A projekt az Európai Unió társfinanszírozásával, az Európa terv keretében valósul meg.

SZÁMÍTÓGÉP-ARCHITEKTÚRÁK

HEFOP 3.3.1–P.-2004-06-0071/1.0

Ez a kiadvány a
„Gyakorlatorientált képzési rendszerek kialakítása és minőségi fejlesztése az agrár-felsőoktatásban”
című program keretében készült

SZÁMÍTÓGÉP-ARCHITEKTÚRÁK

Szerkesztő:

Dr. Harnos Zsolt
Budapesti Corvinus Egyetem

Dr. Herdon Miklós
Debreceni Egyetem

Szerző:

dr. Kovács György
Debreceni Egyetem

Lektor:

Busznyák János
Pannon Egyetem

Lágymányosi Attila
Szent István Egyetem

ISBN 978-963-9732-53-7

E tankönyv teljes mértékben megegyezik a Debreceni Egyetem honlapján, a http://odin.agr.unideb.hu/hefop/ elérési úton megtalálható, azonos című tankönyvvel.

Első kiadás

Kiadó:
Debreceni Egyetem Agrár- és Műszaki Tudományok Centruma
Agrárgazdasági és Vidékfejlesztési Kar

Debrecen, 2007.

Tartalomjegyzék

1. BEVEZETÉS 6

1.1. A SZÁMÍTÓGÉPEK ÁLTALÁNOS MŰKÖDÉSI ELVE 6
1.2. SZÁMÍTÓGÉPEK FEJLŐDÉSTÖRTÉNETE 7
1.2.1. Nulladik generáció - Mechanikus számítógépek 8
1.2.2. Az első generáció – Elektroncsövek 10
1.2.3. A második generáció – Tranzisztorok 13
1.2.4. A harmadik generáció - Integrált áramkörök 14
1.2.5. A negyedik generáció - Nagyon nagy mértékű integráltság 16
1.3. A SZÁMÍTÁSI MODELL 18
1.3.1. Adatalapú számítási modellek 18
1.3.2. Nem adatalapú számítási modellek 21
1.3.3. A problémaleírás modellje 21
1.3.4. A végrehajtás modellje 22
1.4. ARCHITEKTÚRA FOGALMAK 23
1.4.1. A számítógép-architektúra szintjei 23
1.4.2. Számítógép architektúrák értelmezése absztrakciós szinteken 24

2. SZÁMÍTÓGÉP-RENDSZEREK 28

2.1. PROCESSZOROK 28
2.1.1. A CPU felépítése 28
2.2. A PROCESSZOR UTASÍTÁSKÉSZLETE 31
2.2.1. Az utasítás-szerkezet 32
2.2.2. Utasítás-típusok 33
2.3. UTASÍTÁS VÉGREHAJTÁS 39
2.3.1. A processzor működése 39
2.3.2. A processzor számítási teljesítményét meghatározó tényezők 41
2.3.3. A CISC utasításkészletű gépek 41
2.3.4. A RISC utasítás készletű gépek 42
2.4. A PÁRHUZAMOS ARCHITEKTÚRÁK OSZTÁLYOZÁSA 45
2.4.1. Osztályozás a feldolgozott utasítás- és adatfolyamok száma szerint 45
2.4.2. SISD architektúrájú számítógépek 46
2.4.3. SIMD architektúrájú számítógépek 46
2.4.4. MISD architektúrájú számítógépek 46
2.4.5. MIMD architektúrájú számítógépek 47
2.5. A SZÁMÍTÓGÉPRENDSZER ARCHITEKTÚRÁK TELJESÍTMÉNYÉNEK NÖVELÉSE 47
2.6. PÁRHUZAMOS ARCHITEKTÚRÁK 48
2.6.1. Utasításszinten párhuzamos architektúrák 48
2.6.2. A pipeline működése 50
2.6.3. Szuperskalár architektúrák 51
2.6.4. A pipelining működés során fellépő problémák 51
2.6.5. A pipelining során fellépő problémák kezelésének módszerei 52
2.6.6. Párhuzamos dekódolás 53
2.6.7. VLIW processzorok 54
2.6.8. EPIC processzor. 55
2.7. ADATPÁRHUZAMOS ARCHITEKTÚRÁK 57
2.7.1. Vektorprocesszorok 57
2.7.2. Tömbprocesszoros számítógépek: 58
2.7.3. Üzenetátadásos számítógépek (multiprocesszoros architektúrák) 59
2.8. A PÁRHUZAMOS ARCHITEKTÚRÁK KORSZERŰ OSZTÁLYOZÁSA (SIMA 1998) 59

3. ADATTÁROLÁS SZÁMÍTÓGÉPBEN 62

3.1. KÖZPONTI VAGY OPERATÍV MEMÓRIA 62
3.1.1. A memória címzése 62
3.1.2. Hibajelzés és hibajavítás 64
3.1.3. A memória szervezése és típusai 65
3.2. MEMÓRIÁK HIERARCHIÁJA 67
3.2.1. Regisztertárak 67
3.2.2. Gyorsító (rejtett, vagy cache) tárak és megoldásaik 68

3.2.3. A cache tárolók ... 70
3.2.4. Az L1 és L2 cache ... 71
3.2.5. A cache működésének elve, cache hit és miss ... 71
3.2.6. A cache tárak felépítése és típusai ... 72
3.3. ASSZOCIATÍV TÁROLÓK ... 73
3.3.1. Helyettesítési stratégia, blokkbemásolás a cache-be ... 74
3.3.2. A cache-ben megváltoztatott adatok visszaírása a főtárba ... 75
3.3.3. Lemezgyorsító tárak ... 77
3.4. TÁRSZERVEZÉS ... 78
3.4.1. Virtuális tárkezelés ... 78
3.4.2. Szegmentálás ... 81
3.4.3. Lapozás ... 82

4. MIKROPROCESSZOR ALAPÚ SZÁMÍTÓGÉPRENDSZER ... 87

4.1. A SZÁMÍTÓGÉP LEGFONTOSABB RÉSZEGYSÉGEI, ÉS MŰKÖDÉSÜK ... 87
4.1.1. A megszakítási rendszer ... 87
4.1.2. Megszakítások és kivételek kiszolgálása ... 92
4.1.3. A közvetlen memória-hozzáférés ... 93
4.1.4. Input-output eszközvezérlők ... 93
4.2. KOMMUNIKÁCIÓS KAPCSOLATOK A SZÁMÍTÓGÉP RÉSZEGYSÉGEI KÖZÖTT ... 94
4.2.1. A számítógépek sínrendszerének logikai részei ... 96
4.2.2. A sínrendszerek típusai ... 98
4.2.3. Buszprotokoll és a sínvezérlés formái ... 100
4.3. I/O MŰVELETEK VÉGREHAJTÁSA MIKROSZÁMÍTÓGÉPEKBEN ... 102
4.3.1. Az I/O műveletekkel kapcsolatos alapfogalmak ... 102
4.3.2. Az operációs rendszer szerepe az I/O műveletekben ... 102
4.3.3. Az I/O eszközök címzése ... 103
4.3.4. Az I/O átvitel típusai ... 104

5. AZ IBM PC ... 106

5.1. A PC FEJLŐDÉSTÖRTÉNETE, RÉSZEGYSÉGEI ... 107
5.1.1. Processzororok ... 107
5.1.2. Alaplap ... 122
5.1.3. A BIOS és szerepe ... 124
5.1.4. A chip-set ... 128
5.1.5. Sínrendszerek a PC-kben ... 133
5.2. HÁTTÉRTÁRAK, TÖMEGTÁRAK A PC-KBEN ... 140
5.2.1. A merevlemez jellemzői ... 142
5.2.2. Az SCSI ... 147
5.2.3. RAID ... 149
5.2.4. Optikai tárolók: CD ROM ... 153
5.2.5. Az írható és újraírható CD ... 155
5.2.6. Optikai tárolók: a DVD ... 156
5.2.7. Elektronikus elvű hordozható tárak ... 157

1. Bevezetés

Az informatika korunk egyik meghatározó tudománya, az elmúlt közel ötven évben sokszor a szakemberek számára is alig követhető fejlődésen ment át. Az egyre gyorsuló fejlődés, úgy a technológia, mint az alkalmazott megoldások terén generáció váltások sorát eredményezte.

Évezredes kívánság, hogy legyenek számológépek, amelyek tévedés nélkül, gyorsan számoljanak, kalkuláljanak helyettünk. A számolás fárasztó tevékenység, ugyanakkor nélkülözhetetlen számunkra. Az emberiség éppen ezért igénybe is vett minden olyan eszközt, ami segítette ezt a szellemi munkát. A számolást segítő eszközök története egyidős az emberiség történetével.

Az őskori embernek még nem volt szüksége a számolásra. Az emberi munkamegosztás és ezzel együtt a kereskedelem kialakulásával létrejött a mérés, a mértékegység fogalma. A számolás, a mérés átszőtte az emberek mindennapi életét. Az írás kialakulásával egyidőben (i.e. 4000–3000 táján) létrejöttek a számok leírására alkalmas jelek, illetve számjegyek.

Eleinte az emberek az ujjaikat használták a számoláshoz, aminek a latin neve *digitus*. Innen származik az angol számjegy, a *digit* elnevezés. Később a számoláshoz köveket, fonalakat használtak. Ezt követően a nagyobb értékek számossági megjelenítésére kialakult az átváltásos rendszerű számábrázolás, Egyiptomban a tízes, tizenkettes, majd Mezopotámiában a hatvanas számrendszer.

A számolás gyorsításának igénye egyidős a számolással. A babiloniak táblázatokkal dolgoztak, az első egyszerű, és mégis jól használható segédeszköz kínai eredetű. A mintegy 3000 éves *abakusz* lehetővé tette az egyszerűbb műveletvégzést. Ez egy digitális eszköz, és primitív volta ellenére egyes helyeken még a közelmúltban is használták. Japánban például *szorobánnak* nevezik. Az abakusz sínekbe helyezett apró kövekből áll, a kövecske latin neve *calculus,* innen származik a ma annyira elterjedt kalkulátor szó. Az abakuszt némileg módosítva a XVI. századig mint fő számolást segítő eszközt használták. Utódja a golyós kalkulátor ma már inkább csak játék.

Számítógép tágabb értelemben minden olyan berendezés, amely megfelelő bemenő adatok alapján olyan kimenő adatokat képes előállítani, amelyek vagy közvetlenül értelmezhetőek a felhasználók részére vagy más berendezések vezérlésére használhatóak. Szűkebb értelemben a számítógép olyan elektronikus berendezés, amely információk (adatok és programok) tárolására alkalmas memóriával rendelkezik, az adatok feldolgozásához programra van szüksége és saját tevékenységét, működését vezérli

Két fő típusa az analóg és a digitális számítógép. Az analóg számítógépek fizikai jelenségek matematikai leírásával szimulálják a folyamatokat, be- és kimenetük is valamilyen fizikai jellemző (pl. villamos feszültség, hőmérséklet, nyomás). A digitális számítógépek számjegyekre (digit) bontják, fordítják a feladatot, és ezeken hajtják végre az előírt műveleteket. A számítások alapegysége a bit. Létezik egy átmeneti típus is, az un. analogikai számítógépek, amelyek egyesítik a két típus előnyeit. Ezeket elsősorban biológiai, áramlástani feladatok modellezésére, megoldására használják.

1.1. A számítógépek általános működési elve

A számítógépeket a szellemi munka automatizálására tervezték, és ezek közül is elsősorban a számítási munkák megkönnyítésére. Az aritmetikai műveletek visszavezethetők néhány logikai művelettel megvalósított műveletsorra. Nemcsak a számítási műveletek, hanem más adatfeldolgozási problémák műveletei is megvalósíthatók elemi logikai műveletek sorozatával. A számítógépek alapvetően egyszerű szerkezetek, abban az értelemben, hogy

csak és kizárólag azt hajtják végre, amire az ember által készített programok előírást adnak számukra.

A megvalósítás azonban bonyolult automatákat jelent, amelyek vezérlő algoritmusa cserélhető. A számítógéptől, mint automatától megköveteljük, hogy programozható módon, aritmetikai és logikai műveletek végrehajtására legyen képes. Rendelkezzék olyan lehetőségekkel, amelyek révén egy végrehajtandó feladathoz kívülről át tudja venni a kiinduló adatokat, és az eredményt is át tudja adni a környezetének. Tehát ki- és bemenettel rendelkező automata berendezés legyen.

Ahhoz, hogy ez az automata hatékonyan működhessen, a különböző részfeladatok elvégzésére a fejlesztések során önálló egységeket hoztak létre, a munkamegosztás elve alapján.

Ez az elkülönítés különösen jól követhető a mai mikroszámítógépek esetében, mivel az egyes egységek vagy egy-egy integrált áramköri egységben (IC-ben), vagy egy-egy cserélhető, nyomtatott áramköri lapon, kártyán találhatók.

A számítógépek fő funkcionális egységei a következők:

- központi egység(CPU)
    - vezérlő egység(CU),
    - aritmetikai és logikai műveletvégző egység(ALU),
    - központi tár, főtár,
- másodlagos- vagy háttértárolók,
- perifériák(I/O)
    - beviteli egységek(input units),
    - kiviteli egységek(output units),
    - ember-gép-kapcsolat eszközei;

valamint az egyes egységeket összekötő, a gép különböző részei közötti egységesített és gyors adatátvitelt biztosító

- buszrendszer (sínrendszer):
    - címbusz,
    - adatbusz,
    - vezérlőbusz.

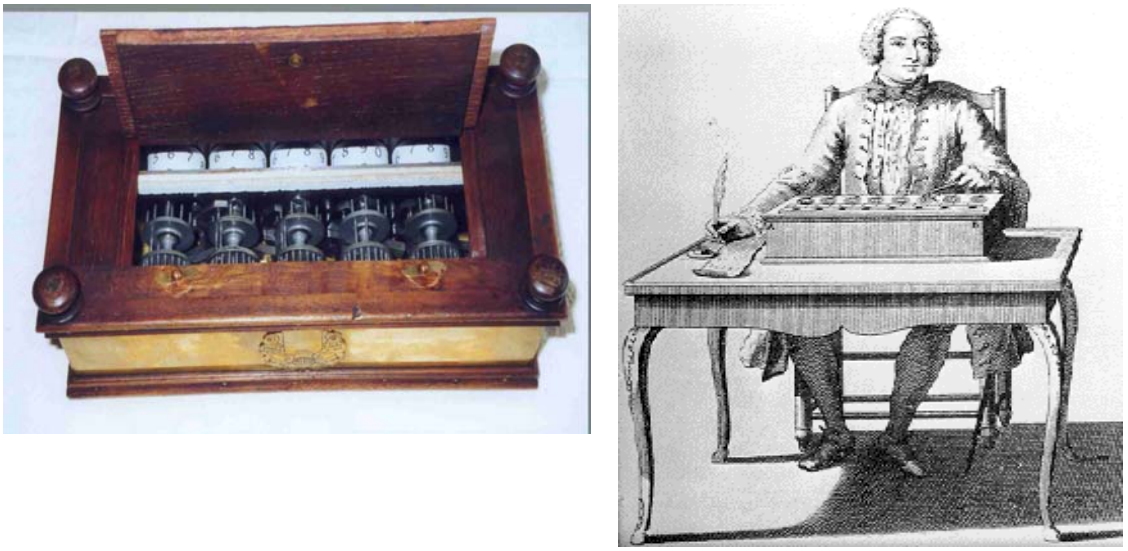
A központi egység(CPU=Central Processing Unit) ismertetett felbontása a (vezérlő egység, aritmetikai egység, memória) egy tágabban értelmezett felfogása a számítógép ezen egységének, és a korábbi gépek (első generáció) esetében volt fontos ez felosztás. A korszerű számítógépek esetében, a memória növekvő fontossága és az egységek jól elválasztható volta miatt, központi egység (CPU) alatt inkább csak a vezérlő- és az aritmetikai egység kettősét értjük. Ezt nevezzük processzornak, vagy a mikroszámítógépek esetében alkalmazott egy-tokos processzorokat mikroprocesszoroknak. A továbbiakban CPU, vagy (mikro) processzor alatt a szűkebben értelmezett központi egységet értjük.

1.2. Számítógépek fejlődéstörténete

A számítógépeket többféle szempontból fogjuk osztályozni. Az egyik ilyen osztályozási szempont a generációkba sorolás.

1.2.1. Nulladik generáció - Mechanikus számítógépek

Az első személy, aki működő számítógépet konstruált, a francia tudós, Blaise Pascal (1623-1662) volt, akiről – tiszteletére – programnyelvet is neveztek el (Pascal). Édesapja a kormány alkalmazottjaként adóbeszedő volt, és ezt a készüléket Pascal 1642-ben, 19 éves korában készítette apja munkájának segítésére. A teljesen mechanikus szerkezetben fogaskerekek „számoltak", és kézi tekerésű hajtókarral lehetett működtetni. Pascal gépe a *Pascaline* csak összeadásra és kivonásra volt alkalmas, és 6 digiten számolt. Hét darab készült belőle, és a mai is fellelhető példányok még mindig működőképesek.



Forrás:http://perso.wanadoo.fr/yves.serra/pages/pasc02.htm

1-1. ábra: A Pascaline kinyitva, és használat közben

Harminc évvel később egy német matematikus, Baron Gottfried Wilhelm von Leibniz (1646-1716) által konstruált gép már szorozni és osztani is tudott. Ezzel a találmányával Leibniz három évszázaddal ezelőtt megalkotta a mai értelemben vett négy-műveletes zsebszámológép megfelelőjét.

150 évvel később a University of Cambridge matematika professzora, Charles Babbage (1792-1871), a sebességmérő feltalálója, hajózási navigációs táblázatok kiszámítására tervez egy gépet, a **difference engine**-t, amely 20 jegyű számokkal végez műveleteket. Nem építi meg, mert a kor technikája nem teszi lehetővé, például a súrlódást nem tudja kiküszöbölni. (100 év múlva, a fennmaradt tervek alapján készítik el a Babbage által megálmodott gépet.) Bár a differenciálgép megépítésének kísérlete megbukott, Babbage egy még bonyolultabb masina, az analitikai gép megtervezésébe fogott és haláláig több tervváltozatot készített.

Babbage megfogalmazta, hogy egy számológépnek milyen követelményeknek kell megfelelnie:

- ne kelljen mindig beállítani a számokat
- meg lehessen adni egyszerre az összes operandust és műveletet (ez lyukkártya segítségével oldható meg).
- legyen input egység (ez a lyukkártya olvasó)
- legyen utasítás (a művelet a lyukkártyán)
- legyen külső programvezérlés (a lyukkártyákon tárolt utasítássorozat, a program)
- legyen olyan egység, amely a kiindulási és a keletkezett számokat tárolja ("memória")

- legyen aritmetikai egység, amely számológépen belül végzi a műveleteket
- legyen output egység (a gép nyomtassa ki az eredményt).

A fő eltérés a differenciálgéptől az volt, hogy az analitikai gépet lyukkártyákkal programozhatóra tervezte. Ez forradalmi ötlet volt a számítógép történetében, bár maga az eszköz nem teljesen új, hiszen a Jacquard-szövőgép már korábban is alkalmazott lyukkártyát. A terveiben számos olyan módszert vezetett be, amelyeket a modern számítógépek alkalmaznak.

Az analitikai gépnek négy alkotóeleme volt: a tároló (memória), a központi egység (számítást végző egység), a bemeneti egység (lyukkártya-olvasó) és a kimeneti egység (adatok kivitelére lyukasztott és nyomtatott formában).

Az analitikai gép jelentősége abban állt, hogy általános rendeltetésű volt. Lyukkártyáról utasításokat tudott beolvasni, adatokat a tárolóból előhívni, és rajtuk a műveletet elvégezni.

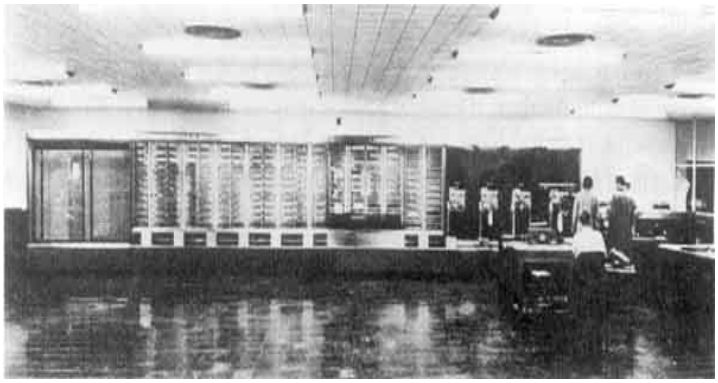
Ha a bemeneti kártyára más programot lyukasztottak, akkor az analitikai gép képes volt az eddigiektől különböző számításokat elvégezni, azaz szoftverre volt szüksége. A világ első programzója Babbage munkatársa, a híres brit költő, Lord Byron lánya, Ada Augusta Lovelace volt.

Németországban Konrad Zuse, 1934-ben építészhallgatóként kezdte fejleszteni mechanikus számítógépét, a Z-1-et. Ez a gép leginkább Babbage számoló-műveivel mutatott rokonságot, de mégis lényegesen különbözött azoktól, ugyanis Zuse nem számkerekes decimális gépet, hanem igen/nem kapcsolókkal működő bináris gépet épített. A géphez mechanikus memóriát konstruált, és ezt a memóriát használta később a jelfogós számítógépében is. Gépét 1936/37-ben jelfogós működésűre tervezte át, mivel a mechanikus logika egyrészt nehézkesen működött, másrészt a gép nem volt eléggé megbízható. Az első jelfogós modell a Z-2 volt, majd - még a háború alatt - elkészült a Z-3 és a Z-4, valamivel később a Z-5, majd a Z-11, valamennyi modell jelfogós berendezés volt.

Az Egyesült Államokban John Atanastoff az Iowa Állami Akadémián és George Stibbitz a Bell Laboratóriumban tervezett számítógépet. Atanastoff gépe korához képest elképesztően fejlett volt. Kettes számrendszert használt, a memóriákat kondenzátorok alkották, amiket folyamatosan frissítettek, hogy kondenzátor önkisülésével az adatok ne vesszenek el. Ezt a folyamatot ő a "memória frissítésének" nevezte. A modern dinamikus memóriachipek (DRAM) is hasonló módon működnek. Ez a gép korának nem megfelelő hardver-technikája miatt soha nem lett működőképes.

Stibbitz számítógépe, bár sokkal egyszerűbb volt, mint Atanastoffé, azonban működött, és 1940-ben a Darmouth Akadémián egy konferencián tartott róla előadást. Howard Aiken, Babbage munkáját megismerve elhatározta, hogy relékből építi fel azt az általános rendeltetésű számítógépet, amit Babbage nem tudott megépíteni fogaskerekekből.

Aiken első gépét, a Mark I-t a Harvardon készítette el 1944-ben, amely egy műveletet 6 másodperc alatt végzett el. Bemeneti és kimeneti egységként lyukszalagot használt.



Forrás: http://www-groups.dcs.st-and.ac.uk

1-2. ábra: ***A Mark I.***

Ezt követően Aiken elkészítette a gép utódját, a Mark II-t, de időközben a jelfogós gépek elavultak. Elkezdődött az elektronikus számítógépek kora.

1.2.2. Az első generáció – Elektroncsövek

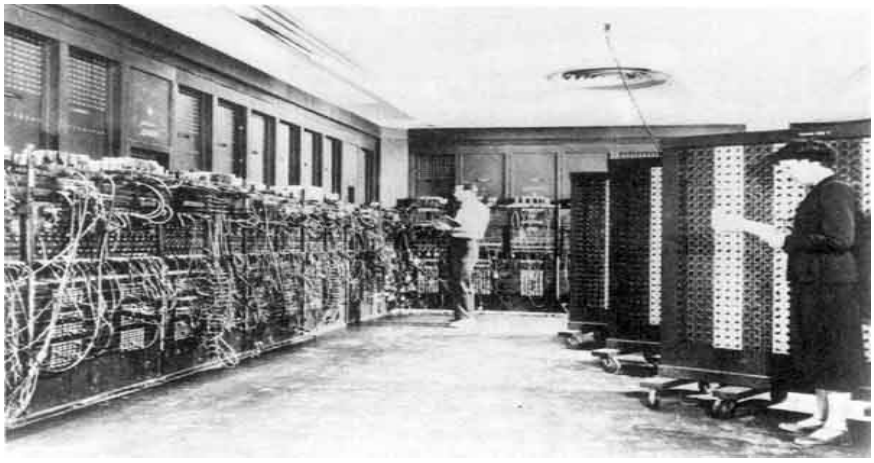
A II. világháború eseményei ösztönöző hatással voltak a korabeli számítástechnikai fejlesztésekre. A közvetlen feladat az ellenség kódolt üzeneteinek megfejtése volt. A német tengeralattjárók angol hajók elleni támadásai óriási károkat okoztak. A briteknek sikerült ugyan lehallgatni a németek titkos rádióüzeneteit, viszont dekódolni nem. Ezeket az üzeneteket egy ENIGMA nevű gépezet segítségével kódolták, aminek elődjét egy amatőr feltaláló és az Egyesült Államok egy korábbi elnöke, Thomas Jefferson tervezte.

Később az angol hírszerzésnek sikerült egy ENIGMA gépet szereznie. Ahhoz azonban, hogy megtudják az ellenség szándékát, hatalmas mennyiségű számításra volt szükség, amelyeket nagyon gyorsan kellet elvégezni, hogy az üzenetet még annak érvényességi időtartamán belül megfejtsék.

Az üzenetek megfejtésére az angol kormány titkos laboratóriumot állított fel, amelyben megépítették a COLOSSUS nevű elektronikus számítógépet. A gép tervezésében a neves brit matematikus, Alan Turing is meghatározó módon részt vett. A COLOSSUS 1943-ban vált működőképessé, de miután a brit kormányzat a kutatás gyakorlatilag minden egyes lépését 30 évre katonai titokká minősítette, így a számítástechnika fejlődésére a COLOSSUS alapvetően nem lehetett hatással. Megemlíteni azonban éredemes, hiszen ez volt a világ első elektronikus, digitális számítógépe.

A háború az Egyesült Államok számítástechnikájára is hatással volt. A hadseregnek a nehéztüzérség számára lőtáblázatokra volt szüksége. Ezeket a táblázatokat mechanikus, kézi számológépek segítségével, több ezer ember munkájával állították össze. A munkafolyamat időigényes volt, és gyakran fordultak elő benne hibák.

John Mauchley, ismerve a hadsereg számítási igényét — Atanasoff elképzeléseit felhasználva — javasolta egy elektronikus elven működő számítógép elkészítésének finanszírozását. A javaslatot az **ENIAC** (**E**lektronic **N**umerical **I**ntegrator **A**nd **C**omputer) megépítésére 1943-ban fogadták el. Az ENIAC csak a háború befejezése után, 1946-ban készült el, így eredeti rendeltésére már nem használták. A gépet egy tudományos rendezvényen bemutatták, ahol óriási érdeklődést váltott ki.



Forrás: www.cellnet.hu

1-3. ábra: ***Az ENIAC számítógép***

Az ENIAC jellemző áramköri eleme az elektroncső volt, melyből 18 ezer darabot tartalmazott. Programozása kizárólag gépi nyelven történt (gépi kód). Energia-felhasználása tetemes volt, és a meghibásodások rendkívül gyakoriak voltak. Átlagosan 15 percenként kellet javítani. Másodpercként mintegy 1.000 - 5.000 műveletet tudott végezni. A gép súlya 30 tonna volt, és a programozáshoz 6000 kapcsolót kellett kapcsolgatni.

Ezután sok más tudományos kutató kezdett elektronikus számítógépek építésébe. Az első működőképes az EDSAC (1949 Cambridge-i Egyetem) volt, majd követte a JOHNIAC (Rand Részvénytársaság), az ILLIAC (Illinosis-i Egyetem), a MANIAC (Los Alamos-i Laboratórium) a WEIZAC (Weizmann Intézet Izrael), és az EDVAC (Pennsylvania-i Egyetem).

Neumann János részt vett az ENIAC terveinek kidolgozásában, és ennek tanulságai alapján fogalmazta meg azokat az elveket, amelyeket ma is érvényesnek tartunk, és Neumann elvként ismerünk.

Neumann a tárolt program elvét alkalmazta az ENIAC-ban, majd elkészítette az ENIAC utáni számítógépnek, az EDVAC-nak a teljes leírását. Ez volt az első igazi tudományos dolgozat a számítógépekről.

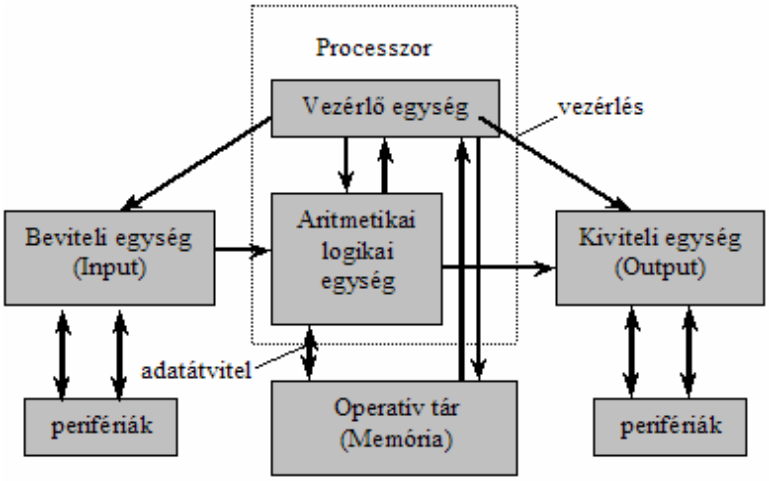
Neumann és Hermann Goldstine Princeton-ban megalkotta az IAS- vagy Neumann-gépet. Ez leginkább abban különbözött a korábbi két számítógéptől, hogy párhuzamos működésű volt, tehát sokkal gyorsabban számolt bármelyik korabeli számítógépnél, felépítése pedig – fő vonalaiban – megegyezett a mai modern számítógépekével.

Forrás: http://www.scl.ameslab.gov

1-4. ábra: ***Első generációs számítógépek alkatrészei***

A számítástechnika korszaka hivatalosan 1951 június 5-én kezdődött, amikor az első **UNIVAC**-ot (**U**niversal **A**utomatic **C**omputer) leszállították az Egyesült Államok Népszámlálási Hivatala számára. Ez volt az **első kereskedelmi forgalomban** elérhető számítógép.

Az IBM a mechanikus irodagépek mellett lyukkártya lyukasztók és mechanikus kártyaválogató gépek gyártásába kezdett, majd 1953 után az elektronikus számítógépek piacán is megjelent a 701-es típusú gépével, melyet tudományos számítások céljára fejlesztettek ki. Az utolsó elektroncsővel működő számítógépet 1958-ban gyártották (IBM 709).



1-5. ábra: ***Első generációs számítógép blokkvázlata***

Neumann János a számítógépek struktúrájára vonatkozó alapelvei szerint a gépnek 5 alapvető funkcionális egységből kell állnia:

- memória,
- bemenőegység,
- aritmetikai egység,
- vezérlőegység,
- kimenőegység.

A működésére vonatkozólag pedig az alábbi feltételeknek kell megfelelnie:

A gép legyen:

- bináris (digitális) működésű,
- belső programvezérlésű,
- soros feldolgozású,
- elektronikus eszközökből épüljön,
- memóriájában tárolja a programot és az adatokat (tárolt program elve),

Ez azt jelenti, hogy a gép a program utasításait az adatokkal együtt a központi memóriában bináris ábrázolásban tárolja, s a műveleteket - a Boole algebra műveleteit - ezek sorrendjében hajtja végre.

A számítógépek az első példányok megalkotása óta páratlan fejlődésen mentek keresztül, de elvi felépítésük nem változott. Csak az utólsó mintegy tíz évben kezdtek megjelenni a nem-Neumann-elvű gépek.

Az 1.5 ábrából kitűnik, hogy az első generációs gép processzor-centrikus, azaz a processzor közvetlenül irányítja minden részegység munkáját.

1.2.3. A második generáció – Tranzisztorok

Az 1948-ban a Bell Laboratóriumban feltalált tranzisztor tíz éven belül forradalmasította a számítógépfejlesztést, és az 1950-es évek végére a vákuumcsöves gépek már elavultnak számítottak. Az első tranzisztoros számítógépet az MIT Lincoln Laboratóriumában építették, majd az ott dolgozó mérnökök egyike, Kenneth Olsen 1957-ben megalapította a Digital Equipment Corporation-t (DEC). Első számítógépük a PDP-1 1961-ben készült el, és sebessége az akkori leggyorsabb gép, az IBM 7090-ének a fele, ára pedig kevesebb, mint tizede volt. Ma már elmondhatjuk, hogy ezzel a géppel született meg a kisszámítógép-ipar. (Az első számítógépes űrháborút, a MIT hallgatói programozták és játszották a gép 512x512 képpontból álló kijelzőjén. Ez volt a világ első videojátéka.) A PDP-8 az elődnél is olcsóbbra sikerült, és igen komoly üzleti siker lett.

A nagy számítógépek piacán az IBM a 7094-es típusával, és egy 1964-ben induló cég, a CDC pedig a 6600-ás gépével jellemezte ezt a korszakot.

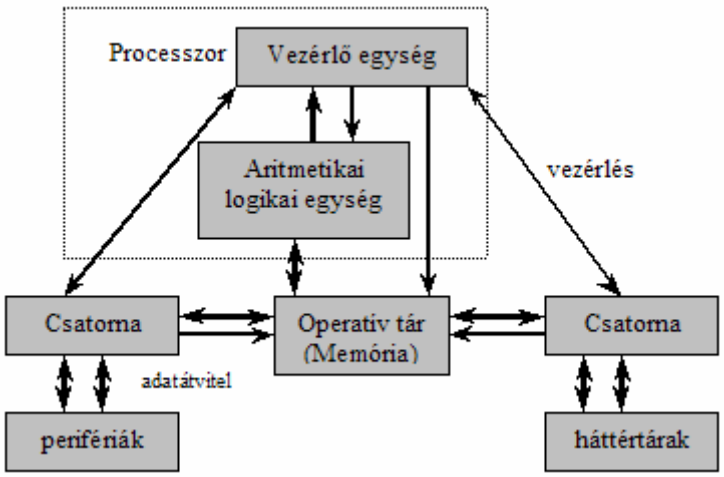
1-6. ábra: ***Az IBM7094 gépe***

A CDC gépe párhuzamos műveletvégzésre volt alkalmas, és mintegy tízszer gyorsabb, mint az IBM7094-e. A 6600-as évtizedekkel megelőzte saját korát. A modern számítógépekben található kulcsötletek nagy része visszavezethető az itt alkalmazottakra. A 6600-at a 7600 típus, majd a Cray-1 követte, amelyeket a mai fogalmaink szerint szuperszámítógép kategóriába sorolnánk. Tervezőjük, Seymour Cray, a számítástechnika egyik legendás alakja volt.

A második generációs gépek korszakára általánosan jellemző, hogy központi memóriaként megjelennek a ferritgyűrűs tárak, amelyek olcsóbbak, megbízhatóbbak és nagyobb kapacitásúak az előzőeknél. A gép alapvetően memóriacentrikus, azaz minden adat és utasítás a beviteli perifériáról a csatornán keresztül a memóriába kerül, a kiszámított eredmény is a memóriába íródik és szintén a csatornán keresztül kerül a kiviteli perifériára. Az átviteleket a csatorna (mint önálló automata) bonyolítja le, a processzor csak elindítja azokat. Háttértárként közvetlen elérésű mágneslemezeket kezdtek alkalmazni. A nagyobb tárkapacitás lehetővé teszi a könnyebben megírható, magasabb szintű nyelveken készített programokat gépi kódba átkódoló ún. fordítóprogramok használatát. Segítségükkel lehetővé válik az emberhez közelebb álló, problémaorientált nyelvek kidolgozása és széles körű elterjedése. Az első ilyen nyelvek a FORTRAN (FORmula TRANslator), ALGOL (ALGorithmic Oriented Language), COBOL (COmmon Business Oriented Language). Az

alkalmazási területek bővülnek, a számítógép teret nyer a gazdasági életben, s ez igen jótékony hatással van a további fejlődés ütemére.

Az 1.7 ábrán látható a 2. generációs számítógépek blokkvázlata.



1-7. ábra: ***Második generációs számítógép blokkvázlata***

1.2.4. A harmadik generáció - Integrált áramkörök

A harmadik generáció időszaka 1965-től 1980-ig tart. Az integrált áramkör lényegében több tranzisztor, és a köztük lévő kapcsolat egy közös szilícium-lapkán, ugyanannyi gyártástechnológiai lépésben előállítva, mint egyetlen tranzisztor. A tranzisztorokat kiszorítják az integrált áramkörök, melyek akkor még alacsony és közepes integráltsági fokúak. Az ezekkel épített gépek kisebbek, gyorsabbak és sokkal olcsóbbak, mint tranzisztoros elődjeik

Megváltozik a gépek struktúrája, átalakulnak funkcionális egységei, s kialakul a fejlett, egységes csatornarendszer, amely közvetlen kommunikációs kapcsolatot biztosít az egységek között.

Rendszertechnikailag ezek a gépek memóriacentrikusak, a funkcionális egységek, perifériák közvetlenül — a központi egységtől, a processzortól függetlenül — igénybe vehetik a központi memóriát.

A periferiális eszközök széles skálája alakul ki. A perifériák és a központi egység sebessége közötti különbség kiegyenlítésére kidolgozzák a multiprogramozás módszerét és az időosztásos rendszert. Előbbi azt jelenti, hogy ha a központi egységnek valamilyen okból szüneteltetnie kell valamelyik program futtatását, akkor átvált egy másik programra, utóbbi pedig azt, hogy a gépbe beépített (real time) időzítő segítségével osztja szét erőforrásait a különböző feladatok között.

A megelőző fejlesztések, bár újabb és egyre jobb gépeket produkáltak, alapvetően hardver orientációjúak voltak. Az eltérő igények kielégítésére, az üzleti számításokra és a műszaki-tudományos számításokra külön fejlesztettek gépeket. A kompatibilitás, azaz, hogy egy régebbi típusra megírt program az új fejlesztésű gépen is fusson, nem volt szempont.

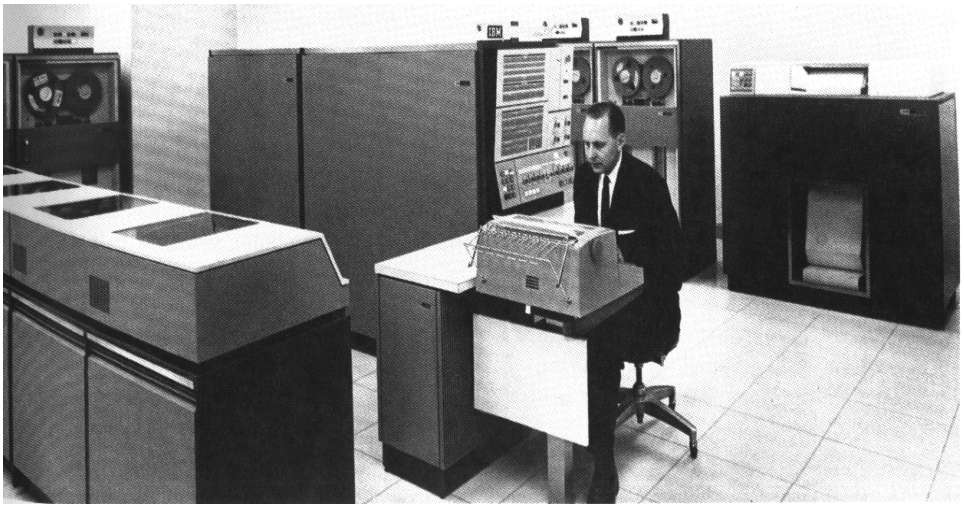
Az IBM új gépének fejlesztésénél radikális lépést tett. Egyetlen termékvonalat vezetett be, a System/360-at, ami az integrált áramkörökön alapult, és mind tudományos, mind kereskedelmi számításokra alkalmas volt. A System/360 sok újítást tartalmazott, ezek közül a legfontosabb, hogy kb. fél tucat gépből álló, ugyanazt az assembly nyelvet használó gép-családot alakított ki, különböző mérettel, és teljesítménnyel és természetesen árral. Az egyiken írott szoftver elvileg, futott a másikon is. Gyakorlatban egy kisebb modellre írt

szoftver működött a nagy modellen (felülről kompatibilitás), de visszafelé ez nem volt teljesíthető.

A harmadik generációs gépek belső tárkapacitása eléri a megabájtnagyságrendet, műveleti sebessége 5 MIPS-et (azaz mintegy ötmillió utasítás végrehajtását másodpercenként). Jellemző vonásuk az univerzalitás és a több-felhasználós párbeszédes üzemmód. A felhasználási területek tovább bővülnek, megjelennek az információs rendszerek, a technológiai, folyamatirányítási, gazdasági-vezetési rendszerek, a számítógépes tervezés csírái stb.

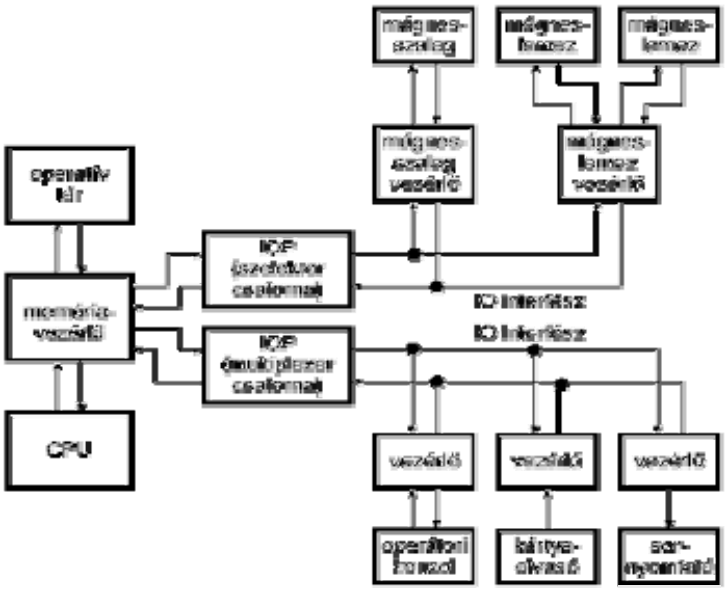
Az 1.8 ábrán a számítógép a két nagy szekrényben, a terem közepén látható, az egyiknek az oldalán a korábbi generációkból örökölt vezérlőpult, amin az egyes regiszterek állapotát lehetett leolvasni és beállítani.

A kép közepén van az operátori ún. konzolírógép: ezen keresztül lehetett parancsokat adni az operációs rendszernek és az üzeneteket is ide írta ki a gép. Balra elöl mágneslemezes egységek láthatók, a háttérben mindkét oldalon mágnesszalag-meghajtók, jobboldalt hátul pedig egy sornyomtató.



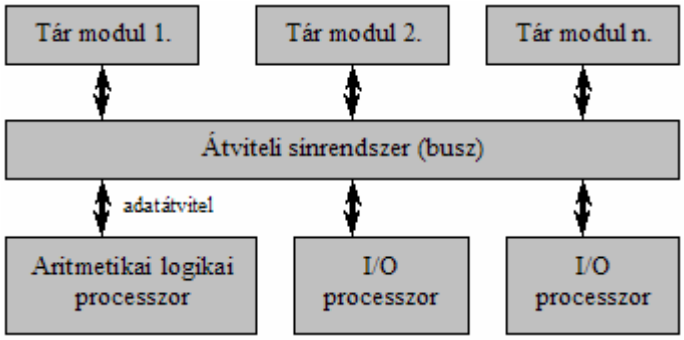
1-8. ábra. ***IBM System/360- as számítógép***

Az IBM System/360 felépítésének blokk vázlatát a 1.9 ábra mutatja be.



1-9. ábra: Az IBMSystem/360 harmadik generációs számítógép blokkvázlata

A DEC PDP családjának első gépei kapcsán már utaltunk arra, hogy a technológia igen gyors fejlesztése lehetővé tette, hogy néhány év eltolással nagyságrendileg ugyanaz a számítási teljesítmény megvalósítható legyen jóval kisebb méretben és jóval olcsóbban, egy másik kategóriában, a miniszámítógépekében. A minikomputer-világ is hatalmas lépést tett előre a 3. generációban a DEC PDP-8-as családját követő, 16 bites PDP-11-es széria bevezetésével. Az architekrúra részletesebb ismeretében megállapítható, hogy a PDP-11-es széria mintha csak a kistestvére lenne a 360-asnak, csakúgy, mint a PDP-1-es a 7094-esnek. A 360-as és a PDP-11-es is szóorientált regiszterű és byteorientált memóriájú, mindkettőnek igen jó az ár/teljesítmény aránya. A PDP-11 rendkívül sikeres volt, különösen az egyetemeken, és biztosította a DEC vezető helyét a többi minikomputer gyártónál.



1-10. ábra: Sínrendszerre alapozott architektúra. A harmadik generációs gépeknél jelenik meg, és a negyedik generációban válik általánossá.

1.2.5. A negyedik generáció - Nagyon nagy mértékű integráltság

A számítógépek negyedik generációját az 1970-es évektől napjainkig számíthatjuk. Az első negyedik generációs gép az IBM 370-es rendszere. A negyedik generációs gépek hardverfelépítésére jellemző, hogy igen nagy integráltságú áramkörökből épülnek fel, és megjelennek a mikroprocesszorok. Az integrált áramkörök gyártástechnológiájának továbbfejlődése (LSI – Large Scale Integration: magas integráltsági fokú áramkör) tette lehetővé a nagykapacitású memória-áramkörök gyártását. Az LSI technológiával más, többféle, speciális feladatú áramkört is kialakítottak:

- a **mikroprocesszort**, ami egyetlen félvezető lapkára került;
- a különböző tárolási feladatokat ellátó **memória áramköröket;**
- egyéb **kiegészítő áramköröket** (órajel-generátorok, meghajtók, időzítők stb.);
- **programozható I/O** (beviteli és kiviteli) **áramköröket.**

A negyedik generációs számítógépek szervezésében nincsenek alapvető változások, a korábban bevett megoldásokat tökéletesítik. A negyedik generáció másik jellemzője, hogy a szoftvergyártás óriási méretűvé válik. A szoftverek árai elérik, egyes esetekben meg is haladhatják a hardverét.

A harmadik generációs számítógépek ára és üzemeltetési költsége olyan magas volt, valamint működtetésük olyan speciális szakértelmet igényelt, hogy külön speciális részlegeket, **számítógép központokat** kellett fenntartani.

Az 1980-as évekre az alkatrészek nagymértékű integráltsága lehetővé tette, hogy a tranzisztorokból milliós nagysegrendű darabot integráljának egy egyszerű chipre. Ez a fejlődés egyre kisebb és egyre gyorsabb komputerekhez vezetett. Az egyre kisebb méretű és árú gépek megjelenésével már egyes részlegek is képesek voltak saját számítógépet venni. Az 1980-as évek végére az árak annyira leestek, hogy magánembereknek is lehetett már saját gépük. A PC-k, a személyi számítógépek korszaka ezzel elkezdődött.

Az első mikroprocesszoros személyi számítógépek inkább csak hobbyeszköznek számítottak. Alkatrészeiket egységcsomagonként árusították, és egy kevés elektronikai ismeret, és némi kézügyesség után bárki összeszerelhette az első számítógépét, egy Intel 8080-as mikroprocesszor köré építve. Szoftverrel nem volt ellátva, a felhasználónak magának kellett megírnia. Később a Gary Kildall általkészített CP/M operációs rendszer nagyon népszerű lett a 8080-ason. Ez egy igazi (floppy) lemezes operációs rendszer volt, file szervezésű, és az operátori parancsokat közvetlenül a klaviatúráról írhatta be a felhasználó.

A korai PC-k másik csoportját az Apple és később az Apple II jelentette, melyeket Steve Jobs és Steve Wozniak terveztek egy híressé vált garázsban. Ez a gép rendkívül népszerű volt az otthoni felhasználók körében. Szintén a korszak kedvelt gépei voltak a Commodor, az Amiga és az Atari. Ezek gyártói az Apple-hez hasonlóan nem Intel processzort alkalmaztak.

IBM, mint a számítógép ipar domináns ereje, viszonylag későn határozta el, hogy kiveszi a részét a PC üzletből, de a számítástechnikai iparban képviselt súlyával a legtöbb konkurens személyi számítógépgyártót azonnal lesöpörte. Rá nem jellemző módon azonban a gép terveit nem tartotta titokban, sőt szabadalommal sem védte le, hanem a terveket, a kapcsolási rajzokkal együtt egy 49 $-os könyvben megjelentette. Mindezt abból a megfontolásból tette, hogy lehetővé tegyék más számítógépes cégeknek az IBM PC-hez való csatlakozást, ezzel is növelve annak népszerűségét és alkalmazhatóságát. Az IBM szerencsétlenségére, mivel a tervek most már teljesen publikusak voltak, és a géphez szükséges valamennyi alkatrész beszerezhető volt a kereskedelmi forgalomban, a tervek alapján számos cég elkezdte a PC-k **klónjait** (hasonmásait) gyártani, általában sokkal olcsóbban, mint az IBM.

Ebben az időszakban a számítógépek előállítási költsége és az alkalmazott technológia lehetővé tette, hogy speciális feladatok elvégzésére, sőt házi használatra egyszerű szerkezetű, a mikroprocesszorhoz közvetlenül kapcsolt memóriával és I/O áramkörökkel „olcsó” számítógépek készüljenek. Ezeket a gépeket **mikroszámítógépeknek** hívjuk, és tömeges elterjedésük tette lehetővé, hogy a számítógép mindennapjaink eszközévé vált. Ebbe a kategóriába tartoznak a személyi számítógépek is. Speciális feladatokat látnak el az **egychipes mikroszámítógépek,** amelyeknél az összes funkciót megvalósító áramkört egyetlen félvezető lapkán alakították ki. Ezek legnagyobb felhasználója a háztartási- és szórakoztató elektronika, valamint az irányítástechnika.

1.3. A számítási modell

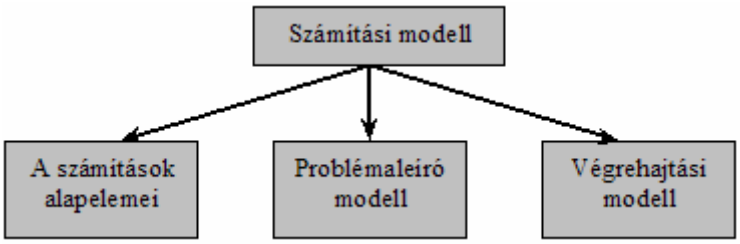
Ha egy teljesen új elven működő számítógép elkészítése a cél, akkor először egy új számítási modellt kell kidolgozni, majd megírni az ennek megfelelő programnyelvet, és ehhez kell elkészíteni a számítógépet.

1966 óta többfajta számítási modellt határoztak meg. Jegyzetünkben a számítási modell leírásánál Sima Dezső csoportosítását használjuk.

Az IBM 370-nel a 70-es évek végére Neumann-architektúrájú gépek fejlesztésében elérték a lehetőségek korlátait, teljesítményüket nem lehetett már jobban növelni. Ezért olyan újdonságok kerültek előtérbe, mint a RISC architektúra, futószalag elv (pipeline), valmint az új programnyelvekben rejlő lehetőségek (konkurens parancsnyelvek, objektum orientált nyelvek). Ezek már számítási modelljükben különböznek.

A számítási modell három absztrakció együttese:

- A számítások alapelemei. (Mi az, amin végrehajtjuk a számítást?)
- A probléma leírás modelljei. (Hogyan képezzük le a számítási feladatot?)
- Végrehajtás modellje. (Mi vezérli a végrehajtást?)



1-11. ábra: ***A számítási modell értelmezése.***

Az 1.11 ábrából kitűnik, hogy a számítási modell magasabb szintű absztrakció, mint a programozási nyelv, vagy a számítógép fogalma. Ennek alapján a programozási nyelv olyan specifikációs eszköz, amely egy bizonyos számítási modellt feltételezve lehetővé teszi számítási feladatok megfogalmazását, a számítógép pedig a programozási nyelv segítségével leírt számítási feladat végrehajtására szolgáló eszköz, amely egy adott számítási modellnek megfelelően működik.

A számítások alapelemeiként értelmezzük a számításokban megadható adatokat, műveleteket, objektumokat, üzeneteket, függvényeket, argumentumokat, predikátumokat, formulákat. Például a Neumann elvű számítási modellben a számítások alapelemei az adatok és a rajtuk végrehajtható műveletek. Az adatok a végrehajtás során neveket kapnak a megkülönböztethetőség érdekében. A programokban változóként hivatkozunk rájuk, és az architektúra szempontjából vizsgálva egy-egy memória vagy regiszter cím reprezentálja ezeket.

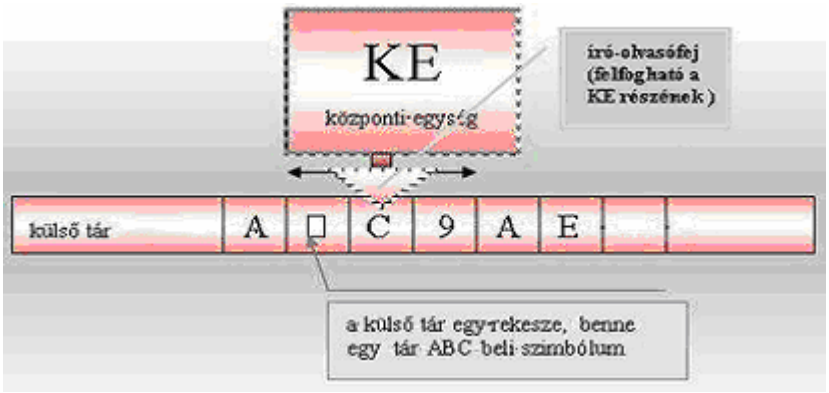
1.3.1. Adatalapú számítási modellek

Számítási modellek közül a legismertebbek adatalapúak. Ezek közé tartozik a:

- **a Turing-modell,**
- **a Neumann- modell és**
- **az adatfolyam-modell.**

A **Turing-gép** fogalmát Alan Turing angol matematikus dolgozta ki 1936-ban megjelent cikkében a matematikai számítási eljárások, algoritmusok precíz leírására, tágabb értelemben pedig, mindenfajta „gépies” problémamegoldó folyamat, pl. a számítógépek működésének modellezésére. Turing bebizonyította, hogy létezik univerzális automata, amely

minden speciális géppel előállított sorozatot szintén képes előállítani, másrészt megmutatta, hogy minden speciális automata leírható véges hosszúságú utasítássorozattal. A Turing-gép úgynevezett absztrakt automata, a valóságos digitális számítógépek nagyon leegyszerűsített modellje (1.12 ábra). Ez az elképzelt gép három hardveres egységből áll a szalagtárból (memória és input-output-perifériák), a vezérlőegységből (CPU) és az író-olvasó fejből (buszrendszer). Szembetűnő lehet, hogy a Turing-gép mennyire hasonlít a megvalósított számítógépek felépítéséhez.



Forrás: www.wikipedia.hu

1-12. ábra: ***A Turing gép***

A **Neumann-féle modell** jellemzői:

- Tárolt program elve, azaz utasítás és adat azonos közegben és azonos formában van tárolva. A tárolóhelyek egydimenziósak és lineárisan címezhetőek, valamint újraírhatóak.
- A soron következő végrehajtandó utasítás kijelölésére utasításszámlálót használ. Az utasításszámláló által címzett memóriahelyen lesz a következő utasítás. Az, hogy egy tárolóhelyen levő bináris információ adat-e vagy utasítás, csak annak értelmezésétől függ.
- Az utasításokat szekvenciálisan hajtja végre, melyre külön egysége van, az aritmetikai-logikai egység, az ALU. (A Neumann-elven működő gépeket szokás vezérlésáramlásos modellnek is nevezni.)

A Neumann elven működő gép **működési elve** röviden:

- A végrehajtandó gépi kódú utasítások és a hozzájuk tartozó adatok a memória rekeszeiben vannak.
- A processzor a memóriából beolvassa a soron következő gépi utasítást, majd értelmezi (dekódolja).
- Az ALU végrehajtja az utasítást.
- A végrehajtás eredménye a regiszterbe, esetleg a memória megfelelő rekeszébe kerül.
- Folytatódik a soron következő utasítással.

A Neumann-féle modellben a **számítások alapelemei**:

- adatok és
- műveletek.

A többszöri értékadás megengedett, az adatok nevesítettek, új érték hozzárendeléséig marad a régi érték, procedurális jellegű modell, utasítások sorozatával adjuk meg a számítási modellt.

A Neumann-féle **végrehajtási modell**:

Megadott utasítássorozat végrehajtása a bemenő adatok felhasználásával az utasítás szemantikája által módosítva történik, közvetlen vezérléssel, soros jellegű végrehajtással.

A végrehajtás pedig állapotátmenetet eredményez.

A végrehajtás során a gép (mint automata) állapotát az összes deklarált változó, a speciális regiszterek (utasítás számláló regiszter/PC Program Counter), valamint az állapot jellemzők aktuális értéke adja meg.

Főbb problémái:

- többszöri értékadás minden változó aktuális értéke múlt-érzékeny, az utasítás végrehajtás sorrendjére érzékeny, nem lehet átrendezni az utasítások sorrendjét, nem szándékolt állapotváltozás jöhet létre (mellékhatás), nehézkes tesztelés
- közvetlen vezérlés a soros feldolgozás időigényes

(A Neumann-féle gép felépítésével a későbbiekben részletesen foglalkozunk.)

Az adatfolyam gép logikai struktúráját az elvégzendő műveletek egymáshoz kapcsolódását leíró *adatáramlási gráf* határozza meg. A gráf csomópontjaiban szeparált processzorok állnak rendelkezésre minden operációra. A gráf éleihez a csomópont által igényelt bemeneti adatok (token-ek), illetve a művelet eredményeként keletkező kimenő adatok vannak hozzárendelve.

Egy operáció lehet aritmetikai, logikai, függvényhívás stb. A vezérlésáramlásos (hagyományos, Neumann-elvű gépek) számítógépek soros utasítás-feldolgozásával szemben, a műveletek végrehajtása a feladatokhoz szükséges adatok rendelkezésre állásától függ.

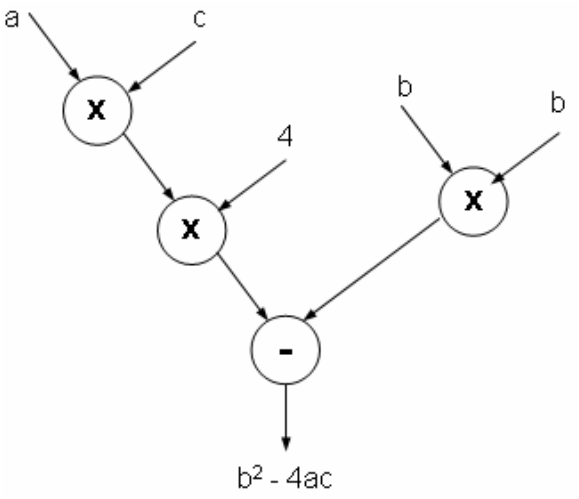
Az adatáramlási gráf csomópontjaihoz (nodes) vannak hozzárendelve az elvégzendő műveletek (utasítások). Az operációk (processzorok) várnak, amíg az operandusok értéke a bemenetükön előáll, majd előállítják eredményüket, és valamely másik - a gráf élei által meghatározott - csomópont felé továbbítják azt.

A processzorok (operációk) függetlenek. Az operációk végrehajtásának sorrendje az adatfolyamból adódik.

Egy csomópont feladataihoz tartozik tehát:

- bemeneti adatok (token) fogadása,
- a művelet (operáció) elvégzése,
- az eredmény továbbítása másik csomópontba (switch).

Az adatvezérelt számítógépek multiprocesszoros rendszerek, amelyben a processzorokat egy kapcsolóhálózaton keresztül kötik egymáshoz és minden ütemben az adatok (tokenek) egy-egy processzor-pár között mozognak a tokenekben előírt módon. Az 1.13 ábra egy egyszerű példán keresztül szemlélteti az adatvezérelt modell működését.



1-13. ábra: A b^2 - 4ac kiszámítása adatfolyam struktúrában.

A Neumann-féle és az adatfolyam-számítási modell összehasonlítását szemléltei az 1.1 táblázat.

1-1. Táblázat: ***A Neumann-féle és az adatfolyam-számítási modell összehasonlítása***

Neumann-féle számítási modell	*Adatfolyam-számítási modell*
Közös memória (adat + program)	Műveletvégzőben „tárolhatóak” az adatok
Változó	Egyszeri értékadás (a bejött adat elvész)
Adatmanipuláló utasításokkal	Adatfolyam gráffal
Implicit szekvencia	Adatvezérelt
Explicit vezérlésátadás	Nincs utasításszámláló, nincs vezérlési szekvencia

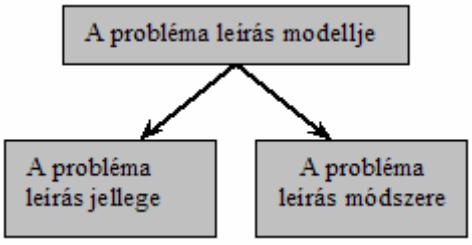
1.3.2. Nem adatalapú számítási modellek

Több, nem adatalapú számítási modell is létezik:

- Objektumalapú számítási modellekben a számítások alapelemei az objektumok a nekik küldhető üzenetekkel, ezek meghatározott műveletsorozatok végrehajtását váltják ki.
- Applikatív számítási modellben a számítások alapelemei az argumentumok és az azokon értelmezett függvények.
- Kijelentés-logikán alapuló számítási modellnél az alapelemek a halmazok elemei és a rajtuk deklarált predikátumok.

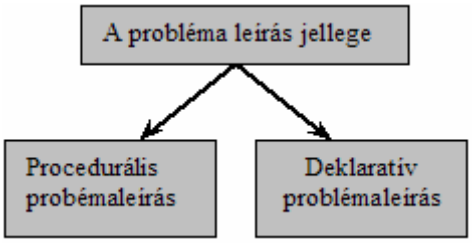
1.3.3. A problémaleírás modellje

Az, hogy milyen modellt választunk a probléma leírására, meghatározza a probléma leírásának jellegét és a leírás módszerét.



1-14. ábra: ***A problémaleírás modellje***

A probléma leírásának jellege alapján *procedúrális* vagy *delkaratív* leírású lehet.



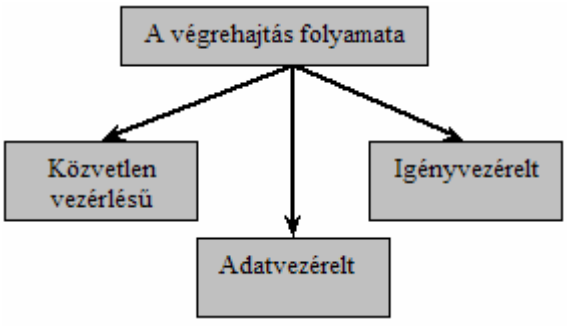
1-15. ábra: ***A problémaleírás jellege***

- A procedurális problémaleírás alapja a feladatmegoldási algoritmus. Fontos az utasítások sorrendje. Általában több, azonos eredményt adó algoritmussal is megoldható ugyanazon probléma. A probléma egy konkét megoldási algoritmusa a feladat egy lehetséges egyedi megoldási módját jelenti.
- A deklaratív problémaleírás az adott problémával kapcsolatos szignifikáns tények és összefüggések megadásán alapul. Ezek a tények és összefüggések vagy függvények, vagy predikátumok (kijelentések) formájában fejezhetők ki. Deklaratív problémaleírás esetén az utasítások sorrendjének nincs jelentősége, ellentétben a procedurális jellegű problémaleírással. A deklaratív leírás magasabb absztrakciós nyelven, sokkal tömörebben írja le az adott problémát, mint a a procedurális.

1.3.4. A végrehajtás modellje

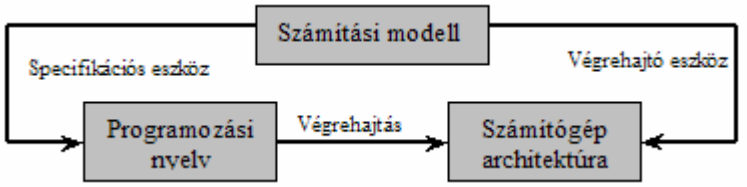
A végrehajtási modell összetevői:

- a számítások végrehajtásának értelmezése, amely a problémaleírás modelljével kölcsönösen meghatározza egymást,
- a végrehajtási szemantika, amely a számítások egyes lépéseinek végrehajtási módját írja elő,
- a végrehajtás módja, ami pedig a végrehajtás folyamatának vezérlését határozza meg (1.16 ábra).
  - Közvetlen vezérlés: a végrehajtást program vezérli, az utasítás sorrendje határozza meg a végrehajtást (implicit módon), de explicit módon el lehet térni (vezérlés átadás utasítás/Neumann, objektum alapú)
  - Adatvezérelt: egy-egy művelet azonnal végrehajtódik, ha a szükséges adatok rendelkezésre állnak (adatfolyam típusú modell jellemzője)
  - Igényvezérelt: minden műveletet az utolsó pillanatig elodázzák, csak akkor hajtja végre a műveleteket, ha a végeredményhez kell (applikatív számítási modell)

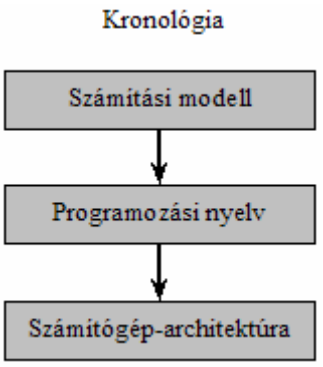


1-16. ábra: ***A végrehajtás modellje***

Az 1.17 és az 1.18 ábrán látható, hogy a számítási modell megalkotása megelőzi egy új elven alalpuló számítógép megjelenését, mert a számítási modellen alapuló gépi programnyelv alapján lehet magát az architektúrát megalkotni.



1-17. ábra: A számítási modell, a programozási nyelv és a számítógép kapcsolata



1-18. ábra: A számítási modell, a programozási nyelvek és a számítógép megjelenési sorrendje.

1.4. Architektúra fogalmak

Azt a kifejezést, hogy „számítógéparchitektúra” először az IBM System/360-as gépcsalád tervezői használták. Értelmezésük szerint „a számítógéparchitektúra alatt a számítógép azon felépítése értendő, amelyet egy alacsony szintű nyelven programokat fejlesztő szakembernek kell ismernie ahhoz, hogy korrekt programokat tudjon írni a gépre”.

Ebbe az értelmezésbe beletartozott az utasításkészlet, az utasításszerkezet, a címzési módok, a regiszterek és a memória deklarálása, de a tényleges hardverstruktúra (implementáció) és annak (áramköri) megvalósítása viszont nem.

1.4.1. A számítógép-architektúra szintjei

A többszintű hierarchikus leírás a számítógéparchitektúrát négy szinten értelmezte (BELL és NEWELL 1970). A négy szint:

- áramköri tervezés,
- logikai tervezés,
- programozás

- processzor-memória sínek.

Az architektúra fogalom változásának következő fázisában az architektúrát először a funkcionális specifikációra és a hardver implementációra is kiterjesztették, majd a többszintű értelmezést kétdimenziósra bővítették (SIMA, 1977). Itt a koordináta egyik tengelye az absztrakció szintjét jelentette a többszintű hierarchikus leírásnak megfelelően, és a merőleges tengely a leírás irányultsága dimenziót jelenti. Ennek megfelelően különbséget tesznek logikai és fizikai architektúra között. Így megkülönböztethető

- tényleges hardver implementáció és
- funkcionális specifikáció.

1.4.2. Számítógép architektúrák értelmezése absztrakciós szinteken

- Mikrogép-szint (Megjegyzendő, itt a mikrogép nem mikroszámítógépet jelent, hanem a processzor mikroprogramozott működésére utal.)
- Processzor-szint
- Számítógép rendszer-szint
- Operációs rendszer-szint

Az architektúra fogalmának értelmezése mikrogép-szinten

Ha a számítógép mikroprogramozott processzorral működik, akkor van jelentősége a mikrogépi szintű architektúrának. Az erre vonatkozó absztrakt automata a mikroprogramozási modellt, a konkrét architektúra a mikrogép belső felépítését és működését írja le. Alapvetően a soros működésű, mikroprogram-vezérelt számítógépeknél van jelentősége, így ez napjainkban elvesztette fontosságát.

Az architektúra fogalmának értelmezése processzor-szinten

Ezt a szintet gyakran nevezik mikroarchitektúrának. A szakirodalomban a processzorok architektúráját rendszerint csak a működésük leírásával, valamint a funkcionális egységek (regiszterek, végrehajtó egységek, sínek) és azok logikai kapcsolatát bemutató blokkdiagram formájában adják meg.

A mikroarchitektúra fizikai felépítését tartalmazó dokumentációk általában nem publikusak.

A processzor hardvermodelljének főbb komponensei:

- a megszakítási illesztőfelület
- a programozói felület és
- az I/O illesztőfelület

A **programozói modell** megegyezik a gépi nyelv leírásával. Igy a processzorszintű absztrakt architektúrát programozási modell értelemben (ISA) utasításszintű architektúrának is szokás nevezni. (ISA Instruction Set Architecture).

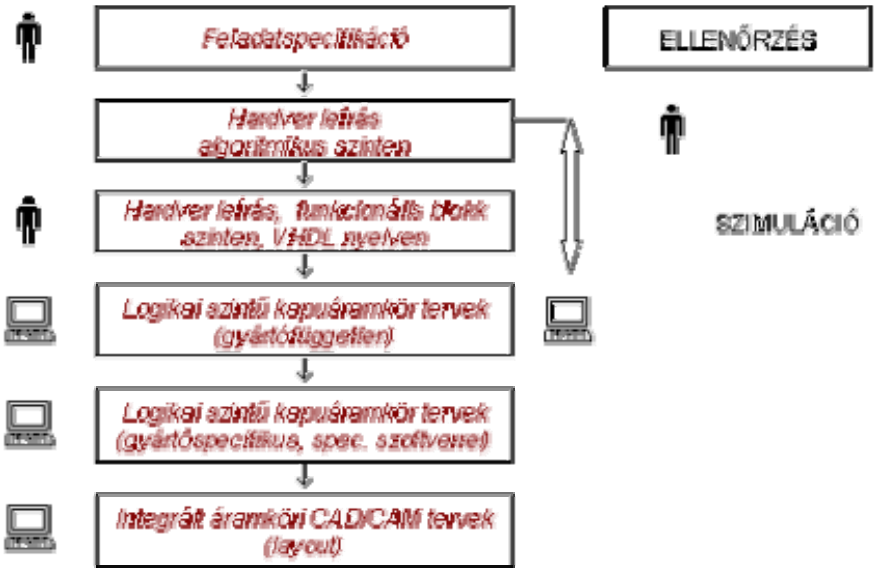
Az architektúra fogalmának értelmezése számítógép rendszer-szinten

A konkrét architektúra leírása rendszerszinten magában fogalja funkcionális elemeket (processzorok, memórák sínek stb). Ezen kívül tartalmazza a közöttük lévő kapcsolatok, valamint a rendszer működésének specifikációját. A hardver az 1980-as években már rendkívül bonyolulttá vált. Ha belegondolunk abba, hogy a korai PC-k is milliós nagyságrendű alapépítőelemből (tranzisztor, kondenzátor, flip-flop) álltak, érthető, hogy ez a mennyiség a gépeket tervező ember számára áramköri és logikai szinten már gyakorlatilag áttekinthetetlenné vált. A megoldásban is a számítógép segített, de ehhez előbb el kellett készíteni egy rétegmodellnek megfelelően kialakított hardver leírónyelvet. Ezzel a leíró nyelvvel kapcsolatban megfogalmazott elvárások a következők voltak:

- adjon lehetőséget az egyes rétegnek megfelelően egy szabványos „nyelven” a hardver leírására,
- tegye áttekinthetővé magasabb szinteken az ember számára is a hardver dokumentációját,
- az alsó szinteken tegye lehetővé a tervek átadását a gyártás CAD/CAM rendszereinek,
- biztosítsa a hardver szimulációs bevizsgálását.

Ebből kiindulva fejlesztették ki a VHDL (Very High Speed Integrated Circuit Hardware Description Language) nevű hardverleíró nyelvet a 80-as évek közepén, melyet azóta nemzetközi szabványnak is elfogadtak. A VHDL alkalmazásával a hardver fejlesztése a hardver rétegmodelljének megfelelően, a hierarchikus szintek szerint, felülről-lefelé (TOP-DOWN) történik:

A hierarchia legfelső három szintjén az ember tervezi meg a gépet azzal, hogy specifikálja a feladatot, majd elkészíti algoritmusszinten a hardver leírását, legvégül pedig VHDL nyelven elkészíti a funkcionális blokkok leírását.



Forrás: Budai: Mikroszámítógép rendszerek

1-19. ábra: ***A hardvertervezés rétegmodellje***

A negyedik szinttől kezdve, a tervezési feladatokat számítógépes rendszerekkel végeztetik el. Ennek helyességét számítógépes szimulációval ellenőrzik, melyet a tervező irányít és értékel.A VHDL tervezés absztrakciós szintjeit, a működésleírási módot, valamint a tervezési alapelemeket az 1.3 táblázat foglalja össze.

1-2. Táblázat: ***A VHDL absztrakciós szintjei***

	A működés leírási módja	Alapelemek
Architekturális szint	A teljesítmény meghatározása	CPU-k, tárolók, vezérlők, kapcsolók
Algoritmikus szint	Algoritmusok és adatszerkezetek kezelése	Hardverelemek, adatstruktúrák
A funkcionális egységek szintje	Műveletek, regiszterátvitelek, állapotátmenetek	ALU, regiszterek, mikrovézerlők, mikrotárolók
Logikai szint	Logikai (Boole-) egyenletek	Kapuk, átmeneti tárolók, flipflopok
Áramköri szint	Differenciálegyenletek	Tranzisztorok, ellenállások, kapacitások

Az architektúra fogalmának értelmezése az operációs rendszerek szintjén

Az **operációs rendszer absztrakt architektúrája** alatt az operációs rendszer funkcionális leírása értendő. Egy operációs rendszer funkcionális specifikációjának megadásához négy felület megadása szükséges (FUNCK 1984):

- felhasználói felület (parancsnyelv megadása),
- alkalmazói programok felé irányuló felület (rendszerhívások köre),
- periféria vezérlőegység logikai szintű felülete (I/O kérések formájának leírása/eszközleíró táblázat),
- adathordozók felé mutató felület (adatábrázolás/adatkezelés a háttértárolókon).

Az **operációs rendszer konkrét architektúrája** az építőelemeinek, kapcsolatainak és működésének a leírását jelenti.

Az operációs rendszer fő összetevői:

- A tárkezelő
- A processzorkezelő
- Az I/O kezelő

Az operációs rendszer elemei közötti kapcsolatok alatt az egyes elemek közötti kommunikációt és szinkronizációt értjük.

Ellenőrző kérdések:

1. Ismertesse az első generációs számítógépek főbb jellemzőit!
2. Mit jelent az a kifejezés, hogy az első generációs számítógépek processzorcentrikusak?
3. Ismertesse a második generációs számítógépek főbb jellemzőit!
4. Mit jelent az a kifejezés, hogy a második generációs számítógép memóriacentrikusak?
5. Ismertesse a harmadik generációs számítógépek főbb jellemzőit!
6. Milyen felhasználási területei voltak a harmadik generációs gépeknek?
7. Ismertesse a negyedik generációs számítógépek főbb jellemzőit!
8. Milyen architektúrabeli különbségek vannak a harmadik és a negyedik generációs gépek között?
9. Mit értünk számítási modell alatt?
10. Milyen adatalapú számítási modelleket ismer?
11. Ismertesse a problémaleírás modelljét!
12. Ismertesse a végrehajtás modelljét!
13. Mit értünk számítógéparchitektúra alatt?
14. Ismertesse a számítógéparchitektúrák értelmezésének szintjeit!

2. Számítógép-rendszerek

2.1. Processzorok

A processzor (vagy angol rövidítésével élve a CPU) (Central Processing Unit) a számítógép azon egysége, amely tartalmazza az utasítások értelmezését és végrehajtását vezérlő áramköröket. A korszerű számítógépekben a CPU az alaplapon helyezkedik el, annak egyik legfontosabb része.

2.1.1. A CPU felépítése

Bár az egyes processzorok felépítésében jelentős eltéréseket tapasztalhatunk, de mindegyikre jellemzőek a következő legfontosabb architekturális építőelemek:

vezérlőegység (CU = Control Unit),

aritmetikai és logikai egység (ALU = Arithmetic Logic Unit),

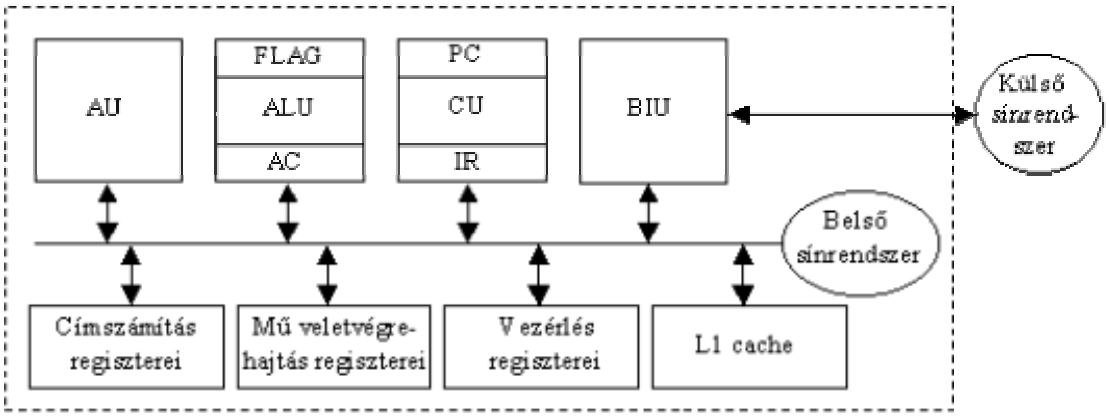
regiszterkészlet,

buszillesztő egység (BIU = Bus Interface Unit),

címszámító és védelmi egység (AU = Adress Unit),

belső gyorsítótár (L1 cache),

- az előző négy részegység kommunikációját biztosító eszközök (mikroszámítógépeknél a belső sínrendszer).



2-1. ábra: A processzor architektúrális elemei

Az **aritmetikai és logikai egység** (ALU) - mint a nevében is benne van - a CPU-n belül az utasításkódokban előírt a számítási és logikai műveleteket végzi el. Általában egyszerű bináris számokkal végzett műveletek végzésére alkalmas, pl.: összeadásra, Boole algebrai műveletek elvégzésére, komplemes képzésre, valamint adatok léptetésére bitenként jobbra vagy balra. Minden egyéb adatkezelési művelet, amelynek elvégzése a CPU feladata, felbontható az előbb felsorolt alapműveletekre. A legegyszerűbb esetben is fixpontos bináris összeadóból, komplemens képzőből, léptető regiszterből és logikai műveleteket végző részből áll.

Az aritmetikai egységhez működéséhez a következő két regiszter mindig hozzátartozik:

- AC = **Ac**cumulator Register, mely a művelet-végrehajtásnál az adatok (operandus) átmeneti tárolására szolgál,

- FLAG regiszter = Állapotjelző regiszter, melyben a végrehajtott utasítás következtében megváltozott állapotok kerülnek bitenként kódolásra (pl. paritáshiba lépett fel, a felhasználói program 0-val akart osztani, stb.)

A **vezérlő egységnek** a feladata a programban lévő utasítások alapján a teljes számítógép részegységeinek (aritmetikai egység, memória, kommunikációs eszközök, háttér és perifériavezérlések) irányítása, összehangolása. A vezérlő egység, az utasításregiszterben (IR) megjelenő utasítás értelmezésével, vezérlő jeleket ad ki a processzor belső és a számítógép processzoron kívüli részegységeinek irányítására. A belső vezérlő jelek az aritmetikai egység működését és a regiszterek közötti adatutak nyitását/zárását irányítják.

Ez a vezérlés, a **műveleti vezérlés** történhet

- közvetlenül, ún. „**huzalozott logikával**”, amikor is a dekódolás hatására, a vezérlő egység minden utasításhoz külön megvalósított bonyolult digitális áramköri rendszer segítségével állítja be a gép egyes részeinek állapotát,
- és történhet a vezérlés közvetett, **mikroprogramozott** módon is. A mikroprogramozott vezérlés, azt jelenti, hogy az utasítás műveleti kódja elindít egy kisméretű, elemi vezérlési lépéseket tartalmazó mikroprogramot és ennek segítségével vezérli a gép egyes részeinek állapotát. A mikroprogram a míkroprogramtárban található, amely csak-olvasható, ROM tároló. A külső vezérlő jelek egyrészt a processzor és a memória, illetve I/O eszközök közötti adatátvitelt irányítják.

A vezérlőegység működése szempontjából a két legfontosabb regiszter:

- PC = Program Counter, mely a soron következő utasítás tárolóbeli címét tartalmazza. (Ezt a regisztert Intel processzoroknál IP-nek nevezik, IP = Instruction Pointer.)
- IR = Instruction Register, mely a memóriából kiolvasott utasítást tárolja. Az ebben található műveleti kód alapján a vezérlőegység meghatározza az elvégzendő műveletet (dekódolás).

Regiszterkészlet

A regiszterek gyors írható-olvasható munkatárak. A különböző regiszterek szigorúan meghatározott feladatokhoz vannak hozzárendelve, emiatt korlátozott funkció betöltésére alkalmasak. A belső sínrendszeren keresztül tartanak kapcsolatot a processzor más részeivel. Ezek a számítógép leggyorsabb működésű tárai, amelyek hossza általában az adatsín szélességével egyezik meg. A regiszterkészlet processzorfüggő. A regiszterek egy része a felhasználó által közvetlenül hozzáférhető, míg egy másik része a felhasználó által közvetlenül hozzá nem férhető (csak a processzor használja). A regiszterek a processzor legkülönbözőbb részeiben (ALU, CU) vagy önállóan tömbökbe szervezetten fordulhatnak elő.

A processzor regiszterei a felhasználói programok szempontjából három kategóriába sorolhatók:

- *Rendszerregiszterek,* melyek a felhasználói programok számára nem „láthatók”, nem elérhetők. Erre példa az IR utasítás regiszter.
- *Speciális célú regiszterek*, melyek a felhasználói programokban csak meghatározott utasításokban szerepelhetnek. Erre példa a flag vagy státuszregiszter.
- *Általános célú regiszterek*, melyeket a felhasználói programok utasításaiban korlátozás nélkül használhatók. Erre példa az akkumulátor regiszter.

Regiszterek fajtái:

- **Az akkumulátor regiszter** (AC) az aritmetikai és logikai műveletek operandusait, vagyis a műveletek tárgyát képező mennyiségeket, vagy azoknak az eredményeit tárolja. A közbenső, részeredmények tárolására is alkalmas és minden műveletben részt vesz. A korszerű számítógépekben az akkumulátor helyett már egy vagy több regisztertömb van, amelyben akár 512 regiszter is elhelyezkedhet. Így csökkenthető a tárhoz-fordulások száma, illetve növelhető a végrehajtás sebessége.
- **Utasításregiszter** (IR): A vezérlő egységhez tartozó regiszter, amelyben a memóriából kiolvasott utasítás tárolódik, amíg a CU az utasítás műveleti jelrésze alapján meghatározza az elvégzendő műveletet és elindítja a vezérlő egységen keresztül a műveletet
- **Utasításszámláló regiszter** (PC v. IP): A soron következő utasítás címét tárolja. Az utasításszámláló tartalmát a program maga is változtathatja, a problémamegoldás az utasításkódok címeinek sorrendjében megy végbe. Vagyis az utasításszámláló által címzett első memóriarekesz elérésekor kiolvasunk a memóriából egy utasításkódot, így az utasításszámláló tartalma egy utasításhossznak megfelelően nő, és így a memória azon rekeszét címezi, ahol a program szerint a következő utasításkód található.
- **Bázis(cím)regiszter** (BR): Az operandusok címzéséhez felhasznált regiszter, amely nem általános használatú. A báziscím egy alapcím, amelyhez viszonyítva adhatjuk meg az utasításban az operandus helyét. Nem minden processzornál használják.
- **Indexregiszterek:** Szintén nem minden processzorban találhatók és ezek is az operandusok címzését segítik elő, különösen adatsorok feldolgozásánál.
- **Állapotregiszterek, vezérlő regiszterek:** Egy vagy több regiszteren belül tárolnak vezérlő és ellenőrző jeleket. Az állapotregiszter egyes bitjei az ALU művelet végrehajtásának eredménye alapján kapnak automatikusan értéket. Jellegzetes állapotbitek:
  - előjelbit - S (sign)
  - nulla bit - Z
  - túlcsordulásbit - O (overflow)
  - átvitelbit - C (carry)
  - félbyte - átvitelbit - H
  - megszakításbit - I
  - paritásbit - P
  - stb.

A **busz interfész egység** (BIU) biztosítja a processzor kapcsolódását a külső sínrendszerhez.

A **címszámító és védelmi egység** feladata a programutasításokban található címek leképezése a főtár fizikai címeire és a tároló-védelmi hibák felismerése.

A **belső sínrendszer** a CPU-n belüli adatforgalmat lebonyolító áramkörök összessége.

A **belső gyorsító tároló** (L1 cache) a főtárból kiolvasott utasítások, és adatok átmeneti tárolására szolgál.

A processzorok üzemmódjai és állapotai

A processzoroknak különböző üzemmódjai, állapotai vannak ezek megkülönböztetésére: védelmi szempontból, a programfolyamatok biztonságos (pl. rendszer-

összeomlás nélküli) működtetése miatt, az operációs rendszer és a felhasználói programok működtetésének elkülönítése miatt, valamint a korábbi processzorokkal való bináris program-kompatibilitás megtartása miatt van szükség.

Egyes processzoroknál megkülönböztetünk *supervisor és felhasználói* (user) állapotokat, melyek elkülönítik az operációs rendszerhez tartozó programokat a felhasználói programoktól. A *privilegizált supervisor* üzemmódban a processzor végrehajthat olyan utasításokat is, melyek felhasználói üzemmódban tilosak.

A Pentium processzoroknak négy üzemmódja van:

- Valós üzemmód: a processzor ekkor úgy működik, mint egy régi (PC-XT) 8086-os processzor. Erre azért volt szükség, hogy a felhasználók változatlan formában képesek legyenek „régi", megszokott DOS-os programjaik futtatására. Ekkor a processzor 8086-os emulációval működik, azaz a felhasználói program úgy látja, hogy a program 8086-os processzoron fut (például 32-bites regisztereknek csak az alsó 8 bitjét engedi használni, az utasításkészlet is ennek megfelelően csökkentett).
- Védett üzemmódban kihasználhatók a 32-bites architektúra összes lehetőségei, ezért a teljesítmény ekkor a legnagyobb. (Az üzemmód onnan kapta a nevét, hogy ebben az állapotban a tárolóvédelmi rendszer bekapcsolásra kerül.) Ez az üzemmód jellegét tekintve multitasking.
- Védett valós üzemmódban a processzor a 8086-os processzort csak egy taszkban emulálja. (Azaz például a DOS egy WINDOWS 95 alkalmazásként indul el egy elkülönült ablakban.)
- Rendszermenedzselő (SMM = System Management Mode) üzemmód független az operációs rendszertől és az alkalmazásoktól és egyben a processzor energiatakarékos működési módja.

2.2. A processzor utasításkészlete

Ahhoz, hogy teljesen megértsük a processzor működését, szükségünk van arra, hogy tudjuk, miféle utasítások végrehajtását várhatjuk el tőle. A részletezett utasítás-végrehajtás a **gépi kódú utasításokra** (machine code, machine instruction) vonatkozik, amelyek a processzor számára közvetlenül értelmezhetők. A gépi utasítások csak egyszerű műveletek, lépések előírására alkalmasak; mint pl. összeadás, kivonás, memóriahely kiolvasása, írása, stb.

Az utasítások azonban nem önmagukban álló teendők, hanem műveletek a hozzájuk tartozó adatokkal, így szükségünk van arra, hogy megvizsgáljuk, milyen módon határozhatjuk meg ezeket az adatokat. Az adatok meghatározásának módjait nevezzük **címzési módoknak** (addressing modes). A címzési módokat részletesebben a 3. fejezteben tárgyaljuk.

Egy adott típusú processzor által ismert (értelmezhető) műveletek alkalmasan kódolt csoportját nevezzük a processzor utasítás-készletének. Természetesen a különböző processzorok utasításkészlete általában eltér, de az is igaz, hogy az elvégzendő műveletek a legtöbb esetben azonosak.

A következőkben nem egy konkrét processzor utasítás-készletét mutatjuk be, hanem az utasítások szerkezetének és a processzor utasítások végrehajtásával kapcsolatos tevékenységeinek alapvető jellemzőit.

A processzor számára csak a **gépi kódú utasítások** (*machine code, machine instruction*) értelmezhetők közvetlenül. Ez azt jelenti, hogy mind az elvégezni kívánt műveletet, mind a műveletben érintett adatokat *bináris formában* kell megadni. A felhasználó számára azonban ez elég körülményes és nehezen áttekinthető kódrendszer, ezért az utasítás-készlet ismertetéskor az **assembly nyelvű** programozási módszer alapelemeire támaszkodunk.

Ebben a felírási módban az egyes gépi műveletek *bináris kódját rövid mozaikszavak* (*mnemonik*) helyettesítik, az adatok pedig vagy szimbolikus nevekkel, vagy 16-os számrendszerben felírt alakban jelennek meg. A gépi utasítások csak egyszerű műveletek, előírására alkalmasak; mint pl. összeadás, kivonás, memóriahely kiolvasása, írása, stb.

2-1. Táblázat: ***A gépi kód és az assembly utasítás-szerkezete***

gépi kód	1011 1000 0011 0100 0001 0010
(hexadecimális alak – csak a könnyebb áttekinthetőség végett írjuk fel!)	B8h 34h 12h
abssembly	ADD AX, 1234h

2.2.1. Az utasítás-szerkezet

Az egyes utasítások megadásához az elvégezni kívánt tevékenység mellett általában meg kell adni az utasítás végrehajtása közben felhasználni kívánt adato(ka)t: az **operandusokat** is. Ez azonban nem olyan egyszerű feladat, ha végiggondoljuk, hogy egy számítógépes rendszerben a program számára szükséges adat hányféle módon állhat rendelkezésre:

- saját értékével (közvetlenül),
- eltárolva egy regiszterben vagy a memóriában,
- esetleg valamilyen külső forrásból (perifériától) származik.

Az utasításban szereplő adatok helyének meghatározásával a címzési módok foglalkoznak (a 3. fejezetben részletesen tárgyaljuk a legfontosabbakat), itt csak felsoroljuk a legfontosabb lehetőségeket:

- **követlen adatcímzés**: az operandus a műveletben szereplő értéket
- **regisztercímzés**: az operandusban annak a regiszternek a kódja szerepel, amelyben a műveletben szereplő érték megtalálható
- **közvetlen memóriacímzés**: az operandusban szereplő érték annak a memória-rekesznek a címe, amely a kérdéses adatot tartalmazza
- **indirekt címzések**: az operandusban szereplő adat egy olyan hivatkozás, amely – esetleg valamilyen további művelet elvégzése után – a keresett adat címének meghatározását teszi lehetővé.

Ezek szerint egy processzor-utasítás általános szerkezete a következő:

utasításkód – címzési mód – operandus(ok).

A gépi kódban mindhárom szerkezeti elemet bináris számok azonosítják, ehhez a nem számszerű értékek esetén kódolásra van szükség. Rendeljünk hozzá minden utasításhoz, minden címzésmódhoz és minden operandusként megengedett, de önálló címmel vagy értékkel nem rendelkező elemhez egy-egy sorszámot. (tipikusan ilyenek pl. a processzor regiszterei vagy a perifériák elérésére szolgáló portok). Pl. legyen az összeadás műveletének a sorszáma 00h, a kivonásé 01h, stb.; legyen a közvetlen adatcímzés kódja a 00b, a direkt memóriacímzésé a 01b, stb.; és végül jelölje a 00h az AX regisztert, a 01h az AX legmagasabb helyiértékű 8 bitjét (AH), 10h pedig az AX legalacsonyabb helyiértékű 8 bitjét (AL), stb.

A fentiek alapján könnyű belátni, hogy az utasítás-készlet megtervezése során figyelembe kell venni:

- a címzésre és az adatok átvitelére használható bitek számát (azaz a buszrendszert)

- és a megkülönböztetni kívánt utasítások, címzésmódok számát.

Ez a két tényező kölcsönösen meghatározza egymást, együttesen pedig meghatározza az utasítás-szerkezetet. Tekintettel arra, hogy a számítógép működését tekintve bájt-szervezésű, az utasítások hossza minden esetben egész számú bájt. Ez azonban nem jelenti azt, hogy az egyes bájtok önmagukban megfeleltethetők az egyes utasítás-elemeknek! Pl. a címzésmódokról látni fogjuk, hogy 4-8 alapeset megkülönböztetése általában elegendő, így ha az összes utasítások száma nem haladja meg a 64-et, akkor ez a két információ az utasításkód egyetlen bájtjában ábrázolható: a felső 5 bit az utasítást, az alsó 3 bit a címzésmódot kódolja.

További egyszerűsítésre ad lehetőséget az utasítások működésének vizsgálata. Egy általános műveletben 2 kiinduló adatból a művelet elvégzése után kapunk egy eredményt: 3+2=5. Ha megnézzük ennek az utasításnak a processzor által értelmezhető felírási módját, akkor az utasításkódban a következő adatoknak kell szerepelniük:

- milyen műveletet akarunk elvégezni
- hol van az egyik operandus (címzésmóddal)
- hol van a másik operandus (címzésmóddal)
- hová kerüljön az eredmény (címzésmóddal).

Ha a legegyszerűbb, 8 bites architektúrában gondolkodunk, és egyéb megszorításokat nem alkalmazunk, ez akkor is legalább 4 bájt. Ha azonban bizonyos (ésszerű) megszorításokat vezetünk be, ez csökkenthető. Ilyen megszorítás lehet, hogy egy műveletben csak egyfajta címzésmódot engedélyezünk, és azt az utasításkóddal együtt 1 bájton tároljuk; vagy hogy az eredmény mindig az első operandus helyére (annak korábbi értékét felülírva) kerül tárolásra. A gyakorlatban ilyen és hasonló megszorításokkal biztosítható, hogy az utasításszerkezet rövid utasításkódokból álljon, ami nagymértékben hozzájárul a gyors működéshez (pl. azáltal, hogy az egyes műveletek elvégzése során kevesebbszer kell a memóriához fordulnia a processzornak).

2.2.2. Utasítás-típusok

Az utasítások általános szerkezetének ismeretében már képet alkothatunk a processzor működéses során lezajló folyamatokról, azonban a pontos megértéshez szükséges a vizsgálni (vagy még inkább programozni) kívánt processzor utasítás-készletének az ismerete. Ennek teljes körű bemutatására jelen jegyzetben már csak terjedelmi okokból sem vállalkozhatunk (ez minden processzor-típus esetén egy akár több száz oldalas technikai leírás, ami általában a gyártó terméktámogatási szolgáltatásán keresztül érhető el), de a módszer megértéséhez szükséges alapismereteket a legáltalánosabban továbbfejlesztett utasításkészlet, az Intel 8086-os processzorának utasításai alapján bemutatjuk.

Az utasítás-szerkezet csoportosítása számos szempont szerint történhet.

Az egyik lehetőség az **utasításhossz** alapján történő kategorizálás. Ebben a csoportosításban az utasítások a méretük szerint jelennek meg, ez azonban megtévesztő lehet, ha figyelembe vesszük, hogy ugyanazon utasításhoz (különböző címzésmódok esetén) más és más műveleti kód és ennek megfelelően eltérő utasításhossz is tartozhat.

A másik lehetőség az **operandusok száma** szerint történő csoportosítás, ami lényegesen egzaktabb – de még mindig nem eléggé egyértelmű (léteznek olyan parancsok, amelyek különböző számú operandussal is értelmezhetőek). A csoportosítás alapja az, hogy a műveleti kódot hány operandus követheti: *nulla (!) egy vagy kettő*.

- **0 operandusú utasítások**: ez nem elírás, vannak olyan utasítások, amelyeknek nincs operandusuk, a legegyszerűbb ilyen a NOP utasítás, ami nem csinál semmit („üres utasítás”), de ide tartozik a HLT rendszervezérlő parancs (ami a processzor

működését felfüggeszti), illetve a státuszregiszter egyes bitjeit beállító parancsok (pl. CLC: átvitelbit törlése).

- **1 operandusú utasítások**: ezek közé az utasítások közé tipikusan azok a műveletek tartoznak, amelyek esetében vagy elegendő egyetlen információ a művelet elvégzéséhez (pl.a művelet eredendően egy-operandusú, ld. NOT: logikai tagadás, vagy a működéséből következik, hogy csak egyetlen adatra vonatkozhat: JMP: ugró utasítás – „egyszerre két helyre nem ugorhatunk…"), vagy alkalmazhatók az előző fejezetben felvázolt működési megszorítások (pl. MUL: szorzás, az akkumlátor-regiszert (AX) szorozza az operandus értékével, eredményt pedig az akkumulátorba tölti vissza).
- **2 operandusú utasítások**: a legáltalánosabb utasítások, a műveletben két operandus szerepel, az eredmény (általában) az első operandus helyére kerül. Ebben az esetben is fordulhatnak elő alapértelmezések illetve megszorítások, pl. szegmensregiszternek közvetlen adatcímzéssel érték nem adható (pl. MOV: adatmozgató parancs esetén a MOV AX, FFh érvényes (az AX regiszerbe betölti a 255 decimális értéket), a MOV DS, 8000h (a DS – az adatterület kezdőcímét tartalmazó szegmensregiszer – értéke legyen 32768) viszont érvénytelen!)

A harmadik lehetőség (jelen jegyzetben ezt alkalmazzuk) az elvégzett művelet jellege szerinti csoportosítás. Ez a kategórizálás sem teljesen pontos, hiszen a műveleti csoportok kialakítása és az egyes utasítások csoporthoz rendelése minden esetben szubjektív (bizonyos mértékig). A továbbiakban az utasítás-szerkezet legáltalánosabb (és ennek megfelelően remélhetőleg a legkönnyebben megérthető) utasításait ilyen csoportosításban adjuk meg.

Adatmozgató utasítások

MOV – adatok mozgatása, két operandusú, a második operandus által meghatározott értéket átviszi az első operandus által meghatározott helyre, a státuszregiszter bitjeit nem állítja.

XCHG – adatok cseréje, két operandusú (mindkét operandus csak regiszter lehet), a két operandusban tárolt értékpárt felcseréli

Számítási műveletek

Az ebbe a csoportba sorolt utasítások a számítógépen általánosan alkalmazott matematikai és logikai (alap)műveletek mellett tartalmaznak olyan (speciális) parancsokat is, amelyek a BCD kódban tárolt illetve a szöveges típusú adatok kezelésével foglalkoznak.

Az előbbi érdekessége az, hogy ha elfogadjuk (*ami egyébként általában igaz*), hogy a processzor (alapértelmezés szerint) fixpontos aritmetikát használ (a lebegőpontos műveletek elvégzéséért általában a segédprocesszor felelős – még akkor is, ha fizikailag a két eszköz ma már nem válik külön), akkor felmerül a kérdés, hogy az összeadás, a komplemensképzés és a léptetés műveletén kívül mi szükség van további műveletekre (hiszen ebből a háromból a további alapműveletek felírhatók)…

A típusok kezelésével kapcsolatban pedig az értelmezésre szeretnénk felhívni a figyelmet: a számítógép számára az „adattípus" fogalma nem létezik (minden memóriarekeszben egy 8 bites, kettes számrendszerben felírt érték szerepel – hogy ez egy számnak az értéke vagy egy karakternek a kódja, annak eldöntése a felhasználó feladata). Ebből viszont az következik, hogy a tipizált műveletek sem kellene, hogy a processzor utasítás-készletének a részét képezzék.

Mindkét látszólagos ellentmondás feloldása az optimalizálás. A számítógép lényegesen hatékonyabban képes működni, ha az alapműveletekkel ugyan felírható, de összetett műveleti sort eseményező, ugyanakkor gyakran használt tevékenységekhez is külön parancsot rendelünk. Ilyen módon ugyan az utasítás-szerkezet bővül, ami az utasítás

értelmezése fázisában többletfeladatot ró a processzorra, de (ezt követően) a végrehajtás is gyorsabbá válik.

Néhány jellemző parancs ebből a csoportból:

aritmetikai műveletek:

ADD – összeadás, két operandusú, a második operandus által meghatározott értéket hozzáadja az első operandus értékéhez, majd az eredménnyel felülírja az első operandust, a státuszregiszter bitjeit az eredménynek megfelelően állítja

SUB – kivonás, a művelet jellegétől eltekintve ugyanaz, mint az ADD.

CMP – összehasonlítás, két operandusú, úgy végez el egy kivonást, hogy az eredménnyel nem írja felül az első operandust (mindkét operandus megőrzi eredeti értékét), de a státuszregisztert az eredménynek megfelelően állítja.

MUL – szorzás, egy operandusú (!), az akkumlátor értékét megszorozza az operandus által megadott értékkel, és az eredményt az akkumlátorba írja. Érdekessége, hogy

1. nem kezeli az előjelet (a legmagasabb helyiértéken álló bit is az érték része),
2. az, hogy a műveletben az AX (16 bit) vagy az AL (8 bit) vesz részt, az operandus értéke határozza meg…

DIV – osztás, egy operandusú, a MUL fordítottja (működési szempontból is), szükség esetén a DX általános célú regisztert is felhasználja:

- ha az operandus 8 bites, akkor a művelet: AX/op, az eredmény: AL-ben, a maradék: AH-ban;
- ha az operandus 16 bites, akkor a művelet DX:AX/op, az eredmény: AX-ben, a maradék: DX-ben.

Logikai (bitenként értelmezett) műveletek:

AND (logikai ÉS), OR (logikai VAGY), XOR (logikai KIZÁRÓ VAGY) – két operandusú műveletek, az eredmény az első operandus helyén

NOT (logikai TAGADÁS, egyes komplemens képzés), NEG (kettes komplemes képzés): egy operandusú műveletek

TEST – bitek értékének tesztelése, két operandusú, gyakorlatilag a CMP logikai megfelelője (csak az állapotregiszer bitjeit állítja annak megfelelően, mintha a két operandusra vonatkozóan AND műveletet hajtott volna végre).

SHL (bitléptetés balra), SHR (bitléptetés jobbra), RCL (forgatás balra), RCR (forgatás jobbra) – bit léptető (mozgató) műveletek, egy/két operandusú műveletek, a művelet hatására az (első) operandus bitjei a második operandusban megadott mértékben (ha a második operandus hiányzik, akkor 1-gyel) elmozdulnak. A két művelet között az a különbség, hogy a „léptetés” során a megadott irányba eső utolsó bit átkerül a státuszregiszterbe, az ellenkező irányból pedig az elmozdult bitek helyére 0 kerül (ez megfelel a 2 hatványokkal való szorzás és osztás műveletének), míg forgatásnál az egyik oldalon „kilépő” bitek (esetlegesen a státuszregiszter bitjeinek értékét is figyelembe véve) a másik oldalon „belépnek” (x-2. táblázat)

2-2. Táblázat: Bitmozgató utasítások hatása

Regiszerek értéke (a művelet előtt)	**Utasítás**	**Regiszerek értéke (a művelet után)**
AX: 0100 1101 S[C]: 1	SHL AX	AX: 1001 1010 S[C]: 0

AX: 0100 1101	SHR AX	AX: 0010 0110
S[C]: 1		S[C]: 1
AX: 0100 1101	RCL AX	AX: 1001 1011
S[C]: 1		S[C]: 0

BCD aritmetika:

A BCD műveletek problémáját az okozza, hogy egy BCD kódban tárolt értékkel végzett művelet (a tárolás és a műveletvégzés eltérő számrendszere miatt) nem ad helyes eredményt (*emlékezzünk vissza a BCD számok ábrázolásáról és műveletiről tanultakra!*). A BCD utasítások az ilyen típusú adatok eredményének meghatározásához szükséges korrekciót végzik, mint pl . AAA (igazítás összeadás után), AAS (igazítás kivonás után), AAM (igazítás szorzás után), AAD (igazítás osztás előtt), stb.

szövegműveletek:

Alapértelmezés szerint **sztring** (*szöveg, karakterlánc*) alatt bájtok (újabban a több-bájtos kódrendszerek, pl. UNICODE megjelenése miatt inkább szavak – *itt a „szó" mint 2 bájtos hivatkozási egység szerepel!*) folytonos sorozatát értjük. Az előbbi eszmefuttatás miatt a memória bármely összefüggő részét tekinthetjük akár sztringnek is, így feldolgozható szöveges műveletekkel.

Ezeknek a parancsoknak közös jellemzője, hogy két operandusú parancsok, azonban az operandusaik „**ál-operandusok**", ami azt jelenti, hogy az utasításban közvetlen módon címeket nem adhatunk meg: a sztringműveletek minden esetben a *DS:SI* regiszter-párban található címet tekintik a kiinduló („forrás") sztring kezdőcímének, az *ES:DI* regiszter-párban található címet pedig „cél"-nak. További közös jellemzője a szövegműveleteknek, hogy a művelet elvégzése után mindegyik utasítás a sztring(ek) következő elemére *állítja az SI-t és/vagy a DI-t*. (A „következő elem" értelmezéséről az állapotregiszter D („Direction", irány) bitje rendelkezik: ha értéke 0, akkor növekvő (a következő „karakter"), ha értéke 1, akkor csökkenő (az előző „karakter") – kezelésére a CLD (0), vagy az STD (1) parancsok szolgálnak.)

MOVS – a „forrás"-szöveg átmozgatása a „cél"-szövegbe

CMPS – összehasonlítása, az állapotregiszert állítja

SCAS (keresés), LODS (betöltés), STOS (tárolás) – az akkumlátort felhasználó szövegműveletek, a SCAS az akkumlátorban tárolt értékkel összehasonlítja a „cél" (ES:DI) szöveg aktuális pozícióján levő értékkel, a LODS a „forrás" (DS:SI) aktuális értékét tölti AX-be, a STOS AX-et mozgatja a „cél" aktuális pozíciójába.

Összehasonlító és elágazás-kezelő utasítások

Az utasítás-végrehajtás (a Neumann-elvű számítógépek esetén) alapértelmezés szerint soros (szekvenciális) szerkezetű, ami azt jelenti, hogy az utasítások végrehajtása egymás után, a tárolás sorrendjében történik. Ezt a szerkezetet két olyan programozás-technikai eszközzel módosíthatjuk, amelyek az utasítások sorrendjét valamilyen módon megváltoztatják: a (feltételes) elágazások és a ciklusok alkalmazásával. Az **elágazás** olyan szegmense a programnak, amelyben a következő utasítás (pontosabban annak a címe) valamilyen logikai döntés eredményének függvényében alakul ki. Az ilyen döntések kezelésére szolgáló parancsok – mivel a processzor számára csak az állapot- (vagy státusz-) regiszter szolgáltat ilyen jellegű információt – a státuszregiszter kezelésére vonatkozó, valamint a közvetlen utasításcímző (ugró) utasítások.

A státuszregiszter bitjei az egyes műveletek eredményének megfelelően (általában) módosulnak: *beállításra* vagy *törlésre* kerülnek, az előbbi fogalommal a bit 1, az utóbbival a 0 értékét szokás jelezni. Ezen felül léteznek közvetlen értékadó utasítások is: az STx

utasítással az egyes bit értéke 1-re állítható (SET), a CLx utasítással pedig az adott bit nullázható (CLEAR, törlés) – az x mindkét esetben a státuszregiszter egy-egy bitjének szimbólikus azonosítója. Két megjegyzés: ezek az utasítások *0 operandusúak,* hiszen a művelet által célzott bit – explicit módon – az utasításkódban szerepel, de a státuszregiszter *nem minden bitje* kezelhető ilyen közvetlen módon.

Példák: CLC (átvitel-bit törlése), STC (átvitel-bit beállítása), CLI (megszakítás-bit törlése – megszakítások letiltása!), stb.

A tényleges elágazást a JMP (ugrás) utasítás különböző alakjai valósítják meg. A JMP önmagában egy feltétel nélküli ugró utasítás, az utána írt cím a következő utasítás helyének meghatározására szolgál, a státuszregisztert nem vizsgálja. Ennek az utasításnak a feltételes alakjai a Jx (x ugyanolyan értelemben azonosítja a státuszregiszter egy bitjét, mint fent) alakú utasítások, amelyek előbb megvizsgálják az x bit értékét, és ha az 1, akkor átadják a vezérlést az operandusban megadott címre. (Ezeknek a parancsoknak egyrészt általában több alakjuk is létezik (azaz ugyanazt a feltételt több különböző parancs is kifejezi), másrészt általában létezik egy JNx alakjuk is (tagadás), ami a kérdéses bit 0 értéke esetén adja át a vezérlést.)

Példák: JC (ugrás, ha az átvitel-bit értéke 1), JNZ (ugrás, ha a nulla-bit értéke 0), JCXZ (ugrás, ha a CX regiszter értéke 0), stb.

Ciklusvezérlés, eljárások

A **ciklusok** a programok olyan szerkezeti egységei, amelyek segítségével egy (vagy több) többször végrehajtandó utasítás ismételt felírása helyettesíthető. A korszerű programozási eszközökben általában több (működésének módjában) különböző ciklusszervezési utasítás is megtalálható, ezek azonban általában (alkalmas kódolással) egymásnak megfeleltethetőek.

A *ciklusszervezés* (nem éppen legelegánsabb) eszköze lehet az ugróutasítások (helyesen alkalmazott!) használata is, de az utasítás-készlet tartalmaz a ciklusok kezelésére szolgáló (az előbb említett megoldás használata helyett – programozási szempontból – inkább javasolt) utasítást is. A LOOP egy operandusú utasítás, a CX (általános célú) regiszert használja ciklusváltozóként (az ismétlődések számának tárolására), operandusa a ciklus kezdőutasításának címe. Végrehajtása során előbb csökkenti CX értékét (1-gyel), majd megvizsgálja, hogy ily módon CX értéke elérte-e a nullát. Ha igen, akkor a *LOOP utáni utasítással* folytatódik a végrehajtás (a ciklus befejeződött), ha nem, a *LOOP operandusában meghatározott címre* kerül a vezérlés (és a ciklusban szereplő utasítások ismét végrehajtódnak).

A fentiekből két dolog következik:

1. a ciklus legalább egyszer biztosan lefut (ha CX az ellenőrzés előtt 0 volt, akkor akár 65535-ször is…),
2. a LOOP utáni cím (a LOOP-hoz képest) negatív (egy, az utasítás-sorozatban korábban már végrehajtott utasítás címére, azaz „visszafelé” hivatkozik).

A LOOP-nak (hasonlóan a feltételes ugró utasításokhoz) létezik olyan változata is, amely a CX mellett más regiszrer(ek) értékét is megvizsgálja, pl. LOOPZ/LOOPNZ: az ismétlődésben az állapotregiszter Z bitjének értéke is szerepet játszik).

A ciklusok működéséhez némiképpen hasonlító, de nem utasítás-szintű, sokkal inkább program-szintű (vagy még inkább programozás-módszertani kérdés) ismétlési elemek az eljárások. Az **eljárás** olyan utasítás-csoport, ami egy program végrehajtása során több alkalommal (de szemben a ciklussal, nem feltétlenül egymás után) ismételten végrehajtható. (Általában az egyes végrehajtások a program különböző helyéről kerülnek *meghívásra* – meghívásnak nevezzük a vezérlésnek egy eljárásra történő átadását.) Az eljárások egy speciális csoportját alkotják a számítógép beépített rendszerszintű eljárásai, a **megszakítások**

is. A megszakítások részben a perifériákkal való kapcsolattartásban, részint a működésben bekövetkező állapot-változások kezelésében játszanak szerepet. A legfontosabb utasítások a következők:

CALL – eljárás „meghívása”, egy operandusú utasítás, hatására a vezérlés átkerül az operandus által meghatározott címre;

RET – visszatérés, 0 operandusú parancs, hatására a vezérlés a megfelelő CALL utáni utasítást tartalmazó címre kerül;

INT – (szoftver-) megszakítás kérése, egy operandusú utasítás, operandusa a hívott megszakítás azonosítója (sorszáma). A megszakítás-kezelés „érdekessége”, hogy bár az utasításban egy operandus szerepel, valójában az egyes megszakítások végrehajtásához szükséges adatokat is át kell adni a megszakításnak, erre általában (megszakításonként meghatározott) általános célú regiszerek szolgálnak – ugyanazon megszakítás-sorszám különböző regiszter-értékek mellett akár más és más műveletet is eredményezhet!

Példa: INT 21h hatása, ha

- AH=3Ch: állomány létrehozása
- AH=3Dh: állomány megnyitása
- AH=3Fh: olvasás az állományból
- stb.

IRET – visszatérés megszakításból.

Veremkezelés

Mind az eljárások, mind a megszakítások kezelése a processzor számára számos érdekes problémát vet fel, amelyek közül a két legfontosabb a paraméterek átadásának és a visszatérés utáni folytatásnak biztosításának a kérdése. Az előbbire több megoldás is elképzelhető, az utóbbit viszont egyértelműen egy veremnek nevezett adatszerkezettel lehet a legegyszerűbben (és leghatékonyabban) kezelni. A **verem** egy LIFO („Last in, first out”, „először az utolsót”) szerkezetű adatstruktúra, amely működése során egy-egy művelettel írható és olvasható, de a műveletei minden esetben csak a verem „legfelső” elemére vonatkozhatnak – ebben az esetben a „legfelső” elem alatt egy rendszerregiszter (*SP*, **veremmutató**) által címzett memóriacímet értünk. (*Megjegyezzük, hogy általában megkülönböztethetünk memória-vermet: a verem a memória egy összefüggő (és fenntartott, azaz más műveletre nem használható) területe; és regiszter-vermet (kaszkád verem): a vermet több, egymással összekapcsolt regiszter alkotja – a továbbiakban a memória-verem működésén keresztül ismertetjük a veremkezelés alapelveit.*)

A veremkezelés műveletei: a PUSH (írás a verembe) és a POP (olvasás a veremből) egy operandusú utasítások, operandusa az érintett adatot tartalmazó regiszer kódja; illetve a PUSHF és a POPF 0 operandusú parancsok az állapotregiszter mentésére és visszatöltésére.

A veremműveletek működése: a PUSH először csökkenti (vagy növeli, a verem szervezésének függvényében) az SP értékét, majd az SP által címzett rekeszbe beírja az operandus értékét. A POP először az SP értékének megfelelő címen található értéket átviszi az operandusba, majd növeli (csökkenti) az SP értékét. (Ha belegondolunk ebbe a működési modellbe, látható, hogy a kiolvasás hatására nem tűnik el az érték a veremből, csak éppen elérhetetlenné válik az SP módosulása miatt, a következő írási művelet pedig úgyis felül fogja írni…)

Mire jó a verem? Sok mindenre (regiszterek értékének átmeneti tárolására, a regiszterek számosságát meghaladó mennyiségű változó elhelyezésére, stb.), de ahogy az előzőekben már említettük, elsősorban a különböző eljárások kezelése során az adatok átadásának illetve megőrzésének a legfontosabb eszköze.

Példák

Végezetül néhány (gondolat-ébresztőnek és felvilágosítónak) szánt példa a fentiek magyarázatául. Az alábbiakban megadott szerkezetek egy képzeletben létező szimbólikus nyelv formalizált utasításai – nem biztos, hogy az Ön által használt számítógépen változtatás nélkül működnek, de nem is ez a céljuk. Sokkal inkább az, hogy képet adjon, szemléletessé tegye az utasítások szerkezetének és működésének azokat az alapelveit, amelyekkel egy processzor is szembesül a felhasználói programok végrehajtása során.

- alapok:

MOV AL, FFh – az akkumlátor alsó bájtjának az értéke 255 lesz

MOV AX, BX – a BX regiszerben szereplő (2 bájtos) érték átkerül AX-be

MOV AX, [BX] – a BX regiszterben található értéket mint memóriacímet értelmezve, az erről a címről kiolvasott (2 bájtos) adat átkerül AX-be

- egy regiszter törlése (feltöltés 0-val):

```
MOV AX, 00h
SUB BX, BX
XOR CX, CX
```

- szövegkiíratás (feltevések: a kiírni kívánt szöveg karakterei A800h-tól tárolódnak, a szöveg hossza 12 karakter, a képernyőkezelés megszakítása a 21h kódú):

```
MOV  AX, A800h        ' kezdőcím
MOV  SI, AX           ' indexregiszter
MOV  CX, 000Bh        '12, hossz
ide: MOV  DL, [SI]    'ciklus kezdete, DL-be a következő betű
     INT  21h         'kiíratás, tfh: INT 21h → 1 karakter a képernyőre
     INC  SI          'következő karakter címe
     LOOP ide         'ha még van, vissza
```

2.3. Utasítás végrehajtás

2.3.1. A processzor működése

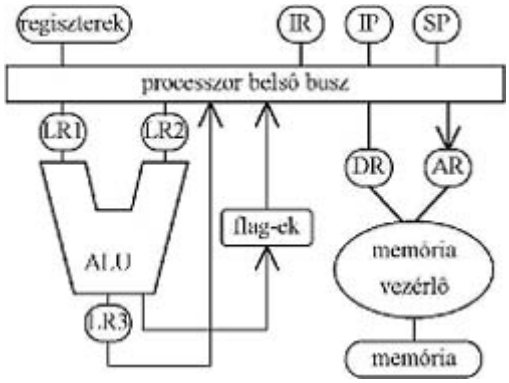
A processzor működését egy utasítás lépésenkénti végrehajtásával mutatjuk be.

1. Az utasítás beolvasása a processzorba. A processzornak először is szüksége van arra, hogy hol találja meg ezt az utasítást a memóriában. Az utasítás memóriabeli címe az IP (instruction pointer - utasítás mutató) regiszterben, vagy más elnevezésben PC (program counter) regiszterben található. Miután az itt tárolt cím alapján kiolvasta az utasítást a memóriából, az utasítást valahol el kell tárolni. Erre való az IR (instruction - utasítás) regiszter. 2.2 ábra.

2. A beolvasott utasítás dekódolása, elemzése. Beolvasás után az utasítást, értelmezni kell. Az utasítások mindig két részből állnak: az *utasítás kódjából*, amely meghatározza, hogy milyen műveletet kell végrehajtani, illetve az *operandusokból*, amelyek meghatározzák, hogy milyen adatokkal kell elvégezni az adott műveletet, valamint azt, hogy az eredmény hol tárolódjon.

A beolvasott utasítás az utasítás kódját minden estben tartalmazza, viszont ez nem minden esetben a teljes utasítás. Ennek oka az, hogy olvasáskor még nem tudjuk, hogy

milyen műveletet kell majd végezni, így arról sincs információnk, hogy az adott műveletnek hány és milyen operandusa van. Utóbbi tulajdonságok viszont benne vannak az utasítás kódjában. (Nyilván egy összeadás utasítás több operandust használ, mint egy regiszter értékét eggyel növelő.)



2-2. ábra: Utasítás végrehajtásban szereplő fontosabb részek

Az ábrán látható rövidítések: IR utasítás regiszter, LR1, LR2 és LR3 a művelet bemenő és kimenő adatainak tárolására szolgáló regiszterek, DR a memória adatregisztere, AR a memória címregisztere, SP stack pointer, IP utasítás mutató

3. Az operandusok beolvasása. Ebben a fázisban kiolvasásra kerülnek a memóriából az operandusok címei (ha ez szükséges) illetve maguk az operandusok. Ezek kétfélék lehetnek, vagy a memóriában vannak (és akkor onnan ki kell olvasni őket) vagy már előző műveletekkel regiszterekbe töltöttük azokat. Magyarázó 2.2 ábrán az ALU két segédregisztere szolgál arra, hogy a kiolvasott operandusokat tárolja (*LR1 és LR2, latch - segédregiszter*). Általában az ALU maximum kétoperandusú műveleteket képes végrehajtani, természetesen, ha speciális műveletvégző egységgel van dolgunk, akkor a maximális operandus számnak megfelelő segédregiszterre van szükségünk.

4. A művelet végrehajtása. Miután összeállt, hogy mit kell csinálni és az, hogy milyen adatokkal, az ALU elvégezheti az utasítást. Az eredményt - a tárolás céljára - egy harmadik segédregiszterbe teszi (LR3).

5. Az eredmény tárolása. Az utasítás tulajdonságának függvényében az eredmény az LR3 regiszterből vagy valamelyik regiszterbe vagy az utasításban meghatározott memóriacímre kerül (persze két lépésben, először a DR regiszterbe és csak utána a memóriába).

6. A következő utasítás címének meghatározása. Miután az utasítást elvégeztük, meg kell határozni az a címet, ahol a programvégrehajtás folytatódni fog. Ha a programunk szekvenciális, azaz a memóriában a következő utasítást kell végrehajtani, akkor az IP-t annyival kell megnövelni, amilyen hosszú az adott utasítás volt. Ha valahol egészen máshol kell a program végrehajtását folytatni, az általában olyan (a művelet eredményétől, mint feltételtől függő) utasítás, amelynek végrehajtása után valamely regiszter tartalmazza majd a folytatás memóriacímét. Így ezt a címet kell az IP-be beírni.

Ha az utasításciklus véget ért a processzor rátérhet a következő utasítás végrehajtására, azaz visszatérhet az 1. pontra.

A 2.2 ábrán található még a belső busz, a flag-ek, illetve az SP regiszter. A belső busz szolgál arra, hogy a processzoron belül adatok áramoljanak rajta az egyik helyről a másikra. A flag-ek (állapotjelzők) olyan speciális, 1 bites regiszterek, amelyek a processzor működését befolyásolják, illetve állapotát mutatják. Nevezetes flag például a carry flag, amely azt

mutatja, hogy az utolsó művelet (mondjuk összeadás vagy kivonás) alkalmával keletkezett-e átvitel/áthozat, vagy a zero-flag, amely akkor igaz, ha az utoljára végrehajtott művelet eredménye nulla volt.

2.3.2. A processzor számítási teljesítményét meghatározó tényezők

A számítógépek digitális áramkörei ciklusokban működnek, „ütemezve" valamilyen frekvenciával. Erre azért van szükség, mert a logikai értékeket reprezentáló feszültség szintek csak bizonyos idő után állnak be a bináris 0-nak vagy 1-nek megfelelő értékre. A ciklikus működés az órajel frekvencia megfelelően választott értékének használatával érhető el.

Egy számítógép teljesítményét az alapján tudjuk meghatározni, ha megmérjük az adott feladat elvégzésére fordított idejét.

Egy feladat végrehajtási idejét a következőképpen határozhatjuk meg:

$$\mathbf{F = C \bullet T \bullet U}$$

ahol F az egy feladat végrehajtásához szükséges idő, C az egy utasításra eső átlagos ciklusszám, T az egy ciklushoz szükséges idő, U a feladat végrehajtáshoz szükséges utasítás szám.

Egy számítógép teljesítményét növelhetjük tehát a feladat végrehajtáshoz szükséges utasítás szám (U) csökkentésével, a gép belső ütemezését meghatározó órajel frekvenciájának növelésével, azaz a periódusidő (T) csökkentésével, és a ciklus szám csökkentésével (mikro-utasítások számának csökkentése, párhuzamos utasítás végrehajtás bevezetése).

A számítógépek teljesítményének mérése

A számítógépek teljesítményének mérésére több népszerű mértékegység is létezik. A legegyszerűbben mérhető az időegység alatt végrehajtott utasítások átlagos száma:

- **MIPS** (Million Instructions per Second azaz a millió művelet/secundum, rövidítve). (Ezt egyes esetekben MOPS = Million Operation per Second mutatóval jelölik, melynek értelmezése azonos a MIPS-el.)
- **MFLOPS** (Millions of Floating Point Operations per Second) egy másik hasonló mértékegység, a másodpercenként végrehajtott lebegőpontos utasítások számát adja meg.
- A számítógép teljesítményéről összetettebb képet adnak a speciális teljesítménymérő programokkal előállított mutatók a „Benchmark"-ok. Ezek közül néhány ismertebb a Whetstone (mérnöki, tudományos programokat reprezentál), a Drystone (rendszerprogramozási környezetet reprezentál) és a SPEC Benchmark (egy alkalmazási célfüggő tesztprogram-készlet, mely egy elméleti alapgéphez viszonyít - pl. játékprogram, grafika, kvantummechanika).

E mutatók mindig átlagteljesítményt fejeznek ki, így például nyilván nem mindegy, hogy a végrehajtott utasításszámot milyen utasítások végrehajtásával állapítjuk meg, a leggyorsabb regiszter-regiszter műveletekkel, vagy a nagyságrendekkel lassúbb memóriaműveletekkel mérjük.

2.3.3. A CISC utasításkészletű gépek

A CISC (Complex Instruction Set Computer) komplex utatsításkészletű gépek kialakulásának fő oka volt, hogy a számítástechnika kezdeti szakaszában az operatív tár rendkívül drága erőforrás volt. Ez adta az alapját annak a gondolkodásmódnak, hogy a legjobb architektúra az, amely - a memóriabeli helyfoglalást tekintve - a legrövidebb programot eredményezi. A vezérlő egység felépítését nagymértékben egyszerűsítette a mikroprogramozás technikájának bevezetése, aminek a lényege az, hogy a végrehajtandó gépi

kódú utasítást több elemi lépésből un. mikroutasításokból (mikrokód) álló mikroprogram segítségével, több ütemben hajt végre a vezérlő egység.

Mivel egyre több funkciót hajtottak végre mikrokóddal, annak mérete növekedett. A mikroprogramok növekedésével szinte természetes igényként jelentkezett, hogy miért ne lehetne egy processzorhoz akár több utasításkészletet rendelni, így a mikroprogramot írható memóriába tették (WCM - Writable Control Memory). Ez lehetővé tette, hogy a mikrokódot a használt programnyelvhez vagy alkalmazáshoz lehessen adaptálni. Ezáltal egyszerűbbé váltak a fordítóprogramok is, hiszen egy magas szintű programnyelv elemet (statement) egyetlen gépi nyelvű utasítássá lehetett fordítani. Továbbá, amennyiben a gépi nyelv tartalmaz olyan utasításokat, amelyek közeli kapcsolatban állnak a magasszintű programnyelv elemeivel, akkor összezárul az a szemantikai rés (gap), amely a magasszintű programnyelv és a gépi nyelv között van.

A fentről lefelé történő kompatibilitás igénye arra kényszeríti a számítógépgyártókat, hogy állandóan növeljék az utasítás-készletüket az újabb és hatékonyabb modelljeikben.

Az utasításkészlet ilyen gazdagítása párhuzamosan folyt a címzési módok számának a hasonlóan gyors növekedésével. A sokfajta utasítás szükségesé tette, hogy változó hosszúságú utasítás-formátumokat használjanak. Így az utasítások műveleti kódja és operandus-specifikáló mezője változó hosszúságú lett. A különféle utasítás-hosszak azt eredményezhetik, hogy az utasítások tetszőleges bájtokon, sőt bizonyos architektúráknál tetszőleges biteken kezdődjenek, ami a későbbiekben a gyorsításnál jelentett akadályt.

A már korábban bevezetett utasításokat nem lehet elhagyni, hiszen ezeket alkalmazták a felhasználói programokban, és ez szoftver-inkompatibilitáshoz vezetne. Tehát az egyes processzorok utasításkészlete — a szoftver-kompatibilitási kényszer miatt — állandóan csak növekedhet.

A mikroprogramozott vezérlés esetében a makroszintű gépi utasítás műveleti jelrésze adja meg a mikroprogramtárban a műveleti vezérlést végrehajtó mikroprogram kezdőcímét, melynek utasításait elvégezve megtörténik a gépi utasítás végrehajtása. Ezt CISC processzorok esetében sokszor úgy is szokták magyarázni, hogy a mikroprogram végrehajtásakor tulajdonképpen „egy számítógép működik a számítógépen belül”. Ez azt jelenti, hogy a műveletvégrehajtás elemi lépéseit egy mikroprogram vezérlőegység (mint miniatűr Neumann elvű számítógép) irányítja a tárolt mikroprogram alapján.

A mikroprogramozott műveleti vezérléshez szükséges adatok, azaz a mikro-utasítások egy mikroprogram tárolóban helyezkednek el. Ehhez szükséges: egy mikroprogram számláló regiszter, mely mikroutasítás címét tartalmazza, mikroutasítások, melyekkel a következő mikroutasítás címe, és *a vezérlőjelek (kódolva vagy közvetlenül) előállíthatók.*

2.3.4. A RISC utasítás készletű gépek

A RISC (Reduced Instruction Set Computer)) csökkentett utasítás készletű gépek kialakulásnak körülményei: Az University of California, Berkeley (UCB) kutatócsoportja a 70-es évek végén azon dolgozott, hogy egyetlen chip-en kialakítson egy nagyteljesítményű processzort, ezért a tervezés során mindent igyekeztek egyszerűsíteni. Több éves program-elemzés eredményeképpen megállapították, hogy a CISC gépek utasításainak mintegy 25%-a használja a gépidő 95%-át (s ráadásul ezek pont a legegyszerűbb utasítások voltak). Ez impliciten magában hordozza, hogy a hardver által támogatott utasításoknak a 75%-át szinte egyáltalán nem használják. S a ritkán használatos utasítások a bonyolultabb utasítások közé tartoztak. Természetesen felmerült a logikus kérdés: Miért kell elpazarolni az értékes lapka-területet ritkán használt utasítások számára?

A modern számítógép CPU áramköreinek többsége az utasítás-dekódolást és a végrehajtásuk vezérlését végzi. A mikroprogramozott CISC számítógépeknél is az áramkörök

több mint a fele vezérlési célra szolgál. Az UCB kutatói észrevették, hogy a kis utasítás-készlet kisebb vezérlő-áramkör mennyiséget igényel, és a felszabaduló helyet más, a teljesítményt tovább növelő funkciók telepítésére lehet használni.

Kutatni kezdték az alkalmazási programokat, hogy mely utasításokat használják tipikusan, milyen gyakran kerülnek végrehajtásra és milyen CPU erőforrásokra van szükség azok támogatására. Ezek a tanulmányok kimutatták, hogy a **nagy regiszter-készlet** növeli a teljesítményt, továbbá, hogy bizonyos utasítás-osztályokat a nagyobb teljesítmény érdekében optimalizálni kell.

Tehát a gondolatmenet lényege:

- az utasításkészlet 75%-át elhagyták, s ezek ráadásul pont a bonyolultabb utasítások voltak
- ezáltal igen jelentősen (akár a tizedére is) csökkenthették a lapkán a vezérlőrész által igényelt területet
- az így felszabadult helyet regiszterekkel töltötték fel.

E munkák eredményeképpen született meg a RISC I típusú gép 1980-ban, mely valóban egyetlen chip-en egy nagyteljesítményű processzort jelentett.

A Stanford University kutatócsoportja John Hennessy vezetésével 1981-ben a MIPS nevű gép tervezésével kapcsolatosan a számítógép és a fordítóprogram (compiler) viszonyát kezdte vizsgálni. A kutatásuk témája az optimalizáló compiler tervezése és megvalósítása, az egyciklusos utasításkészlet.

A kutatások eredményeként új tervezési filozófia alakult ki. A csökkentett utasítás-készletű számítógép-tervezés olyan gépeket eredményezett, amelyek az azonos technológiával készített más számítógépeknél gyorsabban hajtották végre az utasításokat.

A tervezésnél érvényesített szempontok:

- Az **egyszerű utasítások** használatát alapvetően program-végrehajtási statisztikákkal igazolták. Ezeknek gyors a végrehajtásuk, tipikusan utasításonként egy ciklusidőt igényelnek.
- **Fix hosszúságú utasításokat** kell használni, ez az utasításlehívásban és a pipilene szervezésben is gyorsít.
- A **nagy számú utasítás-formátum hátrányos:** az utasítás dekódolás tovább tart, dekódoláshoz több hardvert igényel. A RISC utasítások csupán néhány különböző formátummal rendelkeznek, ami gyorsítja az utasítás-dekódolási műveletet.
- **Load-store architektúra alkalmazása** Ez azt jelenti, hogy a load és a store utasítás biztosítja az adatátvitelt a CPU regiszterek és az operatív tár között.
- **Kevés adattípus alkalmazásával csökkenthető** a műveletek száma.
- **Az egyszerű címzési módok** lehetővé teszik a gyors címszámítást

A CISC és RISC processzorok összehasonlító jellemzőit a következő táblázat foglalja össze:

2-3. Táblázat: CISC és RISC processzorok összehasonlítása

CISC processzorok	RISC processzorok
Összetett utasítások, melyek végrehajtása több gépi ciklust igényel.	Egyszerű utasítások, melyek végrehajtása 1 gépi ciklust igényel.
Bármely, erre alkalmas utasítás igénybe veheti a tárolót.	Csak a LOAD/STORE utasítások fordulhatnak a memóriához.
A futószalag (pipelining) feldolgozás kismértékű.	Erőteljes futószalag (pipelining) feldolgozás.
Változó hosszúságú utasítások.	Rögzített utasításhossz.
Sokféle utasítás és címzési mód.	Kevés utasítás és címzési mód.
Bonyolult mikroprogram, egyszerű fordítóprogram.	Bonyolult fordítóprogram, egyszerű mikroprogram.
Kis számú regiszter.	Nagy méretű regisztertár.
Mikroprogram által vezérelt utasítás végrehajtás (vertikális mikroprogramozás)	Huzalozott, vagy horizontális mikroprogramozás

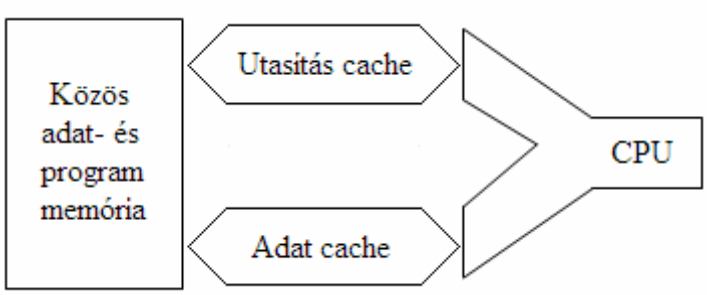
A RISC processzorokat eleve az operációs rendszerekhez és a compilerekhez igazítottan tervezik. Az egyszerű utasítások nemcsak egyforma hosszúságúak, hanem egyforma ciklusidőt igényelnek, ezért az ún. "pipe-line" feldolgozás könnyű. Egyetlen hátrány látszik: a bonyolultabb feladatokat utasítás-szekvenciákkal kell megoldani, ez a programok méretét, — hosszát — növelheti.

RISC gépek memóriája

A Neumann koncepciót követő, hagyományos felépítésű számítógépek ugyanazt a memóriát használják utasítások és az adatok számára. Nyilvánvaló, hogy az egy memória-ciklus alatt átvitt utasítások, és adatok mennyiségét az elérhető memória sávszélesség (a memória által időegység alatt írni/olvasni tudott adatmennyiség) korlátozza. Amennyiben ez a sávszélesség nem elegendő, akkor a processzor nem tud a maximális sebességével működni. A memória-sávszélesség növelésének egy lehetséges módja, hogy két különálló memóriát vezetünk be: egyet az utasítások, egyet, pedig az adatok számára. Ez két memória-elérési útvonalat eredményez, ami a párhuzamosság egyik formája. Az ilyen architektúrát Harvard architektúrának nevezzük. A Harvard architektúra alapkövetelménye, hogy a program ne változtassa meg önmagát. Ez nem túl nagy követelmény, mivel a program ön-módosítást rossz gyakorlatnak tartják.

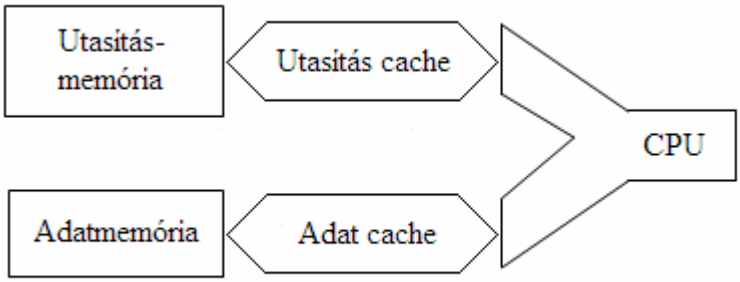
A legtöbb RISC implementáció Harvard architektúrát használ, hogy független elérési utat biztosítson az utasításokhoz és az adatokhoz, s ezáltal megduplázza a memória sávszélességet. Mivel a rövid ciklusidő és minden ciklusban egy utasítás végrehajtása a cél, az operatív memória nem tudja támogatni az igényelt sávszélességet, így cache memória használata a követelmény.

- **Harvard architektúra a CPU-cache interfésznél**: az operatív tárból az utasítások az utasítás cache-be, az adatok pedig az adat cache-be kerülnek. Így tehát két cache memória van, annak ellenére, hogy az egyetlen operatív tárban találhatók mind az utasítások, mind pedig az adatok;



2-3. ábra: Harvard architektúra közös adat-, és utasítás-memóriával

- **Harvard architektúra az operatív tárnál**: ekkor elkülönült utasítás- és adat-memória van.



2-4. ábra: Harvard architektúra külön adat-, és utasítás-memóriával

2.4. A párhuzamos architektúrák osztályozása

2.4.1. Osztályozás a feldolgozott utasítás- és adatfolyamok száma szerint

Utasításfolyam: Az utasítások egymás utáni sorozata, amelyeket egy program lefuttatása során hajt végre a számítógép. (Az utasításfolyam nem azonos a programutasítások sorozatával. Pl. ciklusok utasításai, a programban csak egyszer szerepelnek, az utasításfolyamban viszont annyiszor ahányszor azokat a számítógép végrehajtja.)

Adatfolyam: Az utasításfolyam utasításai mindig meghatározott adatokra hivatkoznak, amelyekkel a műveleteket el kell végezni. Ezek egymásutániságát, amilyen sorrendben rendelkezésre kell állniuk, nevezzük adatfolyamnak.

Értelmezett fogalmak:

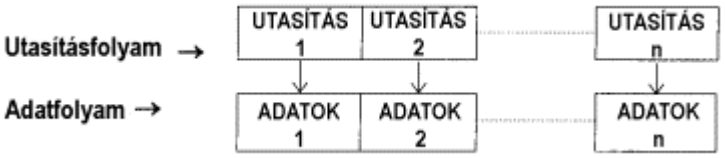
- **SI** – **S**ingle **I**nstruction Stream: egyetlen vezérlő egyetlen utasításfolyamot bocsát ki
- **MI** – **M**ultiple **I**nstruction Stream: a vezérlő több, egymástól elkülönülő folyamatot bocsát ki
- **SD** – Single Data Stream: A műveletvégző egyetlen adatfolyamot hajt végre, dolgoz fel.
- **MD** – **M**ultiple **D**ata Stream: A műveletvégzők több adatfolyamot dolgoznak fel.

A fenti fogalmak kapcsán értelmezett architektúrák:

2-5. ábra: Flynn által bevezetett architektúra osztályok

2.4.2. SISD architektúrájú számítógépek

A SISD (Single Instruction Stream Single Data Stream) típusú számítógép egyetlen utasításfolyammal egyetlen adatfolyamot dolgoz fel, 2.6 ábra.

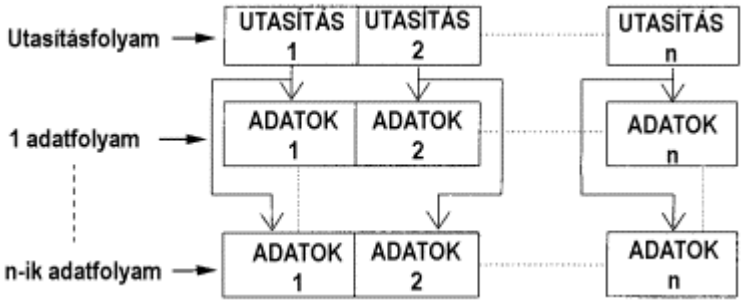


2-6. ábra: Single Instuction, Single Data Stream típusú utasítás végrehajtás

(Példák SISD architektúrájú gépekre: A Neumann elvű gépek , A PC-k központi feldolgozó egysége (CPU = Central Processing Unit), azaz processzora, a Pentium MMX-ig .)

2.4.3. SIMD architektúrájú számítógépek

A SIMD (Single Instruction Stream Multiple Data Stream) típusú számítógép egyetlen utasításfolyammal többszörös adatfolyamot dolgoz fel, amint az alábbi ábrán látható:

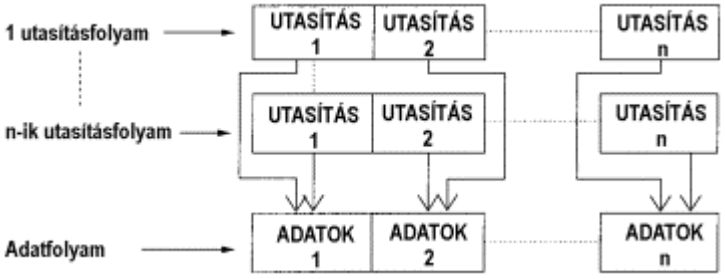


2-7. ábra: ***A SIMD típusú számítógép működése.***

Ezek a számítógépek a párhuzamos művelet végrehajtáshoz több párhuzamos, egyidejű működésre képes műveletvégző egységet (ALU) tartalmaznak. Vektorműveleteket képesek végrehajtani, gépi utasítás szinten (például a Pentium III processzor 3D grafikus utasításai). Két alaptípusuk van: a **közös tárolójú** (Shared Memory) gépek, és az **osztott tárolóval** (Distributed Memory) rendelkező gépek

2.4.4. MISD architektúrájú számítógépek

A MISD (Multiple Instruction Stream Single Data Stream) típusú számítógép több utasításfolyammal egyetlen adatfolyamot dolgoz fel, amint azt a következő ábra mutatja:

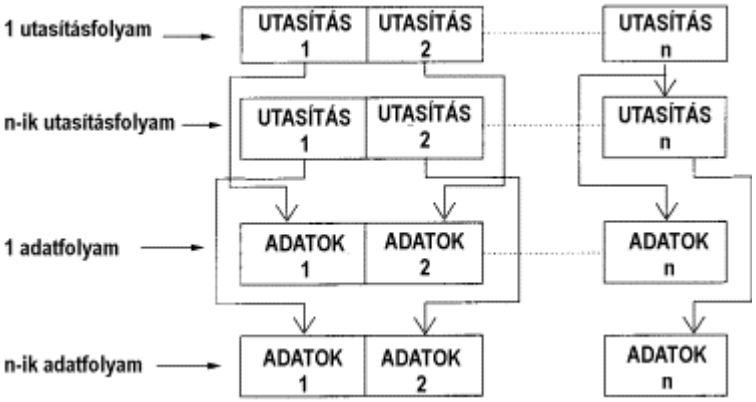


2-8. ábra: ***A MISD típusú számítógép működése.***

A gyakorlatban ilyen gépek nem léteznek. Egyes szakértők ide sorolják a pipeline (futószalag) szervezésű processzorokat, illetve a hibatűrő architektúrákat, melyek többszörösen végzik el a műveleteket azonos adatokon, és az eredményt hibavédelmi célból összehasonlítják.

2.4.5. MIMD architektúrájú számítógépek

A MIMD (Multiple Instruction Stream Multiple Data Stream) számítógéprendszerek több utasításfolyammal több adatfolyamot dolgoznak fel, amint az az alábbi ábrán látható:



2-9. ábra: ***A MIMD típusú számítógép működése.***

Ebbe a kategóriába tartoznak a multiprocesszoros számítógépek (több vezérlőegységgel) és a nem a hagyományos, soros utasítás-végrehajtás elvén működő számítógépek (adatvezérelt számítógépek, neurális hálók). A MIMD gépek memóriakezelése szintén lehet közös, vagy elosztott.

2.5. A számítógéprendszer architektúrák teljesítményének növelése

Az első számítógépek (első, második generáció) a Neumann elvek alapján működtek. Ennek jellemzői:

- számítógép működését tárolt program vezérli,
- a vezérlés control-flow (vezérlésáramlásos) rendszerű, ami azt jelenti, hogy a gép a program utasításait egymás után sorban hajtja végre. Ehhez a számítógép önálló részegysége a vezérlőegység (CU = Command Unit) egy utasításszámláló regisztert (PC = Program Counter) tartalmaz, melyben a következő utasítás tárolóbeli címét tárolja,
- a gép belső tárolójában a program utasításai és a végrehajtásukhoz szükséges adatok egyaránt megtalálhatók (közös utasítás és adattárolás, a program felülírhatja magát),

- az aritmetikai és logikai műveletek (programutasítások) végrehajtását önálló részegység (ALU = Arithmetic Logic Unit) végzi,
- az adatok és programok beolvasására és megjelenítésére önálló egységek (perifériák) szolgálnak.

A piac igénye a számítógépek teljesítményének folyamatos növelését kényszerítette ki, így a Neumann elvű architektúrák továbbfejlesztésére volt szükség. A teljesítménynövelésnek alapvetően két módszere van: nem strukturális és strukturális gyorsítás.

Nem strukturális gyorsítás, amelynek lehetőségei korlátozottak. Szóba jöhető megoldások:

- Az órajel frekvenciájának növelése (Ennek korlátot szab a processzoron belül a villamos jelek terjedési sebessége, valamint megnövelt frekvenciához tartozó megnövekvő disszipáció.)
- A program optimalizált fordítása (ennek különösen a RISC processzoroknál van jelentősége)

Strukturális gyorsítás, amelynek formái:

Párhuzamosítás a CPU-n belül

- vektorszámítógépek (utasítás szinten párhuzamos)
- pipeline feldoplgozás (utasítás szinten párhuzamos)
- szuperskalár processzorok (utasítás szinten párhuzamos)
- társprocesszorok (processzor szinten párhuzamos)
- multiprocesszoros architektúrák (több „hagyományos” CPU-val és ALU-val)
- nem hagyományos elven működő számítógépek (pl.: neurális hálók)

2.6. Párhuzamos architektúrák

A számítógép fejlesztők mindig arra törekszenek, hogy növeljék az általuk tervezett gépek teljesítményét. Az egyik módszer a chipek gyorsítására, hogy megnövelik a belső óra sebességét. Ám minden új technikának meg van a maga határa, amit a kor pillanatnyi szintje határoz meg. Az órajel frekvencia növelése a méretek csökkenésével együtt járhat, de csak bizonyos korlátig, így a legtöbb computer-fejlesztő a párhuzamosság felé törekszik (két vagy több dolog egyidejű elvégzése a cél.).

A párhuzamosságnak két fontos fajtájáról beszélhetünk: az **utasítás-szintű**, és a **processzorok szintjén** levő párhuzamosságról. Az előbbiben a gép azt hasznosítja, hogy a párhuzamosság miatt másodpercenként több utasítás adható ki. Az utóbbiban pedig összekapcsolt CPU-k dolgoznak együtt ugyanazon a problémán. Mindkét megközelítésnek vannak előnyei. Ebben a részben az utasítás-szintű párhuzamossággal foglalkozunk, a következőben pedig a processzor-szintűt tárgyaljuk.

2.6.1. Utasításszinten párhuzamos architektúrák

Az utasítás-szintű (ILP)-processzorok kialakulása és áttekintése:

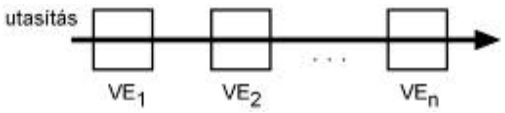
A Neumann-elvű processzorok teljesítményét egyrészt az órajelfrekvencia növelésével, másrészt a processzorok funkcionális fejlesztésével növelték. A funkcionális fejlődésre elsősorban a belső műveletek, azaz az utasítás-kibocsátás és az utasítás-végrehajtás párhuzamosságának a fokozása volt a jellemző. A funkcionális fejlődésnek három, egymást követő szakaszát különböztethetjük meg. A fejlődés kiindulópontja a hagyományos Neumann processzor volt, amelyet az utasítások soros kibocsátása és végrehajtása jellemez (lásd a következő ábrát).

2-4. Táblázat: ***Az ILP architektúrák fő fejlődési útja***

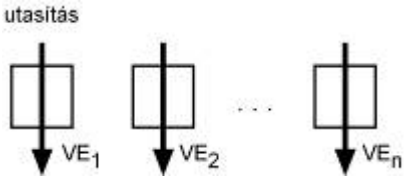
Neumann-féle szekvenciális architektúra 1950	Skalár ILP processzorok Futószalagos 1985	Szuperskalár ILP processzorok 1990	Szuperskalár processzorok Multimédia(MM)/3D kiegészítéssel 1994
Soros kibocsátás, soros végrehajtás	Soros kibocsátás, párhuzamos végrehajtás	Párhuzamos kibocsátás, párhuzamos végrehajtás	Párhuzamos kibocsátás, párhuzamos végrehajtás, utasításokon belüli párhuzamosság (SIMD)
Szekvenciális feldolgozás	Időbeli párhuzamosság	Időbeli párhuzamosság, kibocsátásbeli párhuzamosság	Időbeli párhuzamosság, kibocsátásbeli párhuzamosság, utasításon belüli párhuzamosság
Egyetlen processzorban egyetlen, **nem** futószalagos végrehajtó egység	**Futószalagos processzor**, több, nem futószalagos végrehajtó egységgel	Több futószalagos végrehajtó egységet tartalmazó hosszú utasítás szavú (VLIW) és **szuperskalár processzorok**	MM/3D kiegészítéssel rendelkező **szuperskalár processzorok**

A nagyobb teljesítmény iránti igény vezetett a párhuzamos utasítás-végrehajtás bevezetéséhez. Párhuzamos végrehajtás két, egymástól független elv alkalmazásával érhető el (2.10 és 2.11 ábra): vagy többszörözés révén, tehát több műveletvégző egység (VE) használatával, vagy a futószalagelv bevezetésével. E párhuzamosítási lehetőségek használatával a processzorok fejlődésének második szakaszában megjelentek az utasítás szinten párhuzamos processzorok (ILP Instruction Level Parallel processzorok). Mivel ezek a processzorok az utasításokat sorosan bocsátották ki, a fejlődés második szakaszát a skalár ILP-processzorok jellemzik.

A párhuzamos végrehajtás lehetőségei ILP-processzorokban



2-10. ábra: Futószalag (pipeline) időbeli párhuzamosság ILP processzorokban



2-11. ábra: Többszörözés (térbeli párhuzamosság) az ILP-processzorokban

A végrehajtó egység szakosodott, pl.: fixpontos / lebegőpontos műveletek végzésére.

A futószalag processzorok teljesítményét az ezen az úton elérhető teljesítménynövelés határáig a következő módszerekkel juttaták el:

- hatékony memória alrendszer (gyorsító tárak)alkalmazásával,
- hatékony ugrás előre jelzéssel.

Ezt követően a teljesítmény további fokozása érdekében a futószalagelvű műveletvégzőket többszörözték, és így a processzorok egyre nagyobb fokú párhuzamos végrehajtásra váltak alkalmassá. Természetesen az utasítások soros kibocsátása mellett több

párhuzamosan működő, futószalagelvű műveletvégző egyre kevésbé látható el kellő számú végrehajtható utasítással. Így a végrehajtás párhuzamosságának növelésével az utasítások soros kibocsátása lett a feldolgozás szűk keresztmetszete, ahogy azt FLYNN (1966) már jóval korábban előre látta. Ezért a Neumann-elvű processzorok fejlődésében elkerülhetetlenné vált egy harmadik szakasz is, amikor a soros utasítás-kibocsátást felváltotta a párhuzamos kibocsátás. Ennek eredményeként megjelentek a szuperskalár ILP-processzorok. Kezdetben ezek a processzorok statikusan ütemezett, többműveletes utasításokat kibocsátó VLIW-achitektúrákként jelentek meg. Később követték őket a bonyolultabb, dinamikusan ütemezett szuperskalár processzorok, amelyek minden ciklusban több utasítást tudnak kibocsátani.

Természetesen az utasítás-kibocsátás és az utasítás-végrehajtás szorosan összefüggnek egymással. Minél több utasítást (műveletet) tud egy processzor párhuzamosan végrehajtani, annál több utasítás (művelet) kibocsátására van szükség egy-egy óraciklusban a hardver erőforrások hatékony kihasználásához. Ezért a Neumann-elvű processzorok fejlődését összességében az utasítás-kibocsátás és -végrehajtás párhuzamosságának folyamatos és összehangolt növekedése jellemzi.

2.6.2. A pipeline működése

A rendszertechnikai gyorsítás egyik legfontosabb módszere az utasítás-végrehajtás szintjén átlapolt feldolgozás, melyet pipelineing-nek (futószalag, vagy csővonal) neveznek.

Gondoljunk ki egy analógiát, hogy a pipeline működésének magyarázatára. Képzeljük el egy gyár, késztermék csomagoló részlegét. Tegyük fel, hogy a részlegnek van egy hosszú futószalagja mellette sorban 5 munkással (feldolgozó egységek). Minden 10 másodpercben (a ciklus) az 1. munkás egy üres dobozt tesz a szalagra. Ezután a doboz a 2. munkáshoz kerül, aki belerak egy terméket. Egy kicsivel később a doboz megérkezik a 3. munkáshoz, aki lezárja. Utána folytatja útját a 4. munkáshoz, aki rak egy címkét a dobozra. Végül az 5. munkás leveszi a dobozt a szalagról és egy nagyobb tároló-egységbe rakja a későbbi kiszállítás céljából. Alapvetően így működik a számítógépnél a pipeline: mindegyik utasítás végig megy néhány feldolgozó lépésen, mielőtt végrehajtva kijut a végén.

Tegyük fel, hogy a gépi utasítás elemi lépései a következők:

Utasításkiolvasás (**F**etch), dekódolás (**D**ecode), operandus kiolvasás (**R**ead), végrehajtás (**E**xecute), visszaírás (**W**rite).

Soros feldolgozásnál minden elemi lépés egymást követően sorban hajtódik végre.

F	D	R	E	W	F	D	R	E	W
utasítás *n*					utasítás *n*+1				

2-12. ábra: ***Soros utasításvégrehajtás***

A futószalag (pipeline) feldolgozás lényege az, hogy azokat az elemi lépéseket, amelyek különböző hardver elemeket használnak, időben párhuzamosan hajtunk végre.

utasítás *n*	F	D	R	E	W		
utasítás *n*+1		F	D	R	E	W	
utasítás *n*+2			F	D	R	E	W

2-13. ábra: ***Pipeline utasításvégrehajtás***

Ha az utasítás-végrehajtás folyamatára visszaemlékszünk, akkor megállapíthatjuk, hogy a processzor különböző részegységei működnek az utasítás-végrehajtás egyes fázisai alatt. Így van lehetőség arra, hogy az elsőként lehívott utasítás dekódolása alatt a második lehívását elkezdjük, és így tovább…2.13 ábra.

2.6.3. Szuperskalár architektúrák

A Neumann-architektúrák teljesítményének növelésére a ’60-as évek második felében jelent meg a párhuzamos utasítás végrehajtás ill. pipeline. A párhuzamosan működő végrehajtó egységek “táplálására” azonban már nem volt elég a szekvenciális kibocsátás, ezért jöttek létre a szuperskalár rendszerek, amelyek 1 óraciklus alatt több utasítás kibocsátására is képesek. A szuperskalár kibocsátást a RISC processzorok körében alkalmazták először.

Nagyobb komplexitásuk miatt a szuperskalár CISC processzorok csak jóval később jelentek meg a piacon. Ennek oka az, hogy a változó utasításhossz és a CISC memória architektúra (nem egyszerű load/store) szuperskalár környezetben való megvalósítása nehezebb.

Ha egy pipeline jó, akkor kettő biztos jobb. Amennyiben két futószalagunk van (dual-pipeline) az utasítást lekérdező egység párokban kérdezi le az utasításokat, mindegyiket külön pipeline-on indítja el, és az ALU-val párhuzamos műveletként hajtja végre. Hogy a párhuzamos futtatás létrejöhessen, a két utasítás nem ütközhet az erőforrások kihasználásában (pl. nem használhatják ugyanazt a regisztert) és egyik sem függhet a másik eredményétől. Ezt a problémát a fordítóprogramnak kell feloldania.

Bár az egyszerű vagy dupla pipeline-okat legtöbbször RISC-gépeken használták (a 386-osnak és az elődeinek még nem volt egy se), a 486-tól kezdve az Intel processzorokban is megjelent. Az Intel 486-osának 1 pipeline-ja volt, és a Pentiumnak két 5 szakaszos.

Azt, hogy egy utasítás-pár kompatibilis-e, azaz végre lehet-e hajtani párhuzamosan őket, komplex előírások határozták meg. Pl. Pentium processzor esetében ha az utasítás nem volt elég egyszerű, vagy nem volt kompatibilis, a párból csak az első hajtódott végre (az U pipeline-ban). A másodikat várakoztatta, és a következő utasításhoz párosította. Az utasításokat mindig sorrendben hajtotta végre. Így a Pentiumra írt fordítóprogramok kompatibilis utasítás-párok alkotásával gyorsabban futó programokat tudtak produkálni, mint a régebbi fordítóprogramok. A mérések szerint egy Pentiumra optimalizált kód futási sebessége fixpontos adatokkal számoló programok esetében pontosan kétszerese volt, mint egy ugyanolyan órajelű 486-on. Ez a növekedés teljes egészében a két pipelinenak volt köszönhető.

2.6.4. A pipelining működés során fellépő problémák

A pipeline szervezésben azonban problémák is adódhatnak. Ezek a következők:

- az utasítások elemi fázisainak végrehajtásához szükséges idő igen eltérő lehet (például ha az operandus egy regiszterben található, akkor innen kb. tízszer gyorsabban lehívható, mint a főtárból),
- az utasítás soros végrehajtását a vezérlésátadó utasítások megzavarhatják, mivel ekkor nem a soron következő utasításokat kell betölteni a „futószalagra”,
- a megszakítások, kivételek kezelése is megszakíthatja a futószalag folyamatos feltöltését,
- az utasítás-végrehajtás során sokszor előfordul, hogy egy utasítás a megelőző utasítás eredményadatára hivatkozik. Így például, jelöljön R1, R2, R3 három regisztert és végezzék a sorban következő utasítások a következőt:

$$R1 + R2 \Rightarrow R1$$

$$R1 + 10 \Rightarrow R3$$

Nyilvánvaló, hogy ebben az esetben R1 értékét a 2. utasítás csak akkor használhatja fel, ha azt az 1. utasítás már előállította. Ezt az esetet **adatütközésnek**, nevezzük.

- hardver erőforrások igénybevétele során is előfordulhatnak ütközések, például buszkonfliktusok.

2.6.5. A pipelining során fellépő problémák kezelésének módszerei

Az előbb bemutatott problémákat vagy statikusan, a program fordítása során vagy dinamikusan, azaz futás közben a hardverrel tudjuk kezelni. Módszerek:

NOP utasítások beiktatása a programba (fordítóprogram által)

A memóriautasítások végrehajtásához szükséges többlet-időigény és a hazardok miatti utasításvárakoztatás „üres" (NOP = NO OPERATION) utasítások beiktatásával oldható meg (annyi „üres" utasítást kell beiktatni, amennyi biztosítja a pipeline szükséges ideig történő várakoztatását).

Data forwarding (hardver által)

Az „adat előreengedés" esetében, ha például két egymást követő utasítás számára azonos adat szükséges, akkor az ezek közötti adatátadást a processzoron belül megfelelő áramkörök biztosítják (adatátadás csak az első utasítás végrehajtási fázisa után lehetséges).

Utasítás-átrendezés (fordítóprogram által)

A fordítóprogram (ha ez lehetséges) a program tartalmi megváltoztatása nélkül átrendezi az utasítássorrendet és a (memóriautasítások és hazardok kezelése miatti várakozási időket hasznos utasításokkal tölti ki.

Scoreboarding (hardver által)

Minden regiszterre könyvelésre kerül, hogy azok az utasítások, melyek egy adott regiszterre hivatkoznak benne vannak-e a pipeline-ban. Ha egy további utasítás egy ilyen regiszterhez akar hozzáférni, akkor az késleltetésre kerül.

Harvard architektúra

Az utasításolvasás és az adatkiolvasás, visszaírás ütközéseire jelent megoldást, ha az utasításokat és adatokat fizikailag különálló memóriában tároljuk, amihez külön adat- és utasítássín is van (példa a Pentium processzorcsalád esetében az L1 szintű különálló adat és utasítás cache).

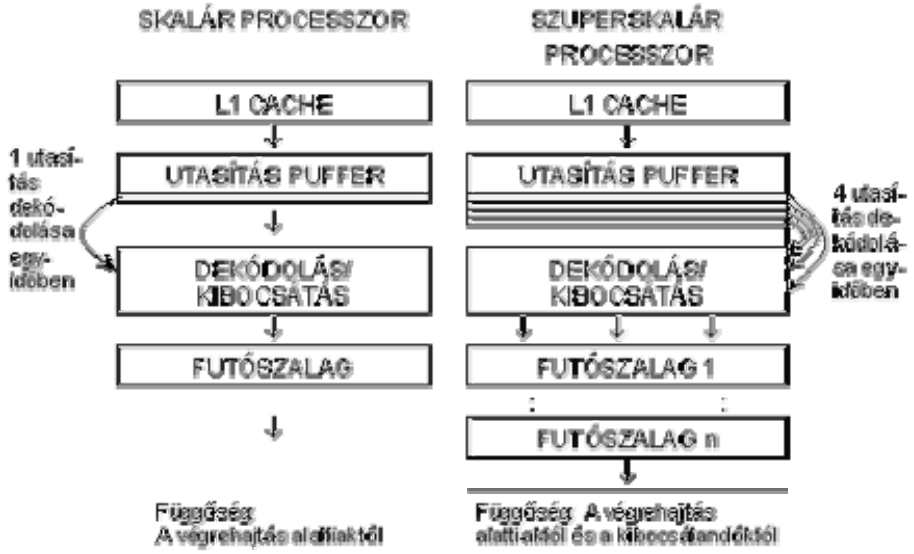
A **vezérlésátadó utasítások kezelése** a pipeline-ban kiemelt jelentőségű feladat (ilyen a gépi utasítások kb. 10–15%), mivel ebben az esetben – ha az elágazás bekövetkezik – más memóriacímtől kezdve kell tölteni a futószalagot.

A vezérlésátadás kezelésének módszerei közül a legegyszerűbb a pipeline leállítása, illetve törlése. Ez alatt azt értjük, hogy vezérlésátadó utasítás esetén

- a processzor leállítja a pipeline töltését mindaddig, míg az ugrás (vagy nem ugrás) kimenetele nem egyértelmű,
- a processzor mindig rögzített módon (például, hogy nem történik ugrás) becsüli meg az elágazás kimenetelét. Ha ez nem teljesül, akkor a pipeline-ben levő utasítássort törölni kell.

Az előbbieknél hatékonyabb megoldást jelent a vezérlésátadás kezelésére a korszerű **processzorok spekulatív elágazás feldolgozása**, melynél a processzor megpróbálja megjósolni – különösen a feltételes ugróutasítások esetében – a vezérlésátadó utasítás várható irányát, kimenetelét. Erre alapvetően két módszer van:

- statikus esetben a fordítóprogram értékeli ki az ugrási feltételeket, és meghatározza a legnagyobb valószínűséggel előforduló ugrási címeket, és ennek megfelelően szervezi a pipeline -t; azaz a feltételezett ugrási cím a dekódolási szakasz végére előállt, a pipiline-ba már erről a címről kerül be a következő utasítás. Ha az előrejelzés hibás volt, akkor eldobja a megkezdett utasítást, és a feltétel másik kimenetéhez tartozó címről folytatta. (pl.:486-os processzorok)
- dinamikus esetben a program futása közben a processzor egy táblázatban vezeti az ugróutasítások címeit és ezek kimenetét, és ezt felhasználva próbálja megjósolni az elágazások lehetséges kimenetét.



2-14. ábra: Egy skalár és egy négyszeres kibocsátású szuperskalár processzor működése.

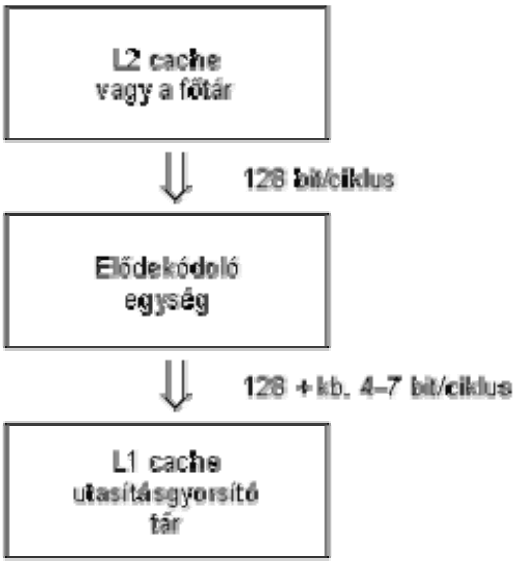
2.6.6. Párhuzamos dekódolás

A párhuzamosan működő végrehajtó egységekhez, melyek egy ciklus alatt több utasítást képesek végrehajtani, ilyen ütemben kell továbbítani a dekódolt utasításokat, ezért a szuperskalár működéshez párhuzamos utasításdekódolás is szükséges. A 2.14 ábra egy skalár és egy négyszeres kibocsátású szuperskalár processzor működését hasonlítja össze.

Az ábráról leolvashatók a különbségek:

- A skalár processzornak ciklusonként csak egy utasítást kell dekódolnia. Ennek során ellenőriznie kell, hogy a kibocsátandó utasítás függ-e (hazard) a végrehajtás alatt állóktól.
- A szuperskalár processzornak ciklusonként 4 utasítást kell dekódolnia. Ellenőriznie kell mind a 4 kibocsátásra váró utasításra, hogy:
  - függenek-e a végrehajtás alatt állóktól,
  - vannak-e közöttük függőségek.

A szuperskalár processzorok esetében a függőségvizsgálatok jóval nagyobb feladatot jelentenek, mint a skalár processzoroknál. Ez természetesen időt igényel, ezért a processzor dekódolási feladatait az elődekódolással próbálják meg csökkenteni. Ennek elvét a 2.15 ábra mutatja be.



2-15. ábra: Elődekódolás elve

2.6.7. VLIW processzorok

Az első VLIW(Very Large Instuction Word) processzorok a 70-es években jelentek meg és főként tudományos célra szánt számítógépekben alkalmazták őket. A VLIW architektúrák lényege, hogy az utasítások párhuzamosításáról nem a processzor, hanem a fordítóprogram gondoskodik, és a hardver gyakorlatilag egy teljesen párhuzamosított kódot kap meg végrehajtásra. Egy utasításban ilyen formán több mező található, amelyek külön végrehajtóegységeket vezérelnek. Innen ered a VLIW, vagyis "**nagyon hossszú utasításszó**" név is, ugyanis a VLIW processzorok utasításai a végrehajtóegységek számától függően akár 1024 bit hosszúak is lehetnek. A VLIW elnevezést egyébként a 80-as évek elején vezették be.

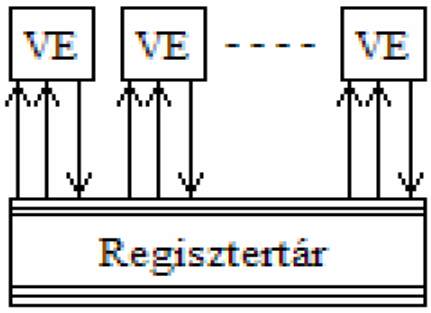
Az, hogy az ütemezést a fordítás közben végezzük el előnyökkel jár:

- A fordítóprogram több információval rendelkezik a futtatni kívánt szoftverről, mint a processzor,
- A függőségek kezelésére és párhuzamosítható utasítások kinyerésére gyakorlatilag végtelen mennyiségű idő és erőforrás áll a rendelkezésre.
- Gyártástechnológiai szempontból előny, hogy a processzorról lekerül a bonyolult ütemezési mechanizmus, így adott tranzisztorszám mellett több végrehajtóegység, vagy nagyobb cache memória kerülhet a chipre.
- Azonos komplexitás és órajel mellett egy VLIW processzor nagyobb teljesítménypotenciállal rendelkezik, mint egy szuperskalár architektúrájú.

A VLIW-ek hátránya, hogy a fordítónak rendkívül részletes ismeretekkel kell rendelkeznie a programot futtató processzorról, az egyes utasítások végrehajtásához szükséges időről, a végrehajtóegységek számáról és felépítéséről, hiszen csak így van lehetőség a hatékony ütemezésre. Könnyen belátható, hogy abban az esetben, ha a programot egy másik — például több végrehajtóegységgel rendelkező — processzoron szeretnénk futtatni, azt újra kell fordítani, hogy kihasználhassuk az új erőforrásokat. Bár úgy tünt, hogy a VLIW-ek a 90-as évek végén eltünnek el a számítástechnika színpadáról, azonban ma szinte reneszánszukat élik, számos beágyazott processzor és DSP (Digital Signal Processor) mellett általános célú processzorok is alkalmazzák a VLIW filozófiát.

A VLIW- és a szuperskalár processzorok felépítésüket tekintve abban különböznek az egyszerű futószalag-processzoroktól, hogy az utasítások párhuzamos végrehajtását alapvetően többszörözéssel, azaz több végrehajtó egység használatával érik el. Mindkét említett

processzorosztály esetén több, egymással párhuzamosan működő végrehajtó egység dolgozza fel az utasításokat. 2.16 ábra



2-16. ábra: A hagyományos VLIW- és a szuperskalár processzorok közös alapstruktúrája

A VLIW- és szuperskalár processzorok a párhuzamosítás mindkét lehetséges dimenzióját az időbeli átlapolást és többszörözést kiaknázzák. A párhuzamos végrehajtást alapvetően többszörözéssel érik el, a párhuzamos végrehajtás mértékének növelése érdekében, mindkét processzor, jellemzően futószalag elvű végrehajtó egységeket használ. Így valójában. VLIW- vagy a szuperskalár processzorokban a párhuzamosan végrehajtható utasítások maximális száma a párhuzamosan működő végrehajtó egységekben, időben átlapoltan feldolgozható utasítások számának összegével egyenlő.

A hasonlóságok ellenére, a VLIW- és a szuperskalár processzorok között alapvető különbségek vannak, különösen utasításaik felépítésében és abban, ahogy a függőségeket kezelik.

A szuperskalár processzorok hagyományos, soros utasításfolyamot dolgoznak fel ezzel szemben a VLIW architekturákat több-műveletes utasítások vezérlik, olyanok, amelyek több mezőből állnak, és az egyes mezők külön-külön vezérlik a rendelkezésre álló végrehajtó egységeket.

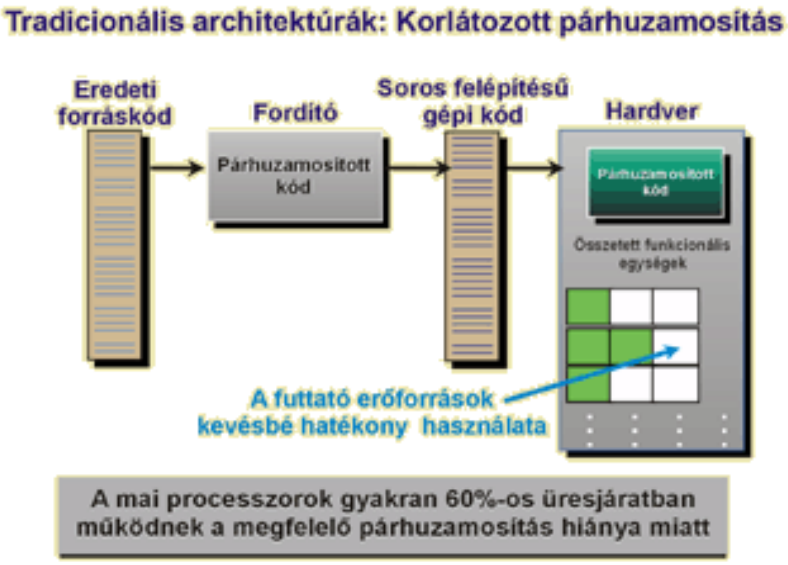
Igy a VLIW-processzorok szükségszerűen hosszú vagy nagyon hosszú utasításokat igényelnek ahhoz, hogy minden végrehajtó egység számára meg tudják adni az elvégzendő műveleteket. Utasításaik hossza többszöröse lehet a hagyományos RISC-utasításokénak. A 2...4 végrehajtó egységből álló ún. keskeny VLIW-ek utasításhossza 100 bit körüli, a tíz...húsz VE-t is magukba foglaló ún. széles VLIW processzorok utasításszavai több száz bit hosszúságúak is lehetnek.

Jelentős különbség a VLIW- és a szuperskalár processzorok között az is, hogy a VLIW processzorok függőségektől mentes (többműveletes) utasításokat várnak el, míg a skalár processzorok maguk birkóznak meg a függőségekkel. Másként fogalmazva, a VLIW-ek esetében a függőségek kezelése statikus, míg a szuperskalár processzorokban alapvetően dinamikus. Ennek következtében egyrészt, ugyanolyan fokú párhuzamónság mellett a szuperskalár processzorok szükségszerűen sokkal bonyolultabbak, mint a VLIW processzorok. Ennek alapján érthető, hogy a VLIW-processzorok miért jelentek meg közel egy évtizeddel hamarabb mint a szuperskalár processzorok.

2.6.8. EPIC processzor.

A hagyományos architektúrák gyorsításának egyik legfőbb akadálya, hogy a processzor mintegy hatvan százalékban üresjáratban dolgozik, mert nem kap párhuzamosan feldolgozható utasításokat (2.17 ábra).

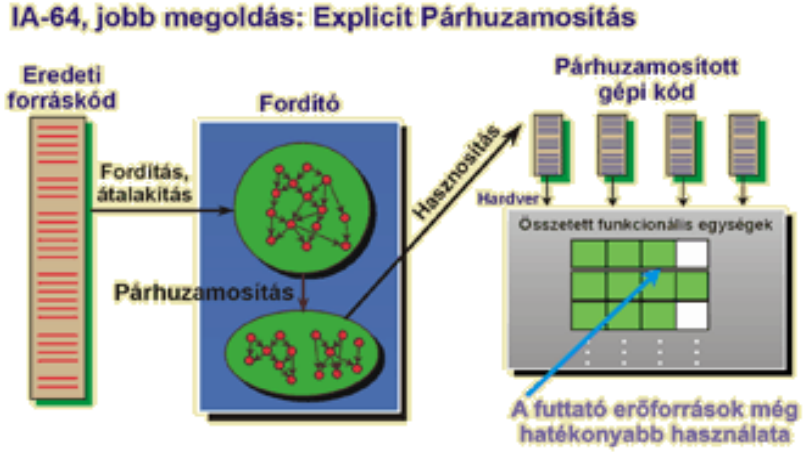
Az **EPIC** hasonlóan a RISC, CISC vagy VLIW fogalmakhoz egy tervezési filozófia, amelynek lényege, hogy a létező legtöbb utasítás fusson párhuzamosan, még akkor is, ha azok egy része feleslegesen hajtódik végre..



Forrás: www.Intel.hu

2-17. ábra: ***A tradicionális architektúrák erőforrás kihasználása***

Ezért például az IA-64 Itanium elágazáskezelésénél, a processzorok az erőforrások leghatékonyabb kihasználása érdekében elágazó utasítás esetén becslést alkalmaznak. Azaz megpróbálják megbecsülni valamilyen szabály alapján, például az addigi elágazások viselkedése szerint, milyen ágon fut majd a program, és abba az irányba folytatják a végrehajtást. (2.18ábra).



Forrás: www.Intel.hu

2-18. ábra: ***Explicit párhuzamosítás az IA64-ben***

A kifinomult elágazásbecslési algoritmusok ma már akár 92-97 százalékos becslési pontosságot is lehetővé tesznek, azonban természetesen még így is előfordul téves becslés. Ilyenkor a futószalagokat ki kell üríteni és a program végrehajtását a másik ágon kell folytatni. Ez nyilvánvaló időveszteség. Az Itanium ezt a veszteséget úgy hidalja át, hogy az elágazó utasítás után mindkét feltételes ágat elkezdi végrehajtani, majd utólag, a feltétel kiértékelésekor dönti el, hogy melyik volt a "helyes út". Mindkét ág kap egy megkülönböztető jelzést, és a feltétel kiértékelődése után a jelzés alapján a processzor eldönti, melyik ág

futtatását kell tovább folytatni és a feleslegesen végrehajtott utasítások eredményei pedig törlődnek. Ezt a megvalósítást az Intel "predication"-nek nevezi.

Korunk processzoraiban egyre nagyobb problémát jelent a memória késleltetése, vagyis az az idő, amennyi egy adatbetöltő utasítás (LOAD) és a kért adat megjelenése között eltelik. Az is nyilvánvaló, hogy minél több erőforrással rendelkezik egy processzor, annál több megy kárba az alatt az idő alatt, amíg a CPU a memóriára vár, hiszen ezt az időt hasznosan, számolással is tölthetné. Mivel az Itanium utasítás-ütemezése fordítási időben történik, lehetőség van ún. spekulatív adatbetöltésre, vagyis az utasítások olyan átrendezésére, hogy az adatbetöltő utasítás és az adat feldolgozása (LOAD-USE) közé más, független utasítások kerüljenek.

2.7. Adatpárhuzamos architektúrák

2.7.1. Vektorprocesszorok

Az alkalmazási területek egy részén, a matematikai-tudományos számítások körében, gyakran kell számsorozatokkal(vektorokkal, mátrixokkal) műveleteket végezni. Ezekre, a műveletekre jellemző, hogy ugyanazt a műveletet kell elvégezni sok adaton egymás után. Az adatsoron történő műveletvégzés lehetőséget ad azok átlapolt (pipelining) végrehajtására, és ezzel a teljesítmény növelésére.

A folyamatok kezelése oldaláról, ez tulajdonképpen az „egy utasításfolyam-több adatfolyam” (SIMD) besorolásnak felel meg.

A teljesítmény növelését egyrészről az segíti elő, hogy ugyanazt a művelet (utasítást) kell végrehajtani az egymást követő vektorelemeken és ehhez elegendő az utasítást csak egyszer lehívni a tárolóból; másrészről az is segíti, hogy a vektor egymást követő elemeinek feldolgozása átlapolható és így egy adat-pipeline alakítható ki. Példaként tekintsük két 3 elemű vektor összeadását:

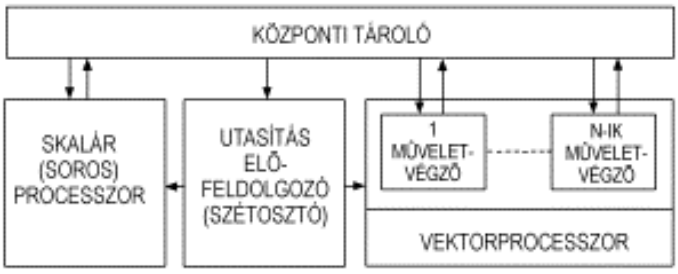
$$a = (a1, a2, a3)$$

$$b = (b1, b2, b3)$$

$$a + b = (a1+b1, a2+b2, a3+b3)$$

Soros utasításvégrehajtásnál (mivel csak egy ALU van) az összeg vektor csak 3 lépésben számítható ki, a SIMD architektúrájú vektorszámítógép viszont a vektor összeadását egy lépésben hajtja végre. Ehhez több (példánkban 3 db) aritmetikai egység szükséges, ezek mindegyike viszont ugyanazt az utasítást, azaz az összeadást hajtja végre.

Az előbbieknek megfelelően a vektorszámítógépek elvi felépítése 2.19 ábrán látható



2-19. ábra: Vektorszámítógépek felépítése

Láthatjuk, hogy a több műveletvégző egységet tartalmazó vektorprocesszor mellett a gépben egy skalárprocesszor is megtalálható, mely a nem vektoros műveleteket végzi. A vektorprocesszorok teljesítőképességét ugyanis erősen rontja az, ha skalár mennyiségekkel kell dolgoznia, mert áthaladásuk a feldolgozó láncon időveszteséget okoz. Ezért a vektorprocesszor mellett, a skalár mennyiségekkel való műveletvégzéshez külön processzor

áll rendelkezésre. (Ez a skaláris műveleteket a vektorprocesszornál hatékonyabban dolgozza fel.) A vektor-számítógépes legismertebb képviselői a CRAY gépcsalád tagjai, de az MMX mikroprocesszorok is a vektorszámítógépek működési elvének megfelelően hajtják végre a multimédiás gépi utasításokat.

2.7.2. Tömbprocesszoros számítógépek:

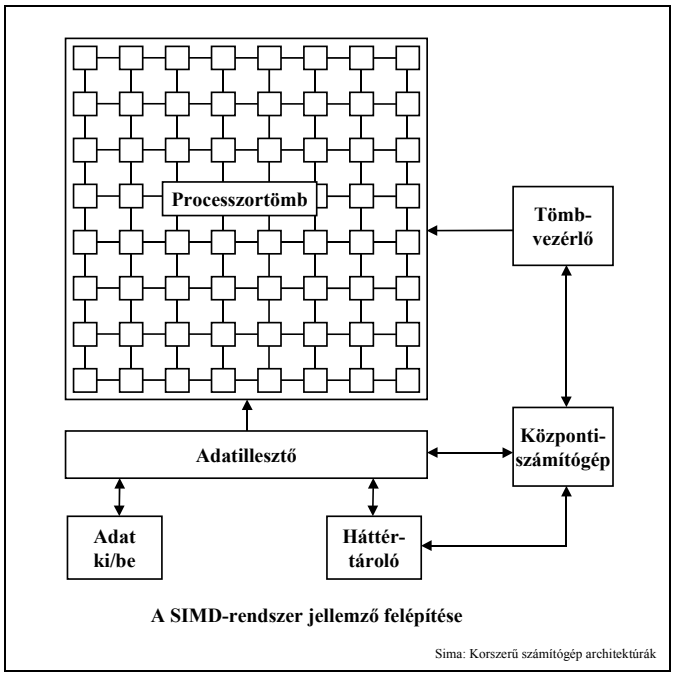
A tömbprocesszoros számítógépek a vektorszámítógépek továbbfejlesztett változatának is tekinthetők. A gépek több processzorral (műveletvégző egységgel) és ehhez kapcsolódó memóriamodullal rendelkeznek

A vektor- és mátrixműveletek végrehajtása nagymértékben gyorsítható, ha párhuzamos műveletvégzés (nem pipeline feldolgozás) történik a gépekben, azaz minden processzoron ugyanazt a műveletet hajtja végre a gép, a vektorok, vagy a mátrixok különböző elemeivel. Az elmondottak alapján, a folyamatok kezelése szempontjából, a tömbprocesszoros számítógépek az „egy utasításfolyam, több adatfolyam" (SIMD) csoportba tartoznak.

A tömbprocesszoros rendszerek legfontosabb elvei:

- Az elemek egy kétdimenziós tömbben helyezkednek el, és mindegyik a négy legközelebbi szomszédjához kapcsolódik
- Minden processzor párhuzamosan ugyanazt a műveletet hajtja végre.
- Minden processzornak van saját belső memóriája.
- A processzorok programozhatók, és különböző feladatok elvégzésére alkalmasak.
- Az adatok gyorsan haladnak át a tömbön.
- Kell egy központi számítógép, ami az utasítások forrása és egy külön rendszer az adatbevitelre, az eredmények megjelenítésére és tárolására.

A processzorokat és a memóriamodulokat egy vezérelhető kapcsolóhálózat köti össze, amely lehetővé teszi bármelyik processzor összekapcsolását bármelyik memóriamodullal. A számítógépek strukturális felépítését a következő ábra mutatja.



Forrás: Sima D.: Korszerű száítógéparchitektúrák

2-20. ábra: Tömbprocesszoros számítógépek

2.7.3. Üzenetátadásos számítógépek (multiprocesszoros architektúrák)

A teljesítőképesség növelésének további lépcsője a több processzoros gépek használata, amelyek minden processzora alkalmas önálló feladat feldolgozására. Így a processzorok között feladatmegosztás lehetséges. A multiprocesszoros architektúrák esetében lényeges annak kérdése, hogy

- milyen módon használják a processzorok a tárolót;
    - ugyanazt a memóriát használja-e megosztott módon mindegyik egység (shared memory),
    - vagy minden processzor számára önálló memória használata lehetséges(distributed memory);
- milyen módon létesítenek egymással kapcsolatot a processzorok;
    - megosztott sínhasználat (shared bus system) révén,
    - vagy közvetlen kapcsolat kiépítésének lehetőségével (interconnection network).

A multiprocesszoros gépek, a folyamatok kezelése szempontjából a „több utasításfolyam, több adatfolyam" (MIMD) kategóriájú számítógépek közé tartoznak.

A processzorok közötti, illetve a processzorok és a megosztott használatú memóriamodulok közötti kapcsolatok megvalósítására szolgáló hálózat statikus, vagy dinamikus kialakítású lehet. Statikus hálózat esetén, a csomóponti egységek (processzorok) között állandó struktúrájú a kapcsolatok kiépítése, míg a dinamikus hálózat esetében, a csomópontok közötti kapcsolatokat az igényektől függően lehet kialakítani.

A csomópontokban elhelyezkedő egységek között az adatok továbbítása többnyire egységes kialakítású üzenetek formájában történik. Az üzenetek tartalmazzák a célállomás (fogadó processzor) és a küldő processzor azonosítóját, valamint magukat a feldolgozandó adatokat és további kiegészítő információkat.

2.8. A párhuzamos architektúrák korszerű osztályozása (SIMA 1998)

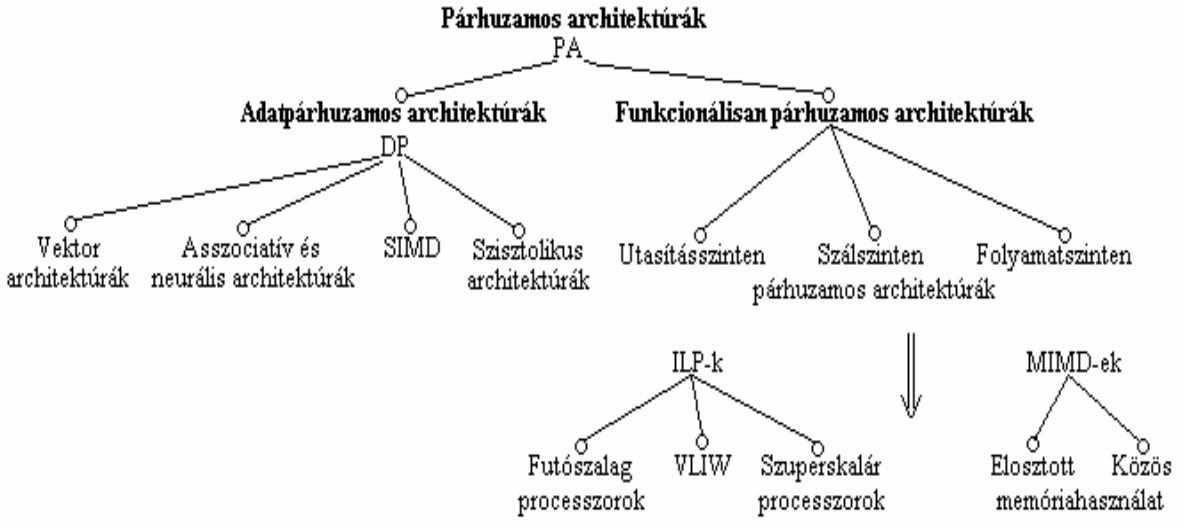
Párhuzamos architektúrák 1966-os Flynn féle osztályozásával szemben az időközbeni fejlesztések tendenciái alapján Sima Dezső osztályozása figyelembe veszi a kihasznált párhuzamosság fajtáját, és a funkcionális párhuzamosság szemcsézettségét. Különbséget tesz az adatpárhuzamos és a funkcionálisan párhuzamos architektúrák között. Így működési elvük alapján:

Adatpárhuzamos architektúrákhoz tartoznak a:

- Vektorprocesszorok
- Asszociatív és neurális architektúrák
- SIMD architektúrák
- Szisztolikus architektúrák

Funkcionálisan párhuzamos architektúrákhoz pedig:

- Utasításszinten párhuzamos architektúrák: (ILP Instruction Level Paralell) ezen belül:
    - Futószalag architektúrák
    - VLIW: Very Long Instruction Word architektúrák
    - Szuperskalár architektúrák
- Szálszinten párhuzamos architektúrák (MIMD architektúrák)
- Folyamatszinten párhuzamos architektúrák



Forrás: Sima D.: Számítógép architektúrák tervezési tér megközelítésben

2-21. ábra: A párhuzamos architektúrák osztályozása

A funkcionálisan párhuzamos architektúrák az általuk kihasznált párhuzamosság szemcsézettsége szerint csoportosíthatók.

A szemcsézettség: Meghatározza a viszonyt a feldolgozóelemek száma és az adatkészlet párhuzamossága között. Így megkülönböztetünk:

- **Finom szemcsézettségű rendszereket** jellemzőjük, hogy minden feldolgozóelemhez csak kevés számú adatelem tartozik.
- **Durva szemcsézettségű rendszereket,** melyeknél minden feldolgozóelemhez sok adatelem tartozik

Szisztolikus architektúrák

Ebben a rendszerben a bevitt adatokat keresztülpumpálják egy műveletvégző rendszeren, az eredmény általában egy hosszú adatsor. A "szisztolikus" elnevezés a szív hasonló pumpáló mozgásának analógiájából született.

Neurális architektúrák

A neurális hálózat három rétegből áll. Egy input rétegből, melyben elhelyezkedő sejtek a külvilággal állnak kapcsolatban. Egy rejtett rétegből, mely hidat képez az output réteg és az input réteg között, valamint egy output rétegből, mely a kívánt eredményt adja ki.

A neurális hálózatok lényege a neuronok (sejtek) közötti kapcsolat, melyet az úgynevezett szinaptikus súlyokkal lehet vezérelni. Ha egy sejt több, más neurontól kap input értéket, akkor mindegyik input érték egy szinaptikus súllyal lesz megszorozva. Az így kapott szorzatokat összeadja a fogadó neuron, majd az értéket egy átviteli függvény szerint áttranszformálja és az így kapott érték lesz a neuronsejt outputja:

2-22. ábra: Neurális hálózatok sejtprocesszorainak egyszerűsített ábrája

A neurális hálózatokban a tanulás során a neuronok közötti kapcsolat erőssége, vagyis a szinaptikus súly változik egy tanulási szabály alapján, azaz a sejt hálózatban a memóriát lényegében a szinaptikus súlyok értékével modellezik.

Ellenőrző kérdések:

1. Melyek a CPU legfontosabb architektúrális építőelemei?
2. Mi a feladata a vezérlő egységnek?
3. Mi a feladata az aritmetikai logikai egységnek?
4. Milyen megoldások vannak a műveleti vezérlésre?
5. Mi a regiszterkészlet funkciója?
6. Milyen fontosabb regisztereket ismer?
7. Milyen processzor üzemmódokat ismer, és mi a funkciójuk?
8. Ismertesse egy utasítás végrehajtásának lépéseit!
9. Milyen tényezők határozzák meg egy számítógép teljesítményét?
10. Hogyan történik a számítógépek teljesítményének mérése?
11. Mit értünk CISC processzorok alatt?
12. Mit értünk RISC processzorok alatt?
13. Hasonlítsa össze a CISC és a RISC processzorokat!
14. Hasonlítsa össze a Neumann és a Harvard architektúrákat!
15. Mit értünk utasításfolyam és adatfolyam alatt?
16. Hasonlítsa össze a SISD, a SIMD, a MISD és a MIMD architektúrájú gépeket!
17. Milyen lehetőségei vannak a processzorok nem strukturális gyorsításának?
18. Milyen lehetőségei vannak a processzorok strukturális gyorsításának?
19. Ismertesse az ILP processzorok fejlődését!
20. Mi a különbség a skalár és a szuperskalár processzorok között?
21. Mi a pipeline és hogy működik?
22. Milyen problémák léphetnek fel a pipeline végrehajtás során?
23. Hogyan lehet a pipeline végrehajtás során fellépő problémákat megoldani?
24. Ismertesse a VILW processzorok lényegét!
25. Melyek a főbb különbségek a VLIW és a szuperskalár processzorok között?
26. Mi az EPIC?
27. Ismertesse a vektorprocesszorok legfontosabb jellemzőit!

3. Adattárolás számítógépben

3.1. Központi vagy operatív memória

A processzor mellett a legfontosabb erőforrás a központi memória. Ebben található a végrehajtás alatt álló program, és a feldolgozásban felhasznált adatok. A központi memória egy címzési logikából és magát a fizikai tárolást végző fizikai memóriából áll.

Adattárolás a memóriában

A memória azonos méretű tároló rekeszek összessége. Minden rekesznek van egy címe (sorszáma), amivel azonosítani tudjuk. Amikor a processzor megcímez egy memóriarekeszt, akkor ez azt jelenti, hogy a rekeszhez rendelt sorszám segítségével kiválasztja azt a rekeszt, amelyet – írásra, vagy olvasásra - el akar érni.

A fizikai memória általában byte szervezésű. Ez azt jelenti, hogy a memória legkisebb címezhető egysége 1 byte, ez esetünkben a szokásos nyolc bit. Általában léteznek 1 byte-nál nagyobb írási/olvasási egységek is, (szavak, duplaszavak, stb.) is.

3.1.1. A memória címzése

A memória a külvilággal a buszrendszeren keresztül tart kapcsolatot. A címbusz a memória címregiszteréhez, az adatbusz pedig a memória adatregiszteréhez kapcsolódik. Azt pedig, hogy a memóriával írási, vagy olvasási műveletet akarunk végeztetni, a vezérlő buszon lévő memória írás/memória olvasás jelekkel választhatja ki a processzor. Az írni, vagy olvasni kívánt rekesz kiválasztását a címbusz címvezetéken lévő cím-adat határozza meg. A központi memória címzési logikája gondoskodik arról, hogy a fizikai memória megfelelő része ki legyen jelölve, és olvasási parancs hatására az ott tárolt adatot a memória átmeneti tárolójába (adatregiszterébe) írja, vagy írási parancs esetén az átmeneti tárolóból közvetlenül a memóriába írja.

A címbitek száma és címezhető memória mérete között szoros összefüggés van: a legkisebb memóriacím a 0, a legnagyobb pedig a címbiteken elküldhető legnagyobb szám. (Például 10 címvezetékkel 2^{10} = 1024, azaz 1K (kilobájt) memória címezhető meg, 11 címvezetékkel 2^{11} = 2048 azaz 2 K (kilobájt), ha ez pl. 20 bit lehet, akkor a maximális tárolóhely sorszám, azaz cím 2^{20}=1M (megabájt) lehet, és így tovább. A címvezetékek számát szokták a címbusz szélességének is nevezni. Lényeges jellemző tehát a cím lehetséges mérete, az, hogy hány bináris helyiértéket lehet felhasználni a cím értékének leírására.

Az utasítások címrésze általában nem az operandusok tényleges címét tartalmazza, azt a processzornak ki kell számítani. Ezt az eljárást címmódosítási eljárásnak nevezik. A processzor általában nem képes a teljes tár közvetlen megcímzésére, ugyanis a tárak mérete ezt nem teszi lehetővé. A teljes tárat ezért a processzor szegmensekre osztja. A szegmensek kezdete rögzített, szegmensbekezdésnek, vagy szegmenscímnek nevezik (i8088 esetében minden szegmens 16-tal osztható címen kezdődik). Az egyes adatok elérésére szükség van egy eltolási címre, amely a szegmensen belüli rekeszcímet mutatja meg. A tényleges cím a következőképpen áll elő:

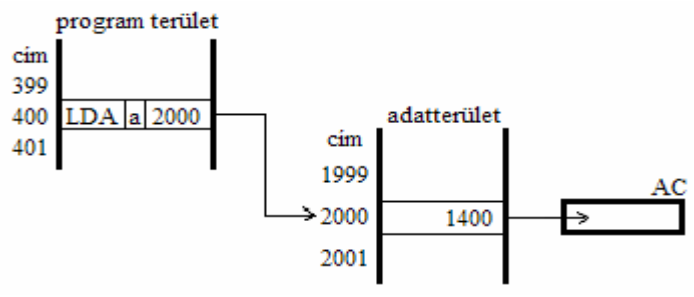
cím = N * báziscím + eltolás, ahol N egy szegmens mérete.

A báziscímet a processzor vagy a bázisregiszterben, vagy a szegmensregiszterben tárolja.

Közvetlen (direkt) címzési módok

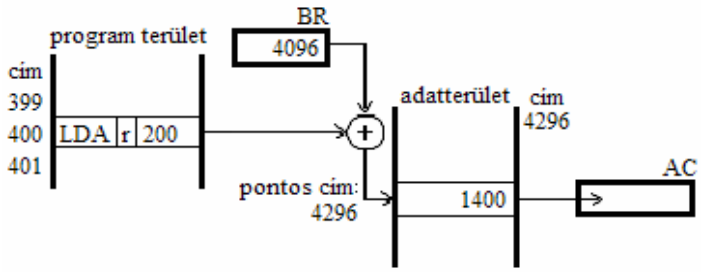
Abszolút címzési mód. Ennél a címzési módnál az utasítás címrészében az operandus valódi, pontos címe található. A cím vonatkozhat a memóriára, vagy a processzor valamelyik regiszterére. Regisztercímzés esetén kisebb hosszúságú címrészre van szükség, mint memóriacímzésnél. Az így történő címzés esetén a program és az adatok a memóriában nem

helyezhetők át, mivel az áthelyezéssel változik a cím. Ez elsősorban vezérlésátadó utasítások esetében okoz gondot.



3-1. ábra: ***Abszolút címzési mód***

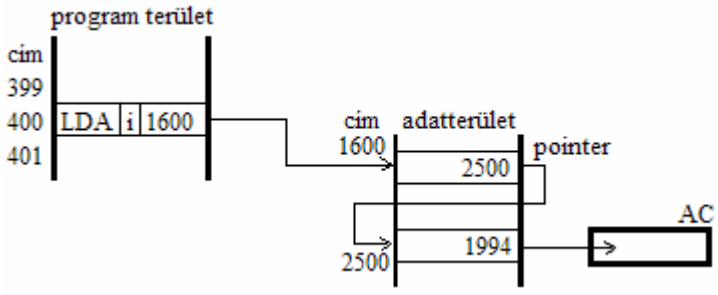
Relatív címzési mód. Az utasítás címrésze az operandus valamilyen alapcímhez viszonyított címét tartalmazza. Ez lehet adatmező kezdőcíme, társzegmens kezdőcíme, program kezdetének címe, vagy magának az utasításnak a tárolóbeli címe. Ez az alapcím. A tényleges, valós címet az alapcím és a relatív cím összeadásával állítja elő a processzor.



3-2. ábra: ***Relatív címzés***

Közvetett címzési mód (indirekt)

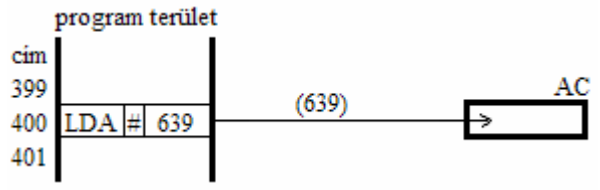
Indirekt címzésnél nem az operandus címe található az utasítás címrészében, hanem annak a tárolóhelynek a címe, ahol az operandus címét megtalálja a processzor. Egyes processzorok esetében ez a címzési mód lehet többszintű is. Az indirekt címzéshez a processzor használhat memóriabeli tárolóhelyet, vagy valamelyik saját regiszterét is.



3-3. ábra: ***Indirekt címzési mód***

Közvetlen adatcímzés

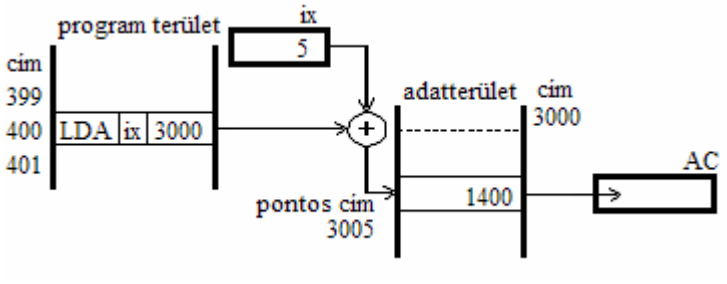
Közvetlen adatcímzés alatt azt értjük, amikor az utasítás operandus részében nem az operandus címe, hanem maga az operandus található (3.4 ábra). Ez megszabja az operandusok legnagyobb értékét (az utasítás címrészének korlátozottsága miatt), ezért elsősorban kis értékű konstansokkal való műveletvégzéseknél alkalmazható.



3-4. ábra: ***Közvetlen adatcímzés***

Indexelés

Adatsorozatokon való műveletvégzés esetén alkalmazandó az úgynevezett indexelt címzés(3.5 ábra). Ennél a módszernél az utasítás címrészében az adatsorozat első elemének a címe található és a címregiszterben (ix) található az ettől való eltérés, azaz hogy az adatsorozat hányadik elemével kell az utasításban megadott műveletet elvégezni. Egyes processzorok esetében alkalmaznak olyan megoldásokat is, ahol az adat előkeresése után az indexregiszter tartalma automatikusan növekszik, vagy csökken, így az adatsorozaton való műveletvégzés egyszerűbbé válik. Ezt autoindexelésnek nevezik. Az indexelt címzési mód nagyban hasonlít a relatív címzésre, a tényleges cím az utasítás címrészének és az indexregiszternek az összeadásával áll elő.



3-5. ábra: ***Indexelés***

Az Intel processzorok címzési módjait a MODE byte írja le. A címzési módok mindegyike indirekt, vagy indirekt relatív címzési mód. Az egyik operandus helyét a MODE byte MOD és R/M mezői, a másikat a REG mező tartalma határozza meg. A memóriabeli helyet, ahol a műveletvégzéshez szükséges adat található, általában ezen operandusok összege adja. Az operandusok mindegyike egy-egy regiszterben található.

3.1.2. Hibajelzés és hibajavítás

Valamennyi memóriarekesz mellett - melyek mindegyike, mint már említettük 8 adatbitet tárol - van egy 9. bit is, az ún. paritásbit. A paritásbit szerepe az, hogy segítségével lehetőség van egy esetleges tárolási hiba felismerésére. A paritásbit értéke 0 vagy 1 lesz attól függően, hogy a rekeszbe történő íráskor a 9 bitre kiegészített adatban az egyesek száma páros vagy páratlan. Ha a rekesz kiolvasásakor nem felel meg az egyesek száma a paritásbit értékének, akkor a CPU memóriahibát jelez. Ezzel a módszerrel csak egy bitnyi hiba jelezhető. Amennyiben további redundáns biteket is hozzáfűzünk a 8 bites adatainkhoz, hibajavító kód alkalmazására van lehetőség (ECC Error Correction Code). Ezek hatékony hibavédelmet biztosítanak, de jelentősen megdrágítják a memóriák árát.

A 8 bites adataink 256 különböző bináris kombinációt tesznek lehetővé. Ha a redundáns hibaellenőrző-javító biteket is hozzávesszük, megnő a bináris kombinációnk száma. Ha a 8 bites adataink tárolására megfelelő módon választunk az új, megnövelt számú kombinációinkból, akkor lesznek olyanok, amelyek érvényesek, és lesznek olyanok, amelyek adatábrázolásra nem használt, érvénytelen kombinációk. Így olvasásnál megfelelő algoritmus segítségével kiszűrhető a téves adat, és az is megállapítható, hogy is hogy milyennek kellett volna érkeznie.

Általános esetet feltételezve tegyük fel, hogy egy memória-szó áll m adatbitből, amihez hozzáadunk r redundáns, vagy ellenőrző bitet. Legyen az egész hossz n (n=m+r). Egy n bites egységet, melyet m adat és r ellenőrző bit alkot n bites **kódszó**nak nevezünk.

Adjunk meg két kódszót, mondjuk: 10001001 és 10110001. Meghatározhatjuk, hogy a megfelelő bitek közül hány különbözik. Estünkben ez szám 3. Hogy meghatározzuk, hogy hány bit különbözik, alkalmazzuk a két kódszóra a bitenkénti *kizáró vagy* műveletet (XOR) és számoljuk meg az 1-esek számát az eredményünkben! Azt a számot, amellyel két kódszó eltér egymástól **Hamming-távolságnak** is nevezik (Hamming, 1950). Tulajdonsága az, hogy ha két kódszó Hamming-távolsága d, akkor éppen d egyes hibára van szükség ahhoz, hogy az egyiket a másikba vigye. Például, ha a két kódszó 11110001 és 00110000, akkor Hamming-távolságuk éppen 3, így 3 egyes hibának kell történnie, hogy a két kódszót egymásba vigye.

Egy m bites memória-szóban az összes 2^m kombinációjú bit-elrendezés lehetséges, de az ellenőrző bitek számítási módja miatt a 2^n bit-elrendezésből csak 2^m bit-elrendezés érvényes. Ha a memóriaolvasás közben érvénytelen kódszót talál, a gép tudni fogja, hogy memóriahiba történt. Ismerve az ellenőrző bitek számítási algoritmusát, készíthetünk egy teljes listát az érvényes kódszavakról, és erről a listáról kikereshetjük azt a két kódszót, melyek Hamming-távolsága a legkisebb. Ez a távolság lesz az egész kód Hamming-távolsága.

A hibakeresés és hibajavítás tulajdonságai függnek a kód Hamming-távolságától. Ahhoz, hogy a gép d darab egyes hibát megkeressen d+1 távolságú kódra van szüksége, mert egy ilyen kóddal d darab egyes hiba képtelen egy érvényes kódszót egy másik érvényes kódszóra változtatni. Hasonlóan, ahhoz, hogy a gép d darab egyes hibát kijavítson 2d+1 távolságú kódra van szükség, mert így a lehetséges kódszavak olyan távolságra vannak egymástól, hogy még d bitnyi változtatás esetében is, az eredeti kódszóhoz még mindig közelebb vannak, mint a többihez, így ez egyedülálló módon meghatározható.

Egy egyszerű példaként a hibakereső kódokhoz vegyünk egy olyan kódot, amelyben egy egyszerű paritás-bitet csatolunk az adathoz. A paritás-bit megválasztott, így az 1-es bitek száma a kódszóban páros (vagy páratlan). Egy ilyen kódnak 2 a távolsága, mivel bármely egyes bit hiba olyan kódszót generál, melynek rossz a paritása. Tehát, ez a módszer csak egyes bithibák észlelésére alkalmazható.

3.1.3. A memória szervezése és típusai

Az aritmetikai műveletvégzés során, egy-egy számadat leírására nem elegendő 1 byte, ezért egy egységként 2-4-8 byte-ot használ a processzor. Ezt, a feldolgozásoknál, adatátviteleknél használt méretet, szokás szónak (word) nevezni. A szó tehát nem fizikai memória mértékegység, hanem inkább logikai, adat-mértékegység.

A központi tárak működésének fontos jellemzője az **elérési idő** (access time), és a **ciklusidő** (cycle time). Az elérési idő az adat kiolvasásának megkezdése, és a kimeneten való megjelenése között eltelt időtartam.

A ciklusidő valamivel hosszabb, mert magában foglalja a kiolvasási utáni, un. feléledési időt, amelyre egyes memóriáknak van szüksége, mielőtt a következő memóriához fordulna.

Tárolóeszközként kétféle típusú tárolót lehet megkülönböztetni: az írható-olvasható tárolókat és a csak olvasható tárolókat.

Az **írható-olvasható (**RWM read-write memory) **tárolók,** általános tárolási célra használhatók. Írható-olvasható tárak a regiszterek, a cache-tárak, a főtár, de ugyanakkor a háttértárak is, amelyek mágneses, vagy optikai elven működnek.

A félvezető alapú **RAM (R**andom **A**ccess **M**emory) modulokból épül fel például számítógépünk operatív memóriája és a regiszterei. Az ilyen típusú tárolók tartalma

tetszőlegesen törölhető, újraírható vagy kiolvasható. A tápfeszültség megszűnésével információ tartalmukat elvesztik. A **közvetlen elérésű (**Random Access**)** jelzőt azokra a komponensekre alkalmazzuk, amelyeknél bármely tárolt elem azonos (konstans) idő alatt érhető el. **Nem közvetlen elérésnél** az elérési idő az elérendő elem helyétől függően változhat.

- Szekvenciális elérésnél az elemek sorban érhetők csak el (pl.:mágnesszalag).
- Nem-szekvenciális elérésnél az elemek tetszőleges sorrendben elérhetők. (pl.: mágnes lemez, CD-ROM). A nem-random média több nagyságrenddel lassabb, mint a random elérésű.

A **RAM** tárolók típusai két nagy csoportba sorolhatók:

- *dinamikus* RAM (DRAM) tároló, amely alacsony elektromos teljesítményigényű, de tartalmát rövid idő alatt elveszti, ezért annak tartalmát ciklikusan fel kell újítani (refresh),
- *statikus* RAM(SRAM) tároló, amely nem igényli az állandó adatújítást és magasabb működési sebességű, egyszerűbb időzítési megoldásokat igényel.

A **csak olvasható** tárolók (**ROM**-**R**ead **O**nly **M**emory), nem változtatható információt tartalmaznak, azaz fix tárak. Az információt a gyártó, vagy a felhasználó írja be a ROM-ba, a benne lévő információ bármikor kiolvasható, tartalmát a felhasználó közvetlenül nem tudja módosítani. A RAM-okhoz képest lassabb működésűek. Jellemző típusaik:

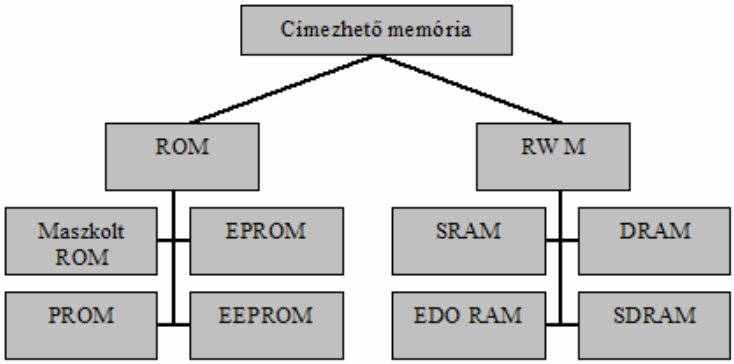
- A ROM-ok: tartalommal csak a gyártás során tölthetők fel (maszkprogramozott ROM)
- A PROM (Progammable ROM) típusú memóriaelemeket már nem a félvezető gyártója, hanem maga a felhasználó programozza be. A programozásához külön programozó-berendezés szükséges, amit a megfelelő égetési utasítás szerint kell beállítani. A programozás viszonylag időigényes, de a berendezés általában egyszerű és olcsó.
- Az EPROMOK kiküszöbölik a PROM-ok azon fogyatékosságát, hogy csak egyszer tölthetők fel adattal. Ezek már törölhetőek és újra programozhatóak. Egyik csoport az UV sugárral törölhető és elektronikusan programozható (EPROM vagy REPROM), a másik pedig elektromos úton törölhető és elektronikusan programozható (EEPROM vagy EAROM). Az EPROM-ok az IC tetején lévő kvarcablakon keresztül kb. 20-30 perc alatt törlődnek, ha UV sugár éri őket. Az EEPROM-ok elektromos impulzussal törölhetőek, a tár egész területe törlődik, szemben az EAROM-okkal (Electrically Alterbale ROM), ahol tetszőlegesen kiválasztható a törölni kívánt tárolócella. Ez utóbbi típusok már megközelítik a RAM tárolókat, de a programozás és törlés körülményesebb és sokkal tovább tart. Az EPROM-ok programozását általában egy külön kiegészítő egységgel, egy speciális programozó készülékkel végzik el.

A központi tárak működésének fontos jellemzője még az elérési idő (access time) és a ciklusidő (cycle time).

- Elérési idő az az időtartam, amely a kiolvasás megkezdése és az adatnak a tároló kimenetén való megjelenése között eltelik.
- A ciklusidő ennél valamivel hosszabb időtartam, mert magában foglalja a kiolvasás utáni feléledési időt (recovery time) is, amelyre egyes memóriáknak szüksége van a következő memóriához fordulást megelőzően.

A statikus RAM (SRAM) tárolók elérési ideje és ciklusideje közel azonos értékel, míg a dinamikus RAM (DRAM) ciklusideje körülbelül kétszerese az elérési időnek.

A memóriák működésüket illetően lehetnek szinkron és aszinkron működésűek. A 3.6 ábrán látható jelölések két kivételtől eltekintve ismertek. Az SDRAM szinkron dinamikus RAM-ot, az EDO RAM pedig Extended Data Output-tal ellátott RAM-ot jelöl. Működésükről, jellemzőikről az 5. fejezetben lesz szó.



3-6. ábra: ***A PC-kben található memóriák***

Az SRAM-ok tárcellái un. bistabil flip-flopok (egybites kétállapotú tárolóáramkörök). Memóriafrissítésre nincs szükségük. Működési sebességük megközelíti a processzor sebességét. Előállításuk költségei magasak.

A DRAM-ok celláiban az információt egy kondenzátor tárolja, amely a szivárgási áram miatt hamar elveszti töltését, ezért 2-4 ms-onként újra fel kell tölteni. Ezt a folyamatot frissítésnek hívják. A dinamikus RAM-okkal akár 4-szeres kapacitás is elérhető azonos méretben, hiszen egységnyi felületen több cella integrálható, mint az SRAM-nál, ezen kívül a címkivezetések száma a felére csökkenthető. A DRAM-ok, mint integrált áramkörök az említett két ok miatt jóval bonyolultabbak, mint bármely egyéb félvezetős társaik, viszont előállítási költségük, és sebességük is lényegesen alacsonyabb, mint az SRAM-oké.

Flash memória

Működéséi elvében az EEPROM-hoz hasonló, de a funkcióját tekintve, inkább a merevlemezhez hasonlítható, mint a hagyományos memóriához. A flash-memória kikapcsolt állapotban is megőrzi az adatokat. Az egyes cellákban tárolt feszültség adott határok közötti értéke felel meg a tárolandó bináris információnak. Az újabb típusok egyetlen cellája több bit tárolására is képes.

A működés a RAM-okhoz képest a törlés időigénye miatt elég lassú. A gyorsítás érdekében a törlés blokkonként történik, mégpedig oly módon, hogy a vezérlő kiolvassa a blokk tartalmát, törli a területet, majd újraírja az adott területet, de most már a felesleges adatok nélkül. A flash-memóriák akkor is megőrzik állapotukat, ha nincsenek feszültség alatt, így a cserélhető médiumok kiváló alapjául szolgálhatnak. (A flash-memória is felejt, de ennek időtartama években mérhető.) Előnyként szokták még megemlíteni a zajtalan működést, a kicsi és könnyű méretet valamint a háttértárakhoz képest rendkívül gyors elérési időt.

3.2. Memóriák hierarchiája

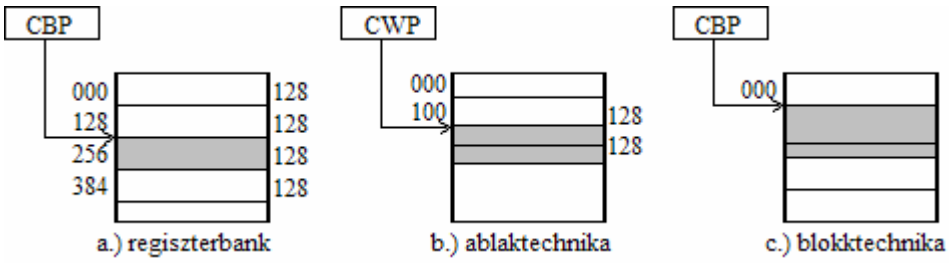
3.2.1. Regisztertárak

A processzorok teljesítményének fokozásában az elmúlt években fontos szerephez jutott a regiszterek darabszámának növelése. Ez segítette például a főtár és a processzor közötti adatforgalom csökkentését, és ezáltal az utasításvégrehajtás gyorsítását is.

A regiszterek általában nagyobb egységet alkotnak, melyeket regisztertáraknak nevezünk. A regisztertárakkal szemben támasztott elvárások pl.:

- a regisztertár általános volta, vagyis, hogy a tár általánosan felhasználható regisztereket tartalmazzon, vagy
- minél több regiszter legyen benne (100-500),
- támogassa a 3 címes elérési formát, avagy legyen képes egyszerre az első operandus (az első művelethez szükséges adat), a második operandus, ill.az eredmény címét is értelmezni; tárolni.

A regisztertárak tartalmának koordinálására több módszert alkalmaznak, melyek a következők lehetnek:



3-7. ábra: ***Regisztertárak megoldásai***

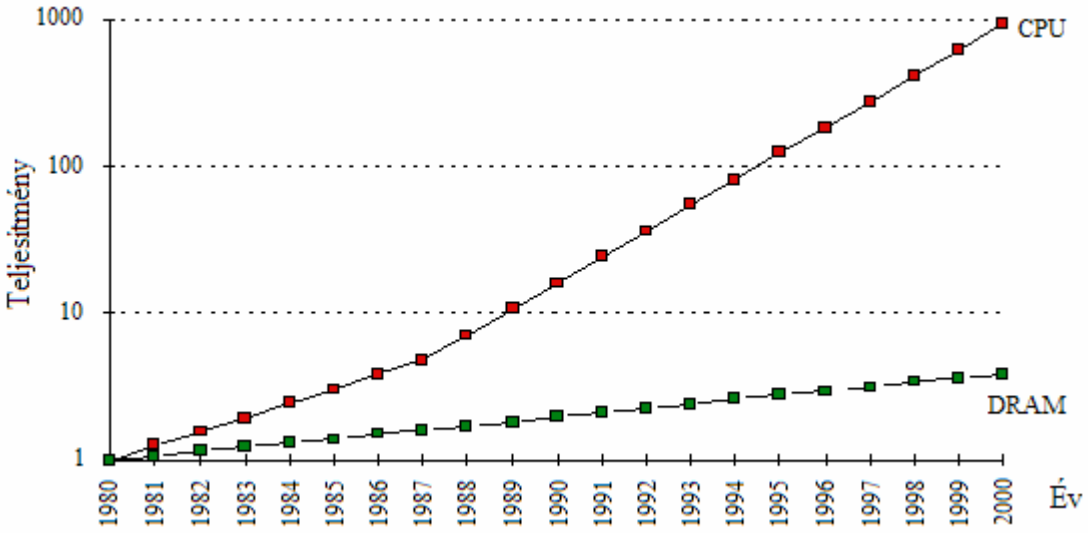
- A regiszterbank (register banking) esetében a regisztertömb fix méretű (azaz meghatározott darabszámú regiszterből álló) részekre van felosztva, melyek a felhasználói taszkokhoz kerülnek hozzárendelésre, a processzor számára látható tartományok kezdetét a CBP (current bank pointer) jelzi.
- Az ablaktechnikát alkalmazó (register windowing) esetben a regisztertömb szintén fix méretű részekre van felosztva, de ezeknek lehetnek az egyes taszkok között átlapolható részei (azaz ez esetben egy taszk egy másik taszk regisztereinek egy részét is „láthatja"). A regisztertár mérete meghatározott nagyságú, 2 valamely hatványa, a processzor számára látható tartományok kezdetét a CWP (current window pointer) jelzi.
- A blokktechnikánál (register blocking) a regisztertömb változó méretű átlapolható részekre van felosztva. CBP (current blokk pointer) jelzi.

Az ablaktechnika és a blokktechnika a különböző taszkok közötti, regiszterekben történő paraméterátadást segíti.

Egyes processzorokban található regiszterek száma lényegesen eltérhet attól a regiszterszámtól, amit a programozó ezekből „láthat". A nem látható regiszterek, a később ismertetendő regiszter átnevezési technikával vehetők igénybe.

3.2.2. Gyorsító (rejtett, vagy cache) tárak és megoldásaik

Ha az elmúlt időszakra visszatekintünk megállapítható, hogy számítógépeink műveletvégző sebessége szédületes iramban gyorsult. Érdemes azonban közelebbről is megnézni, milyen egységek, milyen arányban vettek részt a gyorsításban.



3-8. ábra: A processzorok és a dinamikus memóriák sebességnövekedése

A 3.8 ábrából jól látszik, hogy a memóriák sebességnövekedése egyre jelentősebb mértékben elmarad a processzorokhoz viszonyítva. A dinamikus RAM-ok sebességenövekedése kb. 7% évente, addig a processzoroké 60%. A különbség az idő függvényében pedig egyre nő. (A növekedés szabályát Gordon Moore 1965-ben fogalmazta meg: *az egységnyi területre jutó tranzisztorok száma nagyjából másfél évente megduplázódik.*)

Belátható, hogy a fejlesztőknek az alábbi tényekkel kell számolni:

- a processzor egyre gyorsabb lesz,
- a memória kapacitás növekszik,
- de a memória nem lesz annyira gyors mint a processzor,
- a számítógép sebességét jelentős mértékben meghatározza a memória sebessége,
- a processzor a lassú memóriával való együttmüködés esetén „holtidejű” várakozásokra van kényszerítve.

Kérdés:

- Hogyan tudjuk a CPU-t "teljes időben" foglalkoztatni?

További tények, melyekkel számolni kell:

- A gyors memória drága
- Lassú memória jelentősen ronthatja az összteljesítményt

Tervezési filozófia

- Fogadjunk el egy olyan megoldást, amelyik mindkét aspektust figyelembe veszi!
- Tároljuk a gyakran használt dolgok egy kicsi mennyiségét gyors/drága memóriában, amit cache-nek nevezünk!
- Helyezzünk el minden mást egy lassú/olcsó memóriába!
- Tegyük gyorssá a kettő közötti adatcserét!

A megoldást a számítógépben lévő adattároló-egységek hierarchikus felépítése jelenti. A hierarchikus felépítést az jellemezi, hogy az adathozzáférések gyakorisága szerint a processzor közelében igen gyors elérésű, kis kapacitású és drága tárolók helyezkednek el, majd ahogy távolodunk a processzortól az adattárolók elérése lassul, kapacitásuk nő és az egységnyi információtárolás költsége is csökken.

3.2.3. A cache tárolók

Ha a hagyományos felépítést tekintjük, a processzor ún. várakozó állapotba kerül, ha a szükséges adatot a központi tárból, a memória nem tudja olyan ütemben adni, mint ahogy a processzor ütemezése miatt igényelné. Ekkor semmilyen műveletet nem képes elvégezni addig, amíg a memóriától az igényelt adatot meg nem kapja. Nyilvánvaló, hogy ezek a várakozási állapotok a teljes számítógépes rendszer műveletvégző képességét lényegesen leronthatják. A megoldást a processzor és a főtár közé behelyezett gyorsabb, de drágább cache tár jelenti.

A tároló-hierarchia egyes szintjén lévő tárolók hozzáférési idejét és tárolókapacitásuk nagyságrendjét mutatja be az 3.9-as ábra.

Név:	Regiszter	Cache	Operatív tár	Lemezes tár
Sebesség:	≤ 1ns	< 2 ns	10 ns	10 ms
Méret:	100 Byte	10KByte	100MByte	100GByte

3-9. ábra: Tárolók hozzáférési ideje és tárolókapacitásuk nagyságrendje

- A processzor tartalmazza a **regisztereket**, melyek az információ átmeneti tárolását végzik és ezek a számítógép leggyorsabb (ugyanakkor legdrágább) tárolóegységei.
- A **gyorsító (cache) tárak** tárolókapacitása — melyek a futó programból a processzor számára leggyakrabban szükséges utasításokat és adatokat tartalmazzák—, két nagyságrenddel nagyobb a regiszterekénél, ugyanakkor az adatok kiolvasása kb. 2-szer, annyi időt igényel, mint a regisztertáraknál.
- A **főtár** — mely az aktuálisan végrehajtott programot és ennek adatait tárolja — kb. 10-szer lassúbb a regisztereknél, ugyanakkor viszont adattároló képessége több százerszer nagyobb.
- A programvégrehajtáshoz aktuálisan nem szükséges adatokat tároljuk a háttértarolókon, ezek általában mágneslemezek. Ezek tárolókapacitása 10 milliószor nagyobb a regiszterekénél, viszont 5 milliószor lassúbbak.

A cache vagy más néven gyorsító, vagy rejtett tárolók az utasítások és adatok átmeneti tárolására szolgáló, viszonylag kisméretű, gyors memória pufferek. A főtár bizonyos területének másolt részeit (blokkjait) tartalmazzák. A cache tárral kapcsolatban a blokk alatt a főtár rekeszeinek olyan egymás utáni sorozatát értjük, melynek bemásolása a cache-be egy lépésben megtörténhet.

A cache memóriák önálló vezérléssel rendelkeznek és a felhasználó számára láthatatlanok (elérhetetlenek). A cache vezérlő egy „intelligens” áramkörökből felépített hardver egység. (Intelligencia alatt ez esetben az értendő, hogy például a vezérlő képes a különböző bemásolási és adat-visszaírási stratégiák, algoritmusok kezelésére.)

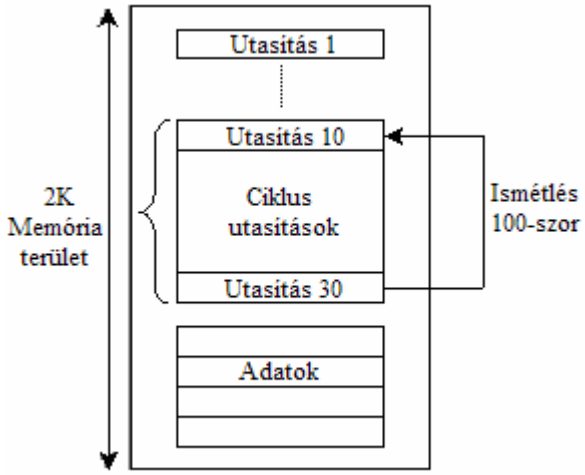
A cache memóriakapacitásával és a cache-ben elhelyezett központi tár blokkmásolatok méretével közvetlen összefüggésben van az a megfigyelés, hogy egy program végrehajtása során a vezérlés igen nagy valószínűséggel egy meghatározott memóriatartományon belül marad.

A **program-lokalitás elve** szerint a programok egy kis időintervallumban a memória címterének csak egy relatíve kis részét veszik igénybe. Ez a megállapítás nemcsak a programutasításokra, hanem az adatokra is igaz. Ennek két oldala van:

- **Időbeli lokalitás***:* Ha egy adatra vagy egy utasításra hivatkozás történik, akkor ez nagy valószínűséggel rövid időn belül újra megtörténik. (ciklus). Másképp megfogalmazva a programok egy kis időintervallumban a címtér viszonylag kis hányadát próbálják meg elérni.

- **Helyi lokalitás**: Ha egy adatra vagy egy utasításra hivatkozás történik, akkor ez nagy valószínűséggel a környezetében lévő címekre is megtörténik. (Soros utasítás végrehajtás.)

A megfigyelt jelenség igencsak hasznosnak bizonyult a gyakorlatban.



3-10. ábra: ***A lokalitás elve***

A programok lokalitásából az következik, hogy lehetőség van arra, hogy az éppen végrehajtás alatt lévő program és adatrész környezetét egy kisebb kapacitású, de nagyon gyors memóriában tároljuk. Ez a memória a cache. Tulajdonképpen a központi memória és a CPU közé illesztjük a gyorsító memóriát. A CPU-ban az operatív tár címe leképződik egy cache-beli címre is: ha kell valami és az a cache-ben is megvan, akkor csak onnan veszi elő. Ha nincs a cache-ben, akkor a kívánt memóriarekesz tartalmát, és a várhatóan kis időn belül szükséges soron következő terület tartalmát is automatikusan bemásolja a cache-be, blokkos átvitellel.

A cache-tárak tartalom szerinti visszakeresést tesznek lehetővé. A visszakeresés a keresett adat címe alapján történik. A cím tárolásakor csak annak egy részét kell a cache-ben tárolni, amely alapján a blokk kezdőcíme meghatározható. Ezt a processzor helyezi el a cache-ben és 'tag'-nek nevezik. A cím mellett a tárolt adatok állapotára vonatkozó információt is tárol a cache. A számítógép rendszerek gyakran többszörös cache-t használnak.

3.2.4. Az L1 és L2 cache

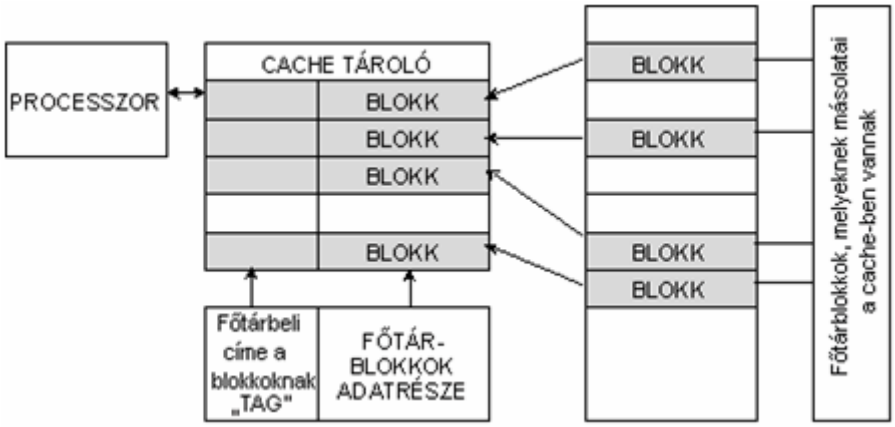
Mikroszámítógépeknél a CPU és a központi memória között található cache több szinten lehet megvalósítva. Így a processzortól kiindulva megkülönböztetünk L1, azaz elsőszintű (Level 1) cache tárolót és L2 másodszintű (Level 2) cache-t. A cache tárolót közvetlenül beépíthetik a processzorba (on-chip), vagy elhelyezhetik a processzoron kívül is (off-chip).

3.2.5. A cache működésének elve, cache hit és miss

A belső cache memória előnyei kézenfekvőek. Ha a processzornak utasításokra vagy adatokra van szüksége, akkor először ellenőrzi a cache tartalmát. Ha ott megvannak az adatok, akkor cache-találatról (cache-hit) beszélünk. A találatot a cache-vezérő azzal állapítja meg, hogy a bemásolt blokkokhoz tartozó címrészek (toldalék vagy „tag”) alapján valamelyik cache blokkban benne van-e a processzor által igényelt adat főtárbeli címe. Ellenkező esetben, ha a processzor által igényelt adat nincs meg a cache-ben, ezt tévesztésnek vagy cache miss-nek nevezzük. Ha tehát nincs találat, elkerülhetetlenné válik a lényegesen lassúbb DRAM memóriához való fordulás, a gyorsító tár vezérlője bemásolja ezt és még néhány rekeszt a cache tárolóba. Ugyanis a lokalitás elve alapján nagy valószínűséggel feltételezhető, hogy a

processzornak legközelebb a soron következő rekeszre lesz szüksége. Ha ez így van a processzor a következő olvasási műveleteket már várakozás nélkül képes lesz végrehajtani, mindaddig, amíg a szükséges adatok a cache-ben rendelkezésre állnak.

Az adatok bemásolása a cache-be blokkonként (azaz a cache sorok méretének megfelelően) történik (lásd a 3.11 ábrát) és igen rövid idő alatt (csak néhány órajel szükséges hozzá) végrehajtható.



3-11. ábra: ***A cache működése***

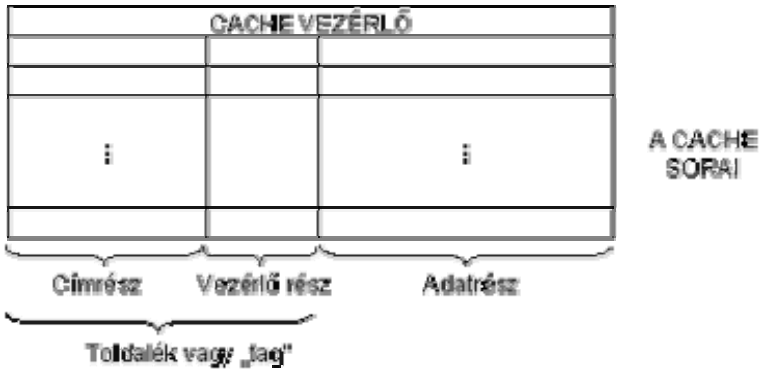
Egy cache tároló legfontosabb jellemzői a következők:

- a cache-tár mérete, a blokk mérete (az adatcsere a főtár és a cache között mindig blokkos formában történik),
- egy blokk kikeresésének eljárása a cache tárban,
- aktualizálási eljárás, amely szerint a processzor által módosított adatot a cache-tárba és a főtárba írjuk,
- a megfelelő helyettesítési stratégia (replacement policy), amivel eldöntjük, hogy a cache-ben melyik blokkot lehet felülírni, ha új blokkot kell bemásolni,

A cache teljesítményét a találatok és tévesztések aránya határozza meg. A találatok és tévesztések százalékos aránya a cache méretétől és a cache vezérlőtől függően általában 90% feletti érték.

3.2.6. A cache tárak felépítése és típusai

A cache tárak általános felépítését a mellékelt ábra mutatja be.



3-12. ábra: ***Cache tárak általános felépítése***

A főtár egyes blokkjai lemásolva a cache soraiban kerülnek elhelyezésre. Minden sor két részből épül fel:

- a toldalék (tag) egyrészt a főtárból bemásolt blokkra vonatkozó címinformációkat tartalmazza, másrészt itt kerülnek bitenként kódolva letárolásra a cache blokk adataira vonatkozó érvényességi információk;
- az adatrész tartalmazza a bemásolt főtár-blokk változatlan vagy a processzor által már módosított adatait.

3.3. Asszociatív tárolók

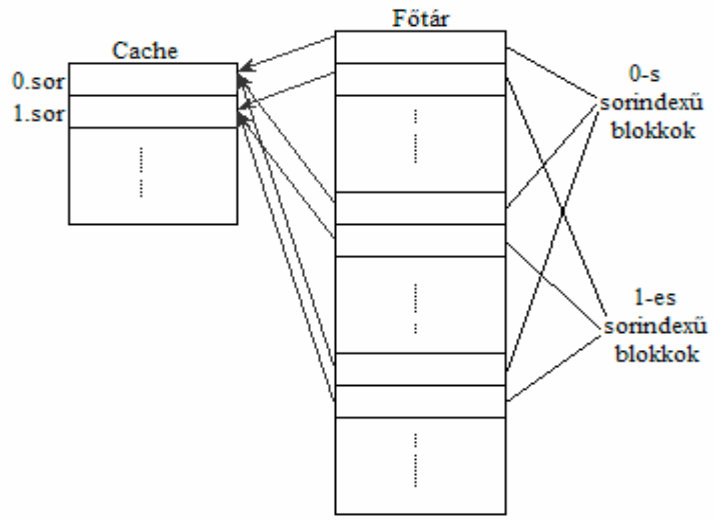
Az eddig megismert tárolók működési elve azon alapult, hogy az adatokat a memória rekesz címe szerint helyeztük el bennük, vagy olvastuk ki onnan. A cache tárak ettől eltérő módon, részben, vagy teljesen asszociatív tárolóként működnek.

Asszociatívnak nevezzük azokat a tárolókat (CAM = Content Addressable Memory), melyek az adatok visszakeresését tartalom szerint végzik, azaz a tárolóban lévő adatokat nem kell megcímezni, hanem a keresett adatokat a tároló bemenetére helyezve eldönthető, hogy az adatokat a tároló tartalmazza-e vagy sem.

Az asszociatív elven működő cache tárolók esetében a visszakeresés adatai, a főtárbeli blokkok címei, vagy részcímei lesznek.

A cache memória mérete jóval kisebb a főtár méreténél. Ezért a végrehajtandó utasításoknak és adatoknak csak egy részét tartalmazza. A program futása közben azonban szükségessé válik a cache memóriába újabb főtár terület bemásolása. Erre vonatkozó eljárások a következők:

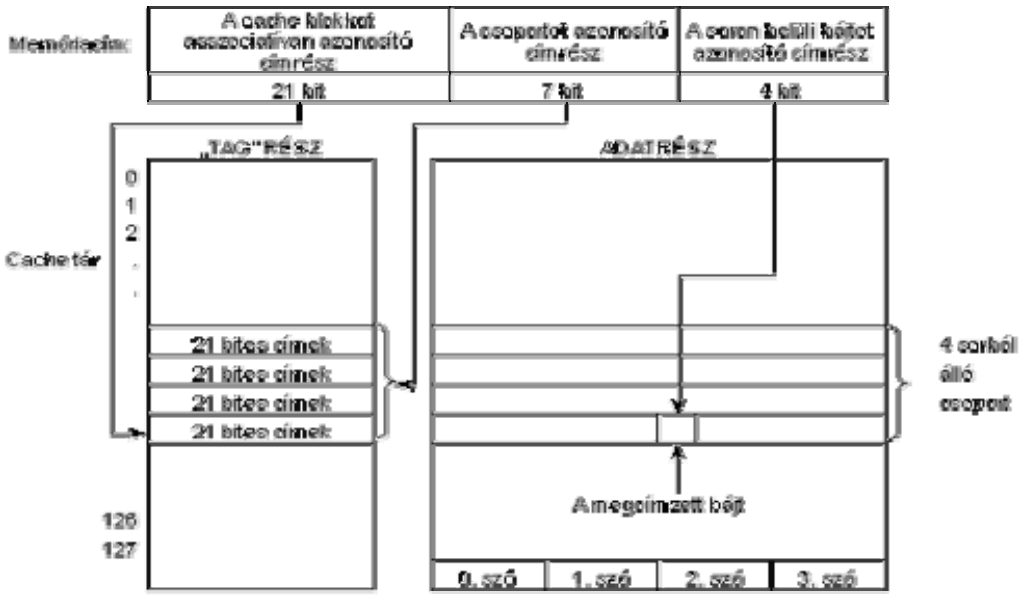
- **Teljesen asszociatív cache** (fully associative cache) tárolóknál egy főtárbeli blokk a cache bármelyik sorába bemásolható, a blokk címe (sorszáma) pedig bekerül a cache toldalék részébe. Ha a processzor egy adatot keres a cache-ben, akkor az adat memóriacíméből képzett blokksorszám asszociatív módon összehasonlításra kerül a cache-ben lévő blokkok sorszámaival. Az összehasonlítás rendkívül gyorsan, minden sorra vonatkoztatva azonos időben történik meg. Ezt a cache típust nagyon gyorsnak és egyúttal nagyon drágának jellemezhetjük, mivel felépítése bonyolult, annyi összehasonlító áramkörre van szükség, amennyi sort tartalmaz a cache.
- **Közvetlen leképezésű** (direct mapping) **cache** tárolóban egy blokk csak a cache egy konkrét sorába kerülhet. Ez a sor úgy kerül meghatározásra, hogy a blokksorszám alsó „n” db bitjét használjuk fel sorindexnek. Például, ha a cache-ben 256 sor van, akkor (mivel $2^8 = 256$) a blokksorszám legkisebb helyiértékű 8 bitjével indexelünk. Ez gyakorlatilag azt jelenti, hogy a főtárban egymástól azonos távolságra elhelyezkedő blokkok azonos cache sorba fognak bekerülni (lásd a 3.13 ábrát).



3-13. ábra: ***Közvetlen leképezésű cache.***

A közvetlen leképezésű cache-k esetében ezért a különböző blokkokban lévő, de azonos sorindexű blokkokban található adatok hozzáférése rendkívül lelassul, mivel ez minden esetben blokkcserét eredményez a cache-ben. Ez azt jelenti összességében, hogy a találati arány is kisebb lesz az asszociatív tárat használó cache-hez képest.

Ugyanakkor azt is meg kell jegyezni, hogy a közvetlen leképezésű cache tárolók olcsók és gyors visszakeresést biztosítanak.



3-14. ábra: ***Csoport asszociatív tároló***

3.3.1. Helyettesítési stratégia, blokkbemásolás a cache-be

Ha egy főtárblokkot be kell másolni a cache-be, az első kérdés annak eldöntése, hogy hogy a bemásolandó blokk adatai a cache melyik sorába kerülhetnek be, azaz az új blokk adataival, melyik más régebben a cache-be másolt blokk adatait lehet felülírni. Erre a célra a cache vezérlésében különböző helyettesítési stratégiákat alkalmaznak. Ezek közül a legelterjedtebb az LRU (least recently used) stratégia, mely esetében a processzor által legrégebben használt blokk kerül felülírásra. A legkevésbé használt blokk kiválasztásához a

cache-ben természetesen valamilyen módon tárolni kell azt az információt, hogy az egyes blokkokat mikor használta a processzor. (Ezt a blokk „életkorának” is szokták nevezni. Ilyen értelemben az a blokk lesz a legfiatalabb, melyet aktuálisan használ a processzor, és az a blokk lesz a legöregebb, melyet legrégebben használt a processzor.)

Az új blokknak a cache-be történő beírására szintén többfajta eljárás létezik. A leggyakoribbak:

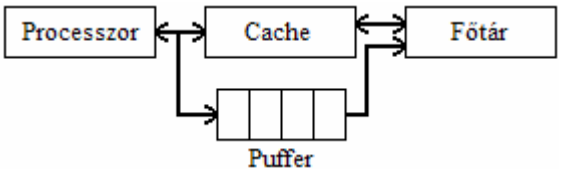
- **demand fetching**, ami azt jelenti, hogy csak a processzor adatigénye esetén keresik ki a főtárból a megfelelő blokkot és töltik be a cache-be. Ezzel párhuzamosan a processzor is azonnal megkapja az adatot (load through);
- a **prefetching**, ami azt jelenti, hogy ha a főtárból be kell tölteni egy blokkot a cache-be, akkor automatikusan betöltésre kerül a főtár következő blokkja is. (Feltételezhető, hogy ha a processzornak szüksége volt egy blokkra, akkor nagy valószínűséggel szükség lesz a rákövetkező blokk adataira is. Ez az eset áll elő, ha utasításcache esetén az utasításszámláló például a blokk végét címzi.).

3.3.2. A cache-ben megváltoztatott adatok visszaírása a főtárba

Ha a processzor egy művelet végrehajtás során megváltoztat egy adatot a cache-ben, akkor igen rövid idő alatt a főtár tartalmát is módosítani kell, hogy a két memória tartalma azonos legyen. Ez két eljárással történhet: közvetlen átírással, illetve visszaírással.

A **közvetlen átírás (write throug)** eljárás automatikusan biztosítja főtár és a cache adategyezőségét, mivel a gyorsítótár írásával együtt megtörténik a főtár írása is. Ennek alkalmazása esetén a főtár írásműveleteinek végrehajtási idejét a cache alkalmazása nem javítja.

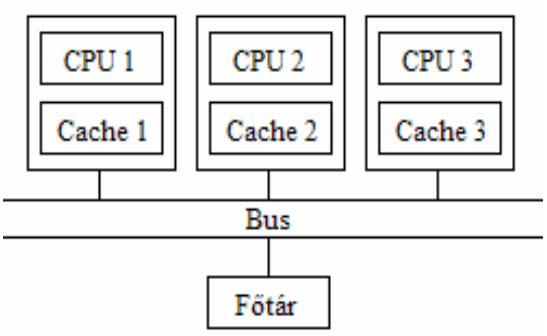
A **pufferelt közvetlen átírás** (buffered write through) módszere a közvetlen átírás hatékonyságát javítja azzal, hogy a megváltoztatandó főtárbeli adatokat egy (tipikusan 4 elemű) íráspufferbe írja be, és nem várja meg a főtár írásának a befejeződését. A főtár aktualizálása azonnal megkezdődik, de a sebességkülönbség miatt némi késleltetés is fellép. Amennyiben műveletet akarnánk végezni a visszaírás alatt álló, (de még nem visszaírt) adatokra, a végrehajtott memóriaművelet hamis eredményeket szolgáltathatna. Ennek kivédéséről hardver szinten kell megfelelő áramkörökkel gondoskodni. Továbbá gondot jelenthet az is, ha a puffer megtelne, mivel ez esetben a processzor, várakozásra kényszerül.A pufferelt közvetlen átírás elvét mutatja be a 3.15 ábra.



3-15. ábra: ***Pufferelt közvetlen átírás módszere***

A **visszaírás** (write back) eljárásnál a gyorsírótárban módosított adat csak akkor kerül visszamásolásra a *főtár*ba, ha a cache-nek azt a sorát mely módosított adatot tartalmaz, felül kell írni egy a főtárból bemásolandó újabb blokkal. Ez a módszer gyorsabb a közvetlen átírásnál, viszont minden cache-beli sor esetében meg kell jegyezni, hogy az adott sor lett-e módosítva. Ennek adminisztrálására soronként egy kiegészítő bit szolgál, melynek állapota "módosult" (alter) vagy "piszkos" (dirty) lehet. Ha írás történik a *cache*-be, akkor megfelelő sorban a "módosult" bit beállításra kerül. Ha a sor tartalmát cserélni kell egy másik főtár blokkal, akkor a módosult bit beállított állapota jelzi a cache vezérlőnek, hogy a sor tartalmát előzőleg vissza kell írni a főtárba. Nyilvánvaló, a visszaírási eljárást akkor célszerű alkalmazni, ha a gyorsító tár sorait sokszor kell módosítani a processzornak, mielőtt azok visszaírásra kerülnének a főtárba.

A fentebb megismert viszonylag egyszerű cache kezelési módszerek a multiprocesszoros rendszerekben azonban nem alkalmazhatók, egyrészt azért, mert több cache-t tartalmaznak, másrészt azért, mert a főtárat a processzorok meghatározott módon, de közösen is használhatják.



3-16. ábra: ***Cache tárolók multiprocesszoros rendszerben.***

A főtár és a cache azonosságának biztosítására szabvány eljárásokat dolgoztak ki. A MESI protokoll egy ilyen szabványos eljárás, amely nevét a blokkok lehetséges állapotainak angol kezdőbetűiből képezték.(**M**odified, **E**xclusive, **S**hared, **I**nvalid). Ezek szerint egy blokk lehet:

- **Módosított**, ha a cache blokkja a főtár-blokkhoz képest módosítva lett és csak a cache tár blokk tartalmazza az aktuális adatokat.
- **Kizárólagos**, ha a cache blokkja megegyezik ugyan a főtár-blokkal, de ez a blokk a többi cache-ben nem található meg.
- **Megosztott**, vagy közös esetén a főtár-blokk és a cache blokk érvényes adatokat tartalmaz, és a blokk a többi cache-ben is megtalálható.
- **Érvénytelen** egy cache blokk amennyiben olyan adatot tartalmaz, amelyet például egy másik cache-ben már módosítottak.

Az előzőekben már megpróbáltuk összehasolítani a regiszterek, a cache és az operatív tár sebességét. A számértékek nagyságrendjét elég nehéz közvetlenül érzékelni, ugyanakkor fontos megérteni a lényegét. Ezért példaként tegyük fel, hogy a regiszterek hozzáférési ideje 1 másodperc és számítsuk ki, hogy ez milyen időt igényelne a különböző tárolóeszközökön az adathozzáférés, kiegészítve ezt egy ún. lassú periféria, a nyomtató működésével, mely példánkban 1 perc alatt nyomtat ki ez lapot.

Ekkor a következő táblázatot kapnánk az adatok elérési idejére:

3-1. Táblázat: Adatelérési idő a tárhierarchia szerint

Regiszter	Gyorsftótár	Főtár	Háttértár	Nyomtató
1 sec	2 sec	10 sec	kb. 12 nap	kb. 170 év

Ebből a táblázatból a következő következtetéseket lehet levonni:

- ha a processzornak a főtárból kell adatot kiolvasni, akkor már komoly várakozásokra kényszerül;
- ha a processzornak a művelet végrehajtáshoz olyan adatra van szüksége, melyet a háttértárolóról kell kiolvasni, akkor már „elviselhetetlen" ideig kell várakoznia;
- egy lassú periféria művelet végrehajtási ideje a processzor számára gyakorlatilag végtelennek tűnik.

Ezért a háttértárakat és a lassú perifériákat önálló vezérlőegységekkel és átmeneti puffer (buffer) rátolókkal kell ellátni, melyek biztosítják, hogy ezek az eszközök az I/O műveleteket relatívan a processzortól függetlenül képesek legyenek végrehajtani.

A tárolókezelés alapfeladatát a következőképpen fogalmazhatjuk meg: biztosítsa a processzor művelet-végrehajtásához szükséges adatokat, összehangolva a tároló-hierarchia egyes szintjein lévő memóriaegységek működését.

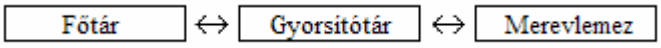
A tároló-kezelés feladatainak megoldását a számítógépek tároló kezelő hardver egysége a MMU = Memory Management Unit biztosítja. Ez lehet a processzorba beépített áramkörökből álló egység (például Intet processzoroknál az AU = Adress Unit), de lehet önálló hardver elem is (például a RISC processzoroknál).

3.3.3. Lemezgyorsító tárak

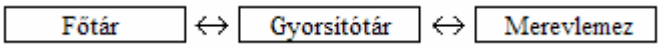
Láttuk már, hogy a mágneslemezes háttértárak elérési ideje több nagyságrenddel nagyobb a főtáréhoz képest. Ezért az adatok írása/olvasása a háttértárolóra, illetve a háttértárolóról egy nagyteljesítményű processzorhoz képest rendkívül lassú. E probléma feloldása érdekében a főtár és a mágneslemez közé is beiktatható egy gyorsító cache-tár.

Gyorsítás nélkül a főtár közvetlenül a lemezről kapja az adatokat:

Gyorsítótár alkalmazásánál első olvasáskor a lemezről a gyorsítótárba hozzuk be az adatokat, majd ott tároljuk:



További olvasáskor, ha az adatok a gyorsítótárban vannak, akkor azokat innen kapja a főtár:



Ha feltételezzük, hogy a mágneslemezről egyszer már kiolvasott adatokat a processzor többször is fel fogja használni, akkor egy gyorsító cache-puffer beiktatásával a processzornak kevesebb számú esetben kell közvetlenül a lemezről kiolvasni adatokat. Az eljárást a mellékelt ábra szemlélteti.

A merevlemez gyorsításnak két különböző megoldása:

Gyorsítás programmal, amely az operációs rendszer részét képezi. Ez esetben a lemezgyorsító tárrészt a főtárból kell kijelölni.

Gyorsítás hardver eszközzel, amely előnyösebb és hatékonyabb megoldás. A korszerű merevlemez interfészek általában tartalmaznak egy a merevlemez vezérlőelektronikájával egybeépített több tíz Mbájt nagyságrendű kapacitással rendelkező hardver gyorsító-tárat is. A hardvergyorsítók működésének a lényege a következő:

Írásnál, processzor által lemezre írandó adatokat a gyorsító vezérlője fogadja, és letárolja a cache-ben, valamint visszaigazolja a processzornak a lemezművelet végrehajtását. Ezt követően kezdi meg a vezérlő az adatoknak a kiírását a lemezre és így ennek ideje alatt a processzor folytathatja a munkáját.

Olvasásánál, a gyorsító vezérlője megvizsgálja, hogy az adatok megtalálhatók-e a cache-ben. Ha igen, akkor az adatokat közvetlenül a cache-ből továbbítja, lemezművelet végrehajtása nélkül. Ha nem, akkor a lemezről a cache-be olvassa, de nem csak az aktuális adatot, hanem a lokalitás értelmében egy egész blokkot, hiszen a következőkben biztosan szükség lesz ebben található adatra.

Memória hierarchia a számítógépekben

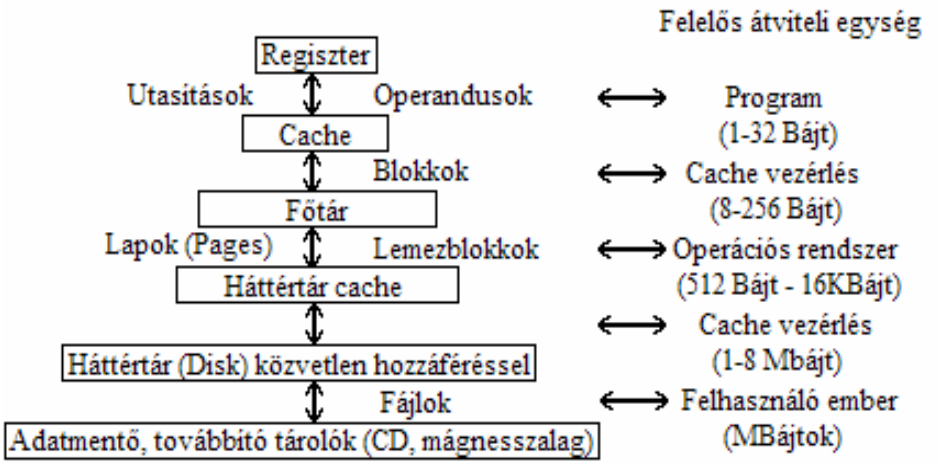
A számítógép-architektúrák tervezése során az egyik legfontosabb megoldandó kérdés a különböző sebességű adattároló eszközök közötti adatforgalomnak oly módon történő biztosítása, mely lehetővé teszi a processzor folyamatos, várakozás nélküli működését a processzor és a központi memória között, illetve a központi memória és a háttértárolók között.

Így megfogalmazhatók a tároló-kezelés legfontosabb feladatai:

- Biztosítsa a processzor művelet-végrehajtásához szükséges adatokat,
- Hangolja össze a tároló-hierarchia egyes szintjein levő memóriaegységek működését

A tároló-kezelés feladatait a Memória Menedzser Egység (MMU, Memory Management Unit) látja el, amely lehet a processzorba építve, pl. az Intel processzorok AU Adress Unit-ja, vagy a RISC processzorok esetén alkalmazott külön hardver elem.

A tároló hierarchia egyes szintjeit és a közöttük léterejövő adatáramlást mutatja a 3. 17 ábra.



3-17. ábra: A tároló hierarchia egyes szintjei közötti adatáramlás

3.4. Társzervezés

Egy program végrehajtásához a megfelelő programrésznek és az általa feldolgozandó adatoknak a főtárban kell lennie. Mivel az aktuálisan nem futtatott programoknak nem kell feltétlenüla főtárban lenniük, így azokat és a hozzájuk tartozó állományokat tarthatjuk a háttértárolón is, és csak akkor töltjük majd be a központi memóriába ha szükség van rá.

A főtár a számítástechnika eddigi története során mindig az egyik legszűkebb erőforrás volt, mivek az egyre komplexebb alkalmazások egyre nagyob méretű programokkal voltak megoldhatók, és a kezelt állományok mérete is egyre nőtt. A multiprogramozott, és a multitasking üzemmód elterjedését követően vált jellemzővé, hogy a számítógépen „egyidejűleg” több olyan nagy memória igényű programot kell futtatni, amelyek együttesen nem férnek be a főtárba. Az éppen nem használt programrészek és adatok háttértáron való elhelyezését indokolja az a tény is, hogy az egységnyi információtárolás költsége a háttértárakon töredéke a főtárhoz viszonyítva.

3.4.1. Virtuális tárkezelés

A virtuális tárkezelés lényege az, hogy a processzor teljes címzési tartománya tulajdonképpen egy háttértároló-területet jelent. Ezen virtuális tárolónak csak egy része helyezkedik el az operatív tárolóban, ahonnan a felhasználó futtat. A felhasználó nem látja a

szükséges háttértár–operatív tár átviteleket, ezért úgy látja, mintha a processzor teljes címzési tartománya rendelkezésére állna. Egy tárkezelő egység *(MMU = Memory Management Unit)* nyilvántartja, hogy az információ hol helyezkedik el, és a virtuális címet az operatív tároló fizikai címévé alakítja át, ha a keresett információ a tárolóban van. Ha a keresett információ nincs az operatív tárolóban, akkor a behozatalát kezdeményezi a háttértárolóról. Ezt az átvitelt az operációs rendszer virtuális tárkezelő része hajtja végre.

A virtuális memória tehát, az operációs rendszer vagy a számítógép hardvere által nyújtott szolgáltatás (legtöbbször a kettő szoros együttműködése által), amit általában egy külső tárolóterület (merevlemez) igénybevételével, a futó program(ok) számára transzparens módon biztosítja, hogy a program végrehajtáskor a központi vagy operatív memória fizikai korlátai észrevétlenek legyenek.

A virtuális memória megvalósításának elve:

- Jelöljünk ki a háttértárolón a főtár memóriakapacitásánál jóval nagyobb tárolóterületet úgy, hogy az egyidőben aktív programfolyamatokhoz tartozó programok és adatállományok ezen elférjenek.
- Nevezzük ezt látszólagos, azaz virtuális tárterületnek.
- A háttértárolón kijelölt virtuális tárolót osszuk fel blokkokra, melynek méretét a lokalitás elvének figyelembevételével határozzuk meg:

VIRTUÁLIS TÁR

0. blokk	1. blokk	2. blokk	….	n-ik blokk

3-18. ábra: ***A virtuális tár felosztása blokkokra***

A virtuális tárolóban 1 bájtot virtuális címekkel lehet megcímezni, mely két alapvető részből áll: egyrészt megadjuk, hogy a keresett bájt a virtuális tár hányadik blokkjában található, továbbá azt, hogy az adott blokk kezdőcímétől számítva hányadik bájt (relatív cím):

virtuális cím	=	blokksorszám	+	relatív cím

3-19. ábra: ***Virtuális címszámítás***

Látható, hogy a virtuális tárkezelés alapelve és a cache működési elve nagyon hasonló, de lényeges különbségek is vannak:

A cache és a főtár viszonya

- A különböző cache blokkok leggyakrabban azonos programokhoz tartoznak.
- A cache miss-t a hardver kezeli.
- A cache nagysága (elvileg) független a processzor címterétől.
- A cache kizárólagosan a tárkezelés céljaira szolgál, a programok nem „látják”.

A főtár és a virtuális tár viszonya

- Egy programfolyamat „egyidejűleg” más programfolyamatokkal együtt fut, mindegyik folyamathoz önálló virtuális tárterület tartozik.
- Ha egy blokk nincs a főtárban (blokkhiba) akkor ezt az operációs rendszer kezeli.
- A címtartományt meghatározza a virtuális tár nagysága.
- A mágneslemezen a virtuális táron kívül más adatállományok is megtalálhatók.

A virtuális tárkezelés alapkérdései a következők:

Mekkora legyen egy blokk mérete? Viszonylag nagy tárterület célszerű blokknak kijelölni, annak érdekében, hogy gyakran ne legyen szükség a blokk cseréjére, mivel ehhez relatíve hosszú beolvasási idő szükséges.

Hova lehet egy blokkot beírni a főtárban? Tetszőleges helyre.

Ha be kell másolni egy blokkot a főtárba, akkor melyik főtár-blokkot írhatjuk felül? Ezt az operációs rendszer dönti el algoritmussal, leggyakoribb LRU (Least Recently Used) stratégia alkalmazása, mely a processzor (programok) által legkevésbé használt blokk kiválasztását eredményezi.

Ha megváltozik egy blokk a főtárban, mikor írjuk be a virtuális tárba? A write back eljárás alkalmazása a célszerű, azaz csak blokkcserénél aktualizáljuk a virtuális tárat, mert az aktualizálás jelentős időt igényel.

A virtuális tárkezelésre jellemző adat az alig 0,00001 – 0,001%-os hibaarány (Miss-Rate), azaz ha a keresett adatot tartalmazó blokk nincs benn a főtárban

A programfolyamatok és a virtuális tár viszonya:

A számítógépen aktuálisan futó programfolyamatok utasításai a virtuális címeket, mint logikai címeket tartalmazzák. A virtuális tár blokkjai akkor kerülnek bemásolásra a főtárba, ha valamilyen programutasításban hivatkozás történik az adott blokkban található címre és az nem található meg még a központi memóriában.

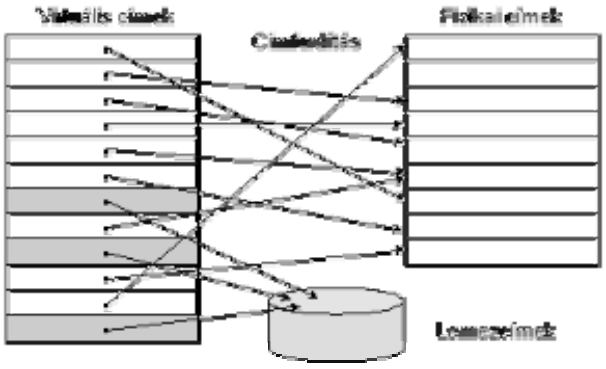
Az előbbiekből következik, hogy a programok a virtuális tárat úgy látják, mintha az a központi tár lenne. A virtuális címzéssel elvileg megcímezhető memóriaterületet virtuális címtartománynak nevezzük. Hangsúlyozni kell, hogy ez az elméletileg lehetséges virtuális címek összessége, a gyakorlatban használható virtuális címek mennyisége attól függ, hogy a háttértárolón mekkora területet jelölünk ki virtuális tárnak.

A virtuális cím leképezése fizikai címmé:

Természetesen a processzornak a műveletvégrehajtás során a főtár valódi, vagy másképpen nevezve fizikai címeire van szüksége, tehát a virtuális címeket át kell alakítani fizikai címekké. Ehhez két dolog szükséges:

- a blokkok háttértárolón található címét, a fizikai memóriába bemásolt blokkok sorszámát, a fizikai kezdőcímét stb. megfelelő táblázatokban nyilván kell tartani,
- egy olyan program vagy hardver egység, mely a memóriába bekerült blokkok adatait tartalmazó táblázatok alapján elvégzi a virtuális cím fizikai címmé történő átalakítását.

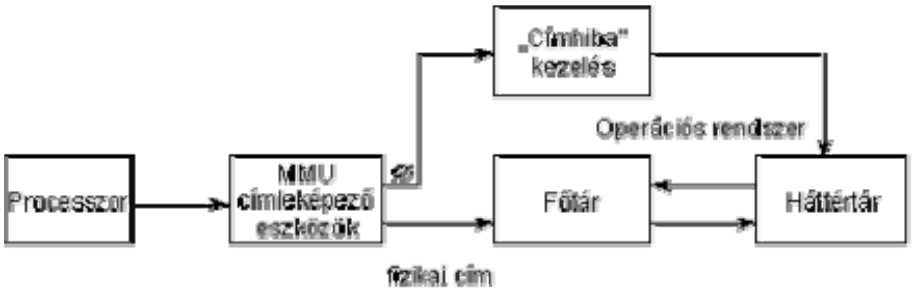
A virtuális címek fizikai címmé történő leképezését az első virtuális tárkezelést alkalmazó számítógépekben még az operációs rendszer végezte. Ma már kizárólagosan az jellemző, hogy ezt a feladatot egy megfelelő címleképezési áramköröket tartalmazó hardver egység, az MMU (Memory Management Unit) végzi el. A virtuális címek és a fizikai címek viszonyát mutatja be a 3.20 ábra.



3-20. ábra: ***Virtuális és fizikai címek viszonya***

3-21. ábra: ***Virtuális és fizikai cím megfeleltetés***

Az MMU a címleképezés mellett azt is figyeli, hogy a végrehajtandó utasításban olyan virtuális cím van-e, mely olyan blokkra hivatkozik, mely még nem került be a főtárba. Amennyiben ilyet talál, akkor ez kivételt okoz, és az operációs rendszer megszakításkezelő rutinja meghatározza azt a főtárba már bemásolt blokkot, melyre várhatóan legkevésbé lesz szükség az elkövetkezendőkben, és ez kiírásra kerül a háttértáron lévő virtuális tárba, helyére pedig bemásolásra kerül az a blokk, melynek adataira az aktuális utasítás végrehajtásához szükség van. A folyamat elvét mutatja be a következő ábra:

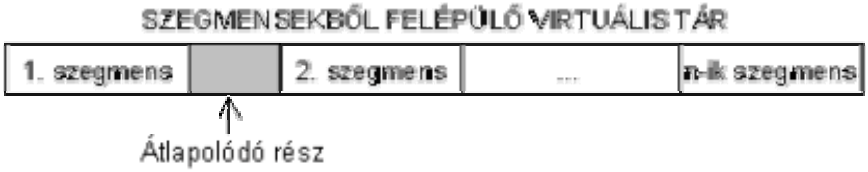


3-22. ábra: ***Az MMU működése***

A virtuális tárkezelésnek két alapvető formája van, a szegmentálás és a lapozás, ekkor virtuális tár blokkjait szegmensnek, illetve lapnak nevezzük.

3.4.2. Szegmentálás

Ha a virtuális tár olyan logikai blokkokból áll, melyeknek mérete nem rögzített, akkor ezeket a blokkokat szegmenseknek, a virtuális tárkezelésnek ezt a formáját pedig szegmentálásnak nevezzük. A szegmensek átlapolódóan is megadhatók, azaz ugyanaz az adat két különböző szegmensen belül is megcímezhető:



3-23. ábra: ***Szegmentálás***

Az átlapolódást a gyakorlatban legtöbbször úgy hasznosítják, hogy több programfolyamat közösen használ szegmenseket.

A szegmentálásnál a logikai cím a szegmens sorszámát és a megcímzett bájtnak a szegmenskezdettől való relatív címét tartalmazza. A szegmens fizikai kezdőcímét a szegmenstáblázat tartalmazza, ebből a szegmens sorszáma – melyet szelektornak is neveznek – alapján lehet kikeresni a konkrét szegmens főtárbeli kezdőcímét. Ezt követően a fizikai cím a következő összefüggéssel meghatározható:

fizikai cím = szegmens fizikai kezdőcíme + relatív cím

A szegmentált virtuális tárkezelésnél az átlapolódás és a különböző blokkméret miatt a szegmenseknek a központi tárból történő kivitele, illetve a központi tárba történő bemásolása során a központi tárban igen sok üres hely keletkezhet. Ezt fregmentációnak hívjuk. Emiatt időközönként szükség lehet a memória oly módon történő átrendezésére, hogy összefüggő lefoglalt, illetve szabad területek jöjjenek létre. Ezt a műveletet szemétgyűjtésnek (garbage collection) is szokták nevezni.

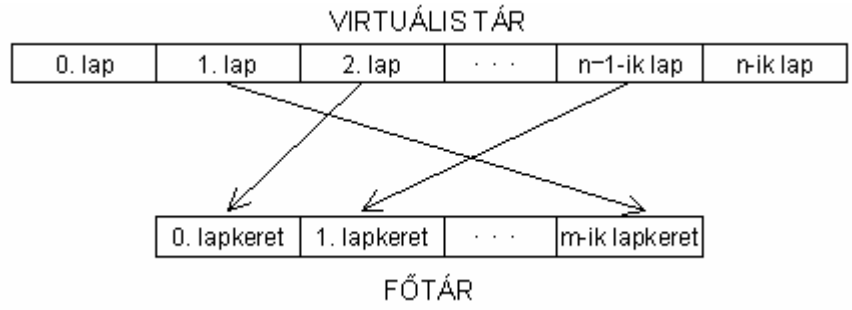
A szegmensek betöltésére a virtuális tárból a főtárba többfajta stratégia is elképzelhető. Így például:

- első szabad helyre, azaz ebben az esetben a memória kezdetétől kezdve megvizsgálásra kerül, hogy hol van az első olyan szabad tárterület, melynek mérete lehetővé teszi a szegmens betöltését,
- következő szabad helyre, azaz utolsóként betöltött szegmenstől kezdve vizsgáljuk az első olyan szabad helyet, ahova a szegmens betölthető,
- az ún. „legjobb helyre”, azaz az összes olyan szabad tárterület közül, melyekben a betöltendő szegmens elfér, azt választjuk ki, melynél a betöltést követően a legkevesebb szabad tárterület marad,
- az ún. „legrosszabb” helyre, amikor az a cél, hogy a betöltést követően a szegmens mellett a lehető legtöbb tárterület maradjon szabadon.

Összefoglalva megállapítható, hogy a szegmentált virtuális tárkezelés előnye a rugalmasság (a változtatható blokkméretek miatt), az osztott felhasználás lehetősége az átlapolódó szegmensekkel. Hátránya, hogy a nagyméretű szegmensek cseréje ronthatja a hatékonyságot, és a memória teljes körű kihasználtsága sem biztosított.

3.4.3. Lapozás

Ha a virtuális tár rögzített méretű nem átlapolható blokkokból áll, akkor ezeket lapoknak nevezzük, a virtuális tárkezelésnek ezt a formáját pedig lapozásnak. A főtár a lapmérettel megegyező nagyságú részekre van felosztva, ezeket lapkereteknek (frame) nevezik:

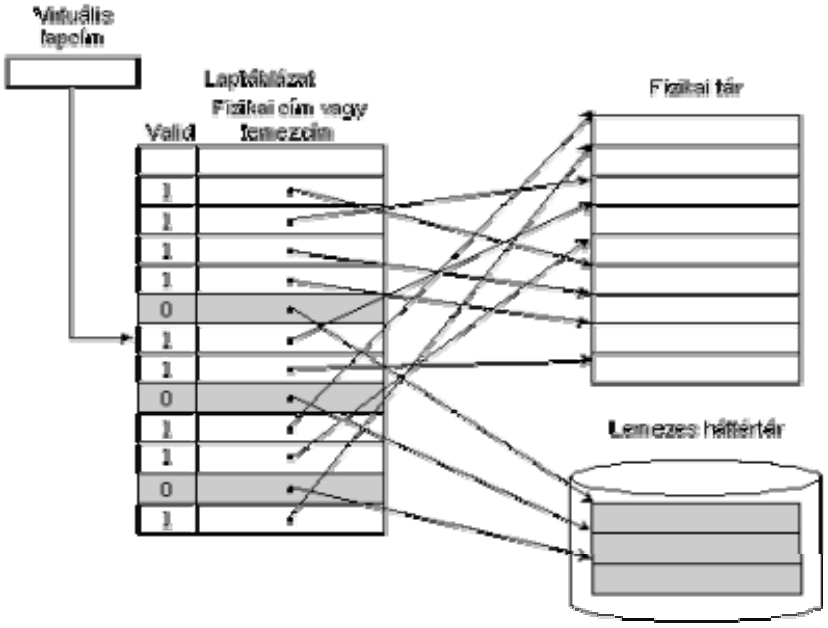


3-24. ábra: ***Lapozás***

A lapok mérete különböző architektúrák esetén eltérő lehet, a legelterjedtebb a 4KB lapméret, de például a Pentium processzor kezeli a 4 Mbájtos lapokat is.

A lapozásnál a virtuális cím hasonlóan épül fel, mint azt a szegmentálásnál láttuk, azaz a lap sorszámát és a megcímzett bájtnak a lap kezdetétől számított relatív címét tartalmazza.

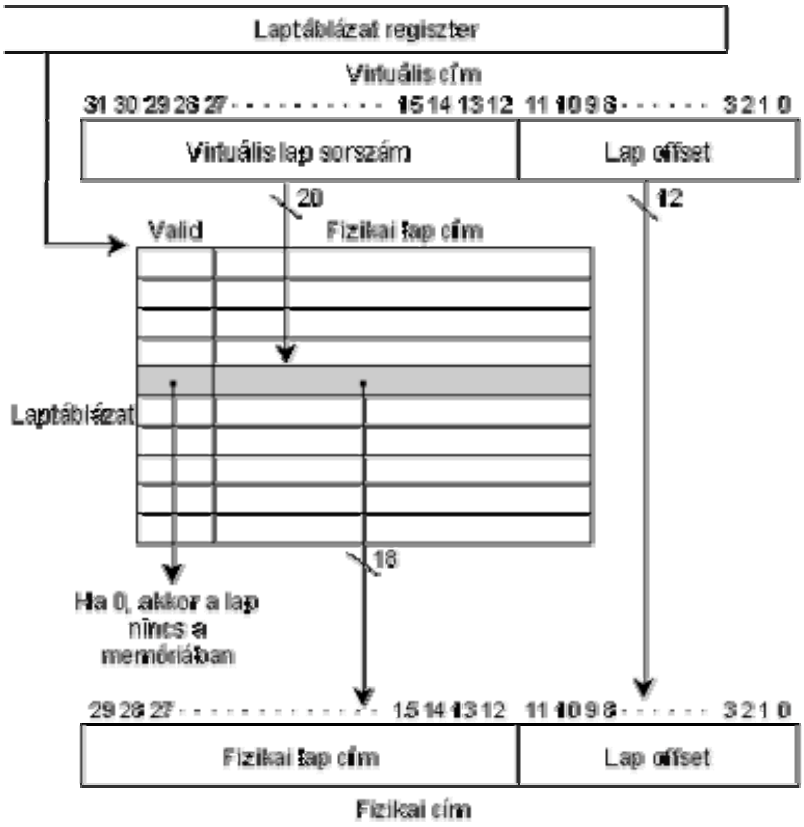
A központi memóriába beolvasott lap fizikai kezdőcímét a főtárban – azaz annak a lapkeretnek a címét, ahová a lap elhelyezésre került – a laptáblázatok tartalmazzák (*3.25 ábra*). Általában szintén a laptáblázat tárolja a főtárban nem szereplő lapok lemezcímét is.



3-25. ábra: ***Laptáblázatok***

Minden programfolyamatnak (processznek) saját laptáblázata van, ennek kezdőcíme sokszor egy regiszterben található. A laptáblázatok segítségével a virtuális cím fizikai címmé történő átszámítása a 3.26 ábra eljárása szerint történik.

Ha egy programfolyamat egy utasítása olyan virtuális címre hivatkozik, melynek megfelelő lap nincs a főtárban, akkor ez „laphiba” kivételt okoz. (A laphibát az MMU azáltal ismerni fel, hogy a laptáblázatban a lap érvényességi bitje 0 értékű.) Ez egy megszakítást eredményez, és a vezérlést megkapja az operációs rendszer laphiba kezelő rutinja, mely a hivatkozott lapot betölti a főtárba.



3-26. ábra: ***Címszámítás ha a lap nincs a főtárban***

Az operációs rendszer úgy szabadít fel operatív memóriát az éppen futó program számára, hogy a memóriában tárolt, de éppen nem használt blokkokat (*lapokat*) kiírja a külső tárolóra, amikor pedig ismét szükség van rájuk, visszaolvassa őket. Mivel a merevlemez sebessége töredéke a memória sebességének, nagyon sok múlik azon, hogy a virtuálismemória-kezelő milyen stratégiát alkalmaz az operatív memóriából kimozgatandó lapok kiválasztásakor.

Lapkiosztási elvek: A folyamatok számára biztosítható lapok számának felső korlátja a fizikai tár mérete. Van azonban alsó korlát is, melyet az adott processzor utasításkészletének ismeretében határozhatunk meg. Vannak ugyanis olyan utasítások, amelyek végrehajtása során több memóriacímet is használunk. Legrosszabb esetben ezek mindegyike különböző lapon található meg, azaz ha a folyamatnak nincs legalább ennyi lapja, akkor az ilyen típusú utasításokat nem tudja végrehajtani. Az alsó és felső korlát között az operáció rendszer dönt valamilyen elv alapján.

- **Egyenletes elosztás**: Legegyszerűbb esetben a rendelkezésre álló lapokat egyenletesen osztjuk el a folyamatok között, azaz minden folyamat ugyanannyi kerettel gazdálkodhat.
- **Arányos elosztás:** Jobb, igazságosabb lapkiosztást tesz lehetővé, ha a folyamatok virtuálismemória-igényeinek figyelembe vételével döntünk. Azaz egy folyamat minél nagyobb virtuálismemória-területet használ, annál több keretet kap. Ez esetben arányos lapkiosztásról beszélünk.
- **Prioritásos elosztás**: magasabb rendű folyamat extra előnyökkel rendelkezik, tehát a hasonló igényű de alacsonyabb prioritású folyamatoknál több lapot kap, és az azonos fontosságú folyamatokat természetesen arányosan, vagy egyenletesen kezeli

- **Lokális elosztásról** beszélünk: ha a rendelkezésre álló lapok száma egy folyamat számára a futás során állandó,
- **Globális lapkiosztási stratégia**: az operációs rendszer úgy rendelkezik, hogy nem oszt ki minden lapot, csak a minimálisan szükségeseket, a fennmaradó szabad lapokból a folyamatok dinamikus igényeit elégíti ki. Előnye az, hogy rugalmasabb. Hátránya viszont, hogy **vergődés**hez (trashing) vezethet. Ez akkor áll elő ha egy folyamat bármely okból olyan kevés laphoz jut, hogy csaknem mindig laphibát okoz, fut ugyan, de futása akár tízezerszer is lassabb lehet, mint normális esetben.

Lapcsere stratégiák

A lapcsere algoritmusok minősítésére mesterséges vagy tapasztalati úton laphivatkozási sorozatokat, ún. referencia stringeket alkalmaznak. A szimuláció eredménye a különböző módszerek hatékonyságára jellemző laphiba szám.

- **Az optimális stratégia** (OPT): Azt a lapot kell lecserélni, amelyre a legkésőbb lesz szükség, hiszen így a bent maradó lapokkal a lehető legtovább ki tudjuk elégíteni a lapok iránti igényeket laphiba nélkül.A lapcserénél előre kell tudni, hogy a későbbiekben milyen sorrendben fogunk a lapokra hivatkozni, így ez a gyakorlatban megvalósíthatatlan.
- **Előbb jött előbb megy** (First In First Out – FIFO): Azt a lapot kell lecserélni, amely legrégebben van bent a memóriában. Ráadásul ez viszonylag egyszerűen megvalósítható, hiszen nem kell hozzá más, mint egy egyszerű várakozási sor, amely olyan hosszú, ahány kerete az adott folyamatnak van. Ezt az új. FIFO sort használjuk az adminisztrációhoz. Minden frissen behozott lap sorszáma bekerül a FIFO sor végére, ezáltal a sorban már bent lévő lapsorszámok eggyel előre lének és a sor másik végén „kipotyog” a legrégebbi lap sorszáma. A FIFO elv az operációs rendszer legfontosabb lapjait állandóan „kilapozza”, és ez sokszor nagyobb probléma, mint a kis hatékonysága.
- **Legrégebben használt (Last Recently Used – LRU)**: Azt a lapot kell lecserélni, amelyre legrégebben hivatkozott a folyamat. A rendszer hardver támogatást igényel, mert csak ezzel választható meg hatékonyan, ugyanis csak hardver megoldás képes arra, hogy elviselhető időveszteséggel minden egyes memóriahivatkozásnál módosítson egy, a lapok felhasználási sorrendjét tartalmazó, FIFO jellegű listát, vagy a laptábla erre szolgáló mezőjébe bejegyezze a hivatkozás időpontját és laphibánál ezek közül megkeresse a legkisebb értéket. A módszer tehát viszonylag kevés laphibát eredményez, de cserébe az adminisztrációs terhek szinte elviselhetetlenül növekedtek.
- **Második esély** (Second Chance – SC): Az eljárás FIFO elven alapul egy kis kiegészítéssel. Minden laphoz – a laptáblába - elhelyezünk egy ún. „hivatkozás” bitet, amelyet, ha a lapot használjuk, automatikusan 1-esbe állítunk. Laphiba esetén megkeressük azt a lapot, amely a FIFO sor elején áll. Ha ennek a lapnak a hivatkozás bitje 1, akkor mégse őt válasszuk, hanem betesszük a FIFO végébe, de 0-s hivatkozási bittel. Ha a FIFO elejére olyan lap kerül, melynek hivatkozási bitje 0, akkor lecseréljük.
- **Mostanában nem használt** (NUR) lapok cseréjét javasolja az LRU módszer enyhített, könnyebben megvalósítható változata. A mostanában kifejezés azt takarja, hogy az előző lapcsere óta használták vagy nem használták a kérdéses lapot. E módszernél általában a nullás jelzőbitű lapokból véletlenszerűen választanak.

Ellenőrző kérdések:

1. Mit értünk operatív memória alatt?
2. Miért van szükség, és hogyan történik a memória címzése?
3. Ismertesse a közvetlen címzési módot!
4. Mi a különbség az abszolút és a relatív címzés között?
5. Ismertesse a közvetett címzési módot!
6. Ismertesse az indexelt címzési módot!
7. Mi a feltétele a memóriák hibajelzésének?
8. Hogy működik a memóriák hibajavítása?
9. Ismertesse a RAM, ROM fogalmakat, és a ROM-ok fajtáit!
10. Mi a különbség az SRAM és a DRAM között?
11. Ismertesse a Flash memóriák lényegét!
12. Miért van szükség cache tárolókra?
13. Mit értünk program-lokalitáson?
14. Mi az időbeli- és a helyi lokalitás?
15. Mit értünk asszociatív tárolón?
16. Milyen helyettesítési stratégiákat ismer a cache-be írásnál?
17. Milyen módszereket ismer a cache-ben megváltozott adatok főtárba való visszaírására?
18. Mit jelent a MESI?
19. Miért van szükség rájuk, és hogy működnek a lemezgyorsító tárak?
20. Ismertesse a számítógépekben alkalmazott memóriahierarchiát!
21. Mi a virtuális tárkezelés lényege?
22. Mi az MMU és mi a feladata?
23. Mi a szegmens és mi a lap?
24. Milyen lapkiosztási elveket ismer?
25. Milyen lapcsre stratégiákat ismer?

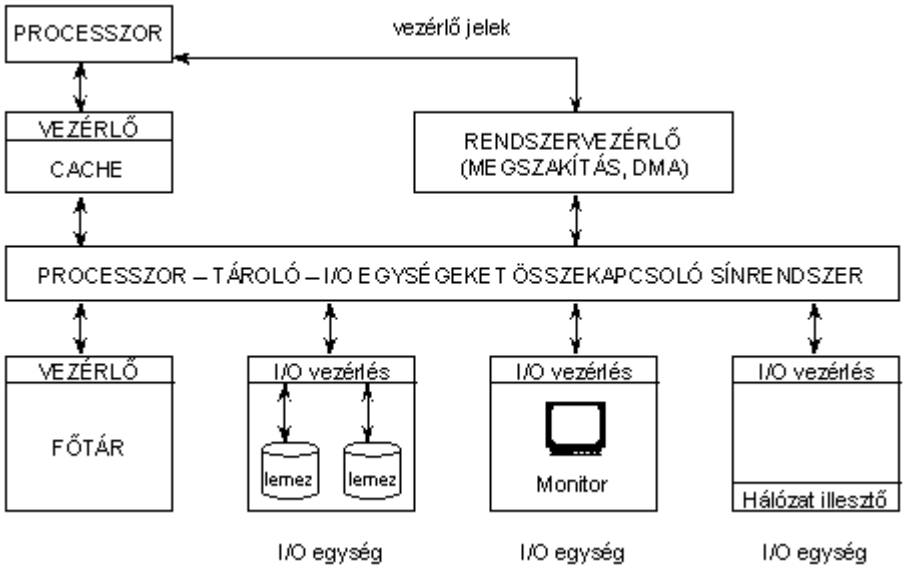
4. Mikroprocesszor alapú számítógéprendszer

4.1. A számítógép legfontosabb részegységei, és működésük

A számítógép meghatározott feladatköröket ellátó részegységekből épül fel, ezek a részegységek önálló vezérléssel és speciális vezérlő és állapotinformációt tartalmazó, illetve adatpuffer regiszterekkel rendelkeznek. Funkcionálisan a következő legfontosabb részegységeket különböztethetjük meg (4.1 ábra):

- a processzor,
- az L2 cache,
- a főtár,
- a megszakításrendszer,
- a közvetlen memória-hozzáférési (DMA) rendszer,
- a busz vagy sínrendszer,
- a perifériák,
- a háttértárak,

Ezekből a részegységekből a számítógép architektúrájától függően egy konkrét gép központi egységébe egy vagy több is beépítésre kerülhet:



4-1. ábra: A számítógép főbb egységeinek kapcsolata sínrendszerrel

A részegységek lehetnek aktív eszközök, illetve működhetnek passzív eszközként. Az aktív eszköz a kezdeményező, a passzív eszköz csak fogadja és végrehajtja az aktív eszköztől származó vezérléseket.

A processzorral, a cache memóriával és a főtárral már az előző fejezetekben megismerkedhettünk, a DMA-val a 4.1.3 fejezetben, a háttértárakkal pedig az 5. fejezetben fogunk foglalkozni.

4.1.1. A megszakítási rendszer

A számítógépnek rugalmasan reagálnia kell a külvilág eseményeire. Erre a célra szolgál a számítógép megszakítási rendszere. Ugyanakkor a számítógép különböző részegységei működésének összehangolásában is fontos szerepe van. Ha nem lenne megszakítási rendszer, akkor például a processzornak folyamatosan ellenőrizni kellene, hogy a felhasználó nem nyomott-e le egy billentyűt vagy más, a program működését kívülről

befolyásoló esemény nem következett-e be. Ez pedig igen nagy pazarlás lenne a processzor teljesítményével, hiszen amíg a külső esemény bekövetkezését figyelő program futna, más hasznos munkát nem tudna végezni. A korszerű nagyteljesítményű számítógép-rendszerektől pedig már nem csupán egy egyszerű reagálást kívánunk meg, hanem ezen túlmenően, hogy folyamatosan változó körülmények között is optimálisan dolgozzanak. Az optimális működés jellegét a rendszer sajátosságai döntik el.

A megszakítási rendszereket elsődlegesen az I/O műveletek és a velük átlapoltan végrehajtott számítási műveltеknek az összehangolására hozták létre. A megszakítás célja az volt, hogy egy I/O tevékenység beindítása után a processzornak ne kelljen várakoznia az I/O művelet befejeződésére, hanem ezalatt más feladat, vagy feladatrész végrehajtásába foghasson. Viszont az I/O tevékenység befejeződéséről valahogy értesíteni kell a processzort. Ezért az I/O tevékenység befejeződése egy megszakítás-kérő jelet generál, amely az éppen végrehajtás alatt álló utasítás befejeződése után, a végrehajtásra következő utasítás megkezdése helyett megszakítást (vezérlés-átadást) hoz létre egy, a megszakítás konkrét okától (forrásától) függő címre, a szükséges megszakító rutinra.

A megszakítás bekövetkezésekor tehát az éppen futó programról a vezérlés ideiglenesen átadódik egy másik program számára, amely kiszolgálja a bekövetkezett eseményt. A megszakítást kiszolgáló program lefutása után pedig a megszakított program végrehajtása a következő utasításától folytatódik. A megszakítás tulajdonképpen egy már futó folyamat felfüggesztése egy másik futtatni kívánt folyamat érdekében

Megszakítások és kivételek kezelése

A ma használatos processzoroknál a programvégrehajtást időszakosan felfüggesztő okokat két osztályba szokták sorolni. Ezek:

- külső eredetű **megszakítások** (interrupt) amikor a program utasításainak átmeneti felfüggesztését a processzortól független, külső esemény idézi elő (pl. DMA megszakításkérés),
- az utasítások szabályszerű végrehajtását megakadályozó **kivételek** (exception), amikor a programot a programutasítás végrehajtása alatt a processzoron belül fellépő esemény miatt kell megszakítani (pl. paritáshiba, osztási hiba, túlcsordulás, BIOS rutin meghívása programvezérlet megszakítással = INT).

A külső események által okozott megszakítások esetén a processzor az éppen aktuális programutasítás végrehajtását szabályszerűen befejezi, és ezt követően kezd csak foglalkozni a megszakítás kérelem kiszolgálásával.

Mivel a megszakítások és kivételek fellépése a programvégrehajtás szempontjából véletlenszerű, ezért előfordulhat, hogy egy megszakítás kiszolgálásának ideje alatt szintén bekövetkezik egy megszakítást igénylő esemény. Ebből a szempontból a program futását átmenetileg felfüggesztő események két kategóriába sorolhatók:

- olyan események, melyek megszakítási igénye átmenetileg letiltható. Ezeket **maszkolható** megszakítási kérelmeknek nevezik, mivel engedélyezésük vagy tiltásuk egy regiszter megfelelő bitjének beállításával történik;
- olyan események, melyek megszakítási igénye nem tiltható le és minden esetben ki kell szolgálni. Ezeket **nem maszkolható** megszakításoknak, azaz NMI = Non Maskable Interrupt-oknak nevezik. (Ilyenek például a súlyos hardver hibák.)

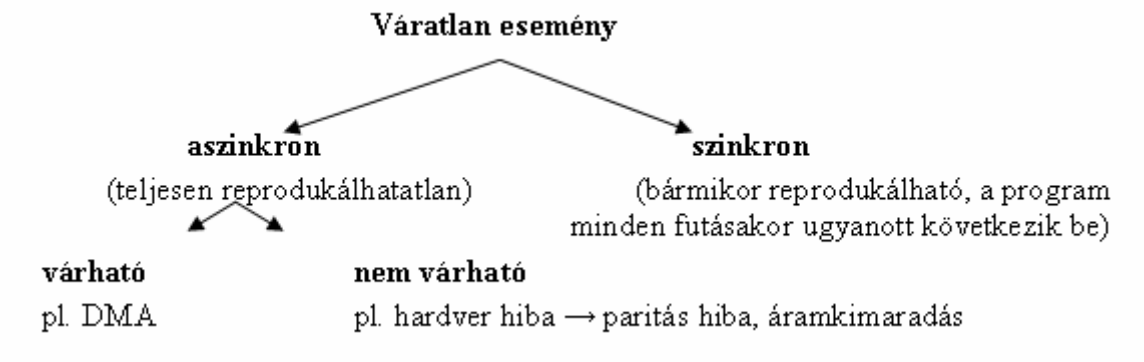
A CPU utasítás-készletében általában külön olyan utasítás van, amely lehetővé teszi a megszakítás-kérések tiltását vagy maszkolását. Az ilyen utasításokkal a programozó kikapcsolhatja az egyes megszakítás-kérési vonalakat, melyek eredményeképpen a CPU bizonyos megszakításokat nem vesz figyelembe.

A nem maszkolható megszakítási igény viszont azonnal kiszolgálandó. NMI-hez általában a rendszerszinten kritikus hibákat szokták kapcsolni, például a RAM paritáshibát vagy a tápellátás hibáját. Mikor ez bekövetkezik, a számítógép még néhány másodpercig működőképes, mivel minden tápegységben van egy kevés energiatároló kapacitás. Az NMI azonnali kiszolgálása révén a számítógép normál módon lekapcsolódik, tehát a számítógép újraindítása könnyen lehetséges.

A megszakításoknak tehát az a lényege, hogy csak akkor foglalják le a processzor figyelmét, ha arra valóban szükség van. Miután a processzor munkaidejének csak töredékét teszik ki azok az esetek, amikor külső kapcsolatokkal kell foglalkoznia, a megszakítások rendszere felszabadítja a processzort a külső érintkezési pontok időt rabló és fölösleges figyelésétől. Másrészt viszont, amikor külső kapcsolatot kell lekezelni, vagyis megszakító jelet kap, késlekedés nélkül, azonnal reagálhat.

A megszakítások csoportosítása

Szinkron—aszinkron megszakítások: Azokat a megszakításokat, amelyek a programnak ugyanazon adatokkal való végrehajtása során mindig ugyanott lépnek fel, szinkron megszakításoknak nevezzük. Ilyen például az integer túlcsordulás. Az aszinkron események viszont véletlenszerűen következnek be. Például az I/O egység által kért megszakítások, a hardver-hibák (4.2 ábra).



4-2. ábra: ***A megszakítások okai vagy forrásai***

Az utasítások végrehajtása között illetve közben fellépő megszakítások

Az utasítások végrehajtása között fellépő megszakítások az éppen végrehajtott utasítás eredményeképpen következnek be (például túlcsordulás, page fault, tárvédelmi hiba). A kezelése rögtön az utasítás végrehajtása után elindulhat, s a programfutás eredményessége aztán a kezelés eredményétől függ;

Az utasítások végrehajtása közben fellépő megszakítások valamely utasítás végrehajtása alatt (tehát nem az utasítás-végrehajtási ciklussal szinkronban) merülnek fel. Ilyenek például a hardver-megszakítások. Ekkor az esetek többségében először befejezésre kerül az éppen végrehajtás alatt álló utasítás, s csak utána kezdődik meg a megszakítás kiszolgálása.

A felhasználó által explicit kért és nem kért megszakítások:

- A felhasználó által explicit kért megszakítás például az operációs rendszer szolgáltatásának meghívása, a nyomkövetés vagy az utasítás töréspont. Véletlenszerűen lép fel.
- A felhasználó által nem kért megszakítás például az integer túlcsordulás, az I/O egység megszakítás, a hardver hiba.

A megszakításkérő egység felismerése

Mivel egy rendszerben több megszakítást kérő eszköz is lehet, ezért először azonosítani kell a megszakítást kérő eszközt.

Lekérdezéses (polling)

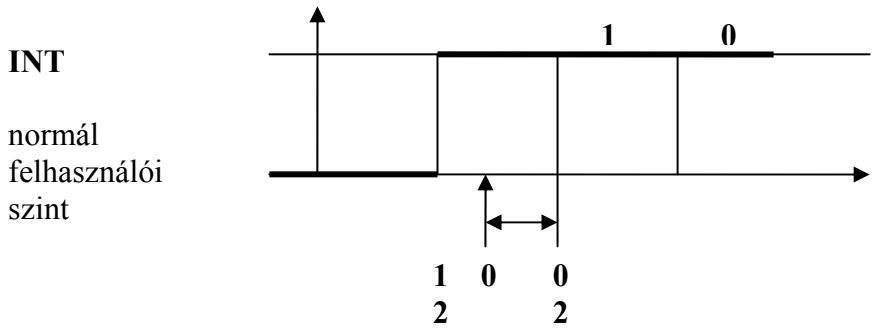
Sorrendben lekérdezi az egyes egységeket, hogy kértek-e megszakítást, s az első olyan egység kérését, amely megszakítást kért, kiszolgálja. Az egységek lekérdezése történhet hardver úton (hardware polling, vagy felfűzéses daisy chain) vagy szoftver úton (software polling);

Vektoros (vectored)

Ez lehetővé teszi, hogy a megszakítást kérő egység azonosítsa magát, ami kiküszöböli az időt rabló lekérdezést. A nevét onnan kapta, hogy a megszakítás-kérésen kívül egy un. megszakítási vektornak nevezett adatot is küld. A megszakítási vektor tartalma alapján meghatározható a megszakítási igénnyel jelentkező periféria továbbá az is, hogy milyen kiszolgálást kér. (A konkrét megoldás architektúránként változó.)

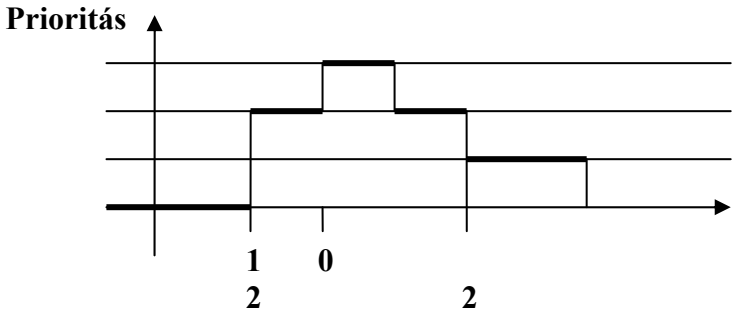
Megszakítási rendszerek szintek szerint

- **Egyszintű**: Nincs lehetőség a kiszolgáló rutin felfüggesztésére egy újabb megszakítási kérelem által. A kiválasztó logika a kiszolgálás közben érkezett megszakítások közül a legnagyobb prioritású engedélyezett megszakítás-kérést fogadja el. A 4.3 ábrán látható, hogy a 2-es szinten futót megszakíthatja az 1-es megszakítási forrás szerinti kérés, azonban ennek feldolgozása alatt hiába érkezik nagyobb prioritású (0-s szintű).



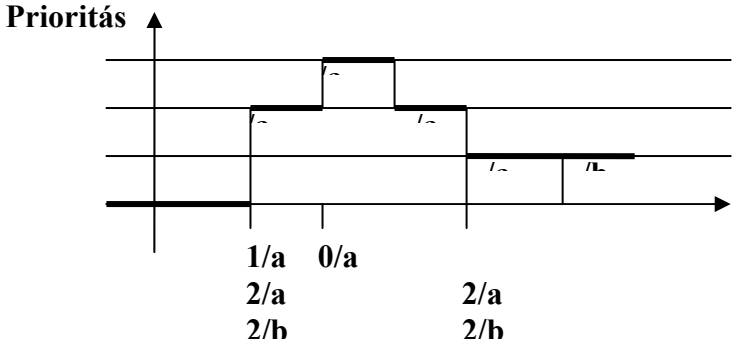
4-3. ábra: ***Egyszintű megszakítási rendszer***

- **Többszintű**: keresi a pillanatnyilag futó CPU-szintnél magasabb prioritás-szintű engedélyezett megszakítás-kéréseket. Kiválasztja a legalacsonyabb prioritás-szintűt. A program állapotszó (PSW, Program Status Word) csere esetén ez oly módon zajlik le, hogy a megszakított szint PSW-je Old PSW-ként tárolódik, a másik rekesz tartalma pedig New PSW-ként betöltődik a programállapot-regiszterbe. Az elfogadott megszakítás-kérés nyugtázódik. Ha nem talál az utolsó szintnél magasabb prioritású engedélyezett kérést, akkor megengedi a legutolsó New PSW-ben megjelölt utasítás végrehajtását. A 4.4 ábrán látható, hogy 2-es szinten futót, megszakíthat egy nála nagyobb prioritású kérés (1), és ezt szintén egy még nagyobb (0).



4-4. ábra: ***Többszintű megszakítási rendszer***

- **Kompromisszum**: az előző kettő ötvözése, azaz szinteket rendelnek a megszakítások egy-egy csoportjához



4-5. ábra: Szinten belül egyszintű, szintek között többszintű megszakítási rendszer

Megszakítások a gyakorlatban

Az Intel processzoroknak mindössze 2 db megszakítási vonaluk van, az egyik a Non Maskable Interrupt, az NMI, a másik vonalon pedig osztozik az összes többi megszakítás, a megszakítás-vezérlőn keresztül. A PC-kbe tervezett megszakítás-vezérlő 8 db megszakítást képes kiszolgálni, mégpedig 8 prioritási szinten. Az IRQ0-tól IRQ7-ig sorszámozott megszakítások esetén az IRQ0 prioritása a legmagasabb, az IRQ7-é pedig a legalacsonyabb.

Az AT megjelenésekor a 8 prioritási szint már kevésnek bizonyult, ezért az IBM a bővítést a kompatibilitás figyelembe vétele mellet, úgy oldotta meg, hogy az XT-ben alkalmazott megszakítás-vezérlő IRQ2-jéhez hozzákapcsolt egy újabb megszakításvezérlőt. Így a 2-es szint (IRQ2) kivételével minden eszköz megszakítási szintje a helyén maradhatott, és az AT-nál alkalmazott gyorsabb perifériák megszakításai az új megszakításkezelőhöz kapcsolódnak, úgy, hogy prioritásban megelőzik a 3-as szinten és fölötte levőket. A meglévő vezérlő kimenete az (NMI-t nem számítva) egyetlen megszakítási vezérlővonalon keresztül pedig csatlakozik a CPU-hoz. A kaszkádosítás eredményeképpen 2x8-1 = 15 megszakítást tudunk kezelni (az összefűzésre elhasználunk egy bemenetet, ezért nem 16).

A PCI-sín bevezetésével megváltoztaták a megszakítási struktúrát, és saját megszakítási vezérlést építettek be a PCI bridge-be. A PCI rendszernek négy megszakítási vezérlővonala van, melyek mindegyike egy-egy bővítősínt szolgálhat ki. A PCI koncepció szerint az ISA bővítősín csak a PCI bridge-n keresztül kommunikálhat a mikroprocesszorral. A PCI tervezői kialakítottak egy úgynevezett sorosított megszakítás-támogató rendszert (Serialized IRQ Support for PCI Systems), mely egy úgynevezett IRQSER jelen alapul. Ebben egymás után jelölik, hogy mely megszakításhoz tartozik és melyikhez nem tartozik megszakítás-kérés. Ezt az információt eljuttatja a mikroprocesszor chipsetében lévő megszakításvezérlőhöz. A vezérlőt elérve a megszakítások kezelése már hagyományos módon történik.

Mind a 8, mind pedig a 15 megszakításos esetre az IBM adott egy hozzárendelési táblát, hogy mely egységet mely megszakításhoz kössünk. A mai modern PC-k és operációs rendszerek világában a Plug and Play szabvány mellett a következő öt megszakítás (4.1 táblázat) hozzárendelése kötött csupán. A táblázatban a szabadon felhasználható megszakítási szintek nem szerepelnek.

4-1. Táblázat: ***Az IBM PC-k kötött megszakításai***

A megszakítás száma	Funkció
IRQ0	timer output 0
IRQ1	billentyűzet
IRQ2	kaszkád (a 15-be nem számítjuk be)
IRQ8	real time clock
IRQ9	video, hálózati adapter
IRQ13	lebegőpontos processzor

A maszkolható megszakítások közül a legmagasabb prioritást a Timernek kell adni, hogy az órajelet minden körülmények között biztosítsa. Mivel személyi számítógépről van szó, ennek illik azonnal reagálni a billentyűzet-leütésre, ezért rendelték a második legmagasabb prioritási szinthez a billentyűzetet.

4.1.2. Megszakítások és kivételek kiszolgálása

1. Lépés (A hardver által vezérelve)

- az eszközvezérlő beállítja a megszakításkérő vezérlővonal jelszintjét, ezzel jelzi a processzornak a megszakításkérelmet (INT jel);
- a processzor visszaigazolja a megszakításkérelem elfogadását (IACK jel);
- ezt követően az eszközvezérlő a sínre küldi a megszakítási vektor elemének sorszámát;
- a processzor tárolja a megszakítási vektorok elemének sorszámát;
- a processzor elmenti a verembe az utasításszámláló és az állapotregiszter tartalmát;
- a processzor a megszakítási vektor elemsorszáma alapján a megszakításkiszolgáló rutin kezdő címét betölti az utasításszámláló regiszterbe és ezzel megkezdődik a megszakítás-kiszolgáló rutin végrehajtása.

2. Lépés (Az operációs rendszer által vezérelve)

- a megszakított program további regisztereinek elmentése a verembe;
- a megszakítás okának behatárolása;
- a kiszolgáláshoz szükséges adatok összegyűjtése;
- a megszakítást okozó esemény kezelése;
- a megszakított program adatainak visszatöltése;
- a megszakítás-kiszolgáló rutin befejezésének jelzése.

3. Lépés (A hardver által vezérelve)

- az elmentett állapot és utasításszámláló regiszter tartalmának visszatöltése és a megszakított program folytatása.

A megszakítások kiszolgálása mindig abban a sorrendben történik, ahogyan beérkeztek. Kivételt képez az az eset, amikor egy időben több megszakítási kérelem is érkezik.

Ilyenkor a processzor feladata, hogy eldöntse melyik megszakítási kérelem áll magasabb prioritási szinten, azaz amelyik végrehajtása a fontosabb.

4.1.3. A közvetlen memória-hozzáférés

A közvetlen memória-hozzáférés lényege, hogy a processzor egy I/O művelet végrehajtásához szükséges információkat átadja egy, a processzortól független DMA vezérlőnek, mely ezt követően az adatátvitelt a memória és az I/O eszköz között önállóan irányítja. Ezáltal a processzor felszabadul más feladatok végrehajtására.

4-6. ábra: ***A DMA logikai sémája***

A DMA átvitelt az I/O eszközök a DMA kérő (DMA REQuest) vezérlő vonalakon kezdeményezhetik. Ezekhez prioritás van hozzárendelve, ami szerint a DMA vezérlő rangsorolja az adatátviteli igények kiszolgálását. A DMA vezérlő a jelzi a processzornak az adatátviteli igényt, melyet a processzor engedélyez. A DMA vezérlő egy I/O művelet befejezését egy megszakítás kérelemmel jelzi.

Az adatátvitel állapotának nyilvántartására a DMA egy címregisztert és egy számlálóregisztert alkalmaz, melynek tartalma minden egyes átvitt adat után aktualizálásra kerül.

A DMA-vezérlő még további három regisztert is tartalmaz:

- DMA módregiszter, mely az adatátvitel irányára (memóriába írás, vagy memóriából történő olvasás) vonatkozó információkat tartalmaz,
- DMA maszkregiszter, mely az egyes DMA átvitelt kérő vezérlővonalak letiltását (maszkolását) tartalmazza,
- DMA állapotregiszter, mely a vezérlő állapotával kapcsolatos információk tárolására szolgál (pl. melyik DREQ vonalon érkezett a kérés, befejeződött-e az átvitel stb.).

Egy számítógépben általában több DMA vezérlő is megtalálható, ezek master, illetve slave kapcsolatban álló eszközök.

4.1.4. Input-output eszközvezérlők

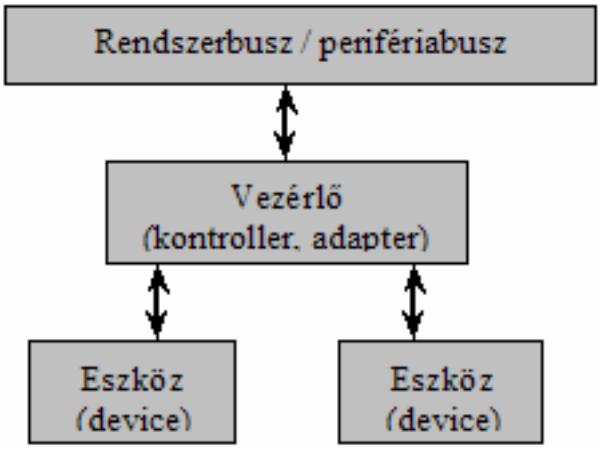
A külvilággal számítógépeink a perifériákon keresztül tartják a kapcsolatot. A kapcsolatok közül néhányat felsorolva:

- Kapcsolattartás a felhasználókkal
- Gépek közötti kapcsolattartás (hálózati kártyák, kommunikációs portok)
- Kapcsolattartás kiviteli és beviteli perifériák és a gép között
- Kapcsolattartás a gép és háttértárak, valamint archiválási perifériák között.

A periféria eszközök valamilyen vezérlőáramkörrel kapcsolódnak – kétirányú adatforgalommal – a számítógéphez, pontosabban a számítógép valamelyik sínjéhez (4.7 ábra). A vezérlőket ráépíthetik az alaplapra, vagy lehetnek egy a rendszersínre vagy valamelyik I/O sínre csatlakó illesztőkártyán kivitelezve. Egy vezérlő vezérelhet több eszközt is, akár különböző fajtájúakat is. A vezérlők különböző bonyolultságúak lehetnek, akár saját vezérlő mikroprocesszoruk is lehet.

Az I/O eszközök és a processzor kapcsolatát az eszközvezérlőkben található *regiszterek* biztosítják. Minden egyes eszközvezérlő funkcionálisan legalább a következő típusú átmeneti tárolókat tartalmazza:

- parancs (command) regiszter, mely az eszközvezérlő által végrehajtandó műveletekhez szükséges információkat tárolja,
- állapot (status) regiszter, melyben az eszközvezérlő az I/O eszköz aktuális állapotára vonatkozó információkat tárolja (például egy merevlemezre egy blokk kiírása megkezdődött, vagy a nyomtatóból kifogyott a papír),
- az adatkiírás illetve beolvasás pufferregiszterei, melyek a folyamatban lévő I/O műveletek adatait tárolják.



4-7. ábra: Perifériák architektúrája

A vezérlők általános feladatai:

- Kapcsolódó felületet biztosítanak a számítógép sínjéhez, ezzel a gép további részeihez.
- Egyes fajtáik képesek a CPU felügyelete nélkül is (DMA) vezérléssel megoldani saját puffer és a központi tár közötti adatátviteleket.
- Szinkronizálást oldanak meg a megszakítási rendszer segítségével.
- Átmenetileg tárolják az adatokat.

4.2. Kommunikációs kapcsolatok a számítógép részegységei között

Annak érdekében, hogy számítógépeink főbb részei (CPU, memória, I/O, kapcsolóegységek) működő egészet alkossanak, valamilyen szervezett módon kell az összeköttetést létrehozni közöttük. Az összeköttetés módja jelentős hatással van a számítógép teljesítményére. Hiába rendelkezünk hatalmas teljesítményű építőelemekkel, ha a közöttük lévő kommunikáció szűk keresztmetszetű.

Az összekapcsolási struktúrát felfoghatjuk egy gráfként is, melynek egyes csomópontjai reprezentálják a rendszerkomponenseket, mint a CPU, memória, stb. s az élek pedig a fizikai összeköttetéseket.

Régen, korai számítógépekben a "busz" egy elektromosan párhuzamos rendszert jelentett, az elektromos vezetők, vezetékek megfeleltek a CPU kivezetéseinek. Ez a megoldás nem sokáig működött így, a buszokon a különböző eszközök kommunikációja külön menedzsmentet igényelt, a mai modern rendszerekben pedig, már egyre inkább összemosódnak a buszok és a hálózatok közötti különbségek, mivel a busz interfészek közötti kommunikáció szabályait — a hálózatokhoz hasonlóan — protokollok határozzák meg.

Mint az első fejezetben már megismertük, a DEC volt az első cég amelyik, felesleges és drága megoldásnak tartotta az addig általánosan használt két buszrendszert a kisméretű, nagy sorozatban készülő harmadik generációs számítógépeinél. Ezért kialakítottak egy olyan memória buszt, amelyre a perifériák is csatlakozni tudtak. A memória műveletek így egyszerűbbé váltak, a busz pedig, bonyolultabbá.

Az első időkben a számítógépek busz rendszerei tulajdonképpen egy passzív hátlapra épített, és egymással összekötött csatlakozókból álltak. Az egyes egységeket, a CPU-t, a memóriát vagy memóriákat, a perifériavezérlőket, stb. a csatlakozókba dugva automatikusan létrejött a kapcsolat az egységek között. Az egységek párhuzamosan voltak összekötve.

A kommunikációt a CPU vezérelte, a periféria vezérlők felé az adatok írása és olvasása előre meghatározott memóriaterületeken keresztül történt (legtöbb esetben), a központi órajeleknek megfelelő sebesség mellett. A külső egységek a CPU egy csatlakozójára adott jellel jelezték, hogy kiszolgálási igényük van, amihez általában valamilyen megszakítás (interrupt) is hozzá volt rendelve.

Például, a diszkvezérlő, jelezni tudta a CPU-nak, hogy az adatok olvasásra készek a kijelölt memóriaterületen (mivel előzőleg betöltötte oda a lemezről leolvasott adatokat), a CPU ezután már hozzáfért a kért adatokhoz. A CPU és a diszkvezérlő között a kapcsolatot a memória biztosította. Az első és második generációs gépek túlnyomó többsége ilyen elven működött.

A sínrendszer feladata

A számítógép különféle komponensei között a gép működése során adatátvitelt biztosít. A legjellemzőbb összekapcsolandó részegységek a következők:

- processzor - memória;
- processzor - periféria;
- memória - periféria.

Annak érdekében, hogy a számítógép megfelelő működési sebességet érjen el, úgy kell szervezni, hogy minden egység egy adott időben a memória adatszélességének megfelelő adatmennyiség, egy egész adatszó kezelésére legyen képes. Amikor egy adatszót továbbítunk, párhuzamosan, egyszerre továbbítjuk minden bitjét. Ehhez, pedig megfelelő számú vonalnak kell rendelkezésre állnia. A több egységet összekötő vonal-halmazt sínnek nevezik. A sín kifejezés ebben az értelemben nem csupán rendszer-komponensek közötti kommunikációs vonalat jelenti, hanem az ezen vonalak elérését vezérlő mechanizmust és a kommunikációt biztosító jelátvitel felügyeletét is.

Egy számítógép sínrendszerének kialakítására számos alternatív megoldás képzelhető el. Ez a változatosság azonban azt is jelenti, hogy egy adott számítógéphez illesztett interface áramkörrel csatolt periféria nem biztos, hogy használható egy másik számítógépben is. Az I/O egységek különbözősége így is különböző illesztőegységet igényel. Amennyiben a változatokat a busz szintjén is megengedjük, áttekinthetetlen rendszer jön létre. Jobb alternatívát jelent az illesztő jelek és protokollok szabványosítása.

Az **illesztő (interface)** szó két áramkör vagy egység közötti kapcsolatot jelenti. Az interface-re vonatkozó szabványban rögzítik a kapcsolatra vonatkozó valamennyi funkcionális, elektromos és mechanikus jellemző összes specifikációját. Ez azt jelenti, hogy meghatározza a csatlakozó fizikai jellemzőit, az abban használt vonalak feszültségszintjeit, és a hozzájuk tartozó logikai jellemzőket.

Ha a **sínrendszert műszaki szempontból** akarjuk meghatározni, akkor a busz vagy sín alatt olyan, azonos feladatot ellátó vezeték-csoportot értünk, mely egyes vezetékein csak

két feszültségszint jelenhet meg, a logikai 0 és 1-nek megfelelő érték. (Ez géptípusonként vátozó érték lehet, általában 0 és 5 V, valamint 0 és 3,3V.)

Funkcionális meghatározás szerint a busz vagy sín alatt olyan vezetékcsoportot értünk, amit arra terveztek, hogy egy n-bites szó valamennyi bitjét átvigye egy specifikált forrásból egy specifikált rendeltetési helyre. A sín ebben az értelmezésben nem csupán a két egység közötti adatátviteli útvonalat jelenti, hanem azt a mechanizmust is, amely ellenőrzi az ezen útvonalakhoz való hozzáférést és felügyeli a jelátvitelt, ezáltal biztosítja a kommunikációt.

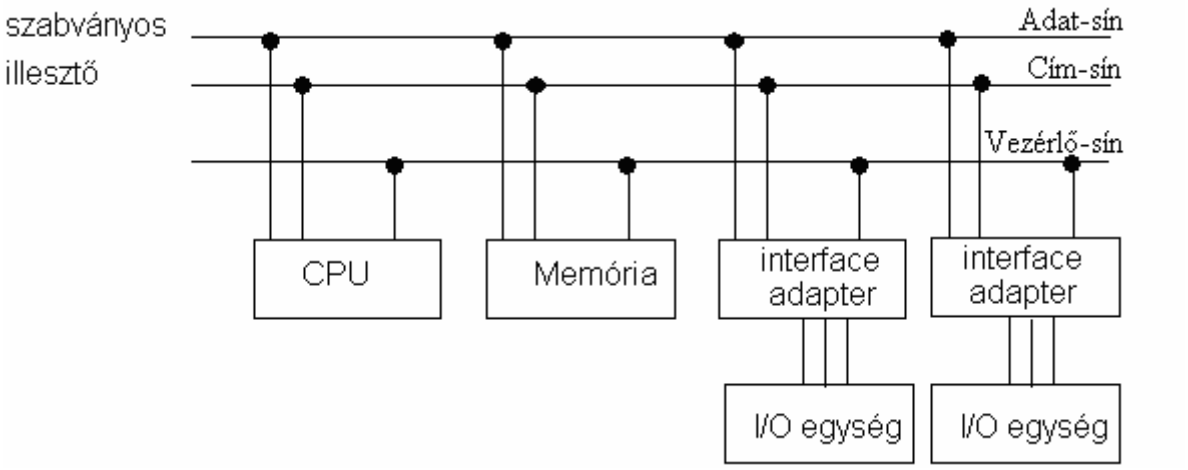
A gép különböző egységei egymás számára adatokat, vezérlő jeleket csak ezen keresztül továbbíthatnak, szabályozott, egységes módon.

A **sínrendszer** használatának **legfőbb előnye**, hogy szabványosított illesztőfelületet biztosít, és emiatt a csatlakoztatott eszközök könnyen cserélhetők. A sínrendszer előírásait teljesítő kártyák bármelyik gépben használhatók. Így gyártótól, géptől függetlenné válik az egyes eszközök használata. A buszrendszerű gépek karbantartása is könnyebbé válik, mivel a hibás részegységek egyszerűbben behatárolhatók.

4.2.1. A számítógépek sínrendszerének logikai részei

A számítógép részegységeinek kommunikációját biztosító sínrendszer, az átvitt információ jellege szerint logikailag három részre osztható:

- **címsín**, mely a címek átvitelét biztosítja. A processzor címkezelésének (pl. 32 vagy 64 bites processzorok) megfelelően általában 32 vagy 64 címvezetéket tartalmaz;
- **adatsín**, mely az adatok átvitelét biztosítja, szélessége általában 32 vagy 64 bit;
- **vezérlősín**, mely a számítógép részegységei között a vezérlőinformációk adatátvitelét biztosítja. Ezek lehetnek például:
  - adatátvitelt, azaz az I/O eszközöket vezérlő jelek,
  - a megszakítási rendszerhez tartozó vezérlőjelek,
  - a DMA vezérlőjelei,
  - a sínvezérlőjelek (például a sínhasználat kérése és ennek visszaigazolása),
  - szinkronizációs jelek.



4-8. ábra: ***I/O illesztőegységek kapcsolódása a sínre***

Mivel minden eszköz ugyanarra a közegre, a sínrendszerre kapcsolódik, és az átvitel eszközpárok között történik, az átvitel létrehozásakor

- ki kell jelölni az adatátvitelben résztvevő eszközöket a cím alapján (adress)
- meg kell határozni az átvitel irányát;
- meg kell oldani a kapcsolatban résztvevő eszközök működésének sebsségbeli összehangolását, szinkronizálását.

Adatsín:

Az adatvonalak egy n-bites szó valamennyi bitjének a párhuzamos átvitelére szolgál; így vagy két készlet, egyenként n darab egyirányú vonalból, vagy egyetlen készlet, egyenként n darab kétirányú vonalból áll. Az Intel 8088 még 8-bites külső sínkapcsolattal rendelkezett. A 80286-os révén vált elterjedtté a 16-bites külső sín. A 80386-tól kezdve az Intel processzorok által kezelt külső sínszélesség - még a Pentium esetén is - 32 bit. A 64 bites sínszélességet csak 2005-től kezdődően alkalmazza az Intel és az AMD a legújabb fejlesztésű processzoraikban.

Címsín:

A címvonalakat az adatátvitelben használt azon egység vagy egységrész identifikálására használják, amelyet a sínhez kell csatlakoztatni. Tehát az eszközök címzésére szolgál. Szélessége 32 (esetleg 16-20-24) bitnek megfelelően ugyanennyi vezeték. Az Intel 8088 címsín szélessége 20 bit volt, amit a 286-nál 24-re, a 386-nál 32-re növelték, s máig is, a Pentiumnál is ilyen széles. Ez 2^{32}, azaz 4GB nagyságú valós címtér címzését teszi lehetővé.

Léteznek olyan processzor megoldások, melyek a címeket és az adatokat ugyanazon a buszon viszik át, természetesen nem egy időben, hanem időbeli multiplexeléssel. Így kétszer annyi ideig tart egy adatátvitel, tehát ez egy kompromisszumos megoldás, melynél a sebesség rovására, mérsékelni lehet a sín költségeit, vagy a processzor kivezetéseinek számát.

Vezérlősín:

A vezérlő vonalak a rendszerben az időzítő jelek és az egység állapotáról szóló információ átvitelére szolgálnak. Ezek jelezhetik az adatvonalakon küldött információ típusát is. A vezérlőjelek száma változó, általában minimálisan 10-15 körül van.

A vezérlőjelek fajtái:

Az **adatátvitelt vezérlő** jelek: (processzorfüggőek**,** a jelölések változhatnak az egyes processzoroknál):

- M/IO (memory/input-output) az adatátvitel helyét jelöli ki, mely választhatóan lehet a memória vagy a perifériák;
- R/W (read/write) az adatátvitel irányát adja meg a processzor oldaláról nézve;
- WD/B (word/byte) az átvitt adat méretét jelöli meg, amelyet egy tárhoz fordulás alkalmával egységként kezel;
- AS (address strobe) a cím sínre helyezését jelzi az eszköz (memória) számára;
- DS (data strobe) az adat sínre helyezését jelzi az eszköz (memória) számára;
- RDY (ready) az átvitel befejeztét, vagy az eszköz rendelkezésre állását jelzi.

A **megszakítást vezérlő** jelek: megszakítást kérő és megszakítást visszaigazoló jelek.

A **sínvezérlő** jelek: a sín kérését, foglalását és a foglalás visszaigazolását szolgáló jelek.

Egyéb jelek: órajelek, melyek a gép ütemezését, szinkronizálását szolgálják, valamint a tápfeszültség.

4.2.2. A sínrendszerek típusai

A sínrendszerek között megkülönböztetünk

- **belső sínrendszert**, mely a processzoron belül, a processzor különböző részeit kapcsolja össze. Sebessége (órajele) megegyezik a processzoréval (például 400 MHz);
- **külső sínrendszert**, mely a processzort köti össze a központi egység különböző részegységeivel. Sebességét a processzor órajelének osztásával határozzuk meg (például 400/4=100 MHz). A külső sínrendszer a sebessége és az összekapcsolt eszközök alapján kétfajta lehet:
    - **helyi sín** vagy local bus, mely a processzorhoz közvetlenül kapcsolódó rendszerelemeket (memória, grafikus kártya stb.) köti össze. A helyi sínen keresztül az adatátvitel a processzor órajelével szinkronban történik és a busz adatátviteli bit szélessége is megfelel processzor működésének (32-bites processzoroknál 32 bit).
    - **rendszersín** vagy system bus, melyet egy sínvezérlő egység hajt meg, és alapvetően az I/O eszközök csatlakoztatását szolgálja.
- **az I/O eszközök saját sínrendszere** (például SCSI lemezcsatoló busza).
- **számítógéprendszerek közötti** buszok (intersystem bus).

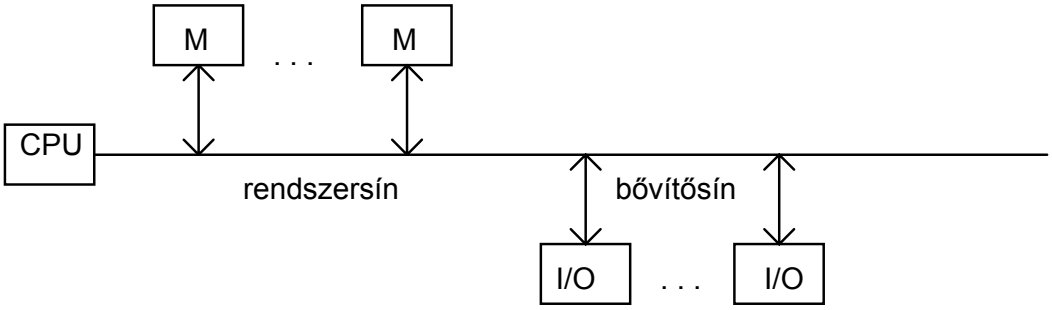
A sínrendszer osztályozása

Igen nehéz lenne a számítógép rendszersínje számára egy univerzális szabványt kidolgozni. A sín struktúrája szorosan kapcsolódik a számítógép architektúrájához, így általában ez számítógépről számítógépre különböző. Hasonlóképpen nehéz lenne egy olyan szabványt kidolgozni, amely valamennyi számítógép perifériát lefedi, mivel rendkívül széles skálája van az átviteli sebességnek és más követelményeknek. Gyakorlatiasabb alternatívát jelent bizonyos kapcsolat-osztályok számára kidolgozni olyan szabványokat, amelyek megfelelnek mind a számítógépeknek, mind pedig a perifériáknak. Így alakult ki a rendszersínt nyitottá alakító, széleskörű periféria-csatlakoztatási lehetőséget biztosító bővítősín.

Egyszintű sínrendszer

Sok számítógép, főleg a kisteljesítményűek egyetlen megosztott sínt tartalmaztak, ezt hívjuk egyszintű sínrendszernek. A működési elve egyszerű:

Valamennyi egységet ehhez a sínhez csatlakoztatjuk, így ez az egyetlen kapcsolódási lehetőségük. Mivel egyetlen sínt egy időben csak egyetlen átvitelhez használhatunk, csupán két egység – egy adó és egy vevő - tudja aktívan használni egy adott időben. Ezek az egyszerű buszok voltak a legfontosabbak az általános célú számítógépek számára. Minden, a buszhoz csatlakozó berendezés azonos sebességgel tudott a buszon kommunikálni, amit a központi órajel határozott meg.



4-9. ábra: ***A rendszersín***

A logikailag rendszersín a processzort köti össze a gép egyéb részeivel, elsősorban a memóriával és közvetlenül, vagy közvetve (például egy sínmeghajtó) az I/O eszközökkel. Az ábránkon közvetlen összeköttetés van.

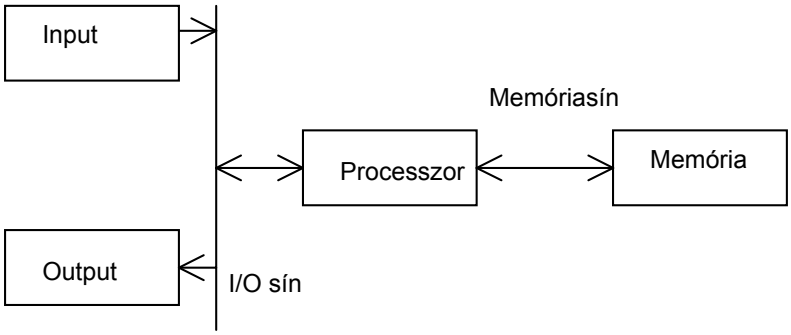
Az eredeti IBM PC, a PC XT és az első PC AT is még csupán egyetlen sínnel rendelkezett. Ezt a 6-8,33 MHz frekvenciával meghajtott egyetlen sínt azonban két részre bonthatjuk, igaz, hogy csak funkcionális szempontok alapján:

- rendszersín
- bővítősín.

Az egyetlen sín rendszersínnek nevezett részéhez csatlakozik a CPU, a matematikai társprocesszor, a memória (RAM és ROM egyaránt), a DMA vezérlő, a megszakítás-vezérlő és az időzítő, a bővítő sínnek nevezett részéhez pedig a különféle perifériák illesztői.

A CPU-k sebességének növekedése komoly problémát jelentett, mivel az előzőek értelmében a periféria vezérlők sebességét is azonos mértékben növelni kellett volna. Ez nem minden esetben sikerült, így előfordult, hogy egy gyors CPU-nak alkalmazkodnia kellett a leglassúbb perifériavezérlő sebességéhez, a CPU sebesség növekedését nem lehetett kihasználni. Amíg ezt a megoldást a beágyazott rendszerek (mikroprocesszoros vezérkő rendszerek, célszámítógépek) esetében elfogadták, a kereskedelmi célú számítógépek esetében a piac már nem tolerálta.

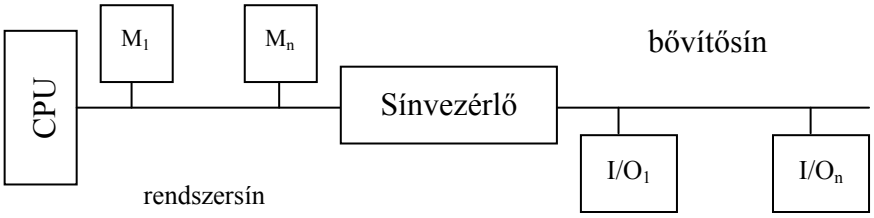
Kétszintű sínrendszer



4-10. ábra: ***Kétszintű sínrendszer elve***

Ez a legegyszerűbb formája a kétsínes számítógép-architektúrának. A processzor a memóriával a memóriasínen keresztül áll kapcsolatban és az input és az output műveleteket az I/O sínen keresztül kezeli. Az adatok a processzoron keresztül mennek át a memóriához vezető útjukon. Az ilyen konfigurációban az I/O átvitel általában a processzor közvetlen felügyelete alá tartozik, amely kezdeményezi az átvitelt és felügyeli a végrehajtásukat egészen a teljesülésükig.

A későbbi megoldásokban a buszok mereven két részre osztották a "világot": az egyikbe a CPU és a memória($M_1..M_N$ modulok) tartoztak, a másikba pedig a számos vezérlőegység. A két világ közötti kapcsolatot a *sín vezérlő* (bus controller) biztosította.

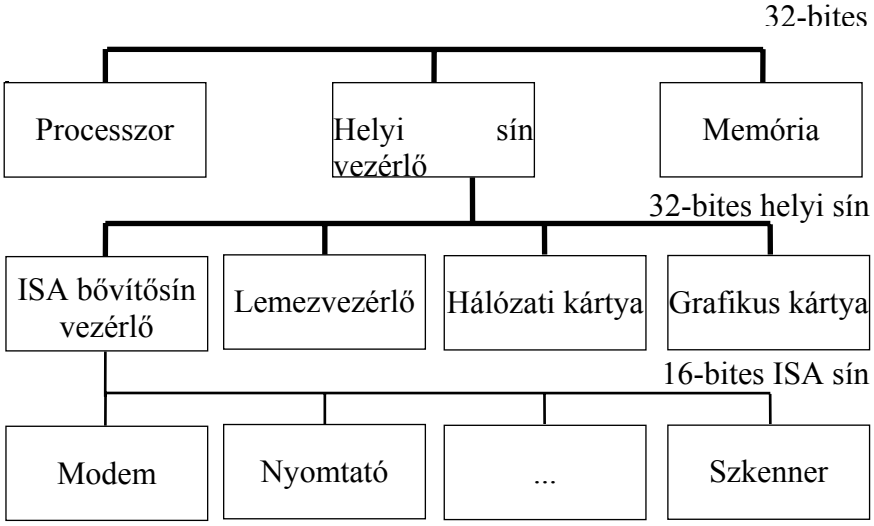


4-11. ábra: ***Kétszintű sínrendszer sínvezérlővel***

Ez a megoldás megengedte a CPU sebességének növelését anélkül, hogy buszra hatással lett volna. Ez a megoldás a terhelést a CPU-ról a csatolókártyák felé mozdította, a csatolókártyáknak egymással és a buszvezérlővel kellett kommunikálniuk, és ehhez nem volt szükség a CPU igénybevételére. Sikerül ilyen módon a jobb teljesítményt elérni, cserébe viszont a csatolókártyák sokkal bonyolultabbak lettek.

Háromszintű sínrendszer

A háromszintű sínrendszer az I/O eszközök sebességkülönbségét is figyelembe veszi, és a második szinten a gyorsabb kiszolgálást igénylő perifériákat, pl.: merevlemezes tár, USB, stb. priférіákat csatlakoztatja, míg harmadik szinten a lassú eszközök kapcsolódhatnak.



4-12. ábra: ***Háromszintű sínrendszer***

4.2.3. Buszprotokoll és a sínvezérlés formái

A sínt egyidőben csak egy eszközpár használhatja. A sín használatát valamelyik eszköz kezdeményezi, amelyet aktív eszköznek (master) neveznek, szemben a kapcsolatban résztvevő másik, passzív eszközzel (slave), amely csak fogja és végrehajtja az aktív eszköztől származó vezérléseket. A mikroszámítógépeknél a sín irányítását megszerző eszköz

- a processzor;
- a DMA vagy
- I/O processzor.

MASTER:

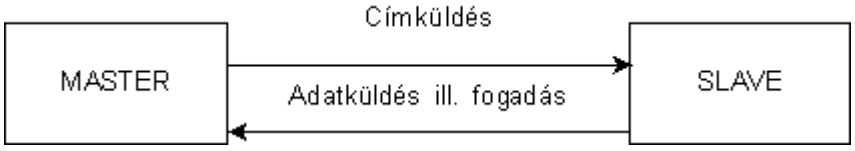
Elindít és befejez egy busztranzakciót,

Címet küld.

SLAVE:

Válaszol az igényekre és címekre,

A sínre teszi illetve fogadja az adatokat.



4-13. ábra: ***Master és slave eszközök viszonya a buszon***

A mikroszámítógépeknél master általában a processzor vagy valamilyen DMA-t használó I/O eszköz lehet, a memória viszont mindig slave szerepkört tölt be a sínhasználat során.

A PC-kben alkalmazott buszok ismertetésére az 5. fejezetben térünk ki részletesen.

A sínrendszeren keresztül zajló kommunikáció csak abban az esetben valósulhat meg, ha minden résztvevő ugyanazokkal a paramétereket használja a sín elérésére. Azokat a szabályokat, amelyek a rendszer működésében alapvető jelentőséggel bírnak, sínprotokollnak nevezzük.

A sínprotokoll magába foglalja a működési szabályokat, az elektromos- és mechanikus jellemzőket, valamint azt, hogy melyik eszköz hogyan kapcsolódhat a sínre.

A sínvezérlés

Digitális rendszerekben az adatátvitel és ezzel együtt a sínhez kapcsolódás csak megadott rendszerben történhet. Annak érdekében, hogy minden eszköz a helyes időpontban érzékelje a vezetékeken lévő információt, azokat össze kell hangolni, vagyis szinkronizálni kell.

Az adatátvitel vezérlését két különböző módon lehet megvalósítani:

- **Szinkron vezérlésről** beszélünk akkor, ha az eseményeknek rögzített időpontjai vannak, a sínre kapcsolódó eszközök ugyanazt a szinkronizáló jelet (órajel) használják a működés vezérléséhez. Az órajelnek az éleit hasznosítják szinkronizációra, mivel az érzékelése annak egyszerű, a lefolyása pedig gyors (elméletileg nulla idő alatt történik meg a váltás). Lehetőség van az azonos éleket felhasználni, de van mód az ellenkező élek használatára is. Az adatok adása és vétele azonos sebességgel történik, mivel a kapcsolat mindvégig fennáll, nem kell kapcsolatfelvétel, valamint nincs szükség a visszaigazolásra sem. A megoldás gyors adatátvitelt tesz lehetővé, azonban plusz vezetéket igényel az órajel továbbítása miatt. A másik sajátossága a megoldásnak, hogy folyamatos adatátvitelt követel meg, az előzőekben leírtak alapján. Nagyon fontos megjegyezni, hogy a szinkron adatátvitel csak abban az esetben használható hatékonyan, ha a kommunikációban résztvevő eszközök azonos sebességűek. A szinkron rendszerek tervezésekor nagyon körültekintően kell eljárni, mert a legkisebb hiba is a rendszer működésképtelenségét vonhatja maga után.
- **Aszinkron sínvezérlés** alkalmazásakor az események tetszőleges időpontban bekövetkezhetnek, közös órajellel nem, de sajáttal rendelkezhetnek az összekapcsolódó eszközök. A kapcsolat csak az adatátvitel idején áll fenn. Ebből a tényből következik, hogy szükség van a kapcsolatfelvételre, valamint az adatok

vételét vissza kell igazolni. A folyamatok lefutását és összehangolását az egymást követő elemi lépések befejezése szabályozza. Az aszinkron adatátvitelnél nincs szükség az órajel továbbítása, megoldható a kommunikáció a nagyon lassú eszközökkel is, valamint nem jelent problémát az sem, ha az adatátvitel nem folyamatos. Ennek oka, hogy minden átvitelt megelőz egy kapcsolatfelvétel. A hátránya viszont hogy lassabb átvitel tesz lehetővé, mint a szinkron vezérlés, valamint szükség van a visszaigazolásra is.

4.3. I/O műveletek végrehajtása mikroszámítógépekben

4.3.1. Az I/O műveletekkel kapcsolatos alapfogalmak

I/O kapcsolatok kezelése: a perifériális eszközök kapcsolatát a processzorral az eszközvezérlőkben található regiszterek segítségével oldják meg.

Minden adatforgalom, parancskiküldés illetve állapotlekérdezés ezeken keresztül valósul meg.

A kapcsolatok kezelésére az I/O portok szolgálnak. Ezeket a:

- parancsregiszter,
- az állapotregiszter és
- az I/O regiszter alkotja.

Egy regiszternyi adat inputjának esemény-sorozata a következő

- az I/O cím alapján kiválasztásra kerül a megfelelő I/O port;
- a processzor beírja az adott egység parancsregiszterébe a kiszolgálási igényét, majd mással kezd foglalkozni;
- az adott periféria egy regiszternyi adatot küld az adat-bemeneti regiszterbe;
- mihelyt itt rendelkezésre áll az adat, a periféria egy megszakítás-kérést küld a processzor felé;
- a processzor megszakítás-kezelő rutinja beolvassa a megszakítás-kérést küldő periféria állapot-regiszterének tartalmát, mely a READY bit beállításával jelzi, hogy a kért adat rendelkezésére áll az adatbemeneti regiszterben;
- a processzor beolvassa az adatbemeneti regiszter tartalmát az akkumulátorába;
- ezután a soron következő programutasítástól függ, hogy az adott akkumulátor-tartalommal mi történik, például STORE utasítás esetén a memóriában kerül eltárolásra, ADD utasítás esetén hozzáadásra kerül az utasításban szereplő memóriacím tartalmához.

4.3.2. Az operációs rendszer szerepe az I/O műveletekben

Az operációs rendszernek fontos feladatai vannak az input/output műveletek végrehajtásának vezérlésében. Ezek közül a fontosabbak:

- **A védelmi funkció**: Az operációs rendszernek gondoskodni kell arról, hogy a taszkok által közösen használt I/O eszközöket a felhasználói programok csak felszabadítást követően vegyék igénybe. Emiatt általában nem engedhető meg, hogy multitaszking üzemmódban a felhasználói programok közvetlen kapcsolatba kerüljenek a perifériákkal. Így amennyiben az IBM PC-n Assembly-ben az I/O műveleteket programvezérelt megszakítással (INT) programozzuk, akkor az operációs rendszer eszközkezelő részének adjuk át a vezérlést.

- **Eszközspecifikus szolgáltatások**: Amennyiben kódkonverziós átalakításra van szükség két különböző hardver egység között, ez is operációs rendszer szinten történik.
- Az **I/O műveletek hibakezelésének** egységes megoldása. A hibakezelés végrehajtásához a perifériavezérlőknek adatokat kell átadni az operációs rendszernek. Így például az operációs rendszert informálni kell:
    - ha az I/O eszköz befejezett egy műveletet,
    - ha az I/O művelet hibátlanul végrehajtásra került.

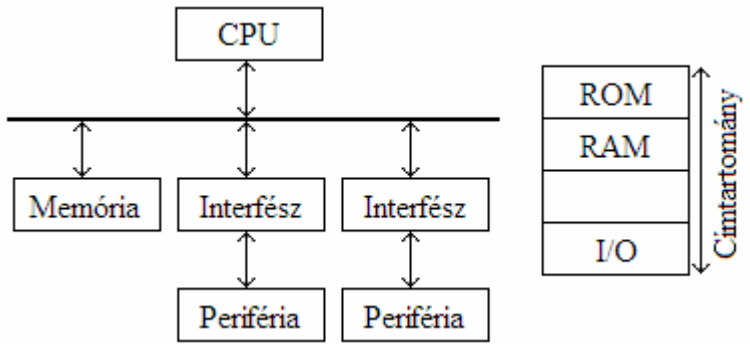
Ezek az információk két módszerrel adhatók át:

- **Polling** alkalmazásával (azaz az eszközök állapotának „körbekérdezésével"). Ez esetben az I/O egység az állapot (státus) regiszterben kódolja az átadandó információt, az operációs rendszer pedig periodikusan és ismétlődően lekérdezi a státusregiszter bitjeit, hogy az eszköz I/O műveletre való alkalmasságát vagy az adatátvitel hibátlan lefutását megállapítsa.
- **I/O megszakítás** alkalmazásával. Ez esetben az I/O eszköz a megszakítási rendszer segítségével informálja az operációs rendszert az I/O eseményekről.

4.3.3. Az I/O eszközök címzése

Minden esetben igaz, hogy a vezérlő eszköz végzi el a passzív eszközök címzését. Ez a művelet három különböző módon történhet:

- **Helyfüggetlen** a címzés abban az esetben, ha az eszközöknek egyértelmű és egyedi címe van. Ebben az esetben egy címre csak egy eszköz válaszolhat. A busz bármelyikcsatlakozójába tehetjük az I/O eszköz csatolóját.
- **Memóriába ágyazott** címzésről (memory mapped addressing) akkor beszélünk, amikor az I/O eszközt úgy címezzük, mintha az operatív tár része lenne. Ebben az esetben nincs kifejezett perifériaművelet. A megoldás előnye, hogy könnyű a kezelhetősége, a hátránya pedig, hogy a címtartomány megoszlik a memória és a I/O eszközök között. Fontos ennél a címzésnél, hogy biztosítani kell, hogy nem történhessen átcímzés a másik területre.



4-14. ábra: ***I/O eszközök memóriába ágyazott címzése***

- **Csoportcímzés** (broadcast) esetén minden passzív egység címzése egyszerre történik meg. Ez akkor célszerű, amikor szeretnénk az összes eszközt azonos állapotba beállítani.

4.3.4. Az I/O átvitel típusai

Programozott I/O esetében a perifériát először meg kell címezni, ez a cím vagy az I/O port sorszáma vagy a memória tároló helyének a címe. A perifériautasítások a processzor egy regisztere és a periféria között valósítanak meg átvitelt. Az átvitel lehet feltétel nélküli (melyet egyszerű esetekben alkalmaznak) illetve feltételes (melynél a periféria állapotjelzőinek ellenőrzése alapján, és után történik az átvitel).

Megszakításos I/O esetében a cél az, hogy processzor idejét felszabadítsuk az átviteli feladatok alól. Az átvitel kezdetén a processzor jelzi az I/O eszköz számára az átvitelre vonatkozó indítási igényt. Az eszköz ezután az átvitel kezdetére alkalmas időpontot a processzor felé küldött megszakítási kérelmével jelzi.

Közvetlen tárolóhoz fordulás esetén a processzor által elindított DMA vezérlő önállóan irányítja az adatátvitelt a tároló és a kijelölt I/O eszköz között a processzor kihagyásával.

Ellenőrző kérdések:

1. Ismertesse a számítógép legfontosabb részegységeit!
2. Mi a megszakítási rendszer szerepe, és hogy működik?
3. Mi a különbség a megszakítások és a kivételek között?
4. Milyen megszakítás kiszolgálási módszereket ismer?
5. Ismertesse az egyszintű és a többszintű megszakítási rendszer közötti különbségeket!
6. Ismertesse a megszakítások és kivételek hardver és szoftver által vezérelt lépéseit!
7. Ismertesse a DMA működésének lényegét!
8. Mi az input-output eszközök feladata?
9. Milyen funkciójú regiszterek találhatók az I/O eszközvezérlőkben?
10. Mi a sínrendszer feladata?
11. Milyen célt szolgál a címsín?
12. Milyen célt szolgál az adatsín?
13. Milyen célt szolgál a vezérlősín?
14. Ismertesse a vezérlő jelek fajtáit!
15. Hogyan lehet a sínrendszereket osztályozni?
16. Mi a többszintű sínrendszerek létrejöttének az oka?
17. Ismertesse az I/O műveletek végrehajtásának lépéseit!
18. Mi az operációs rendszer szerepe az I/O műveletekben?
19. Hogyan történik az I/O eszközök címzése?
20. Ismertesse az I/O átvitel típusait!

5. Az IBM PC

A személyi számítógép, angolul Personal Computer, azaz PC elnevezés azt követően vált általánossá, hogy az IBM az 1981-ben bemutatott mikroszámítógépét PC néven vitte piacra (IBM típusszáma 5150). Ehhez az Intel cég által gyártott processzort építették be, és a Microsoft szállította az operációs rendszert.

IBM kompatibilis PC-nek, nevezzük azokat a mikroszámítógépeket, amelyek Intel vagy azzal kompatibilis processzorcsaládra épülnek, és működésük lefelé kompatibilis az IBM-PC hardver- és szoftverelőírásaival.

A PC-k lefelé való program-kompatibilitásának biztosításához a legfontosabb eszköz az un. x86-os utasításkészlet. Ez alatt azoknak az utasításoknak az összességét értjük, amelyek lényegében nem változtak az IBM PC- megjelenése óta. Ezeket, az utasításokat minden PC processzora ismeri és képes végrehajtani.

Az x86-os utasításkészlet szintjén való kompatibilitás viszont nem jelent teljesen azonos módon történő működést, azaz hardver-kompatibilitást. Egy Intel vagy azzal kompatibilis processzor utasítás-végrehajtását *natívnak* nevezzük, ha a processzor egy x86-os utasításkészletbe tartozó gépi utasítást közvetlenül, emuláció nélkül képes végrehajtani.

A fejlettebb processzorok a program-kompatibiltást azzal érik el, hogy az x86-os utasítások végrehajtását *emulálják*.

A szabványos utasításkészlet miatt a PC-k szoftverellátottsága széleskörű, ebből fakadóan a legkülönbözőbb alkalmazási területeken – például ipari, mezőgazdasági, egészségügyi, oktatási, irodai, multimédia, hálózati munkaállomás stb. - megtalálhatók.

A PC elterjedését az is felgyorsította, hogy a PC nyilvános specifikációit felhasználva egyre több cég másolta az eredeti IBM PC-t és utódait. Így jöttek létre a neves (Compaq, Dell stb.) és a névtelen (noname) gyártók IBM PC másolatai, amelyeket klónoknak szoktak nevezni.

A szabványos buszspecifikációk lehetővé tették azt is, hogy egyes cégek a PC-k egyes hardveregységeinek, például mágneslemezes táraknak, a gyártására szakosodjanak. A teljesítmény fokozásának állandósult igénye, amely főként a processzor- és busz-sebesség, valamint a tárolókapacitás növelésére irányult, azt eredményezte, hogy az elmúlt 20 évben a processzorok sebessége több százszorosára, a központi memória- és háttértár-kapacitás pedig több ezerszeresére nőtt.

Az integrált áramkör-technológia fejlődése egyre inkább lehetővé tette a PC-k gazdaságos gyártását. A PC olcsósága gyorsan bővítette felhasználásának szakterületeit, ez egyúttal újabb keresletet teremtett, és az ebből fakadó nagyobb gyártási sorozatok ismét alacsony árszintet eredményeztek.

A számítógép és alkatrészeinek tervezési, technológiai munkáinak koncentrálódása miatt a felszabaduló szellemi energia átcsoportosulhatott a szoftver fejlesztésére. A PC ipari szabvánnyá válása és az eredeti PC-vel való program-kompatibilitásának a teljes fejlődési folyamat során való megőrzése jó feltételeket is teremtett ehhez. (Ha egy alkalmazási területen egy problémát megoldottak egy PC-re írt programcsomaggal, akkor az azonnal több százezer, majd később több millió PC-n futtathatóvá vált.)

A PC-k az 1990-es évekre már a számítástechnikai piac meghatározó tényezőjévé váltak, tömeges elterjedésük megteremtette a számítástechnika társadalmi méretekben történő hasznosításának alapjait.

5.1. A PC fejlődéstörténete, részegységei

A PC fejlődéstörténete lényegében a bennük működő processzor történetével határozható meg legegyszerűbben, hiszen a gépen futó operációs rendszer, a felhasználói programok, de az új perifériák is a processzor teljesítményétől nagymértékben függenek.

5.1.1. Processzororok

Első generációs processzorok

Az első IBM PC-t 1981 április 24.-én mutatták be. Ezzel a géppel fektette le az IBM azokat a szabványokat, amelyek máig is érvényesek. Ez a gép még nem hozott átütő sikert, mivel „tudása" éppen csak meghaladta a korszak hasonló gépeit. A használhatóság fő korlátja a tárolókapacitás volt, hiszen ebben még merevlemezt nem alkalmaztak, háttértárként a floppy szolgált.

1983-ban jelentette meg az IBM a javított PC-t, XT néven (eXtended Technology). A fő különbség a két típus között a háttértár alkalmazásában volt. 10 Mbyte-os merevlemez egységet ajánlottak a géphez. Az első IBM PC-kben és XT-kben az Intel **8086**-os vagy **8088**-as processzora működött. Ezeket nevezzük első generációs processzoroknak. Az első változat 4,77 MHz órajelen működött, majd később megjelent a 8 ill. a 10 MHz-es változat is. A processzorba 29 000 tranzisztort integráltak. A lebegőpontos műveletek végrehajtásához egy különálló társprocesszort lehetett a 8086-os processzorhoz csatlakoztatni, ennek típusjele 8087 volt.

Második generációs processzorok

Az XT gépeket követő AT (Advanced Technology) típusú számítógépek gyártásához az IBM az a **80286**-os processzort használta. Ennek első változata 1982-ben jelent meg. A processzor 16 bit széles adatsínt és 24 bit széles címsínt használt, 16 Mbájt memória megcímzésére volt képes. Ez volt az első, olyan mikroprocesszor, amely már virtuális tárkezelésre is alkalmas volt. Kezdetben a processzor órajele 6 MHz volt, és a későbbi változatok elérték a 20 MHz-et is. A virtuális tárkezelés miatt kétféle üzemmódban működött, a 8086-os kompatibilitás megőrzése érdekében szükséges a valós, és a virtuális tárkezelést alkalmazó védett üzemmód. A processzor 1 Mbájtnál nagyobb memóriaterületet, így a maximális 16 Mbájtos memóriát csak védett üzemmódban képes kezelni, valós módban – a 8086-os kompatibilitás miatt – csak 1 Mbájtot. A 286-os processzorhoz is lehetett matematikai társprocesszort kapcsolni, melynek típusjele 80287 volt.

Harmadik generációs processzorok

Az Intel első 32 bites processzora a **80386**, 1985-ben jelent meg. A belső regiszterei, valamint az adat- és címsínjei is 32 bitesek, így a processzor 4 Gbájt memória fizikai címzésére volt képes, virtuálisan pedig 16 Terabyte-ot tud megcímezni.

. Az első széria 16 MHz órajel frekvenciával, az ezt követő különböző változatok 20, 25, 33 és 40 MHz-es órajellel működtek. Az Intel mikroprocesszorai közül először ebben alkalmazták a futószalag (pipeline) utasítás végrehajtást.

A 80386DX processzor megnövelt teljesítménye és a kibővített processzor üzemmódok (védett mód és virtuális valós mód) lehetővé tették a grafikus kezelői felület elterjedését. Az utasításkészlete a szabványos, x86-os utasításokat foglalja magába. Az Intel 386-os processzorával kompatibilis processzorokat az AMD (**A**dvanced **M**icro **D**evices) és a Cyrix cég is gyártott.

Az AT-nál említett két üzemmódon felül tud egy harmadik működési módot is, ez pedig a Virtual Real Mode (virtuális valós mód). Ez egyesíti a Real és a Protected mód előnyeit, úgy hogy egyszerre több program futására van lehetőség Real módban. A 80386-os processzor SX és DX változatban is készült. Az SX jelzésű a DX (teljes értékű) processzor

egyszerűsített változata. Az adatsín szélességét lecsökkentették 16 bitre, a címsín szélességét pedig 24 bitre. A 386SX később jelent meg, mint a DX, és gyártásának az volt a célja, hogy a 286-os alaplapokon a 286-os processzorokat kiváltsák egy azonos interfésszel rendelkező, de nagyobb teljesítményű processzorral.

Negyedik generációs processzorok

Az Intel **80486DX** típusjelzésű volt a negyedik generációs processzorok első tagja. Az alaptípus 1989-ben jelent meg, és CMOS technológiával gyártották. Összesen 1,2 millió tranzisztort tartalmazott. A processzorban a címsín és az adatsín is 32 bites. Az első széria 25 MHz-es órajellel működött, ezt követték a 33 és 50 MHz-es órajellel működő típusok.

Az azonos órajelű 80386DX-hez képest, 150%-os teljesítménynövekedést értek el. Ennek oka a magon belüli gyorsabb utasítás-végrehajtás a már öt fokozatú pipelinenal, a processzorba integrált elsőszintű cachetároló (L1) , a processzorba integrált lebegőpontos műveletvégző egység, valamint a gyorsabb memória-hozzáférést biztosító un. burst mód. A 486-os PC-k sebesség növekedéséhez hozzájárult az ebben az időszakban általánossá váló, alaplapon elhelyezett ún. másodszintű gyorsító-tár (L2).

Az Intelen kívül az AMD és a Cyrix is gyártott 486-os processzorokat. A **486SX** processzort a 486DX processzorból alakították ki úgy, hogy kihagyták belőle a lebegőpontos egységet. A 486SX processzorok 16, 20, 25 és 33 MHz-es sebességű változatban készültek **80486DX2** processzor típusjelzésben a 2-es szám az órajel frekvencia szorzótényezőjére utal, 486DX2 volt az első processzor, amely a rendszerbusz órajelét megduplázta, és a processzoron belüli egységeket ezen a frekvencián működtette. A **80486DX4**-nél az órajelet pedig megháromszorozták, így a processzor háromszor gyorsabban működött, mint a rendszerbusz. Ez a módszer egyszerűbbé tette a sebesség növelését, mintha az alaplap áramkörein módosítottak volna.

AMD 5x86 (Am80486DX5) az AMD 5x86-os processzora, nem egy ötödik generációs processzor, hanem egy négyszeres órajel-többszörözést használó 486-os processzor. Ez a processzor már a 0,35 mikronos technológiával készült, és teljesítménye összemérhető volt az Intel Pentium 75 MHz-es processzorával, amelyet az Intel a 100 MHz-es 486DX4 processzorok fejlesztésének befejezését követően adott ki.

Ötödik generációs processzorok

Az első ötödik generációs processzor az Intel által gyártott szuperskalár architektúrájú Pentium volt. Ez 1993-ban jelent meg, az első változatok 3,1 millió tranzisztort tartalmaztak, és bipoláris CMOS technológiával készültek, valamint 60 MHz-es órajellel működtek. A Pentium az első 64 bites processzor, mely továbbra is 32 bites címsínt használ a memória és a perifériák elérésére. A belső szervezése olyan, hogy a 64 biten érkező adatokat 32 bit szélességben dolgozza fel. A sebesség jelentős növekedését okozta az, hogy ez a processzor egyszerre két műveletet tud végrehajtani. Két gyorsítótárat építettek bele, az egyiket az adatoknak, a másikat az utasításoknak. A belső felépítésben a következő főbb változások történtek:

- Szuperskalár felépítés. A Pentium processzor szuperskalár felépítésű, két végrehajtó egységet használ párhuzamosan.
- Szélesebb adatsín. Az előzőekben már szó volt róla, a processzor a külvilág felé 64 bites adatsínnel kapcsolódik, azonban a belső felépítése továbbra is 32 bites.
- Sokkal gyorsabb memória sín. A legtöbb Pentium 60 vagy 66 MHz-es külső órajelet használ, míg a 486-os csak 33 MHz-est.

- Ugrás előrejelzés. A Pentium ugrás előrejelzést használ a pipeline hatékonyabb kihasználásához.
- Integrált energiatakarékos üzemmód. Minden Pentium processzor támogatja az energiatakarékos üzemmódot.
- Szétválasztott első szintű cache. 8 Kbyte cache áll az adatok rendelkezésre, és 8 Kbyte az utasítások számára. Ezek használata egymástól független.
- Továbbfejlesztett lebegőpontos egység.

A sebesség jelentős növekedését okozta az is, hogy a processzorban két fixpontos és egy lebegőpontos futószalag található, amelyek párhuzamosan működhetnek (szuperskalár architektúra). A pipeline hatékonyabb kihasználásához a processzor spekulatív elágazás-feldolgozást alkalmaz egy 256 elemű ugrás-előrejelző pufferrel (Branch Target Buffer = BTB) 8 KB L1 cache áll rendelkezésére az adatok, 8 KB az utasítások számára, és ezek használata egymástól független.

A Pentium MMX volt az Intel következő, és egyben utolsó (1997), ötödik generációs processzora. A processzort a multimédiás alkalmazások hatékonyabb futtatása érdekében fejlesztették ki. A legnagyobb eltérés a Pentium és a Pentium MMX között a kibővített, multimédiás MMX utasításkészlet. Az olyan alkalmazások futtatása során, amelyek képesek kezelni az MMX utasításkészletet, nagy sebességkülönbséget tapasztalhatunk. Emellett a Pentiumhoz képest még további változtatások is történtek:

- Megnövelt elsődleges cache: külön választott, 16 KB méretű adat és 16 KB méretű utasítás cache.
- Mélyebb futószalag: a futószalag állapotok számát ötről hatra emelték.
- Új MMX egység (8 db MMX regiszter, de ezek fizikailag megegyeznek a régi lebegőpontos regiszterekkel).
- Továbbfejlesztett ugrás-előrejelzés 512 elemű BTB-vel. RSB = Return Stack Buffer, azaz visszatérési verem tároló a szubrutin visszatérési címek előrejelzésére.

A Cyrix a 6x86 processzorral lépett be az ötödik generációs processzorok piacára. A gyártó cég ezt a processzort már hatodik generációsnak is nevezte, bár a teljesítménye alapján még az ötödik generációba tartozik. A lábkiosztást és a feszültségértékeket tekintve a Pentium processzorral teljesen kompatibilis. A processzor azonban teljes egészében Cyrix fejlesztés, és emiatt belső felépítése a Pentium processzorétól jelentősen eltér.

A fontosabb különbségek a következők:

- A belső pipeline mélységét ötről hétre növelték.
- Regiszterátnevezést alkalmaztak az áladatfüggőségek kiküszöbölésére.
- Data forwarding a futószalagok között.
- 16 KB-os egyesített L1 adat- és utasításcache.
- Többprocesszoros rendszerek támogatása.

AMD K5-ös típusú az AMD első ötödik generációs processzora melyet az Intel Pentium vetélytársának szántak. A Cyrix 6x86-hoz hasonlóan az egyes CPU-kat nem tényleges működési frekvenciájukkal, hanem úgynevezett PR (Pentium Rating) jelöléssel jellemezték. A processzor fontos tulajdonsága, hogy a mag már RISC felépítésű. Ez akkoriban újdonságnak számított, manapság már ez az általánosan elterjedt megoldás az összes CISC processzornál. 16 KB, négy-utas csoport-asszociatív, utasítás-előrejelzővel ellátott utasítás cache-t valamint 8 KBos, dupla portos, négy-utas csoport asszociatív adat cache-t tartalmazott a processzor. Az utasítás-végrehajtás teljesen spekulatív módon történt. Az 1 KB-os ugrás-

előrejelzési tároló igen hatékonyan működött, téves előrejelzés esetén is csak 3 órajelnyi időt veszített a processzor. A K5 talán legfontosabb tulajdonsága, hogy Pentium kompatibilis. A processzorban természetesen megtalálható volt egy gyors lebegőpontos műveletvégző egység, és támogatta az energiatakarékos üzemmódot is. Az AMD azonban bukásként könyvelhette el ezt a fejlesztését, mivel későn hozta ki, és jelentős teljesítménykülönbség volt a Pentiumok, és a K5 között.

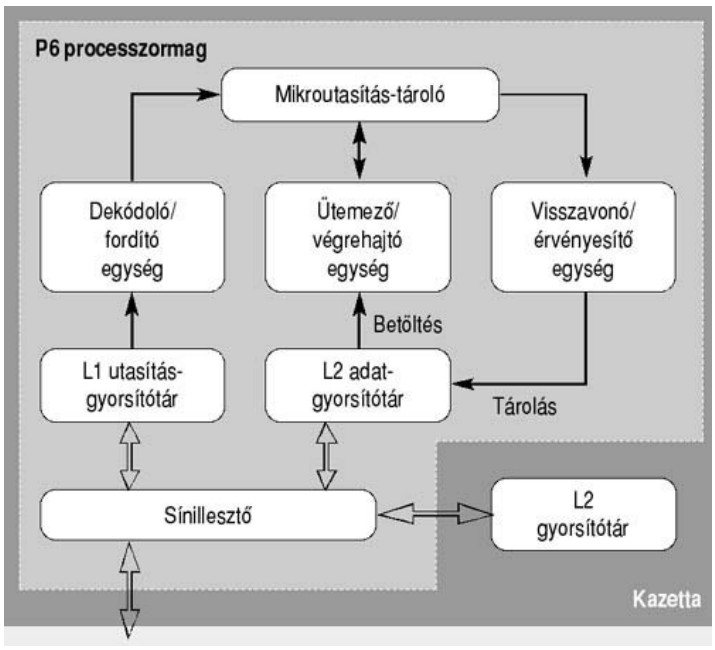
A hatodik generációs processzorok

1995-ben jelentette meg a piacon az Intel az első hatodik generációs processzorát, a Pentium utódját, a Pentium Pro-t. A processzor P6-os magra épült, 5,5 millió tranzisztort tartalmazott, és integrált második szintű (L2) cache-t (5.1 ábra). A címsint 36 bitesre növelték, ezáltal 64 Gbájt fizikai memória megcímzését tették lehetővé(5.2 ábra). További újdonság, hogy a Pentium Pro támogatja a többprocesszoros architektúra megvalósítását is. A Pentium Pro processzorral működő számítógépeket legtöbbször helyi hálózatok szervereként alkalmazták.

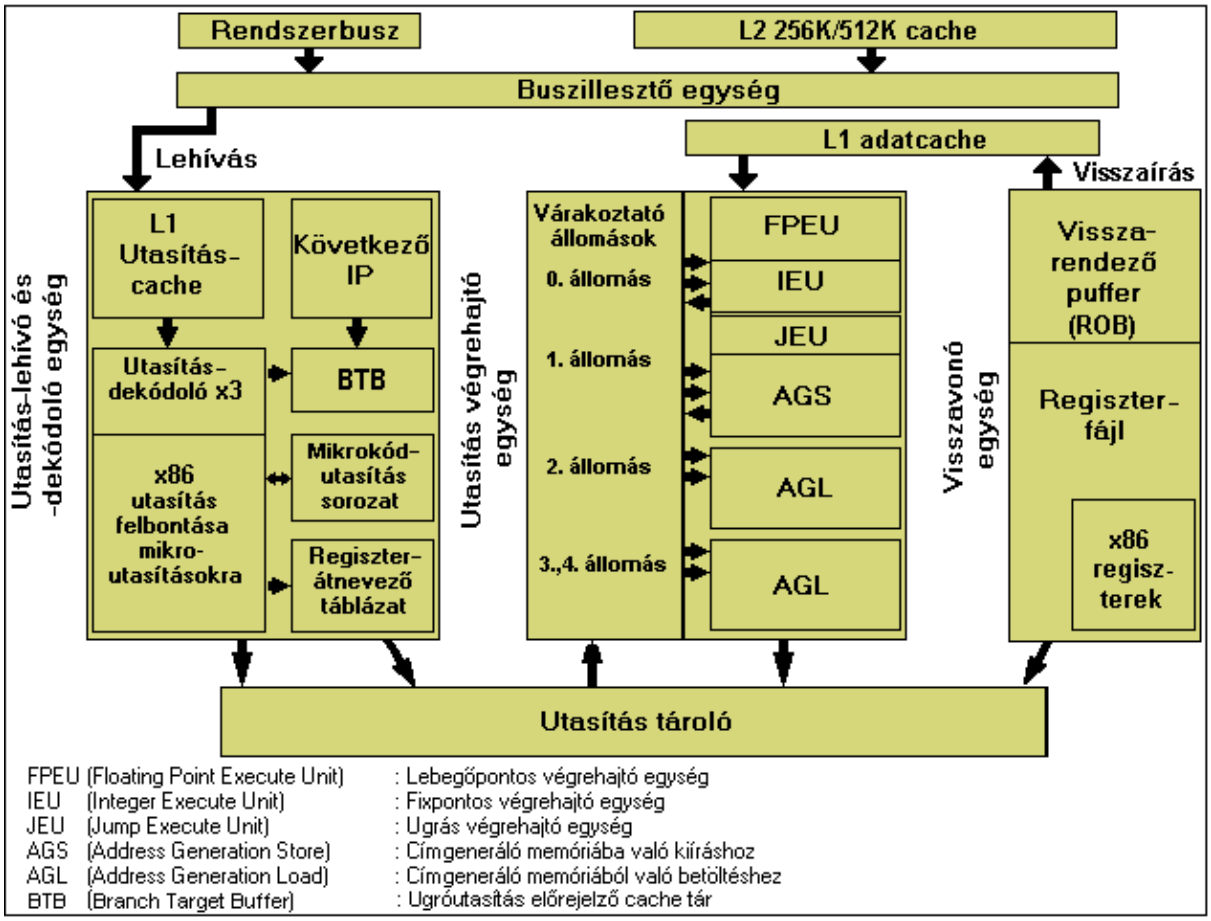
A P6 processzormag legfontosabb jellemzői:

- Az x86-os CISC utasítások lefordítása RISC mikroutasításokra;
- Spekulatív elágazás-feldolgozás, elágazás-történeti cachetár (BTB) 512 bejegyzéssel (BTB);
- Szuperpipeline 14 fokozattal (8 fokozat az előkészítésre: dekódolás, mikroutasításokra bontás, regiszterátnevezés, kibocsátás, 3 fokozat a végrehajtásra, 3 fokozat a befejezésre);
- 5 futószalag párhuzamos működése;
- Regiszterkészlet (általános) bővítése: 40 fizikai regiszter;
- Regiszterátnevezés és ROB alkalmazása.

A processzormag RISC processzorként működik, és ennek is köszönhetően, a Pentium Pro teljesítménye a 32 bites alkalmazások esetén kb. 50%-kal jobb a Pentium processzorénál ugyanazon az órajelen.



5-1 ábra. A P6-os processzormag



5-2. ábra: Pentium Pro processzor

A **Dynamic Execution (dinamikus végrehajtás) módszerét** a Pentium Pro processzorokban használták először. Ez három olyan feldolgozási technikát tartalmaz, melyek hatékonyabb működést eredményeznek:

- A Multiple Branch Prediction (többágú előrejelzés) segítségével 90%-os vagy még jobb találati aránnyal megjósolható, hogy melyik lesz a következő utasítás.
- A Data Flow Analysis (adatfolyam-vizsgálat) révén az utasításokat elemzés után a végrehajtás szempontjából ideális sorrendbe lehet rendezni. Az ideális sorrend eltérhet az eredeti sorrendtől, de a műveletek végeredménye ugyanaz lesz.
- A Speculative Execution (spekulatív végrehajtás) során a processzor végrehajtja azokat az utasításokat, amelyek valószínűleg következni fognak a sorban. Hogy melyek ezek, azt előtte a többágú előrejelzés határozza meg.

A processzor két független buszt (Dual Independent Bus) használt:

- Az egyik a processzor és a memória közötti adatátvitelt,
- a másik a külső cache és a processzor közötti kommunikációt végzte.

Mindkettőt egyszerre lehetett használni, emellett lehetővé teszi, hogy a külső cache teljesítménye a processzor sebességével együtt növekedjen.

A Pentium II-vel az Intel nem titkolt szándéka egy olyan processzor kifejlesztése volt, amely teljesítményben többet nyújt, mint a Pentium Pro, ennek ellenére a nagy vásárlóközönség számára is megfizethető.

A processzor egyesíti a Pentium Pro és MMX főbb tulajdonságait:

- Ebben a processzorban is P6 processzormagot használtak RISC mikroutasításokkal.
- A Pentium II tartalmaz egy második szintű cache-t, amely a processzor sebességének a felével működik. A cache és a processzor egy közös áramköri lapra került. A második szintű cache mérete 512 KB, amelyet az adatok és az utasítások megosztva használhatnak.
- Külön sínrendszeren valósul meg az L2 cache-el és külön sínrendszeren a rendszermemóriával való kommunikáció.
- A rendszersíneknél különválasztott adat- és utasítássín van.
- 32 bites operációs rendszer és 32 bites alkalmazás futtatására optimalizált MMX-utasításkészlet.
- 16 KB adat- és 16 KB elsőszintű utasítás cache.

5-1. Táblázat: A Pentium II processzorok különböző típusai

Típus	Kiadási év	Tranzisztor-szám	Gyártási technológia	Sebesség (MHz)
Klamath	1997	7,5 millió	0,28 µm	233 / 266 / 300
Deschutes	1998	7,5 millió	0,25 µm	333 / 350 / 400 / 450 / 500
Xeon I	1998	7,5 millió	0,25 µm	400 / 450

A Xeon-okat multiprocesszoros architektúrájú kiszolgáló gépekbe tervezték.

A Celeron processzorokat az Intel a Pentium II kategóriájú olcsó, nagy tömegek számára elérhető árú processzorként fejlesztette ki. A Pentium II nagy előállítási költségének az oka a második szintű cache volt, ezért ezt kezdetben kihagyták a Celeron processzorból, emiatt teljesítménye jóval kisebb lett. Egy 300 MHz-es Celeron processzor egyes alkalmazások futtatása esetén nem volt képes elérni egy Pentium 200 MMX processzor teljesítményét sem. Ezen a későbbiekben úgy javítottak, hogy az "A" jelzésű Celeronokba (Mendocino) mégiscsak beépítettek egy 128 KB méretű másodszintű gyorsítótárat.

A P II alapú Celeron processzorok fejlődését mutatja be az *5.2 táblázat*:

5-2. Táblázat: P-II alapú Celeron processzorok

Típus	Kiadási év	Tranzisztorok száma	Gyártási technológia	Sebesség
Covington	1998	7,5 millió	0,25 µm	266 / 300 MHz
Mendocino	1998	19 millió	0,25 µm	300 / 333 MHz
Mendocino	1999	19 millió	0,25 µm	366 / 500 MHz
Mendocino	2000	19 millió	0,25 µm	533 MHz

Az AMD a K6-ot az Intel az új Pentium MMX processzorának riválisaként, 1997-ben jelentette meg. A többi hatodik generációs processzorhoz hasonlóan az x86-os utasításokat emulálva hajtja végre, a mag RISC technológián alapszik. A K6 fő különlegessége, hogy műveleti egységei nem x86-, hanem RISC86- utasításokat hajtanak végre

A teljesítménynövekedést eredményező belső felépítésbeli tulajdonságok:

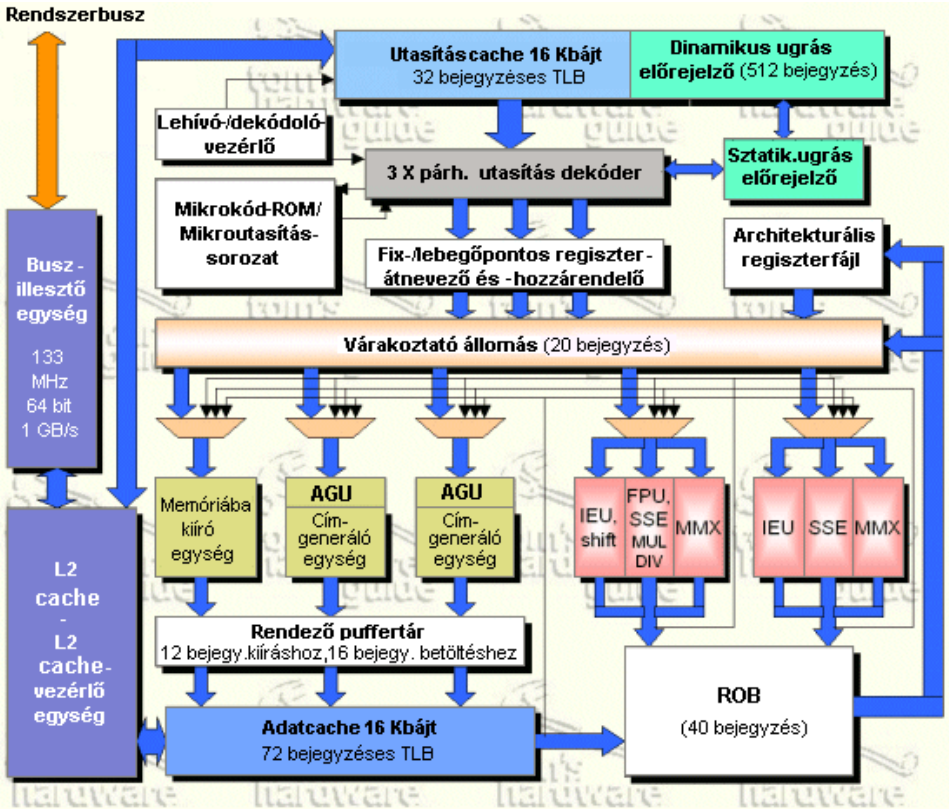
- Nagyobb elsőszintű cache. Az integrált elsőszintű cache méretét 64 KBra növelték, amelyből 32 KB az adatok, 32 KB pedig az utasítások átmeneti tárolására használható.

- Négy x86 utasítást dekódoló egységet tartalmaz a processzor, amelyek teljesítménye összehasonlítható a Pentium II és a Pentium Pro dekóderének teljesítményével.
- Az ugrás-előrejelzési tábla 8192 elemű. E táblázattal az előrejelzés hatékonysága az AMD szerint 95%-os.
- Egy MMX, két fixpontos és egy lebegőpontos végrehajtóegység.
- Az x86-os utasításkészletet kiegészítették 57 MMX utasítással.

A K6 második generációja, **AMD K6-2** 1998 májusában jelent meg. Elődjéhez képest a legnagyobb újdonság az **új 3DNow!** utasításkészlet, valamint a 100 MHz-es rendszerbusz volt. **A** 3Dnow! 21 darab új SIMD utasítást jelent, amelyeket a processzor vektorszámítógépként hajt végre, és amelyek meggyorsítják a 3D megjelenítést és más lebegőpontos művelete igénylő programokat. Ellentétben az Intel MMX utasításaival, amelyek csak egész számmal tudnak műveletet végezni, a 3DNow! lebegőpontos operandussal is. (A DirectX a 6-os verziójától teljes mértékben támogatja ezt a technológiát.)

A háromszintű cache technikát az asztali PC-k világában elsőként az AMD alkalmazta, az 1999-ben piacra került **K6-III** típusjelzésű processzorában

A **Pentium III** processzorok első változatát a Pentium II alapokra épülő Katmai-t, az Intel 1999-ben hozta ki, a Deschutes közvetlen utódaként.



5-3. ábra: A P-III processzor felépítése

A processzor legnagyobb újítása a kibővített MMX mellett egy új SIMD (**S**ingle **I**nstruction **M**ultiple **D**ata) utasításkészlet, SSE néven (**S**treaming **S**IMD **E**xtensions), valamint a javított memóriakezelés. Az új processzortípus kifejlesztésének legfontosabb célja a 3D grafikához (virtuális valóság modellezése, 3D-s játékok, CAD stb.) szükséges SIMD műveletek beépítése volt a processzor utasításkészletébe. Az SSE (Streaming SIMD Extension) utasításkészlet-bővítés lehetővé teszi, hogy a processzor a 3D grafikus műveleteket vektorszámítógépként (négy vektorkomponensre párhuzamosan) hajtsa végre. Az

új utasításkészlethez a játékok alkalmazkodtak a leggyorsabban, a Microsoft az SSE utasításkészlethez néhány hónapon belül módosította a DirectXAPI-t (DirectX Version 6.1). A processzorban 5 db futószalag működik, amely közül kettő vagy fixpontos vagy SSE (lebegőpontos) vagy MMX.

Az L2 cache-t 256 bit szélességű busz köti össze a processzorral, ami a korábbiakhoz képest (64 bit) négyszeresre növelte az adatátviteli teljesítményt.

A Pentium III processzor különböző változatait mutatja be a következő táblázat:

5-3. Táblázat: A Pentium-III processzor különböző változatai

Típus	Kiadási év	Tranzisztor-szám	L2 cache	Gyártási technológia	Sebesség
Katmai	1999	9,5 millió	512 KB	0,25 µm	450-600 MHz
Tanner	1999	9,5 millió	512K/1M/2MB	0,25 µm	500/550 MHz
Coppermine	1999	28,1 millió	256 KB	0,18 µm	533-800 MHz
Coppermine	2000	28,1 millió	256 KB	0,18 µm	850-1.300 MHz
Tualatin	2001	44,0 millió	256 KB	0,13 µm	1.133-1.200 MHz
Celeron II	2000	7,5 millió	128 KB	0,18 µm	533-1.100 MHz
Celeron (Tualatin)	2001	44 millió	256 KB	0,13 µm	1.000-1.400 MHz

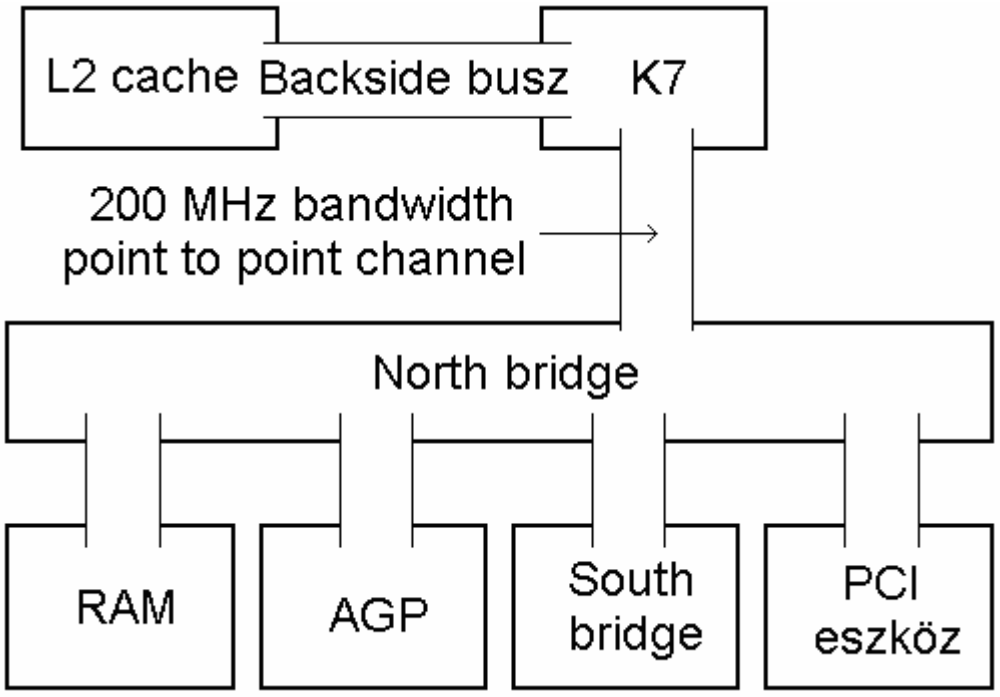
A hetedik generációs processzorok

A K7 (Athlon) *az AMD* első olyan terméke, mely nem az Intel architektúráját követi, hanem teljesen saját fejlesztésként valami egészen újat nyújt. Az Athlon néven megismert első hetedik generációs processzorát, a K7-et 1999 júniusában hozták ki. A processzor architektúráját teljesen átalakították a K6-hoz képest. Az Athlon-ban lévő 3DNow!-t kiegészítették 24 új utasítással. Ebből 12 az MMX utasításkészletet bővíti (beszédfelismerés, high-quality videó enkódolás/dekódolás), 7 pedig streaming utasítás (nagy multimédiás adatmozgás, Internetes plug-in-ek). Ez a 19 utasítás nagyon hasonlít (majdnem teljesen azonos) az SSE megfelelő utasításaival. Ami azonban sokkal érdekesebb, az az öt új DSP (Digital Signal Processing) gyorsító funkció (soft modem, soft ADSL, MP3, Dolby Digital); ezek nem találhatóak meg az Intel SSE készletében.

Az Athlon működésének fontosabb jellemzői:

- 3 db x86-os párhuzamos utasítás-dekódoló;
- 3 db lebegőpontos 15 lépéses futószalag, amelyek közül kettő vagy lebegőpontos vagy MMX vagy 3DNow! utasításokat dolgoz fel;
- 3 db fixpontos és 3 db címszámító 10 lépéses futószalag;
- 24/32 bejegyzéses L1 és 256 bejegyzéses L2 TLB (utasítás/adat);
- 72 bejegyzéses várakoztató állomás és függőségellenőrző egység.

Az Athlon processzorral az AMD egy teljesen új kommunikációs architektúrára tért át a DEC Alpha (21264) processzorának mintájára (5.4 ábra). Ehhez a DEC Alpha nevű RISC processzorából néhány rendszertechnikai megoldást is átvettek, így pl. az EV6 Alpha busz protokollt.

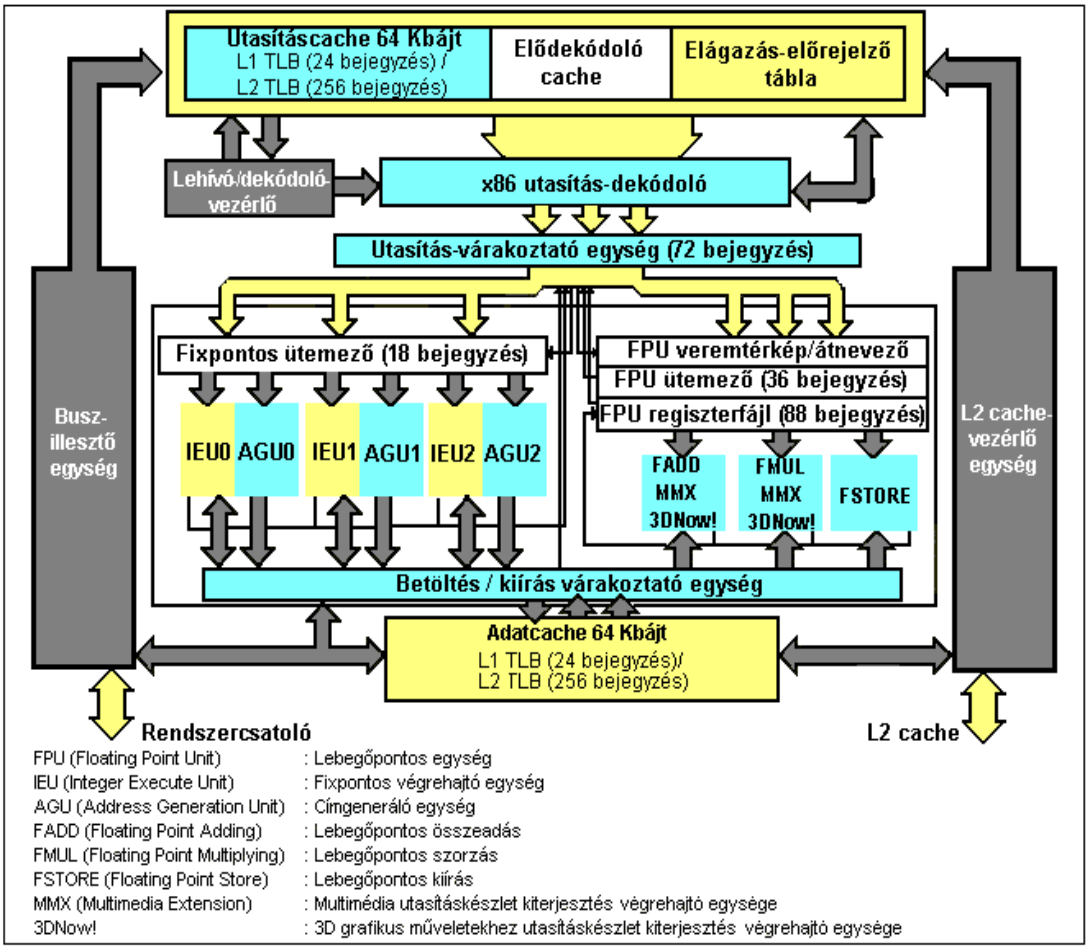


5-4. ábra: Az AMD K7 Athlon kommunikációs architektúrája

A processzor egy közvetlen pont-pont összeköttetéssel (200 MHz) tart kapcsolatot az alaplapi vezérlőáramkör-készlettel, a P6 architektúrában szokásos rendszerbusz kiküszöbölésével. Az 5.5 ábrán látható a K7 felépítése, az 5.4 táblázat pedig az AMD hetedik generációs processzorait foglalja össze.

5-4. Táblázat: Összefoglaló táblázat az AMD hetedik generációs processzorairól

Típus	Kiadási év	Órajel (MHz)	L2 cache
K7	1999/2000	500 -1000 között	512 KB
Thunderbird	2000/2001	650 -1400 között	256 KB
Palomino	2001/2002	1333 -1733 között	256 KB
Thoroughbred	2002	1467 - 2200 között	256 KB
Barton	2003	1800	512 KB
Duron (Spitfire)	2000/2001	650 - 950 között	64 KB
Duron (Morgan)	2001/2002	1000 - 1300 között	64 KB
Sempron	2004	2000	256 KB
AthlonXP	2004	2200	512 KB



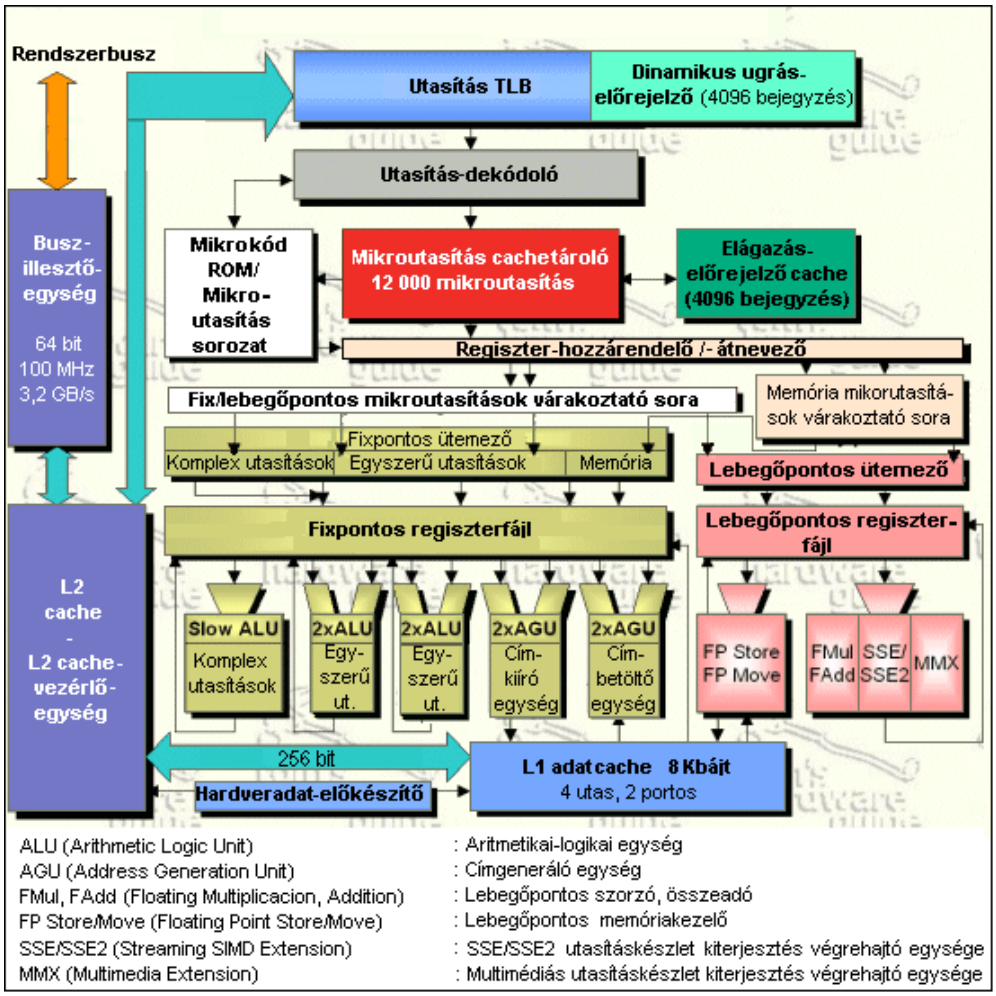
Forrás: http://www.amd.com

5-5. ábra: Az AMD K7 processzora

A Pentium 4 processzor első változatát Willamette néven az Intel 1,4 és 1,5 GHz órajel-frekvenciával 2000 novemberében dobta piacra. Az új hetedik generációs, 42 millió tranzisztort tartalmazó processzorral az Intel — még megmaradva a 32 bites keretek között — a 3D grafika és MPEG kódolás/dekódolás, a beszédfelismerés és rejtjelezés felgyorsítását tűzte ki célul.

A Pentium 4 Willamette nevű magja 0,18 mikronos gyártástechnológiával, és különböző processzorfoglalatokkal (Socket 423, Socket 478) készült. Utasításkészlete tovább bővült 144 (76 SIMD és 68 fixpontos) utasítással (a PIII SSE utasításkészletének a továbbfejlesztése). Ezt SSE2-nek (Streaming SIMD Extension2) vagy WPNI-nek (Willamette Processor New Instructions) nevezik. Ennek néhány fontosabb jellemzője:

- lehetővé teszi az XMM0-7 regiszterekben a 2x64 bites duplapontosságú lebegőpontos számok feldolgozását;
- az MMX pipeline az XMM regiszterekkel működik, és így lehetőség van 2x64 illetve 128 bites fixpontos számok feldolgozására is (Ez például a rejtjelezési algoritmusok miatt is fontos.)



5-6. ábra: A Pentium 4

A Pentium 4 működésének és felépítésének főbb jellemzői:

- Az L1 szintű utasításcache helyett egy, az utasítás-dekódoló mögé helyezett 12 KBos Execution Trace Cache tárolót használ a processzor, amely a RISC mikroutasításokat puffereli. (Az L1 adatcache maradt 8 KBos.)
- Az L2 cache 256 KBos és a csipre integrált, 256 bites adatúttal (Advanced Transfer Cache) a processzorhoz;
- Kétszeres sebességű ALU, amely külön 2x-es órajelet használ (Rapid Execution Engine);
- Hiper pipeline: 20 lépéses futószalag;
- Az elágazásbecslés javítása (30 %-kal jobb, mint a P III);
- 7 párhuzamosan működő futószalag: 3 fixpontos, 2 memóriacím-generáló, 2 lebegőpontos (beleértve az SSE/SSE2 és MMX pipeline-t is). Ezzel esetenként 6 utasítás is végrehajtható órajelenként.

A Pentium 4 processzor 400 MHz-es rendszerbuszra van felkészítve (Quad-Pumped 4x100), ami 3,2 Gbájt/sec átviteli teljesítményt jelent.

5-5. Táblázat: A Pentium 4 processzor különböző típusainak összefoglaló táblázata

Típus	Kiadási év	Órajel	L2 cache	Gyártási technológia
Willamette	2000/2001	1,4 - 2,0 GHz	256 KB	0,18 μ
Northwood	2002	2,0 - 3 GHz	512 KB	0,13 μ
Celeron (Willamette)	2002	1,7 - 1,8 GHz	128 KB	0,18 μ
Celeron (Northwood)	2002	1,4 - 1,5 GHz	256 KB	0,13 μ
Prescott	2003	1,8 – 3,2 GHz	1024 KB	0,09 μ

Nyolcadik generációs processzorok

Az Intel 64 bites architektúrájával kapcsolatban meg kell jegyezni, hogy az új architektúra nem egyszerűen az x86-os processzor-architektúra 64 bites változata, hanem a 64 bites RISC processzorok felépítését és működési elveit adaptálták, illetve fejlesztették tovább. Az EPIC (Explicitly Parallel Instruction Computing) technológia és az ennek megfelelő új 64 bites utasításkészlet mellett az x86-os program-kompatibilitás, csak külön hardver emulátor- és vezérlőegységgel biztosítható. Ezeknek, a processzoroknak például nincs valós üzemmódja. A régi x86-os programokkal a kompatibilitást firmware-emuláció biztosítja. A processzorba 320 darab (korábban csak a RISC processzorokra jellemző mennyiségű) regisztert építettek be (128 db fixpontos, 128 db lebegőpontos és 64 db az elágazáskezeléshez szükséges regiszter.)

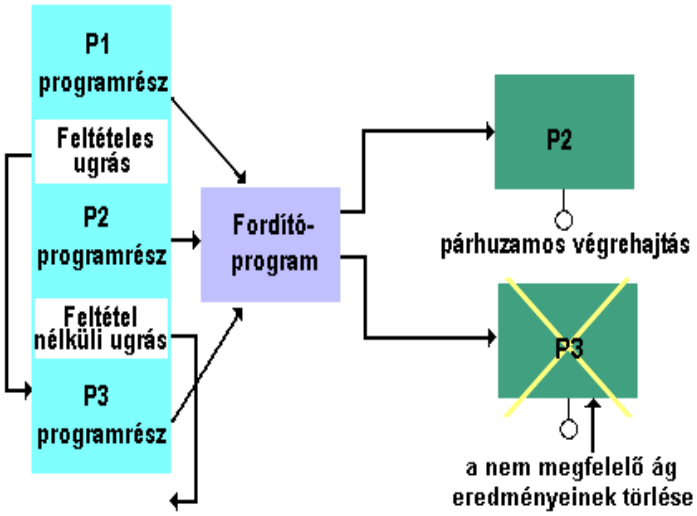
Az Intel 64 bites processzorának prototípusa **Itanium** néven 1999 augusztusában készült el 0,18 mikronos technológiával, 800 MHz-es órajellel. Ezen tesztelték a Microsoft új, 64 bites operációs rendszerét. Ennek ellenére a sorozatgyártásra csak a 2001-es évben került sor.

Az IA-64 architektúrájú processzorokat többprocesszoros szerverekbe, munkaállomásokba (szimuláció, 3D grafika) építik be. Ezzel az új architektúrával az Intel, a különböző RISC processzoros UNIX gépek piacát célozza meg.

Az IA-64-es architektúrájú 64 bites processzor felépítésére jellemző az EPIC (Explicitly Parallel Instruction Computing) architektúra. Ez azt jelenti, hogy a processzor működési eleve, a párhuzamos utasítás-végrehajtásra épül, párhuzamosan végrehajtható gépi kódú utasítás sorozatokat feltételez. A 64 bites Intel processzorok ISA (Instruction Set Architecture) utasításkészletének jellemzője, hogy a fordítás során a programozási nyelv fordítóprogramjának nem soros, hanem eleve párhuzamosított tárgykódot kell generálnia a processzor számára. Ez az eleve párhuzamos felépítésű kód jobb párhuzamos végrehajtási tulajdonságokkal rendelkezik, mint a korábbi 32 bites Intel processzorok soros felépítésű gépi kódja, amelyet a hardver próbált meg párhuzamosítani. Emiatt - mint a RISC felépítésű processzoroknál - az IA-64-es architektúrájú processzorok teljesítményét döntően a fordítóprogram hatékonysága határozza meg.

A statikus és dinamikus elágazás kezelés az IA-64-es architektúrában talán az egyik legfontosabb újítás. Az Intel 32 bites architektúrái dinamikus, azaz futás közben a hardver által megvalósított programirány-előrejelzéssel (elágazásjóslással) működnek, amely statisztikai számításokon alapul, 5.7 ábra. Az IA-64-es architektúra az elágazások előrejelzését első szinten a fordítóprogramra bízza (statikus elágazás előrejelzés). Statikus elágazás előrejelzéssel a fordító a forráskód tárgykódra való fordítása során nagyrészt ki tudja elemezni, hogy adott elágazásokból az egyik vagy másik irányba halad-e majd a vezérlés (ezzel a gyakoribb ágakat lehet valószínűsíteni, ill. a soha sem használt ágakat lehet kiküszöbölni). Így a processzornak a dinamikus hardverelágazás-előrejelzéshez képest kevesebbszer kell megjósolnia egy ugró utasítás kimenetelét — így kisebb a hibás előrejelzés

esélye is —, és a processzor erőforrásai az igazán bonyolult ágak hardver-kiértékelésére koncentrálhatók.



Forrás: http://www.intel.hu

5-7. ábra: ***Az elágazások mindkét ágának végrehajtása (Predication)***

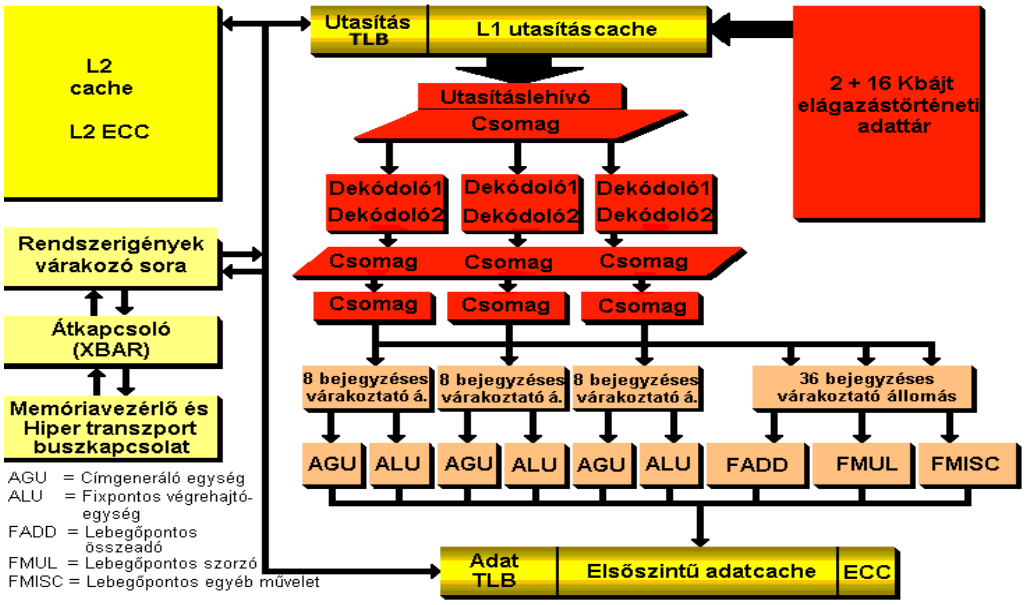
A futtatás folyamatosságát szolgálja - főként a bonyolult elágazásoknál - a Predication nevű rendszertechnikai megoldás. A Predication technika lehetővé teszi, hogy egy feltételes elágazásnál a processzor mindkét programágat egyszerre, párhuzamosan hajtsa végre.

5-6. Táblázat: Összefoglaló táblázat az Intel 64 bites processzorairól

Típus	Megjele-nési éve	Órajel (GHz)	Rendszer-busz sebessége	L1 (KB)	L2 (KB)	L3 (MB)	Gyártási techn. (µm)	Tranz. - szám (mill.)
Itanium (Merced)	2001. május	0,733; 0,8	266	32	96	2-4 off-chip	0,18	?
Itanium2 (McKinley)	2002. július	0,9; 1	400	16 + 16	256	1,5-3 on-die	0,18	221
Madison	2003	1,8-2,2	?	?	?	6 on-die	0,13	560 (350 L3)
Deerfield	2003	1,2-1,6	?	?	?	8 on-die	0,13	?
Montecito	2004	2,62-2,8	?	?	?	12 on-die	0,09	700-800

(megjegyzés: „?”: a gyártó nem adta ki az adatot.)

Az AMD 64 bites Hammer processzorainál alkalmazott fejlesztési stratégiája lényegesen különbözik az Intelétől, az x86-os program-kompatibilitás megvalósításában. Míg az IA-64-es architektúra "tiszta" 64 bites működésű, így a korábbi 32 bites szoftverek csak firmware emulációval futtathatók, az AMD Hammer, ún. x86-64 architektúrája egyaránt alkalmas 32 és 64 bites alkalmazások futtatására. Ezt az AMD fejlesztői úgy érték el, hogy a 64 bites címkezelést egyesítették a 48 bites virtuális és a 40 bites fizikai címtér kezelésével.



5-8. ábra: Az AMD 64 bites Hammer processzorának blokkjai

Az AMD x86-64 architektúrájú processzorainak jellemzőit a következőkben foglalhatjuk össze:

- a processzorban 6 egész és 3 lebegőpontos futószalag működik;
- a processzor támogatja a korábbi 32 bites processzorok utasításkészletének (x86, MMX, 3DNow, SSE és SSE2) megfelelő alkalmazások és operációs rendszerek futtatását;

5-7. Táblázat:

AMD Athlon 64	Clawhammer	Newcastle	Winchester	Venice	San Diego
Gyártási technológia	0,13µ	0,13µ	0,09µ	0,09µ	0,09µ
Foglalat	Socket 745	Socket 745/939	Socket 939	Socket 939	Socket 939
Memóriavezérlő	egycsatornás	egy/kétcsatornás	kétcsatornás	kétcsatornás	kétcsatornás
L1 cache mérete	128kB	128kB	128kB	128kB	128kB
L2 cache mérete	512kB/1MB	512kB	512kB	512kB	1MB
Alapfeszültség	1,5V	1,5V	1,4V	1,4V	1,4V
MMX+támogatás	van	van	van	van	van
3DNow!+támogatás	van	van	van	van	van
SSE támogatás	van	van	van	van	van
SSE2 támogatás	van	van	van	van	van
SSE3 támogatás	nincs	nincs	nincs	van	van
x86-64 támogatás	van	van	van	van	van

Az Intel tervezői korábban 10 GHz-ig skálázódó Pentium 4-ről álmodtak, azonban szembesültek a ténnyel, miszerint az igen hosszú életűre tervezett architektúra nem fogja beváltani a hozzá fűzött reményeket, hiszen a fogyasztás és a melegedés gátat szab az órajel korlátlan emelésének. A Prescott lett az eddigi legmelegebb, legtöbbet fogyasztó processzor. A család leggyorsabb tagjainak fogyasztása elérte a 150-160 wattot is, így az órajel gyorsítás megállt 3,8 GHz-nél. Az Intel és az AMD is a teljesítménynövelés lehetőségét a továbbiakban a processzor magjának többszörözésében látja. Így az AMD az Athlon X2,es az Intel pedig a kétmagos P4-et a Pentium D, vagy más néven Smithfield-et készítette el, amelyet a 0,065

mikronos gyártástechnológiára való átállás eredményeként nemsoká követett a Presler. A következő táblázatunk összehasonlításra ad lehetőséget az Intel processzorairól.

5-8. Táblázat

Processzor neve	**Pentium 4 5xx**	**Pentium 4 6xx**	**Pentium 4 Extreme Edition**	**Pentium D és EE (0,09 mikron)**	**Pentium D és EE (0,065 mikron)**
Processzormag neve	Prescott	Prescott 2M	Prescott 2M	Smithfield	Presler
Órajel	2,8-3,8 GHz	2,8-3,8 GHz	3,73 GHz	2,8-3,2 GHz	2,8-3,46 GHz
Rendszerbusz	800 MHz	800 MHz	1066 MHz	800 MHz	800 / 1066 MHz
Gyártás-technológia (mikron)	0,09	0,09	0,09	0,09	0,065
Tranzisztor (millió)	125	169	169	230	376
Magméret (mm^2)	112	135	135	206	162
L1 cache	16 kB adat (8 utas), 12k uop trace cache	16 kB adat (8 utas),12k uop trace cache	16 kB adat (8 utas),12k uop trace cache	16 kB adat (8 utas),2 x 12k uop trace cache	16 kB adat (8 utas),2 x 12k uop trace cache
L2 cache	1 MB (8 utas)	2 MB (8 utas)	2 MB(8 utas)	2 x 1 MB(8 utas)	2 x 2 MB(8 utas)
L3 cache	Nincs	Nincs	Nincs	Nincs	Nincs
SIMD	MMX, SSE, SSE2, SSE3	MMX, SSE, SSE2, SSE3	MMX, SSE, SSE2, SSE3	MMX, SSE, SSE2, SSE3	MMX, SSE, SSE2, SSE3
EM64T	Modelltől függően támogatja	Támogatja	Támogatja	Támogatja	Támogatja
Execute Disable Bit (NX)	Támogatja	Támogatja	Támogatja	Támogatja	Támogatja
Enhanced Speedstep (EIST)	Nem támogatja	Támogatja	Nem támogatja	Támogatja	Támogatja
Intel Virtualization Technology	Nem támogatja	Nem támogatja (csak 6x2)	Nem támogatja	Nem támogatja	Támogatja

Az 5.8-as táblázatban szereplő eddig nem tárgyalt fogalmak:

- **EM64T**a hivatalos megnevezése a 64 bites technológiájának, amely segítségével a Xeon és a Pentium 4 E processzorok nagyobb méretű memóriát címezhetnek meg, és speciálisan megírt 64bites kód futtatására is képesek.
- **Enhanced speedstep** az Intel fogyasztás és hőtermelés csökkentő technológiája. Lényege, hogy a processzor automatikusan összehangolja órajeleinek frekvenciáját az igénybevétellel, és ezzel minimalizálja az „üresjárathoz” tartozó energia felvételt.
- Az **Execute Disable Bit** biztonsági funkció a káros programok ellen. Megfelelő operációs rendszerrel, a processzor képes védekezni bizonyos túlcsordulásos (buffer overflow) támadások ellen, mivel osztályozza a memóriát a szerint, hogy program futhat-e ott vagy sem. Amennyiben a féregvírus megpróbál védett helyre programkódot beilleszteni, a processzor megállítja a program futását.

- **Intel Virtualization Technology** megengedi egyidejűleg két külön operációs rendszer futtatását virtuálisan, ugyanazon a processzoron egy időben.

Az Intel P4 processzora kapcsán érdemes megjegyezni, hogy az un. trace cache-t használja fel a CISC utasítások RISC –re fordításának tárolására. Ehhez a processzorban egy huzalozott fordító egység gondoskodik az utasítások konverziójáról. A végrehajtó egységek egy belső kódmemóriából dolgoznak (trace cache), ahol már tiszta RISC utasítások találhatóak. (Egy komplex x86-os utasítás néha akár több tucat RISC utasítássá fordul le).

A processzorok fejlesztése felgyorsult ütemben halad, és a tendenciát figyelve évenként több új típus is megjelenik.

5.1.2. Alaplap

Az alaplap szerepe egy PC-s konfiguráció kialakításában alapvetően meghatározó. Típusa közvetlenül befolyásolja:

- a PC fizikai felépítését,
- a számítógéprendszer teljesítményét,
- a számítógép konfigurációjának kialakítását, annak továbbfejleszthetőségét.

Az alaplapi sínrendszerek, áramkörkészletek (chip-set), az alaplapra integrált periféria- és háttértárvezérlések, az alaplapon található memória- és processzorfoglalatok, kártyahelyek, portok megszabják többek között:

- az alaplapba helyezhető processzor(ok) típusát, sebességtartományát;
- a felhasználható RAM modulok típusát, darabszámát, a memória maximális kapacitását;
- az alaplap kártyahelyeibe helyezhető vezérlő- és adapterkártyák típusát, maximális számát;
- a számítógépes konfiguráció háttértároló- és perifériaegységeinek típusát és darabszámát;
- az alaplaphoz használható számítógépházat és tápegységet.

Ebből két nagyon gyakorlatias következetés rögtön adódik:

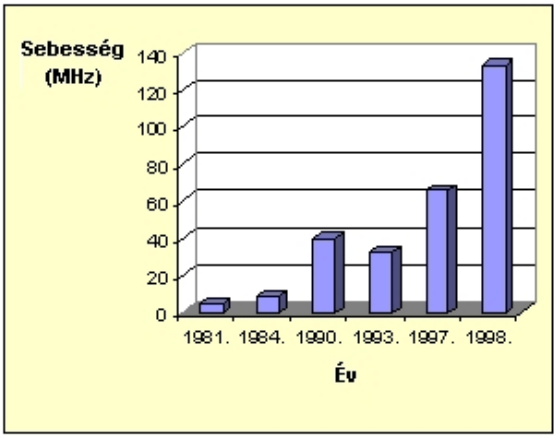
- ha számítógépet vásárolunk, akkor célszerű alaposan megvizsgálni az alaplap műszaki paramétereit, mert ezek később meghatározó módon befolyásolják a konfiguráció bővítésének (például processzorcsere, memóriabővítés) lehetőségeit és pénzügyi vonzatait;
- ha PC-s konfigurációnkat bővíteni, módosítani szeretnénk (például egy újabb lemezegységet szerzünk be), mindig alaposan meg kell vizsgálni, hogy elképzelésünk az alaplap műszaki paramétereivel összhangban van-e?

Az alaplapok fejlődésének fontosabb tendenciái

5-9. Táblázat: ***Alaplapi sínrendszerek fejlődése***

Számítógéptípus	Év	Buszszélesség (bit)
PC	1981.	8
XT	1983.	8
286	1984.	16
386	1985.	32
486	1989.	32
Pentium	1993.	64
Pentium II	1997.	64
Pentium III	1999.	64
Pentium 4	2000.	64

A sínrendszerek adatátviteli teljesítménye egyrészt az egyre nagyobb buszfrekvencia, másrészt a buszon egy lépésben átvitt egyre több adatbit miatt nőtt. Ezek értéke az eltelt húsz év alatt többszörösére nőtt.



5-9. ábra: Alaplapi sínrendszerek sebessége

A sínrendszerek nem alkotnak az alaplapon jól látható és fizikailag elkülönülő egységet. Ezek vezérlőáramköreit az alaplapi áramkörkészlet tartalmazza, a buszok vezetékrendszere az alaplapba szervesen beépített.

A jelenleg használatos alaplapok többségénél három típusú sínrendszer kártya csatlakozó helyeit találhatjuk meg:

- a perifériák, háttértárak valamint a hálózati kártyák csatlakoztatását biztosító PCI sín kártyahelyei;
- a monitorvezérlő kártya és a rendszermemória elkülönített, gyors kommunikációját biztosító AGP sín kártyahelye, amelyhez értelemszerűen a monitorvezérlő kártyát kell csatlakoztatni;
- a legújabb alaplapokon pedig a gyors perifériak csatlakoztatására a PCI-X kártyahelyet.

Az elmúlt években egyre "intelligensebb" alaplapok és I/O eszközvezérlések (kártyák) jöttek létre, amelyekkel a konfigurációnak a módosítása, a Plug and Play (PnP), vagyis a "Csatlakoztasd és használd" elv alapján automatizáltan történhet.

A PnP egy szabványcsoport, amely lehetővé teszi, hogy a PC-konfiguráció módosítása esetén az I/O eszközök használatához szükséges beállítások automatikusan, emberi beavatkozás nélkül megtörténjenek. A PnP támogatására felkészített hardvereszköz a BIOS ill. az operációs rendszer kérdése esetén azonosítja magát, és megadja a számára szükséges kiszolgáló rutinok adatait, ami alapján kijelölik számára a megszakítást kérő vezérlő vonalat (IRQ), az I/O címtartományokat stb. Az ehhez szükséges adatokat az eszköz ROM tárolója tartalmazza.

A szabványnak megfelelően működő eszközt számítógépünkhöz csatlakoztatva, az operációs rendszer felismeri az új hardvert és telepíti a megfelelő fájlokat. (Nem biztos, hogy az optimális beállításokat használja, de az eszköz működik.)

A PnP mellett napjaink egyre intelligensebb alaplapjait az energiatakarékos üzemmód, a biztonsági hardware monitoring, a hálózati üzemeltetés egyre jobb menedzselhetősége is jellemzi.

Az alaplapok korábban különálló funkcionális áramköreit (megszakítás-vezérlő, DMA-vezérlő stb.) egyre inkább integrált formában állítják elő, és így jönnek létre napjaink alaplapi vezérlőáramköri készletei (chipset).

Az alaplapra egyre több olyan lemezinterfészt és perifériavezérlést integrálnak, amelyek korábban csak bővítőkártya formájában voltak használatosak. Napjainkban már olyan alaplapok is léteznek, amelyek 3D gyorsítási képességű monitor, SCSI és RAID vezérlőket is tartalmaznak. Az alaplapra integrált perifériák csatlakozói is az alaplapon vannak (pl.:USB, hangkártya, Infravörös csatoló stb.)

Az alaplapi processzorfoglalatok meghatározzák a processzor kivezetéseinek vagy lábainak számát (pins), a lehetséges processzor órajelet és az alkalmazható feszültségszintet.

Az alaplap fontos jellemzője, hogy mennyire támogatja a számítógép üzembiztos és energiatakarékos működtetését. Az ezzel összefüggő jellemzők a következők:

- **hardware monitoring** vagy működési felügyelet biztosítja, hogy az alaplap – megfelelő programmal – automatikusan és folyamatosan figyelemmel kísérje saját működését, időben jelezze a felhasználó számára a hardver meghibásodásait (például túlmelegedés), megakadályozza a végrehajtás alatt álló programok adatállományának elvesztését. Ennek érdekében a korszerű alaplapok mérik például a processzor és az alaplap hőmérsékletét, az alaplaphoz csatlakozó hűtőventilátorok fordulatszámát és a tápegységtől kapott feszültségszinteket. A processzor túlmelegedése esetén – a felhasználó értesítése mellett - csökkentik az órajel-frekvenciát.
- az alaplapok többsége hálózatba kötött számítógépek esetén a hardverműködés **távoli felügyeletét** is lehetővé teszi a rendszergazda számára. (Ilyen lehetőséget biztosít például a LDCM = LAN Desk Manager program.)

5.1.3. A BIOS és szerepe

A BIOS (Basic Input/Output System - alapvető ki-/beviteli rendszer) a legelemibb ki-beviteli funkciókat ellátó szoftver, amely minden PC-ben megtalálható. A BIOS szoros egységet képez a hardver elemekkel, éppen ezért nem is szofvernek, hanem firm-ware-nek szokták nevezni. A BIOS nem más, mint inicializációs rutinok és primitív eszközmeghajtók gyűjteménye.

A rendszer BIOS-on az alaplaphoz tartozik, de a bővítő kártyák is rendelkezhetnek BIOS-okkal, amik speciális egységek vezérlését végzik (pl. LAN adapter — Boot eprom; SCSI vezérlő — SCSI BIOS, stb.). Ezeken kívül minden rendszer tartalmaz egy billentyűzet-vezérlő BIOS-t (Keyboard Controller BIOS) is a billentyűzet-illesztőben.

Bár a PC-kben több BIOS is található, a ROM BIOS szó alatt általában specifikusan a rendszer-BIOS-t (a továbbiakban BIOS) szokás érteni.

A BIOS elsődleges feladata — szoftver-megszakításokon keresztül — olyan funkciók nyújtása, melyek segítségével egyszerű műveletek végezhetők el, mint olvasás vagy írás a merevlemezre, a hajlékonylemezes meghajtóra vagy a képernyőre.

Ezen rutinok jelentősége absztraktságukban rejlik: olyan eszköz-független szolgáltatásokat bocsátanak az operációs rendszer és a programok rendelkezésére, melyek a rendszerben installált konkrét eszköz típusától függetlenül, minden környezetben egységes módon teszik lehetővé a minden egység által támogatott, de amúgy különböző módon kiváltható funkciók elérését. A gyakorlatban ez azt jelenti, hogy például a video-megjelenítő típusától függetlenül, ugyanazzal a BIOS funkcióhívással lehet egy karaktert kiírni a képernyőre, annak ellenére, hogy például a különböző adapterek video-memóriája eltérő címeken helyezkedik el, így közvetlen elérésük esetén nem lehetne — ebből a szempontból — egységesen kezelni őket.

E technika részint jelentősen csökkenti az alkalmazások méretét - hiszen azokat nem kell felkészíteni az összes ismert, de esetleg eltérő programozású egység kezelésére - másrészt lehetőséget biztosít a rendszer, az alkalmazások számára "láthatatlan" bővítésére, átalakítására, esetlegesen emulációk közbeiktatására.

Az újabb alaplapokon a BIOS általában ún. Flash-EPROM-ban van tárolva. A Flash-EPROM elektronikus úton - meglehetősen gyorsan - törölhető és újraírható memória-egység. Ezen BIOS-ok előnye, hogy időközben megjelenő újabb változataik a ROM modul fizikai kicserélése helyett, egy egyszerű segédprogram segítségével betölthetők.

A számítógép bekapcsolása vagy hidegindítása (RESET) után a processzor a vezérlést a 0FFFF0h fizikai címre adja. A memória ezen területére a ROM-BIOS van betükrözve. A ROM-BIOS POST (Power-On Self Test - bekapcsolási önteszt) ezek után a következő műveleteket végzi el:

- letiltja a megszakításokat (köztük az NMI-t is, mert a memóriacellák bitjei véletlenszerűen állítódnak be a reset után, és ez azok elérésekor "paritáshibát" okozhat),
- teszteli a flageket és a CPU egyéb regisztereit,
- megvizsgálja a ROM-BIOS ellenőrzöösszegét (checksum),
- engedélyezi a megszakításokat,
- inicializálja és teszteli a DMA vezérlőt,
- ellenőrzi a memória első 64KB-ját (a megszakítás-vektor táblázat miatt),
- inicializálja és teszteli a megszakítás-vezérlőt és beállítja a 10h-17h BIOS megszakításokat,
- rendszer-konfiguráció (megjelenítő, memória, stb) megállapítása,
- inicializálja és teszteli a CRT-kontrollert, a video-memóriát és a video-BIOS-t,
- inicializálja és teszteli a programozható időzítőt,
- inicializálja, teszteli majd engedélyezi a billentyűzetet,
- beállítja a hardver megszakítás-vektorokat,
- memória tesztelése (kivéve, ha a CMOS-ban a reset word értéke 1234h),
- a C8000h-EFFFFh közti területen ROM-bővítéseket keres és ha talál elindítja azokat,
- inicializálja és teszteli a floppy- és merevlemezes meghajtó(ka)t, ha van(nak),

- megkeresi és inicializálja a soros és párhuzamos illesztőket,
- engedélyezi az NMI-t,
- a CMOS–ban beállított sorrendben az első hajlékony- vagy merevlemezes meghajtóról, CD-ről, vagy hálózatról megpróbálja betölteni a boot-szektort ill. a partíciós táblát.

Minden teszt megkezdése előtt a diagnosztikai portra a tesztnek megfelelő kódot ír ki. Ez a kód egy 16-bites (word) szám, melynek felső 8 bitje az egységet azonosítja, míg az alsó nyolc bitje a teszt eredményét tartalmazza (00h - ha a teszt sikeres volt).

A bővítőkártyákon elhelyezhető BIOS rutinokat tartalmazó ROM egységek adnak lehetőséget a csatlakoztatott eszköz speciális igényeinek kiszolgálására, így egy újabb csatolókártya a „hátán hordozva” az adott egység BIOS-át is a rendszerbe illeszti. A bővítő ROM-okat a rendszer-BIOS a POST során keresi meg és inicializálja. A ROM-ok a C0000h-DFFFFFh memória-tartományban helyezkedhetnek el. A bővítő-BIOS inicializációs rutinja az egység alaphelyzetbe állítása és a megfelelő megszakításvektorok esetleges átirányítása után visszaadja a vezérlést a POST rutinnak, ami tovább folytatja a bővítések keresését.

A **BIOS-t** olykor összekeverik a CMOS-szal. A CMOS valójában a BIOS beállításainak tárolásához használt memóriachip gyártási technológiája. A BIOS beállításai a legtöbb hardveregységen — az alaplapot kivéve — nem változtathatók.

Az alaplap azonban egy olyan kitűntetett alkatrész, amelynek bonyolultsága és sokirányú feladatai miatt szükség van a beállíthatóságra.

BIOS Setup programba belépés különböző módon történhet egyes géptípusoknál. A legáltalánosabban a bekapcsolás után a DEL billentyű megnyomásával. A BIOS gyártójától függően más-más képernyővel jelentkezik. Általában billentyűvel de van egérrel kezelhető is. A menüpontok beállításai általában számok, vagy egy „kétállású kapcsoló” állapotainak egyike (például enabled/disabled - engedélyezve/tiltva). A menüpontokat tematikus oldalakra osztva tárja elénk a BIOS beállítóprogramja.

A **valós idejű óra és a CMOS** az alaplapok mindegyikén megtalálható. Az eredeti PC-kben nem volt a gép kikapcsolt állapotában is működő valós idejű óra (RTC - Real Time Clock), így ahhoz, hogy a pontos időt és dátumot mutassa, a felhasználónak a gép minden egyes bekapcsolásakor meg kellett adnia azt. (E miatt kérdezi meg a DOS máig is a pontos időt és dátumot, ha nincs CONFIG.SYS és AUTOEXEC.BAT fájlunk.) A gép ezután a programozható időzítő áramkör (8253) segítségével már "fenn tudta tartani" a pontos időt - kikapcsolásáig.

Bár már az XT-khez is fejlesztettek ki ezt a problémát áthidaló bővítőkártyákat, a valós idejű óra csak az AT kategóriájú gépekben jelent meg standard kiegészítő áramkörként. Az idő tárolásához egy kis kapacitású, CMOS technológiával készült - és ennek megfelelően kis teljesítményfelvételű - akkumlátorral vagy elemmel táplál RAM memóriát alkalmaznak, amely így a gép kikapcsolt állapotában is működőképes marad. Ugyanez az áramforrás látja el a memóriaegységgel egybeépíttet órajelgenerátort is, amely az idő számlálásáról gondoskodik. Az óraáramkör az aktuális dátum és idő karbantartásán kívül egy tetszőleges "ébresztési" időpont bekövetkeztekor, az időadatok frissítésekor, valamint periodikusan is képes megszakítás generálására a 8. szinten (IRQ8).

A CMOS memóriaegységek mérete 64 ill. 128 bájt lehet. Mivel azonban az óraáramkör ebből csak az első 14 bájtot használja, a fennmaradó cellákat a számítógép konfigurációs adatainak tárolására alkalmazzák, ezáltal kiváltva a PC-k és XT-k esetén alkalmazott, kissé nehézkes DIP-kapcsolós megoldást.

A CMOS memória tartalma három csoportra bontható:

- az RTC által használt cellák,
- a szabványos konfigurációs területek és
- a (BIOS)-gyártó-specifikus konfigurációs adatok.

A valós idejű óra adatait minden rendszer ugyanúgy használja fel és értelmezi. A szabványos konfigurációs adatok értelmezése gyakorlatilag szinte minden BIOS típus esetében megegyezik, csak az eredetileg nem definiált bitek értelmezésében ill. a merevlemez-típuskódok értelmezésében adódhatnak eltérések. A BIOS-specifikus konfigurációs adatok értelmezése mindig magától a renszer-BIOS-tól függ, feldolgozására csak a beépített BIOS típusának ismeretében van lehetőség.

A BIOS beállítási lehetőségeit egy Award gyártmányú BIOS-on keresztül mutatjuk be. A Setup program konkrét használatához fontos, hogy csak az alaplap gyártója által kiadott kézikönyv áttanulmányozása után érdemes hozzákezdeni.

A menűpontok:

Standard CMOS Setup

- Date: a rendszerdátum beállítása;
- Time: a rendszeridő beállítása;
- IDE Primary Master: az elsődleges IDE eszköz (általában merevlemez) adatainak beállítása;
- Floppy Drive A: első hajlékonylemezes meghajtó paraméterei.

Features setup

E menüpont alatt a számítógép teljesítményével összefüggő paramétereket lehet megadni. Ilyenek például:

- Internal cache: Itt tudjuk engedélyezni (enable), tiltani (disable) a processzorba integrált elsőszintű gyorsító tár használatát.
- External cache: A második szintű cache használatát tudjuk engedélyezni vagy tiltani.
- Boot sequence: Itt határozhatjuk meg, hogy a BIOS a POST lefutása után milyen eszközökön és milyen sorrendben keresse az operációs rendszert. Az alapbeállítás szerint a hajlékonylemezes meghajtóról (A) történik meg az operációs rendszer betöltése. Ha itt nem található adathordozó, akkor a BIOS a C: meghajtón keresi az operációs rendszert.
- Video BIOS shadow: Ha engedélyezzük ezt a beállítási pontot, akkor a monitorvezérlő kártyán található video-BIOS átmásolódik a rendszermemóriába, ami a képernyőkezelés felgyorsulását eredményezheti.

Chipset features setup

A Chipset features setup menüpont alatt található beállítások az alaplapi vezérlőáramkör-készletre vonatkoznak. Itt különösen nagy a veszélye annak, hogy a paraméterek helytelen megadása miatt a számítógép működésképtelen vagy instabil lesz.

System BIOS Cacheable

A ma használatos rendszerek megengedik, hogy a rendszer BIOS műveleteit egy ún. árnyékmemória felhasználásával gyorsítsuk fel. Ez azt jelenti, hogy a lassú ROM-ból a gyorsabb rendszermemóriába átmásoljuk a BIOS-t, ahol annak funkciói sokkal gyorsabban elérhetők. A beállítás alapértelmezés szerint engedélyezett, azonban ha a rendszer működésében hibát észlelünk, akkor megpróbálkozhatunk ennek a paraméternek a letiltásával.

Power Management Setup

Ebben a csoportban található paraméterekkel tudjuk beállítani az automatikus tápellátással és az energiatakarékos üzemmóddal kapcsolatos tulajdonságokat.

PnP/PCI Configuration

Ebben a csoportban található paraméterek segítségével tudjuk konfigurálni a PCI sínt ill. a Plug and Play eszközöket.

Video Power Down Timeout

E paraméter beállításával meghatározhatjuk, hogy ha nem használjuk a számítógépet, akkor a monitor mennyi idő múlva kapcsoljon át energiatakarékos üzemmódba.

Integrated Peripherials

E menüpont alatt az alaplapra integrált perifériákkal kapcsolatos paramétereket tudjuk beállítani. Például Integrated Floppy Disc Controller paraméterrel tudjuk tiltani (disabled) ill. engedélyezni (enable) az alaplapra integrált hajlékonylemezes meghajtóvezérlő áramkörét. Ha ez a vezérlő egy perifériakártyán már megtalálható, akkor az alaplapra integrált vezérlő működését itt kell letiltani.

Supervisor/User password

E menüpont alatt megadott jelszavakkal a számítógépünket megvédhetjük az illetéktelen használóktól. Ha beállítunk egy jelszót, akkor annak ismerete nélkül nem lehet elindítani a számítógépet és nem lehet belépni a BIOS Setup programjába. Általában két féle jelszót lehet megadni, egy rendszergazdai (supervisor) és egy felhasználó (user) kulcsszót.

Auto Configuration and Defaults

Abban az esetben, ha nem szeretnénk a hardver beállításaival foglalkozni, vagy nem szeretnénk valamit elrontani a hibás beállításokkal, használhatjuk a BIOS alap beállításait. Az automatikus beállítás nem változtatja meg a Standard Setup beállításokat, mivel ezeket minden esetben a felhasználónak kell beállítania (például időt, dátumot, hajlékonylemezes meghajtók számát és típusát).

Exit Setup

Itt tudunk kilépni a BIOS Setup programjából.

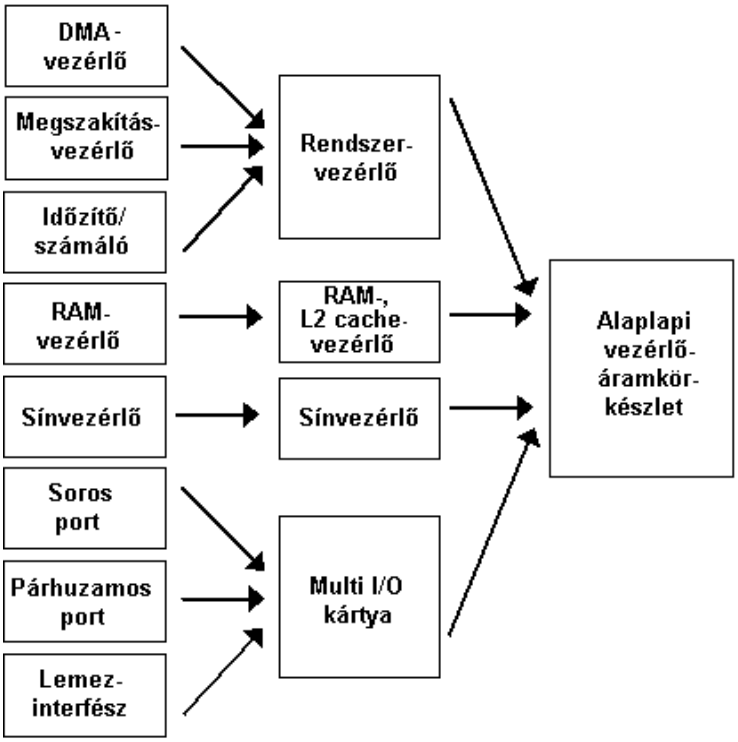
Save and Exit Setup

Ha ezzel a menüponttal lépünk ki a BIOS Setup programjából, akkor az a változtatásokat elmenti. Vagy az 'Y' (Yes/Igen) vagy 'N' (No/Nem) pontot kell kiválasztanunk.

5.1.4. A chip-set

Lapkák egy csoportja, amelyek egymással valamilyen módon összeköttetésben állnak, és együttesen látnak el összetett feladatokat.

Már a 286-os PC-k fejlesztése során felismerték, hogy az alaplap szabványos vezérlőáramköreit célszerű lenne berendezés-orientált áramkörökbe integrálni. A funkcionális vezérlőáramkörök egybe integrálása egyúttal azt is jelentette, hogy a PC-k világában minden egyes fejlettebb rendszertechnikai megoldás bevezetéséhez az alaplapi áramkörkészletek új generációját is létre kellett hozni. Így a chip-setek fejlődése a processzorokéval és buszrendszerekével összekapcsolódott, azzal párhuzamosan folyt.



5-10. ábra: ***A chip-set-ek kialakulása***

A chip-set tartalmazza a számítógép működéséhez elengedhetetlen vezérlőket. Az első generációs alaplapokon ezek a vezérlők még külön chip-ekben kaptak helyet. Az idő folyamán fokozatosan integrálódtak egyetlen (vagy két) chip-be. A mai korszerű chip-készletek általában egy, kettő vagy három chip-ből állnak.

A Triton 430TX típusú chip-set két tagját nevezték először **Northbridge-**nek és **Southbridge-**nek azaz északi és déli hídnak, (utalva a topográfiai elhelyezkedésükre az architektúra sematikus rajzán, fizikailag különválasztva a memória-/rendszerbusz vezérlés (MTXC) és az I/O eszköz- és buszvezérlés (PII X4) funkcióit.

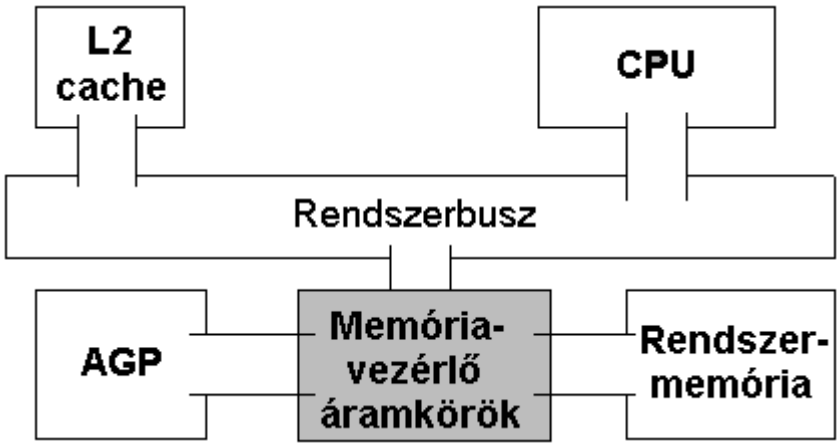
A memória és az I/O vezérlőáramkörök feladatai

A hatodik generációs processzorok mindegyik típusához egy meghatározott alaplapi áramkörkészlet tartozik. Funkcionálisan ezek a vezérlőáramkörök két csoportba oszthatók:

- memóriavezérlő áramkörök
- I/O vezérlő áramkörök

A memóriavezérlő áramkörök feladatai:

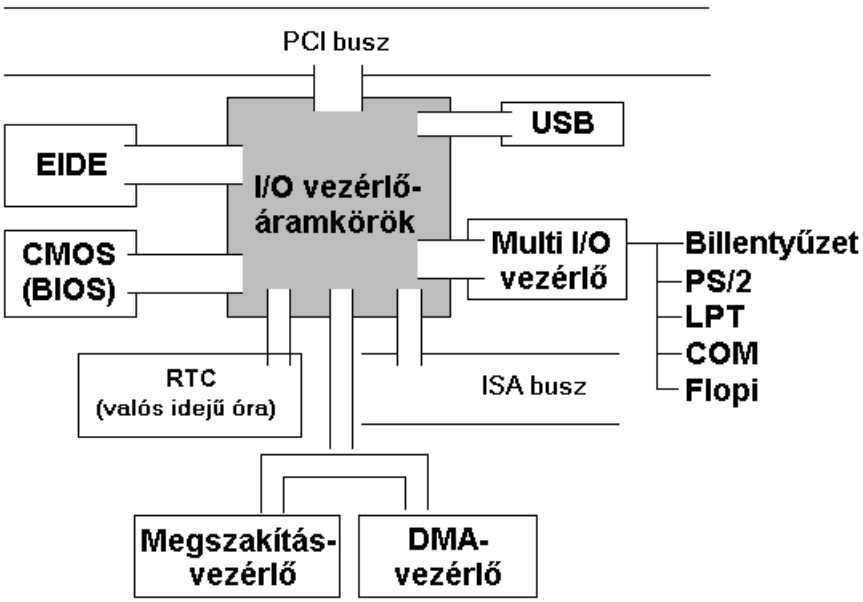
- a másodlagos gyorsító tár (L2 cache) vezérlése
- rendszerbusz vezérlése
- rendszermemória hozzáféréseinek és frissítésének vezérlése
- AGP vezérlése
- PCI hostbridge funkció ellátása



5-11. ábra: *Memória vezérlő*

Az I/O vezérlőáramkörök feladatai:

- az I/O buszok (PCI, ISA) vezérlése
- merevlemez-csatoló (EIDE) kezelése
- CMOS kezelése
- a valós idejű óra áramkör (RTC) kezelése
- a megszakítás-vezérlés
- a DMA-vezérlés
- a USB vezérlése
- PCI-ISA bridge funkciók ellátása
- a Multi I/O vezérlés, amely biztosítja meghatározott perifériák: billentyűzet, PS/2, soros (COM) és párhuzamos port (LPT) és floppymeghajtó vezérlését

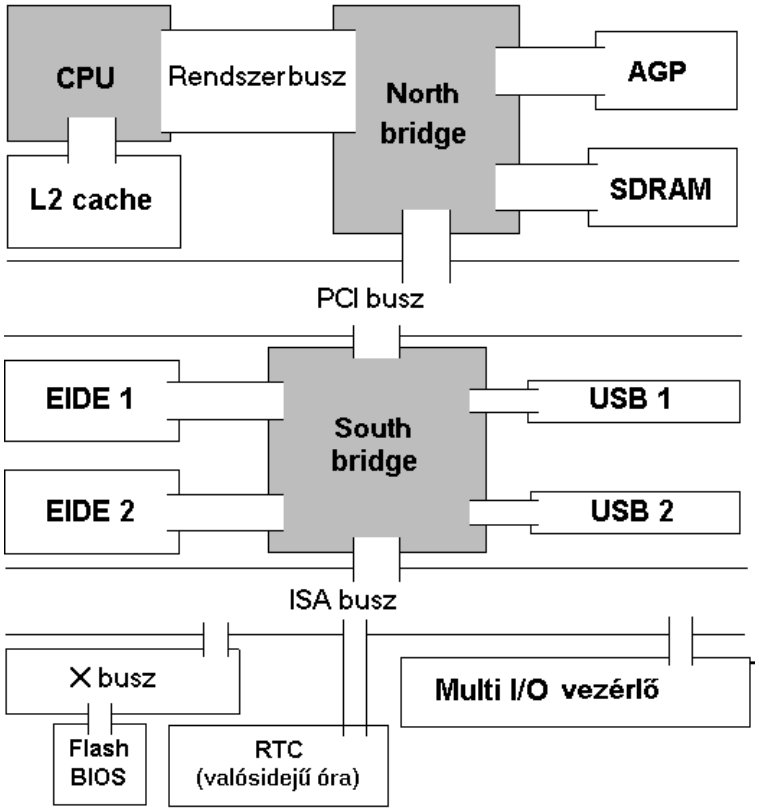


5-12. ábra: *I/O vezérlő*

A bridge architektúrájú áramkörkészletek

A P6 processzorcsaládhoz tartozó áramkörkészletek kezdetben bridge, majd később hub architektúrájúak.

A bridge architektúrájú áramkörkészlet felépítését az 5.13 ábrán, a 82440LX példáján mutatjuk be.

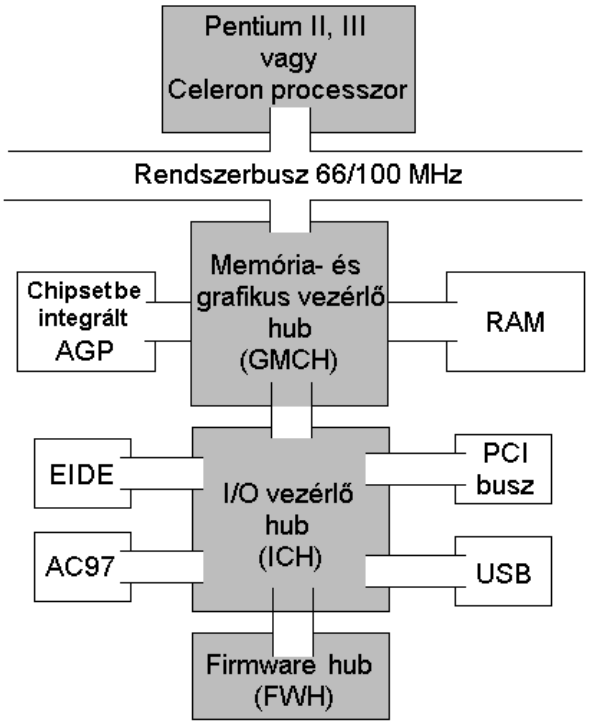


5-13. ábra: ***Bridge architektúra***

Látható, hogy ennél az architektúránál az északi híd (North Bridge) a gyors adatátviteltigénylő memóriát (SDRAM) és az AGP-t kapcsolja a rendszerbuszhoz a déli híd (South Bridge) az USB és EIDE eszközöket, valamint az ISA buszt szolgálja ki és a hidak közötti adatáramlást a PCI busz biztosítja.

A hub architektúrájú áramkörkészletek

Az első hub architektúrájú, 810 típusszámú áramkörkészletét az Intel 1999-ben jelentette meg "Whitney" fantázianéven. Ennek blokkvázlatát mutatja be az 5.14 ábra:



5-14. ábra: ***HUB architektúra***

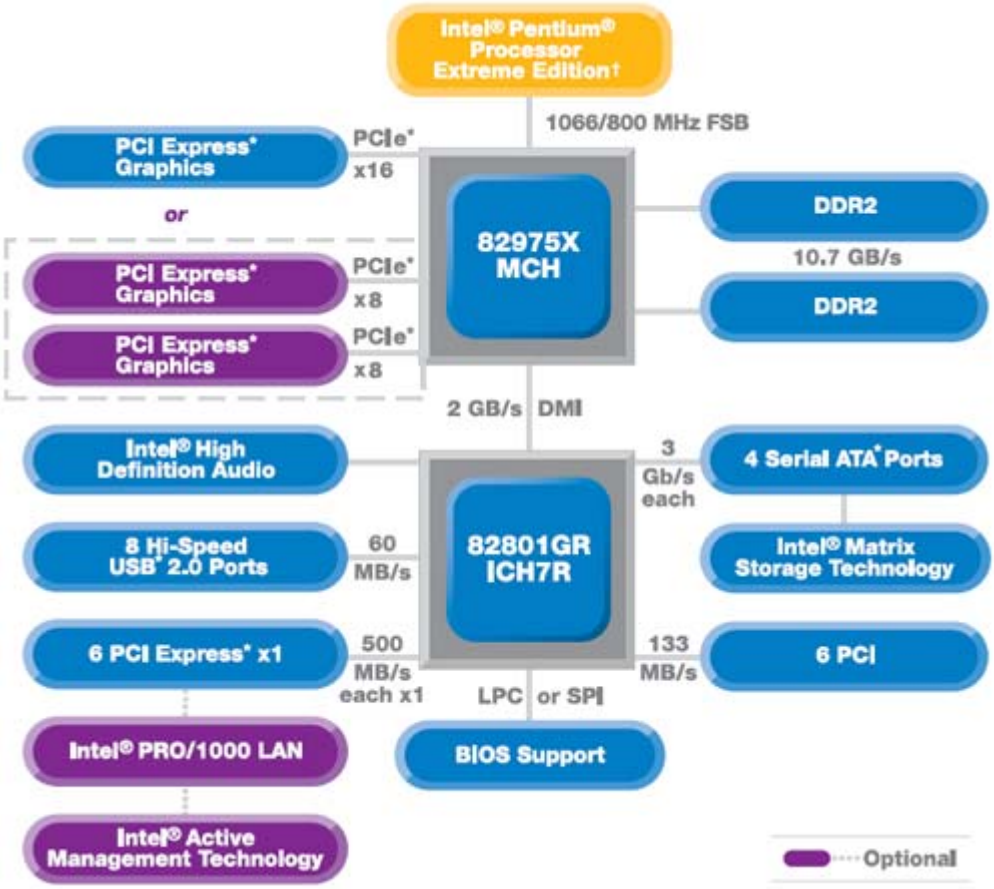
A számítógép-hálózatoknál használatos hub (kerék-agy, ahová a kerék küllői csatlakoznak) elnevezést az indokolja, hogy a vezérlőáramkörök GMCH(**G**raphical **M**emory **C**ontrol **H**ub), ICH(**I**/O **C**ontrol **H**ub), FWH (**F**irm**W**are **H**ub) csoportjait nem a PCI busz kapcsolja össze, hanem egy - csak a hubok kommunikációjára szolgáló - külön sín), amelynek pont-pont közötti adatátviteli teljesítménye kétszerese a PCI sínének (266 MB/sec).

A 810-es áramkörkészletet olcsó számítógép-konfigurációkhoz fejlesztette ki az Intel. Így a hubarchitektúra mellett a következők is megkülönböztetik minden korábbi elődjétől:

- a korábban a monitorvezérlő és 2D, 3D gyorsítókártyák és az AGP által végzett feladatokat a GMCH-ba integrálták. Ez a grafikus csip viszont a rendszermemóriából lefoglalt frame-bufferrel (min 1 Mbájt) működik együtt, amit a Windows a felbontástól függően dinamikusan kiegészít. Emellett a grafikus vezérlő drivere további 10 Mbájtot foglal le;
- az FWH hub egyrészt a Flash- memóriában tárolt BIOS kezelését biztosítja, másrészt egy fizikai véletlen számgenerátort tartalmaz, amely például titkosításra (rejtjelezés) hasznosítható;
- az AC97 CODEC interfész egy olcsóbb lehetőséget kínál a hangkártya és a modem helyett. Ezen keresztül kapcsolódhat az alaplaphoz az AMR (Audio Modem Riser) kártya, amely lényegében csak a telefonvonal kezeléséhez szükséges áramköröket tartalmazza. Természetesen ekkor a hangkártya társprocesszora által végzett feladatokat a CPU-nak kell átvállalnia;
- a 810-es áramkörkészlet nem támogatja az ISA buszt, így az ISA-PCI bridge áramkörei megtakaríthatók. Ennek viszont az az ára, hogy a 810-es áramkörkészlettel ellátott alaplapokat tartalmazó PC konfigurációban az ISA eszközök már nem alkalmazhatók.

A chipset képességei határt szabnak az egész rendszernek. A processzort kicserélhetjük egy gyorsabbra, a memória méretét növelhetjük (persze csak addig, ameddig a chipset engedi...), a BIOS-t frissíthetjük stb. Egy új chip-készlet azonban új alaplapot is jelent.

Az Intel az új processzorgenerációihoz hasonló gyakorisággal (többször hozzájuk igazítva) adta ki friss lapkakészleteket. Ez az Intel stratégia különösen azok számára kedvezőtlen, akik akár alkatrészenként összerakott, akár egyben megvásárolt konfigurációjukat szeretnék oly módon továbbfejleszteni, hogy egy erősebb vagy nagyobb tudású processzor miatt ne kelljen rögtön alaplapot, memóriát, netán más komponenseket is cserélni.



Forrás: http://prohardver.hu

5-15. ábra: Intel 975X Express chip-set blokk vázlata

A 975X chipset-et önkényesen kiválasztva egy modern (2006 I. félév) chipset funkciót végig követhetjük az 5.15 ábrából. Mint látható a lapkakészlet 2 részből áll, a 82975X jelzésű és a 82801GR jelzésű hub-okból.

Az MCH (Memory Control Hub) a processzorral a 10,7 GB/s sebességgel csatlakozó memória modulokkal, valamint a gyors perifériák csatlakoztatására alkalmas 16 szoros és az opcionális 8 szoros PCI Express portokkal, valamint a periférák csatlakoztatását vezérlő ICH-val (Input/Output Control Hub) tart kapcsolatot.

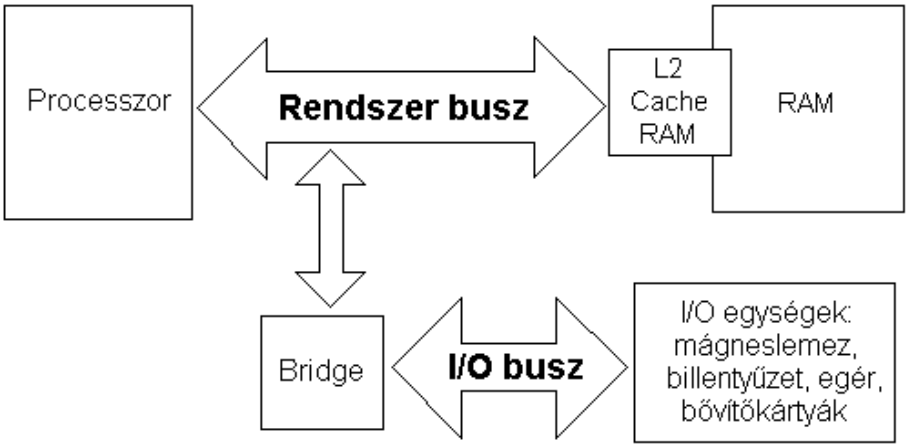
Az ICH feladata a lassúbb eszközök csatlakoztatása, igy ehhez kapcsolódnak a 8 szoros sebességű USB portok, az audio rendszer, a BIOS, PCI Express-en keresztül a hálózati csatoló, a merevlemezek 4db serial ATA csatlakozón, (amelyek RAID üzemmódban is dolgozhatnak) valamint az egyéb periféria csatolók a 6db PCI porthoz csatlakozhatnak.

5.1.5. Sínrendszerek a PC-kben

A processzor-memória busz és az I/O busz

A modern PC-architektúrában két alapvető busztípus különböztető meg:

- a rendszerbusz (system bus), amely a processzort köti össze a rendszermemóriával (L2 cache és DRAM modulok);
- az I/O busz, amely a periféria- és háttértárvezérlésekkel biztosítja a kapcsolatot.

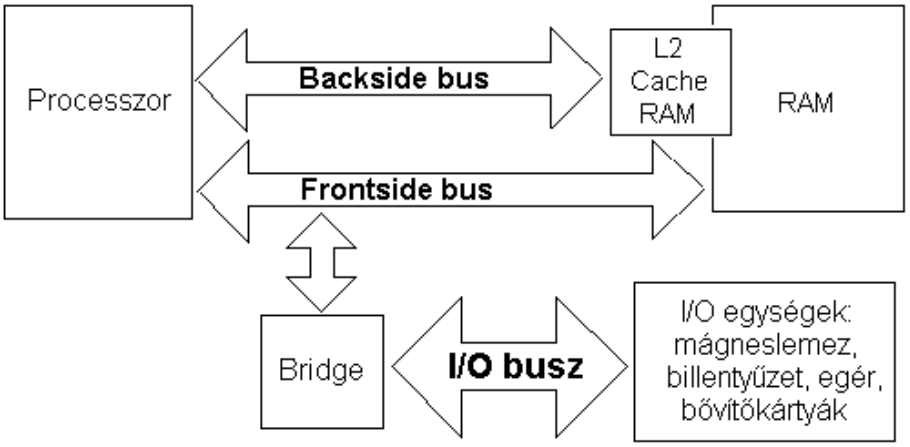


5-16. ábra: ***Rendszerbusz és I/O busz***

A processzor a rendszermemóriával közvetlenül kommunikál, az I/O buszt viszont - az eltérő sebesség és kommunikációs protokoll miatt - az alaplapi vezérlőáramköri készletben található bridge áramkörök közbeiktatásával éri el.

A Dual Independent Bus architektúra és az FSB

A Pentium II processzortól kezdve a PC-k rendszerbusza Dual Independent Bus (DIB) architektúrájú. Ez azt jelenti, hogy két, egymástól független működésre képes buszrendszer köti össze a processzort a RAM modulokkal (Frontside Bus = elől levő busz) és az L2 cachesel (Backside Bus = hátul levő busz).



5-17. ábra: ***A frontside és a backside busz***

A negyedik generációs processzoroktól (486DX2 és DX4) kezdve a processzor belső órajele magasabb lehet a rendszerbusz órajelénél. A kettő közti kapcsolatot az órajelszorzó teremti meg.

A processzor belső (mag) sebessége = FrontSide Bus (FSB) sebessége x Órajelszorzó

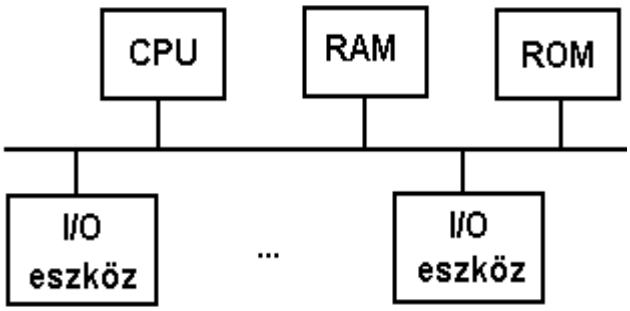
Az órajelszorzó (Clock Multiplier) értéke csak 0,5 egész számú többszöröse lehet. A Frontside Bus tehát fizikailag a memóriát és az alaplapi vezérlőáramkör-készletet köti össze a processzorral. Ebből fakadóan ezek az eszközök az FSB órajele által meghatározott sebességgel működnek.

Az egyre gyorsabb processzorok és memóriák az FSB órajelének további növelését is szükségessé tették.

Az I/O buszrendszerekről általában

Az első és második generációs PC-kben egyetlen buszrendszer kapcsolta össze a processzort, a memóriát és az I/O eszközöket. Ekkor még a processzor és a memória relatíve kis sebessége miatt a sínrendszer és az összes rákapcsolt eszköz a CPU órajelével kommunikált.

Az egyre gyorsabb CPU-k és memóriák sebessége ma már nem teszi lehetővé, hogy ezeket és a náluk jóval lassúbb I/O eszközöket egy közös sínrendszerre kapcsoljuk



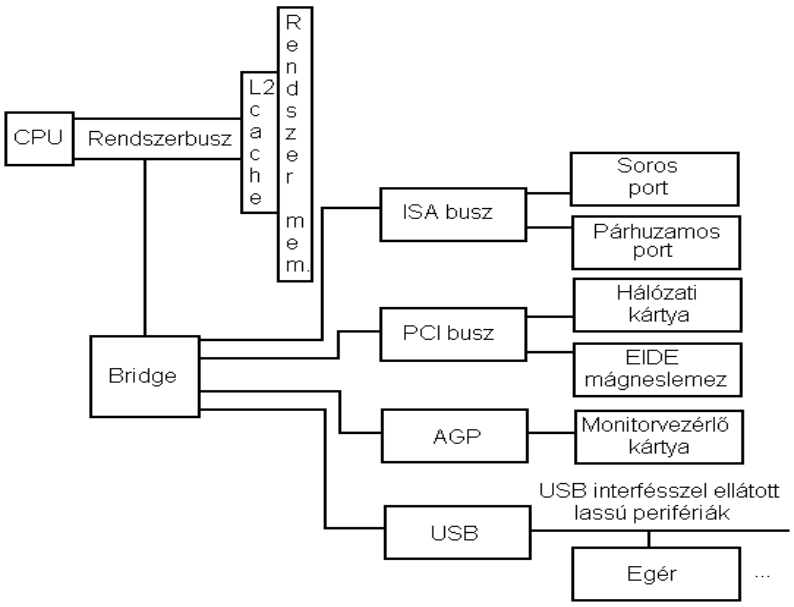
5-18. ábra: ***Közös I/O busz***

A rendszerbusztól leválasztott, és más sebességgel működő I/O buszt tartalmazó architektúrát a Compaq cég vezette be 1987-ben. Ez az ún. Multi-Bus-Architecture azóta szabvánnyá vált, és napjaink PC-i már több I/O busszal működnek

Egy PC-ben általában a következő I/O buszok valamilyen kombinált alkalmazásával találkozhatunk:

- ISA busz, amely régi, lassú és ma már elavult sínrendszer;
- PCI busz, amely a Pentium gépcsaládba tartozó számítógépek 1990-es évekbeli általánosan használt I/O buszrendszere;
- AGP busz, amelyet a 3D grafika elterjedésével egyre nagyobb adatátviteli teljesítményt igénylő monitorvezérlő kártyák kiszolgálásához fejlesztettek ki;
- USB, amely egy nagy teljesítményű soros busszal csatlakoztatja a központi egységhez a kis és közepes teljesítményű adatátvitelt igénylő eszközöket.

Az I/O buszok egy bridge áramkörrel kapcsolódnak a rendszerbuszhoz, amely az alaplapi *vezérlőáramkör-készlet része.* Ezt mutatja be sematikusan a következő ábra.



5-19. ábra: ***I/O bridge***

- Az ISA busz az elmúlt években fokozatosan eltűnt a PC-k alaplapjairól, korszerűtlen, nem támogatja aPNP-t szerepkörét az USB fokozatosan átvette.

A PCI (Peripherial Component Interconnect, azaz a perifériákat összekötő) sín

A nagyobb teljesítményű processzorokat alkalmazó Pentium gépcsalád megjelenése, a grafikus felhasználói felület használatának általánossá válása jelentősen megnövelte az I/O buszrendszer adatátviteli teljesítményével kapcsolatos igényeket. Lényeges követelménnyé vált, hogy a sínrendszer "intelligens" legyen, azaz például meghatározott feltételek mellett a processzortól viszonylag függetlenül is képes legyen működni. A mai és az 1990-es évek PC-iben I/O buszrendszerként a PCI sínt használják.

A PCI szabvány első változatát az Intel tette közzé 1992-ben, ezt követően több verziója is megjelent. A fejlődés állomásairól tájékoztat az 5.10 táblázat.

A PCI Express (PCI-X) szabvány 1.0-ás verzióját a legnagyobb szervergyártók kezdeményezésére, a nagy átviteli sebességet igénylő interfészek (Gigabit Ethernet, Ultra3 SCSI) miatt 1999-ban dolgozták ki. Ez nem csak egy új grafikus kártya illesztési szabványt jelent, hanem új átviteli technikára épül. Ez egy új architektúra ami nem csak a grafikus kártyára van hatással, hanem a chipsetre, processzorra, perifériák csatlakozására, és az alaplapokra is. Gyorsabb átvitelt, nagyobb áteresztő képességet takar, soros átvitelt alkalmaz.

Az egyes PCI verziók jellemző teljesítményadatait a következő táblázat mutatja be:

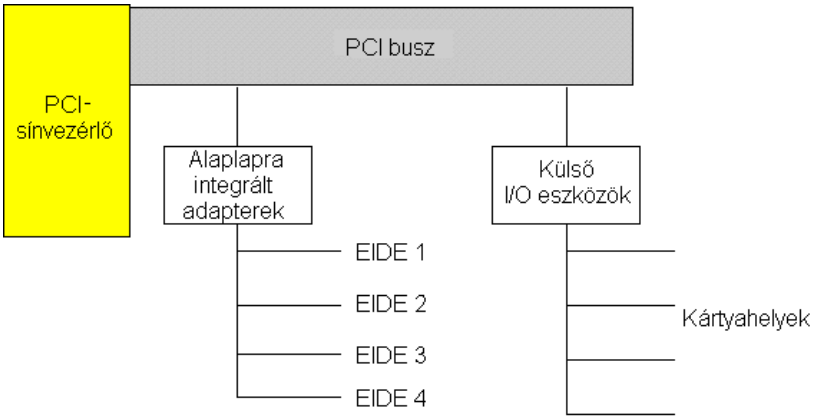
5-10. Táblázat: ***A PCI szabvány egyes verzióinak összehasonlító táblázata***

PCI szabvány	Év	Maximális buszfrekvencia	Adatátvitel ciklusonként	Maximális adatátviteli sebesség Mbájt/sec
PCI 1.0	1992	33 MHz	32 bit	132
PCI 2.0	1993	33 MHz	32 bit	132
PCI 2.1	1995	66 MHz	64 bit	524
PCI-X	1999	133 MHz	64 bit	1048

A PCI buszrendszer felépítése

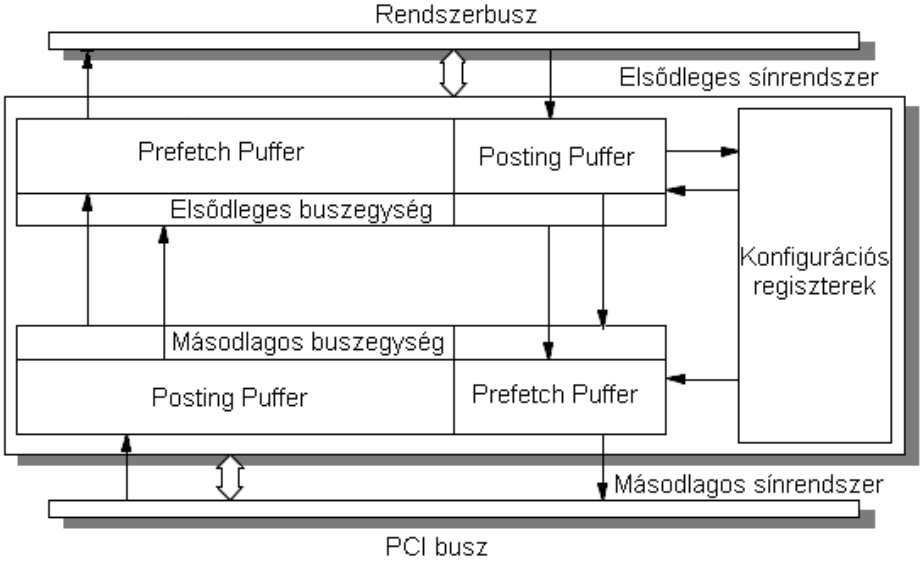
A PCI busz alapkoncepciója az I/O egységek szabványos csatlakoztatását biztosító sínrendszer (I/O sín) és a processzor-tároló alrendszer (rendszersín) határozott szétválasztása volt. A két buszrendszert az alaplapi vezérlőáramkör-készletben található PCI-Bridge (PCI-híd) áramkörök kapcsolják össze. A PCI buszhoz kapcsolt eszközöket PCI-egységeknek (PCI-Agent) nevezik, ezekből maximum 10 db lehet. Ilyen PCI-egység lehet a SCSI adapter, a hálózati csatoló (például Ethernet), vagy egy monitorvezérlő kártya. Fizikai megvalósításukat tekintve a PCI-egységek lehetnek az alaplapra integráltak vagy a PCI sín slotjaiba illeszthető PCI-kártyák.

A szabványos bővítő buszok (például ISA vagy más buszrendszer) interfészeit szintén PCI-egységként kezeli a sínrendszer.



5-20. ábra: PCI busz és az I/O eszközök kapcsolata

A PCI Bridge-el megteremtették a lehetőségét a sínrendszer processzorfüggetlen alkalmazásának. A PCI buszt a PC-ken kívül egyes RISC processzoros architektúrákban, például Apple, Sun, Unix gépek is alkalmazzák.



5-21. ábra: A PCI bridge felépítése

Prefetch és Posting Pufferek lehetővé teszik adatolvasáskor és adat-íráskor azoknak az adatoknak az átmeneti tárolását, amelyek egy későbbi sínciklusban kerülnek továbbításra a rendszer vagy a PCI buszon. Így pl. a CPU a PCI sebességénél gyorsabban írhatja a Posting Puffert, amely később a PCI sebességének megfelelően továbbítja az adatokat a buszon. A PCI Bridge-ben található konfigurációs regiszterekkel lehet címezni, illetve olvasni vagy írni a PCI egységekhez hozzárendelt címtartományt, amely a konfigurációra vonatkozó vezérlő

információkat tartalmazza. (pl. a PCI egység azonosítója, gyártójának kódja, az eszköz típuskódja: SCSI, IDE, floppy-olvasó stb.)

A PCI Bridge képes a CPU-val teljesen párhuzamosan működni, ha a processzor ugyanakkor nem címzett meg egy PCI-egységet. Ekkor két PCI-egység a PCI Bridgen keresztül adatot cserélhet, miközben olyan program fut, amelynek például csak a cache-ben található adatokra van szüksége. Így a rendszermemória és egy I/O eszközvezérlő, vagy két I/O eszközvezérlő processzortól független adatcseréjét is biztosítja a PCI sín.

A PCI szabvány terminológiájában a sínen adatátvitelt, és ennek érdekében sínfoglalást kezdeményező eszközt "Initiator"-nak, a megcímzett PCI-egységet pedig "Target" (cél) eszköznek nevezik. Ez teljesen megfelel a korábbi terminológiában használatos Master ill. Slave eszköz fogalmának.

Az AGP (Accelerated **G**raphics **P**ort) **gyorsított grafikus port**

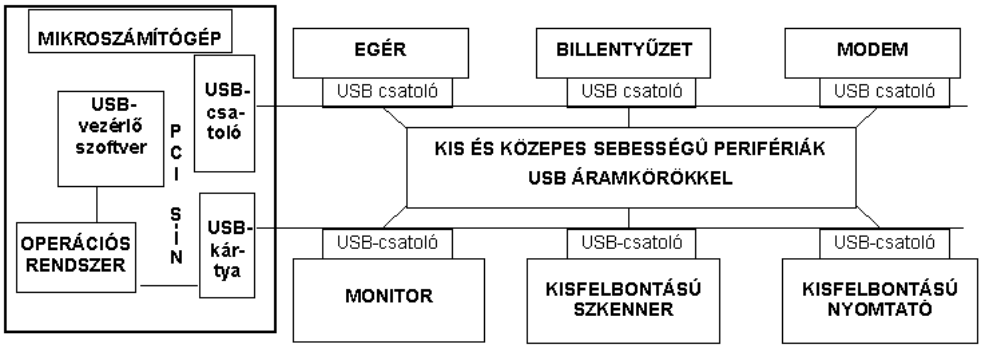
Az 1990-es évek második felében a nagyfelbontású monitorok, a 3D grafika (különösen a folyamatos mozgóképeket előállító játékprogramok) miatt a monitoron megjelenített kép felépítéséhez már igen nagy tömegű adatot kellett továbbítani az I/O buszon a monitorvezérlő kártyához.

Ebben csak részben segítettek az egyre gyorsabb videó-processzorral és egyre nagyobb memóriával ellátott monitorvezérlő- és 2D, 3D gyorsítókártyák, mivel a drága képmemória (VRAM) kapacitása minden határon túl való növelésének a költségek határt szabtak. A szűk keresztmetszetet a PCI sín adatátviteli teljesítménye jelentette, amely különösen akkor szembetűnő, ha figyelembe vesszük, hogy a központi I/O sínt több nagy sebességű eszköz (például LAN adapter, UDMA lemez stb.) is használhatja.

Az Intel a Pentium II processzorcsaládhoz vezette be a gyorsított grafikus portra vonatkozó sínszabványt, az AGP-t amely közvetlen (pont-pont), nagy teljesítményű adatátviteli kapcsolatot létesített a rendszermemória (a Front-Side Bus) és a monitorvezérlő kártya között.

Az USB

Az USB a kis és közepes teljesítményű adatátvitelt igénylő eszközöket egy nagy teljesítményű soros adatátvitellel csatlakoztatja a számítógép központi egységéhez.

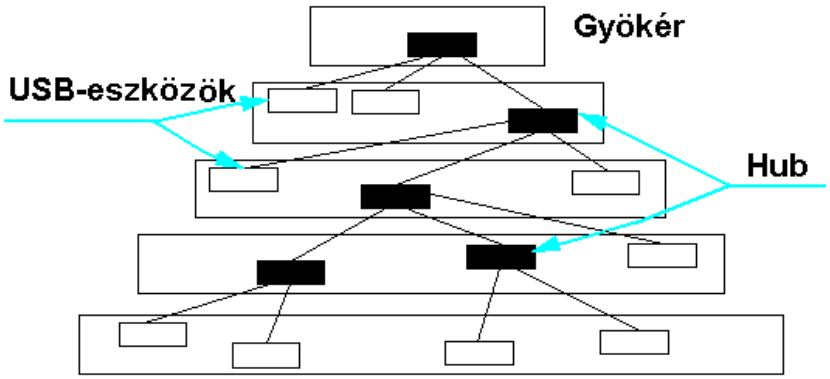


5-22. ábra: ***USB kapcsolatok logikai ábrája***

Az USB jellemzői:

- tiered star topológia (emeletes csillagfelépítés);
- 12 Mbit/sec átviteli sebesség;
- elvileg 127 USB-eszköz kiszolgálása egy szinten, max. 5 db hub;
- soros adatátvitel;

- az eszközök sorosan felfűzhetők (csak ha két kaput tartalmaznak);
- lehetővé teszi a Plug and Play perifériatelepítést;
- az eszközök tápellátása USB-kábelen keresztül lehetséges (kis áramfelvételt igénylő perifériák);



5-23. ábra: USB eszközök csatlakoztatása Hub-okon keresztül

A jövőben várhatóan el fog tűnni a soros és a párhuzamos port, mivel ezeket az USB kiváltja. Az USB 2.0-s verziója már 480 Mbit/sec átviteli teljesítményű.

A Firewire (IEEE - 1394-es szabvány)

A számítástechnika alkalmazásában egyre nagyobb szerephez jutnak a közepes teljesítményű adatátvitelt igénylő multimédiás, grafikus perifériák (videokamerák, nagy felbontású nyomtatók, nagy kapacitású streamerek stb.), ezek csatlakoztatása érdekében jött létre a Firewire (IEEE - 1394) szabvány.

A Firewire jellemzői:

- adatátviteli teljesítmény 400 Mbit/s és 800 Mbit/s;
- max. 16 eszköz csatlakoztatható;
- Plug and Play telepítés támogatása.

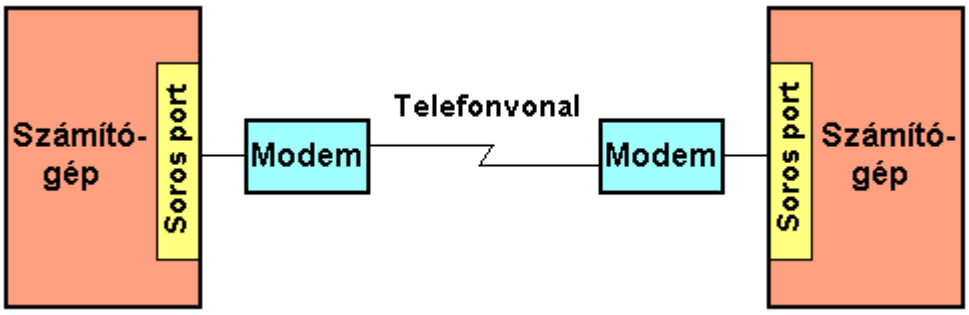
A soros port

A soros port a számítógépek összekapcsolására legrégebb óta használt megoldás. Az adatbitek egymás utáni (soros) átvitelét teszi lehetővé egyazon vonalon. Ebből rögtön következik, hogy ez egy viszonylag lassú, de gazdaságos átvitel. Az aszinkron átvitel lényege, hogy minden 8 bites adatsorozat előtt egy START bit áll, amely a vevő óráját szinkronizálja az adóéhoz. A sorozat egy esetleges paritásbittel és egy vagy két, az adatkeretet lezáró STOP bittel fejeződik be.

A soros portra csatlakoztatható eszközök:

- egér,
- modem,
- ISDN kártya,
- soros csatlakozójú nyomtató stb.,
- másik számítógép.

Az eszközök közötti átvitelnél a maximális kábelhossz nem lehet túl nagy (kb. 20 m), ezért a távolabbi adatküldéshez szükséges egy modem közbeiktatása (5.24 ábra).



5-24. ábra: ***Soros átvitel modem közbeiktatásával***

A modem (modulátor/demodulátor) a digitális adatot átalakítja analóg jellé az adó oldalon, a vevőnél pedig visszaalakítja az analóg jelet digitálissá. Az analóg hordozó egy szinuszos jel, amelynek valamelyik paraméterét kell megváltoztatni (modulálni) a bináris információnak megfelelően. Ennek a paraméternek függvényében a következő modulációs lehetőségeket különböztetjük meg:

- amplitúdó-moduláció,
- frekvencia-moduláció,
- fázis-moduláció.

A moduláció segítségével átalakított adataink továbbíthatók a hagyományos telefonvonal segítségéve. Napjainkban az átvitelnek ez a módszere egyre inkább háttérbe szorul.

A párhuzamos port

A párhuzamos interfész egyszerre (párhuzamosan) küld egy 8 bites adatot a vevő felé, mindegyik bitet külön vezetéken. Az átvitelt un. handshaking kézfogásos módszerrel bonyolítják le. Az adatvonalak mellett külön vonalon jelzi a gép a perifériának, hogy érvényes adatot helyezett az adatvonalakra (STROBE jel), majd az adatok átvételét a periféria nyugtázza egy másik erre a célra szolgáló vonalon(ACKNOWLEDGE).

Az eredetileg csak nyomtatók számára kifejlesztett egyirányú un.Centronics porthoz a későbbiekben újabb perifériákat is csatlakoztattak, ami szükségessé tette a kétirányú adatátvitelt a porton. Ennek megfelelően a következő adatátviteli lehetőségek léteznek:

- csak output irány (Centronics kompatibilis mód) pl. nyomtató számára;
- csak input irány (Nibble mód 4 bitenként, illetve Byte mód bájtonként) pl. szkenner részére;
- kétirányú (félduplex) adatforgalom lebonyolítására:
- EPP mód (Enhanced Parallel Port);
- ECP mód (Extended Capability Port).

Az EPP módban négy (cím írása, cím olvasása, adat írása, adat olvasása), míg az ECP módban két (parancs és adat) átviteli ciklust különböztetünk meg. Ezekben az üzemmódokban mintegy 2 Mbájt/sec sebességű adatátvitel valósítható meg. A maximális kábeltávolság nem lehet hosszabb 5-10 méternél.

5.2. Háttértárak, tömegtárak a PC-kben

A háttértárakat számos szempont szerint lehet kategorizálni, mi az adattárolásra használt fizikai alapelvet tekintjük a csoportosítás alapjának. Ilyen értelemben megkülönböztetünk történeti megjelenésük sorrendjében:

- *mechanikus* (lyukkártya, lyukszalag)

- *mágneses* (mágnesszalag, mágnesdob, mágneslemez)
- *optikai* (magneto-optikai lemez, CD, DVD)
- *elektronikus* (memóriakártya, pendrive)

elven működő háttértárakat.

Az egyes háttértárak jellemzésére olyan tulajdonságok szolgálnak, mint a **kapacitás**, a jellemző **adatátviteli sebesség**, az **elérési idő** vagy a megbízhatóság. Régebben ide sorolták a hordozhatóságot is, de ma már ennek a jellemzőnek nincs akkora gyakorlati jelentősége: szinte az összes aktuális háttértár kialakítható oly módon, hogy – igény esetén – hordozható legyen. A háttértárak a tár-hierarchiában a legalsó szinten helyezkednek el mint az a 3.17-es ábrán is láthattuk.

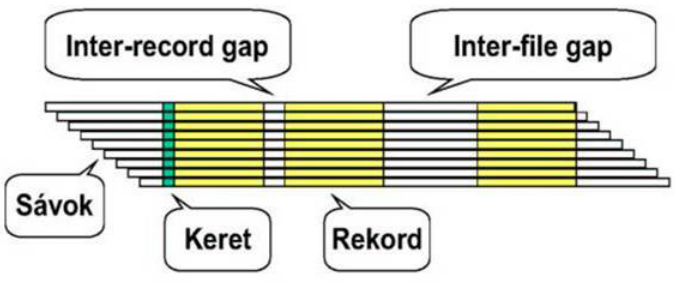
A következőkben (a teljesség igénye nélkül) bemutatjuk a (számítógépes munkavégzés gyakorlata szempontjából) fontosabb háttértárak elvi felépítésének és működésének alapvető jellemzőit.

A **mechanikus adattárolás** a lyukkártya és a lyukszalag esetében a papír alapú hordozó mechanikus megváltoztatásán, kilyukasztásán alapult. Azaz a kilyukasztott helyen 1-esnek, míg ellenkező estben 0-nak érzékelte az olvasó az adatot, a számítógépek harmadik generációjával gyakorlati jelentőségét teljesen elveszítette.

A mágneses elvű háttértárak esetében az adatok tárolása a **mágneses indukció** elvén alapszik: az elektromos áram a vezető körül mágneses mezőt hoz létre, a változó mágneses mezőbe helyezett vezetőben elektromos áram indukálódik. Az ilyen háttértárakban a *hordozófelületen* vékony réteg *mágnesezhető anyag* található, valamint az eszköz tartalmaz egy vagy több *író-olvasó fej*et, ami a fentebb említett fizikai állapotváltozásokat képes előidézni írásnál és érzékelni olvasásnál.

A **mágnesszalag** a mágneses elvű tárak közül, ugyan nem tartozik a legkorszerűbb háttértárak közé (a második generációs számítógépnél már alkalmazták…), de archiválási célokra még ma is sok helyen használják – elsősorban a viszonylag nagy (100 GB nagyságrendű) tárolókapacitású un. streamer-eket. A hordozóréteg a rögzített író-olvasó fej alatt halad el, ennek következtében soros elérésű a tár, átviteli sebessége 2-10 Mbit/s. Ez utóbbi érték azonban – a soros elérés miatt – nem túl informatív, hiszen az elérési idő nagyságrendekkel nagyobb.

Jellemző az adatfelírásnál használatos **sávos szerkezet.** Adattárolási szempontból a szalag hossz-irányban eltolt egybites sávok csoportjaként definiálható. **Rekord-szervezést** alkalmaz az egyes állományok tartalma azonos méretű részekben (rekord) kerül felírásra, és mind az egyes rekordokat, mind az egyes állományokat különböző méretű üres helyek „gap-ek" választják el egymástól (*5.25 ábra*). Ennek a jelentősége abban áll, hogy az elérési idő nagymértékben csökkenthető az által, hogy az író-olvasó fej gyorscsévélés közben is képes érzékelni ezeket az üres helyeket.



5-25. ábra: ***A mágnesszalag szerkezete***

A **mágneslemezek** esetében az alapvető különbség (a mágnesszalaghoz képest) az, hogy az író-olvasó fej *mozog* a hordozóréteg fölött Alapvetően két típust különböztetünk meg:

- **hajlékony lemezt** (floppy disk, FDD: **F**loppy **D**isk **D**rive) és
- **merevlemezt** (winchester, HDD: **H**ard **D**isc **D**rive).

A **floppy lemezt** vékony műanyag védőtokozás borítja, ami a hordozóréteg felületének szennyeződését hivatott megakadályozni. A hajlékony lemezek sokáig elsődleges hordozható háttértárnak számítottak, azonban viszonylag kis tárolókapacitásuk miatt az utóbbi időben háttérbe szorultak. Ugyan a jelentősebb gyártók megpróbálták „átmenteni" a hajlékony lemezt a korszerű háttértárak közé: pl. a 3M „SuperDrive" (LS-120), vagy Iomega „zip-drive", azonban előbb az optikai háttértárak, majd az elektronikus tárak megjelenése és elterjedése miatt ezek a kísérletek sikertelennek bizonyultak.

Floppy lemezek tárolókapacitása a gyakorlatban előforduló típusok estében maximum 1,44 MB, és nem is fejlesztik tovább. Helyüket fokozatosan átveszik a CD ROM-ok valamint a PEN drive-ok.

5.2.1. A merevlemez jellemzői

A merevlemez (winchester) nagy kapacitású, közvetlen hozzáférésű tömegtároló, amely több közös tengelyen elhelyezett mágneses lemezből áll.

A merevlemez *jellemző adatai:*

- kapacitása40 -300 Gbájt;
- forgási sebessége 5400, 7200, 10 000, 14 000 fordulat percenként;
- lemezátmérője 5-30 cm;
- hozzáférési ideje (napjainkban) 4-8 msec.

A mágneslemez hozzáférési ideje (Disk Access Time) alatt egy adatblokk kiolvasásának idejét értjük. Ez nyilvánvalóan csak átlagértékként értelmezhető, mivel ez nagyon függ az olvasófejnek a kiolvasás megkezdése előtti helyzetétől, amint a következő képletből is kiderül:

Hozzáférési idő = Pozicionálási idő átlaga + Forgási idő + Adatátviteli idő + Vezérlési idő.

A merevlemez fizikai felépítése: cilinderek, sávok és szektorok

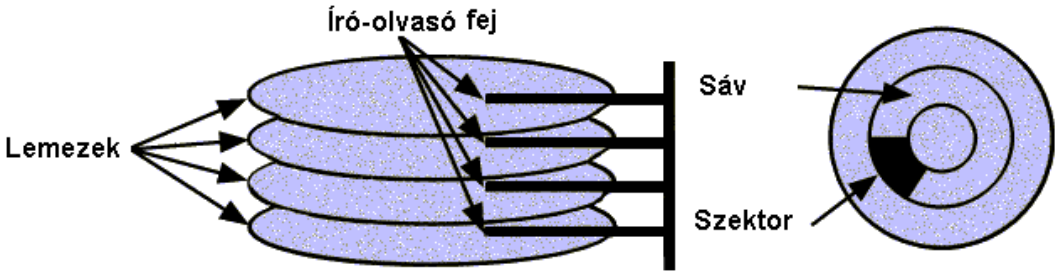
A lemezek koncentrikus körökre vannak felosztva, ezeket nevezzük sávoknak. A sávok a fejek sugárirányú elmozdulásával érhetők el. Az egymás alatti sávok egy cilindert alkotnak, ennek adatai fejmozgás nélkül elérhetőek. Mindegyik sáv megadott számú, egyenlő kapacitású adattároló helyet tartalmaz, ezek a szektorok (5.26 ábra). A szektor a lemezen kezelhető legkisebb fizikai adategység. Egy szektorhoz való hozzáférésnek (írásnál vagy olvasásnál) a szektort három paraméter megadása azonosítja:

a cilinder száma (C = cylinder),

a fej száma (H = head),

a szektor száma (S = sector).

Ezeket az információkat a szektor kezdő része, az ún. szektorfej tartalmazza.



5-26. ábra: *A merevlemez sematikus ábrája*

A merevlemez fizikai olvasása, írása, formázása

A lemezen minden szektor két részből áll: a szektorfejből és az adatblokkból. Ezek kezdetét egy-egy azonosító mutatja meg (IM = Identifier Mark, DM = Data Mark). Az adatblokk mérete tipikusan 512 bájt. Mivel a legkisebb adategység a szektor, a merevlemez fizikai kezelése szektorszintű műveletekkel történik, mint például:

- szektorok írása és olvasása,
- törölt jelzésű szektorok írása és olvasása,
- azonosító olvasása.

Ezeket a műveleteket vagy közvetlenül az interfész parancsregisztereinek feltöltésével, vagy a lemezt kezelő BIOS programmal (ezt egy szoftvermegszakítással kell hívni) lehet elvégeztetni. Minden elindított fizikai műveletet a meghajtó a fej pozícionálásával készít elő, azaz a fejet a kívánt sáv fölé irányítja.

Ahhoz, hogy a lemezen létrejöjjenek a sávok és a szektorok, egy speciális műveletet kell végrehajtani, amit formázásnak nevezünk. Az alacsony szintű, vagy fizikai formázás (Low Level Format) lényegében abból áll, hogy felírják a lemezre az azonosítókat, és mindegyik szektorfejbe beírják a sáv számát, a fej sorszámát és a szektor számát (CHS). A szektorok számozása 1-től kezdődik, és a hibás szektorokat figyelmen kívül hagyják. Az alacsony szintű formázást a mai merevlemezeknél már a gyárban végrehajtják, így ezt nem kell, sőt általában nem is szabad a felhasználónak elvégezni.

A merevlemez logikai kezelése

A merevlemezek használatához egy logikai formázást is kell végeznünk, amely kialakítja a lemezen az alkalmazni kívánt fájlrendszert. A fájlok elhelyezkedését a lemez elején létrehozott FAT tábla mutatja, amely után a hierarchikus fájlrendszer gyökérkatalógusa következik

A logikai lemezkezelés alapegysége a több szektorból álló szektorcsoport, a klaszter (cluster). A fájlok a lemezen klaszterekre vannak osztva, így az operációs rendszer írni és olvasni a merevlemezt csak klaszterenként tudja. Egy klaszterben található szektoroknak a száma a lemez kapacitásától függ, de mindig 2-nek valamelyik hatványa.

Állomány-elhelyezési tábla

Az állomány-elhelyezési tábla (FAT = File Allocation Table) az állományok rekordjainak, azaz klasztereinek a lemezen történő elhelyezkedését tárolja. Ezzel tartja nyilván az operációs rendszer a lemezterületek foglaltságát, vagyis innen tudja meg, hogy hol van szabad hely a lemezen.

A FAT hibátlansága a rendszer működésének elengedhetetlen feltétele. Annak érdekében, hogy véletlenszerű sérülés esetén se legyen probléma, az állomány-elhelyezési táblát két példányban tárolják. A két tábla egymás mögött helyezkedik el, a második az első pontos másolata.

A Windows 95 előtti operációs rendszerek 16 bites bejegyzésű FAT táblát használtak, amely jelentősen korlátozta a lemez méretét. Az újabb rendszerek FAT 32-es táblát alkalmaznak, ezzel a lemez-tárolókapacitás elméleti határa 2048 Gbájtig terjed ki.

Partíciós tábla

Minden merevlemez egy fizikai partícióból áll, ennek mérete a lemez teljes területével egyenlő. A fizikai partíciót fel lehet osztani több logikai partícióra. A logikai partíciókat úgy látjuk, mintha azok külön merevlemezek volnának. A partíciók felhasználásával több operációs rendszer futtatására is lehetőségünk nyílik, ha ezeket külön partíciókban helyezzük el.

A merevlemez első szektorában található a mester betöltő bejegyzés (MBR = Master Boot Record), amely a gép indításánál nyújt információt a betöltendő operációs rendszerről. Az MBR része a partíciós tábla, amely a lemezen található logikai partíciókat írja le.

A **merevlemezek** legfontosabb jellemzői közé tartozik csatlakozási felületük, azaz milyen perifériaillesztőn keresztül csatlakoznak a számítógéphez, valamint a belőlük kialakított tároló szervezése (pl. RAID szervezés).

Az IDE csatoló és a lemezvezérlés

A legtöbb asztali és hordozható számítógépben megtalálható legalább egy merevlemezes háttértároló, ami többnyire nem cserélhető, gyors, és "Winchester" néven is ismert.

A tároló csatolójának neve IDE = Integrated Drive Electronics, utal arra a megoldásra, amelynek lényege az, hogy a meghajtó vezérléséhez szükséges elektronikát fizikailag a lemezegységgel egybeépítik, ezzel növelni lehetett az adatátviteli sebességet, és egyszerűsíteni a PC buszrendszerét. Az IDE rendszetechnikailag az I/O buszok közé tartozik.

A merevlemez-műveletek irányítását általában a lemezegységbe beépített mikroprocesszorral és a vezérlési információkat tartalmazó, csak olvasható tárolóval oldják meg. A lemezvezérlő a számítógép fő processzorától függetlenül képes irányítani a lemezműveleteket, ellátja az ezzel összefüggő szinkronizációs és adatkonverziós feladatokat, ellenőrzi a végrehajtott műveletek hibátlanságát. A legtöbb vezérlőbe beépítenek egy ROM-BIOS tárolót is, amely az alaplapon található BIOS-nak az adott típusú mágneslemezre vonatkozó speciális kiegészítését tartalmazza.

Mivel maga a lemezvezérlő már a merevlemez-meghajtóban kapott helyet, a lemezcsatoló kártya, vagy az alaplapra integrált csatoló, host adapterként működik, mint adatpuffer, összeköti az I/O buszt a lemezvezérlő-egységgel. (Emiatt nevezik az IDE-t AT-sínes csatolónak is.)

Az EIDE és ATA specifikációk

Az EIDE (Enhanced IDE), az IDE csatoló továbbfejlesztett változata, amelyet a Western Digital 1993-ban specifikált. Ennek főbb jellemzői a következők:

- Közvetlenül a PCI sínhez csatlakozik az ISA helyett, ami jelentősen növeli az átviteli sebességet (kb. 16 Mbájt/sec);
- A csatlakoztatható egységek számát kettőről négyre változott;
- Az eddigi 504 Mbájtos korlátot jelentő (1024 cilinder x 16 fej x 63 szektor x 512 bájt) maximális lemezkapacitást kiterjeszti 8,4 Gbájtra, úgy hogy megnövelték a kezelhető fejek számát;
- Bevezették a logikai blokkcímzést (LBA = Logical Block Addressing) 24 bites lineáris lemezcímekkel, amelyeket a BIOS fordít át megfelelő CHS (Cylinder, Head, Sector), azaz cilinder-fej-szektor alakú logikai címekre;

- Kibővített IDE hozzáférési parancskészlettel (ATAPI = AT Attachment Packet Interface) lehetséges CD-ROM és mágnesszalag csatlakoztatása is.

Az EIDE csatoló teljes mértékben kompatibilis az IDE csatolóval. A cégek által IDE-nek elnevezett specifikációt az ATA (Advanced Technology Attachment) elnevezésű amerikai interfész-protokoll szabványba foglalták bele. Ennek az idők folyamán több változata is született.

Az adatok átvitele a meghajtó és a memória között kétféle módon történhet:

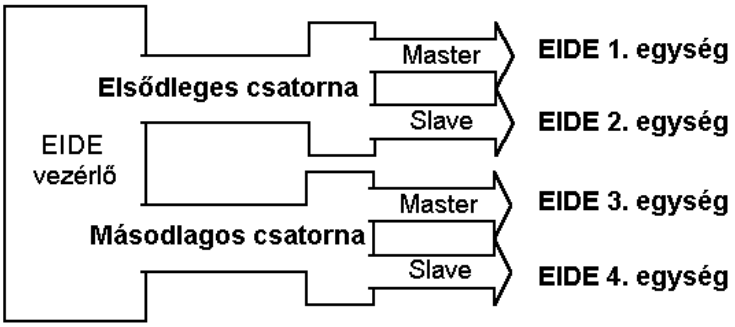
- programozottan (**PIO** = Programmed Input/Output), azaz a CPU vezérlésével, vagy
- közvetlen memória-hozzáféréssel (**DMA** = Direct Memory Access), amikor a meghajtó vezérlője a master eszköz.

Az ATA szabványok több PIO és DMA módot határoznak meg, amelyek elsősorban az adatátvitel sebességében különböznek:

- **Fast ATA (gyors ATA vagy ATA-2)**: 16 Mbájt/sec átviteli sebesség PCI buszon keresztül,.
- **Ultra ATA (ATA-4 vagy ATA-33):** 33 Mbájt/sec-os átviteli rátával, hibaellenőrző kód alkalmazása (CRC = Cyclic Redundancy Check).
- **Ultra ATA/66 (ATA-5):** 1999-ben keletkezett, és a sebesség további növelését hozta újabb módok specifikálásával:
  - Ultra DMA Mode 3 (44 Mbájt/sec)
  - Ultra DMA Mode 4 (66 Mbájt/sec)
  - **Ultra ATA/100**-as szabvány: már 100 Mbájt/sec-os átvitelt specifikál.

Az alaplapra integrált EIDE csatoló

Az EIDE csatolót 1995 óta az alaplapi vezérlőáramkör-készlet tartalmazza. Ez összesen négy eszköz kezelését teszi lehetővé, két (elsődleges és másodlagos) csatornán keresztül. Mindegyik csatorna két darab, master/slave konfigurációba kötött eszközt képes kiszolgálni.



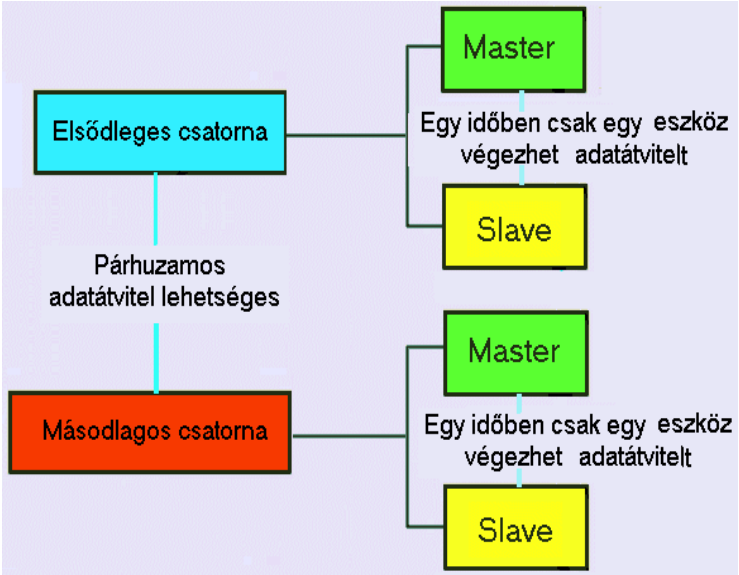
5-27. ábra: ***EIDE vezérlő és csatlakoztatott egységek***

Az EIDE-eszközök lehetnek:

- merevlemezek,
- CD- vagy DVD-meghajtók,
- streamer,
- Zip meghajtó.

A PC-s konfigurációk többségében *master*-ként az elsődleges csatornára egy merevlemezt, a másodlagos csatornára egy CD-ROM vagy DVD-meghajtót helyeznek el.

Több egység esetében figyelembe kell venni, hogy csak a két csatorna képes egymással párhuzamosan adatátvitelre (multitasking.)



5-28. ábra: EIDE vezérlő adatátviteli lehetőségei.

Serial ATA

A Serial ATA illesztőt a hagyományos (párhuzamos) ATA szabvány továbbfejlesztéseként hozták létre háttértároló eszközök illesztéséhez. Hasonló elveken működik, de soros átvitellel.

A soros ATA a párhuzamos ATA-val ellentétben egy kábelre kizárólag egyetlen merevlemez vagy más eszköz csatlakoztatását teszi lehetővé, és mindössze négy vezetéket használ fel a kommunikációhoz, de ennek ellenére hasonlóan magas átviteli sebesség (>150 MByte/s) elérését teszi lehetővé.

Az Ultra ATA-100 -ra való áttérés bár jelentős sebességnövekedést eredményezett, de a további sebességnövelésre nem sok kilátás van a párhuzamos buszrendszerben. A szalagkábel hossza korlátozott, nem haladja meg a 18 hüvelyket (46 cm). Ez pedig különösen a termetes számítógép házakban uralkodó viszonylag nagy távolságok miatt okoz gyakran gondot.

A Serial ATA technológia alapspecifikációjának kidolgozásakor már megdőlt az a korábban általánosan elfogadott "tény", miszerint a párhuzamos adatátviteli megoldások nagyobb sebességet tesznek lehetővé, mint a soros megoldások. Bár több adatvonal egyidejű alkalmazásával az átviteli sávszélesség egy bizonyos határig viszonylag könnyedén növelhető, e határ fölött ismét a soros megoldások kerülnek előtérbe.

A jelenlegi Serial ATA megoldások általában csak két soros csatlakozású meghajtó egyidejű használatát teszik lehetővé. Ennek oka elsősorban az, hogy a legtöbb egység továbbra is a párhuzamos ATA interfészt alkalmazza, a PCI csatlakozású Serial ATA bővítőkártyák pedig többletterhelést jelentenek a rendszernek. A PCI busz ráadásul az adatátviteli sávszélességet is korlátozza.

Az Ultra ATA-100 szabvány 100 sebességéhez képest a Serial ATA első generációja 150 MByte/s sávszélességet nyújt, ráadásul a számítógép, illetve a processzor sokkalta kisebb leterhelése mellett. A 150 MByte/s eléréséhez a Serial ATA 1,5 GHz-es órajelet használ, de a

vezérlő az átviteli csatornán minden 8 bites adatot 10 bites, kódolt formában továbbít, ami mintegy 20 százalékos teljesítménycsökkenést eredményez.

Bár a Serial ATA vezérlő folyamatosan képes tartani a 150 MByte/s adatátviteli sebességet, ez az érték a közeljövőben aligha lesz elérhető. Ennek oka elsősorban az, hogy az adatok egyelőre a PCI buszon keresztül áramolnak, amely legfeljebb 133 MByte/s adatátviteli sebességre képes. Ez a sávszélesség ráadásul csak abban az esetben érhető el, ha a rendszer többi eleme éppen nem bonyolít adatátvitelt a buszon keresztül, ilyen pedig a valóságban csak ritkán fordul elő. Tovább rontja a helyzetet, hogy a jelenlegi alaplapi chipkészletek a valóságban csak mintegy 80-90 MByte/s sebességet képesek elérni a PCI buszon keresztül.

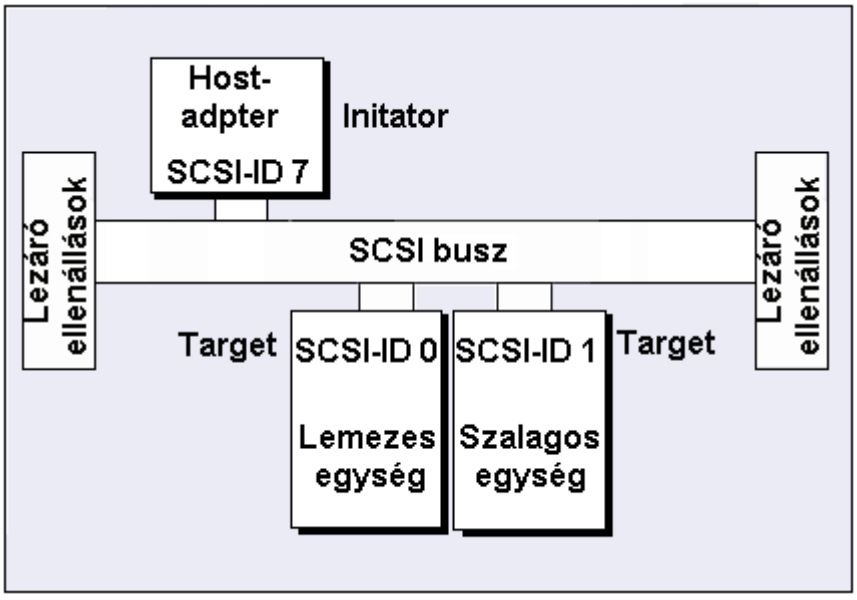
A Serial ATA a jelenlegi, párhuzamos ATA megoldások leváltására megalkotott, jelentős teljesítménytartalékokkal és skálázhatósággal rendelkező technológia. A Serial ATA 2X, kétszeres, 300 MByte/s adatátviteli sebesség elérését teszi majd lehetővé. A technológia skálázhatóságának köszönhetően - a 600 MByte/s sebességű Serial ATA 4X bevezetésével - a szabvány várhatóan legalább tíz évig, komolyabb módosítás nélkül is meg fog felelni az ipar elvárásainak.

5.2.2. Az SCSI

A SCSI jellemzői

A SCSI (ejtsd: szkázi) = Small Computer System Interface, azaz kisszámítógép-rendszer interfész, egy olyan szabványos csatoló, amely önálló sínrendszerként működik. Fizikai megvalósításában lehet PCI kártya vagy alaplapra integrált.

A SCSI buszra csatlakozó eszközök lehetnek adatátvitelt kezdeményező (Initiator vagy Master), illetve az adatátvitelben résztvevő (Target vagy Slave) eszközök. (Legalább egy Target és egy Initiator eszköznek kell lennie, de ezek számának csak a maximuma rögzített a szabványban.) A PCI sínre csatlakoztatott host adapter eszköz (SCSI csatolókártya) mindig Initiator. A Target és Initiator szerepek felcserélődhetnek.



5-29. ábra: ***SCSI busz***

Az adatforgalom szempontjából a sínen a következő fázisokat különböztetjük meg:

- szabad (a sín nem foglalt),
- arbitráció (egy kezdeményező kéri a sínvezérlés jogát),

- szelekció (a cél eszköz kiválasztása),
- reszelekció (a cél ujraválasztása),
- parancsküldés,
- adatküldés,
- üzenetküldés.

Minden, a sínhez csatlakozó eszközt a címe (SCSI-ID) egyértelműen azonosít. Az adatvonalak minden bitje egy eszköznek felel meg az arbitráció alatt. (Pl. 8 bites adatsín esetén a hoszt adaptert a 7. bit azonosítja, a többi a 6, 5 ... 0 kis helyiértékű bittel kerül azonosításra.)

Egy céleszköznek a SCSI-ID-n túlmenően van egy vagy több logikai címe is (LUN = Logical Unit Number), ezzel lehet címezni például különböző lemezpartíciókat.

A tranzakciót mindig az Initator-eszköz kezdeményezi, de az arbitrációs és kiválasztási fázis után átadja a vezérlést a céleszköznek. (Ez eltér a bus master szereptől.)

A Target jelez vissza az Initatornak, hogy a tranzakció megfelelően lefutott-e.

A SCSI-1, -2, -3 és az Ultra 160 SCSI szabvány

Az eredeti SCSI szabvány (1986) egy 8 bites összeköttetést határozott meg 5 Mbájt/sec sebességgel, összesen 8 eszköz között (a host + 7 merevlemezes meghajtó).

A SCSI-2 szabvány (1994) két újabb választható lehetőséget is definiált:

- Fast SCSI = "gyors" SCSI, kétszeres, azaz 10 Mbájt/sec sebességgel;
- Wide SCSI = "széles" SCSI 16 vagy 32 bites átvitellel (ehhez a meglévő 80 pólusú csatlakozó mellett egy második, 68 pólusú is szerepel). Az eszközök száma max. 16 lehet.

A kettő kombinációjából jött létre a Fast Wide SCSI, 20, illetve 40 Mbájt/sec-os átviteli rátával.

A SCSI-2 szabványban specifikálták a szinkron üzemmód elektronikus jeleit valamint az összes SCSI utasítást és paramétereket.

Az 1990-es évek második felétől megjelent új SCSI szabványok:

- Az Ultra SCSI (vagy Fast-20) a SCSI-2 kiterjesztése 20 MHz-es órajelre.
- Az Ultra SCSI-2-nél tovább növelték az órajel frekvenciáját (40 MHz-re), ami 16 biten 80 Mbájt/sec-os sebességhez vezetett.
- SCSI-3 szabványt folyamatosan specifikálták. Kiiktatták a második kábelt, és bevezették az üvegszálas soros adatátvitel lehetőségét, valamint a folyamatos üzemeltetés melletti készülékcserét.
- A SCSI-3 Ultra 160-as változata (1999) már 16 bites átvitelnél max. 160 Mbájt/sec adatátviteli sebességet tesz lehetővé (80 Mhz-es órajellel), és használja a hibajavító CRC (Cyclic Redundancy Check) kódolást.

A SCSI-3 része a SCAM (SCSI Configuration AutoMatically), amely a Plug and Play megvalósítását segíti azzal, hogy a SCSI-eszközöknek automatikusan osztja ki az ID címeket.

A SCSI egy buszrendszer, és nem a klasszikus értelemben vett interfész, fontosabb jellemzői a következők:

- A SCSI intelligens csatoló, ha megkapja a kommunikációra vonatkozó adatokat, akkor azt önállóan intézi, nem terheli a processzort, ezáltal segíti a multitasking üzemmódot;

- A SCSI egy buszrendszer, amelyre különböző típusú, "SCSI eszközök" csatlakoztathatók, pl. merevlemezek, streamerek, nyomtatók, szkennerek stb.;
- A különböző SCSI szabványok szerint a buszra csatlakozó eszközök 8, 16 vagy 32 bit szélességben képesek kommunikálni;
- A SCSI csatolóval gyorsabb lemezek is csatlakoztathatók mint az EIDE-vel (max. 160 Mbájt/sec-ig);
- A SCSI csatolók és lemezek műszaki megbízhatósága általában jóval nagyobb az EIDE-nél, ennek megfelelően áruk is többszörös lehet;

A sínprotokollt a SCSI-1, -2, -3 szabványokban specifikálták.

Az ATA/EIDE és SCSI csatolóval rendelkező merevlemez-meghajtókhoz dolgozták ki a SMART (Self Monitoring Analysis Reporting Technology) technológiát, amely a lemezegységet folyamatosan ellenőrzi, például a fordulatszámot vagy a fej távolságát a lemeztől, és előre jelzi a hibát, mielőtt az az adatok megsérülését eredményezné.

5.2.3. RAID

A RAID (**R**edundant **A**rray of **I**ntependent **D**isks), független lemezek redundáns tömbje, (de használatos az **I**nexpensive azaz olcsó jelző is) technológiát a Kaliforniai Egyetem (University of California) kutatói 1987-ben publikálták először, azóta széleskörűen elterjedt (pl. Linux is támogatja). A RAID technológia lényege, hogy több független merevlemez összekapcsolásával egy nagyobb méretű logikai diszket hoznak létre. A módszer kifejlesztésekor a tervezők többféle célt tűztek ki maguk elé:

- Nagy kapacitások kezelése, azaz a logikai diszk az egyes fizikai diszkek méretét jóval meghaladhatja
- Teljesítménynövelés, azaz az összekapcsolt diszkek együttes teljesítménye (írási-, olvasási sebesség) meghaladja az egyes diszkek teljesítményét.
- Hibatűrés, azaz az egyes diszkek meghibásodásával szembeni tolerancia.

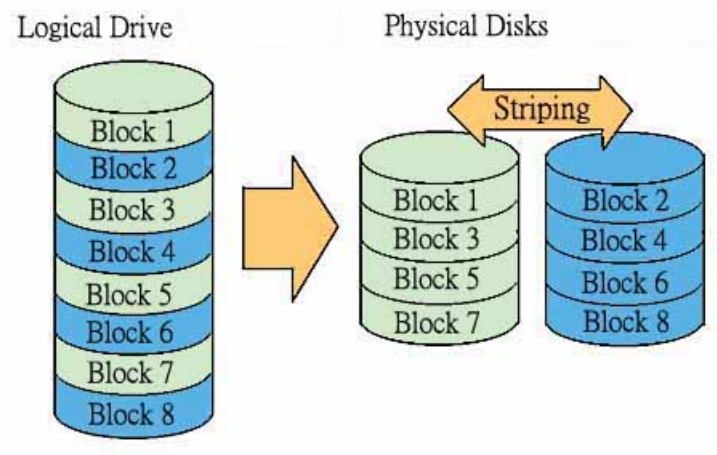
A RAID-ben eredetileg 5 szintet (RAID1-től RAID5-ig) definiáltak. Az egyes szintek általában nem a fejlődési, illetve minőségi sorrendet tükrözik, hanem egyszerűen különböző megoldásokat javasolnak. A kezdeti 5 szinthez később hozzávették a RAID 6-ot. RAID 0-ként szokták emlegetni azt a változatot, ahol a diszkeket redundancia nélkül kapcsoljuk össze. Ezenkívül használjak még a RAID 10, vagy RAID 1+0 elnevezéseket is, amelyek a RAID 1 és a RAID 0 kombinálásával hoznak létre. Hasonlóan a RAID 50 a RAID 5 és a RAID 0 kombinációja. A RAID alapötlete a lemezegységek sávokra (stripes) bontása. Az itteni sávok nem azonosak a lemez fizikai sávjaival (tracks), amit az angol elnevezés különbözősége is jelez.

RAID 0 – Striping

A RAID 0 nagyteljesítményű rendszerekhez ajánlatos, alkalmazza az egyes lemezegységek sávokra bontását, viszont semmilyen plusz redundanciát nem visz a rendszerbe, így nem biztosít hibatűrést, azaz egyetlen meghajtó meghibásodása az egész rendszer hibáját okozza. Mind az írási, mind az olvasási műveletek párhuzamosítva történnek. A módszer az összes RAID eljárás közül a legjobb teljesítményt nyújtja, ugyanis a többi módszernél a redundancia kezelése lassítja a rendszert. A tárkihasználás a redundancia hiánya miatt szintén hatékony.

Nem ajánlott, amennyiben az adatbiztonság a legfontosabb. Ezen mód kihasználásához legalább két merevlemez szükséges.

Striping (magyarul csíkozás): RAID-0 tömb létrehozásakor a merevlemezeket összefűzzük (kihasználva így a két merevlemez teljes kapacitását), és tárolófelületüket csíkokra osztjuk, melyek leginkább a sávokhoz hasonlít. Logikailag a két merevlemez egy merevlemeznek látszik. A tömbben létrehozott csíkok mérete beállítástól függ, elrendezésük a képen látható:



Forrás: http://www.pcguide.com

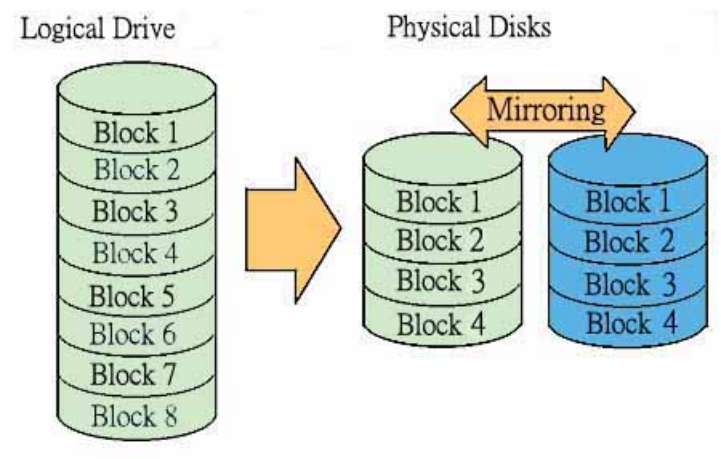
5-30. ábra: RAID-0

Mint látható, a blokkok felváltva helyezkednek el, így az történik, hogy amikor egy adat felírásra kerül a RAID 0 tömbre, úgy mindkét merevlemezre történik írás. Tulajdonképpen a lemezműveletek során mindkét merevelemez használatban van, így az írás-olvasás sebessége is közel a duplájára nőhet.

Lemezre íráskor az adott fájlt a csíkozásnak megfelelő méretű darabokra kell bontani. Ha a stripe mérete túl kicsi, akkor egy nagy fájlt túl sok részre kell bontani, aminek felírása sok időt vesz igénybe, tehát az írás nem gyorsul számottevően. Ha a stripe mérete túl nagy, akkor előfordul, hogy egy kis fájlt egyáltalán nem kell szétbontani, mert az egész fájl belefér 1 adott csíkba. Ebben az esetben sem gyorsul az írás. Érdemes RAID 0 köteteken 64-128KByte-os stripe-méretet választani.

RAID 1 - Mirroring

RAID 1, eljárás alapja az adatok duplikált tárolása, azaz tükrözése (disk mirroring). Az eltárolandó információ mindig párhuzamosan két meghajtón kerül felírásra, amely meghajtópárost, a számítógép egy szimpla kapacitású logikai meghajtónak lát. Az adatok olvasása párhuzamosan történik a két diszkről, felgyorsítván az olvasási teljesítményt. Az írás normál sebességgel, párhuzamosan történik a két meghajtón. Az eljárás igen jó hibavédelmet biztosít, bármely meghajtó meghibásodása esetén folytatódhat a működés. Ezen nagymértékű hibatolerancia ára a kétszeres tárolókapacitás-felhasználás. A RAID 1 önmagában nem használja a sávokra bontás módszerét. A RAID 0 meghajtóit RAID 1 megkettőzött meghajtókkal helyettesítve kapjuk a kombinált RAID 10-et.



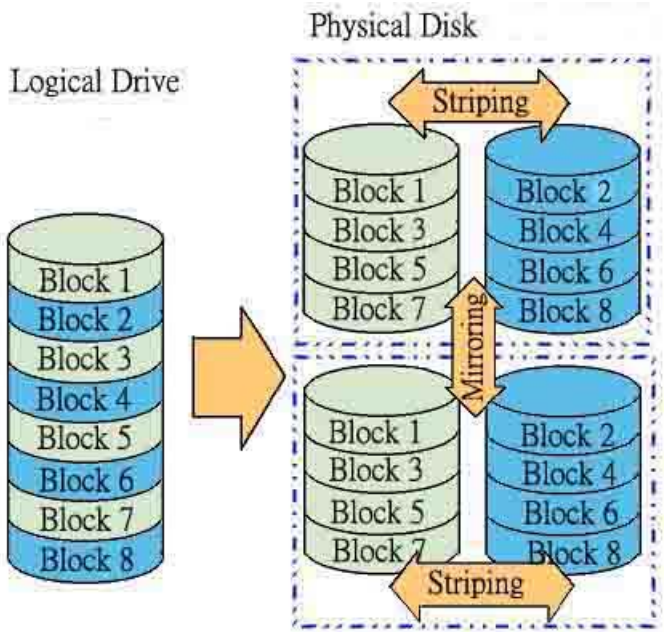
Forrás: http://www.pcguide.com

5-31. ábra: RAID-1

Hátránya, hogy hiába van két merevelemezünk, nem lesz gyorsabb az átvitel, és csak egyik merevlemezünknek megfelelő kapacitást használhatjuk ki, mivel a másik backup funkciót lát el, és ugyanazt az adatot tartalmazza.

RAID 1+0

A RAID 1-nek, és a RAID 0-nak is megvan a maga előnye. A RAID 0 meghajtóit RAID 1 megkettőzött meghajtókkal helyettesítve kapjuk a kombinált RAID 1+0-et. Ebben kombinálva van a RAID 0 gyorsasága a RAID 1 biztonságával. Persze ennek ára van, ami nem kevesebb, mint 4 merevlemezünkbe kerül. A RAID 1+0 esetén két merevlemez RAID 0-hoz hasonló módon működik, míg a másik kettő a RAID 1-hez hasonlít, azaz a RAID 0 tömbökön lévő adatot tükrözik (tartalmazzák). Ennek előnye, hogy gyors és megbízható. Hátránya, hogy nem tudjuk mind a négy merevlemez kapacitását kihasználni (mivel a négyből kettő az első kettő tükörképe).

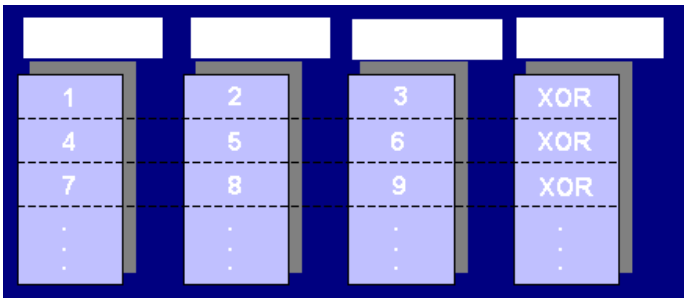


Forrás: http://www.pcguide.com

5-32. ábra: RAID 0+1

RAID 2 használja a sávokra bontás módszerét, emellett egyes meghajtókat hibajavító kód1 (ECC: Error Correcting Code) tárolására tartanak fenn. Ezen meghajtók egy-egy sávjában a különböző diszkeken azonos pozícióban elhelyezkedő sávokból képzett hibajavító kódot tárolnak. A módszer esetleges diszkhiba esetén képes annak detektálására, illetve kijavítására. Manapság nem használják, mivel a (SCSI) meghajtókban már minden egyes szektorban az adott szektorhoz tartozó ECC is eltárolásra kerül.

RAID 3 felépítése hasonlít a RAID 2-re, viszont nem teljes hibajavító kód, hanem csak egy diszknyi paritásinformáció kerül eltárolásra. Egy adott paritássáv a különböző diszkeken azonos pozícióban elhelyezkedő sávokból XOR művelet segítségével kapható meg. A rendszerben egy meghajtó kiesése nem okoz problémát, mivel a rajta lévő információ a többi meghajtó (a paritást tároló meghajtót is beleértve) XOR-aként megkapható. Az alapvető különbség a RAID 2-ben alkalmazott hibajavító kóddal szemben, hogy itt feltesszük, hogy a meghajtó meghibásodását valamilyen módón (pl. többszöri sikertelen olvasás hatására) észleljük, majd a meghibásodott diszken lévő információt a többi diszken lévő adatok segítségével állítjuk elő. A RAID 2 a diszkhibák ellen is védelmet nyújt, pl. egyes bájtok megsérülése esetén. (*Vegyük észre, hogy csak az XOR-os paritásbit technikát használva az egyik meghajtón egy adott bájt megsérülése esetén, csak azt vennénk észre, hogy a különböző meghajtókon az azonos sávba tartozó részek XOR-a nem nullát adna, de nem tudnánk sem azt, hogy melyik meghajtón van a hiba, sem azt, hogy hogyan javítsuk ki. Ezért van szükség a szektoronkénti hibajavító kód alkalmazására.*).



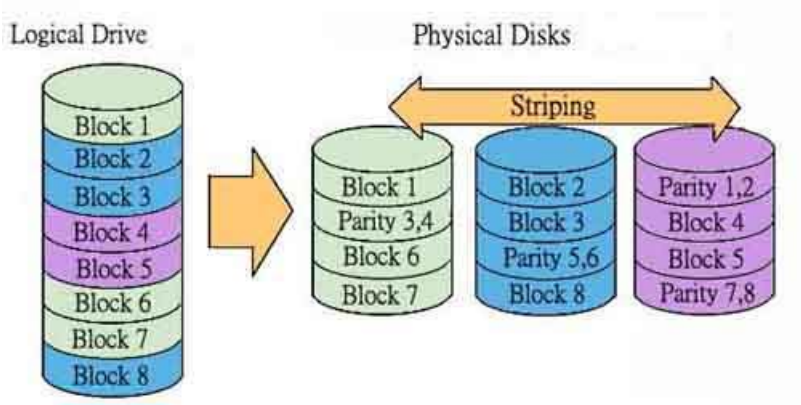
5-33. ábra: RAID 3, illetve RAID 4 három normál és egy paritás meghajtó esetén

A **RAID 3**-nál kisméretű sávokat definiálnak, így az egyes fájlok olvasása és írása párhuzamosan történhet (Természetesen a paritássávot minden egyes íráskor módosítani kell, amihez szükséges a korábbi tartalom kiolvasása. Viszont például fájltranszfer esetén, pont a kisméretű sávok miatt, az azonos pozícióban lévő sávok általában az összes diszken felülírásra kerülnek, így ez esetben a probléma kevésbé jelentkezik.) az egyes meghajtókon, viszont a módszer nem támogatja egyszerre több kérés párhuzamos kiszolgálását (single-user mode).

A **RAID 4** felépítése a RAID 3-mal megegyező. Az egyetlen különbség, hogy itt nagyméretű sávokat definiálnak, így egy rekord egy meghajtón helyezkedik el, lehetővé téve egyszerre több (különböző meghajtókon elhelyezkedő) rekord párhuzamos írását, illetve olvasását (multi-user mode). Problémát okoz viszont, hogy a paritás meghajtó adott sávját minden egyes íráskor frissíteni kell (plusz egy olvasás és írás), aminek következtében párhuzamos íráskor a paritás meghajtó a rendszer szűk keresztmetszetévé (bottleneck) válik. Ezenkívül valamely meghajtó kiesése esetén a rendszer olvasási teljesítménye is lecsökken, a paritás meghajtó jelentette szűk keresztmetszet miatt.

RAID 5 a leggyakrabban alkalmazott módszer, nagy merevlemez kapacitások esetén. A technológia már 3 merevlemezzel használható, hibatűrő rendszert alkot. Az alkalmazott merevlemezek számának növekedésével a lemezelérési sebességet növelhetjük és a visszaállíthatóság érdekében tárolt adatok miatti kapacitásveszteséget csökkenthetjük. Ez

három merevlemez alkalmazásánál csak 33%. Ha egy merevlemez meghibásodik, a másik kettő veszi át a szerepét, mely folyamat meglehetősen időigényes, ez teljesítményvesztést okoz, de adat nem veszik el. Ha egységet cserélünk, a RAID felépítése időigényes és átmenetileg jelentős teljesítménycsökkenést okoz.



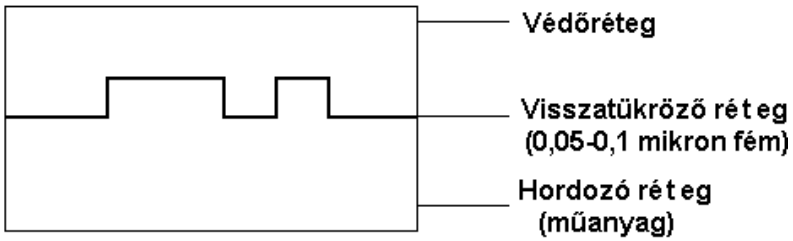
Forrás: http://www.pcguide.com

5-34. ábra: RAID 5

5.2.4. Optikai tárolók: CD ROM

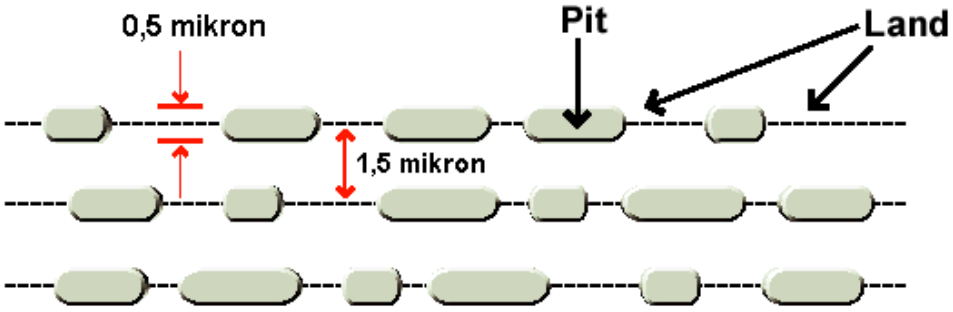
A digitális adatok optikai tárolására a kompakt lemezeket (CD = Compact Disk) használják, amelyeknek legismertebb formája a csak olvasható CD-ROM.

Az adatrögzítés egy három rétegű adathordozón történik:



5-35. ábra: *CD ROM-ok rétegei*

Az adatok tárolása a visszatükröző réteg bemélyedéseivel (pit), illetve változatlan felületével (land) történik. Valójában a pit és a land is logikai "0"-t jelent, a logikai "1" a kettő közötti átmenetnek felel meg. Az olvasás egy lézersugár visszaverődésének érzékelésével történik.

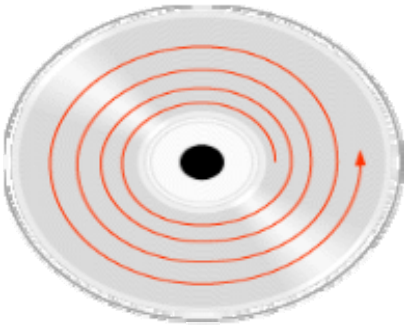


5-36. ábra: *Pit-ek és land-ek*

Az adathordozó egy 12 cm átmérőjű és 1,2 mm vastag korong, amelyre az adatok spirál formában kerülnek felírásra.

A spirálon az adattárolás szektorokban történik. Egy szektor mérete kb. 2 Kbájt, és az általános felépítése a következő:

Szinkronizációs mező	Fejléc a szektorazonosítókkal	Adatrész	EDC	ECC



5-37. ábra: Spirális adatpálya a CD ROM-on

Az adatok felírásánál hibaellenőrző (EDC = Error Detecting Code) és hibajavító (ECC = Error Correcting Code) kódolást alkalmaznak.

A lemez elején található egy tartalomjegyzék (TOC = Table of Contents), amely a lemezen található adatokról és ezek helyéről ad információt. Az újabb meghajtók (CD-írók) lehetőséget adnak az adatok több szesszióban (multi-session) történő felírására, ami azt jelenti, hogy később újabb információt lehet hozzáadni az eddigihez a fennmaradt szabad területen. Ebben az esetben a tartalomjegyzéket aktualizálni kell, és még egyszer felírni. Olvasásnál a meghajtónak támogatnia kell ezt a formátumot, mivel több tartalomjegyzék között kell keresnie a lemezen.

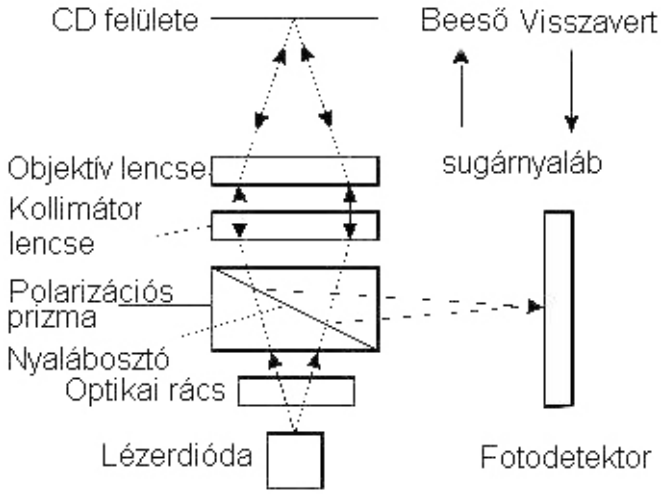
A régebbi meghajtók a korongot változó szögsebességgel forgatták, abból a célból, hogy az olvasófej mindig azonos sebességgel haladjon a lemez fölött (CLV = Constant Linear Velocity). A korszerűbb, magas átviteli sebességgel dolgozó meghajtók állandó szögsebességet használnak (CAV = Constant Angular Velocity).

A legelterjedtebb optikai tárolók különböző típusait az 5.11 táblázatban foglaltuk össze.

5-11. Táblázat: Optikai tárolók

Rövidítés	Értelmezés	Leírás
CD-DA	Compact Disk Digital Audio	Zene tárolására szolgáló CD
CD-ROM	Compact Disk Read Only Memory	Préselt korong formátumban gyártott, csak olvasható 650 Mbájt kapacitású optikai lemez (ISO 9660)
CD-R	Recordable Compact Disk	CD-íróval írható és CD-ROM-meghajtóval olvasható optikai lemez
CD-RW	ReWritable Compact Disk	CD-íróval többször írható optikai lemez
DVD-ROM	Digital Versatile/Video Disk Read Only Memory	Csak olvasható digitális videolemez (4.7, 8.5, 17 Gbájt)
DVD-RAM	Digital Versatile/Video Disk Random Access Memory	Olvasható és írható video-lemez (4.7, 8.5, 17 Gbájt)

Minden CD-ROM-meghajtó optikai rendszere lézersugarat használ a lemezen lévő információ leolvasásához. Az olvasás folyamata minden kompakt lemeznél azonos, az eltérés az olvasott elektromos jelek értelmezésében van.



5-38. ábra: ***CD ROM olvasása lézerrel***

Az optikai rendszer két fő feladatot lát el:

- a CD-n tárolt (lyukak formájában rögzített) információ digitális jelekké alakítása;
- a spirálisan kanyarodó felírás pontos követése az olvasófej által.

Mind az adatok olvasását, mind a sávon tartást az optikai rendszerből sugárzott és a lemezről a lyukakról a fejbe visszaverődő lézersugárral oldották meg.

A CD-ROM-ok fejlődésével egyre gyorsabb és gyorsabb meghajtókat hoztak forgalomba. Alapsebességnek az audió CD sebességét tekintjük ami 150 Kbájt/s, és a CD-ROM-ok sebességét az ehhez viszonyított szorzószámmal fejezzük ki. A 2X meghajtók 300 Kbájt, a 4X sebességűek 600 Kbájt/s átvitelére is képesek. Ma az 52X sebességűek általánosak.

5.2.5. Az írható és újraírható CD

Az **írható CD**-k esetében a hordozó rétegben a fényáteresztő lépesség megváltoztatásával helyettesítik a pit-ek és a land-ek reflexió változásait. A lézersugár olvasásnál alkalmazott intenzitását megnövelve a CD anyagának átlátszósága megváltozik.

Az **újraírható CD**-k esetében még nagyobb energiájúra kapcsolva a lézersugarat, olyan lokális melegedést idéz elő az anyagban, hogy az visszanyeri eredeti átlátszóságát, így törölhető a lemez.

Az optikai elvű háttértárak esetében a tárolt információ típusának függvényében a tárolási szerkezet (gyakorlatilag az adatblokkok szerkezete) eltérő lehet – az egyes típusokra jellemző szabványokat különböző színnel jelölt szabványgyűjtemények („Rainbow Books” – kb. „szivárvány-könyvek”) tartalmazzák:

- „Red Book” („Vörös”): CD-DA (audió CD), az alapszabvány;
- „Yellow Book” („Sárga”): CD-ROM
- „Orange Book” („Narancssárga”): CD-ROM/XA, kiterjesztett módú CD: eredetileg képek tárolásához dolgozták ki (Photo CD), de gyakorlatilag az írható-újraírható optikai adathordozók (CD-R, CD-RW) szabványává vált;

- „White Book” („Fehér”): CD-ROM/XA mode 2: videó CD-k szabványos formátuma, stb.

5.2.6. Optikai tárolók: a DVD

A CD kapacitása akkor lett igazán kicsi, amikor a felhasználók mozgóképes alkalmazásokat készítettek, és megnőtt az érdeklődés a mozifilmek számítógépes és házi lejátszása iránt. A házi videóipar a VHS formátumnál jobb minőségű és hosszabb játékidejű médiát keresett.

1995-ben a nagy gyártók (Matsushita,Toshiba, Time Warner, Mitsubishi, Philips, Sony, Pioneer, Hitachi, Thomson és JVC) konzorciumot alkottak, és megállapodtak az új, nagy kapacitású optikai tárolólemez alapvető műszaki paramétereiben és abban, hogy a neve DVD legyen. A név kezdetben Digital Video Disc-et jelentett, később a Digital Versatile Disc (sokoldalú digitális lemez) használata terjedt el. A DVD rendszer felülről kompatibilis a létező CD lemezekkel.

A DVD-ROM lemez ideális adathordozó multimédiás alkalmazásokhoz, és az alkalmazói programok terjesztésére. Az átlagos keresési idő 150-200 msec, hozzáférési idő pedig 200-250 msec körül alakul. Adatátviteli sebessége 1,3 Mbájt/sec, burst átvitelnél meghaladja 12 Mbájt/sec értéket. A meghajtók a CD-ROM-hoz hasonló felülettel illeszthetők a számítógéphez, az ATAPI (EIDE) vagy SCSI interfésszel találkozhatunk. A file-rendszer az UDF (Universal Disk Format). Az UDF legnagyobb előnye az ISO9660-al szemben, hogy támogatja az újraírható médiumok használatát, illetve az inkrementális írást és a file-törlést. emellett kijavították benne az ISO9660 sok hibáját is (pl. 8.3-as file-nevek).

A DVD lemez tárolási kapacitása - az oldalak és tárolási rétegek számától függően - 7-25-szöröse a CD-jének. Ezt a megnövelt kapacitást a következőképpen érték el:

- a szomszédos sávok közötti kisebb távolság,
- kisebb méretű lyukak alkalmazása,
- két, egymásra helyezett adathordozó réteg lehetősége (a felső félig átlátszó),
- kétoldalas lemez kialakításának lehetősége (két 0,6 mm vastagságú lemez összeragasztásával),
- hatékonyabb logikai formátum (rövidebb hibajavító mező).

A DVD családok hasonlítanak a CD lemezeknél fellelhető családokra, azzal a különbséggel, hogy a DVD technológia még mindig újabb és újabb szabványokkal újul meg. DVD-R és a DVD+R az írható, a DVD-RW, és a DVD+RW az újraírható típusokat jelöli.

A DVD+RW szabvány (amely magában foglalja az egyszer írható DVD+R és az újraírható DVD+RW lemezeket) az újabb, technológiailag fejlettebb szabvány, gyorsabb és könnyebb, ezáltal olcsóbb a gyártású technológiát ír le, mint a DVD-R. A DVD-k tárolókapacitás szerint csoportosított típusait mutatja az 5.12 táblázat.

5-12. Táblázat: DVD típusok

Típus	Felépítés	Kapacitás
DVD-5	Egyrétegű egyoldalas lemez	4,7 Gbájt
DVD-9	Kétrétegű egyoldalas lemez	8,5 Gbájt
DVD-10	Egyrétegű kétoldalas lemez	9,4 Gbájt
DVD-18	Kétrétegű kétoldalas lemez	17 Gbájt

5.2.7. Elektronikus elvű hordozható tárak

Az elektronikus elven működő háttértárak a tároló típusú perifériák legújabb generációját képviselik. A különböző típusú memória-kártyák és a tárolási képesség mellett általában további szolgáltatásokkal is felruházott pen-drive-ok ugyan hordozható tárolóeszközök, de működési jellemzőik tekintetében sokkal közelebb állnak a memóriákhoz, mint a háttértárakhoz. Működési elvüket tekintve visszautalunk a 3.1.3 fejezetben ismertetett Flash memóriákra.

Ellenőrző kérdések:

1. Mit nevezünk IBM PC-nek, vagy személyi számítógépnek?
2. Sorolja fel a PC-k legfontosabb tulajdonságait
3. Mit nevezünk x86-os utasításkészletnek?
4. Mi a különbség az x86-os emuláció és a natív utasítás végrehajtás között?
5. Mi főbb jellemzője a PC-nek, az XT-nek és az AT-nak?
6. Hasonlítsa össze a 386-os és a 486-os processzort!
7. Hasonlítsa össze a 486-os és a Pentium processzort!
8. Ismertesse a Pentium II., a PIII, és a P4 közötti főbb különbségeket!
9. Milyen processzorok tartoznak a 8. generációba?
10. Mi az alaplap , és mi a szerepe?
11. Milyen főbb részek találhatók általában az alaplapokon?
12. Mi a BIOS, és mi a szerepe?
13. Mi a CMOS és milyen kapcsolatban van a BIOS-sal?
14. Mi az EIDE, és milyen jellemzői vannak?
15. Mi az SCSI és mi a különbség közte és az EIDE között?
16. Ismertesse a CD- fontosabb jellemzőit!
17. Mi a különbség a CD és a DVD között?

Irodalomjegyzék:

Abonyi Zsolt: PC hardver kézikönyv. Computerbooks 1992

Agárdi Gábor – Hadi János: Fókuszban a Pentium. LSI 1996

AMD: 3D Now! Technology Manual 1998

AMD-K5 Processor Technical Reference Manual 1996

AMD-K5 Processor. Data Sheet 1996

AMD-K6-2 Processor Data Sheet 1998

AMD-K6-2 Processor, 100-MHz Bus Specification 1998

AMD K7 http://195.56.48.34/pcabc/hardver/K7.htm

AS-400-as rendszer http://www.hu.ibm.com/products/as400

Az Intel Pentium Pro, Pentium MMX és Pentium II processzorai
http://info3.tech.klte.hu/~mudry/pentium/pentium.html

Budai Attila: Mikroszámítógépek

Coelho, Rohon – Hawash, Maher: DirectX, RDX, RSX and MMX technology. Addison-Wesley Developers Press 1997

Cserny László: Mikroszámítógépek. LSI 1996

Cserny László: RISC processzorok. LSI 1995

Dembowski, Klaus: PC táblázatok. Adatok és tények a PC hardverhez. Kossuth 1997

Dickschus, Arthur: Egyszerűen PC ismeretek, Hardver. Panem 1998

Németh Gábor, Horváth László: Számítógép-architektúrák

http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm

http://www.intel.com/products/desk_lap/processors/

http://www.geek.com/procspec/procspec.htm

http://developer.intel.com/design/pentium4

http://developer.intel.com/design/pentiumIII

http://developer.intel.com/design/itanium

http://developer.intel.com/design/itanium2

http://www.amd.com/

http://www.intel.com

http://www.cyrix.com

http://developer.intel.com/design/pentium4

http://developer.intel.com/design/pentiumIII

http://developer.intel.com/design/itanium

http://developer.intel.com/design/itanium2

http://www.pctechguide.com/

http://www.tomshardware.hu/storage/index.html

http://www.computer-tutorial.de/store/storage.html

http://www.hardwaregrundlagen.de/

http://www.tweakhardware.com/

Ábrajegyzék:

1-1. ábra: A Pascaline kinyitva, és használat közben 8
1-2. ábra: *A Mark I.* 10
1-3. ábra: *Az ENIAC számítógép* 11
1-4. ábra: *Első generációs számítógépek alkatrészei* 11
1-5. ábra: *Első generációs számítógép blokkvázlata* 12
1-6. ábra: *Az IBM7094 gépe* 13
1-7. ábra: *Második generációs számítógép blokkvázlata* 14
1-8. ábra:: *IBM System/360- as számítógép* 15
1-9. ábra: Az IBMSystem/360 harmadik generációs számítógép blokkvázlata 16
1-10. ábra: Sínrendszerre alapozott architektúra. A harmadik generációs gépeknél jelenik meg, és a negyedik generációban válik általánossá. 16
1-11. ábra: *A számítási modell értelmezése.* 18
1-12. ábra: *A Turing gép* 19
1-13. ábra: A b^2 - 4ac kiszámítása adatfolyam struktúrában. 21
1-14. ábra: *A problémaleírás modellje* 21
1-15. ábra: *A probléma leírás jellege* 22
1-16. ábra: *A végrehajtás modellje* 23
1-17. ábra: A végrehajtás folyamatának alapvető vezérlési módjai ..... **Hiba! A könyvjelző nem létezik.**
1-18. ábra: A számítási modell, a programozási nyelv és a számítógép kapcsolata 23
1-19. ábra: A számítási modell, a programozási nyelvek és a számítógép megjelenési sorrendje. 23
1-20. ábra: *A hardver tervezés rétegmodellje* 25
2-1. ábra: A processzor architektúrális elemei 28
2-2. ábra: Utasítás végrehajtásban szereplő fontosabb részek 40
2-3. ábra: Harvard architektúra közös adat-, és utasítás-memóriával 45
2-4. ábra: Harvard architektúra külön adat-, és utasítás-memóriával 45
2-5. ábra: Flynn által bevezetett architektúra osztályok 46
2-6. ábra: Single Instuction, Single Data Stream típusú utasítás végrehajtás 46
2-7. ábra: *A SIMD típusú számítógép működése.* 46
2-8. ábra: *A MISD típusú számítógép működése.* 47
2-9. ábra: *A MIMD típusú számítógép működése.* 47
2-10. ábra: Futószalag (pipeline) időbeli párhuzamosság ILP processzorokban 49
2-11. ábra: Többszörözés (térbeli párhuzamosság) az ILP-processzorokban 49
2-12. ábra: *Soros utasításvégrehajtás* 50
2-13. ábra: *Pipeline utasításvégrehajtás* 50
2-14. ábra: Egy skalár és egy négyszeres kibocsátású szuperskalár processzor működése 53
2-15. ábra: Előredekódolás elve 54
2-16. ábra: A hagyományos VLIW- és a szuperskalár processzorok közös alapstruktúrája 55
2-17. ábra: *A tradicionális architektúrák erőforrás kihasználása* 56
2-18. ábra: *Explicit párhuzamosítás az IA64-ben* 56
2-19. ábra: Vektorszámítógépek felépítése 57
2-20. ábra: Tömbprocesszoros számítógépek 58
2-21. ábra: A párhuzamos architektúrák osztályozása 60
2-22. ábra: Neurális hálózatok sejtprocesszorainak egyszerűsített ábrája 60
3-1. ábra: *Abszolút címzési mód* 63
3-2. ábra: *Relatív címzés* 63
3-3. ábra: *Indirekt címzési mód* 63
3-4. ábra: *Közvetlen adatcímzés* 64
3-5. ábra: *Indexelés* 64
3-6. ábra: *A PC-kben található memóriák* 67
3-7. ábra: *Regisztertárak megoldásai* 68
3-8. ábra: A processzorok és a dinamikus memóriák sebességnövekedése 69
3-9. ábra: Tárolók hozzáférési ideje és tárolókapacitásuk nagyságrendje 70
3-10. ábra: *A lokalitás elve* 71
3-11. ábra: *A cache működése* 72
3-12. ábra: *Cache tárak általános felépítése* 72
3-13. ábra: *Közvetlen leképezésű cache.* 74
3-14. ábra: *Csoport asszociatív tároló* 74

3-15. ábra: *Pufferelt közvetlen átírás módszere* 75
3-16. ábra: *Cache tárolók multiprocesszoros rendszerben.* 76
3-17. ábra: A tároló hierarchia egyes szintjei közötti adatáramlás 78
3-18. ábra: *A virtuális tár felosztása blokkokra* 79
3-19. ábra: *Virtuális címszámítás* 79
3-20. ábra: *Virtuális és fizikai címek viszonya* 81
3-21. ábra: *Virtuális és fizikai cím megfeleltetés* 81
3-22. ábra: *Az MMU működése* 81
3-23. ábra: *Szegmentálás* 82
3-24. ábra: *Lapozás* 83
3-25. ábra: *Laptáblázatok* 83
3-26. ábra: *Címszámítás ha a lap nincs a főtárban* 84
4-1. ábra: A számítógép főbb egységeinek kapcsolata sínrendszerrel 87
4-2. ábra: *A megszakítások okai vagy forrásai* 89
4-3. ábra: *Egyszintű megszakítási rendszer* 90
4-4. ábra: *Többszintű megszakítási rendszer* 91
4-5. ábra: Szinten belül egyszintű, szintek között többszintű megszakítási rendszer 91
4-6. ábra: *A DMA logikai sémája* 93
4-7. ábra: *Perifériák architektúrája* 94
4-8. ábra: *I/O illesztőegységek kapcsolódása a sínre* 96
4-9. ábra: *A rendszersín* 99
4-10. ábra: *Kétszintű sínrendszer elve* 99
4-11. ábra: *Kétszintű sínrendszer sínvezérlővel* 100
4-12. ábra: *Háromszintű sínrendszer* 100
4-13. ábra: *Master és slave eszközök viszonya a buszon* 101
4-14. ábra: *I/O eszközök memóriába ágyazott címzése* 103
5-1. ábra: Pentium Pro processzor 111
5-2. ábra: A P-III processzor felépítése 113
5-3. ábra: Az AMD K7 Athlon kommunikációs architektúrája 115
5-4. ábra: Az AMD K7 processzora 116
5-5. ábra: A Pentium 4 117
5-6. ábra: *Az elágazások mindkét ágának végrehajtása (Predication)* 119
5-7. ábra: Az AMD 64 bites Hammer processzorának blokkjai 120
5-8. ábra: Alaplapi sínrendszerek sebessége 123
5-9. ábra: *A chip-set-ek kialakulása* 129
5-10. ábra: *Memória vezérlő* 130
5-11. ábra: *I/O vezérlő* 130
5-12. ábra: *Bridge architektúra* 131
5-13. ábra: *HUB architektúra* 132
5-14. ábra: Intel 975X Express chip-set blokk vázlata 133
5-15. ábra: *Rendszerbusz és I/O busz* 134
5-16. ábra: *A frontside és a backside busz* 134
5-17. ábra: *Közös I/O busz* 135
5-18. ábra: *I/O bridge* 136
5-19. ábra: PCI busz és az I/O eszközök kapcsolata 137
5-20. ábra: A PCI bridge felépítése 137
5-21. ábra: *USB kapcsolatok logikai ábrája* 138
5-22. ábra: USB eszközök csatlakoztatása Hub-okon keresztül 139
5-23. ábra: *Soros átvitel modem közbeiktatásával* 140
5-24. ábra: *A mágnesszalag szerkezete* 141
5-25. ábra: *A merevlemez sematikus ábrája* 143
5-26. ábra: *EIDE vezérlő és csatlakoztatott egységek* 145
5-27. ábra: EIDE vezérlő adatátviteli lehetőségei. 146
5-28. ábra: *SCSI busz* 147
5-29. ábra: RAID-0 150
5-30. ábra: RAID-1 151
5-31. ábra: RAID 0+1 151
5-32. ábra: RAID 5 153
5-33. ábra: RAID 3, illetve RAID 4 három normál és egy paritás meghajtó esetén 152
5-34. ábra: *CD ROM-ok rétegei* 153

5-35. ÁBRA: *PIT-EK ÉS LAND-EK* 153
5-36. ÁBRA: SPIRÁLIS ADATPÁLYA A CD ROM-ON 154
5-37. ÁBRA: *CD ROM OLVASÁSA LÉZERREL* 155

Táblázatok jegyzéke:

1-1. Táblázat: *A Neumann-féle és az adatfolyam számítási modell összehasonlítása* 21
1-2. Táblázat: *A VHDL absztrakciós szintjei* 26
2-1. Táblázat: *A gépi kód és az assembly utasítás-szerkezete* 32
2-2. Táblázat: Bitmozgató utasítások hatása 35
2-3. Táblázat: CISC és RISC processzorok összehasonlítása 44
2-4. Táblázat: *Az ILP architektúrák fő fejlődési útja* 49
3-1. Táblázat: Adatelérési idő a tárhierarchia szerint 76
4-1. Táblázat: *Az IBM PC-k kötött megszakításai* 92
5-1. Táblázat: A Pentium II processzorok különböző típusai 112
5-2. Táblázat: P-II alapú Celeron processzorok 112
5-3. Táblázat: A Pentium-III processzor különböző változatai 114
5-4. Táblázat: Összefoglaló táblázat az AMD hetedik generációs processzorairól 115
5-5. Táblázat: A Pentium 4 processzor különböző típusainak összefoglaló táblázata 118
5-6. Táblázat: Összefoglaló táblázat az Intel 64 bites processzorairól 119
5-7. Táblázat: 120
5-8. Táblázat 121
5-9. Táblázat: *Alaplapi sínrendszerek fejlődése* 123
5-10. Táblázat: *A PCI szabvány egyes verzióinak összehasonlító táblázata* 136
5-11. Táblázat: Optikai tárolók 154
5-12. Táblázat: DVD típusok 156

Ellenőrző kérdések

A kérdésre kattintva megkapja a választ

1. fejezet

1.1. Melyek a számítógép fő funkcionális egységei?
1.2. Sorolja fel a Neumann János által megfogalmazott, a számítógépek struktúrájára vonatkozó alapelveit!
1.3. Mutassa be a második generációs számítógépek elvi felépítését (blokkvázlatát)!
1.4. Melyek az LSI technológia főbb alkalmazási területei a negyedik generációs számítógépekben?
1.5. Milyen adatalapú számítási modelleket ismer?
1.6. Mit értünk számítógép-architektúra alatt?

2. fejezet

2.1. Melyek a CPU legfontosabb architekturális építőelemei?
2.2. Mi a feladata a vezérlő egységnek?
2.3. Milyen regiszter-kategóriákat ismer?
2.4. Melyek a Pentium processzorok üzemmódjai?
2.5. Határozza meg az utasítás-készlet fogalmát!
2.6. Melyek a fontosabb címzésmódok?
2.7. Soroljon fel az utasítások működésének alapján megkülönböztethető utasítás-típusokat!
2.8. Magyarázza el a verem működését!
2.9. Hogyan határozható meg egy feladat végrehajtásának időigénye?
2.10. Hasonlítsa össze a CISC és a RISC processzorokat!
2.11. Mit értünk utasításfolyam és adatfolyam alatt?
2.12. Melyek a párhuzamos architektúrák alapvető típusai?
2.13. Melyek a pipeline technika alkalmazása során előforduló problémák kezelésére szolgáló módszerek?
2.14. Melyek a tömbprocesszoros rendszerek jellemzői?
2.15. Ismertesse a párhuzamos architektúrák korszerű (Sima szerinti) csoportosítását!

3. fejezet

3.1. Mit értünk (operatív) memória alatt?
3.2. Hogyan működik az abszolút memóriacímzés?
3.3. Hogyan működik az indirekt memóriacímzés?
3.4. Mi a Hamming-távolság?
3.5. Mi a különbség az elérési idő és a ciklusidő között?
3.6. Mi a különbség az SRAM és a DRAM között?
3.7. Mi a cache (gyorsító, rejtett) tár?
3.8. Mit értünk asszociatív tárolón?
3.9. Ismertesse a számítógépekben alkalmazott tároló-hierarchiát!
3.10. Mit értünk virtuális memória alatt?
3.11. Mi a szegmentálás?
3.12. Milyen lapkiosztási elveket ismer?
3.13. Milyen lapcsere-stratégiákat ismer?

4. fejezet

4.1. Mi a különbség a megszakítások és a kivételek között?
4.2. Ismertesse az egyszintű és a többszintű megszakítási rendszer közötti különbségeket!
4.3. Melyek a megszakítások kiszolgálásának lépései?
4.4. Mi a közvetlen memória-hozzáférés (DMA) lényege?
4.5. Mit értünk illesztő (interface) alatt?
4.6. Hogyan csoportosítható (az átvitt információ jellege szerint) a számítógép sínrendszere?
4.7. Ismertesse a kétszintű sínrendszer elvét!
4.8. Ismertesse az I/O műveletek végrehajtásának lépéseit!
4.9. Melyek az I/O átvitel alapvető típusai?

5. fejezet

5.1. Mikor nevezünk egy személyi számítógépet „IBM PC kompatibilis”-nek?
5.2. Jellemezze az első generációs Intel processzorokat!
5.3. Melyek a 486-os (negyedik generációs) processzoroknál alkalmazott fontosabb technológiai újítások?
5.4. Mi a különbség a Pentium II és a P-II Celeron processzorok között?
5.5. Mi az Intel 8. generációs processzorainak legfontosabb jellemzője?
5.6. Hogyan támogatja a 32 bites alkalmazásokat az AMD 8. generációs processzora?
5.7. Milyen főbb részek találhatók általában az alaplapokon?
5.8. Mi a PnP?
5.9. Mi a BIOS, és mi a szerepe?
5.10. Mi a chip-set?
5.11. Mi a (bridge architektúrájú) chip-setekben az északi és a déli hidak szerepe?
5.12. Mi a PCI?
5.13. Hogyan csoportosíthatjuk a háttértárakat?
5.14. Mi az IDE csatolók működésének alapelve?
5.15. Mi a SCSI?
5.16. Miben áll a RAID technológia jelentősége?
5.17. Sorolja fel és jellemezze a fontosabb optikai tárolókat!

Internetes linkek, hasznos weboldalak

Gyártók:

- Intel: http://www.intel.com
- AMD: http://www.amd.com
- Cyrix: http://www.intel.com

Leírások, dokumentumok, ismertetők:

- áttekintések, leírások: http://www.pctechguide.com (angol)
- cikkek, elemzések, értékelések: http://www.tomshardware.com (angol)
- processzorok összehasonlítása: http://www.geek.com/procspec/procspec.htm (angol)
- doksi.hu: http://www.doksi.hu/doksik.php?order=subcat&fid=71