP		; Várakozás J betöltésére
	ADD	%R1, %R2, %R2	; R2 = R1 + R2
	SETHI	%HI(N), %R1	; R1 = N címének felső bitjei
	ST	%R2, [%R1+%LO(N)]	; N = I + J tárolása
I:	.WORD 3		; 4 bájtos terület, értéke 3
J:	.WORD 4		; 4 bájtos terület, értéke 4
N:	.WORD 0		; 4 bájtos terület, értéke 0

SZÁMÍTÓGÉPES ARCHITEKTÚRÁK

Az operációs rendszer gép szintje I.

Nagy Antal

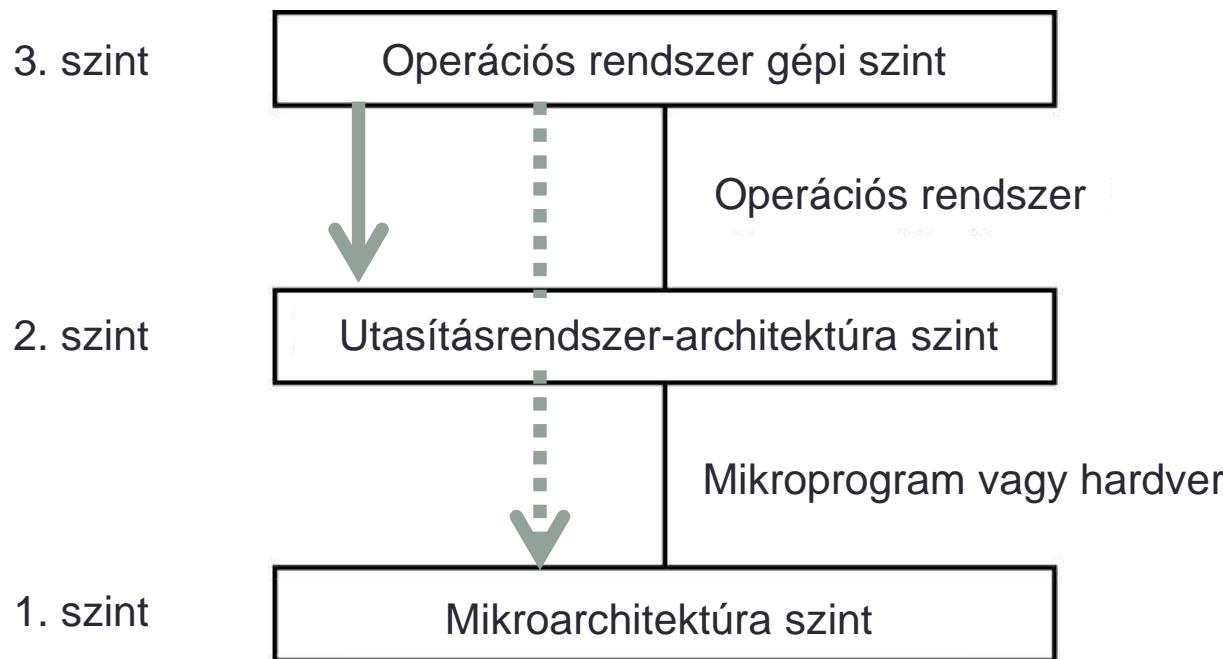


Gyönyörű kapcsolódási felület

Ronda kapcsolódási felület

Az operációs rendszer gépi szintje (OSM)

- Az ISA-szint felett és alatt
 - Új utasítások
- Szoftveres úton van megvalósítva
 - Hardveres megvalósításnak sincs akadálya



Az operációs rendszer gépi szintje (OSM)

- Nem igazi hardveres szintek
 - ISA
 - OSM
- Az OSM-szintű utasítások a teljes utasításkészletet jelentik
 - Az összes ISA-szintű utasítás
 - Operációs rendszer által hozzáadott **új utasítások**
 - Rendszerhívások (system calls)
- A rendszerhívások az OS valamely előre definiált szolgáltatását hívja meg
 - Pl. adatok olvasása fájlból

Az operációs rendszer gépi szintje

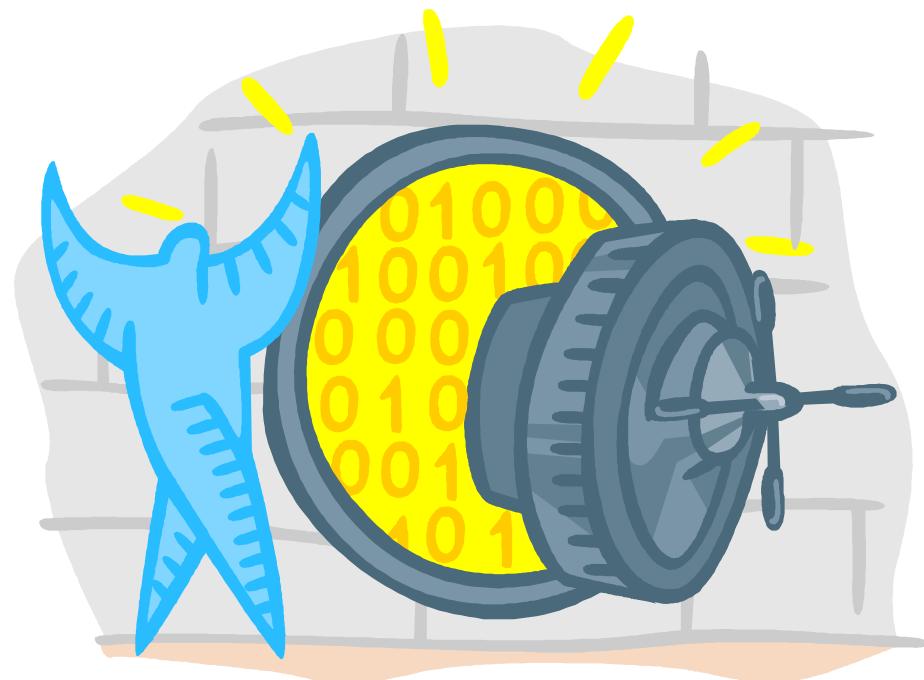
- OSM szint mindenkor interpretált
 - OSM utasítást lépésekkel hajtja végre
 - ISA-szintű utasítás végrehajtása
 - Az OS közreműködése nélkül
 - Közvetlenül az alatta lévő mikroarchitektúra szintjén végzi el
- Témakörök
 - Virtuális memória
 - Nagyobb memória elérése
 - Fájlszintű B/K
 - Párhuzamos feldolgozás
 - Processzus
 - Futó program
 - Állapot információk
 - Memória
 - Regiszterek
 - Utasításszámláló
 - B/K állapot

Virtuális memória

- A memória kicsi és drága
 - Programozók feladata volt, hogy a programok beleférjenek az adott memória mennyiségébe
 - Lassabb programok a kis memória méret miatt
- Másodlagos memória
 - Lemez terület
- Programok felosztása kisebb részekre
 - Befértek a memóriába
 - Átfedések/Overlays
 - A részek külön-külön töltődtek be a memóriába
 - Gondoskodni kellett
 - Átfedésekről
 - Egyes részek elhelyezéséről a másodlagos memóriában
 - Memória és a lemez közötti mozgatásukról
 - ...

Virtuális memória

- Overlays
 - Sok munkával járt a kezelése
- Virtuális memória
 - Adminisztrációs feladatok elvégzése
 - 1960-s évek
 - Kutatási projektek
 - 1970-s évek
 - Legtöbb gépen elérhető volt

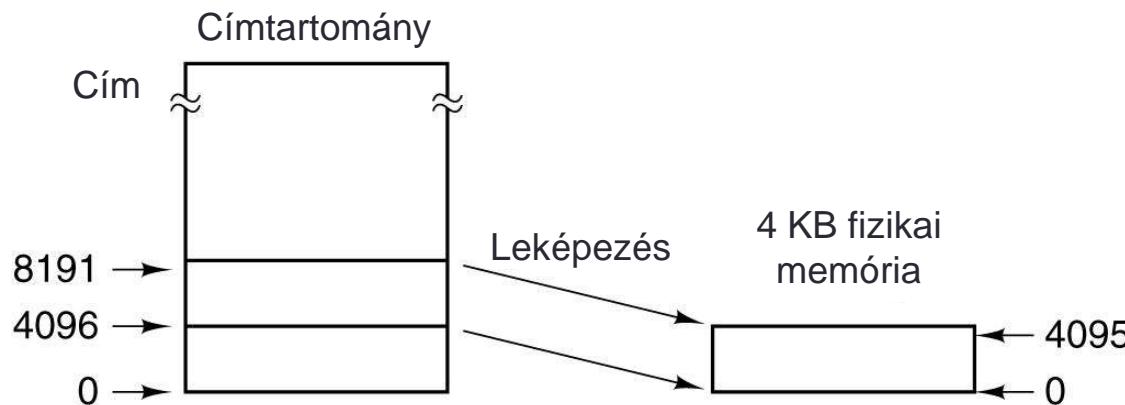


Lapozás

- Címtartomány és memóriarekeszek fogalmának a szétválasztása
 - 4096 szavas memória
 - Utasításokban 16 bites címrész
 - $2^{16}=65\ 536$ szót tud megcímezni
 - 4096 alatti és feletti címek megkülönböztetése
 - Hasznos és haszontalan címtartomány
 - A 4096 feletti címekhez nem tartozott valódi memória
 - A hardver egy-egy értelmű megfeleltetést hozott létre a címtartomány és a memóriarekeszek között

Lapozás

- A címtartomány és a memóriacímek elkülönítésének az elve
 - Bármely időpillanatban 4096 szó érhető el a memóriából
 - Ezek nem feltétlenül a 0 – 4095 közötti memóriacímeknek felelnek meg
 - Pl. a 4096 címre való hivatkozás esetén a memória 0-ás című szavát kell használni
 - A 4097-es cím esetén pedig a memória 1-es memóriacímét, stb.
 - Leképzés a címtartományból a valódi memória címek halmazára



Lapozás

- Egy 4KB-os, virtuális memória nélküli gép esetén
 - Fix leképezésünk van a 0-4095 közötti címek és a memória 4096 szava között
 - Mi történik, ha 8192 és 12287 közötti címre hivatkozunk?
 - Futási hibát kapunk
 - Nem létező memória címre való hivatkozás
- Virtuális memóriával rendelkező gép esetén a következő lépések hajtódnak végre
 1. A memória tartalmának a lemezre való mentése
 2. A 8192 és 12287 közötti szavak megkeresése a lemezen
 3. A lemezről való betöltése a 8192 és 12287 szavaknak
 4. A memória térkép megváltoztatása
 - A 8192 és 12287 közötti címek leképezése a 0 és 4095 közötti memóriarekeszre
 5. A végrehajtás folytatása

Lapozás

- Az átfedés automatikus kezelésének ezt a technikáját lapozásnak (paging) nevezük
 - A lemezről beolvasott részeket pedig lapoknak (pages) nevezük
- Virtuális címtartomány
 - Program által hivatkozott memóriacímek
- Fizikai címtartomány
 - Tényleges memória címek
- Memória térkép/laptábla
 - A virtuális címeknek megfelelő fizikai címek
- A programok a teljes virtuális címtartományt használhatja
 - A programozónak nem kell tudnia a háttérben folyó műveletekről
 - Transzparens mechanizmus

Lapozás

- A fizikai memória lehet nagyobb és kisebb is a virtuális címtartománynál
- Az operációs rendszer által szolgáltatott virtuális gép
 - minden virtuális cím mögött valódi memória van
 - Illúzió
 - Csak az operációs rendszer íróinak kell tudnia, hogy hogyan tartható fent ez az illúzió



A lapozás megvalósítása

- Virtuális memória
 - Lemezterület
 - Lemezen lévő változat és a memóriába betöltött darabjai közötti frissítés
 - Ha változott a tartalom
- Azonos méretű **lapokra** való felosztása a virtuális címtartománynak
 - 2 valamely hatványa
- Fizikai címtartomány **lapkeretekre** való felosztás

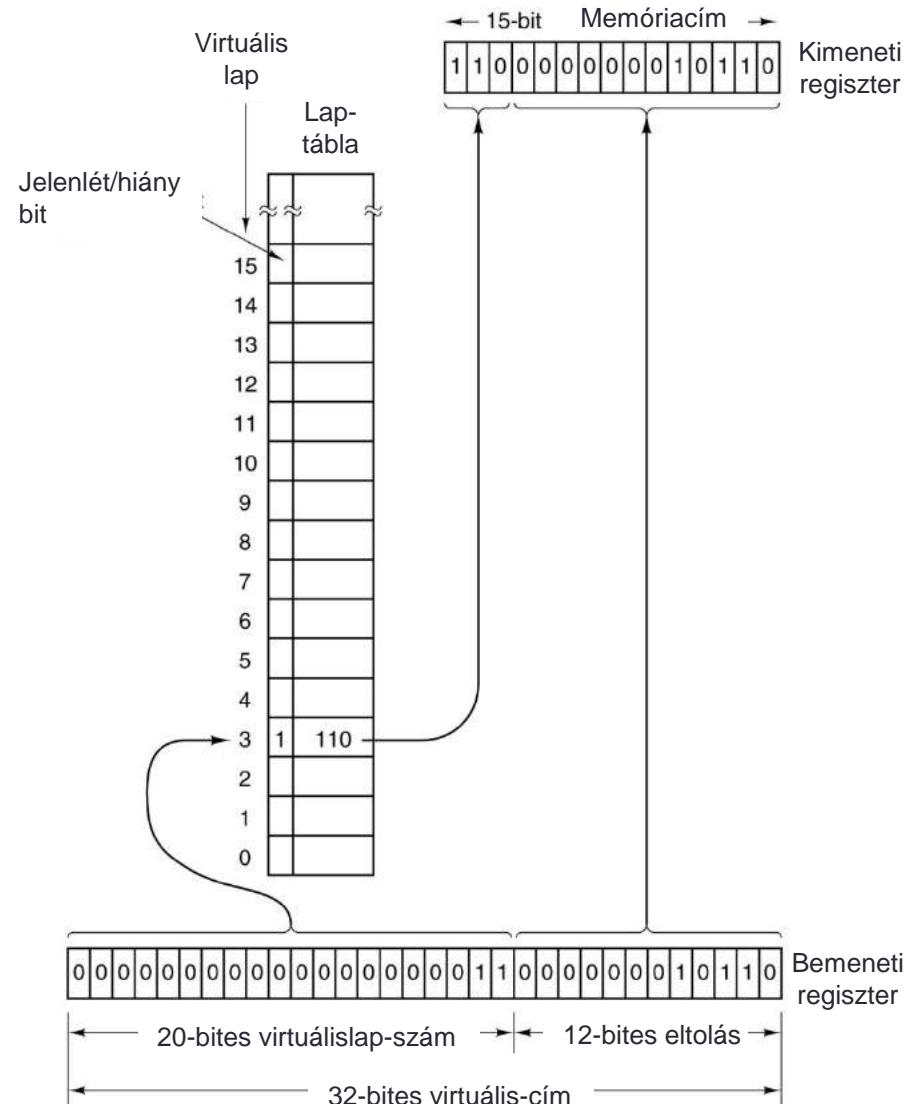
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

Lap	Virtuális címek
~	~
15	61440 – 65535
14	57344 – 61439
13	53248 – 57343
12	49152 – 53247
11	45056 – 49151
10	40960 – 45055
9	36864 – 40959
8	32768 – 36863
7	28672 – 32767
6	24576 – 28671
5	20480 – 24575
4	16384 – 20479
3	12288 – 16383
2	8192 – 12287
1	4096 – 8191
0	0 – 4095

A fizikai memória alsó 32 KB-ja	Lap-keret	Fizikai címek
7	28672 – 32767	
6	24576 – 28671	
5	20480 – 24575	
4	16384 – 20479	
3	12288 – 16383	
2	8192 – 12287	
1	4096 – 8191	
0	0 – 4095	

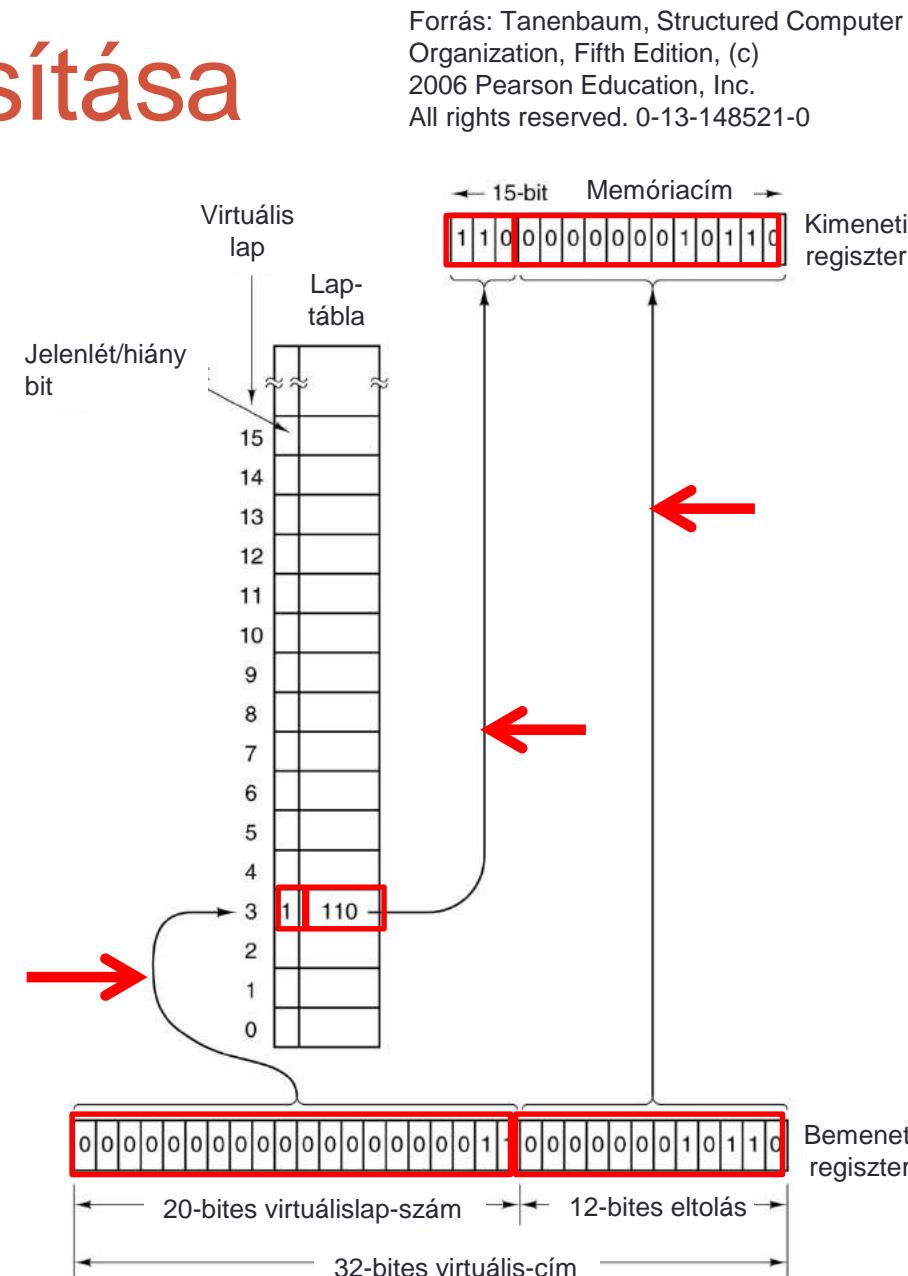
A lapozás megvalósítása

- Hogyan képezhető le egy 32 bites virtuális cím a fizikai memória egy címére?
 - Leképezést megvalósító egység
 - Memória kezelő egység
 - Memory Management Unit (MMU)
 - Vagy a CPU lapkán vagy külön helyezkedik el
 - 32 bites bemeneti regiszter
 - 15 bites kimeneti regiszter



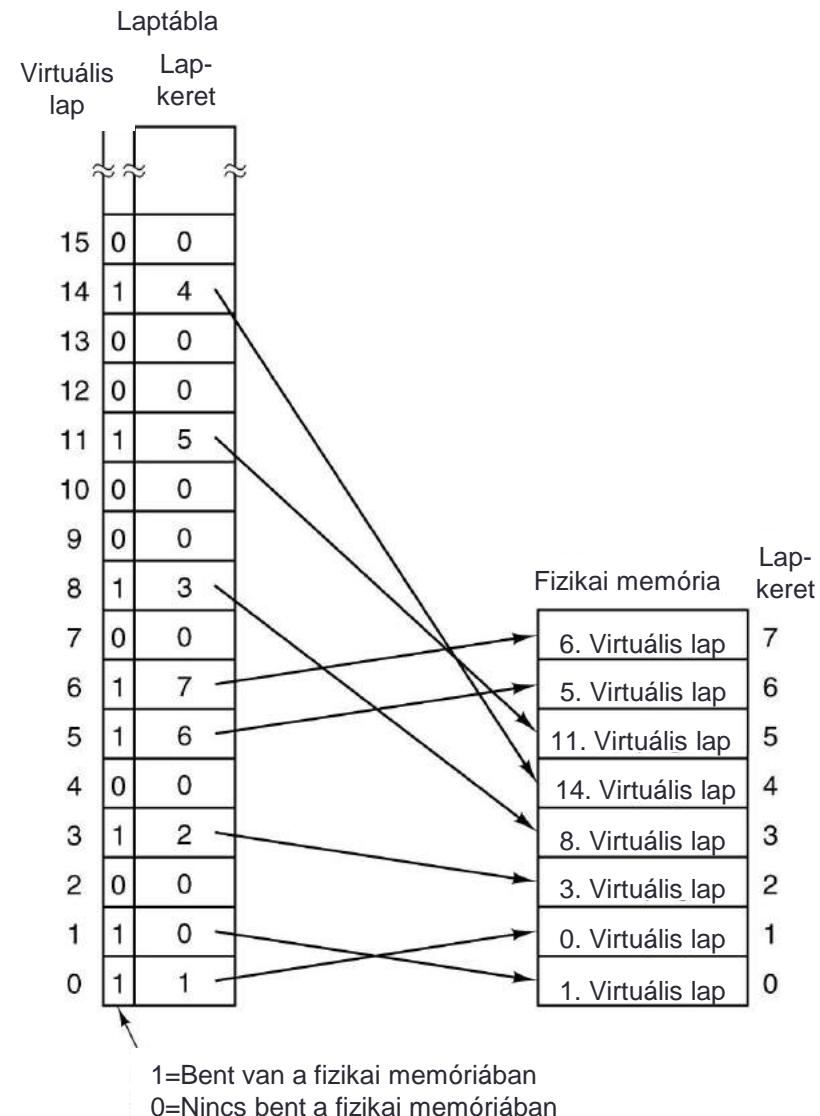
A lapozás megvalósítása

- 32 bites virtuális cím
 - 20 bites virtuális lapszám
 - 4 KB-os lapméret miatt
 - 12 bites eltolás (offset)
 - Lapszám → Index
 - MMU megvizsgálja, hogy a lap bent van-e a fizikai memóriában
 - Jelenlét/hiány bit
- 3 bit átmásolása a 15 bites kimeneti regiszter felső 3 bitjére
 - Párhuzamosan az eltolás is átmásolódik az alsó 12 bitre



Lapozás megvalósítása

- Virtuális lapok egy lehetséges leképezése
- A leképezésnek gyorsnak kell lennie
- Jelenlét/hiány bit = 0
 - Page fault hiba
 - Kezelní kell



A kérésre lapozás és a munkahalmaz modell

- A virtuális lap nem mindenkor található meg a fizikai memóriában
 - Nincs elég memória
- Laphiány (page fault) esetén az operációs rendszer feladatai
 - Be kell olvasni a lemezről a kért lapot
 - Előfordulhat, hogy egy bent lévő lapot ki kell írni a lemezre előtte
 - Ki kell választani azt, hogy melyiket dobjuk ki
 - Be kell írnia az új helyét a laptáblába
 - Meg kell ismételni a hibát okozó utasítást

A kérésre lapozás és a munkahalmaz modell

- Amikor egy program egyetlen része sincs a memóriában
 - A laptáblában nincs bejegyzés
 - Indításkor a CPU az első utasítást megróbálja betölteni
 - Egyből laphiány lép fel
 - Hatására az első lap betöltődik a memóriába
 - Másodlagos tárolóból
 - Az információ bejegyzésre kerül a laptáblában
 - Elkezdődhet az első utasítás végrehajtása
- Újabb laphiány léphet fel
 - Több is
- Ezt a fajta virtuális memória használatot kérésre lapozásnak (demand paging) nevezzük

A kérésre lapozás és a munkahalmaz modell

- A program elindulásakor lényeges, hogy a kérésre lapozást használjuk-e
 - Bizonyos idő elteltével a szükséges lapok bent lesznek a memóriában
- Időosztásos rendszernél
 - Több program van bent egyszerre a memóriában
 - Programváltáskor is változik a memória térkép
 - Kritikussá válhat a helyzet
 - Sok laphiány/page fault keletkezhet

A kérésre lapozás és a munkahalmaz modell

- A programok címtartományra való hivatkozása néhány lap körül sűrűsödnek össze
 - Lokalitás elv (locality principle)
- Egy memória hivatkozás
 - Utasítást tölthet be
 - Adatokat olvashat vagy írhat
- Munkahalmaz (working set)
 - Adott t időpillanatban a legutóbbi k memóriahivatkozásban szereplő lapok halmaza
 - Lassan változik
 - Becsülhető, hogy a program indításakor melyik lapokra van szükség
 - Betölthetőek az indítás előtt

Lapcserélő eljárások

- Ideális esetben a munkahalmazban lévő lapokat a memóriában lehetne tartani
 - Csökkenteni lehetne a laphiányok számát
 - Szimulációkat lehetne végezni
 - Ezek az eredmények csak az adott környezetben lennének érvényesek
 - A memória térkép állandóan változik
 - Az operációs rendszer feladata a munkahalmazhoz tartozó lapok felderítése
- Laphiány esetén a szükséges lapot be kell tölteni a lemezről a fizikai memóriába
 - Ha tele van a fizikai memória, akkor először helyet kell csinálni
 - Ki kell választani azt a lapot, amelyiket „kidobunk” a memóriából
 - Változás esetén ki kell írni a lemezre (virtuális memória)
 - Kell egy megfelelő algoritmus, ami jól választja ki a kidobandó lapot

Lapcserélő eljárások

- Ha olyan lapot dobunk ki, amelyikre nem sokkal későbbi utasításban hivatkozunk, akkor újabb laphiba keletkezik
- Az operációs rendszerek megpróbálják megjósolni
 - Leghaszontalanabb lapok kiválasztása
 - Eltávolítása minél későbbi időpontban fog laphiányt okozni
 - minden lapra meg kellene határozni a várható legközelebbi hivatkozás időpontját
 - Azt kell kiválasztani, amelyiknél ez az érték a legnagyobb



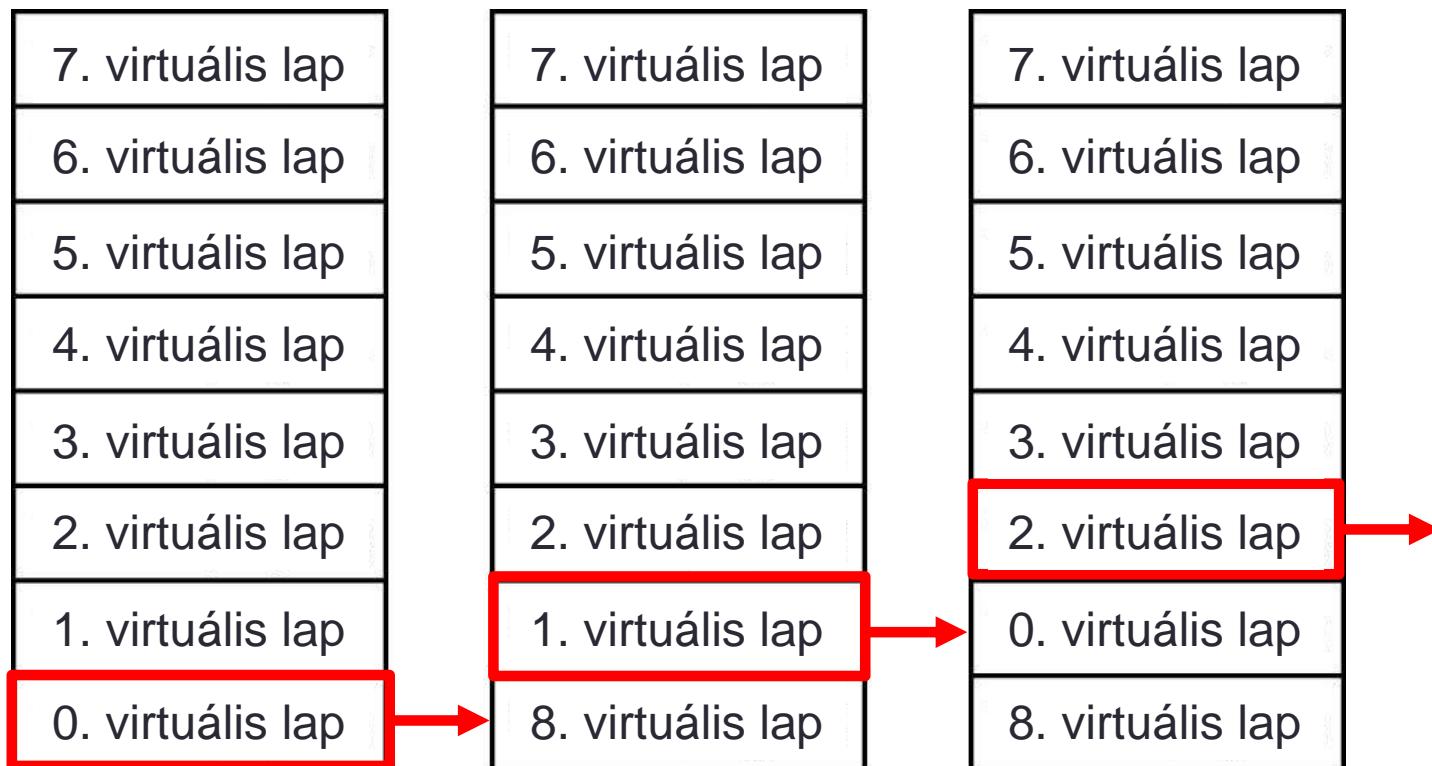
Lapcserélő eljárások

- A legrégebben nem használt lap kiválasztása
 - Valószínűleg nincs benne a munkahalmazban
 - Least Recenly Used (LRU)
 - Lapok listában vannak tárolva
 - Elől a legutoljára használt lap található
 - minden memóriahivatkozáskor az adott lapot a lista elejére rakjuk
 - Hardveres implementáció
 - 64 bit-es számláló
 - minden memóriahivatkozásnál automatikusan eggyel nő
 - minden laptáblabejegyzésben van egy mező, amely az előző értéket tárolja
 - Laphiba esetén a legkisebb számlálóértékkel rendelkező lapot választjuk
 - Ez lesz a legrégebben használt lap

Lapcserélő eljárások

- LRU Jól használható
 - Bizonyos esetekben nem működik jól

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Lapcserélő eljárások

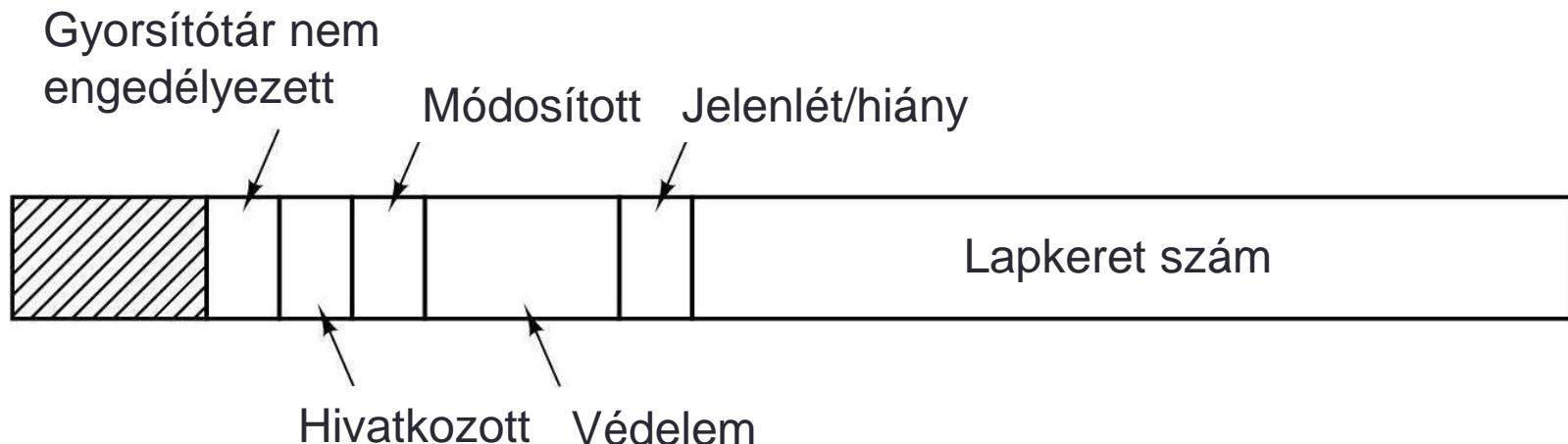
- Ha a rendelkezésre álló memória nagyobb a munkahalmazénál
 - Az LRU minimalizálja a laphiányok számát
- Másik algoritmus
 - FIFO (First In First Out)
 - A legrégebben betöltött lapot távolítja el
 - Nem számít az, hogy mikor hivatkoztak rá utoljára
 - Számláló, kezdetben 0
 - Eggyel növeljük a memóriában tartózkodó lapok számlálóját laphiánykor
 - Az új lapok számlálója 0
 - A legnagyobb számlálójú lapot választjuk ki

Lapcserélő eljárások

- Ha a munkahalmaz nagyobb, mint a rendelkezésre álló lapkeretek száma
 - Gyakori laphibák
 - Egyik algoritmus sem fog jó eredményt adni
 - Folyamatos laphiány
 - Vergődés
 - Nincs gond azokkal a programokkal
 - Nagy virtuális címtartományt használnak
 - Kicsi a munkahalmazuk
 - Időben lassan változó
 - Mindig elfér a rendelkezésre álló memóriában

Lapcserélő eljárások

- Ha a kiválasztott, „eldobandó” lapon nem történt módosítás
 - Nem kell visszaírni a lemezre
- Módosítás esetén frissíteni kell a lemez tartalmat
 - Virtuális memória lemezterület
- Laptábla bejegyzés



Lapméret és elaprózódás

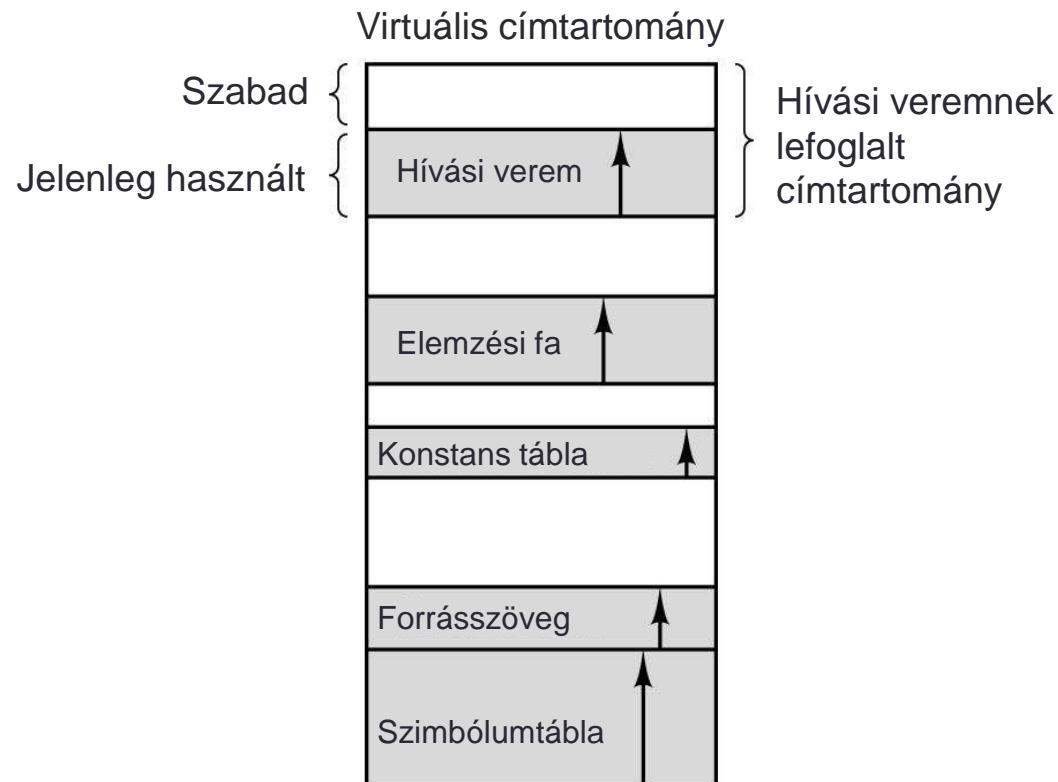
- A programokhoz tartozó utolsó lapon lehet kihasználatlan hely
 - Nem használjál ki teljesen a lapméretet
 - Belső elaprózódás
- Ha lapmáret n bájtos
 - Átlagosan $n/2$ lesz a veszteség
- Kis lapmáretet választva
 - Sok lap
 - Nagymáretű laptábla
 - Hardveres megvalósításnál több regiszter
 - Drágább

Szegmentálás

- Az eddigi virtuális memória lineáris (egydimenziós)
 - A címek 0-tól kezdődtek
- Sok esetben előnyösebb több külön címtartomány alkalmazása
- Például egy fordító programnak a fordítás során több táblázatot használ
 1. Változók nevét és attribútumait tartalmazó szimbólum tábla
 2. A listázáshoz megőrzött forráskód
 3. Az összes felhasznált egész és lebegőpontos konstanst tartalmazó tábla
 4. A program szintaktikus elemzésekor létrehozott elemzési fa
 5. A fordítóprogramon belüli eljáráshíváshoz tartozó verem

Szegmentálás

- Az első négy tábla mérete folyamatosan nő
- A verem mérete nő illetve csökken
- Egydimenziós memória
- Kivételesen sok változó esetén problémák adódnak



Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

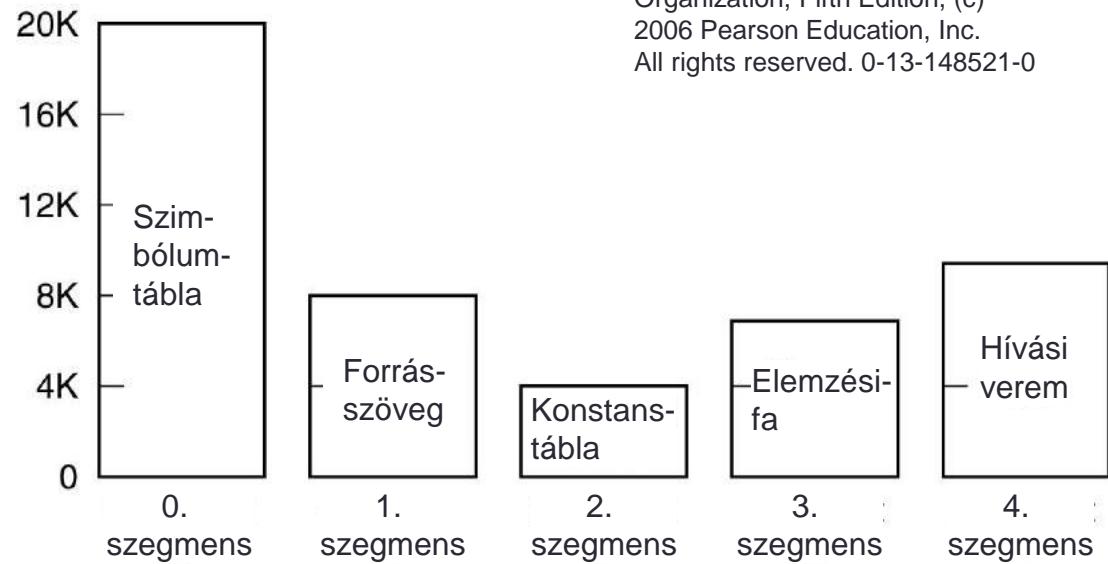
Szegmentálás

- Változó méretű táblák probléma

- Szegmensek

- Független címtartományok
 - 0-tól egy megadott méretig
 - Hosszuk változhat
 - Két dimenziós memória
 - Szegmens száma
 - Szegmensen belüli cím
 - Logikai egység
 - Programozó számára látható

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Szegmentálás

- A szegmens tartalmazhat
 - Eljárást
 - Tömböt
 - Vermet
 - Skaláris változókból álló gyűjteményt
- Más előnyök
 - Különböző eljárások 0 kezdőcímtől más más szegmensekben helyezkednek el
 - A külön lefordított eljárások könnyebben szerkeszthetőek össze
 - Megcímzése $(n,0)$ címmel történik
 - Ha módosul az egyik eljárás, akkor a többit nem kell újrafordítani
 - Különböző programok közötti közösen használt kód- és adatmegosztás

Szegmentálás

- Szegmensek védelme
 - Eljárás szegmensek, csak végrehajthatóak
 - Tilos olvasni és felülírni
 - Lebegőpontos tömbök
 - Írhatóak, olvashatóak
 - Nem végrehajthatóak
 - Segíti a programozási megtalálását
- Tudjuk, hogy mi van eltárolva az adott szegmensekben
- A lineáris esetben nem így van
 - Lapok tartalma bizonyos értelemben véletlenszerű
 - A programozó nem tud a lapozásról

Összehasonlítás

Szempontok	Lapozás	Szegmentálás
Tudnia kell-e róla a programozónak?	Nem	Igen
Hány lineáris címtartomány létezik?	1	Sok
Meghaladhatja-e a virtuális címtartomány nagysága a fizikai memória méretét?	Igen	Igen
Könnyen kezelhetők-e a változó méretű táblák?	Nem	Igen
Milyen céllal dolgozták ki ezt a technikát?	Nagy memória szimulálása	Több címtartomány biztosítása

A szegmentálás megvalósítása

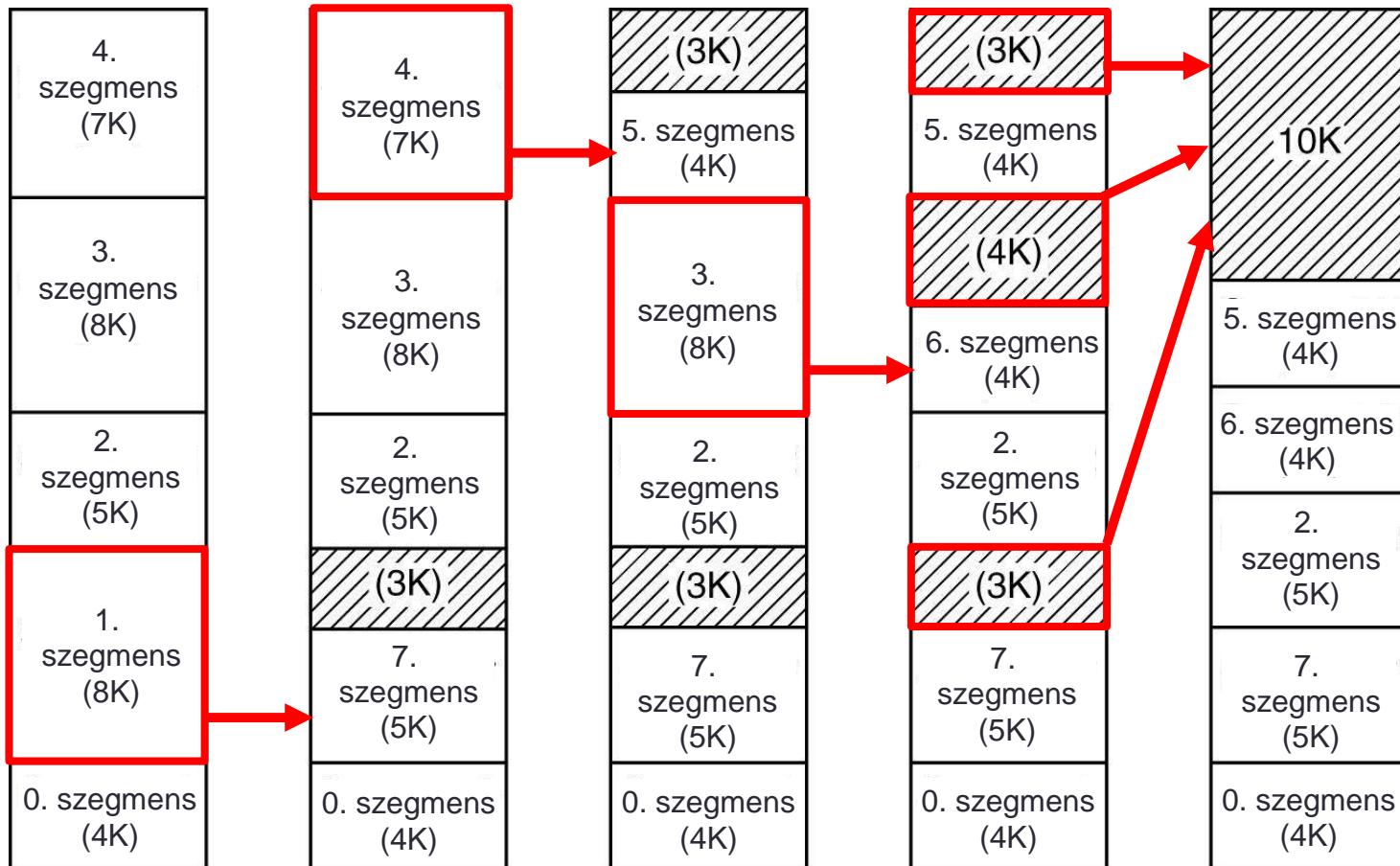
- **Csere (swapping)**

- minden időpillanatban a szegmensek bizonyos halmaza van a memóriában
- Ha olyan szegmensre hivatkozunk, ami nincs bent a memóriában
 - Betöltődik
- Ha nincs elég hely
 - Egy vagy több szegmenst ki kell írni
 - Ha tiszta, akkor eldobható
- Hasonlít a kérésre való lapozáshoz
 - A szegmensek mérete változó

A szegmentálás megvalósítása

- Példa változó méretű szegmens cseréjére

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Szegmentálás megvalósítása

- Memória szegmensek és lyukak
 - Külső elaprózódás
- Előfordulhat, hogy az elaprózódás miatt nem tölthető be egy szegmens
 - Hiába van összességében elegendő memória
 - Pl. a 3. szegmensre hivatkozva az előző példában
 - Mérete 8K
 - Összességében 10K szabad terület van
 - Nem egybefüggő
 - 3 részben
- Elaprózódás elkerülése
 - 0. memóriacím felé toljuk a lyuk mögötti szegmenseket
 - Túl sok időt igényel
 - Megvárjuk, amíg a külső elaprózódás egy „komolyabb” szintet el nem ér
 - Csak ezután rendezzük át a memóriát

Szegmentálás megvalósítása

- Algoritmusok, amelyek meghatározzák, hogy az adott szegmenseket melyik üres helyre töltük be
 - Szükség van
 - Lyukakat tartalmazó lista
 - Cím
 - Méret
 - Legjobb illeszkedés
 - A legkisebb üres terület kiválasztása, amelyikbe még belefér a szegmens
 - Nagy lyukak „kímélése”
 - Nem darabolódnak fel
 - Kicsi üres területet keletkeznek, amit nem lehet később már felhasználni
 - Első illeszkedés
 - A listán körbe megy
 - Az első üres terület kiválasztása, amely elegendően nagy a szegmens beolvasásához
 - Általános hatékonyság szempontjából jobb
 - Gyorsabb

Szegmentálás megvalósítása

- Mindkét algoritmusnál csökken a lyukak átlagos nagysága
 - Az idő műlásával apró, „haszontalan” üres területek keletkeznek
 - Szükség van kis üres területek összevonására
 - Szegmens eltávolításakor
 - Két szomszédos üres terület összevonásával nagyobb üres területet hozhatunk létre

Szegmentálás megvalósítása

- **Lapozás**
 - Szegmensek fix méretű lapokra való felosztása
 - Előfordulhat, hogy valamely szegmens lapjainak egy része a memóriában van a többi meg a lemezen van tárolva
 - Lapozáskor minden szegmensnek külön laptábla kell
 - A szegmenseken belül egy lineáris címtartományt használunk
 - A korábbi lapozási technikák használhatóak
- **MULTICS**
(MULtiplexed Information and Computing Service)
 - Lapozással kombinált szegmentálás használt
 - A címek két részből álltak
 - Szegmensszámból
 - Szegmensen belüli címből

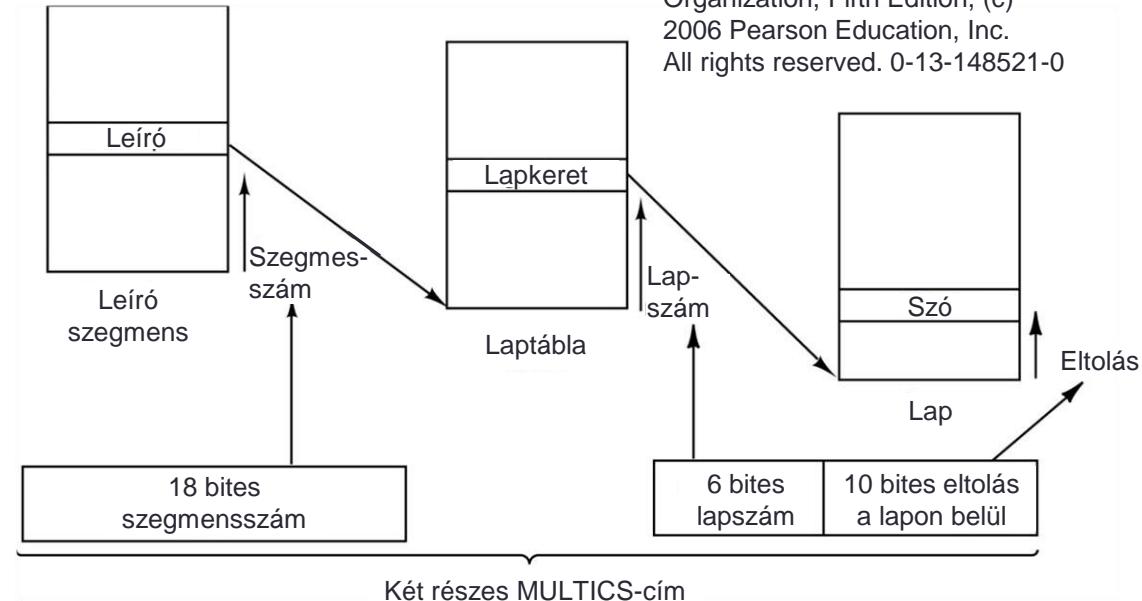
Szegmentálás megvalósítása

- MULTICS

- minden processzushoz tartozott egy leíró szegmens
 - Szegmens leírók
 - Amikor a hardver egy virtuális címet kapott
 - Indexként használta és kikereste a megcímezendő szegmens leírót
 - A leíró egy szegmens lapjára mutatott
 - Szegmensek lapozásának a végrehajtására szolgált
 - Asszociatív memória
 - Gyorsítás
 - Legutóbb használt szegmens/lap kombinációja

- Intel 386-os óta
 - Virtuális memóriakezelés

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Virtuális memória és gyorsítótár

Virtuális memória

- A program a lemezen van
 - Fix méretű lapokra van felosztva
 - A lapok egy részhalmaza van a memóriában
 - Kevés laphiány esetén a futása gyors lesz
- Különbségek
 - Laphiányokat
 - Operációs rendszer kezeli
 - A virtuális címek legnagyobb helyértékű bitje szerint szervezik a laptáblákat

Gyorsítótár

- Az egész program a memóriában van
 - Fix méretű gyorsítóblokkok
 - A program többnyire a gyorsítótárban lévő blokkokat használja
 - Kevés gyorsítótárihiány esetén a futása gyors lesz
- Különbségek
 - Gyorsítótárihiányokat
 - Hardver kezeli
 - Mérete kisebb, mint a laptáret
 - 64 bájt illetve 8 KB
 - A memória címek legkisebb helyértékei szerint indexelnek

Implementációs eltérések, de hasonló alapelvek

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Perifériák

Tanács Attila

Áttekintés

- **Háttérmemória**

- Mágneslemezek, IDE, SCSI, RAID
- SSD
- CD-ROM-ok, DVD, Blu-Ray

- **Bemenet/kimenet**

- Terminálok (billentyűzet, kijelző), egér
- Nyomtatók
- Telekommunikációs berendezések
- Digitális kamerák

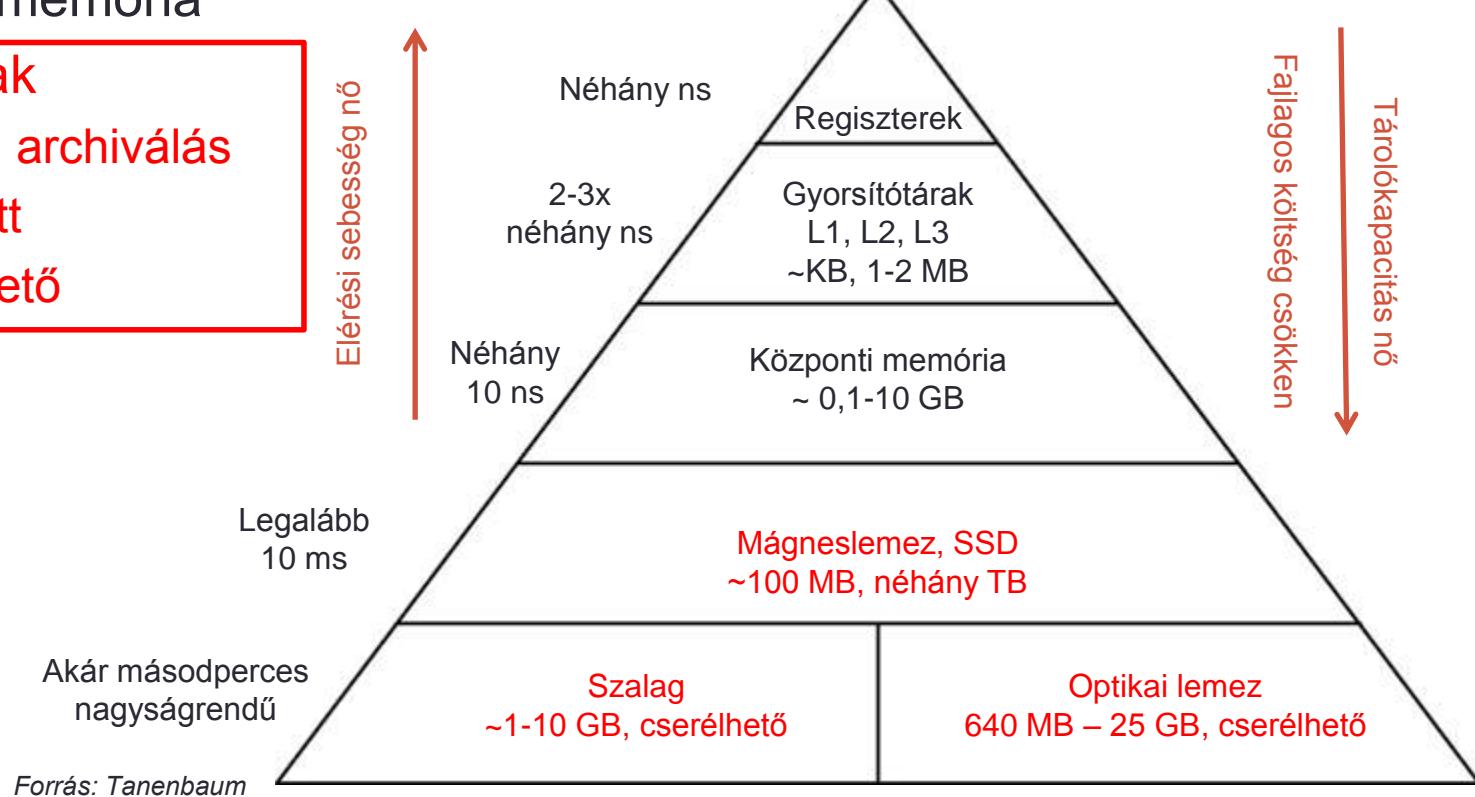
Háttérmemória

- **Memóriahierarchia**

- Regiszterek, gyorsítótárak
 - Gyors CPU elérés, kis méretű, drága!
- Központi memória

- **Háttértárak**

- Tárolás, archiválás
- Rögzített
- Cserélhető



Háttértárak



Mágneslemezek

- **Részei**

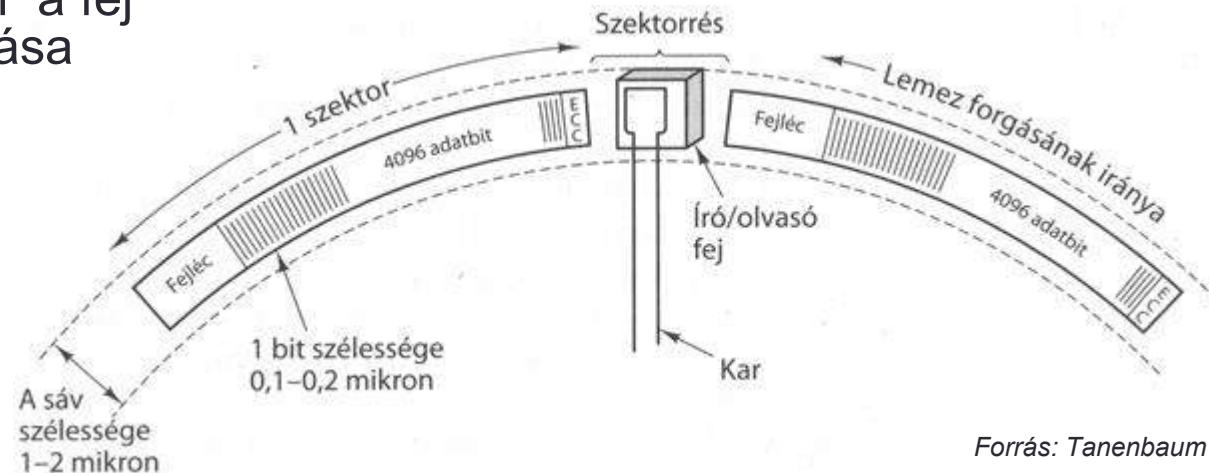
- Mágnesezhető felületű, forgó alumínium**korong**
 - Átmérő kezdetben 50 cm, jelenleg 3-12 cm
- Indukciós tekercset tartalmazó **fej**
 - Lebeg vagy érinti a felszínt
- **Kar**
 - Sugárirány mentén a fej egyvonalú mozgatása

- **Írás**

- Pozitív vagy negatív áram az indukciós tekercsben, a lemez adott helyen mágneseződik

- **Olvasás**

- Mágnesezett terület felett elhaladva pozitív vagy negatív áram indukálódik a mágneses polarizációnak megfelelően



Mágneslemezes adattárolás

- **Sáv**

- Koncentrikus körök mentén
- Egy teljes körülfordulás alatt felírt bitsorozat
- Centiméterenként 5-10 ezer sáv (szélesség)

- **Szektor**

- 1 sávon több szektor
 - *Fejléc*: fej szinkronizálásához
 - *Adat* (pl. 512 bájt)
 - *Ellenőrző kód*
 - Hamming vagy Reed-Solomon
 - **Szektorrész**



- **Lineáris adatsűrűség**

- Kerület mentén, 50-100 ezer bit/cm

- **Merőleges rögzítés**

- Tárolás hosszirány helyett „befelé” történik

- **Kapacitás**

- Formázott és formázatlan

Mágneslemezek felépítése

- **Felépítés**

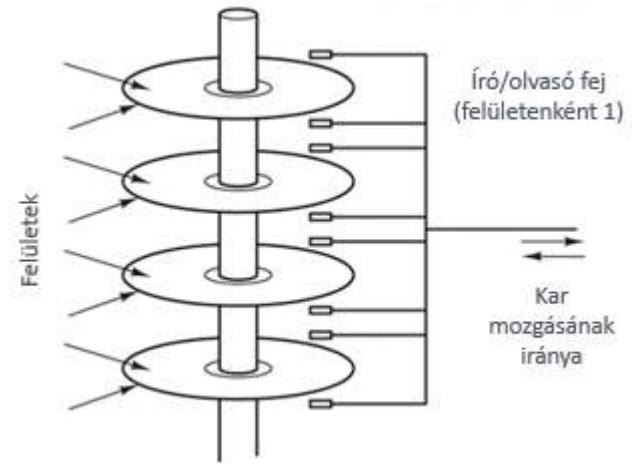
- Fontos a tisztaság és a pormentesség → zárt merevlemezek

- **Lemezegység**

- Közös tengelyen több lemez (6-12), azonos fej pozíció!
- Cilinder: adott sugárpozíción lévő sávok összessége

- **Teljesítmény**

- Keresés (seek): fej megfelelő sugárirányba állítása (kar)
 - 1 ms: egymás utáni sávok
 - 5-10 ms: átlagos (véletlenszerű)
- Forgási késleltetés
 - A kerület mentén a fej alá fordul a kívánt terület
 - Fél fordulat ideje, 3-6 ezredmásodperc
 - 5400, 7200, 10880 fordulat / perc mellett





Mágneslemezek jellemzői

- **Átviteli sebesség**

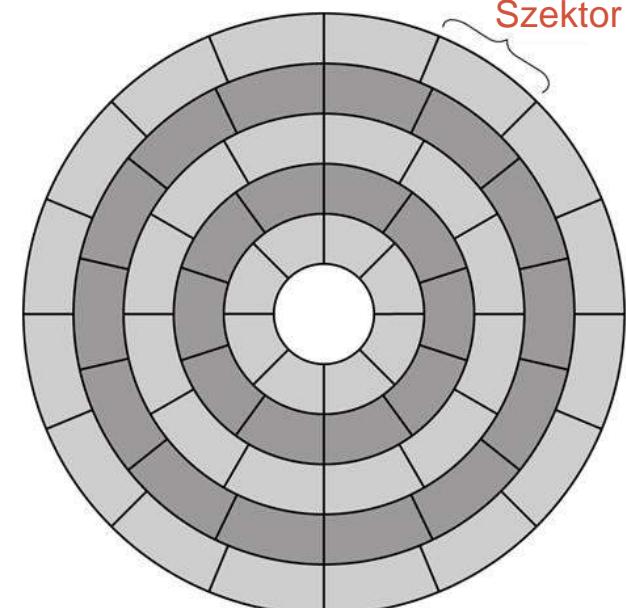
- 20-40 MB/mp
- Különböző maximális és átlagos átviteli sebesség!

- **Írás sűrűsége**

- A külső sávok kerülete nagyobb, de a szögsebesség állandó!
- Régebben
 - Sávonként rögzített számú szektor
 - Változó bitsűrűség a kerület mentén
- Manapság
 - Cilinder zónák kialakítása (10-30 között)
 - Külső részen több szektor
 - Bonyolultabb, de nagyobb kapacitás

- **Mechanikai sérülés előfordulhat!**

- Fizikai behatásra a fej megsértheti a lemezt!



Mágneslemezek jellemzői

- **Lemezvezérlő lapka**
 - Sokszor saját CPU-t is tartalmaz
 - Szoftverből érkező parancsok fogadása
 - READ, WRITE, FORMAT
 - Kar mozgatása
 - Hibák felismerése és javítása
 - Javíthatatlan hiba esetén fizikai áthelyezés
 - Bájtok oda- és visszaalakítása bitek sorozatává
 - Pufferelés (gyorsítás)

Mágneslemez típusok

• Hajlékonylemez (floppy)

- Több évtizedig használták (elavult)
- Nyitott kialakítás, kisebb bitsűrűség
- Fej hozzáér a lemezhez → gyorsabb elhasználódás, sérülékenység
- Ha nincs olvasás/írás, fejet visszahúzza, forgást leállítja
 - Újra művelet esetén fel kell „pörgetni” a lemezt (kb. fél másodperc)

• IDE-lemezek

- (Zárt) merevlemezek vezérlésére (Integrated Drive Electronics)
- Meghajtóba integrált megoldás, 1980-as évek közepétől
- Lemezcímzés
 - A számítógép BIOS (Basic I/O System) rendszere kezelte
 - Regiszterekbe töltött fej, cilinder, szektor sorszámokkal
 - Eredetileg 4, 6 és 10 biten kódolva → 504 MB maximális kapacitás

8, 5.25 és 3.5 collos floppy-k



Forrás: [WikiMedia Commons](#)

Mágneslemez típusok

- **EIDE-lemezek** (Extended IDE)
 - LBA (logical block addressing) lemezcímzési mód
 - Szektorok sorszámozása 0 – 2^{28} -1 között (128 GB)
 - A vezérlő alakítja fej, cylinder, szektor indexekre
 - 4 eszköz kezelhetősége
 - Két csatorna, csatornánként egy elsődleges és egy másodlagos egység
 - CD-ROM és DVD-meghajtók kezelése is
 - Sebesség 4 MB/mp-ről 16,67 MB/mp-re
- **Újabb változatai**
 - PATA – Parallel ATA, párhuzamos ATA
 - ATA-3, ATAPI-4, ATAPI-5
 - Sebességnövelés 66 MB/mp-ig
- **ATAPI-6**
 - 48 bites LBA címek bevezetése (128 petabájt max. kapacitás)
 - Zajcsökkentés, 100 MB/mp sebesség

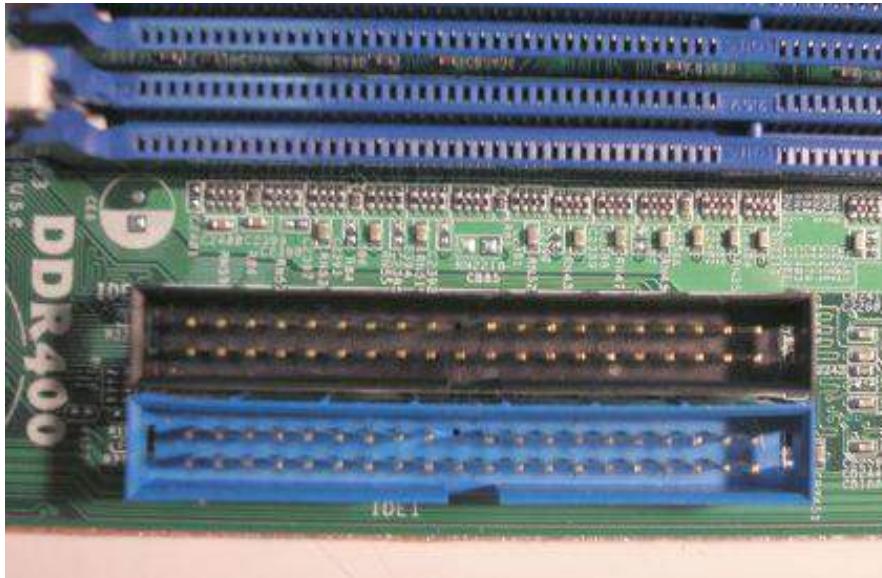
Mágneslemez típusok

- PATA (EIDE – ATAPI 6) csatlakozó típusok



Kép forrása:
[WikiMedia Commons](#)

PATA alaplapi csatlakozó



Kép forrása: [WikiMedia Commons](#)

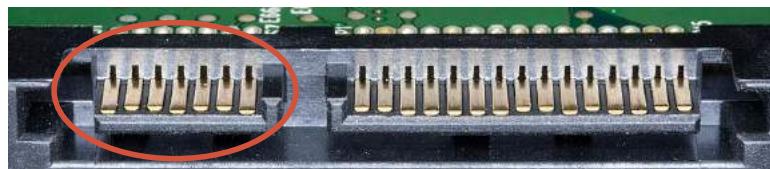
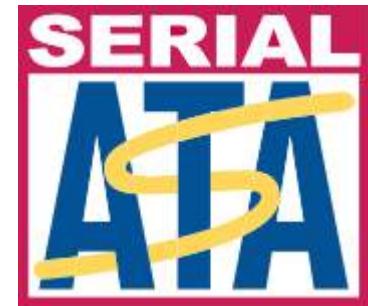
PATA merevlemezek csatlakozói



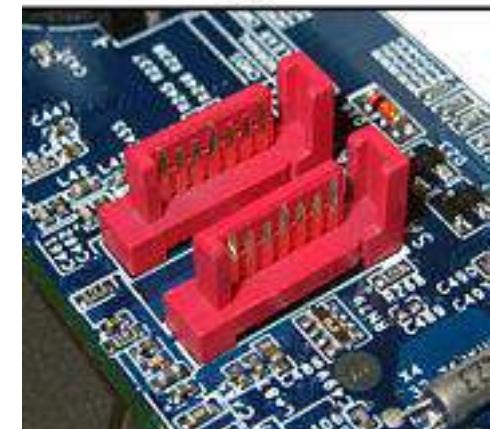
Kép forrása: [WikiMedia Commons](#)

Mágneslemez típusok

- **ATAPI-7 (SATA, serial ATA, soros ATA)**
 - 7 vezetékes kábel, soros átvitel
 - Hotplug
 - Jobb légáramlás
 - 150 MB/mp kezdeti sebesség → 1,5 GB/mp
 - 5 Volt helyett 0,5 Volt feszültségigény → kisebb fogyasztás



Kép forrása: [WikiMedia Commons](#); Photo by Bas Bloemsaat

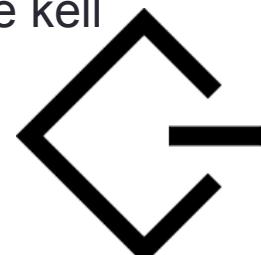


Képek forrása: [WikiMedia Commons](#)

Mágneslemez típusok

- **SCSI**

- Small Computer System Interface
 - „kis számítógép-rendszerek interfésze”, kiejtése „szkazi”
 - Cilinderek, sávok, szektorok, de más interfész
 - Sokkal gyorsabb adatátvitel
 - De sokkal drágább (volt), mint az ATAPI
 - Felépítése
 - Sín + vezérlő + max. 7 eszköz (széles SCSI esetén 15 eszköz)
 - A sín „átmegy” az eszközökön
 - az eszközöknek van egy bemenő és egy kimenő csatlakozója
 - A visszaverődő jelek kiszűrése miatt az utolsó eszközön a sínt le kell zárni
 - minden eszköznek 0-6 (14) közötti azonosítója van
 - Egyszerre több eszköz is aktív lehet (EIDE: csak egy).



SCSI logó

Mágneslemez típusok

- **SCSI**

Néhány SCSI típus

Név	Adat bitek	Sín MHz	MB/mp
SCSI-1	8	5	5
Fast SCSI	8	10	10
Wide Fast SCSI	16	10	20
Ultra SCSI	8	20	20
Wide Ultra SCSI	16	20	40
Ultra2 SCSI	8	40	40
Wide Ultra2 SCSI	16	40	80
Ultra3 SCSI	8	80	80
Wide Ultra3 SCSI	16	80	160
Ultra4 SCSI	8	160	160
Wide Ultra4 SCSI	16	160	320

Forrás: Tanenbaum

SCSI csatlakozók



Kép forrása: [WikiMedia Commons](#)

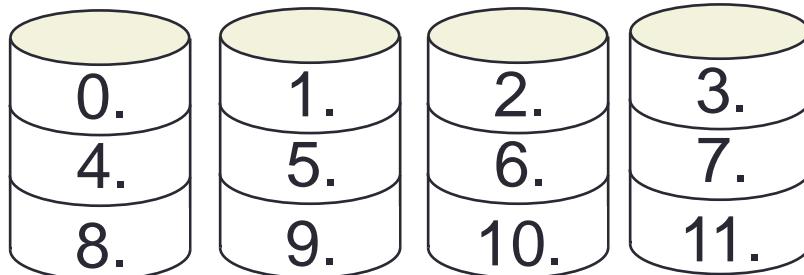
Photo: Rainer Knäpper, Free Art License
(<http://artlibre.org/licence/lal/en/>)

Mágneslemez típusok

- **RAID** (Redundant Array of Inexpensive Disks)
 - Olcsó lemezek redundáns tömbje, 1988
 - Ipar: olcsó → független (inexpensive → independent)
 - Ellentéte: SLED (Single Large Expensive Disk), egyetlen nagy drága lemez
- **Lényege**
 - Több merevlemez egységbe foglalása
 - SCSI alkalmazása a párhuzamossága miatt
 - A rendszer felé egy nagy lemezként jelenik meg
 - Az adatok a lemezeken szétosztásra kerülnek
 - A redundancia javítja a megbízhatóságot
- **Többféle szervezési mód (RAID szintek)**
 - RAID 0; ...; RAID 6
 - Megvalósítása lehet hardveres vagy szoftveres

RAID 0

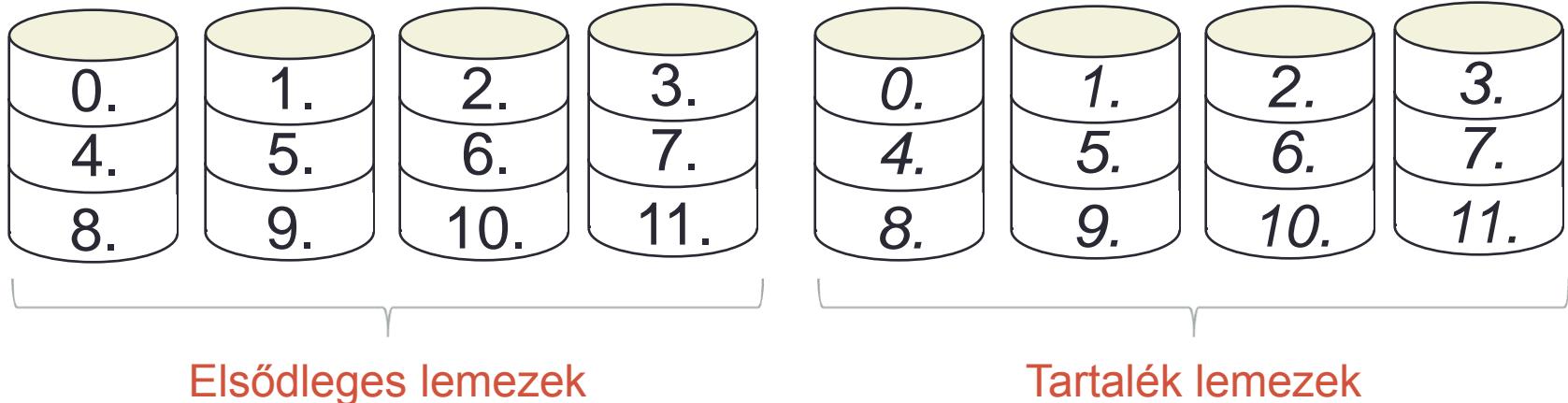
- **Adatok párhuzamos tárolása a lemezeken**
 - k darab szektorból álló **csíkok** (stripes)
 - Csíkok egy-egy lemezen tárolódnak
- **Nincs hibajavítási képessége, nem „igazi” RAID**
 - Sebességen javít
- **Nagyméretű blokkokkal működik legjobban**



Csoportok
Csíkozás (striping)

RAID 1

- **Adatok írása két példányban (két különböző lemezre) csíkozással**
 - 4 elsődleges és 4 tartalék lemez
 - Írás nem gyorsabb, mint a SLED esetén
- **Olvasás párhuzamosítható**
 - Egyes szektorok az elsődleges, mások a tartalék lemezekről
- **Hibatűrés**
 - Hibás lemezegység cserélhető, csak rá kell másolni a „párja” tartalmát



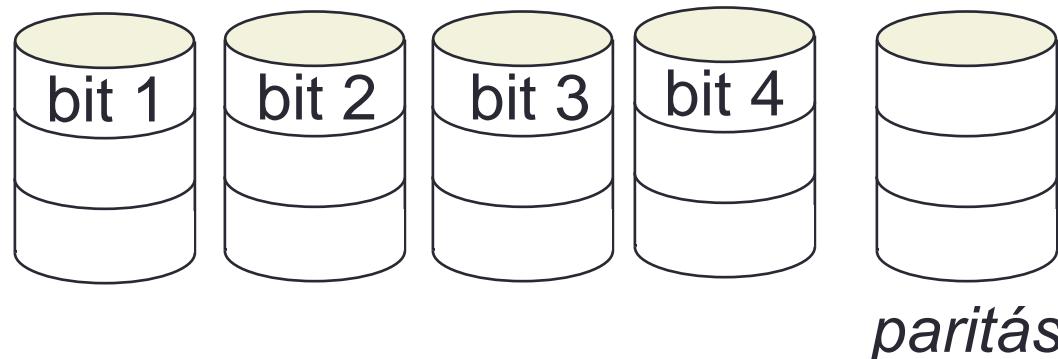
RAID 2

- **Bájt- vagy szó-alapú tárolás** (szektorcsoportok helyett)
- A lemezeknek és a karoknak **szinkronban kell mozogniuk**
- **Adat + Hamming kód** bitjeinek egyidejű tárolása külön lemezeken
 - Pl. 4 adatbit + 3 paritásbit = 7 tárolandó bit; 7 szinkron lemez kell
 - Pl. 32 bites adatszóhoz 6 paritásbit (19% a redundáns adat); 38 lemez
 - 1 szektornyi adat írásideje alatt 32 érdemi szektor írható!
- **Sok merevlemez esetén használható jól**
- **Vezérlőnek plusz munka a Hamming kód kezelése!**
- **Hibatűrés**
 - Hamming kóddal pótolható a kieső lemez tartalma



RAID 3

- **A RAID 2 egyszerűsített változata**
 - minden adatszóhoz egyetlen paritásbit
 - Paritásbit tárolása dedikált lemezegységen
 - Itt is szükséges a lemezek szinkron kezelése
- **Hibatűrés**
 - Javításra is alkalmas, ha tudjuk, hogy melyik lemezegység romlott el
 - Egyszerre maximum 1, tehát cseréljük gyorsan!



RAID 4

- **Csíkozással dolgozik**

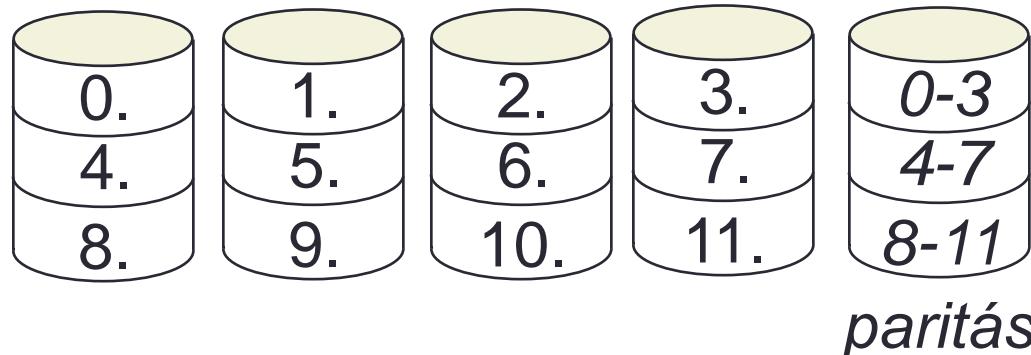
- Nem szükséges a lemezegységek szinkron kezelése
- Csíkonkénti paritást felírja egy dedikált *paritás lemezegységre*

- **Hibatűrés**

- Kieső meghajtó tartalma előállítható a paritásmeghajtó segítségével

- **Probléma**

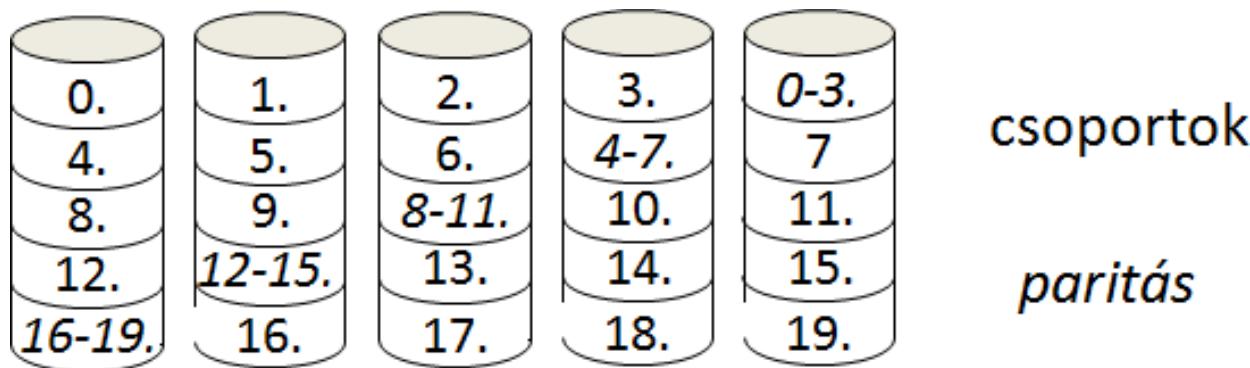
- Íráshoz olvasni is kell minden lemezről
- Nagyon leterheli a paritásmeghajtót



csoportok

RAID 5

- RAID 4-hez hasonló elv, de nincs dedikált paritásmeghajtó
- A paritásbiteket körbejárásos módszerrel szétosztja a lemezegységek között
- Legalább 3 lemezegység kell, legalább 4 ajánlott



RAID alkalmazása

- **Jellemzően a RAID 0, RAID 1 vagy RAID 5 használatos**
- **Kombinált megoldások is lehetségesek**
 - RAID 0+1 (4 meghajtó: 2 összefűzve, 2 másikra tükrözve)
 - Meghajtó kiesés esetén a tömb kiesik
 - RAID 1+0 (4 meghajtó: tükrözés, majd tömbök összefűzése)
 - Meghajtó kiesés csak a tömbön belül jelentkezik (jobb megoldás)
 - RAID 5+0
 - RAID-5 kombinálása a tükrözéssel

SSD (Solid-State Disk)

- **Nem felejtő memória** (1980-as évek)
 - Áramellátás megszűnésével is megmarad a tárolt adat
 - Nincs mechanikus alkatrész
 - Gyorsabb: nincs keresési idő, lemezre várakozás (2-3x sebesség)
 - Kevésbé érzékeny a fizikai behatásokra
 - Sokkal csendesebb
 - Viszont jelenleg fajlagosan sokkal drágább, mint a merevlemezek

SSD (Solid-State Disk)

- **Nem felejtő memória (1980-as évek)**
 - Működési elve
 - NAND flash lapkák (adattárolás) + vezérlőelektronika (írás/olvasás vezérlése)
 - Bitek tárolása tranzisztorral (vezérlő és követő kapu)
 - Oxidréteg: töltés megtartása a követőkapuban, ha a vezérlőkapu már nincs feszültség alatt
 - Követőkapun áthaladó feszültség mérése adja a bit értékét
 - Cellák törlése
 - Követő kapun átvezetett „nagy” feszültség (12 Volt)

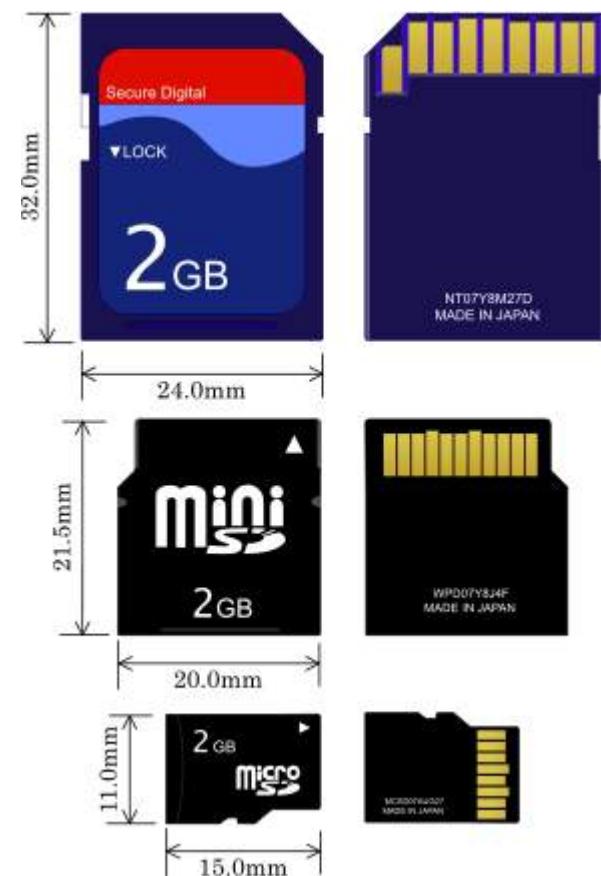
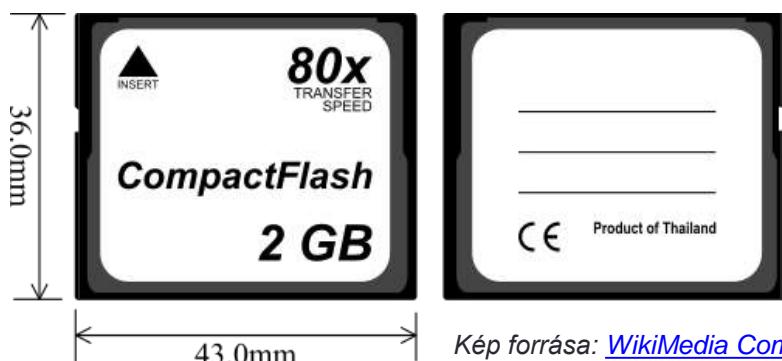
SSD jellemzők

- **Hátrányuk**
 - Rövidebb élettartam, mint a merevlemezeknél
 - Egy cella kb. 100 ezerszer írható (régi SSD-k esetén)
 - Elhasználódás kiegyenlítés (wear leveling) szükséges lehet
 - Írás műveletek egyenletes elosztása a cellákon
 - Bonyolultabb adminisztráció
- **A technológia fejlődésével megbízhatóságuk manapság már jó**
- **Típusaik**
 - SLC (Single Level Cell)
 - Egy cella 1 bit értéket tárol
 - MLC (Multi Level Cell), TLC (Triple Level Cell)
 - Egy cella több bitnyi információt képes tárolni
 - 3D V-NAND

Flash kártyák

- **Flash memória használata**

- Nem felejtő adattároló
 - CF-kártya, SD kártyák, pendrive, ...
- SSD-hez hasonlóak, de ...
 - Az SSD-k tartósan be vannak építve a számítógéphez
 - A memóriakártyák cserélhetők
 - Lassabb az adatátvitel
 - Jóval kisebb méretűek
 - Kevesebbszer újraírhatók, mint az SSD-k

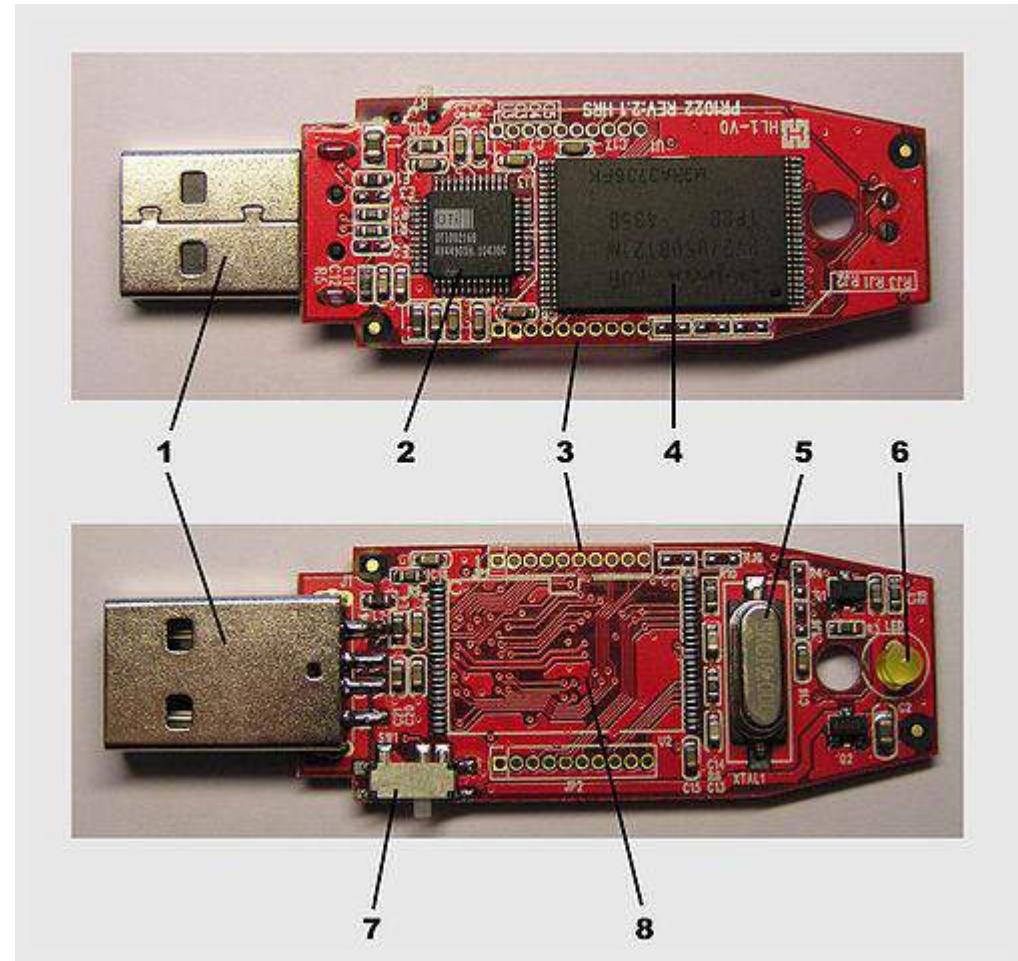


Kép forrása: [WikiMedia Commons](#)

Flash memóriák

- „Pendrive”
 - NAND flash tároló

1	USB csatlakozó
2	USB vezérlő lapka
3	Teszt
4	Flash memória
5	Kristályoszcillátor
6	LED (opcionális)
7	Írásvédelem (opcionális)
8	Szabad hely újabb Flash memóriának



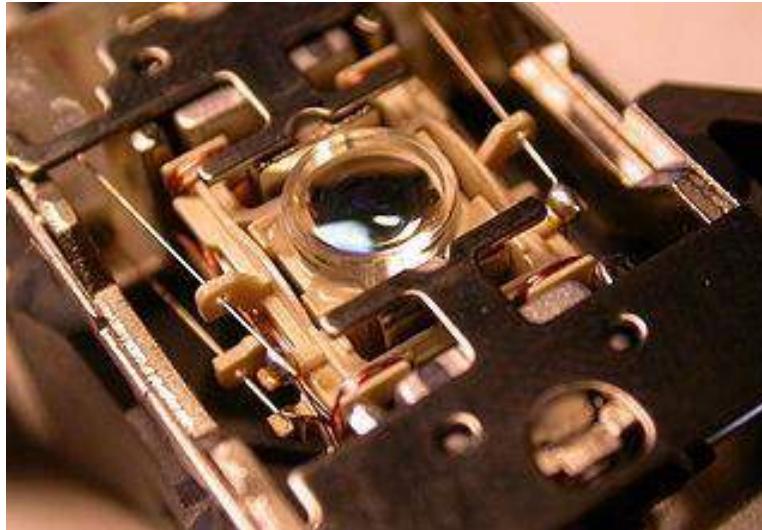
Kép forrása: [WikiMedia Commons](#)
Photographs taken and annotated by [John Fader](#)

Optikai lemezek

- **LaserVison, 1978**
 - Mozifilmek tárolására, de a VHS lett népszerű
 - 30 cm-es átmérő
- **CD (Compact Disk)**
 - Philips és Sony, 1980
 - Hanganyagok tárolására
- **CD-ROM**
 - Számítógépes adattárolásra
- **CD-R**
 - Írható CD
- **CD-RW**
 - Újraírható CD
- **DVD**
 - Mozifilmek, adatok tárolására
 - Egy/kétoldalas, egy/kétrétegű
 - Rétegenként 4,7 GB adat
- **Blu-Ray**
 - DVD leváltására
 - Egy/kétoldalas
 - Oldalanként 25 GB

Optikai lemezek

- **CD** (CDDA – Compact Disk Digital Audio)
 - Philips és Sony, 1980
 - IEC 60908 számú szabvány (Red Book, Vörös könyv)
 - Hanganyagok tárolására
 - 120 mm átmérő, 1,2 mm vastagság, középen 15 mm-es lyuk
 - 100 éves becsült élettartam



Optikai lemezek

- **CD írás folyamata**

- Üveg mesterlemez: írás nagy energiájú lézerrel
 - Üreg (*pit*, $\varnothing=0,8 \mu\text{m}$, $\frac{1}{4}$ hullámhossznyi mély) – szint (*land*).
 - A mesterlemezről negatív öntőforma készül
 - A negatív öntőformába olvadt polikarbonát gyantát öntenek
 - Megszilárdulás után tükröző alumínium réteget visznek rá
 - Védő lakk réteggel vonják be és rányomtatják a címkét

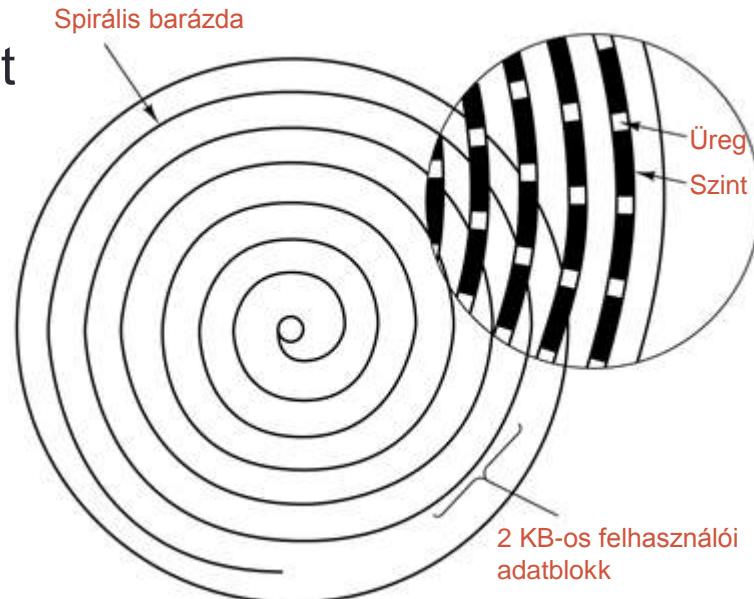
- **CD olvasás folyamata**

- Olvasás kis energiájú infravörös lézerrel ($0,78 \mu\text{m}$ hullámhossz)
 - Az üregből visszavert fény fél hullámhossznyival rövidebb utat tesz meg, mint az üreg pereméről visszavert, ezért gyengíteni fogják egymást

Optikai lemezek

- **Audio CD adattárolása**

- Spirál alakban, belülről kifelé haladva, kb. 5,6 km hosszú
- A jel sűrűsége állandó a spirál mentén
- 74 percnyi anyag (Beethoven IX. szimfóniája kiadható legyen)
 - Első album: Abba: Visitors, de jobbára klasszikus zenék kezdetben
- Állandó kerületi sebesség biztosítása (120 cm/mp)
 - Változó forgási sebesség
- Üreg → szint vagy szint → üreg átmenet
 - 1 érték
- Átmenet hiánya
 - 0 érték
- Hibajavítás
 - Nincs
 - Nem gond, ha néhány bit elvész az adatfolyamból (audio anyag)



Optikai lemezek

- **CD-ROM**

- Philips és Sony, 1984
- Digitális adattárolásra (Yellow book – Sárga könyv)
- Többszintű hibajavítás bevezetése
 - A hanggal ellentétben itt nem lehet adatvesztés!
- 650 MB tárolható, 28%-os kihasználtság



Szimbólum: 1 bájt ábrázolása 14 biten

Keret: 42 szimbólum

Szektor: 16 bájt fejléc + 98 darab keret

Fejléc: 00FFFFFFFFFFFFFFFFFF00

(12 bájt) + 3 bájt szektor sorszáma
+ 1 bájt mód

mód = 1: 2048 adat + 288 ECC bájt,

mód = 2: 2336 adat bájt.

ECC: Error Correction Code (Reed-Solomon)
hibajavító kódolás



Forrás: Tanenbaum

Optikai lemezek

- **CD-ROM**
 - Pozícionálás olvasáskor
 - Nehezebb feladat, mint merevlemeznél
 - Itt spirál van koncentrikus körök helyett
 - Meghajtó szoftvere nagyjából a célterület fölé viszi az olvasófejet
 - Fejlécet keres, abban ellenőrzi a szektor sorszámot
 - Forgási sebesség
 - 1x (75 szektor/s) – 32x
 - Keresési idő
 - Több 100 msec, sebesség < 5 MB/sec
- **Multimédiás kiterjesztés**
 - Green book (Zöld könyv), 1986
 - Egy szektoron belül vegyes adattárolás (hang, videó, adat)

Optikai lemezek

- **CD-ROM fájlrendszer**

- High Sierra formátum, **ISO 9660**
- **1-es szint** (legmagasabb kompatibilitás)
 - Maximum 8 karakteres könyvtárnevek vagy fájlnevek + 3 karakteres fájlkiterjesztés (MS-DOS konvenció)
 - Csak nagybetűk, számok és aláhúzás karakter
 - Alkönyvtárak legfeljebb 8 mélysgig ágyazhatók egymásba
 - Állományok elhelyezkedése folyamatos legyen (a spirál mentén)
- **2-es szint** (kisebb kompatibilitás)
 - 32 karakter hosszúságú fájlnevek használhatók
- **3-as szint** (kisebb kompatibilitás)
 - Nem folytonos elhelyezkedésű fájlok megengedettek
- **Rock Ridge** kiterjesztés
 - UNIX számára hosszú fájlnevek, UID és GID értékeket, szimbolikus linkeket

Optikai lemezek

- **CD-R**

- CD-ROM-okhoz hasonló polikarbonát felépítés
- Saját író berendezéssel rögzíthető az adat

- **Újdonság**

- Író lézernyaláb
- 0,6 µm szélességű **barázda** a spirálon az író lézernyaláb irányítására
- Barázda 0,03 µm-es szinusz hullámú kilengése 22,05 kHz frekvenciával
→ visszacsatolás forgási sebesség ellenőrzéséhez/korrekciójához
- Alumínium helyett arany felület
- Üregek és szintek helyett festékréteg alkalmazása
 - Kezdetben átlátszó a festékréteg (cianin (zöld) vagy ftalocianin (sárgás))
 - Írás
 - a nagy energiájú lézer roncsol → sötét folt marad véglegesen
 - Olvasás
 - az ép és a roncsolt területek detektálása



Optikai lemezek

- **CD-R**

- Orange Book (Narancssárga könyv), 1989
- CD-ROM XA specifikáció
 - Adatok több részletben való felírhatósága
 - Sáv (track): egyszerre felírt sávok
 - minden sávot megszakítás nélkül, folyamatosan kell kiírni
 - Puffer alulcsordulási hiba!
 - VTOC (Volume Table of Contents): tartalomjegyzék
 - minden sávhoz külön VTOC
 - Mindig az utoljára felírt az aktuális, ami tartalmazhatja az előző sávok adatait is
- Szekció (Session): több sáv együttese
- Másolásvédelem
 - Pl. hibás ECC kódokkal, sávok közötti nem szabványos hézagok, ...
- PhotoCD (Kodak)
 - Előhívott fotók CD lemezre írása
 - Új tekercsek mehettek a régi lemezre, amíg be nem telt
 - Drága volt a CD-ROM lemez!



Optikai lemezek

- **CD-RW**
 - Újraírható optikai lemez
 - Újdonság
 - Más adattároló réteg
 - Ezüst, indium, antimon és tellűr ötvözet
 - Kétféle stabil állapot: kristályos és amorf (más fényvisszaverő képesség)
 - 3 eltérő energiájú lézer
 - Legmagasabb energia: megolvad az ötvözet → amorf
 - Közepes energia: megolvad → kristályos állapot
 - Alacsony energia: anyag állapotnak érzékelése, de meg nem változik



Optikai lemezek



• DVD

- 10 tagból álló konzorcium alkotta meg
- CD koronggal egyező méret
- Nagyobb jelsűrűség
 - Kisebb üreg: 0,4 µm (CD: 0,8 µm)
 - Szorosabb spirál: 0,74 µm rés a sávok között (CD: 1,6 µm)
 - Vörös lézer: 0,65 µm hullámhossz (CD: 0,78 µm)
 - CD-ROM kompatibilitáshoz a másik lézer is kell
- Több adat
 - Egy/két oldalas, egy/két rétegű (4,7 GB – 17 GB)
 - 133 percnyi videó anyag MPEG-2 formátumban tárolható
- Új filmipari funkciók
 - Szülői felügyelet, hatcsatornás hang, képarány dinamikus választása (4:3 vagy 16:9), ...
 - Régiókódok (mozibevételek magasan tartása)

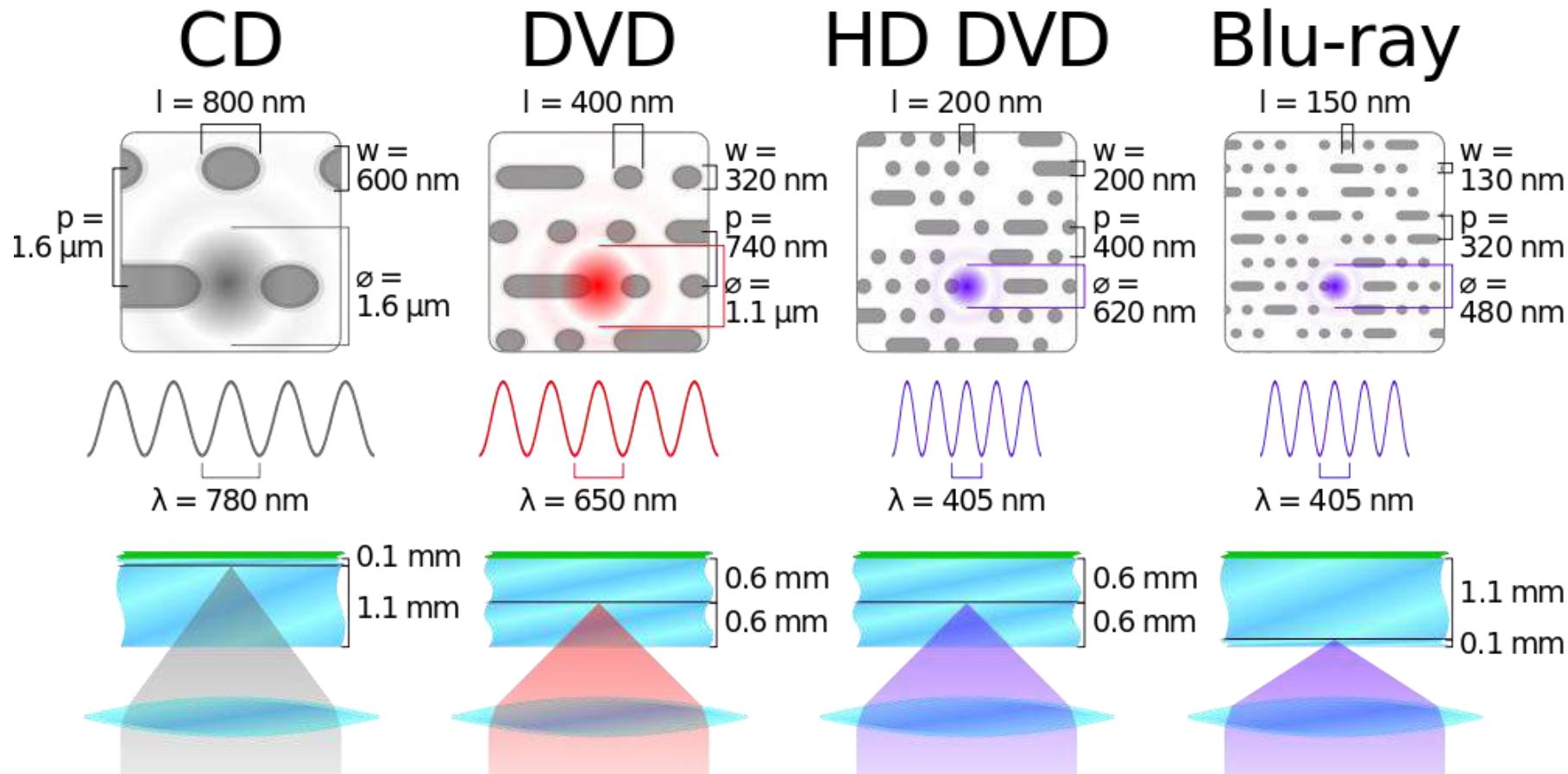
Optikai lemezek



- **HD-DVD**
 - Blu-Ray versenytársa, hasonló tulajdonságokkal
 - Elvesztette a piaci versenyt, megszűnt
- **Blu-Ray**
 - Kék lézer használata a vörös helyett
 - Rövidebb hullámhossz, jobban fókuszálható, kisebb mélyedések
 - 25 GB (egyoldalas) és 50 GB (kétoldalas) adattárolási képesség
 - 4,5 MB/mp átviteli sebesség

Optikai lemezek

- Formátumok tulajdonságainak összehasonlítása



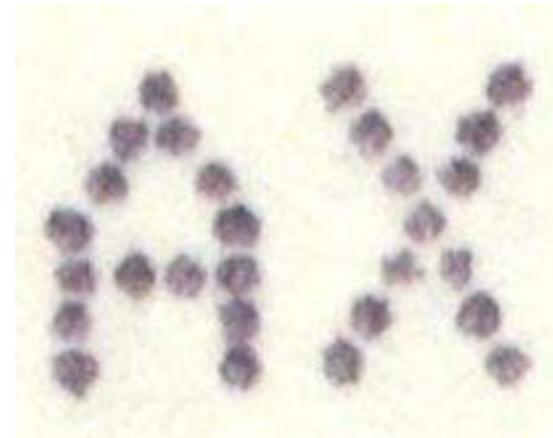
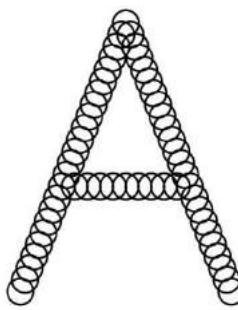
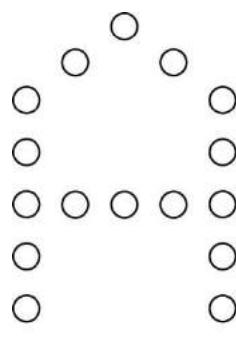
Kimenet/bemenet

- **Nyomtatók**
 - Monokróm, színes
- **Egér**
- **Terminál**
 - Billentyűzet, monitor
- **Telekommunikáció**
 - Modem, DSL, kábel

Nyomtatók

- **Mátrixnyomtatók**

- Monokróm nyomat
- Tintaszalag + elektromágnesesen irányítható tűk
- Olcsó technika, elsősorban cégeknél (volt) jellemző
 - Egészségügy, bérszámfejtés, ...
 - Nagy zaj a mechanikus alkatrészek miatt!
- Pontmátrix karakterek
 - Jobb minőség → több tű és/vagy átfedő körök
 - Egy soron többször is végigfuthat a fej kicsit pozícióban → vastagítás
 - Üzemmódok: sebesség vagy minőség



AX

Nyomtatók

- **Tintasugaras nyomtatók**

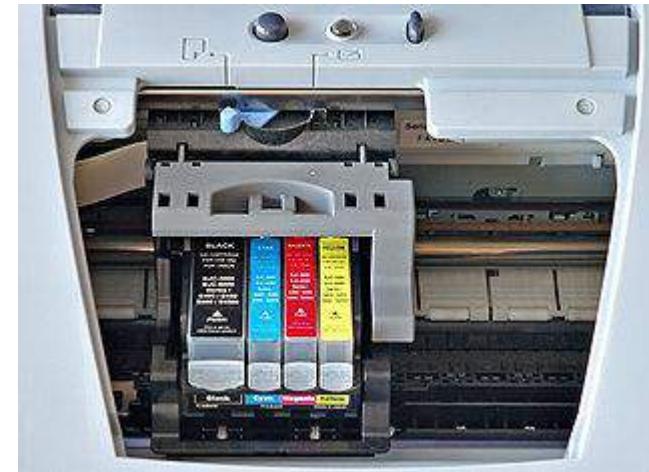
- Elsősorban otthoni használatra
- Lassú, de relatíve olcsó
- 1200 – 4800 dpi (pont/hüvelyk) felbontás

- **Működése**

- Tintapatront tartalmazó, mozgatható fej
- Lapra tintát permetez (1 csepp kb. 1 pikoliter)

- **Fajtái**

- Piezoelektromos
 - Tintapatron mellett kristály, amely feszültség hatására deformálódik → tintacseppet présel ki
 - Nagyobb feszültség → több tinta
- Hővezérlésű vagy festékbuborékos
 - Fúvókákban kis ellenállás, amely feszültség hatására felhevül, a festék felforr és elpárolog, túlnyomás keletkezik, papírra kerül, fúvókát lehűtik, a keletkező vákuum újabb tintacseppet szív be a tartályból

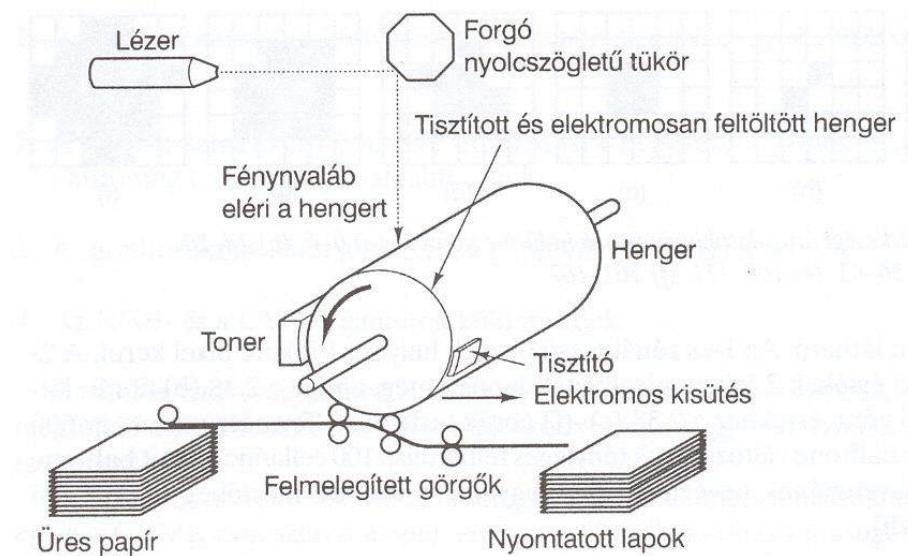


Kép forrása: [WikiMedia Commons](#)

Nyomtatók

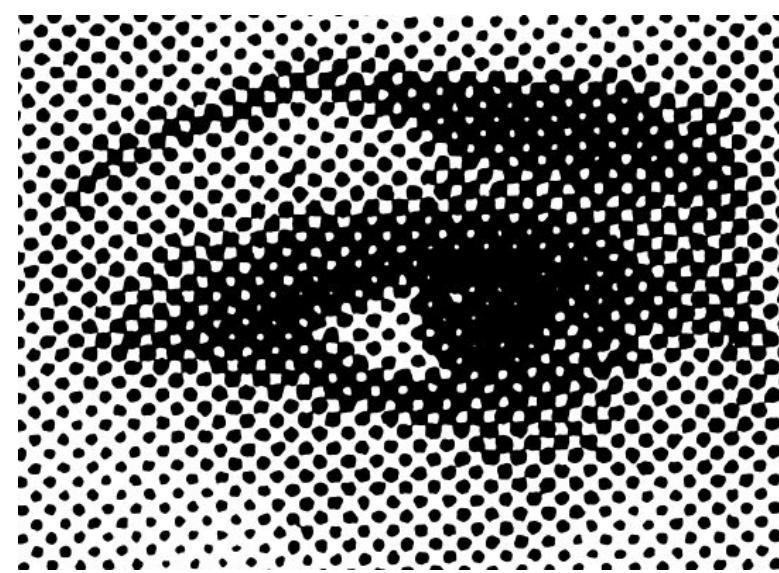
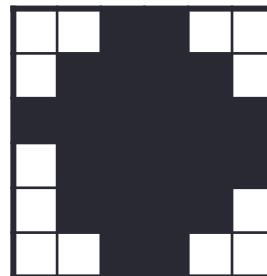
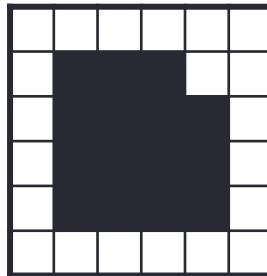
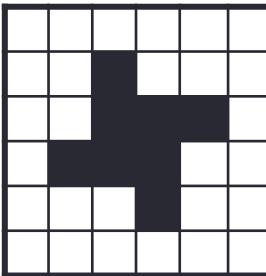
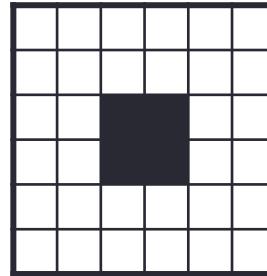
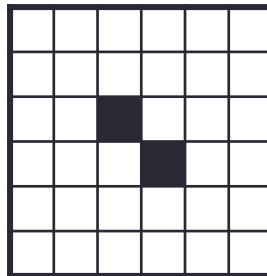
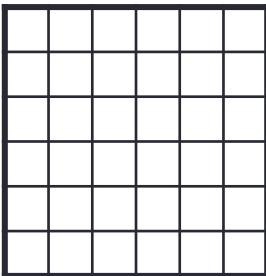
- **Lézernyomtatók**

- Kiváló minőségű kép, gyors működés
 - A fényérzékeny, precíziós 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.
 - Töltés és a maradék por eltávolítása.
- Saját CPU, memória
- Elsőszorban monokróm
- Van színes változat is



Nyomtatók

- „Szürkeárnyalatos” képek nyomtatása (halftoning)
 - A monokróm nyomtatók csak egyszínű nyomtatásra képesek
 - A felbontásuk viszont nagyon jó (kicsi a pontméret)
 - Szürkeárnyalat közelíthető eltérő képmátrix mintákkal
 - Több festéket tartalmaz → sötétebbnek látjuk
 - Példa: 6x6 mátrixok használata



Színképzés

- **Additív színképzés:** RGB komponensek
 - Képernyőknél szokásos, kibocsátott fény
- **Szubsztraktív színképzés:** CMYK komponensek (komplementer színek)
 - Nyomtatásnál, visszavert fényen alapul
 - Cyan (ciánkék), magenta (bíborvörös), yellow (sárga)
 - Black (fekete): elvileg nem kellene, de ...
 - ... nagyon gyakori szín (pl. írásos dokumentumok)
 - ... tökéletes feketét nehéz kikeverni az alapszínekből
- **Gamut**
 - Az előállítható színek összessége
 - Közel sem az összes látható szín „keverhető” ki
 - Komponensek erőssége 0-255 közötti skálán: max. 16 millió árnyalat
 - Technikai tökéletlenségek, festékeloszlás egyenetlensége rontja a helyzetet
 - RGB és CMYK gamutok különböznek!

Színes nyomtatók

- **Tintasugaras nyomtatók**

- Festék alapú: élénk színek, de könnyen fakul
- Pigment alapú: nem olyan élénk, nem fakul
- Papírválasztás fontossága: a jó minőségű nem engedi szétfolyni a festékcseppeket, szebb, akár fotóminőségű képet ad

- **Szilárd tintás nyomtatók**

- Meg kell olvasztani a tintát, néha a bekapcsolás után 10 percig is eltart.

- **Viasznyomtatók**

- Négyszínű viasz tartalmazó szalag, olvasztással kerül a papírra.
- Drága az üzemeltetése.

- **Festék szublimációs nyomtatók**

- Sok fokozatú fűtéssel szublimált CYMK festék kicsapódik a speciális (drága) papírra. Nagyon szép, nem kell half-toning.

- **Lézernyomtatók**

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

Nyomtatók

- **3D nyomtatás**

- Digitális tervrajzokból → 3D tárgy
 - CAD modell, 3D szkenner, ...
- Porrétek + ragasztó komponens
- Jelenleg még drága hardver, de cégek vállalnak 3D modell alapján nyomtatást már Magyarországon is

- **Alkalmazása**

- Prototípusok gyors készítése (ipar!)
- Egyedi tárgyak, objektumok készítése
 - Termékminta, orvosi implantátumok, egyedi ajándék, ...
 - Tárgyak helyett tervezetek küldése nagy távolságokra

- **Megoldandó problémák**

- Jogvédelem, biztonság!

- **Magyar nyelvű blog újdonságokkal**

- <http://freedee.blog.hu/>



Kép forrása: [Desktop Factory](#)



Kép forrása: [WikiMedia Commons](#)
Visual Computing Lab of ISTI-CNR, Pisa, Italy

Egér

• Jellemzői

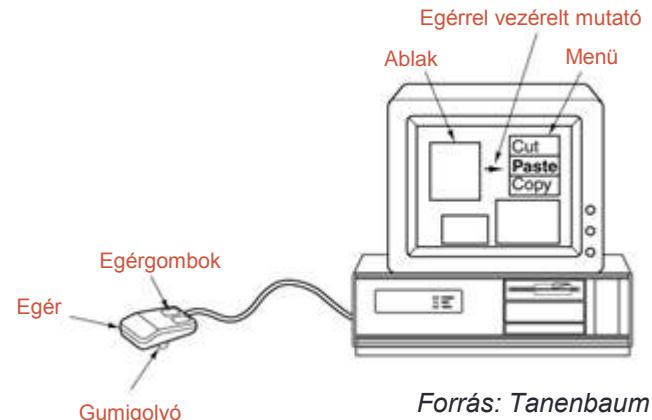
- Grafikus felületen egy mutató mozgatására
- Egy, kettő vagy akár több nyomógomb vagy görgő

• Típusai

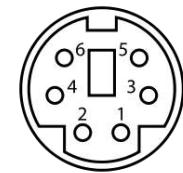
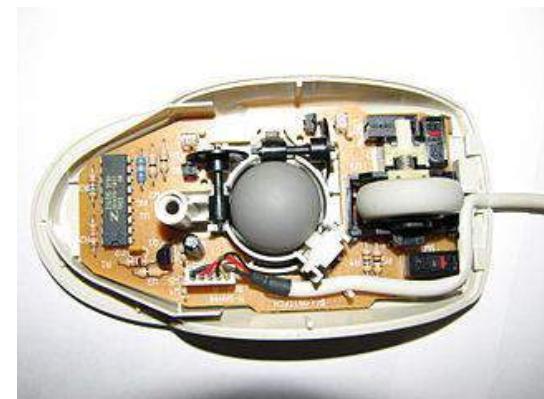
- Mechanikus
 - Kerekek vagy gumi golyó, potenciométerek
- Optikai
 - LED fény, visszaverődés elemzése
- Optomechanikus
 - Golyó, két tengelyt forgat (merőlegesek)
 - Résekkel ellátott tárcsák, LED fény
 - Mozgás hatására fényimpulzusok

• Működése

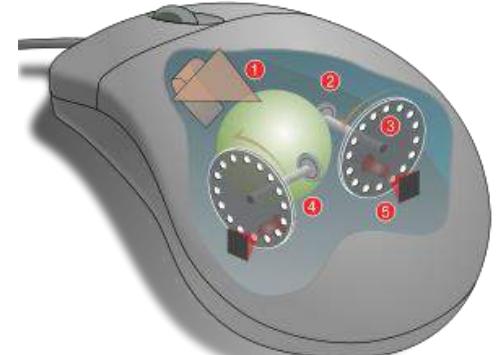
- Bizonyos időnként (pl. 0,1 sec) vagy esemény hatására 3 adatos (általában 3 bájtos) üzenetet küld a soros vonalon (PS-2 vagy USB) a számítógépnek
 - x, y irányú elmozdulás + az egér gombok állapota



Forrás: Tanenbaum



PS-2 soros port



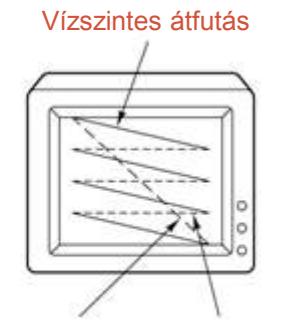
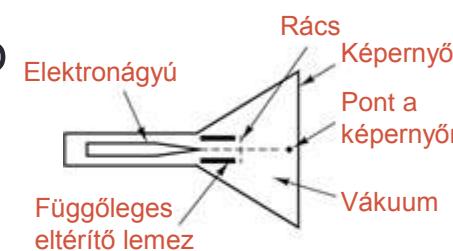
Terminálok

- **Billentyűzet + monitor**
 - Nagyszámítógépek
 - Egybeépített; soros vonalon vagy telefonon kapcsolódik
 - Asztali PC-k
 - Különálló hardver elemek
- **Billentyűzet**
 - Egy-egy billentyű leütése áramkört zár
 - Megszakítás generálódik
 - Az operációs rendszer kezeli és továbbítja a programoknak
 - Kivétel Windows esetén: CTRL + Alt + Del minden az operációs rendszer kezeli

Monitorok

- **CRT (Cathode Ray Tube) – katódsugárcsöves**

- Nagy méret, nagy tömeg – elavult!
- Soronként összeállított raszteres kép
- Másodpercenként 30-60 újraraajzolás
- Működése
 - Elektron ágyú
 - Foszforeszkáló ernyő
 - Vízszintes és függőleges eltérítő lemezek
 - Növekvő feszültség hatására eltérítik a sugárnyalábot
 - Elektromos vagy mágneses elven
 - Rács
 - A képernyőt elérő elektronok mennyiségének szabályzására
 - Feszültség állításával
 - Képpontok létrehozása
 - Színes megjelenítés esetén 3 elektronágyú



Forrás: Tanenbaum

Monitorok

- **LCD (Liquid Crystal Display) – Folyadékkristályos**

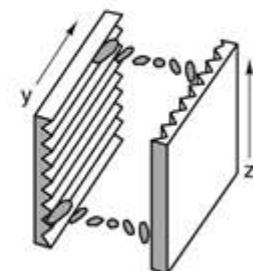
- Hordozható számítógépek, lapos kijelzők, tévék
- Szerves molekulák: folyékonyak + térbeli szerkezet
- Elektromos mezővel a molekulák optikai tulajdonságai állíthatók
- Hátsó megvilágítás
- Polarizált fényszűrők
- **Passzív technika**
 - Vízszintes és függőleges elektródák
 - Sor-oszlop kiválasztása

- **Aktív mátrix megjelenítő**

- Drágább, de szebb képet ad
- Vékonyfilm-tranzisztorok (TFT)
 - Feszültségállítás képpontonként kapcsolva

- **OLED, AMOLED**

- Organic Light Emitting Diodes

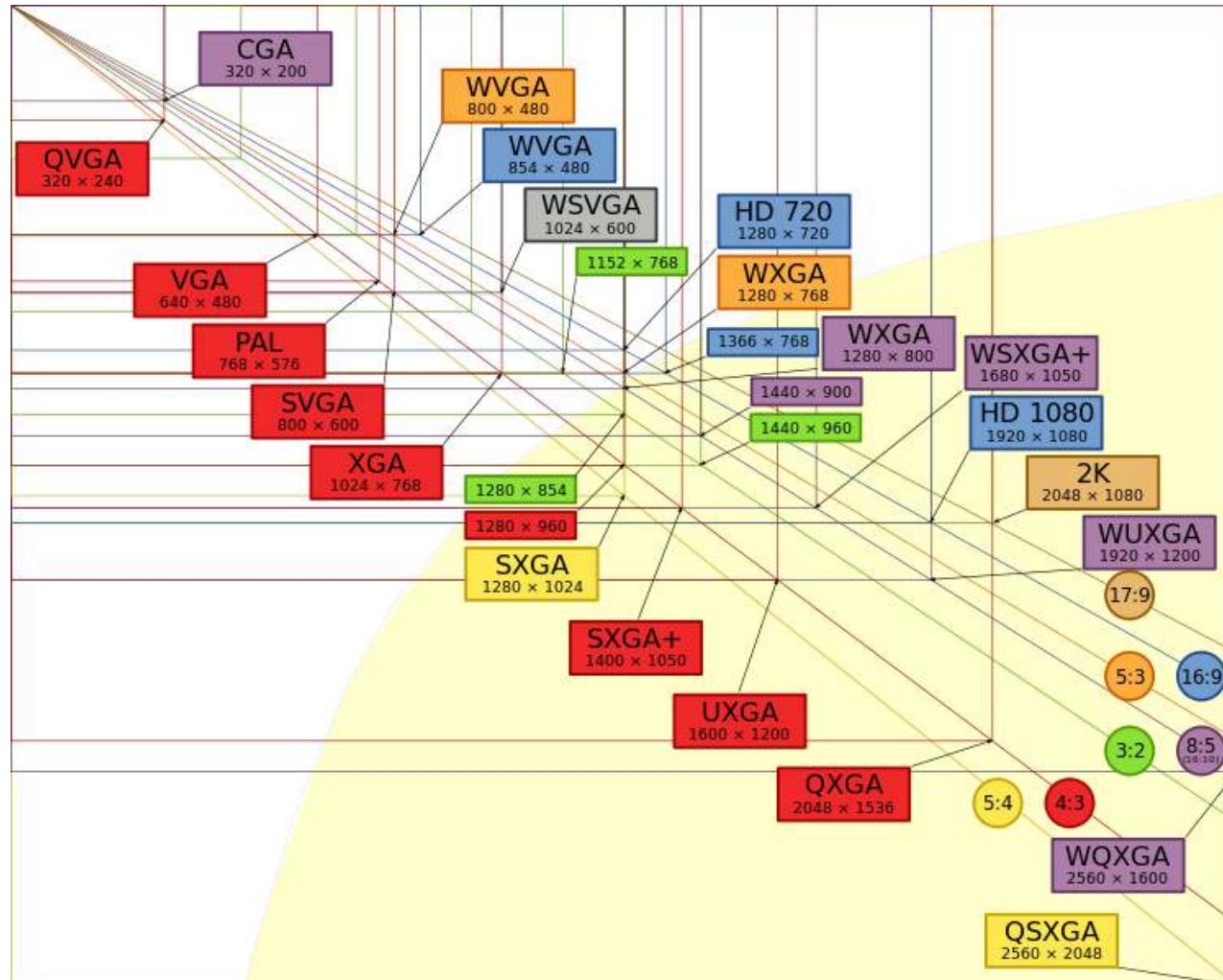


Forrás: Tanenbaum

Monitorok

- **Videó RAM**
 - Képernyővezérlő kártyán vagy központi memória megosztott része
 - **Memória, amely a képernyő tartalmát adja**
 - Karakteresen (ASCII kód) vagy képpontonként (pixel)
 - **Pixel reprezentáció**
 - 3 bájton RGB értékek
 - Indexelt színelőállítás
 - Kisebb tárigény, de csak limitált mennyiségű szín (pl. 256)
 - Régi technika
- **Nagy sávszélesség igény**
 - Másodpercenként 30-60-120 újraraajzolás
 - Képkockánként több MB adat
 - Pl. 1600 x 1200 x 3 ≈ 5,5 MB
 - AGP vagy gyors PCI sín kapcsolat szükséges

Monitor felbontások

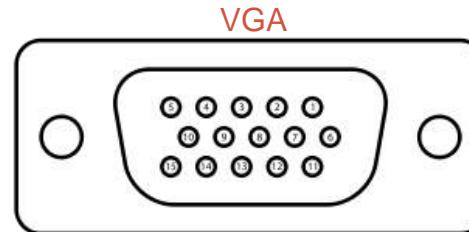


Kép forrása: [WikiMedia Commons](#)

Monitor csatlakozó típusok

• Analóg

- Kompozit, komponens video
- VGA (1987)



• Vegyes

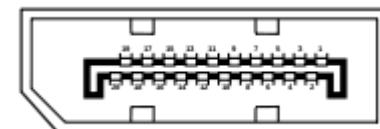
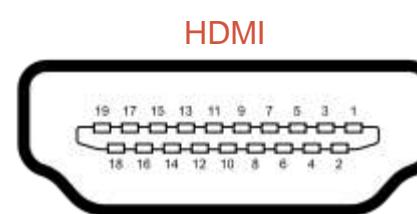
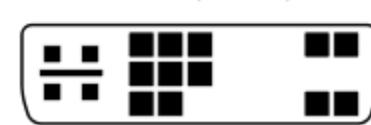
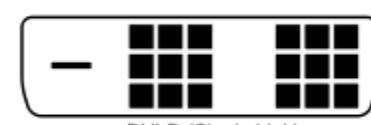
- DVI (1999)
 - DVI-D (csak digitális), DVI-A (csak analóg),
DVI-I (integrált, vegyes)

• Digitális

- HDMI (2002)
- DisplayPort (2006)

• Vezeték nélküli

- WiDi (Intel), Airplay (Apple)
- Miracast (Wi-Fi Alliance): nyílt szabvány, Wi-Fi Direct alapú
 - Okostelefon → TV; Laptop → kivetítő, ...



Képek forrása: [WikiMedia Commons](#); a képekre kattintva elérhetők

Monitor csatlakozó típusok

- **Passzív átalakítók**

- Bizonyos analóg-analóg vagy digitális-digitális típusok között
- Viszonylag olcsók

- **Aktív átalakítók**

- Analóg-digitális átalakítók külön áramkört igényelnek → drágák!

- **Példák**

- DVI-A vagy DVI-I (analóg) ↔ VGA
- DVI-D vagy DVI-I (digitális) ↔ HDMI



Telekommunikációs berendezések

- **Feladatuk**

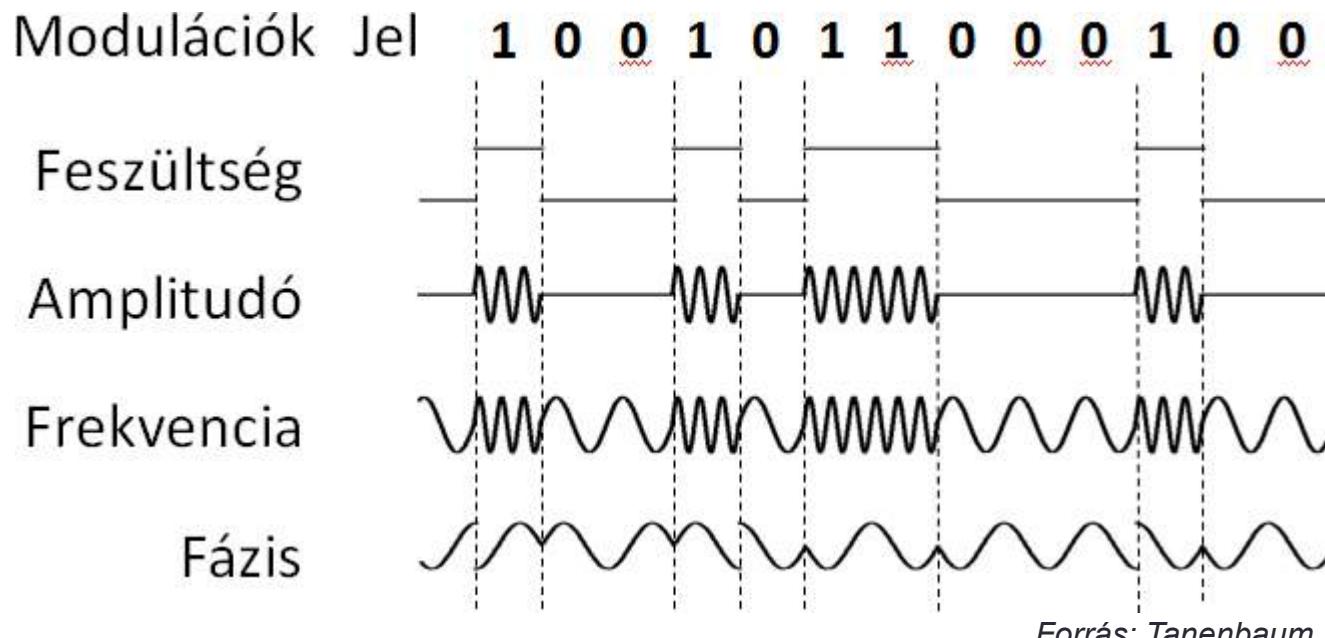
- Számítógépek hálózatba szervezése
- Meglévő telekommunikációs csatornát felhasználva

- **Típusai**

- Modem
 - Normál hang-alapú telefonvonalon
- ADSL
 - Telefonvonalon, de magas frekvenciákon is
- Kábeles internet
 - KábelTV szolgáltatók csatornáin

Modem

- **Adatkommunikáció analóg telefonvonalon**
 - Az analóg vonalat hangátvitelre találták ki
 - Adatátvitelhez: vivőhullám (1000-2000 Hz-es szinusz hullám)
 - A bitek csak sorosan, egymás után vihetők át
 - 1 bájt átvitele: start bit + 8 adatbit + stop bit = 10 bit



Forrás: Tanenbaum

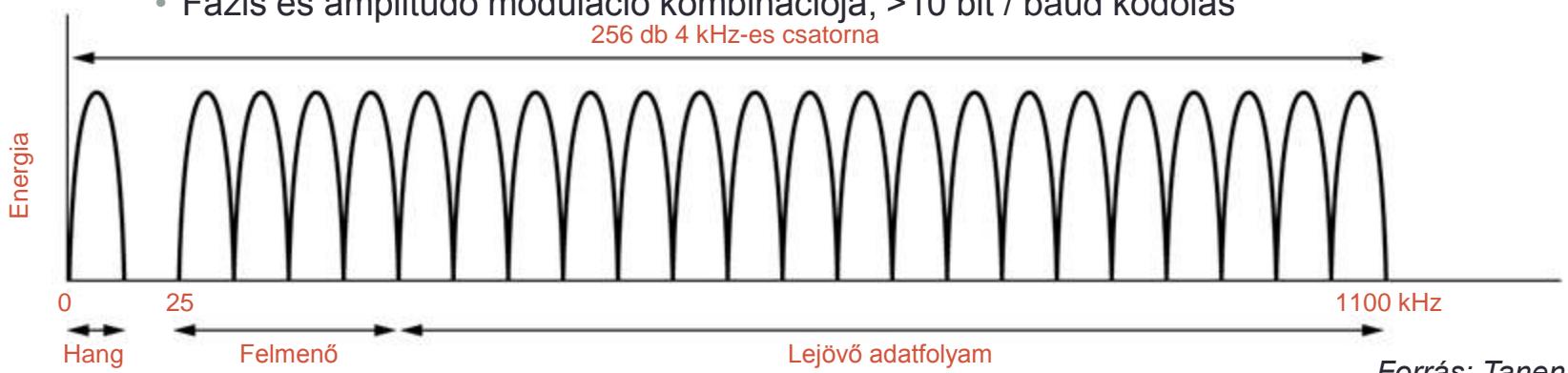
Modem

- **Modulációk**
 - Amplitúdó, frekvencia
 - Fázis: dabit kódolás
 - 45, 135, 225 és 315 fokos fáziseltolódások az időintervallumok elején
 - 2 bit átvitele egységnyi idő alatt (45 fok: 00, 135 fok: 01, ...)
 - Kombinálva is használhatók
- **Definíciók**
 - **Baud:** Jelváltás / másodperc
 - 1 jelváltás több bitnyi információt is hordozhat (lásd dabit kódolás)
 - **Adatátviteli sebesség:** bit / másodperc
 - Jellemzően 28800 vagy 57600 bit / mp, jóval alacsonyabb baud értékkel!
 - **Kommunikációs vonal típusa**
 - Full-duplex (kétirányú kommunikáció egyidőben), fél-duplex (egyszerre csak 1 irányban), szimplex (egyirányú kommunikáció lehetséges csak)

ADSL

- **Szélessávú adatforgalom analóg telefonvonalon**

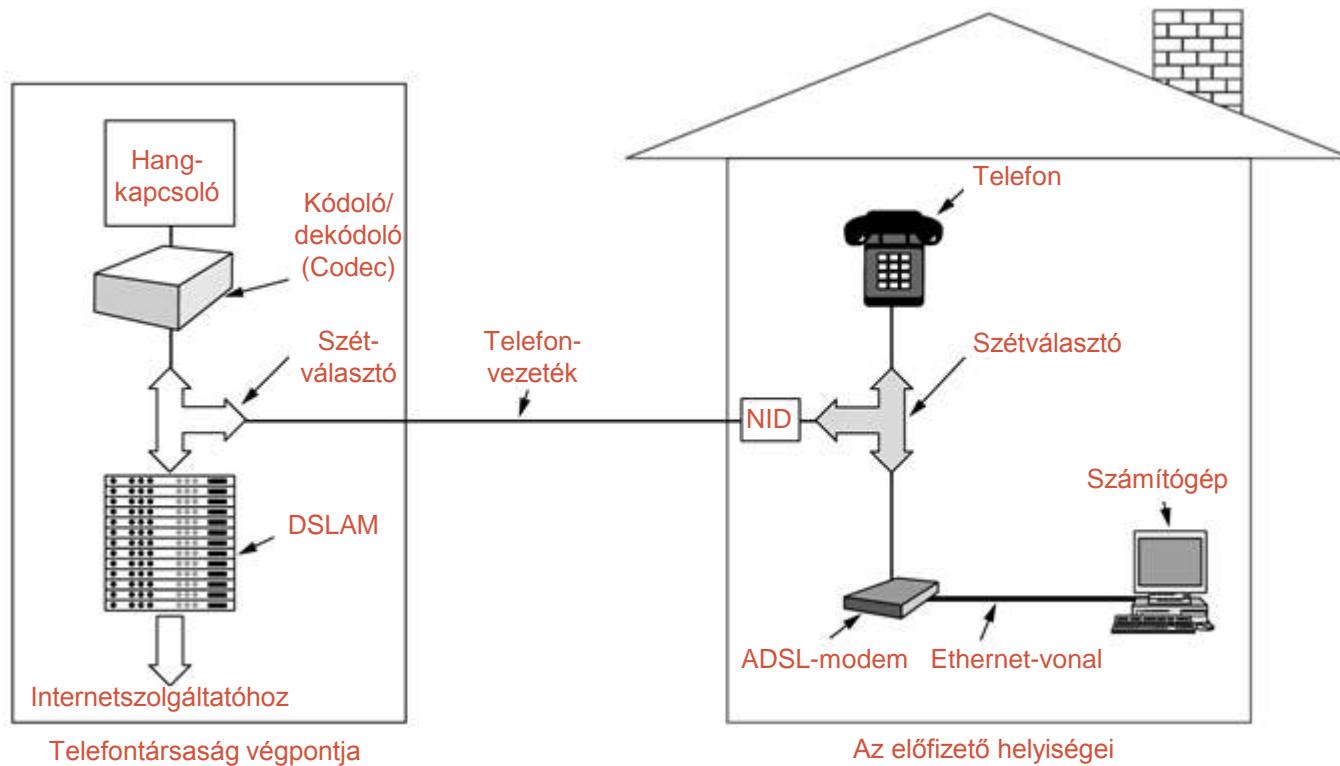
- Hangátvitel: 3000 Hz-es szűrő alkalmazása a vonalon
- DSL technika: 1,1 MHz méretű tartomány használata
 - 256 darab 4 kHz-es csatorna
 - Szétválasztó (splitter)
 - Az alsó tartomány leválasztása hangátvitelre: 0. csatorna
 - A felső tartomány az adatátvitelé: 4-8 Mbps sebesség
 - 1-5. csatornák nem használtak (ne zavarja a hangátvitelt)
 - Két vezérlő csatorna a le- és feltöltés vezérlésére, a többi az adatátvitelre (nem mind!)
 - Különböző számú le- és feltöltő csatorna (több a letöltésre – aszimmetria)
 - Fázis és amplitúdó moduláció kombinációja, >10 bit / baud kódolás



Forrás: Tanenbaum

ADSL

- Tipikus hálózati konfiguráció



Kábeltévés internet

- **Kábeltévé társaságok**
 - Fő telephely + fejállomások
 - Fejállomások üvegkábelben a fő telephelyhez kapcsolódnak
 - A felhasználók felé induló vonalakon sok eszköz osztozik!
 - Kábelek sávszélessége 750 MHz körüli
 - A sávszélesség függ a felhasználók pillanatnyi számától!
 - ADSL: független, minden felhasználó saját (1,1 MHz) vonalon kapcsolódik
 - Bonyolultabb kommunikáció a fejállomás és az előfizetői eszközök között
 - Sávkiosztás
 - 54 – 550 MHz: TV, rádió (lejövő frekvenciák)
 - 5 – 42 MHz: felmenő frekvenciák adatfeltöltésre és vezérlésre
 - Vezérlés: kétirányú, konkurens kommunikáció!
 - Ütemezési stratégia szükséges! Kis méretű, ún. *minislot*-ok használata.
 - 550 – 750 MHz: lejövő frekvenciák adatletöltésre
 - Egyirányú kommunikáció (fejállomás → előfizető), nagyobb adatcsomag-méret.
 - Aszimmetrikus adatkommunikáció!
 - Szükséges eszköz
 - Kábelmodem

Digitális kamerák

- **Digitális képalkotás**
 - 3D színtérről 2D vetített kép
 - Számítógépre feltölthető
 - Digitális kamerák, mobiltelefonok, webkamerák, ...
- **Kamera felépítése**
 - Objektív (lencserendszer)
 - Fény irányítása a képérzékelőre
 - Képérzékelő
 - Fényből elektromos jel/töltés
 - CCD: olcsóbb; de zajosabb, gyengébb minőség; alacsony energiaigény
 - CMOS: drágább; jobb kép; nagy energiaigény
 - D/A átalakító
 - töltöttség függvényében digitális kvantált jel
 - 0-255 (olcsó) vagy akár 0-4095 közötti egész szám (profi)
 - Saját CPU, RAM, Flash háttértár

Digitális kamerák

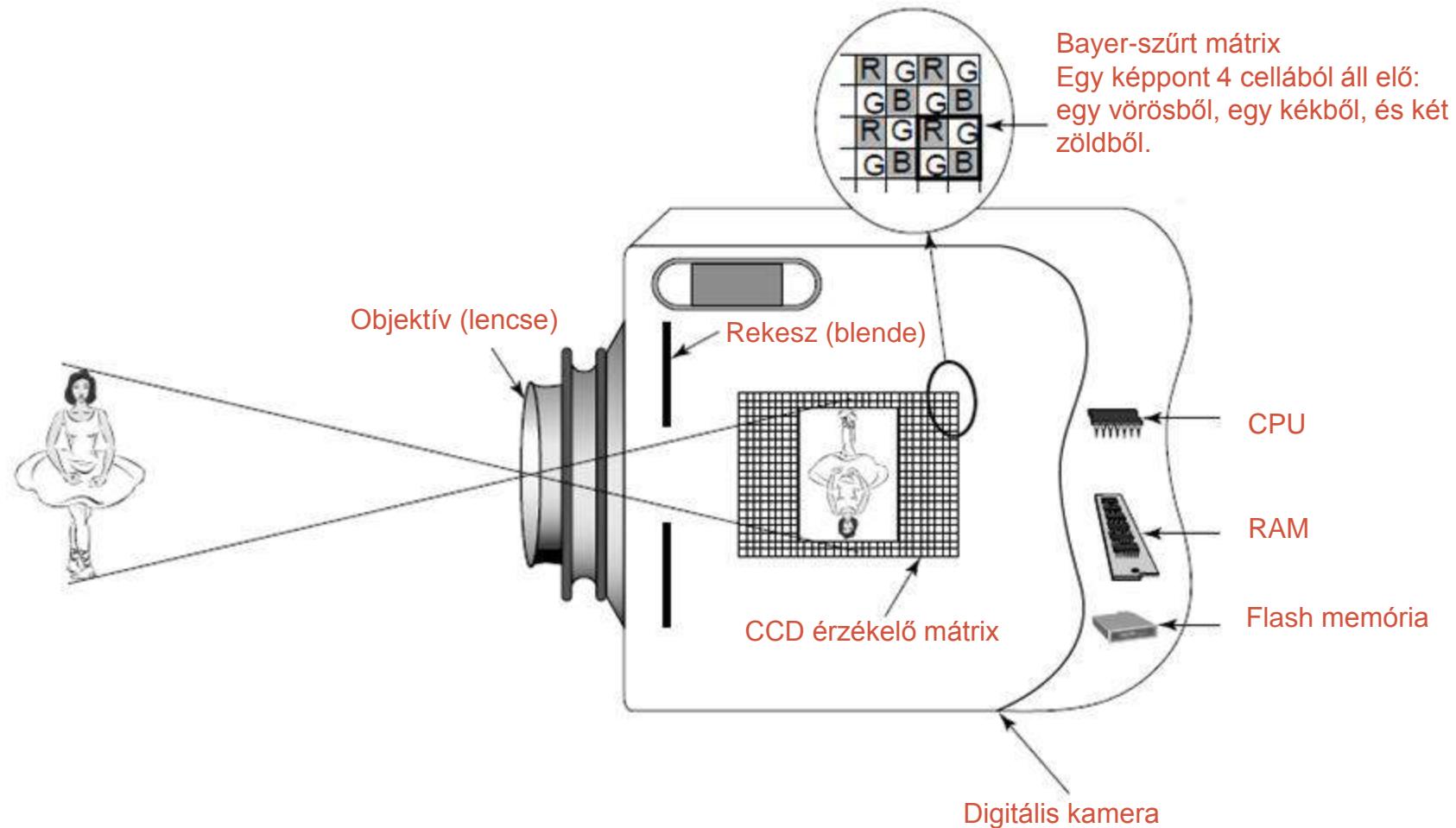
• Képalkotás

- Bayer-szűrő
 - Képérzékelő nem érzékel színt, csak intenzitáserősséget
 - A képérzékelő elé speciális színszűrőt helyeznek
 - 1-1 CCD pont 1 színszűrt értéket mér így
- RGB képpontok előállítása
 - 2x2-es méretű mezőkből: 1 vörös, 1 kék és 2 zöld (az emberi szem a zöld színre a legérzékenyebb)
 - Interpoláció: RGB képmátrix előállítása

• Kép kódolása

- Képmátrix tárolása
 - Megapixel: a képet alkotó képpontok száma (millió képpont egységen)
 - Manapság akár 40 millió képpont is lehet a képen; 8-13-21 a jellemző
 - Nagy helyigény!
 - RAW: nyers mért értékek az érzékelőből (drágább eszközökben)
 - JPEG: nagymértékű, veszteséges tömörítés; állítható mértékben

Digitális kamerák



Digitális kamerák

- **Átvitel számítógépre**

- Kamera összekötése a számítógéppel
 - USB vagy FireWire; manapság akár WiFi is
- Szoftveres utófeldolgozás
 - GIMP, Adobe PhotoShop, ...
 - Képek kivágása, manipulációja
 - Vörös szem effektus korrekciója
 - Fényerősség, kontraszt, fehéregyensúly, színtelítettség, ... állítása
 - Szűrők alkalmazása (élesítés, homályosítás, HDR, ...)

- **Képek felhasználása**

- Számítógépen tárolható és megjeleníthető
- Megosztható interneten
- Papírkép készíthető/nyomtatható
- Adathordozóra írható

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Kiegészítő anyag

Számonkérésre nem kerül!

PC indítási folyamat

Egyéb perifériák

Személyi számítógép típusok

Okos eszközök

Tanács Attila

Áttekintés

- **PC indítási folyamat**
 - BIOS, UEFI
- **Egyéb perifériák**
 - Játékvezérlők, 3D kijelzők, AR-VR-MR
- **Személyi számítógép típusok**
 - PC-k
 - Asztali, laptop, netbook, ultrabook
 - Miniatűr PC-k
 - Raspberry Pi, Arduino
 - Okostelefonok
 - Menedzserkalkulátor, PDA, „buta”telefon, okostelefon, tábla, okosTV
 - Viselhető eszközök (közeljövő)
 - Okosórák, Google Glass

PC indítási folyamat

- **Bekapcsolás**
- **Programszámláló** a BIOS EPROM-ban található gépi kódú rendszerbetöltő programjának az elejére (rögzített cím)
- A bájt értékek által definiált **utasítások végrehajtása**
 - Rendszerteszt
 - Rendszerbetöltő megkeresi a rendszerindító egységet (pl. merevlemez) és annak betöltő programjára „ugrik” (rögzített helyen van)
 - Az tölti az operációs rendszert
 - Az operációs rendszer
 - Tartja a kapcsolatot felhasználóval,
 - Ütemezi a processzusokat,
 - Kezeli az erőforrásokat
 - ...



BIOS (Basic Input/Output System)

- **EPROM-ba égetett rövid kód**

- Rögzített memóriacímen kezdődik, ide ugrik
- Régi technika, valós módban indul a PC!

- **Rendszerteszt**

- Hiba esetén sípolás (memóriahiba, nincs billentyűzet, ...)

- **Rendszerindításra alkalmas háttértárak keresése**

- Az első egység betöltő programjára ugrik
 - Régebben floppy vagy merevlemez; manapság DVD-ROM, pendrive is lehet
 - Nem értelmezi a fájlrendszeret: rögzített szektor + kezdőcím kell!
 - „Boot sector” (indító szektor)
 - Ez az operációs rendszer betöltő része
 - Átvált védett üzemmódba
 - Betölti a rendszer kódját és átadja a vezérlést

- **Operációs rendszer**

- Tartja a kapcsolatot felhasználóval, ütemezi a processzusokat, kezeli az erőforrásokat

PC indítási folyamat

- **BIOS problémák**

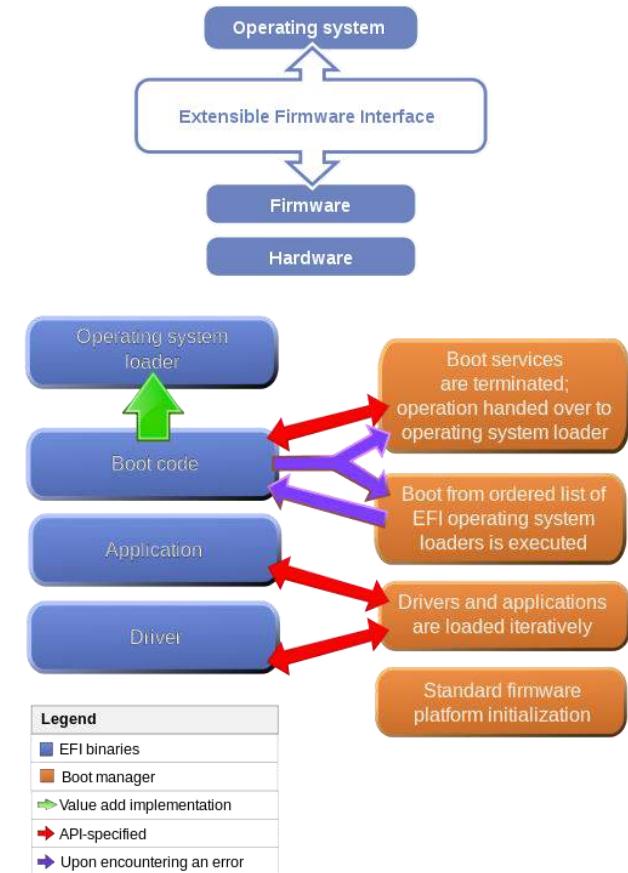
- Nagyon elavult (az 1970-es évek végén még elfogadható volt)
- Valós üzemmódú indulás (1 MB címezhető)
- Rendszervédelem teljes hiánya
 - Nem ellenőrzi, hogy milyen rendszert tölt be
 - Az sértetlen-e? Van-e ott egyáltalán értelmes betöltő kód?

PC indítási folyamat

- **EFI** (Intel, 1998), **UEFI** (Unified Extensible Firmware Interface – 2005)
 - Távoli diagnosztika
 - Számítógép helyreállítása operációs rendszer nélkül
 - Titkosítás, hálózati azonosítás
 - Szebb felhasználói felület
 - Több interfész szint (Class 0 - 3)
 - 0: BIOS, 2: BIOS + UEFI, 3: csak UEFI
 - 2020-tól a tervezet szerint már csak Class 3 lesz!
 - „Boot Manager”
 - fájlrendszeret értelmezi, nincs szükség indító szektorra
 - „Secure Boot”
 - Csak digitális aláírással rendelkező rendszer indítható!
 - Opcionális, de gyártók megkövetelhetik



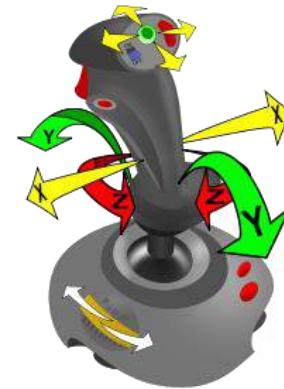
Kép forrása: [TechSpot.com](#)



Játékvezérlők

- **Botkormány („joystick”)**

- Balra, jobbra, előre, hátra, akció



- **Kormány + pedálok**

- Pl. autószimulátor

- **Nintendo Wiimote (2006)**

- Mozgásérzékelő (3 tengelyű gyorsulásmérő) + nyomógombok
 - Valósághű mozgások a vezérlésre (pl. golf, tenisz, teke, ...)
 - Bluetooth kommunikáció konzollal
 - Sony Playstation Move (2009) hasonló



- **Microsoft Kinect (2009)**

- Gesztusvezérlés külső eszköz nélkül
 - Infravörös lézer + távolságérzékelő + mozgásdetekció és -követés
 - Számítógépes látás algoritmusok



3D kijelzők

- **Képkészítés elve**

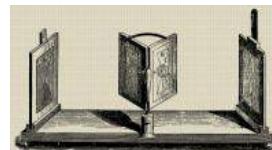
- Sztereó kamerapárral
- Kamerák elhelyezkedése az emberi látást közelíti
 - Kicsit más a nézőpont



- **Megjelenítési lehetőségek**

- **Mechanikus eszközök**

- Biztosítják, hogy a képek a megfelelő szembe kerüljenek
 - Régi technika
 - 1800-as évek



- **Anaglif**

- A két kép egybe montírozása különböző színskálákkal
- Kék-vörös színszűrős szemüveggel látható
- Színek torzulnak



- **Passzív szemüveg**

- Polarizációs technika
- Mozikban szokásos

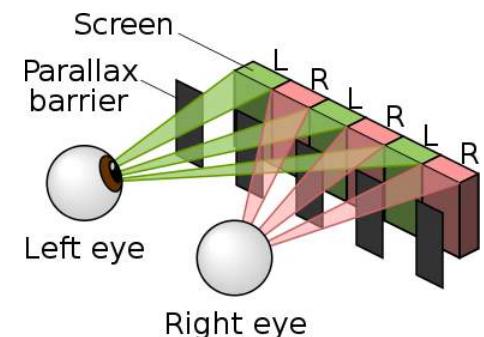


- **Aktív szemüveg**

- 120 Hz-es monitor + videókártya → felváltva jeleníti meg a két képet (60 Hz)
- A szemüveg vezetékkel rá van kötve a készülékre
- Az vezérli, hogy melyik pillanatban melyik szemüveglencse legyen nyitva/takarva
- NVIDIA Stereo Vision, 3D tévék

- **Autosztereoszkópia**

- Rögzített pozícióból szemüveg nélkül érzékelhető 3D hatás
- „Parallax korlát” elv
 - A kijelző elő egy rácson kerül
 - Ez biztosítja, hogy a bal és jobb szem felváltva látja a kijelző pixel oszlopait
 - A vízszintes felbontás feleződik
 - Kis méretű (egyszemélyes) eszközökön



AR-VR-MR fogalmak

- **Kiterjesztett valóság (Augmented Reality – AR)**

- Kamera élőkép + 2D/3D grafika elegyítése
- Elsősorban információk elhelyezésére
- Okostelefonokon jól megvalósítható szenzor információkra építve



- **Virtuális valóság (Virtual Reality – VR)**

- Korábbi koncepció, de manapság új lendületet kapott
- Képernyőn *mesterségesen létrehozott* 3D látvány
- Interaktívan manipulálható, bejárható



- **Kevert valóság (Mixed Reality – MR)**

- Interaktív kiterjesztett valóság
 - Valós látvány + szemüvegre vetített információ
 - Microsoft Hololens (fejlesztés alatt)



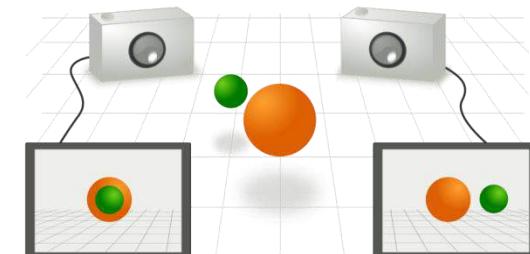
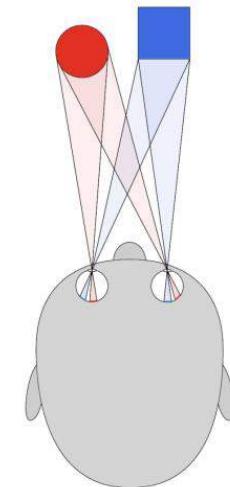
Térlátás működési elve

- **Emberi térlátás alapja**

- Két „képalkotó berendezés”
 - Szem = Lencserendszer + érzékelő
 - Kicsit más nézőpontból képzik le a világot
 - Diszparitás: közeli és távolabbi tárgyak leképződése eltérő
- Fénymérés idegpályákon az agyba, ahol az érzel létrejön

- **VR működési elve**

- Készítsünk két képet, ahogyan a bal és a jobb szemünk látná a tájat
- Juttassuk ezeket a képeket a két retinára
- Kövessük a felhasználót és frissítsük a látványt
 - Fejmozgás
 - Térbeli mozgás



Kép: Arne Nordmann

Virtuális valóság (VR)

- **Speciális szemüveg szükséges**
 - Bal és jobb szemünk előtt eltérő kamerapozícióból generált 3D kép juttatásához
- **Nagyon erős számítási háttérkapacitás kell!**
 - Jellemzően PC + erős GPU
 - Vezetékes kapcsolat a szemüveggel (2016)
- **A fejmozgatásra nagyon gyorsan reagálni kell!**
 - Korai VR próbálkozások legnagyobb hibája a nagy késleltetés volt. Az agy „nem értette” az okát.
- **Látvány realisztikus és folyamatos előállítása**
 - Kieső képkockák, perem menti torzulások hányingert okozhatnak!
 - Idegméreg hatásához hasonló élmény, amire a szervezet reagál)
 - Fárasztó az agynak/szemnek



Virtuális valóság (VR) eszközök

- **PC + vezetékes kapcsolat**
 - Oculus Rift, HTC Vive
 - Sony Playstation VR (Project Morpheus)
- **(Erős) okostelefon a szemüvegben**
 - Samsung Gear VR (Oculus technológia)
 - Google DayDream platform
- **Olcsó alternatíva: Google Cardboard**
 - Pár dollárból kihozható
 - Kartonlapból hajtoghatható doboz az okostelefon tartására
 - Lencsék a látvány szemre vetítéséhez
 - Mágnes nyomógombként
 - **Az okostelefon számítási teljesítménye, szenzorainak minősége és kijelzőjének felbontása befolyásolja az élményt!**



Tartalom

- **3D játékok**

- Évtizedek óta fejlődő ágazat
- Áttérés viszonylag egyszerű
 - Két virtuális kameraképet kell generálni
 - Valós 3D modellezés kell (a 2D közelítő trükkök helyett)
- A népszerű grafikus motorok támogatják a VR-t
 - Unity, Unreal, Cry Engine, ...

- **3D modellezés**

- CAD modellek
- Animációk



Tartalom

- **360 fokos videók**

- Most megjelenő alkalmazási terület!
- A kamera pozíciójából bármely pillanatban szabadon körültekinthetünk
 - Nem feltétlenül 3D a hatás, a fejmozgás követés a fontosabb
- Kamera rendszerek
 - Két halszem optika
 - Több kamera együttese a teljes szögtartományt lefedve
- Youtube, Facebook, StreetView már jelenleg is támogatja



Személyi számítógép típusok

- **PC-k**
 - Asztali, laptop, netbook, ultrabook
- **Miniatűr PC-k**
 - Raspberry Pi, Arduino
- **Okostelefonok**
 - Menedzserkalkulátor, PDA, „buta”telefon, okostelefon, tábla, okosTV
- **Viselhető eszközök**
 - Okosórák, okos/fitnesz karkötők
 - Google Glass

PC-k

- **Asztali PC**

- Nagy méret megengedett → olcsóbban kivitelezhető
- Nem hordozható: helyhez kötött, feladatok széles skálájára

- **Laptop**

- Hordozhatóság → kisebb méret, drágább
- Normál asztali PC kiváltására (munka, játék, szórakozás)

- **Netbook**

- Kis méret mellett cél az olcsóság
- Gyenge teljesítmény!
- Népszerűségük erőteljes visszaesésben van

- **Ultrabook**

- Kis súly, hosszú akkumulátoridő, de nagy elvárt teljesítmény → drága!

Miniatűr PC-k

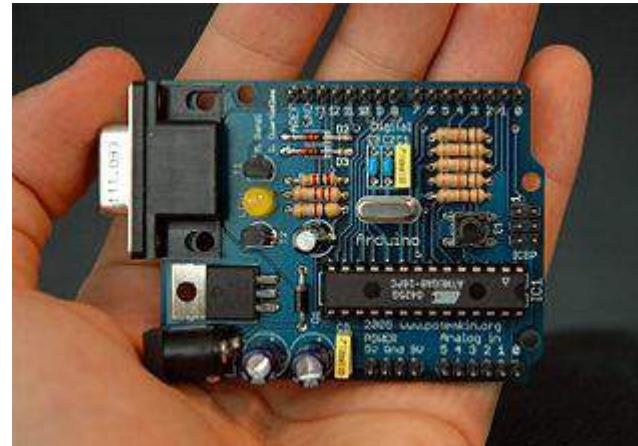
- **Kis méret, PC-szintű tudás**

- Teljes rendszer (CPU, B/K vezérlő, memória, ...)
- Külső perifériák nélkül
 - De ráköthető USB billentyűzet, pendrive; csatlakozik TV-re, monitorra

- **Raspberry Pi**

- 800 MHz processzor, 512 MB RAM
- Olcsó! \$35

- **Arduino**



Okostelefon

Személyi titkári funkciók

- **Menedzserkalkulátorok**
 - EPOC, Apple Newton
 - Monokróm kijelző
 - Digitális titkári feladatok
 - Naptár, kontaktlista, teendők
- **PDA-k**
 - Palm OS, Windows Mobile
 - Asztali PC adatok szinkronja
 - Nyitás a multimédia felé
 - Nincs telefon funkció

Telefónia

- „**Buta”telefonok**
 - Hangátvitel, SMS, telefonkönyv
 - Manapság is népszerű
- **Kezdeti okostelefonok**
 - Nokia Communicator
- **PDA + telefónia**
 - RIM Blackberry, Palm, Symbian
 - Apple iOS, Android, Bada, WebOS, Windows Phone

Okostelefon

- **Viharos és gyors fejlődés az utóbb 10 évben**
 - Változatos hardveres és szoftveres környezet
 - Korábbi „egyeduralkodó” platformok tűnnek el szinte nyom nélkül (pl. Palm, Windows Mobile, Symbian)
 - A felhasználói igényeket időben le kell reagálni! Sőt, ki kell találni azokat!
 - Készülékek csatája helyett ökoszisztemák háborúja! (S. Elop, Nokia)
 - Az üzlet a hozzáadott (fizetős) tartalomszolgáltatásban van
 - Szoftver, e-könyv, film, zene, ...
 - Az egyik ökoszisztemában megvásárolt tartalmak nem minden vihetők át másikra!
- **Sok feladatot átvesznek a PC-ktől!**
 - Tartalomfogyasztás, multimédia

Képek forrása: Wikimedia Commons; a képekre kattintva elérhetők

Okostelefon

- **Hardver fejlődése**

- CPU
 - Rendszerint ARM architektúra
 - 16 MHz → 2.3 GHz 4-8-10 mag
 - Dedikált GPU megjelenése

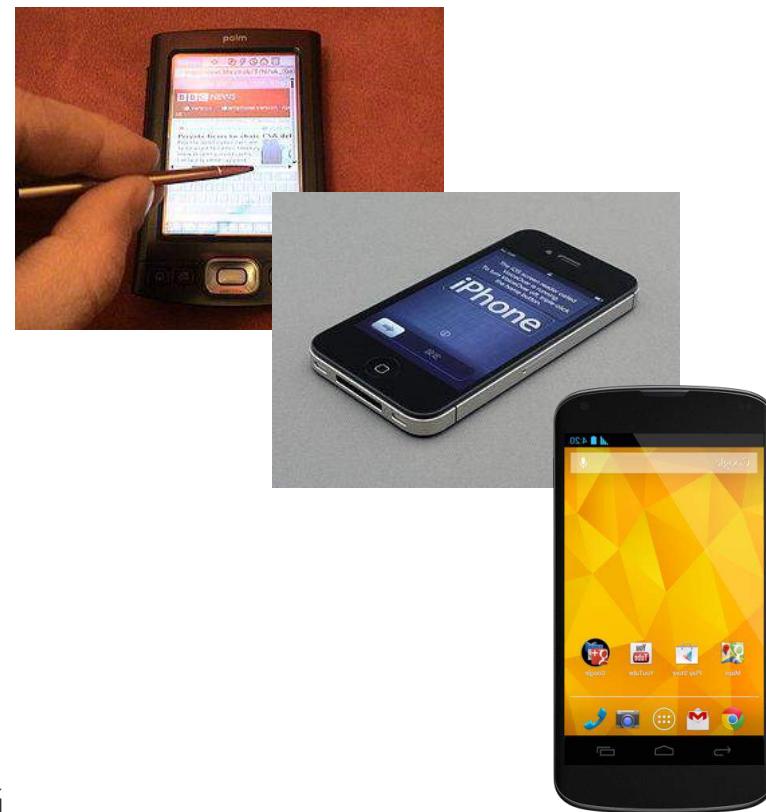
- Memória
 - 1 MB → 1-6 GB

- **Új szenzorok**

- **kamera, GPS, gyorsulásmérő, iránytű, ...**

- **Kijelző**

- Rezisztív: stylus (ceruza-alapú vezérlés) – régi technológia!
- Kapacitív: vezérlés tapintással
- Felbontás: 160x160 fekete-fehér -> 1280x800, 2.55"-6.3"
- Autosztereoszkóp 3D kijelzők megjelenése



Táblák (tabletek)

- **A hardver fejlődése lehetővé teszi**

- nagy méretű (10") kijelzők használatát,
 - akár 8-10 órán keresztül.

- **Kategóriák**

- Ebook olvasók
 - E-ink kijelzők: lassú frissítési idő, de olvasáshoz szemkímélőbb, hosszú üzemiidő (statikus megjelenítéskor nem fogyaszt áramot)
 - Amazon Kindle, Barnes&Noble Nook, ...
- Táblák
 - Apple iPad, iPad2, új iPad dominancia
 - Android Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat rendszerek
 - webOS tábláakra, PC-kre (megszűnik)
 - RIM Playbook (Adobe AIR alapú fejlesztések, QNX-alapú operációs rendszer, újracsomagolt Android alkalmazások futtatása)

Közeljövő: viselhető eszközök

- **Okostelefonok kiegészítőjeként**

- Vezeték nélküli adatátvitel
- Értesítések megjelenítése a telefon elővétele nélkül
- Okosórák, fitnesz karkötők, bioszenzorok, ...



Kép forrása: [The Register](#)

- **Okosórák**

- [Pebble](#), Samsung Gear, ...
- Android Wear, Apple Watch
- Nem teljesen kiforrott technológia!



- **Google Glass**

- Szemüvegre vetített információ
- Okostelefon platform (Android)
- Erősen kísérleti fázis!

Kép forrása: [WikiMedia Commons](#)

SZÁMÍTÓGÉPES ARCHITEKTÚRÁK

Párhuzamos számítógép-architektúra

Nagy Antal

Párhuzamos számítógép-architektúra

- A gépek órajel-frekvenciája folyamatosan nő
 - Az áramkörök sebességét nem lehet a végtelenségig növelni
 - Fénysebesség \Leftrightarrow elektronok és fotonok sebessége
 - Hőelvezetési problémák
 - Méret problémák
 - Folyamatos csökkenés \Leftrightarrow kvantummechanikai hatások
- Párhuzamos számítógépek
 - 1db CPU 0,001 ns ciklusidő \Leftrightarrow 1000db CPU 1ns ciklusidő
 - A teljes számítási teljesítményük **elméletileg** azonos

Párhuzamos számítógép-architektúra

- A párhuzamosítás különböző szinteken vezethető be
 - CPU-n belül
 - Csővezeték vagy szuperskaláris architektúra
 - Több funkcionális egység
 - Utasítás szinten
 - Nagyon hosszú utasításszavak
 - Implicit párhuzamossággal kiegészítve
 - CPU speciális funkcióval valóellátása
 - Több végrehajtási szál kezelése
 - Több CPU elhelyezése egy lapkán
- A módszerek együttes alkalmazásával a szekvenciális tervezésű gépek teljesítményének 10-szerese is elérhető

Párhuzamos számítógép-architektúra

- A következő szinten kiegészítő CPU-k illeszthetők a rendszerbe
 - Speciális feladatokra
 - Hálózati csomagfeldolgozást
 - Multimédia-feldolgozást
 - Titkosítási funkciók
 - Speciális alkalmazások esetén sebesség növekedést lehet elérni
- Nagyobb sebesség növekedéshez sok CPU-t kell felhasználni
 - Alkalmassá kell tenni azokat a hatékony együttműködésre
 - Nagy multiprocesszoros gépek
 - Klaszter számítógépek
- Teljes szervezetek összekötése internet segítségével
 - Lazán kapcsolódó számítási hálózatok

Párhuzamos számítógép-architektúra

Szorosan kapcsolt rendszerek

- Két CPU közel van egymáshoz
- Nagy átvivő képességű
- Kis késleltetésű kapcsolat
- Ugyanazon a feladaton dolgoznak



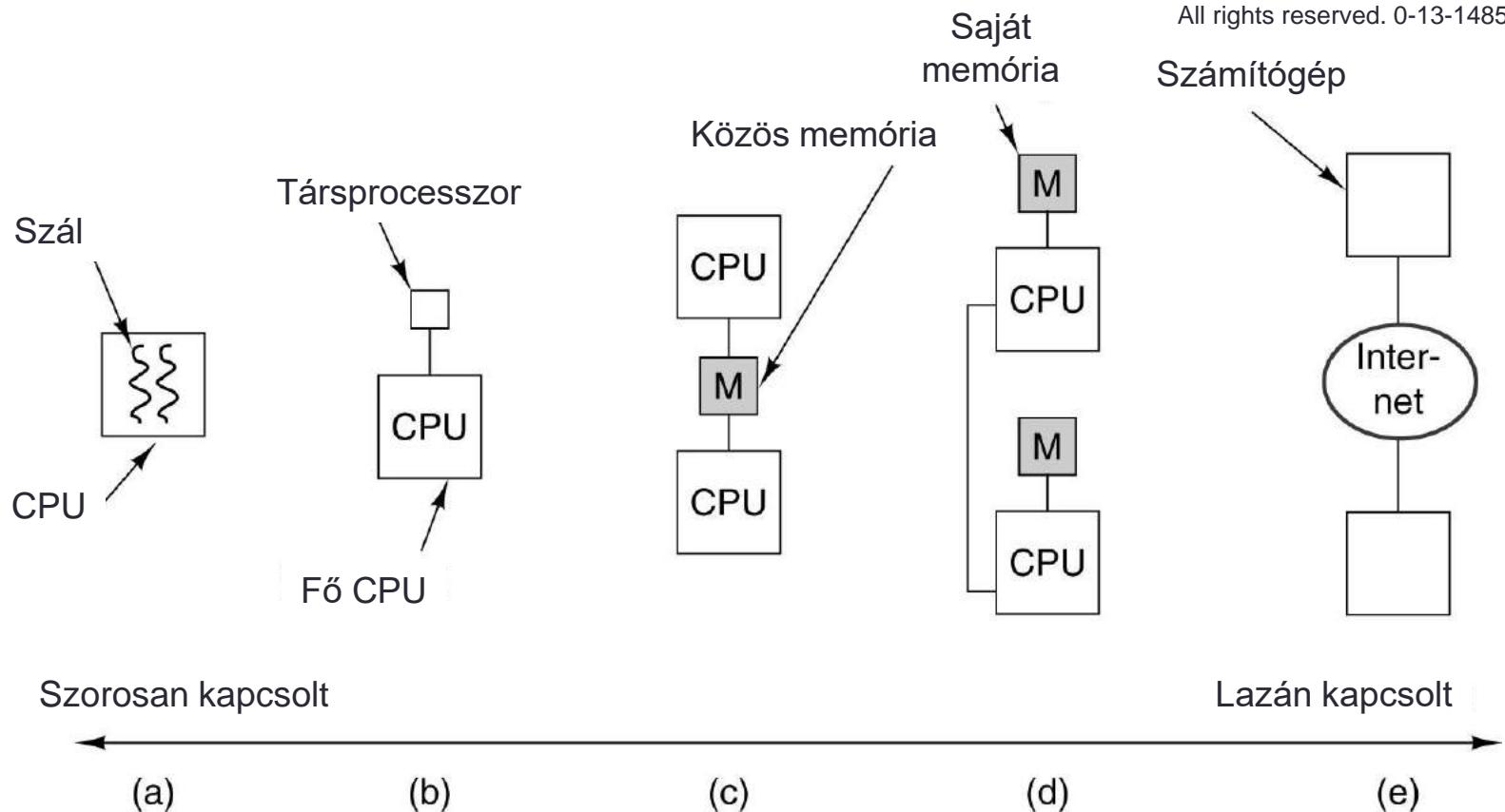
Lazán kapcsolt rendszerek

- Távol lévő
- Kis áteresztőképességű
- Nagy késleltetésű
- A számítási feladatoknak kevés közük van egymáshoz



Párhuzamos számítógép-architektúra

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



- (a) Lapkaszintű párhuzamosság (b) Társprocesszor
 (c) Egy multiprocesszor (d) Egy multicomputer (e) Egy grid.

Lapkaszintű párhuzamosság

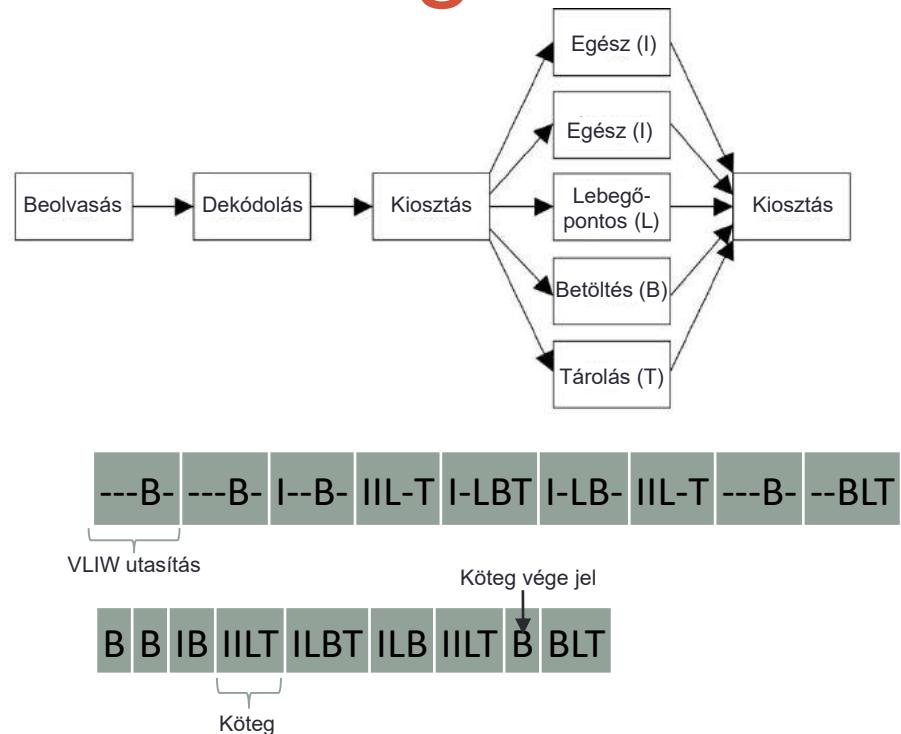
- **Utasításszintű párhuzamosság**
 - Óraciklusonként több utasítást kezdünk el végrehajtani
- Két CPU változat
 - *Szuperskaláris processzorok*
 - A csővezeték egy bizonyos pontjában egy utasítás készen áll a végrehajtásra
 - Képesek több utasítást kiadni a végrehajtó egységeknek minden óraciklusban
 - Az utasítások száma függ
 - A processzor felépítésétől
 - Konkrét körülményektől
 - Előfordulhat, hogy egy utasításnak olyan egységre van szüksége, ami nem áll rendelkezésre vagy olyan adatra, ami még nincs kiszámítva
 - Az utasítás végrehajtása nem kezdődik el

Lapkaszintű párhuzamosság

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

- **Utasításszintű párhuzamosság**

- *Very Long Instruction Word (VLIW)*
 - Nagyon hosszú utasításszavú
 - Korábban több funkcionális egységet igénybe vevő utasítások



- 5 funkcionális egység
 - Két egész, egy lebegőpontos, egy betöltő és egy tároló utasítás véghajtása egyszerre
 - 5 műveleti kód – öt operandus pár
 - Egy utasítás 134 bit hosszú lehet

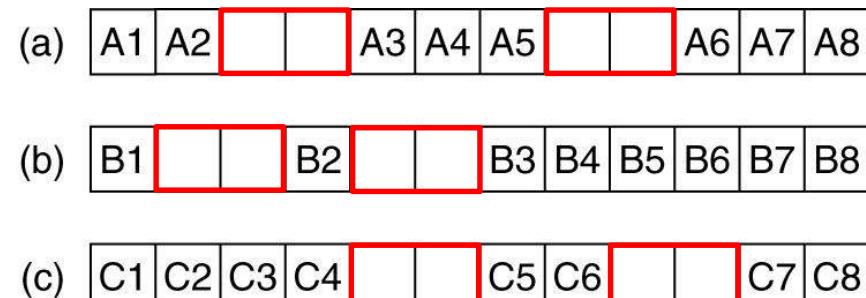
Lapkaszintű párhuzamosság

- **VLIW**
 - Túl merev felépítés
 - Nem minden utasítás tudta kihasználni mind az összes funkcionális egységet
 - NOP (helykitöltő) utasítások
 - Modern VLIW gépek
 - Egymás után következőutasításokat összetartozó kötegként jelölnek meg
 - Köteg vége bit
 - A processzor beolvashatja az egész köteget
 - Egyszerre indíthatja az utasításokat
 - A fordító program feladata a kompatibilis utasítások kötegeinek az előállítása
 - A problémát fordítási időben oldják meg

Lapkaszintű párhuzamosság

- Memóriacímekre való hivatkozás
 - Nincs benne az első, második szintű gyorsítótárban
 - Sok időt igényel a kért szó betöltése \Rightarrow a csővezeték elakad
- Lehetséges kezelési mód a **lapkaszintű többszálúság**
 - Elakadások elfedése
 - Több végrehajtási szál egyidejű kezelésével
 - Egy szál blokkolódásakor egy másik szálat futtathat a CPU
- Megvalósítások
 - Finom szemcsézettségű többszálúság
 - Durva szemcsézettségű többszálúság

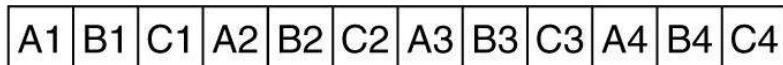
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Lapkaszintű párhuzamosság

- **Finom szemcsézettségű többszálúság**

- A szálakat körben egymás után futtatja
- minden ciklusban másik szál utasítását veszi elő



- Esetünkben a maximális elakadás két ciklushossznyi
 - Három szállal az elakadt művelet minden befejeződik
 - Ha egy memóriára várakozás négy ciklust vesz igénybe
 - A folyamatos működéshez 5 szálra lenne szükség
- A szálaknak semmi közük nincs egymáshoz
 - Mindegyiknek saját regiszterkészlet kell
 - Az egyszerre futtatható szálak maximális száma a tervezéskor eldől

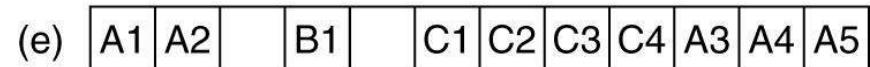
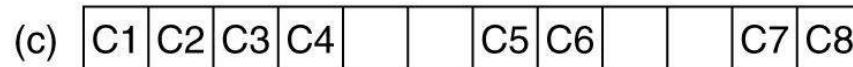
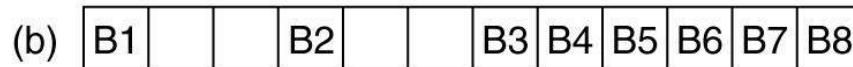
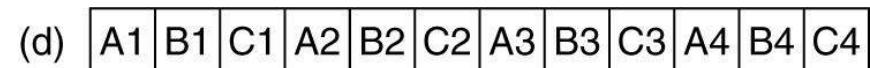
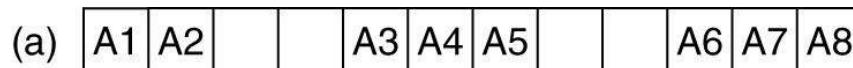
Lapkaszintű párhuzamosság

- Nem csak memóriaműveletek miatt vannak elakadások
 - Szükség lehet egy korábbi utasítás eredményére
 - Amely még nem által rendelkezésre
 - Néha egy utasítás azért nem kezdődhet el
 - Olyan feltételes ágat követ, amelynek a kimenete kétséges
- Ha egy csővezetéknek k fázisa van
 - De van legalább k darab, körben egymás után futtatható szál
 - Soha nem lesz a csővezetékben szálanként egynél több utasítás
 - Tehát konfliktus sem léphet fel
 - A CPU megakadás nélkül, teljes sebességgel működhet

Lapkaszintű párhuzamosság

- Nem biztos, hogy minden rendelkezésre áll annyi szál, ahány fázisa van a csővezetéknek
- Durva szemcsézettségű többszálúság**
 - A szál mindaddig fut, amíg el nem akad
 - Váltás történik elakadáskor a másik szálra
 - Elakadáskor egy ciklus elveszítünk
 - Nem lesz olyan hatékony, mint a fimon szemcsézettségű többszálúság
 - Kevesebb szál kell a CPU kihasználtságához

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Óraciklus →

Óraciklus →

Lapkaszintű párhuzamosság

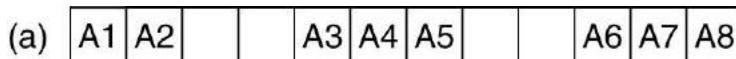
- Durva szemcsézettségű többszálúság elakadás esetén vált a szálak között
 - Másik megközelítés. Azonnal váltunk azoknál az utasításoknál, amelyek elakadást idézhetnek elő
 - Utasítások dekódolása utáni váltásokat tesz lehetővé
 - Elkerülheti a kihasználatlan ciklusokat
- Fusson a szál addig, amíg tudjuk, hogy nem lesz probléma
 - Egyébként váltsunk
 - Hasonlítani fog a finom szemcsézettségű módszerhez

Lapkaszintű párhuzamosság

- Nyílván kell tartani, hogy melyik utasítás melyik szálhoz tartozik
 - Finom szemcsézettségű esetén szál azonosítót rendelünk minden utasításhoz
 - A csővezetékben való áthaladáskor közben mindig egyértelmű, hogy melyik szálhoz tartozik
 - Durva szemcsézettségű esetén hagyjuk kiürülni a csővezetéket
 - Csak ezután kezdjük el az új szálat
 - A csővezetékben minden csak egy szál van
 - Figyelembe kell venni a csővezeték kiürülési idejét és a szál váltások gyakoriságát

Lapkaszintű párhuzamosság

- Tegyük fel, hogy a CPU több utasítást képesek kiadni
 - Például két utasítást
 - Finom szemcsézettségű
 - Durva szemcsézettségű
 - **Egyidejű többszálúság**
 - Egy szál addig indíthat ciklusonként két utasítást, amíg erre képes
 - Ha elakad, akkor azonnal a következő szál utasításai következnek



Óraciklus

A1	B1	C1	A3	B2	C3	A5	B3	C5	A6	B5	C7
A2		C2	A4		C4		B4	C6	A7	B6	C8

Óraciklus →

A1	B1	C1	C3	A3	A5	B2	C5	A6	A8	B3	B5
A2		C2	C4	A4			C6	A7		B4	B6

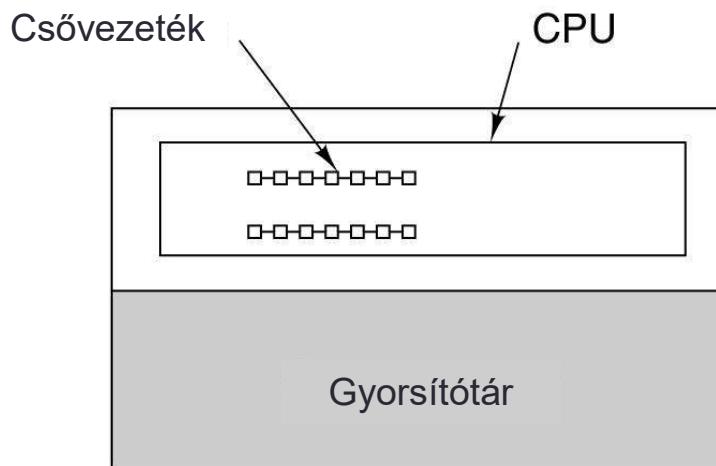
Óraciklus →

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

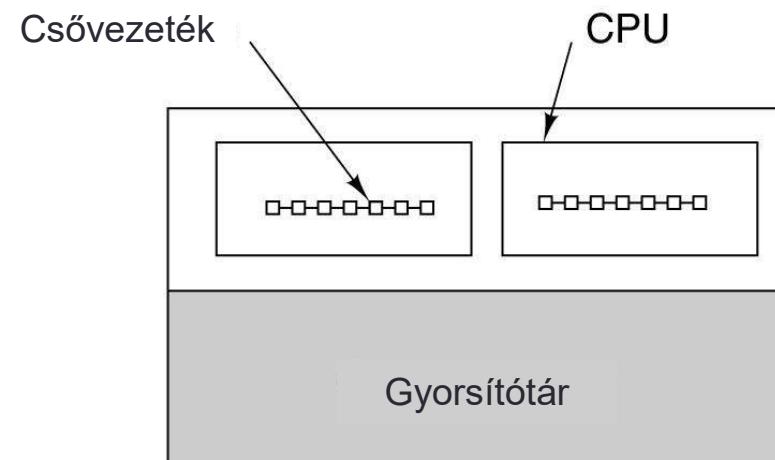
Óraciklus →

Egylapkás multiprocesszorok

- Kettő vagy több CPU-t tartalmazó lapkák
- **Homogén multiprocesszorok egy lapkán**
 - Osztoznak az első és második szintű gyorsítótáron és a központi memórián
 - Osztozhatnak még lemezeken és a hálózati csatlakozón is



Két csővezetékes lapka



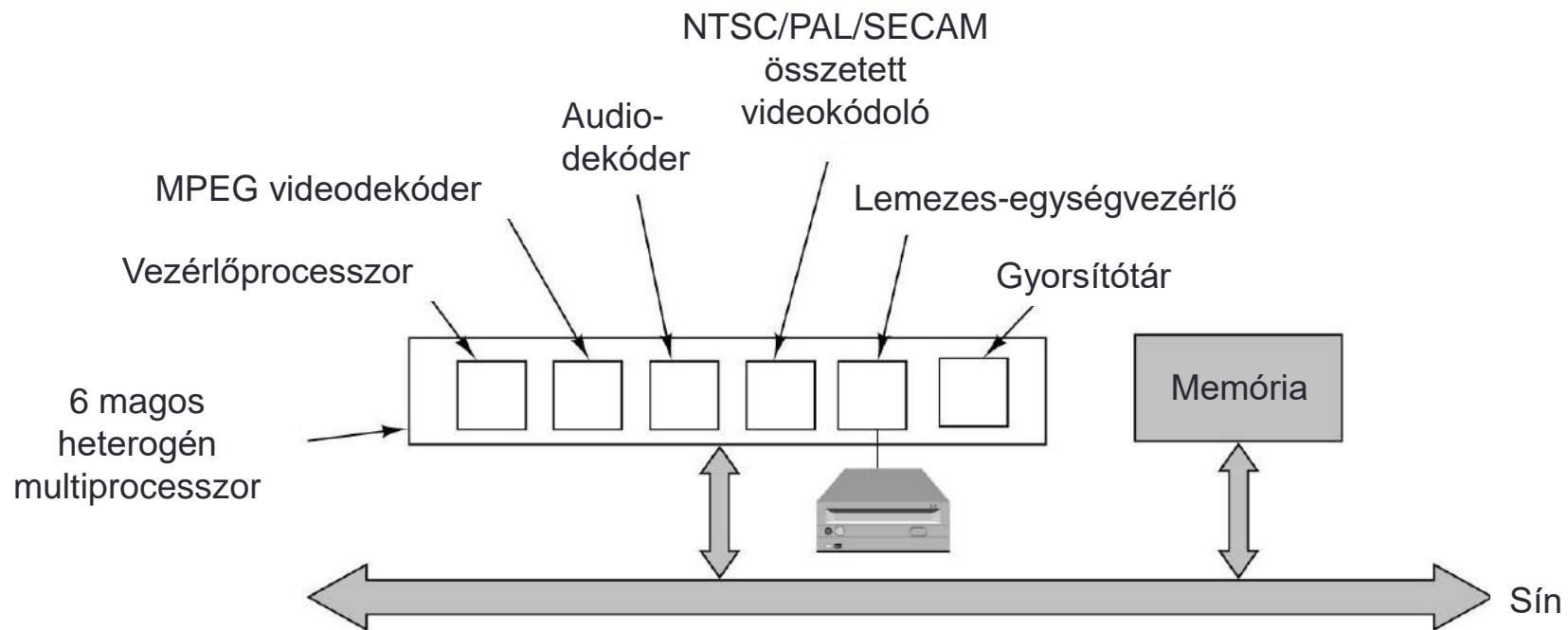
Kétmagos lapka

Egylapkás multiprocesszorok

- **Heterogén multiprocesszorok**

- Beágyazott rendszerek
 - Audiovizuális elektronikai cikkek
 - Magas teljesítménybeli elvárások
 - Szűk specifikációs korlátok
 - Pl. DVD lejátszó
 - Kis megbízhatóságú fejmozgató szervomechanika vezérlése
 - Analóg jelek digitálissá alakítása
 - Hibajavítás
 - Dekódolás és szerzőjogok kezelése
 - MPEG-2 video kibontása
 - Tömörített hang kibontása
 - Kimenet kódolása NTSC, PAL és SECAM rendszerű TV-hez

Egylapkás multiprocesszorok



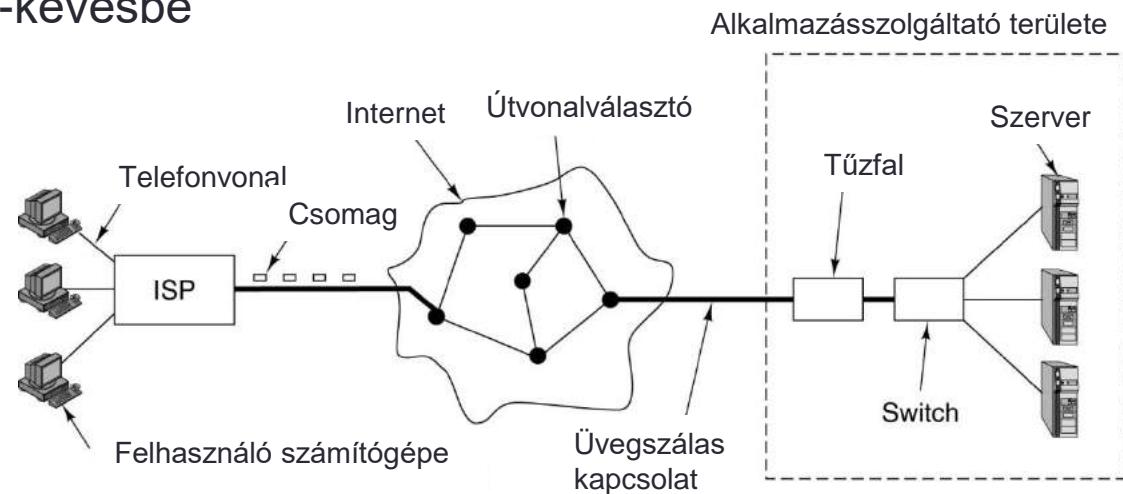
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c)
2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

Társprocesszorok

- Speciális célú processzor
 - Grafika
 - Lebegőpontos aritmetika
 - DMA lapka is tekinthető annak
- A végrehajtandó utasításokat a CPU adja át
 - A társprocesszor függetlenebb is lehet
 - Önállóan dolgozik többé-kevésbé
- Hálózati processzorok
 - Adatforgalom kezelése



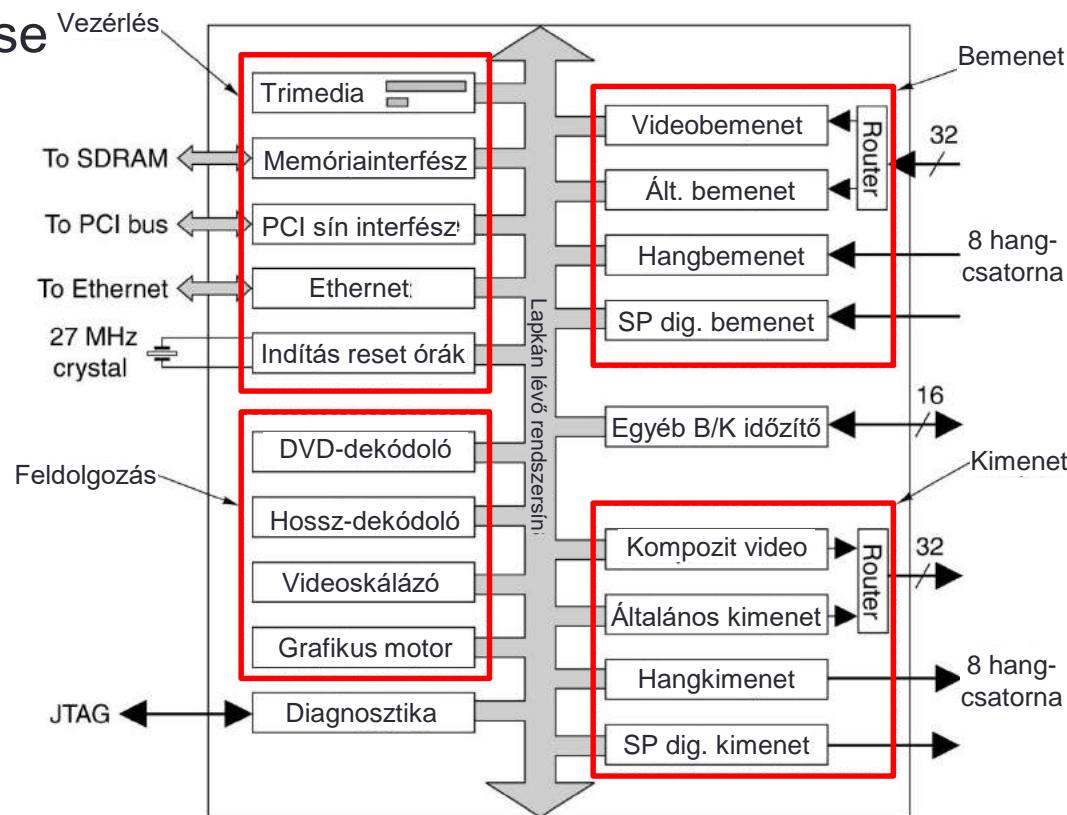
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Társprocesszorok

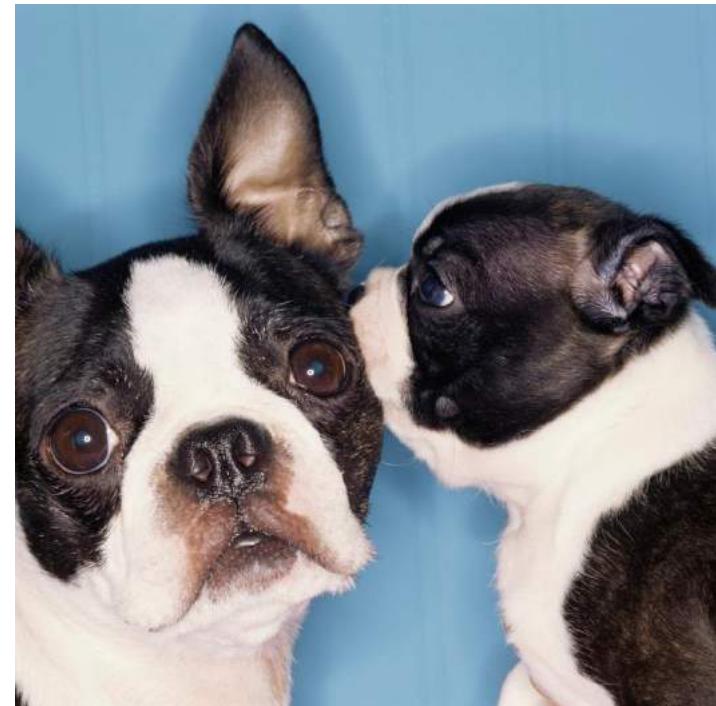
- Médiaprocesszorok
 - Nagy felbontású fényképek, hang és mozgókép kezelése
 - CPU-k nem hatékonyak
 - Nagy mennyiségű adat
 - Nagy műveletigényű
 - NEXPERIA multiprocesszor
 - Egy lapkás
 - Heterogén
 - 456 lába van

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Társprocesszorok

- Kriptoprocesszorok
 - Kliens-szerver közötti kapcsolat hitelesítés után
 - Biztonságos kapcsolat felépítése
 - Lehallgatás elleni védelem
 - A titkosítás szintén számításigényes
 - Szimmetrikus kulcsú titkosítás
 - Bitek alapos összekeverése
 - Nyilvános kulcsú titkosítás
 - 1024 bites számok szorzásán és hatványozásán alapul
 - Speciális műveletek gyors elvégzése
 - Pl. újlenyomat ellenőrzése

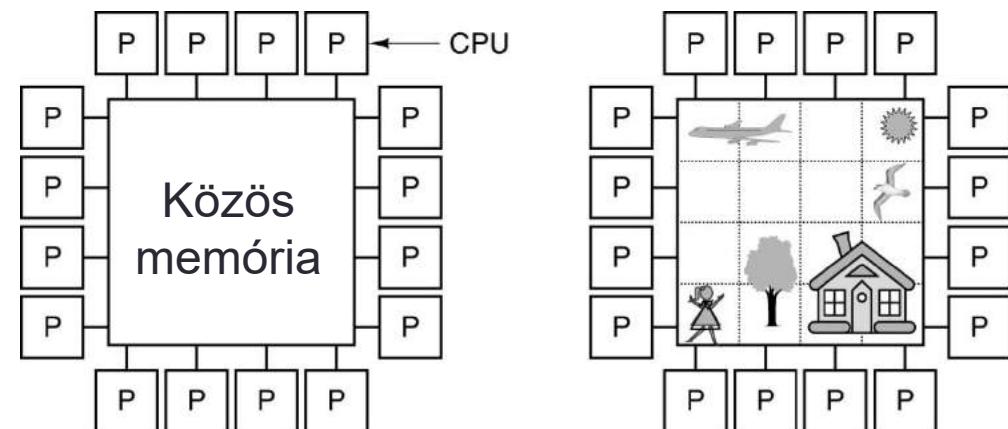


Közös memóriás multiprocesszorok

- Több komplett CPU egy rendszerbe való összeépítése
 - Multiprocesszorok, multiszámítógépek
 - Egy feladat különböző részein végeznek műveleteket
 - Kommunikálniuk kell egymással
- **Multiprocesszorok**

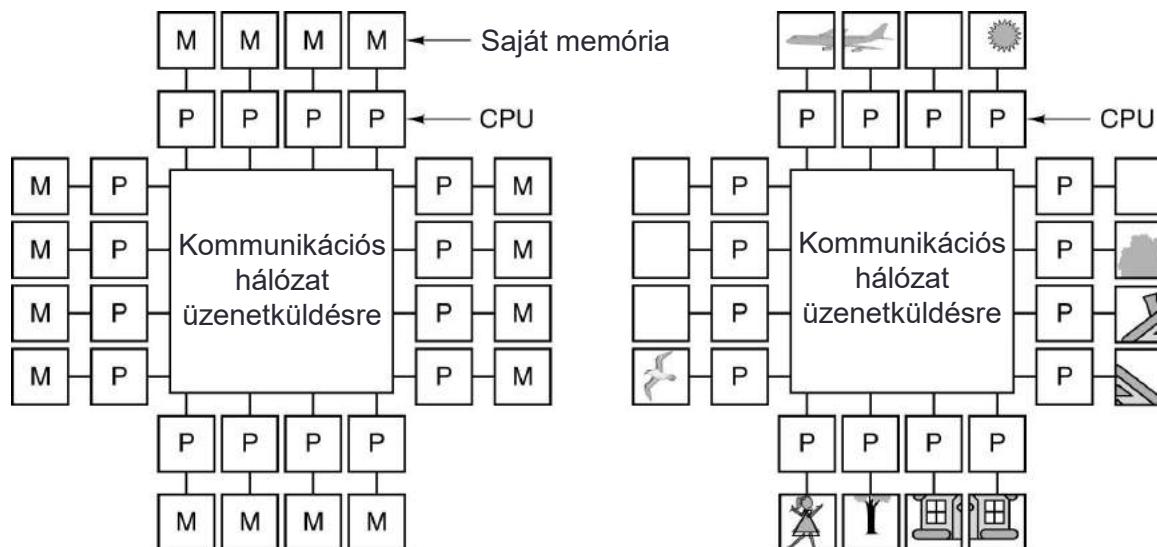
- Az összes CPU egy közös memórián osztozik
 - Egy közös logikai címtartományt lát
 - A közös fizikai memóriára van leképezve
- STORE – LOAD utasítások
- Két folyamat kommunikációja
 - Egyik memoriába ír
 - Másik kiolvassa
- Egy operációs rendszer
 - Egy laptábla
 - Egy folyamattábla

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



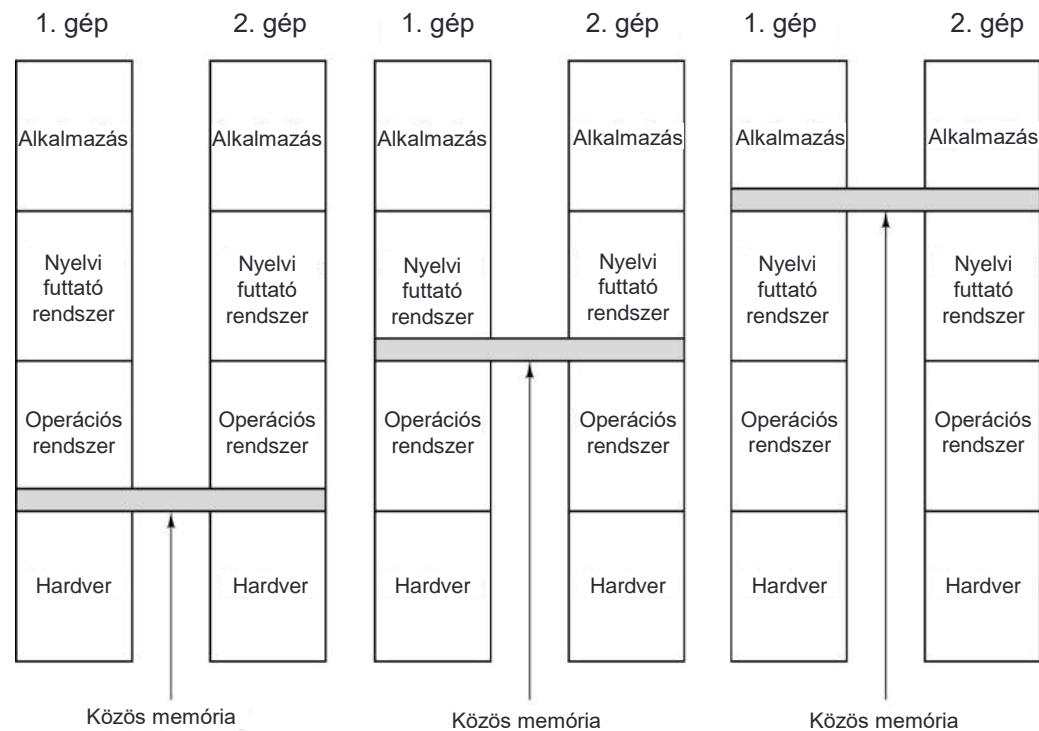
Üzenetátadásos multiszámítógépek

- **Multiszámítógépek**
- minden CPU-nak saját memóriája van
 - Csak az adott CPU érheti el
 - Osztott memóriájú rendszereknek is nevezik
 - minden CPU-hoz külön fizikai címtartomány van rendelve
- Folyamatok kommunikációja
 - Hálózaton keresztpáros üzenetek küldésével és fogadásával



Multiprocesszorok - Multiszámítógépek

- Multiprocesszorok
 - Nehéz megépíteni
 - Könnyű programozni
- Multiszámítógépek
 - Könnyű megépíteni
 - Nehéz programozni
- Hibrid rendszerek létrehozása
 - A számítógépek rétegekből épülnek fel
 - Közös memória kezelésének megvalósítása valamely rétegen



Multiprocesszorok - Multiszámítógépek

- **Hardveres közös memória**

- Egyetlen egy operációs rendszer
- Egyetlen táblázathalmaz
- Memóriafoglaltsági tábla
- Több féle módon meg lehet valósítani

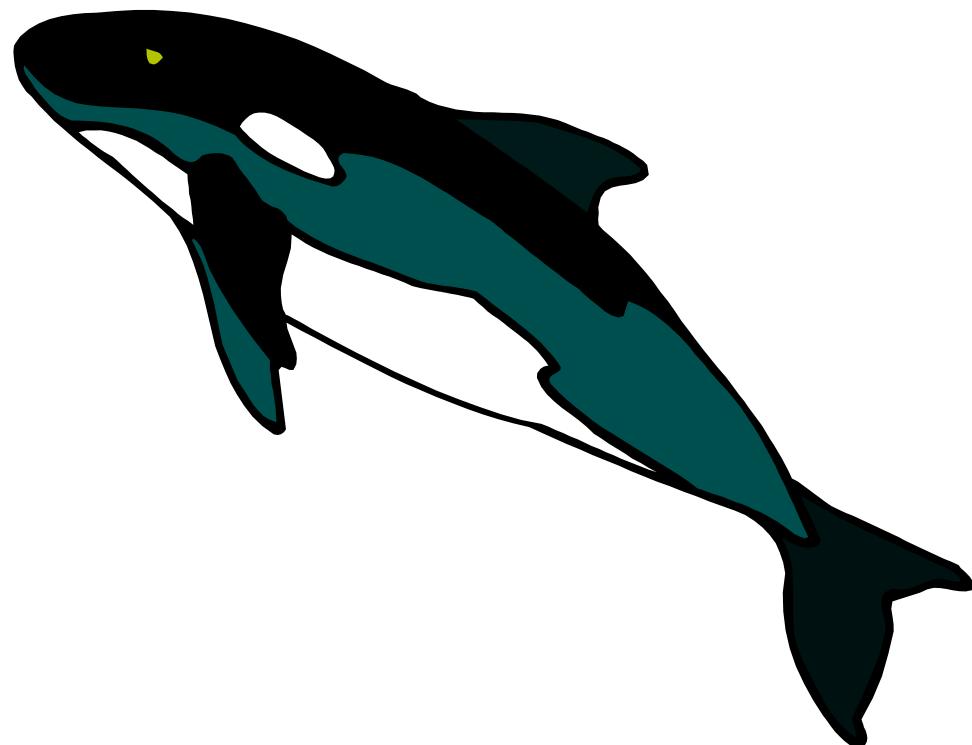
- **Multiszámítógép hardveres megoldás**

- Operációs rendszer szimulálja a közös memóriát
 - Egy lapozott közös virtuális címtartomány
 - Distributed Shared Memory (DSM) – elosztott közös memória
 - minden egyes gépnek saját virtuális memóriája és laptáblája van
 - Ha az egyik CPU-n olyan memóriahivatkozást hajt végre, ami egy másik CPU lapjára vonatkozik
 - Az operációs rendszer aktivizálódik
 - Leválasztja a virtuális memóriából a lapot
 - Küldje el az összekötő hálózaton
 - A lapot magához kapcsolja
 - Újra indítja a laphiányt kiváltó utasítást

Multiprocesszorok - Multiszámítógépek

- **Felhasználói szintű
(nyelvfüggő)
megvalósítás**

- A programozási nyelv biztosít közös memóriás absztrakciókat
 - A fordító és a futtató rendszer valósít meg
- Linda modell: közös adategységtér használata
- Orca modell: közös adatobjektumok megosztása



Párhuzamos számítógépek osztályozása

- Flynn osztályozása
 - Utasításáram és adatáram

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0

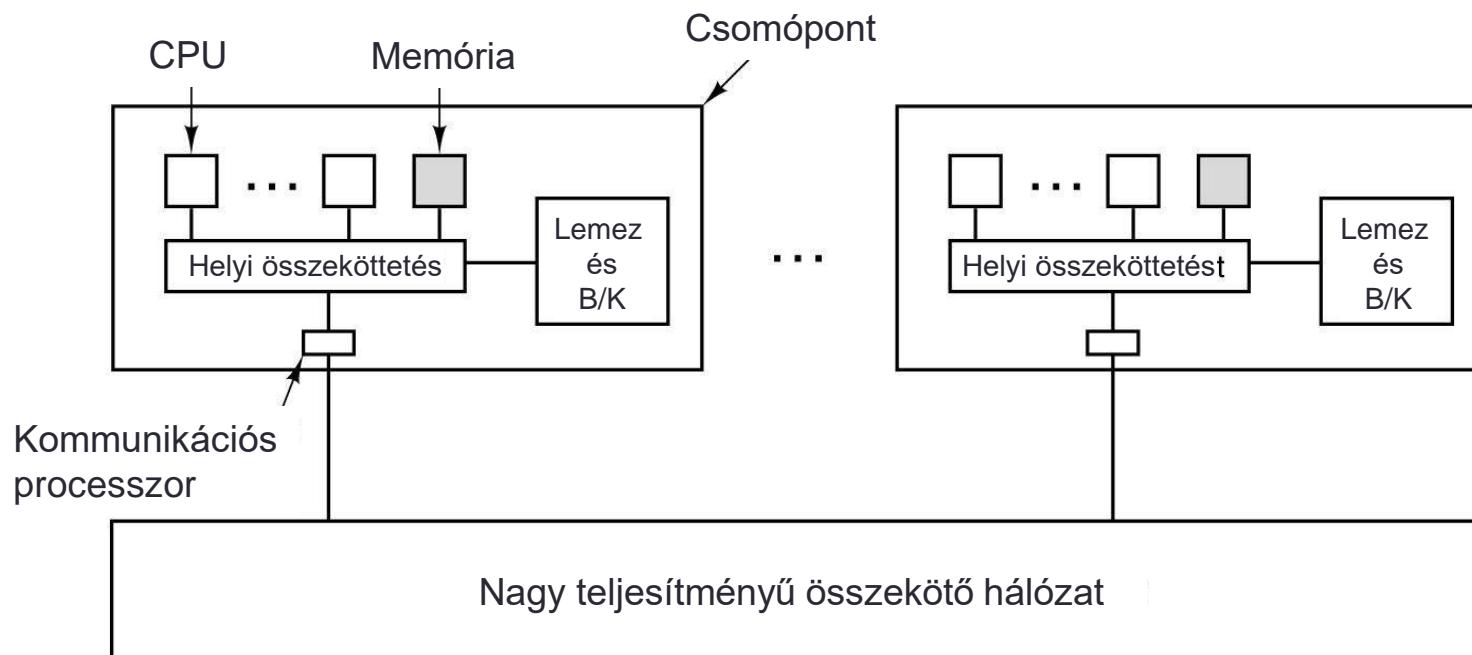
Utasításáram	Adatáram	Elnevezés	Példák
1	1	SISD	Klasszikus Neumann-féle gép
1	Többszörös	SIMD	Vektor szuperszámítógép, tömbprocesszor
Többszörös	1	MISD	Létezése vitatható
Többszörös	Többszörös	MIMD	Multiprocesszor, multiszámítógép

Egyesek a csővezetékes gépeket MISD gépeknek tekintik

Üzenetátadásos multiszámítógépek

- Kommunikációs processzor
 - Nagy sebességű összekötő hálózat kapcsolja össze
 - Sokféle topológia
 - Kapcsolási séma
 - Útvonalválasztó algoritmus

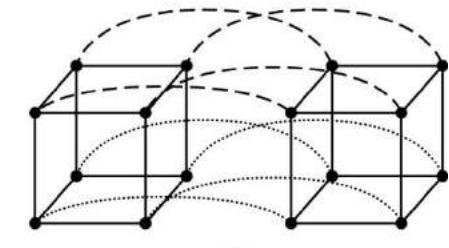
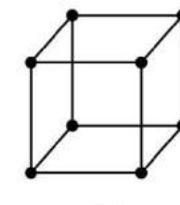
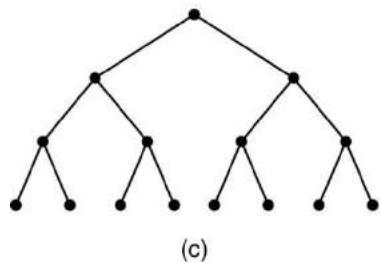
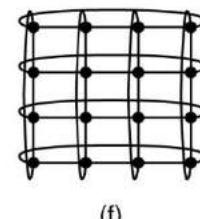
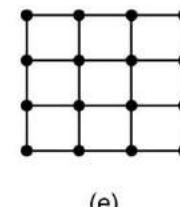
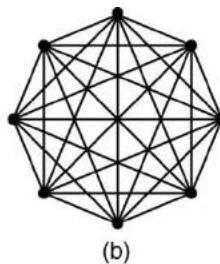
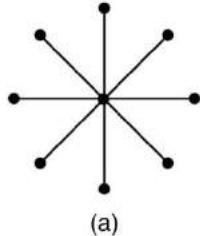
Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Üzenetátadásos multiszámítógépek

- Topológiák

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



Csillag, teljes összekötés,
fa, kör

Rács, kettős tórusz,
kocka, 4D-s hiperkocka

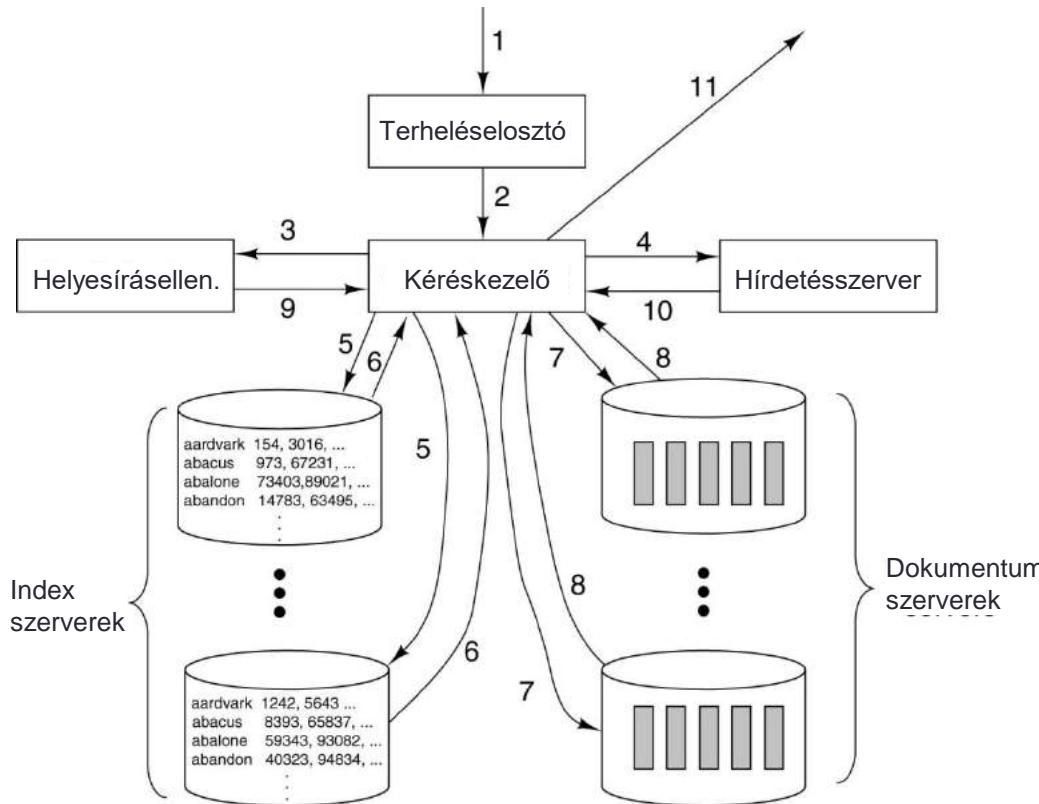
Klaszterszámítógépek

- Közönséges, kereskedelmi forgalomban elérhető PC-k vagy munkaállomások
 - Szabványos hálózati kártyákkal csatlakoznak egymáshoz
- Cluster of Workstations (COW)
 - Központosított
 - Egyetlen egy helyiségben
 - Gépek homogének
 - A hálózati kártyán és a lemezeken kívül nincs másik perifériájuk
 - Széttagolt
 - Egy épületben
 - Gyakran LAN-on (Local Area Network) keresztül vannak összekapcsolva
 - Heterogének
 - Tétlen munkaállomások kihasználása

Google

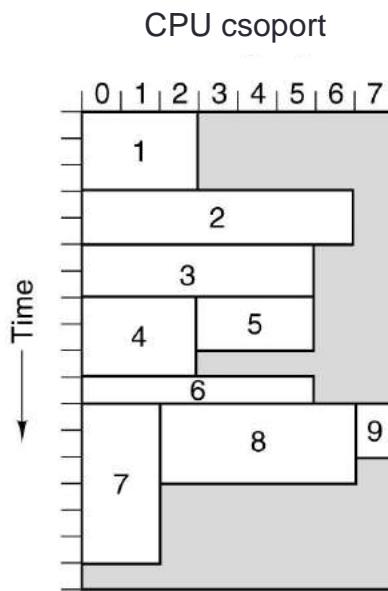
- Egy Google kérés feldolgozása

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.
All rights reserved. 0-13-148521-0



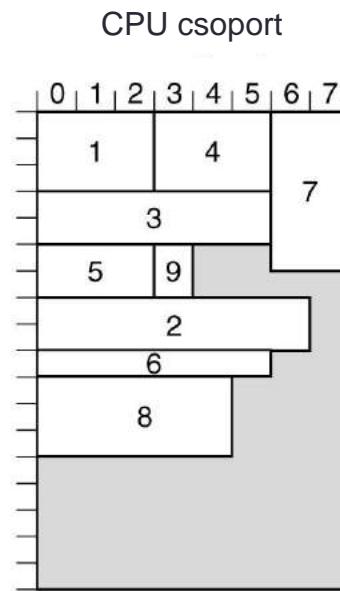
Ütemezés

- Több felhasználó, egymástól függetlenül különböző hosszúságú időre kér CPU-kat
 - Klaszter munkaütemezőre van szükség

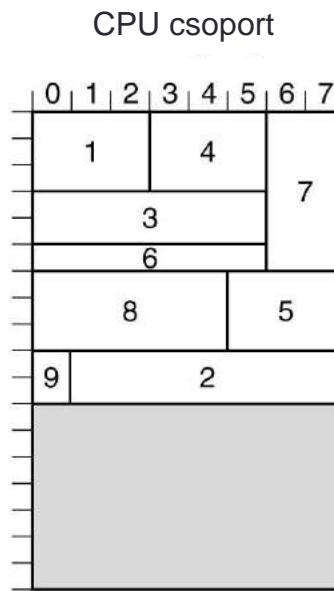


FIFO

Sor eleji blokkolás nélkül



Sor eleji blokkolás nélkül



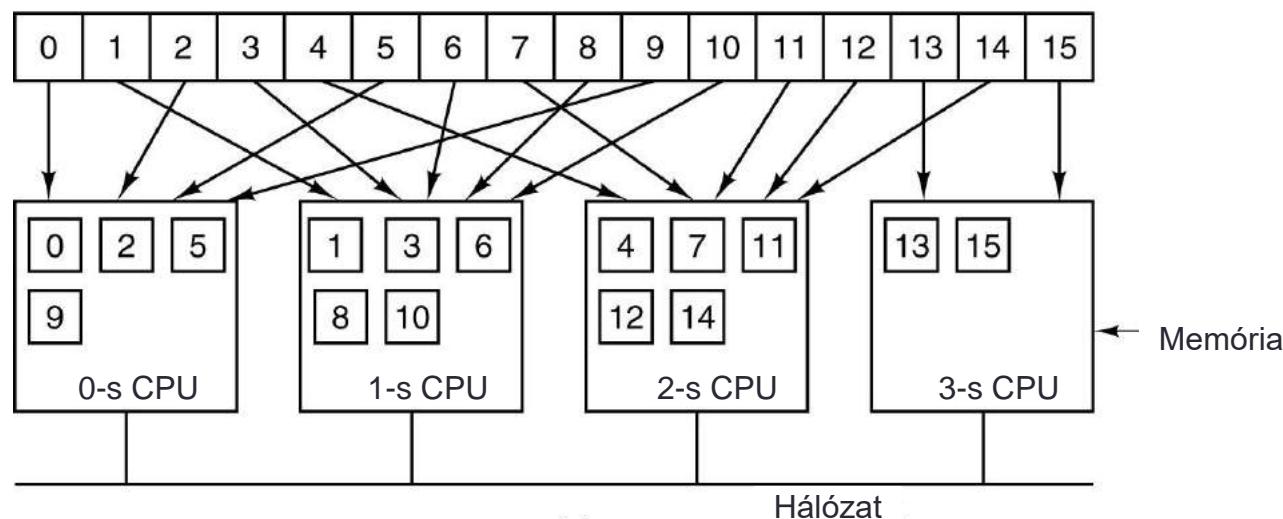
Lefedő ütemezés

Szükséges a felhasználó CPU-k számának az ismerete

Alkalmazásszintű közös memória

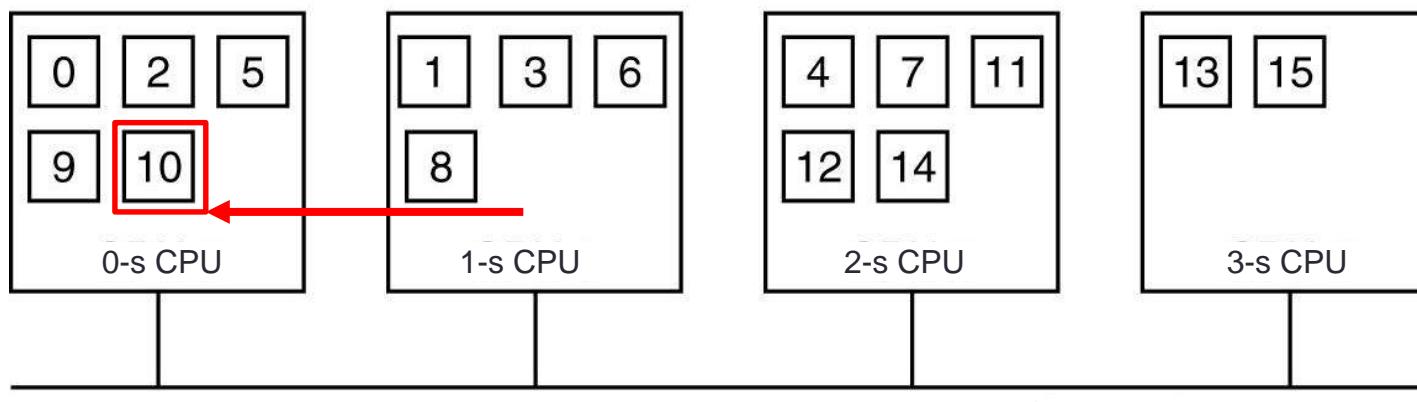
- Osztott közös memória
 - Alkalmazás szintű
 - Multiszámítógép CPU-inak egy halmaza osztozik
 - Közös, lapokból álló virtuális címterületen
 - Legegyszerűbb esetben minden lap pontosan egy CPU RAM-jában található meg

16 labból álló, globálisan megosztott virtuális memória



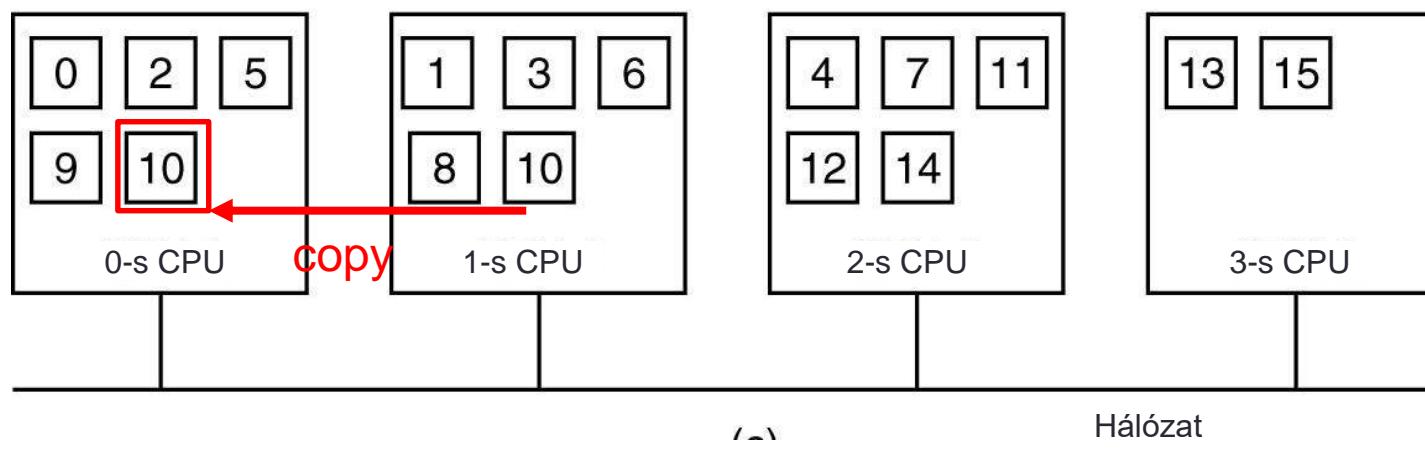
Alkalmazásszintű közös memória

- Ha a CPU saját RAM-jában lévő lapokra hivatkozik
 - Az olvasás/írás késedelem nélkül megtörténik
- Távoli memóriában lévő lapra való hivatkozáskor
 - Laphány lép fel
 - Üzenet küldés útján a birtokló csomópont felszabadítja és elküldi a hiányzó lapot
 - A lap megérkezése után a CPU RAM-jába kerül



Alkalmazásszintű közös memória

- Teljesen megosztott, sorosan konzisztens memória
 - Csak olvasható lapok egyszerre több csomópontnál is jelen vannak
 - Laphiány esetén a másolat kerül elküldésre
 - Az eredeti lap a helyén marad
 - Nincs konfliktus veszély



Alkalmazásszintű közös memória

- Téves megosztás
 - Tfh. egy folyamat egy lap tetejére, egy másik CPU-n futó másik folyamat pedig ugyanennek a lapnak az aljára ír folyamatosan
 - A lapnak csak egy másolata létezik
 - A lap állandóan oda-vissza „pattog”
 - Elhárítás Treadmarks rendszernél
 - Új másolat nem készül addig, amíg egy **release** művelet végre nem hajtódik
 - Ezután a lap újra megosztottá válhat
 - Kezdetben minden lapot csak olvashatóra állít be
 - Védelmi hiba lép fel első íráskor
 - „Iker” másolat készül
 - A lap állapota írható/olvasható
 - Egymáskövető írások teljes sebességgel hajtódnak végre
 - Távoli laphiány esetén az ikerszavak összeghasonlítása után csak a megváltozott lapok kerülnek elküldésre

Teljesítmény

- A cél az, hogy az egy processzoros gépnél gyorsabb gépet állítsunk össze

- Megfelelő eredmény
- Költséghatékony

Hardvermértékek

- CPU-k sebessége
- B/K egységek sebessége
- Összekötő hálózat teljesítménye
 - Az első kettő ugyanaz az egy processzoros gépnél is
 - Fontos paraméter
 - Késleltetési idő
 - Sávszélesség

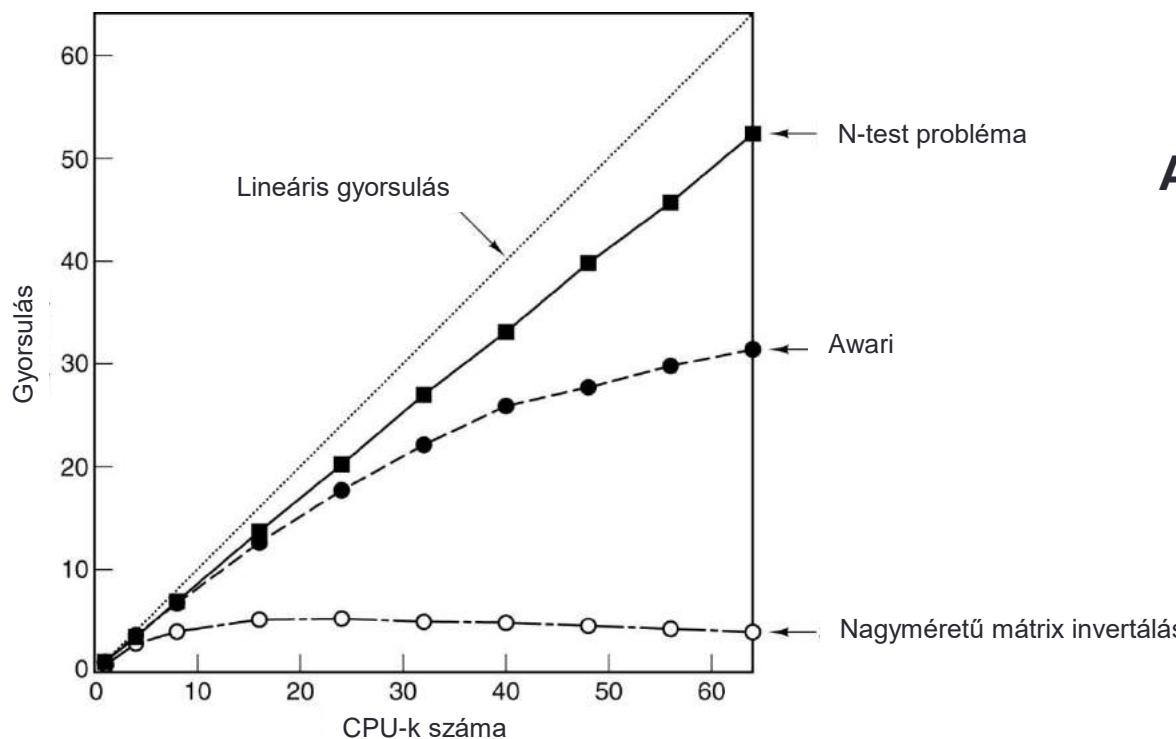
Menettéri késleltetés egy CPU esetében

- Az az időtartam, amely egy csomag elküldése és a válasz megérkezése között eltelik
- Memória esetén
 - Egy szó vagy blokk kiolvasásához illetve beírásához szükséges idő
- Másik CPU esetén
 - A processzorok kommunikációs ideje adott méretű csomag esetén

Teljesítmény

- **Szoftvermértékek**

- Mennyivel lesz a program gyorsabb?
 - Hányszor lesz gyorsabb egy program egy n processzoros rendszerben az egyprocesszoros rendszerhez képest?

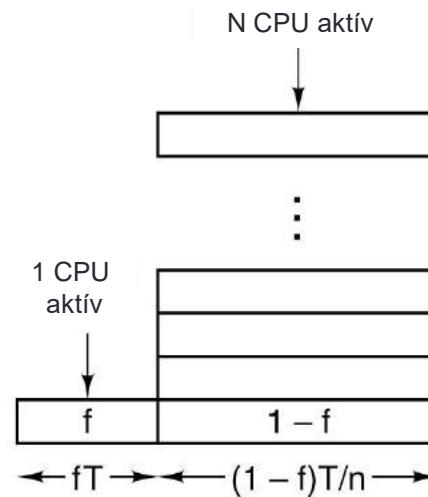
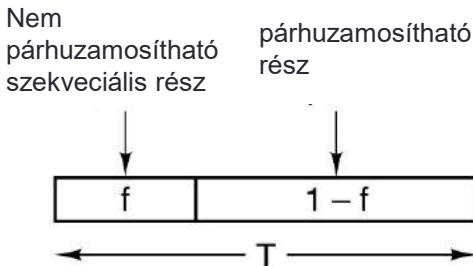


**A gyorsulás FÜGG
a problémától és
a megvalósítástól**

Teljesítmény

- Egy program T ideig fut
- Vannak szekvenciális részek f és vannak párhuzamosítható részek ($1 - f$) a programokban

Forrás: Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Gyorsulás =

$$=^T / \left(fT + \frac{(1-f)T}{n} \right) =$$

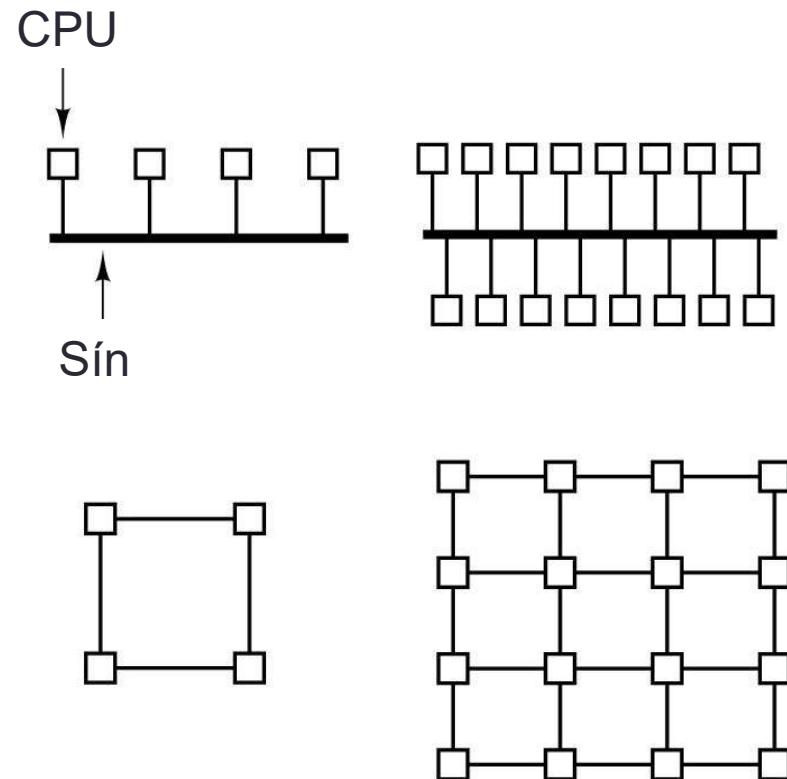
$$= \frac{n}{1 + (n-1)f}$$

Amdahl törvénye

Teljesítmény

- Lehetetlen az optimális gyorsulás elérése
 - $f = 0$ lineáris gyorsulás
 - Kommunikációs késleltetési idő
 - Véges sávszélesség
 - Algoritmusok hatékonysága
- Nagy teljesítmény elérése
 - CPU-k számának növelése
 - Ne keletkezzen szűk keresztmetszet
 - Skálázható rendszerek
 - Arányos számítási teljesítmény növekedés több CPU esetén

- Sín rendszerrel összekötött CPU-k
 - Sín sávszélesség



Teljesítmény

- CPU-k számának a növekedésével
 - Nő az átmérő és az átlagos késleltetési idő
- Ideális esetben egy skálázható rendszernél CPU-k hozzáadása mellett nem szabadna változni
 - CPU-nkénti átlagos sávszélességnek
 - Konstans értékű átlagos késleltetési időnek
- A gyakorlatban a késleltetési idő a mérettel együtt nő
 - A logaritmikus növekedés a legjobb
 - Hipercockánál érhető el
- A skálázásból származó késleltetési idő a finom és közepes szemcsézettségű alkalmazások hatékonyságát rontja

Teljesítmény – késleltetési idő csökkentése

- Technikák
 - Adat többszörözés
 - Az adat elérés felgyorsul
 - Pl. gyorsítótár
 - Másolat a felhasználási helyekhez közel
 - Előolvasás
 - Mielőtt szükségünk lenne az adatra előre betöljük
 - Automatikus és program által vezérelt
 - A fordító program által közvetlenül is lehet vezérelni
 - Információ a hardverről, időzítésről
 - Vezérelni tudja az adatok elhelyezését
 - Többszálú végrehajtás
 - Folyamatok közötti gyors váltás
 - Gyors kapcsolás egy futtatható folyamatra
 - Nem blokkoló írás
 - STORE után folytatódik a végrehajtás
 - LOAD esetén nehezebb a folytatás



Grid számítások

- Nagy távolságban lévő számítógépek összekapcsolása grid rendszerekbe
 - Nagyon nagy lazán kapcsolt heterogén klaszter
 - Virtuális szervezetet alkotnak
 - Rugalmas
 - Együttműködés szükséges
 - Saját erőforrásai feletti ellenőrzés megtartása adott mértékig
 - Szolgáltatások, eszközök és protokollok kifejlesztése
 - Sokféle erőforráshoz kell hozzáférést biztosítani
 - A tulajdonos dönti el, hogy az erőforrásokból mennyit, kinek és mikor bocsát rendelkezésre
 - Erőforrásokhoz való hozzáférés és azok kezelése

Grid számítások

- Grid szerkezet
 - Legalsó szint
 - Grid építő komponensek
 - Erőforrásréteg
 - Konkrét erőforrások kezelése
 - Szabályozott hozzáférés
 - Biztonságos felület nyújtása magasabb szintek felé

Rétegek	Funkció
Alkalmazási	Alkalmazások, amelyek a kezelt erőforrásokat a szabályozott módon megosztják
Kollektív szolgáltató	Erőforráscsoportok felfedezése, figyelése, vezérlése; brokerszolgáltatás
Erőforrás	Szabályozott és biztonságos hozzáférés konkrét erőforrásokhoz
Szerkezeti	Fizikai erőforrások; számítógépek, tárolók, hálózatok, érzékelők, programok és adatok

Grid számítások

- Grid szerkezet

- Kollektív szállító réteg

- Erőforráscsoportok kezelése
- Szabad számítás kapacitás, lemezterület megkeresése
- Erőforrás elosztás
- Adatok többszörözése

- Alkalmazási réteg

- Felhasználói alkalmazások
- Jogok szerzése erőforrásokhoz való hozzáféréshez

Rétegek	Funkció
Alkalmazási	Alkalmazások, amelyek a kezelt erőforrásokat a szabályozott módon megosztják
Kollektív szolgáltató	Erőforráscsoportok felfedezése, figyelése, vezérlése; brokerszolgáltatás
Erőforrás	Szabályozott és biztonságos hozzáférés konkrét erőforrásokhoz
Szerkezeti	Fizikai erőforrások; számítógépek, tárolók, hálózatok, érzékelők, programok és adatok

Grid számítások

- A biztonság kulcsfontosságú
 - Erőforrás-tulajdonosi igények
 - Jogosultságok ésszerű kezelése
 - Jelszavak a hozzáféréshez
- Egyszeri feliratkozás
 - Azonosítás utáni jogosítvány megszerzése
 - Kinek a nevében történik a munkavégzés
 - Továbbadható
 - A számítási feladat részekre bomlik
 - Le kell képezni a távoli gépek biztonsági mechanizmusaira
- Szabványok szükségesek
 - Global Grid Forum
 - Open Grid Services Architecture (OGSA) keretrendszer
 - Fejlesztés alatt álló szabványok összefogása

