BUDAI ATTILA

MIKROSZÁMÍTÓGÉP-RENDSZEREK

LSI OKTATÓKÖZPONT

BUDAI ATTILA

MIKROSZÁMÍTÓGÉP-RENDSZEREK

Nyitott rendszerű képzés
– távoktatás – oktatási segédlete
Felsőoktatási tankönyv

Gábor Dénes Főiskola
BUDAPEST, 2001

Lektorálta: **Dr. Kónya László**

Dr. Szittya Ottó

Szerkesztés, tördelés: **OFFICE-CAR Bt.**
2040 Budaörs Csata u. 17.

A könyv megrendelhető vagy megvásárolható:

LSI Oktatóközpont
1037 Budapest Bécsi út 324.
Tel/fax: 250-6013

ISBN 963 577 278 5

Kiadó: **LSI Oktatóközpont**
Felelős vezető: **Dr. Kovács Magda**
Témafelelős: **Flier István**

LIGATURA KFT – NASZÁLY PRINT KFT

TARTALOMJEGYZÉK

ELŐSZÓ

A „Mikroszámítógéprendszerek" című jegyzet a hardver legfontosabb fogalmait, rendszertechnikai megoldásait mutatja be. Azokat az alapfogalmakat, működési elveket, a számítógépek teljesítménynövelésének eszközeit és eljárásait, az egyes számítógép architektúrák jellemzőit foglalja össze a jegyzet, melyek a hardver gyors fejlődése mellett is viszonylag időtállóak.

A jegyzetben megtalálható hardver alapismeretek jórészt általánosak, így egyaránt érvényesek a nagy-közép és mikroszámítógép architektúrákra. Ennek ellenére a tananyag kifejtése során főként a legelterjedtebb személyi számítógépek (PC-k) jellemzőire fogunk koncentrálni, így a jegyzetben megtalálható példák döntő része is ezekre vonatkozik.

A jegyzet első bevezető fejezetében röviden áttekintjük a hardver fogalmát, és rétegmodelljét, az elmúlt évtizedek hardver fejlődésének legfontosabb tendenciáit, és a VHDL szabványos hardver leíró nyelv jellemzőit.

Ezt követően a számítógéprendszer architektúrákkal kapcsolatos alapismereteket tárgyaljuk, bemutatjuk a legfontosabb számítógéprendszertípusokat, ezek működésének lényegét, fontosabb jellemzőiket és üzemmódjaikat.

Ezt követően a számítógépek processzorait mutatjuk be, így az aritmetikai-logikai egység működését, a műveleti vezérlés különböző formáit, különös tekintettel az utasításvégrehajtás gyorsítására (pipeling). E fejezet végén napjainkban elterjedten használt szuperskalár processzorok legfontosabb jellemzőit foglaljuk össze.

A jegyzet tárolókezeléssel foglalkozó fejezete a tárolóhierarchia egyes elemeit (regisztertár, cache, főtár, virtuális tár) és jellemző rendszertechnikai megoldásait, valamint a számítógépek tárolóvédelmi rendszerét tárgyalja.

A jegyzet utolsó fejezete a mikroszámítógéprendszerek részegységei közötti kapcsolatokkal foglalkozik, a sín (busz) rendszer működését, a megszakítási rendszert és az I/O műveletek végrehajtását mutatja be.

A jegyzetben tárgyaltak lényegét a dőlt betűs (kurzív) szedés emeli ki és a tanagyag feldolgozását a távoktatásban szokásosan használt szimbolikus jelek próbálják meg megkönnyíteni.

A tanagyag elsajátításának ellenőrzését a fejezetek végén található kérdésekkel hajthatja végre az olvasó. Ezek megegyeznek a Gábor Dénes Főiskola „Mikroszámítógépek" című tantárgya vizsgakérdéseinek első részével.

A jegyzet érdemi részét egy viszonylag bő irodalomjegyzék zárja le. Ez kiindulópontul szolgálhat mindazok számára, akik a hardver felépítését és működését részletesen és mélyebben meg kívánják ismerni. A szerző külön felhívja az olvasók figyelmét a legfontosabb hardver gyártók WEB lapjainak tanulmányozására, mely nélkül a hardver gyors fejlődése naprakészen ma már nem követhető.

Végezetül a szerző köszönetet mond mindazoknak, akik közreműködésükkel hozzájárultak e jegyzet megírásához és kiadásához, és előre is elnézést kér mindenkitől, ha a többszörös átnézés és ellenőrzés után is maradtak hibák a jegyzetben.

Budapest, 1999. november

A Szerző
E-mail: budai@gdf-ri.hu

JELÖLÉSEK MAGYARÁZATA

! – Kiemelt jelentőségű kérdésekre, tananyag lényegére hívja fel a figyelmet. Ellenőrző kérdésre adja meg a választ. *Ezeket a részeket kurzívan szedtük, így elkülönülnek a tananyagban.*

 – Fontos összefüggésekre hívja fel a figyelmet.

 – Magyarázat, indoklás, példa.

 – Kiegészítő ismeretek, melyek betűformájukkal is elkülönülnek a tananyagtól. Nem tartozik szorosan a főiskolai tananyaghoz.

? – Ellenőrző kérdés(ek)

(1)2.3 – (Al)Fejezet azonosító

I.

A HARDVER FOGALMA, FEJLŐDÉSE ÉS LEÍRÁSA

1.1. A HARDVER FOGALMA

A hardver szó fogalmát a következőképpen határozhatjuk meg:

A hardver (hardware) eredeti angol jelentése „kemény áru" (vas-áru). Lényegében ez *a számítógépet alkotó, kézzel fogható eszközök összefoglaló neve, a számítógép elektronikus áramköreit, mechanikus berendezéseit, kábeleit, csatlakozásait és perifériáit nevezzük így.*

Rögtön látszik, hogy a definícióban különböző, méretében igen jelentősen eltérő eszközök szerepelnek. Ugyanis definíciónk értelmében a hardverhez tartozik a számítógép egy tranzisztora, de ugyanúgy a hardver része a 7,5 millió tranzisztort tartalmazó Pentium II processzor is.

Képzeljünk most el, hogy az előző processzor tranzisztorokból való felépítését kapcsolási rajzokkal dokumentáljuk. Ez csak olyan tömegű papírmennyiséggel lenne megoldható, ami emberileg áttekinthetetlen adathalmaz.

Hogyan lehet mégis *hardver egységeket*, például processzort tervezni és működésüket *az ember számára átlátható módon leírni, jellemezni?*

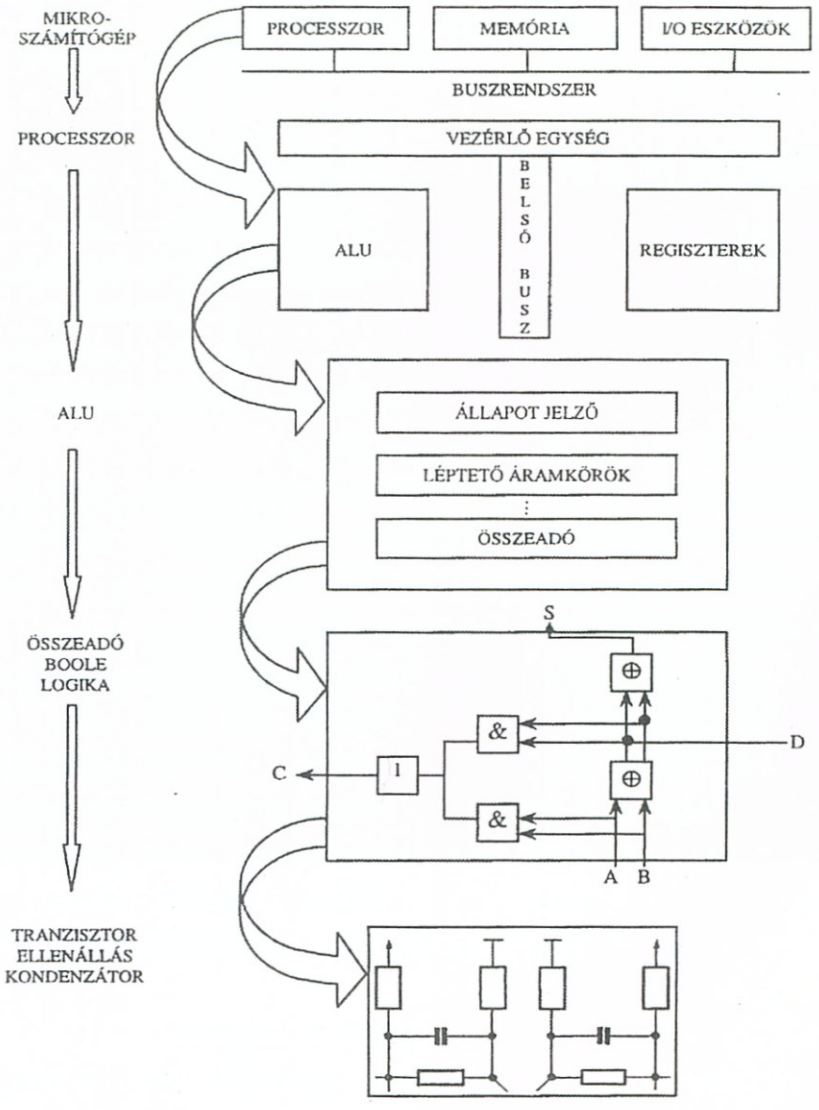
A megoldás csak az lehet, ha a hardvert hierarchikus szintekkel próbáljuk modellezni, melynél egy alsóbb rétegbeli építőelemekből épülnek fel a következő réteg hardver egységei.

Így például egy mikroszámítógép (PC) esetében:

- a PC alaplapból, processzorból, memória modulokból, kártyákból stb. épül fel,

- a processzoron belül megkülönböztethetjük a műveletvégző egységet (ALU), a vezérlő egysége (CU), a regisztertárolót stb.,
- ha az ALU működését vizsgáljuk, akkor a műveletvégző egységen belül figyelembe vesszük az állapotjelző regisztert, az akkumulátor regisztert és a különböző léptető, összeadó áramkörökből felépülő részegységeket, valamint az ezeket összekapcsoló adatutakat,
- ha már olyan részletesen szeretnénk megismerni az ALU működését, mely alapján megérthető egy összeadási művelet áramköri szintű végrehajtása, akkor az ALU összeadó egységét kapuáramköri szinten kell bemutatni,
- ha a kapuáramkörök fizikai megvalósítását nézzük, akkor ezek tranzisztorokból, kondenzátorokból, ellenállásokból és vezetékekből stb. épülnek fel.

Ezeket az összefüggéseket az 1. számú ábra szemlélteti:



1. sz. ábra: A hardver hierarchikus felépítése

1.2. A HARDVER RÉTEGMODELLJE

1.2.1. A rétegmodell szintjei

A hardver rétegmodelljében öt szintet különítünk el:

Rendszerszint (System, Architectural Level)

Rendszerszinten a számítógép hardverét félautomatikus főrészegységeivel írjuk le (Például processzor, interfész, tároló stb.). A részegységeket funkcionálisan és protokolljaikkal jellemezzük. (Például processzor esetében az utasításkészlettel.)

Algoritmikus szint (Algorithmic Level)

Az algoritmikus szinten megadjuk az egyes részegységeket alkotó hardver modulokat (például a processzor vezérlőegységből, ALU-ból, regisztertárolóból stb. épül fel), és az ezeket jellemző algoritmusokat, melyek segítségével a hardver modulok a bitsorozatokat értelmezik.

Funkcionális blokkok szintje (Functional Block, Register Transfer Level)

Itt meghatározzuk a hardver modulok funkcionális blokkokból (például regiszterek, számlálók, összeadók stb.) *való felépítését, valamint ezek műveletvégzését* (például összeadás) *és a köztük való adatátvitelt.* Itt már figyelembe vesszük a jelek időbeli ütemezését (órajel).

Logikai szint

Ezen a szinten a hardvert kapuáramkörökkel specifikáljuk, ennek eszköze a Boole-algebra. Strukturálisan ez kapuáramkörökből felépülő hardvert jelent.

Áramköri szint

Az előző szinteken *a jeleket* diszkrét értékeket hordozó digitális értékekként kezeltük. Az áramköri szinten *ezeket már folytonos mennyiségeknek tekintjük (feszültség, áramerősség stb.).* A szintnek megfelelő strukturális építőelemek a tranzisztorok, kondenzátorok stb.

A mikroszámítógépek tantárgy keretein belül a hardvert döntően rendszer és algoritmikus szinten fogjuk tárgyalni. Esetenként példákkal fogunk utalni a funkcionális blokk és logikai szintre. Az áramköri szinttel nem foglalkozunk.

Látható, hogy a rendszerszinttől az áramköri szintig az építőelemek mérete fokozatosan csökken, az egyes rétegek a megfelelő modellek absztrakciós szintjében is különböznek.

1.2.2. A hardver rétegek jellemzése

Az egyes rétegeket és ezek egymáshoz való viszonyát három fő szempont szerint szokták vizsgálni. Ezek:

! *Strukturális felépítés*

Ezzel írjuk le, hogy az egyes rétegekhez tartozó hardver elemek, milyen alacsonyabb szintű építőelemekből és milyen szerkezetben épülnek fel.

! *Viselkedési modell*

Ez határozza meg az egyes rétegekhez tartozó építőelemek működésének modellezését. Például:

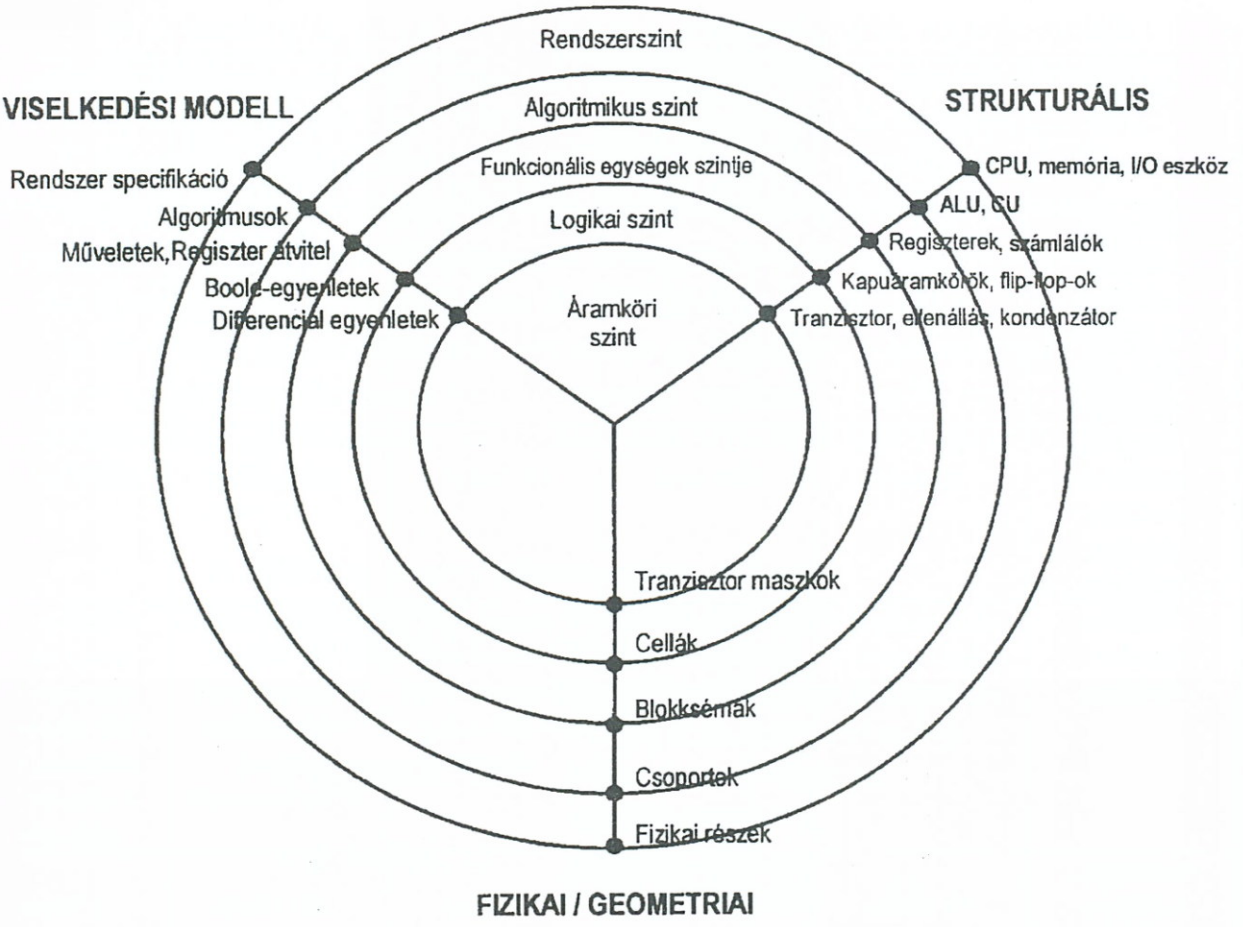
- Az áramkörök működését az elektromosságtan differenciálegyenleteivel,
- A kapu áramkörök működését a Boole-algebra egyenleteivel

jellemezzük.

Fizikai, geometriai jellemzők

! Ezzel írjuk le az egyes szintekhez tartozó hardverelemek jellemző formáját, méretét stb.

A hardver rétegmodelljét a Y diagrammal szokták szemléltetni (lásd 2. sz. ábra).



2. sz. ábra
A hardver rétegmodelljének Gajski-Kuhn féle Y diagrammja

1.3. A hardver fejlődésének jellemzői

A számítógépes hardver fejlődésében ún. számítógépgenerációkat szoktak megkülönböztetni.

Az egyes számítógépgenerációkat egymástól az alkalmazott alapáramkörök típusa, és a műveleti sebesség nagyságrendje szerint különböztetjük meg. !

A számítógépgenerációk legfontosabb jellemzőit a 3. sz. ábra foglalja össze.

Jellemzők	Első generáció kb. 1950–1955	Második generáció kb. 1956–1964	Harmadik generáció kb. 1965–1977	Negyedik generáció kb. 1978–1990	Ötödik generáció kb. 1991–2005
Műveleti sebesség maximumának (művelet/sec) nagyságrendje	10 ezer	1 millió	10 millió	100 millió	1000 millió
Központi memória max. méretének (bájt-ban) nagyságrendje	50 ezer	500 ezer	10 millió	100 millió	1000 millió
Jellemző hardver építőelemek	*Elektroncső, mágnesdob és szalag, relék*	*Tranzisztorok Mágneslemez Ferrit-mátrix*	*Integrált áramkörök félvezetőtárak 16 kilobites chipek*	*LSI áramkörök, mikroprocesszorok 64 kilobites chipek Winchesterek*	*VLSI, programozható áramkörök 128 kilobites chipektől – 1 Gigabites chipekig*
Maximális háttértár kapacitás nagyságrendje	10 Mbájt	1000 Mbájt	10 Gbájt	100 Gbájt	1 Tbájt
Jellemző szoftver elemek	Gépi kód programkönyvtár	Operációs rendszer programnyelvek	Többfelhasználós operációs rendszer fejlett 3 generációs programnyelvek Adatbáziskezelők	Egységes, hordozható operációs rendszerek és adatbáziskezelők Hálózati szoftverek 4GL programnyelvek	Világhálózati szoftverek Multimédia Mesterséges intelligencia
Jelölés:	M=2^{20}bájt	G=2^{30}bájt	T=2^{40}bájt		

3. sz. ábra
A számítógépgenerációk legfontosabb jellemzői

Meg kell jegyezni, hogy napjainkban különösen a teljesítményjellemzők vonatkozásában az egyes generációk között egyre inkább elmosódnak a határok. Ennek ellenére a nagyságrendi teljesítmény jellemzők és a felhasználás fejlődése fontos információkat tartalmaz és rámutat a fejlődés gyorsulására.

A teljesítmény/ár mutatót tekintve a számítógépek exponenciális trend alapján fejlődnek. Ennek legismertebb kifejezése a Moore-törvény, mely szerint a számítógépek teljesítménye kb. másfél évente megduplázódik.

A számítógéprendszerek fejlődésében mindig minőségi ugrás eredményez az új áramköri technológiák alkalmazása. Ezek kifejlesztése viszont jelentős kutatási ráfordítást igényel, ezért a minőségileg új áramköri technológiákra való áttérés általában éveket vesz igénybe.

Az átmeneti időszakokban sem áll le a fejlődés, a számítógéprendszerek tervezői az alapépítőelemekbeli minőségi ugrások között *rendszertechnikai megoldásokkal igyekeznek a számítógépek teljesítményét megnövelni.*

A fejlődésre néhány jellemző példa:

1. GENERÁCIÓ:	Neumann elvű gépek
2. GENERÁCIÓ:	I/O Processzorok
	Komplexebb utasításkészlet
	Verem, Memória Interleaving
3. GENERÁCIÓ:	Mikroprogram-vezérlés
	Virtuális tároló
4. GENERÁCIÓ:	Mikroprocesszorok
	RISC processzorok
	Párhuzamosítás pl. az utasításvégrehajtás szintjén
5. GENERÁCIÓ:	Számítógéphálózatok
	Erőteljes párhuzamosítás pl. szuperskalár processzorok
	SMP architektúra

1.4. A VHDL SZABVÁNYOS HARDVER LEÍRÓNYELV

A hardver az 1980-as években már rendkívül bonyolulttá vált. Ma már egy PC is 10 milliós nagyságrendű alapépítőelemből (tranzisztor, kondenzátor, flip-flop) áll. Ez a gépeket tervező ember számára áramköri és logikai szinten már gyakorlatilag áttekinthetetlenné vált.

Az ezekből fakadó gondokat a következő példák illusztrálják:

- Az ember képtelen kapuáramköri mélységben számítógépet tervezni.
- A megtervezett számítógép működésének hibátlansága milyen módszerekkel ellenőrizhető?

- Hogyan dokumentálhatjuk áttekinthetően a hardvert?
- Hogyan tud a hardverfejlesztésén együtt dolgozó nagyszámú team (esetenként több ezer fejlesztőmérnök) együttműködni?
- A hardverfejlesztő cégek és az integrált áramkörök gyártói hogyan tudnak kommunikálni? (Pl. szerződések tartalma)
- Hogyan adhatják át a tervezési információkat a tervezők a hardvergyártók CAD/CAM rendszereinek?

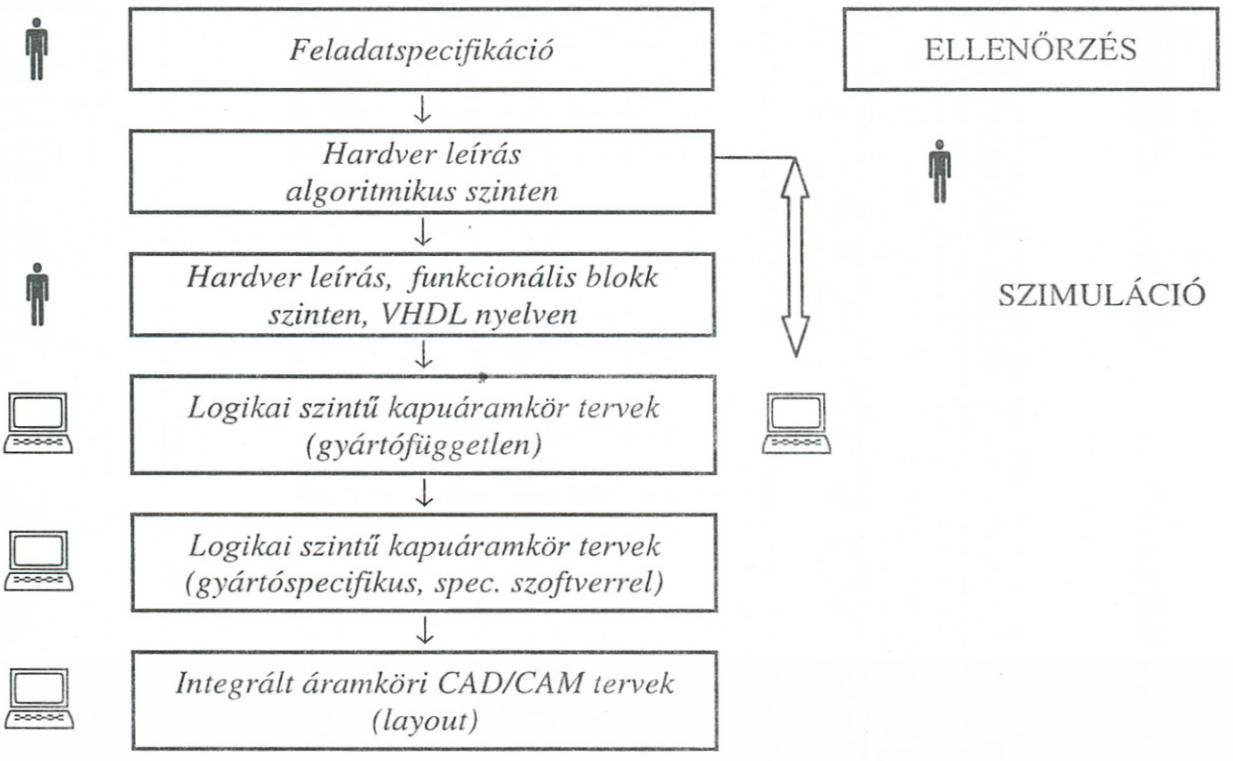
E problémákra megoldást csak egy olyan, *a rétegmodellnek megfelelően kialakított hardver leírónyelv* adhat, *mely* (a magas és gépközeli programnyelvekhez hasonlóan) *biztosítja*

- *az egyes rétegnek megfelelően egy szabványos „nyelven" a hardver leírását,*
- *a magasabb szinteken az ember számára is áttekinthetővé teszi a hardver dokumentációját,*
- *az alsó szinteken lehetővé teszi a tervek átadását a gyártás CAD/CAM rendszereinek,*
- *biztosítja a hardver szimulációs bevizsgálását.*

Az USA Védelmi Minisztériuma 1980-ban hozta létre a VHSIC (Very High Speed Integrated Circuits) nevű projektet, mely alapján bebizonyosodott, hogy a nagysebességű integrált áramkörökből felépülő eszközök tervezéséhez egy hardver leíró nyelv szabványra van szükség.

Ebből kiindulva fejlesztették ki a *VHDL (VHSIC Hardware Description Language) nevű hardver leíró nyelvet* a 80-as évek közepén, *melyet azóta nemzetközi szabványnak is elfogadtak* (IEEE-1076, 1987 és ANSI 1989).

A VHDL alkalmazásával a hardver fejlesztése a hardver rétegmodelljének megfelelően, a hierarchikus szintek szerint, felülről-lefelé (TOP-DOWN) *történik* (lásd 4. sz. ábra).



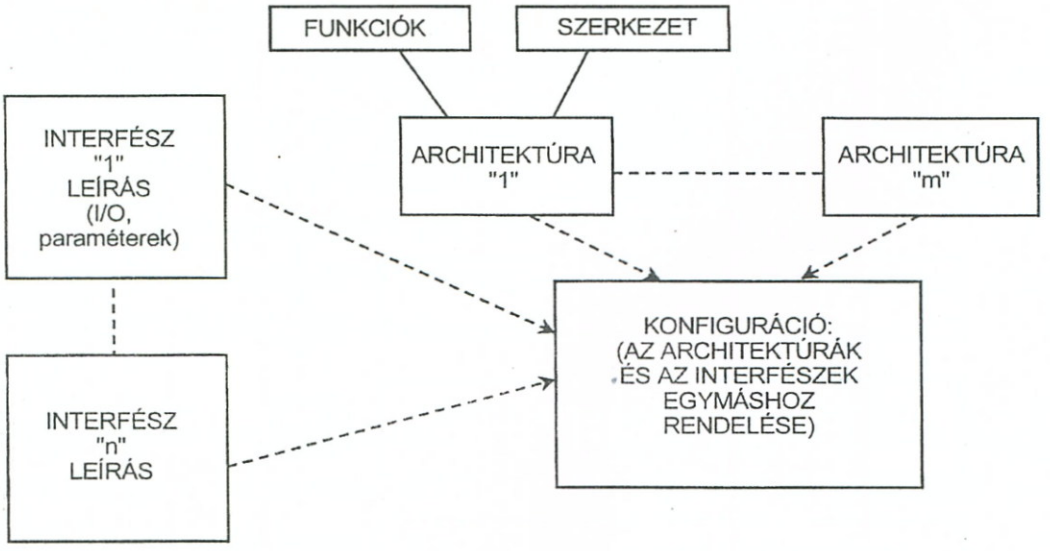
4. sz. ábra
A hardverfejlesztés lépései VHDL-el

Az ember közvetlenül csak a hierarchia felső 3 szintjén kapcsolódik be a tervezőmunkában, ezt követően a tervezési feladatokat számítógépes rendszerekkel végeztetik el. Ennek helyességét számítógépes szimulációval ellenőrzik, melyet a tervező irányít és értékel.

A VHDL hardver leíró nyelv három alapegysége:

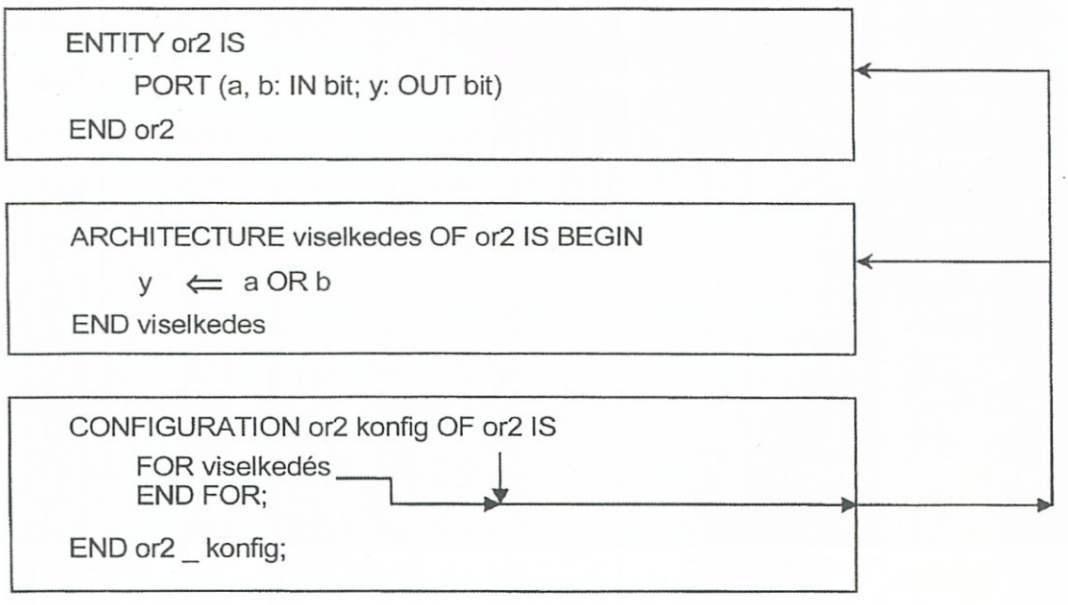
- *az architektúra,*
- *az interfész és*
- *a konfiguráció.*

Az „Architektúra" leírásokban kell megadni egy hardver egység funkcióit és szerkezetét. Az „Interfész" leírások tartalmazzák a hardver egységek közötti kommunikáció (protokoll) jellemzőit. A „Konfiguráció" leírásában kell megadni az Architektúrák és Interfészek egymáshoz rendelését (lásd 5. sz. ábra).



5. sz. ábra
A VHDL modell alapegységei és ezek kapcsolata

A VHDL nyelvű hardver leírás leginkább egy emberi nyelvhez közel álló, magas szintű programnyelvhez hasonlítható. Erre mutat példát a 6. sz. ábra, mely egy „VAGY" kapu leírását tartalmazza VHDL nyelven.



6. sz. ábra
Egy „VAGY" kapu leírása VHDL nyelven

A VHDL szabványos hardver leírónyelvet felhasználva működnek a különböző *hardver tervező programcsomagok*. Ezek *funkcionális alapegységei* a következők:

Project manager !

A tervező munka szervezését, irányítását és részben végzését segítő programokat tartalmazza. Ilyenek például:

- VHDL editor
- Séma editor

Design manager !

A tervezést segítő programokat tartalmazza. Például:

- VHDL compiler
- Idő elemző
- Kapcsolási rajz előállító

Szimulátor, hardver hibakereső !

A megtervezett hardver eszközök szimulációival történő bevizsgálását biztosító programokat tartalmazza.

Építőelem könyvtárak (Resource Libraries) !

A gyakran előforduló hardver elemek standard terveit tartalmazza.

? ELLENŐRZŐ KÉRDÉSEK

1. Határozza meg a hardver fogalmát!
2. Miért van szükség a hardver rétegmodelljére?
3. Sorolja fel a hardver rétegmodelljének szintjeit!
4. Mondjon legalább két példát a rendszerszint hardver részegységeire!
5. Mivel jellemezzük a hardvert a rétegmodell algoritmikus szintjén?
6. A hardver melyik szintjén használjuk fel a modellezés eszközéül a Boole-algebrát?
7. A rétegmodell melyik szintjén kezelünk folytonos fizikai mennyiségeket?
8. Az Y diagram milyen három nézőpontból szemlélteti a hardver rétegmodelljét?
9. Mit nevezünk számítógépgenerációnak?
10. Ismertesse az egyes számítógépgenerációk jellemző hardver építőelemeit!
11. Az áramköri fejlesztés mellett milyen eszközökkel növelték a hardver teljesítményét?
12. Miért van szükség szabványos hardver leírónyelvre?
13. Mi a VHDL?
14. Milyen lépései vannak a VHDL alapú hardverfejlesztésnek?
15. A hardverfejlesztés milyen lépéseiben vesz részt közvetlenül a tervező ember?
16. Milyen alapelemei vannak a VHDL nyelvnek és ezekben mit kell megadni?
17. Milyen részekből épül fel egy hardver tervező szoftver?

II.

SZÁMÍTÓGÉPRENDSZER ARCHITEKTÚRÁK

2.1. A SZÁMÍTÓGÉPRENDSZER ARCHITEKTÚRA FOGALMA

Számítógéprendszer architektúrának a számítógép funkcionális felépítését, a részegységek kommunikációs kapcsolatait, valamint a rendszer specifikációjának együttesét nevezzük.

A meghatározásból következik, hogy *egy számítógép architektúrát a részegységek és funkcióik, valamint ezek kapcsolatát meghatározó interfész-protokollok együttes leírásával jellemezhetünk.*

Korszerű értelmezés szerint az architektúra fogalma három szempontot fog át:

- a számítási modellt,
- az értelmezés szintjét (a hardver hierarchikus leírásának megfelelően) és
- a leírás irányultságát (logikai és fizikai architektúra, mely körül az első alatt a funkcionális specifikációt, utóbbi alatt pedig a számítógép konkrét megvalósítását értjük).

Ezek a definíciók az architektúra fogalmát egy magasabb absztrakciós szintre terjesztik ki, mely az operációs rendszert is magába foglalja.

A számítógéprendszer architektúrák fejlesztésének meghatározó oka a számítógépek teljesítményének növelése. Ezért az egyes jellegzetes számítógép architektúrák bemutatása előtt röviden megvizsgáljuk azokat a tényezőket, amelyektől a számítógépek teljesítménye döntően függ. Ezek megismerése egyúttal ahhoz is hozzásegít, hogy az architektúrafejlesztések fontosabb irányait is mélyebben megérthessük.

2.2. A SZÁMÍTÓGÉP TELJESÍTMÉNYÉT MEGHATÁROZÓ TÉNYEZŐK

A számítógépekben található chipek ciklusokban működnek, valamilyen frekvenciával. Erre azért van szükség, mert a feszültség szintek csak bizonyos idő után állnak be a bináris 0-nak vagy 1-nek megfelelő értékre. A ciklikus működés az órajel vagy meghajtó frekvencia használatával érhető el. (Ha például az órajel 400 MHz, azaz 400 millió órajel/másodperc, akkor másodpercenként maximum ennyi alkalommal van lehetőség bináris jelek értelmezésére.)

A számítógéprendszerek teljesítményét nyilván egy adott feladat elvégzéséhez szükséges idővel hasonlíthatjuk össze (ha egy adott feladat végrehajtásához kevesebb időre van szüksége egy számítógéprendszernek, akkor ennek a rendszernek a teljesítménye nyilván nagyobb).

Ha egy feladat végrehajtási idejét „F"-el jelöljük, akkor

- $F = C \cdot T \cdot U$, *ahol* C = *az egy utasításra eső átlagos ciklusszám*
  T = *az egy ciklushoz szükséges idő*
  U = *a feladatvégrehajtáshoz szükséges utasítás szám.*

Ezt figyelembe véve *egy számítógéprendszer teljesítményét a következő módon növelhetjük:*

- *U csökkentése* ⇒ *Hatékony programozás és fordítóprogram*
- *T csökkentése* ⇒ *Magasabb órajel frekvencia, ez áramköri fejlesztést jelent*
- *C csökkentése* ⇒ *Párhuzamosítás az utasításvégrehajtásban, azaz az architektúra fejlesztése*

2.3. A SZÁMÍTÓGÉPEK TELJESÍTMÉNYÉNEK MÉRÉSE

A számítógépek teljesítményének legegyszerűbb mértékegysége az időegység alatt végrehajtott utasítások átlagos száma, azaz a millió művelet/secundum:

- *MIPS, azaz* *Million Instructions per Second*

(Ezt egyes esetekben MOPS = Million Operation per Second mutatóval jelölik, melynek értelmezése azonos a MIPS-el.)

- *MFLOPS, azaz* *Millions of Floating Point Operations per Second,* ami a másodpercenként végrehajtott lebegőpontos utasítások számát jelenti

A számítógéprendszerek teljesítményéről összetettebb képet adnak a speciális teljesítménymérő programokkal előállított mutatók a „Benchmark"-ok.

Ezek közül néhány ismertebb:

- Whetstone ⇒ mérnöki, tudományos programokat reprezentál
- Dhrystone ⇒ rendszerprogramozási környezetet reprezentál
- SPEC Benchmark ⇒ alkalmazási cél függő tesztprogramkészlet (1992, 1995), mely egy elméleti alapgéphez viszonyít (pl. játékprogram, grafika, kvantummechanika)

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

Napjainkban már – ha például számítógépet vásárolunk – a számítógépek teljesítményét csak a felhasználás céljától függően szabad értékelni.

Példa: INTEL ICOMP INDEX Versio: 2.0 képzési szabálya

- hagyományos üzleti alkalmazás: 40%
- nagy teljesítményigényű végfelhasználás 15%
- fixpontos alkalmazás: 20%
- lebegőpontos alkalmazás: 5%
- multimédia és 3D grafika: 20%

Azaz az INTEL cég a teljesítményindexet a processzoreladások piaci szegmenseinek arányában súlyozza. Ezért az ICOMP index nyilvánvalóan nem használható egy konkrét számítógépfelhasználással összefüggő beruházási döntéseknél.

2.4. A SZÁMÍTÓGÉPRENDSZEREK OSZTÁLYOZÁSA

A számítógéprendszereket különböző szempontok szerint csoportosíthatjuk. Ezek közül a legfontosabbak:

- *Teljesítmény szerint: Mikro (kis), közép és nagyszámítógépek*
- *Az utasítás és adatfolyamatok száma szerint*
- *Az utasításkészlet szerint: Komplex és egyszerűsített utasításkészletű gépek* (CISC és RISC)
- *A számítógép működési elve szerint: Neumann és nem Neumann elvű architektúrák*
- *Az egyidőben kiszolgált felhasználók száma és a kiszolgálás időbelisége szerint*

2.4.1. A számítógépek osztályozása teljesítményük szerint

A teljesítmény szerinti osztályozásnál általában nagy, közép és mikroszámítógépeket szoktak megkülönböztetni.

A nagyszámítógépek (MAINFRAME vagy hálózatokban HOST) jellemzői:

- *nagy műveleti sebesség,*
- *nagy tárolókapacitás,*
- *speciális működési feltételek* (kondicionálás, pormentesség stb.),
- *nagy megbízhatóságú működés* (pl. 24 órás folyamatos, leállás nélküli üzem).

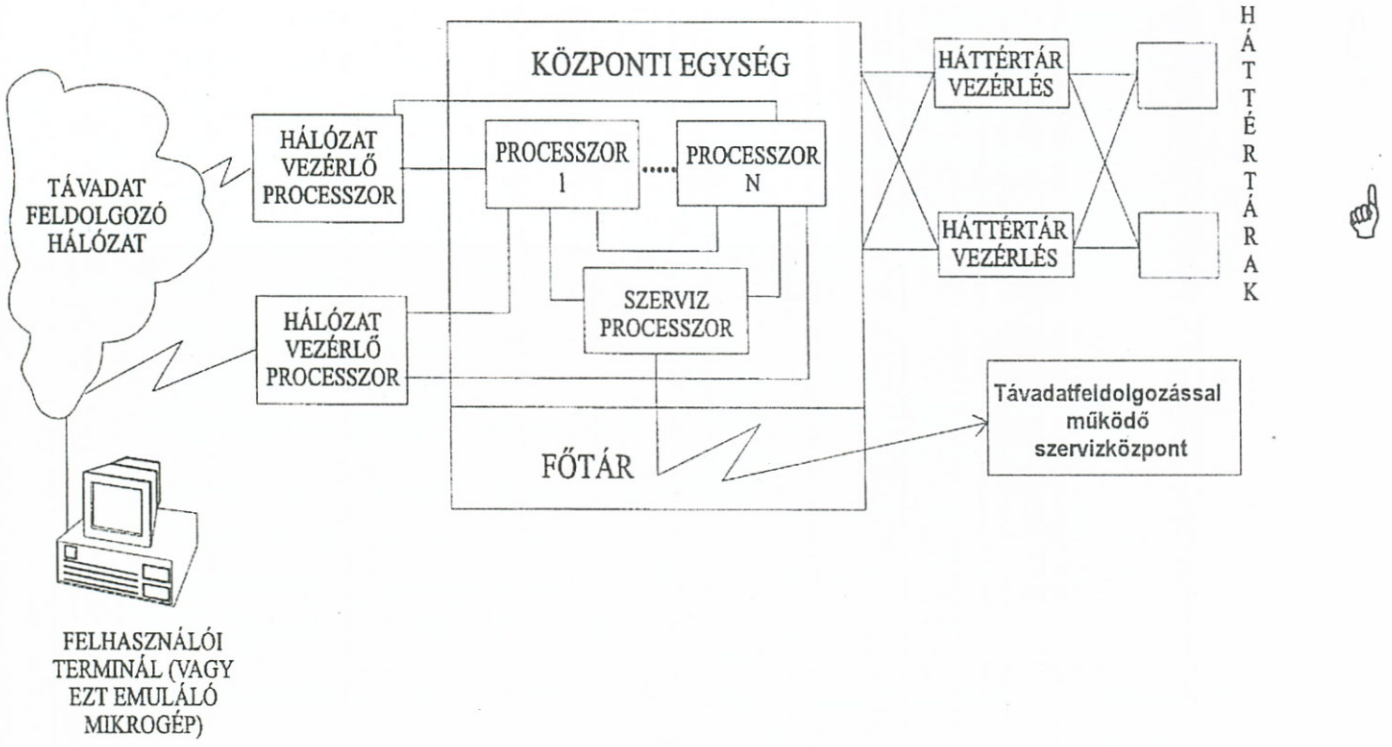
Felhasználásukra példák:

- nagy szervezetek, nagy megbízhatóságú rendszerei (katonaság, rendőrség, bankok),
- nagy tömegű adatot feldolgozó adatbázisszerverek,
- média szerverek,
- bonyolult jelenségek szimulációja (például időjáráselőrejelzés).

Jellemző adatok:

- Processzorszám: 8 – 1024 db
- Memóriaméret: 4000 – 256.000 Mbájt
- Háttértárkapacitás: 10 millió – 50 millió Mbájt

A számítógéprendszerek legtöbbször hibatűrő (fault tolerant) architektúrával kerülnek kiépítésre. Erre mutat példát a 7. sz. ábra.



7. sz. ábra
Hibatűrő számítógéprendszer architektúrájának blokkvázlata

A hibatűrő architektúrák néhány fontosabb *jellemzője:*

- *a paritásellenőrzés mellett (helyett) hibajavító kódok alkalmazása,*
- *rekonfigurációs képesség, azaz a meghibásodott hardveregységeket a rendszer automatikusan, a folyamatos működés leállítása nélkül felismeri és izolálja és ezek nélkül működik tovább,*
- *többszörözött hardver egységek,*
- *automatikus tömörítés és titkosítás a háttértárolókon.*

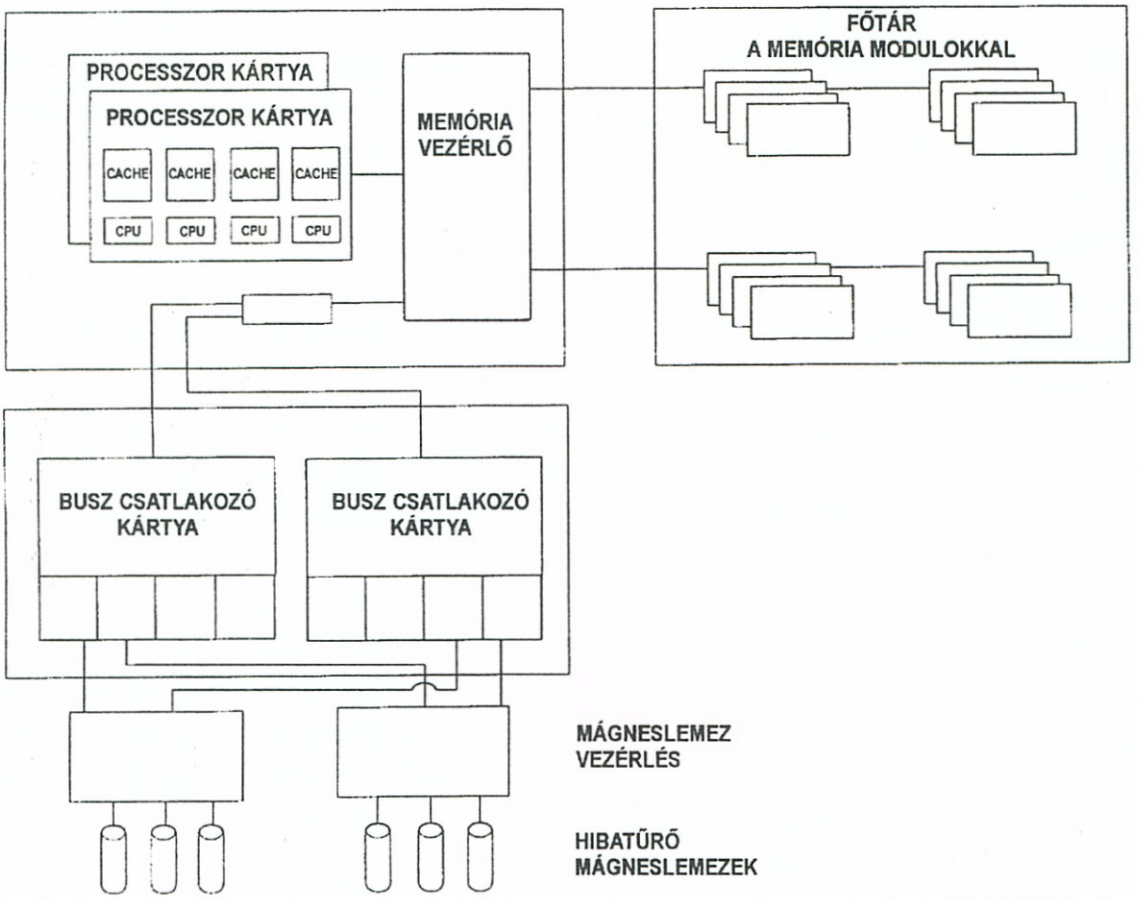
A középes teljesítményű számítógépek (MINICOMPUTER, WORKSTATION) átmenetet képeznek a nagy és a mikroszámítógépek között.

Jellemző a kategóriára, hogy – bár az alkalmazott megoldások nem olyan hatékonyak, mint a nagyszámítógépeknél – *az adatbiztonságra* (védett INTERNET szolgáltatások, biztonságos titkosítási és jogosultsági rendszer stb.) *általában komoly figyelmet fordítanak a gyártók. Jellemző operációs rendszereik a UNIX különböző változatai, a WINDOWS-NT, OS (IBM), VMS (DEC).* Fizikai méretük a kisebb íróasztal nagyságtól kb. egy szekrény nagyságáig terjed.

Jellemző adatok:

- Processzorszám: 2–12
- Memóriaméret: 128–12 000 Mbájt
- Maximális háttértár kiépítés: 4000–1 millió Mbájt

A miniszámítógépek jellemzőit az IBM AS/400 típusú számítógépének példáján keresztül mutatjuk be (lásd 8. sz. ábra).



8. sz. ábra
Az IBM AS/400 elvi architektúrális felépítése

Az AS/400 néhány fontosabb tulajdonsága:

- SMP (Simmetric Multi Processor) architektúra, azaz szimmetrikus multiprocesszoros felépítés, ami azonos típusú processzorokból felépülő gépeket jelent.
- Párhuzamos elérési utak a processzorok, a memória és a háttértárak között.
- Moduláris felépítés, ikeregységek.
- Skálázhatóság (választhatóan 2–12 processzoros kiépítés).

A kisszámítógépek (Microcomputer vagy PC = Personal Computer) kategóriába tartozó gépek teljesítménye a másik két osztályba tartozó számítógépekénél lényegesen kisebb. *A mikrogépek processzora egyetlen áramköri egységet (chip) alkot. Ezeket a gépeket általában hálózatok munkaállomásaként vagy önállóan használják. Egyes nagyobb teljesítményű mikroszámítógépek működhetnek kisebb helyi hálózatok kiszolgáló (szerver) gépeként is.*

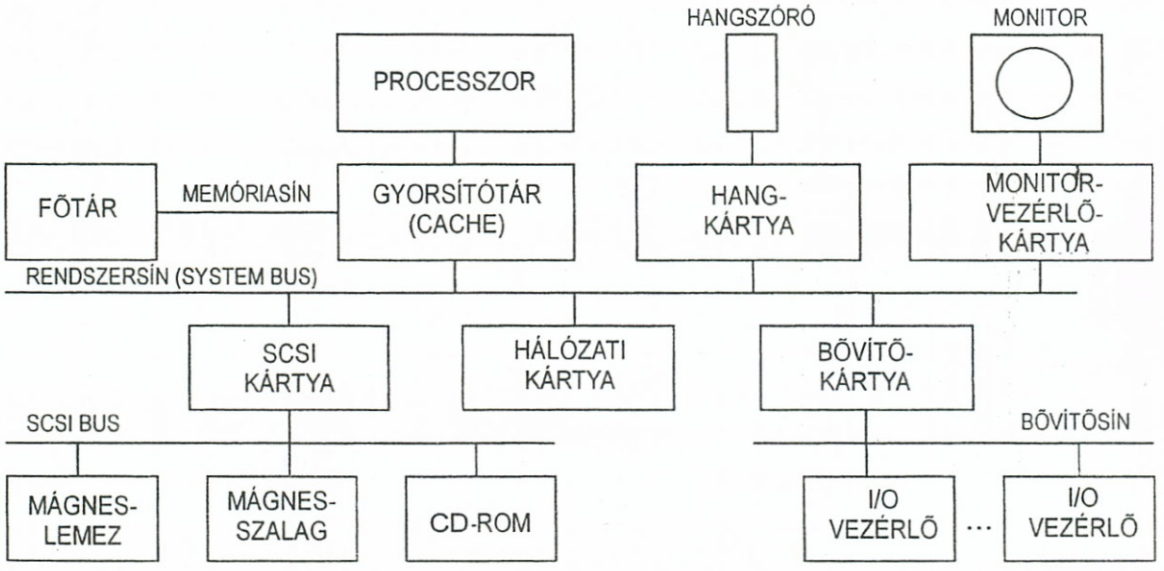
Jellemző adatok:

- Processzorszám: 1
- Főtár memóriaméret: 4–128 Mbájt
- Maximális háttértár kapacitás: 1000–20 000 Mbájt

Fizikai megvalósításuk a következő lehet:

- Torony és asztali kivitel
- Hordozhatók: Laptop, Notebook
- Speciálisak: pl. PEN-PC, NETPC

A mikroszámítógépek jellemző elvi felépítését a 9. sz. ábra szemlélteti:



9. sz. ábra
A mikroszámítógépek (PC) elvi felépítése

A mikroszámítógépek speciális változata a NETPC.

A hálózatok elmúlt években történt igen gyors fejlődése motiválta a hálózati mikroszámítógép (például az IBM Networkstation-ja) létrehozását. *Ennek alapgondolata, hogy helyileg nincs szükség teljes körű PC funkciók-*

ra, mert ezeket hálózati szolgáltatásokkal lehet helyettesíteni. (Például azt a programot, melyet a helyi mikroszámítógépen futtatunk, le lehet futtatni a hálózati szerveren is.) *Így a hálózati mikroszámítógép hardver, szoftver architektúráját jelentősen le lehet egyszerűsíteni, és ezáltal áruk a szokásos mikroszámítógépekénél lényegesen olcsóbb lehet.*

Ugyanakkor mindenképpen meg kell említeni, hogy *a nagy átbocsátóképességű hálózat kiépítésének és működtetésének költségei is befolyásolják a NETPC-k felhasználását.* Ezért elterjedésük lényegesen lassúbb, mint azt korábban megjósolták.

2.4.2. A számítógépek csoportosítása a feldolgozott utasítás és adatfolyamatok száma szerint (FLYNN 1966)

Az utasítás és adatfolyam definíciója:

- ***Utasításfolyam**: Az utasítások egymásutáni sorozata, amelyeket egy program lefuttatása során végre hajt 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.)

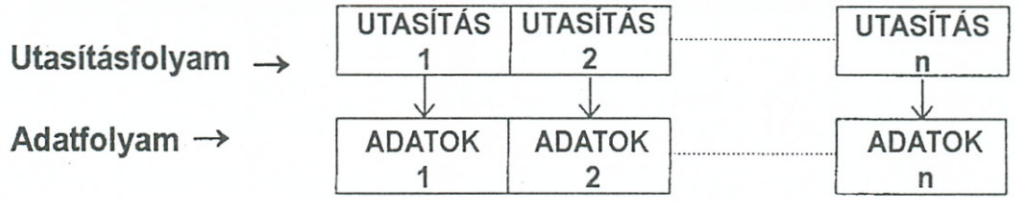
- ***Adatfolyam:** Az utasításfolyam utasításai mindig meghatározott adatokra hivatkoznak, amelyekkel a műveleteket el kell végezni. Ezek egymásutániságát, amilyen sorrendben rendelkezésre kell állniuk, nevezzük adatfolyamnak.*

(E két fogalomról szemléletes képet alkothatunk, ha elképzeljük, hogy a program futásakor sorban felírjuk a végrehajtott utasításokat és az ezekben szereplő adatokat.)

Ez az osztályozás azért hasznos, mert segítségével könnyebb megérteni a bonyolultabb számítógéparchitektúrák működésének lényegét.

SISD architektúrájú számítógépek

A SISD (Single Instruction Stream Single Data Stream) típusú számítógép egyetlen utasításfolyammal egyetlen adatfolyamot dolgoz fel (lásd 10. sz. ábra).



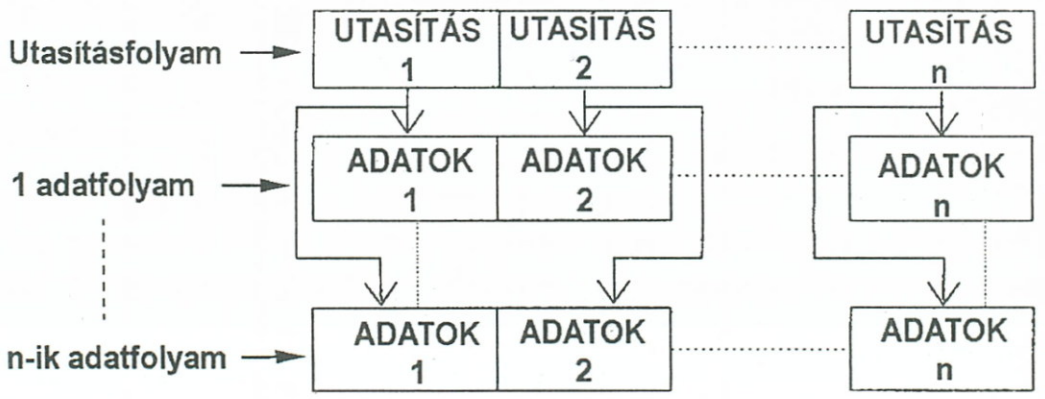
10. sz. ábra
A SISD számítógép működése

SISD architektúrájú gépekre példa:
- A Neumann elvű gépek
- A PC-k CPU-ja a Pentium MMX-ig

Fontos megjegyezni, hogy vannak olyan SISD gépek, melyek nem a (az összes) Neumann elvek szerint működnek. PÉLDA: Harvard architektúrájú gépek, melyeknél az utasítástároló fizikailag elkülönül az adattárolótól (például Pentiumtól kezdve az L1 cache tároló).

SIMD architektúrájú számítógépek

A SIMD (Single Instruction Stream Multiple Data Stream) típusú számítógép egyetlen utasításfolyammal többszörös adatfolyamot dolgoz fel (lásd 11. sz. ábra).

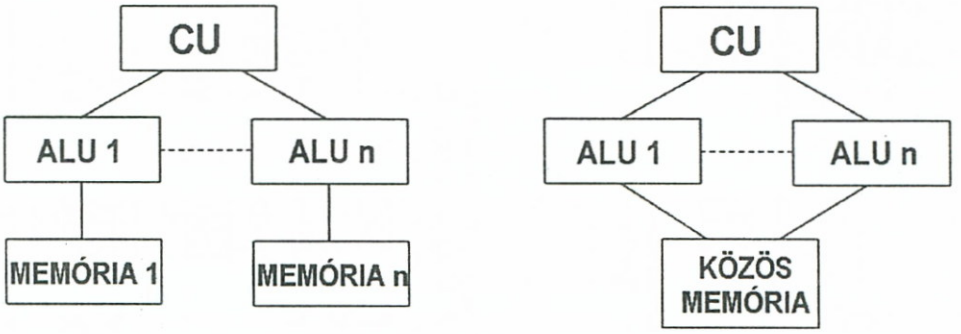


11. sz. ábra
A SIMD számítógép működése

Ezek a számítógépek több párhuzamos, egyidejű működésre képes műveletvégző egységet (ALU) tartalmaznak. Vektorműveleteket képesek végrehajtani, gépi utasításszinten (például: Pentium III, 3D grafikus utasítások).

Két alaptípusuk van:

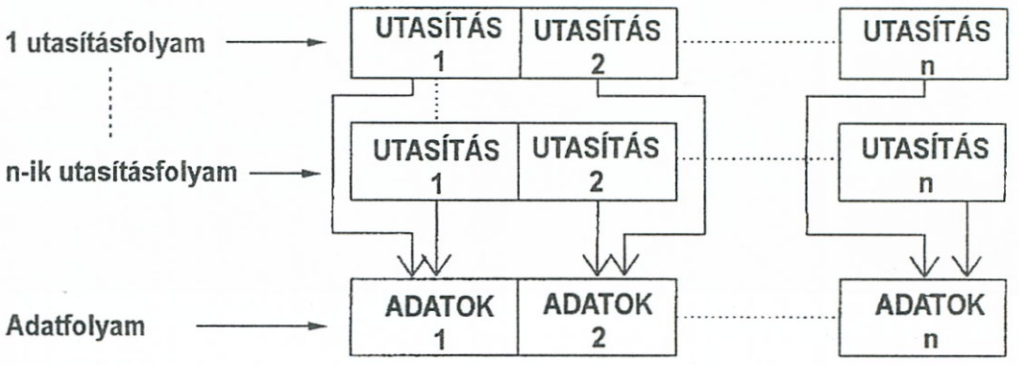
- a közös (Shared Memory) tárolójú gépek, és
- az osztott (Distributed Memory) tárolóval rendelkező gépek (lásd 12. sz. ábra).



12. sz. ábra
Osztott és közös memóriájú SIMD számítógép

MISD archtektúrájú számítógépek

A MISD (Multiple Instruction Stream Single Data Stream) típusú számítógép több utasításfolyammal egyetlen adatfolyamot dolgoz fel (lásd 13. sz. ábra).

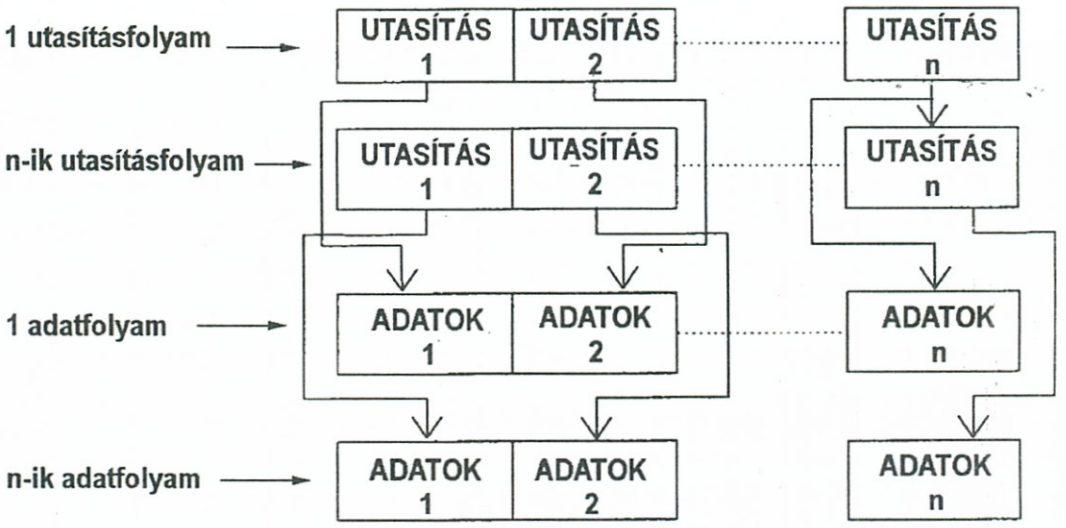


13. sz. ábra
A MISD számítógép működése

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.

MIMD architektúrájú számítógépek

A MIMD (Multiple Instruction Stream Multiple Data Stream) számítógéprendszerek több utasításfolyammal több adatfolyamatot dolgoznak fel (lásd 14. sz. ábra).



14. sz. ábra
A MIMD számítógép működése

Ebbe a kategóriába tartoznak a multiprocesszoros számítógépek (több vezérlőegységgel) és a nem a hagyományos, soros utasításvégrehajtás elvén működő számítógépek (adatvezérelt számítógépek, neuronális hálók).

A MIMD gépek memóriakezelése szintén lehet közös, vagy elosztott.

2.4.3. 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) 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) végzi,*
- *az adatok és programok beolvasására és megjelenítésére önálló egységek (perifériák) szolgálnak.*

A Neumann elvű architektúrák lényegi továbbfejlesztését a teljesítménynövelés piaci igénye kényszeríttette ki.

A teljesítménynövelésnek alapvetően két módszere van:

a) nem strukturális gyorsítás
- *Órajel növelése*
- *A program optimalizált fordítása* (Ennek különösen a RISC processzoroknál van jelentősége).

b) Strukturális gyorsítás
- *Párhuzamosítás a CPU-n belül*
  - → *vektorszámítógépek*
  - → *pipeline feldolgozás*
  - → *szuperskalár processzorok*
  - → *társprocesszorok*
  - → *multiprocesszoros architektúrák* (több „hagyományos" CPU-val és ALU-val)
- *Nem hagyományos elven működő számítógépek*
  - → neuronális hálók

2.4.4. CISC és RISC architektúrájú processzorok

Kezdetben a számítógépeket gépi kódban programozták, majd az Assembly nyelv vált elterjedt. A szoftverfejlesztők egyre „komfortosabb" utasításokat igényeltek (például változó hosszúságú adatmezők mozgatása a főtáron belül), ezért állandóan bővítették a számítógépek utasításkészletét. Ennek eredményeként:

- egyre több és bonyolultabb utasítást tartalmazott az utasításkészlet,
- ezek hardver megvalósítását mikroprogramvezérléssel kellett megoldani, azaz egy gépi utasítás végrehajtását több elemi lépésre felbontották és az ezekre vonatkozó adatokat a számítógép csak olvasható memóriájában (ROM = Read Only Memory) tárolták (lásd 3.3.3. fejezet).

Így alakultak ki a komplex utasításkészletű számítógépek (Complex Instruction Set Computer = CISC). Ezeket

- bonyolult mikroprogramok, és
- a „gép a gépben" jellegű működés (a mikroprogramvezérlő felfogható egy miniatűr Neumann elvű processzornak)

jellemezte.

A mikroprogramvezérelt utasításvégrehajtás komoly korlátává vált a számítógépteljesítmény növelésének. Ezért olyan architektúrát terveztek, melynél

- csak a gyakori, „egyszerű" utasítások szerepeltek az utasításkészletben,
- lehetővé vált mikroprogramozás kiküszöbölése, azaz magas szintű programnyelvről fordítás lényegében a korábbi mikroutasítások szintjére történt.

Így jöttek létre a RISC processzorok, azaz redukált utasításkészletű számítógépek (Reduced Instruction Set Computer).

A CISC processzorok tehát nagy számú utasítást tartalmazó utasításkészlettel rendelkeznek, az utasítások szerkezete bonyolult, többfajta memóriacímzési módot tesznek lehetővé, az utasításvégrehajtás mikroprogramvezérelt.

A CISC processzorokra példák a mikroszámítógépek köréből a Pentium és a vele kompatibilis processzorcsalád.

A RISC processzorok utasításkészlete csökkentett, egyszerűsített, a memóriahozzáférés csak két utasítással: memóriából való adatbetöltés (LOAD) és memóriába való adatírás (STORE) történhet, a műveleti vezérlés huzalozott (azaz beépített áramkörökkel végrehajtott) *vagy horizontális* (egyszerűsített) *mikroprogramvezérelt.*

A RISC processzorokra példa: SPARC, MIPS, POWER PC különböző változatai. (Ezeket hétköznapi szóhasználattal sokszor UNIX gépeknek is nevezik.)

A CISC és RISC processzorok összehasonlító jellemzőit a 15. sz. ábra foglalja össze.

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.*
Tárolóvédelem hardver úton.	*Tárolóvédelem szoftver segítségével.*

15. sz. ábra
RISC és CISC processzorok jellemzői

Napjainkban mindkét processzortípusnál ugyanazokat a teljesítménynövelő módszereket alkalmazzák, és így a különböző architektúrák közelednek egymáshoz.

Erre példák:
- a szuperskalár CISC processzorok
- a Pentium Pro, II, III. (RISC mag a CISC processzorban).

2.4.5. Vektorszámítógépek

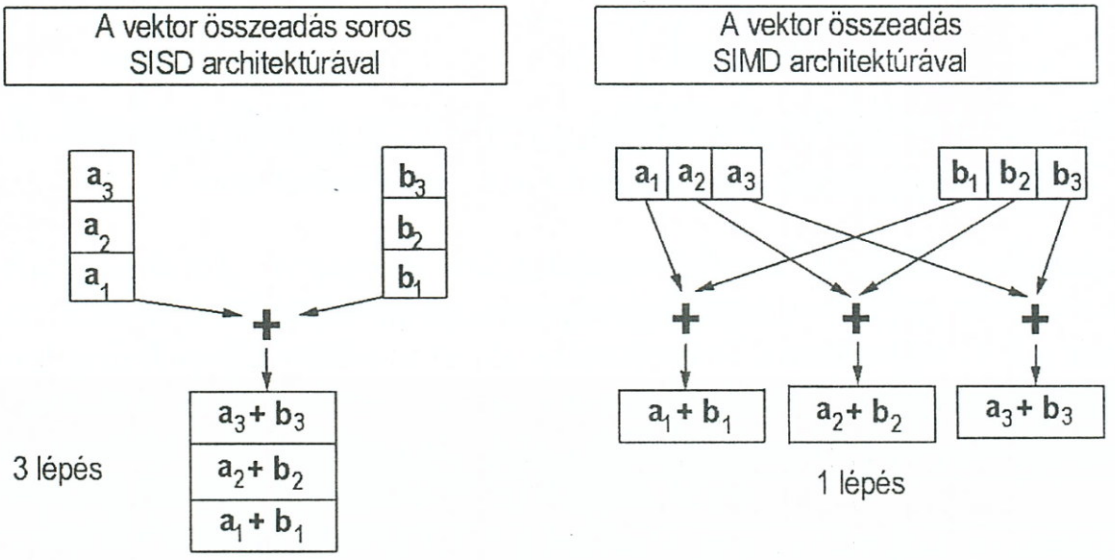
A számítógépek alkalmazási területeinek egy részénél (tudományos-műszaki számítások, modellezés, 3D grafika stb.) *gyakran kell vektorokkal és mátrixokkal műveleteket végrehajtani. Ezt hatékonyan elvégezni korábbi osztályozásunk szerint SIMD* (egyszeres utasításfolyam, többszörös adatfolyam) *architektúrájú gépekkel lehet, melyeket vektorszámítógépeknek hívunk.*

Példaként tekintsünk egy 3 dimenziós vektor összeadását.

$\vec{a} = (a_1, a_2, a_3)$

$\vec{b} = (b_1, b_2, b_3)$

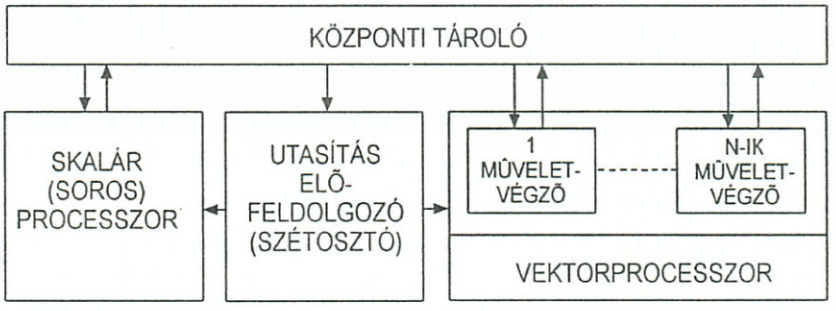
$\vec{a} + \vec{b} = (a_1+b_1, a_2+b_2, a_3+b_3)$



16. sz. ábra
A soros és a vektoros összeadás végrehajtása

Látható, hogy *soros utasításvégrehajtásnál az egy ALU miatt az összeg vektor csak 3 lépésben számítható ki, a SIMD architektúrájú vektorszámítógép viszont a vektor összeadását 1 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 a következő:

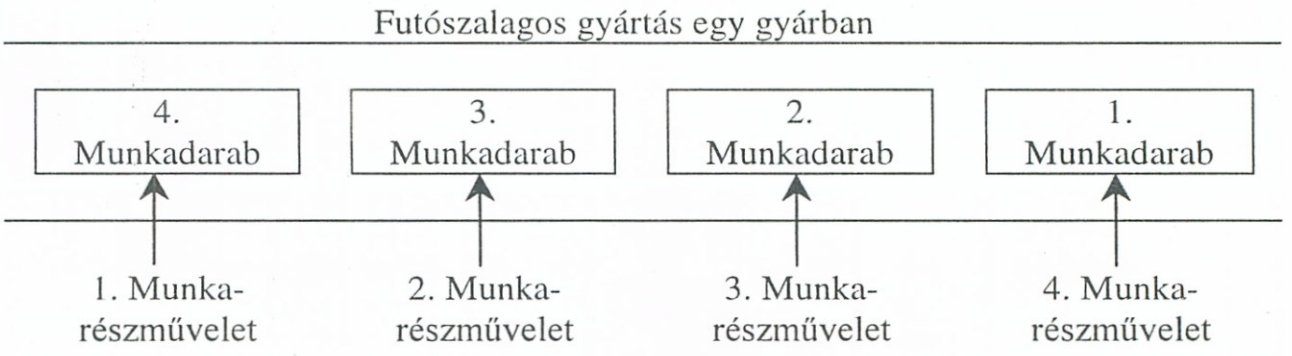


17. sz. ábra
A vektorszámítógépek elvi felépítése

Láthatjuk, hogy a több műveletvégző egységet tartalmazó vektorprocesszor mellett a gépben egy skalárprocesszor is megtalálható, mely a nem vektoros műveleteket végzi. (Ez a skaláris műveleteket a vektorprocesszornál hatékonyabban dolgozza fel.) A vektorszámítógépes legismertebb képviselői a CRAY gépcsalád tagjai, de az MMX mikroprocesszorok is a vektorszámítógépek működési elvének megfelelően hajtják végre a multimédiás gépi utasításokat.

2.4.6. Az utasításvégrehajtás gyorsítása pipelining-gel, ILP processzorok

A pipelining átlapolt utasításvégrehajtást magyarra fordítva csővonal, adatcsatornás vagy futószalag feldolgozásnak nevezhetjük (részletes kifejtését lásd a 3.4. fejezetben). A futószalag jól szemlélteti a pipelining lényegét (lásd 18. sz. ábra):



18. sz. ábra
A futószalagos gyártásnál az egyes részműveleteteket időben párhuzamosan hajtják végre

A gépi utasítások elemi lépései (előkészítés, dekódolás, operandusok címszámítása stb.) különböző hardver erőforrásokat igényelnek. Ezért, ha egy utasításban egy elemi lépést végrehajtottunk (például első lépésben már kiolvastuk az utasítást a memóriából) és az ehhez szükséges hardver egység felszabadul, akkor ezt igénybe vehetjük egy következő utasítás elemi lépésének végrehajtására.

Példa a pipeline szervezésű utasításvégrehajtásra:

Tegyük fel az egyszerűség kedvéért, hogy a CPU egy utasítást három fázisban hajt végre:

- utasítás kiolvasás (fetch: F),
- dekódolás (decode: D),
- végrehajtás (execute: E).

Az egyes fázisok egymással egyidőben, párhuzamosan hajthatók végre, az első utasítás E fázisa alatt a második utasítás D fázisa és a harmadik utasítás F fázisa (19. sz. ábra).

Utasítás/Fázis	i-2	i-1	i	i+1	i+2
1.	F1	D1	E1		
2.		F2	D2	E2	
3.			F3	D3	E3

F = utasításkiolvasás D = dekódolás E = végrehajtás

19. sz. ábra
Pipelining utasításvégrehajtás

A pipeline alapgondolata tehát párhuzamosítás, annak kihasználása, hogy az utasítás feldolgozás különböző fázisait autonóm és párhuzamosan működtethető hardver alrendszerek hajthatják végre.

A pipeline utasításfeldolgozást alkalmazó processzorokat utasításszinten párhuzamos működésű (Intruction Level Parallel), vagy ILP processzoroknak nevezzük.

2.4.7. Szuperskalár processzorok, társprocesszoros számítógépek

A több végrehajtó egység párhuzamos működésével történő gyorsításnak két alapmegoldása van: a műveletvégző egységek számának növelése a processzszoron belül és kívül.

Párhuzamosítás a CPU-n belül

A processzoron belül nemcsak úgy tudunk párhuzamosítani, hogy a gépi utasításokon belüli elemi lépéseket végrehajtó hardver egységek átlapolva működnek, hanem a *végrehajtó hardver egységeket is lehet még többszörözni.*

Erre jó példa a Pentium processzor, mely két fixpontos, és egy lebegőpontos műveletvégrehajtó egységet tartalmaz.

Ezeket a processzorokat szuperskalár processzoroknak nevezzük, ha egy gépi ciklus alatt esetenként több utasítást is képesek végrehajtani.

Ennek természetesen feltétele, hogy a processzor a műveletvégző egységekhez vezető, párhuzamos működésre képes belső buszokkal rendelkezzen.

Párhuzamosítás a CPU-n kívül társprocesszorokkal

A számítógéprendszer összteljesítményét azzal is növelni tudjuk, ha rögzített, *konkrét feladatokra, önálló működésre alkalmas processzorokkal bővítjük az architektúrát a processzor tehermentesítése érdekében. Ezek a társprocesszorok utasítások, parancsok küldésével „felprogramozhatók", ezután teljes önállósággal vezérlik az egyes részfeladatok végrehajtását.*

Társprocesszorok alkalmazására mutatnak példát a következők:

- *Mainframeknél az I/O Processzor* (Blokkmultiplexer és Multiplexer csatorna) *alkalmazásával a lassú I/O rendszeregységek leválaszthatók a processzorról. Ezek az I/O műveletek vezérlését, önállóan a processzortól függetlenül képesek elvégezni.*
- *Mikroszámítógépeknél*
  - *a műveletvégző társprocesszor, mely önállóan hajtja végre például a lebegőpontos műveleteket* (i80486-os processzortól kezdve a processzorlapkára integrált),
  - *a 3D gyorsító társprocesszor a monitorvezérlőkártyában, mely a vektorgrafikus adatokat konvertálja önállóan raszteres, azaz a képernyőn megjeleníthető adatokká,*
  - *hangfeldolgozó társprocesszor a hangkártyában.*

2.4.8. Multiprocesszoros architektúrák, szuperszámítógépek

A számítógépek teljesítményének további növelése *és a hibatűrő arhitektúrák iránti igény* (azaz például a szervergépek 24 órás folyamatos, leállásmentes üzemeltetésének követelménye) valamint a *skálázható számítógéprendszerek előnyei (azaz nagyobb teljesítményigény esetén a számítógép kiegészítésekkel továbbépíthető legyen az eredeti rendszer lecserélése nélkül)* az elmúlt években a multiprocesszoros számítógépek fejlődését is felgyorsította.

A multiprocesszoros számítógépek lényege, hogy több vezérlésre és műveletvégrehajtásra önállóan alkalmas processzorral működnek. Abban különböznek a vektorszámítógépektől, hogy *az egyes processzorok nem szükségképpen azonos utasításokat hajthatnak* végre, azaz a korábban megadott osztályozás szerint *a MIMD kategóriájú* (több utasításfolyam, több adatfolyam) *számítógépek* közé sorolhatók.

Ahhoz, hogy érzékeltessük a több processzoros architektúrákkal kapcsolatos rendszertervezési, operációs rendszer működési és programozási problémákat, érdemes visszagondolni az űrhajózás első multiprocesszoros fedélzeti számítógépére, mely a MARINER-4 amerikai űrszondába volt beépítve. A szonda számítógépe lényegében három funkcionális feladatra elkülönült processzort tartalmazott, melyeknek az űrrepülés „átlagos" időszakában a következő feladataik voltak:

- az első processzor a csillagászati megfigyeléseket végezte,
- a második a Földdel kapcsolatos kommunikációt vezérelte,
- a harmadik feladata a repülési pálya számítása és a hajtóművek vezérlése volt.

Ezek – amíg a szonda nem közelítette meg a Mars légkörét – egymástól függetlenül dolgoztak.

A repülésnek azonban volt egy kritikus időszaka, amikor a szonda megközelítette a Mars légkörét.

A probléma lényege abban fogalmazható meg, hogy amennyiben a szonda nem a megfelelő szögben közelíti meg a Mars légkörét, vagy visszapattan (mint a víz felszínére nagy sebességgel dobott kavics), vagy elég.

Ezért ebben a kritikus időszakban az űrszonda fedélzeti számítógépeinek processzorai összekapcsolódtak:

- az 1. és 2. processzor a pályaadatok alapján a kritikus légkörbelépési szögeket számolta és az ez alapján szükségessé váló pályakorrekciókat állapította meg,
- a 3. processzor ellenőrizte (összehasonlította) az 1. és 2. processzor működésének hibátlanságát.

Ebből a példából két tanulságot érdemes levonni:

- vannak olyan feladatok, melyek szükségszerűen megkövetelik a multiprocesszoros architektúra kiépítését,
- meglehetősen bonyolult feladat egy multiprocesszoros hardver és szoftver megtervezése, programozása és működtetése.

A multiprocesszoros architektúrájú gépek megalkotásakor a tervezőknek a következő három lényeges kérdésre kell megtalálniuk a választ:

- milyen módon történik az elvégzendő feladatok szétosztása a processzorok között;
- milyen módon tudják a processzorok az erőforrásokat használni? Ez a memória esetében lehetséges megosztott módon (shared memory), vagy elkülönült módon (distributed memory) amikor minden processzor egy számára rezervált, fizikailag vagy logikailag elkülönülő memóriával rendelkezik;
- milyen módon tudnak a processzorok egymással kommunikálni adatot közölni, illetve kapni, azaz milyen az üzenettovábbítás algoritmusa.

A processzorok közötti feladatkiosztásra két lényegében eltérő megoldás lehetséges:

- *Az egyes processzorok nagyobb, önálló feladatokat oldanak meg (kis mértékű kommunikáció), ez a „durva szemcsés" feladatkiosztás,*
- *Az egyes processzorok kisebb, egyszerű feladatokat oldanak (erős kommunikációs igény), ez a „finom szemcsés" feladatkiosztás.*

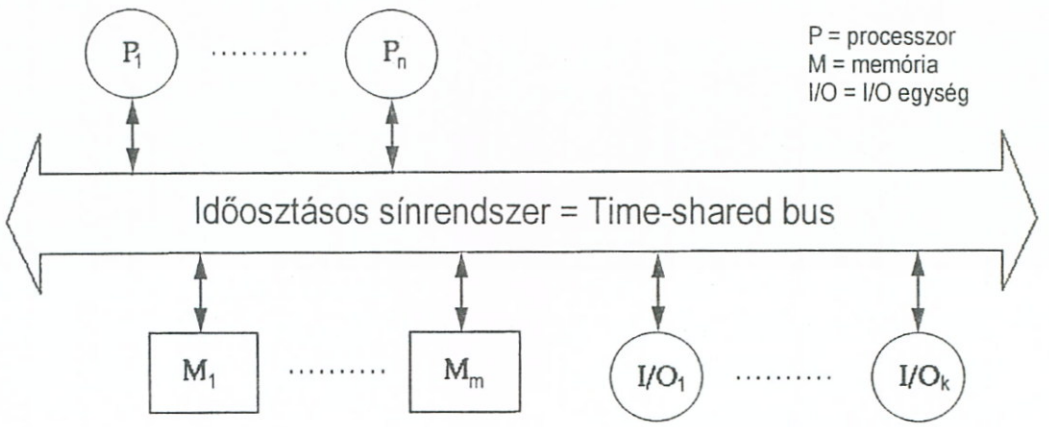
A processzorok erőforráshasználatát két módon lehet megoldani:

- *Közös erőforrások, (pl. memória) esetén meg kell határozni a processzorok és az erőforrások kiosztási eljárását. A közös memóriahasználat adat-koherencia problémákat vet fel (ugyanazt az adatelemet több processzor is módosíthatja),*
- *Részben vagy teljesen saját erőforrásokkal rendelkező processzorok esetében a processzorok között intenzív kommunikációt és szinkronizálást kell megszervezni.*

A processzorok közötti kommunikáció megszervezéséhez a következő eljárásokat kell meghatározni.

- *Ütemezés* (aszinkron, szinkron)
- *Kapcsolatfelépítési algoritmus* (circuit switching, packet switching stb.)
- *A kommunikációs kapcsolatok vezérlése* (központi, decentrális)
- *A kommunikációs hálózat térbeli kiépítése vagy topológiája* (statikus, dinamikus)

Erre példa lehet a buszorientált kommunikáció a processzorok között, (lásd 20. sz. ábra) amikor a processzorok, a memóriák és az I/O eszközvezérlések egy közös átviteli csatornát használnak.



20. sz. ábra
Multiprocesszoros rendszer buszkommunikációval

Az elmúlt években a multiprocesszoros rendszerek felépítésére az „építőkocka" elv alkalmazása a jellemző.

2.4.8.1. A transzputer

Ennek az elvnek az első fizikai megvalósítása a „transzputer" volt. *A transzputer egy egyetlen chipen megvalósított RISC alapú processzor, melyhez egy memóriamodul is tartozik. Rendelkezik egy DMA vezérlővel* (DMA = Direct Memory Access = közvetlen memória-hozzáférés), *mely képessé teszi más transzputerekkel való kapcsolattartásra. Ezért megfelelő számú transzputer, „építőkocka" szerű összekapcsolásával igen nagy teljesítményre képes szuperszámítógépek építhetők fel.*

2.4.8.2. Az SMP architektúrájú multiprocesszoros rendszerek

Lényegében a transzputer alapgondolatát fejlesztette tovább *az SMP (Symmetric Multiprocessors) általánosan elterjedt típusú multiprocesszoros architektúra napjainkban. Ez azt jelenti, hogy azonos, sorozatban gyártott processzorokból épülnek fel a szuperszámítógépek.*

Erre példa a SUN által gyártott 3 központi egységből felépített SMP architektúrájú gép. Ezekből kettő 8–8 processzoros, a harmadik 2–4 processzoros. A 18–20 processzor egymással való kommunikációját, a több gépbe szétosztott memóriát és a csatlakozó berendezések együttműködését a kezelő szoftver és a hálózat együttesen biztosítja. A 3 gépet többszörös, egyenként 622 Mbit/sec sebességű hálózat kapcsolja össze. Ezt a COMPS (Scalable Coherent Interface = változtatható léptékű koherens interfész) architektúrát például egy részecskegyorsítótól érkező 200 Mbit/sec sebességű adatáram feldolgozásával próbálták ki.

2.4.8.3. A S2MP architektúrájú multiprocesszoros rendszerek

A Silicon Graphics és a szuperszámítógépekre specializálódott leányvállalata a *Cray Research 1996-ban jelentette be a S2MP új multiprocesszoros architektúrát, amely jelentősen kibővíti a szimmetrikus multiprocesszoros* (SMP) *rendszerek skálázhatóságának eddigi határait.*

Az „építőkocka"-elv alapján tervezett S2MP rendszerben 1-1024 processzoros gépek építhetőek. Az SGI ORIGIN 2000 rendszerben minden építőkocka egy nagy teljesítményű számítógép, ami típustól függően 2–512 db 64 bites RISC processzorból, 1 Tbájtig kiépíthető memóriából, I/O

alrendszerből és egy CrayLink csatolóból áll. Az építőkockák a CrayLink Interconnect csatoló segítségével kapcsolhatók össze egyetlen nagyszámítógéppé.

Az S2MP architektúra kulcsa tehát a CrayLink technológia, ami leginkább egy nagy teljesítményű telefonközponthoz hasonlítható. Ez a processzszorokat összefogó adatátviteli központ biztosítja, hogy a rendszer részei egyetlen számítógépként viselkedjenek. A központ tartalmaz egy külön „tudakozó szolgálatot", ami a memória koherenciáját biztosítja.

Az építőkocka-elv lehetővé teszi a számítási erőforrások rugalmas átcsoportosítását. Például egy vállalat fejlesztő, pénzügyi és kereskedelmi részlege által külön-külön beszerzett számítógépeket a CrayLink segítségével egyetlen nagygéppé alakíthatják át csúcsterhelés esetén (pl. az éves zárás idejére).

Emellett az S2MP architektúra felépítésénél fogva képes a hardver és szoftver meghibásodások izolálására (rekonfigurációs képesség) és többszáz Gbájt memória közvetlen címzésére.

2.4.9. Tudásalapú architektúrák, neurális hálók

A mesterséges, neurális (az emberi idegsejtet utánzó) hálózatokon alapuló számítógépek gyakorlati hasznosítása az elmúlt években kezdődött meg.

Az emberi agy könnyedén megbirkózik olyan (pl. alakfelismerési) feladatokkal, amelyek Neumann-elven működő számítógépekkel csak nagyon bonyolult módon kezelhetők. Ez is motiválta *a mesterséges neuronhálózatokon alapuló számítógépek kifejlesztését*, melyek *az emberhez hasonlóan taníthatóak, képesek nagyfokú általánosításra (absztrakcióra) és asszociációra.*

Alkalmazásuk

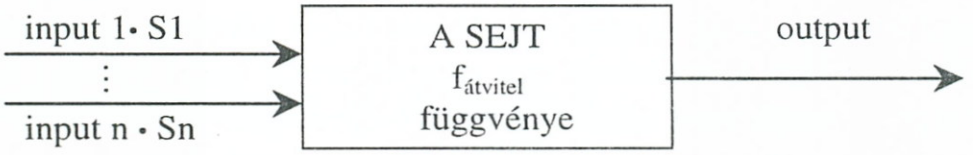
- *az alakfelismerés,*
- *a képfeldolgozás,*
- *a hangfelismerés stb.*

felhasználási területeken már a gyakorlatban is megkezdődött.

Így például hazánkban is használnak neurális automata elvén működő rendszámfelismerő rendszert, mely képes – a videokamera által felvett gépjármű képéből kiemelve – a rendszámot felismerni és azonosítani.

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 neuronsejt outputja* (lásd 21. sz. ábra).



21. sz. ábra
A neuronális háló egy sejtjének működése

A neuronális 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.

Így például egy kávéscsésze fogalmát a következőképpen „kódolja" a neuronális automata: felbontja a kávéscsésze látványát, formáját részjellemzőkre, és ezek a részinformációk a hálózat szinaptikus súlyaiban tárolódnak.

Egy lehetséges további fejlesztési terület az emberi agy és a számítógép összekapcsolása. Ezt tűzik ki célul az ún. biológiai szuperchip létrehozására irányuló kutatások, melyek az első lépést jelentik a biológiai számítógép felé. Ez megoldást jelenthet látásvesztés, hallászcsökkenés stb. esetén, mivel közvetlenül az agy szürkeállományára „rákapcsolhatnák" az elektronikus protéziseket.

2.5. A SZÁMÍTÓGÉPRENDSZEREK ÜZEMMÓDJAI

Attól függően, hogy a felhasználó a számítógépes rendszert milyen formában használhatja, a számítógéprendszereknek különböző üzemmódjait különböztethetjük meg. *A számítógéprendszerek üzemmódjait*

- *az egyidőben kiszolgált felhasználók száma,*
- *az „egyidőben" futtatható programfolyamatok száma,*
- *a felhasználói igények kiszolgálásának időbelisége,*
- *a rendszer térbeli felépítése*

szerint fogjuk vizsgálni.

Mielőtt rátérnénk az egyes üzemmódok tárgyalására, egy megjegyzést kell tenni arról, hogy az egyidejűség fogalmát hogyan használjuk az elkövetkezőkben. A lényeg az, hogy a felhasználó ember szempontjából fogjuk vizsgálni az egyidejűséget, azaz azokat az eseményeket, melyeket az ember egyidejűnek érzékel, egyidejűnek fogjuk nevezni. Így például egy egyprocesszoros gépnek vannak olyan üzemmódjai, melyeknél több felhasználó (például terminálokkal) párhuzamosan dolgozik a számítógéppel és mindegyik úgy érzékeli, hogy a gép az igényeit teljes körűen, egyidejűleg kiszolgálja. (Például egy adatkérésre egy másodpercen belül választ kap.) Ekkor viszont a programok csak látszólag futnak egyidejűleg, mivel egy processzor egyidőben csak egy program utasításait képes végrehajtani.

2.5.1. Egyfelhasználós és többfelhasználós rendszerek

Egy számítógépes rendszer egyidőben egy vagy több felhasználó igényeit szolgálhatja ki. Eszerint megkülönböztetünk:

- *egyfelhasználós (Single User) és*
- *többfelhasználós (Multi User) rendszereket.*

Kezdetben a számítógépek egyfelhasználós üzemmódban működtek, egyidejűleg csak egy felhasználói program volt a gép memóriájában és ez rendelkezett a gép összes erőforrásával (Single Programming Mode).

Ennek az üzemmódnak komoly hátránya volt, hogy ha a program a processzor sebességéhez képest nagyon lassú (pl. egy I/O lemezművelet 10 milliószor lassúbb egy gépi utasítás végrehajtásánál) input/output műveletet hajtott végre, akkor a processzor hosszú időn keresztül várakozásra kényszerült és nem volt megfelelően kihasználva. Ennek idődiagramját mutatja a következő ábra.

I/O egységek leterhelése	VÁRAKOZIK	DOLGOZIK	VÁRAKOZIK	
processzor leterhelése	DOLGOZIK	VÁRAKOZIK	DOLGOZIK	
program	processzort igénylő műveletek	I/O műveletek	processzort igénylő műveletek	idő

22. sz. ábra
A processzor és I/O műveletek végrehajtása

A többfelhasználós üzemmódú számítógépes rendszerek egyidejűleg több felhasználó kiszolgálására képesek. Ehhez arra van szükség, hogy a számítógép erőforrásait (processzor, memória stb.) valamilyen stratégia (algoritmus) szerint az operációs rendszer megossza az egyes feladatok között. Ez elsőként a multiprogramozásos üzemmód bevezetésével valósult meg.

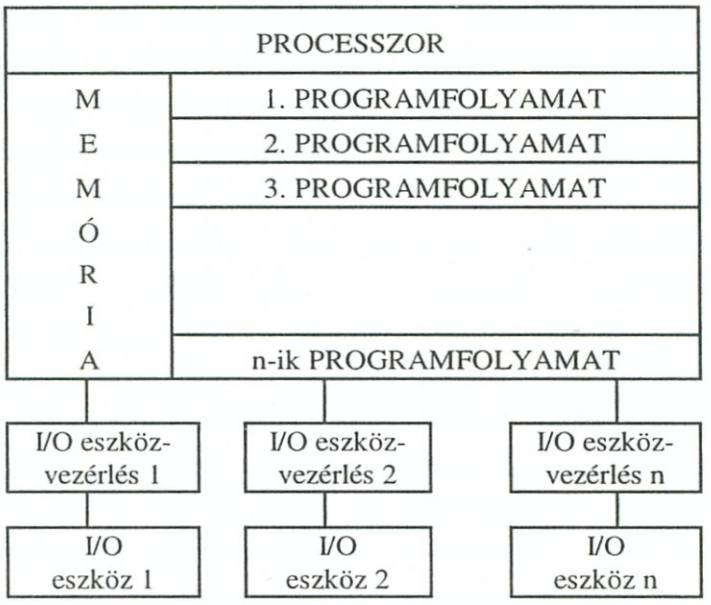
2.5.2. Multiprogramozás, multitasking

A multiprogramozásos rendszerekben egyidőben több programfolyamat (job, task, process) kerül betöltésre a számítógép memóriájában, melyek valamilyen eljárás szerint osztozkodnak a programfutáshoz szükséges erőforrásokon (memória, I/O eszközök használata stb.). Amennyiben egy program lassú input/output műveletet végez, a processzor ezalatt elkezdi a következő programfolyamat végrehajtását. Ha ez is input/output műveletet igényel, a processzor a sorban következő programfolyamat végrehajtását kezdi meg (lásd 23. sz. ábra).

<table>
<tr><td>1 program-folyamat</td><td>pro-cesszort igénylő művelet</td><td colspan="3">I/O művelet</td><td>processzort igénylő művelet</td><td>I/O művelet</td></tr>
<tr><td>2 program-folyamat</td><td>várako-zik</td><td>processzort igénylő művelet</td><td colspan="3">I/O művelet</td><td>processzort igénylő művelet</td></tr>
<tr><td>3 program-folyamat</td><td colspan="2">várakozik</td><td>proces-szort igénylő művelet</td><td colspan="3">I/O művelet</td></tr>
<tr><td>⋮</td><td colspan="6">⋮</td></tr>
<tr><td>n-ik program-folyamat</td><td colspan="3">várakozik</td><td>processzort igénylő művelet</td><td colspan="2">I/O művelet</td></tr>
</table>

23. sz. ábra
Multiprogramozás több programfolyamattal

A multiprogramozott üzemmódnak két lényeges feltétele, hogy a memória a programfolyamatoknak megfelelően részekre (partíciók) legyen osztva, és az I/O műveletet végrehajtó egységek önálló vezérléssel rendelkezzenek, azaz a processzortól függetlenül is képesek legyenek a feladatvégrehajtásra.



24. sz. ábra
Memóriafelosztás és az I/O eszközök vezérlésének szerepe multiprogramozásnál

A multitasking üzemmód definícióját tekintve az informatikai szakma véleménye nem teljesen egységes:

- egyes szerzők ezt a multiprogramozott üzemmód más nevének (szinonimájának) tekintik. Míg mások
- a multitaskingot az egyfelhasználós mikrogépeknek azzal a képességével azonosítják, hogy „egyidejűleg" lehetséges több programfolyamat futtatása is. (Például a WINDOWS 95-ben ezeket a programokat a megnyitott ablakok száma határozza meg.)

A multiprogramozott üzemmódnak két lényegében különböző változata van, melyek abban különböznek, hogy az erőforrások szétosztása (hozzárendelése) milyen módon történik meg az egyes programfolyamatok között.

! ***Prioritásos elv:*** *az egyes feladatoknak megfelelő programfolyamatoknak prioritása van, azaz fontosságuknak megfelelő sorrendben kapják meg a számítógép erőforrásait (processzor, memória, I/O vezérlések).* A 23. sz. ábrán bemutatott feladatvégrehajtásnál az 1 programfolyamatnak van a legnagyobb prioritása, 2 programfolyamatoknak a következő és így tovább.

! ***Az időosztásos elv:*** *az egyes feladatoknak megfelelő programfolyamatokhoz ún. „időszeletek" kerülnek hozzárendelésre, melyek meghatározzák, hogy az adott feladat mennyi időre veheti igénybe a processzort.*

Az időosztás elvén megszervezett multiprogramozásos üzemmódot time-sharing (időosztás)-nak nevezik. Ezen az elven működnek az összes korszerű operációs rendszerek (LUNIX, WINDOWS 98 és NT stb.).

Természetesen a két elv együttesen is alkalmazható, azaz time-sharing esetén is lehet a programfolyamatoknak prioritása. !

2.5.3. Kötegelt feldolgozás (Batch Processing) és interaktív feldolgozás (Interactiv Processing)

A felhasználói igények kiszolgálásának időbelisége szerint van kötegelt és interaktív feldolgozási üzemmód.

Kötegelt feldolgozás *esetén a programfolyamathoz szükséges összes adatot először összegyűjtik, és számítógépes feldolgozásukat időszakosan hajtják végre* (például naponta, havonta, évente meghatározott időpontban). Erre példát a mikroszámítógépeknél a *.BAT kiterjesztésű fájlok végrehajtása során találhatunk. *Amennyiben az adatkommunikáció kötegelt feldolgozásnál távadat-feldolgozással történik, akkor ezt remote batch-nek nevezzük.* !

Interaktív feldolgozás *esetén a felhasználó állandó és folyamatos kapcsolatban áll a feladatot végrehajtó programfolyamattal. Ennek előnye, hogy a feladatot előre nem kell részleteiben megtervezni, mivel a felhasználó a feladat végrehajtását maga vezérli, szükség esetén lépésenként módosíthatja a részfeladatok sorrendjét.* !

Az interaktív rendszereknek két altípusa van, lényegében attól függően, hogy a felhasználó az ember, vagy egy gépi folyamat.

Dialógus üzemmód *esetén a felhasználó kérdez-felelek módon, lépésenként folyamatosan kommunikál a programfolyamattal. Ennek tipikus megvalósítása a terminál üzemmód, amikor a felhasználó ember egy képernyőn közli a rendszerrel a parancsokat és adatokat és a rendszertől az adatokat szintén egy képernyőn jeleníti meg. Ez lehet tranzakció orientált üzemmód (Transaction Mode), amikor a felhasználók több munkahelyen, előre jól definiált és azonos feladatokat végeztetnek el a számítógéppel* (tranzakció = a rendszer legkisebb önálló feldolgozási egységének összefüggő eljárássorozata). !

Folyamatvezérelt üzemmód *(Process Control): Ez esetben a „felhasználó" egy gépi folyamat, amit a számítógép vezérel. Ilyenek lehetnek például az ipari folyamatok irányítását, forgalmi rendszerek vezérlését stb. végző számítógépes rendszerek, melyek különböző mérőműszerektől, érzékelőktől érkező jeleket fogadnak és dolgoznak fel.* !

2.5.4. Real-Time rendszer

! *A dialógus és a folyamatvezérelt üzemmód egyaránt lehet valós idejű (real-time) feldolgozás, ami azt jelenti, hogy a számítógépes rendszer meghatározott kritikus időn belül azonnal feldolgozza a fogadott adatokat.* Ez általában:

- dialógus üzemmódban 1 sec alatti reakció- (válasz) időt jelent,
- folyamatvezérelt üzemmódban ez az időtartam jóval kisebb lehet (például a mikroszekundumos tartományba eshet).

A real-time rendszereknek folyamatvezérelt üzemmódban általában négy összetevője van:

- egy adatfogadó egység, mely a külső környezetből (általában valamilyen érzékelőtől vagy mérőműszertől) jövő információkat fogadja,
- egy elemző egység, mely megfelelő formára alakítja át a fogadott jelet és elemzi azt,
- egy esemény vezérlő egység, mely reagál a bejövő információkra, például az irányított folyamatot befolyásoló vezérlő jelek kiadásával,
- egy ellenőrző/koordináló egység, mely az előző három feladat végrehajtását koordinálja és ellenőrzi (például abból a szempontból, hogy a rendszer reakcióideje az időkorlát alatt maradjon).

2.5.5. Centralizált és osztott számítógéprendszerek

A feladatvégrehajtó egységek térbeli elhelyezkedése szerint megkülönböztetünk centralizált és elosztott üzemmódot.

! ***Centralizált rendszer**ben csak egy központi vezérlés és feladatvégrehajtó egység található* (ez nem zárja ki, hogy a rendszer szolgáltatásait távadatfeldolgozó hálózattal a központi számítógéprendszertől igen messze is igénybe vegyék a felhasználók).

! ***Elosztott számítógéprendszer**ben több, térben esetleg távoli, önálló vezérléssel rendelkező és feladatvégrehajtásra képes számítógép található, melyeket egy hálózat köt össze,* és adatátviteli vonalakon (adatbuszok, telefonvonalak stb.) tartanak kapcsolatot egymással. *Az egyes részfeladatokat ellátó számítógépek együttműködését (Slave Computer) általában egy kitüntetett számítógép (Master Computer) vezérli.* Az elosztott számítógéprendszerek alapvető célja tehát az, hogy a feladatok és erőforrások a lehető legésszerűbben és hatékonyan kerüljenek felosztása a rendszer tagjai között.

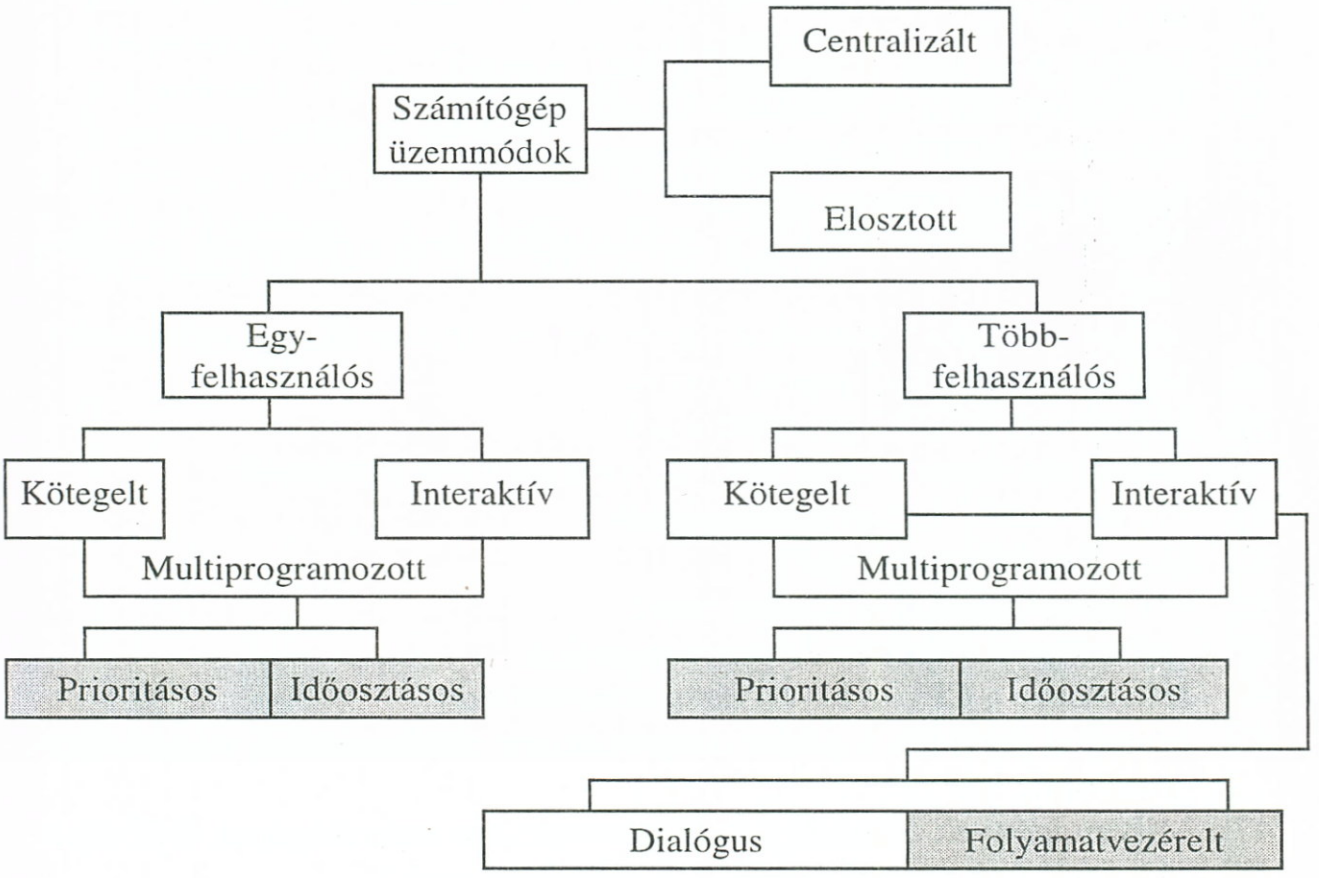
2.5.6. On-line (közvetlen bemenet) és off-line (közvetett bemenet) kapcsolatok

A számítógépes rendszerekben a központi egység és az ettől különböző más részegységek (háttértárak, perifériák, valamint a távoli feldolgozóegységek) közötti kapcsolat lehet:

- *on-line, ha a két eszköz fizikailag úgy összekapcsolt, hogy közöttük aktív vezérlési és/vagy adatátviteli kapcsolat áll fenn,*
- *off-line, ha két eszköz között nincs aktív vezérlési és/vagy adatátviteli kapcsolat.*

Így például, ha egy nyomtató nyomtat, akkor kapcsolata a számítógéppel on-line, ha papírt teszünk a nyomtatóba, akkor az off-line állapotban van.

A számítógéprendszerek különböző üzemmódjainak összefüggéseit a 25. sz. ábra szemlélteti.



25. sz. ábra
A számítógéprendszerek üzemmódjai

? ELLENŐRZŐ KÉRDÉSEK

18. Mit nevezünk számítógéprendszer architektúrának?
19. Mivel jellemzünk egy számítógéprendszer architektúrát?
20. Miért szükséges a chipek ciklikus működése?
21. Hogyan számíthatjuk ki a számítógép feladatvégrehajtási idejét?
22. Milyen módon növelhetjük meg egy számítógéprendszer teljesítményét a feladatvégrehajtási idő csökkentésével?
23. Mi a MIPS és a MFLOPS?
24. Mit mutatnak a „Benchmark"-ok?
25. Milyen szempontok szerint csoportosíthatjuk a számítógéprendszereket?
26. Mi jellemzi a nagyszámítógépeket?
27. Mutassa be a hibatűrő architektúrák fontosabb jellemzőit!
28. Jellemezze a közepes teljesítményű számítógépeket!
29. Mit nevezünk kis vagy mikroszámítógépnek?
30. Mutassa be a mikroszámítógépek elvi felépítését rajzban!
31. Mi a NETPC?
32. Definiálja az adat és utasításfolyam fogalmát!
33. Mit értünk SISD és SIMD architektúra alatt?
34. Mit értünk MISD és MIMD architektúra alatt?
35. Sorolja fel a Neumann elveket!
36. Milyen módszerei vannak a számítógépek nem strukturális gyorsításának?
37. Milyen módszerei vannak a számítógépek strukturális gyorsításának?
38. Mit nevezünk CISC processzornak?
39. Mit nevezünk RISC processzornak?
40. Hasonlítsa össze a RISC és CISC processzorok jellemzőit!
41. Mit nevezünk vektorszámítógépnek?
42. Mi a pipelining lényege?
43. Mit nevezünk ILP processzornak?
44. Definiálja a szuperskalár processzorokat!
45. Jellemezze a CPU-n kívüli társprocesszorokat!
46. Mondjon legalább 3 példát társprocesszor alkalmazására mikroszámítógépeknél!
47. Milyen igények okozzák a multiprocesszoros számítógépek fejlődését?
48. Mit nevezünk multiprocesszoros számítógépnek?
49. Multiprocesszoros architektúrában mit jelent a durva és a finom szemcsés feladatkiosztás?
50. Milyen változatok vannak multiprocesszoros rendszerekben az erőforráshasználat megszervezésére?

51. A processzorok közötti kommunikáció megszervezéséhez milyen algoritmusokat kell meghatározni?
52. Mi a transzputer?
53. Mi a SMP és S2MP architektúra?
54. Melyek a legfontosabb jellemzői a neuronális hálóknak és milyen tipikus felhasználói területeik vannak?
55. Hogyan állítja elő egy sejt a neuronális hálóban az inputból az outputot?
56. Hogyan „emlékezik" a neuronális háló?
57. Milyen osztályozási szempontjai vannak a számítógéprendszerek üzemmódjainak?
58. Mit értünk egy és többfelhasználós üzemmód alatt? Mi a feltétele a többfelhasználós üzemmódnak?
59. Mit nevezünk multiprogramozásos rendszernek?
60. Milyen lényeges feltétele van a multiprogramozásnak?
61. Definiálja a prioritásos és az időosztásos multiprogramozást!
62. Mit jelent a batch processing és a remote batch?
63. Mikor interaktív a számítógépes feldolgozás?
64. Határozza meg a dialógus üzemmód fogalmát!
65. Mi jellemzi a folyamatvezérelt üzemmódot és milyen fontosabb felhasználási területei vannak?
66. Definiálja a real-time rendszer fogalmát!
67. Mi a különbség a centralizált és az elosztott számítógéprendszer között?
68. Mit jelent az on-line és az off-line kapcsolat?

III.

A PROCESSZOR

E fejezetben a processzorok felépítésével és működésével kapcsolatos, általánosan érvényes elveket, jellemzőket és eljárásokat tekintjük át.

Az egyes processzorok vizsgálatát, bemutatását általában a következő témakörök elemzésével szokták elvégezni:

- **A processzor felépítése, részei és ezek kapcsolata**

  A processzorok meghatározott feladatokat ellátó részegységekből épülnek fel. Ezeket és adatkapcsolataikat elvi blokkvázlatokkal fogjuk bemutatni.

- **A processzorok üzemmódjai, állapotai**

  A processzorok működésében védelmi, biztonsági szempontok szerint, illetve a korábbi típusokkal való programkompatibilitás megőrzése érdekében különböző üzemmódokat (állapotokat) különböztetünk meg.

- **Regiszterkészlet**

  A regiszterek a processzorokban az adatok átmeneti tárolására szolgáló, nagyon gyors (és drága) tárolók, melyek a processzoron belüli kommunikációs vezetékeken (belső sín) keresztül tartják az információs kapcsolatot a processzor részegységeivel. Méretük ma általában 32, 64 vagy 80 bit.

- **Utasításfelépítés, utasításkészlet**

Egy processzor „képességeit" és működésének bonyolultságát alapvetően meghatározza a gépi utasítások felépítése és mennyisége (mint már láttuk a CISC és RISC architektúra közötti legfontosabb különbség az utasításfelépítésben és utasításkészletben van).

- **Utasításvégrehajtás**

E kérdéskörhöz tartozik a gépi utasításokat végrehajtó egységek működése, a műveleti vezérlés eljárásai, és a különböző sebességnövelő technikák (az utasításvégrehajtás átlapolása, párhuzamosítása).

- **Memóriakezelés**

A processzor mellett a különböző típusú memóriák a számítógép egyik legfontosabb részegységeit képezik. Strukturális illesztésük, kezelésük hatékonysága, kihasználtságuk meghatározó jelentőségű a teljes számítógép teljesítménye és gyorsasága szempontjából. Emiatt fontosak a processzor memóriakezelési eljárásai (pl. címszámítás).

- **Megszakítások, kivételek**

A processzor megszakítás és kivétel kezelése főként a periféria és háttértárkezelés, valamint a hibaelhárítás szempontjából meghatározó jelentőségű.

3.1. A PROCESSZOROK JELLEMZÉSE

E fejezetben a processzorok jellemzéséhez felhasznált legfontosabb fogalmakat mutatjuk be.

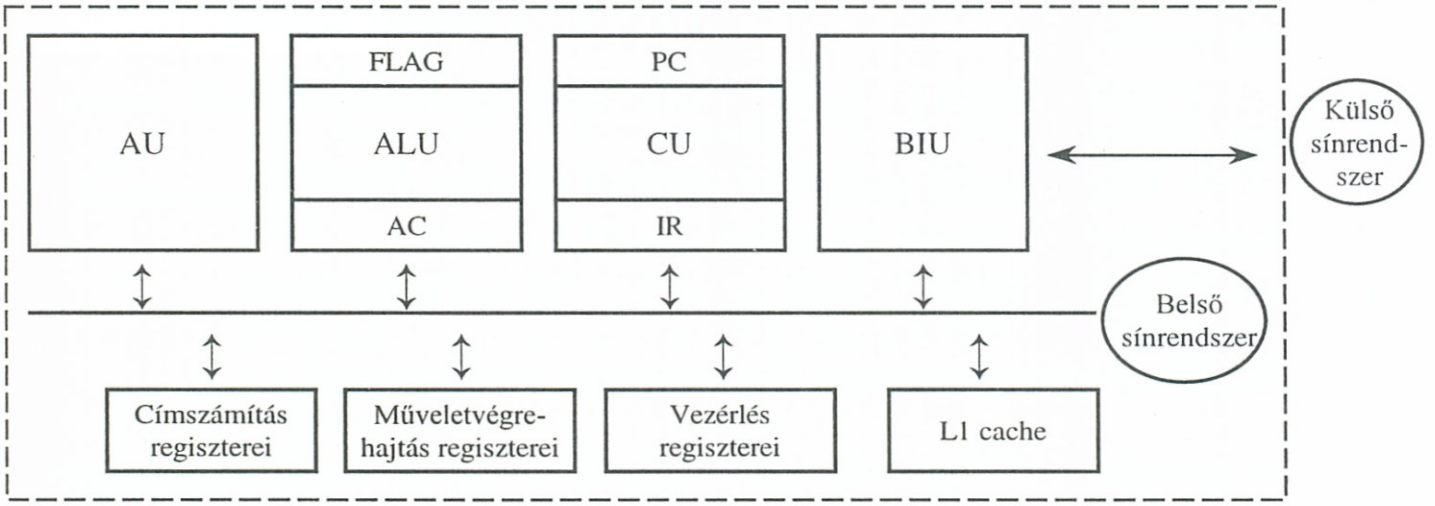
3.1.1. A processzor részei és ezek kapcsolata

Az egyes processzorok felépítésében (például a mainframek, középgépek és mikrogépek) jelentős eltéréseket tapasztalhatunk, de mindegyikre *jellemzőek a következő legfontosabb architekturális építőelemek:*

- *vezérlőegység* (CU = Control Unit),
- *aritmetikai (logikai) egység* (ALU = Arithmetic Logic Unit),
- *regiszterkészlet,*
- *busz illesztőegység* (BIU = Bus Interface Unit),
- *címszámító és védelmi egység* (AU = Adress Unit),

- *belső gyorsító tár* (L1 cache),
- *az előző négy részegység kommunikációját biztosító eszközök (mikroszámítógépeknél a belső sínrendszer).*

Ennek logikai sémáját mutatja a következő ábra:



26. sz. ábra
A processzor logikai felépítése

*A **vezérlő egységnek** a feladata a programban lévő utasítások alapján a teljes számítógép részegységeinek (arithmetikai egység, memória, kommunikációs eszközök, háttér és perifériavezérlések) irányítása, összehangolása.*

Az ehhez felhasznált két legfontosabb regisztere:

- *PC = Program Counter, mely a soronkövetkező 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).

*Az **aritmetikai egység** az utasításkódokban előírt aritmetikai és logikai (pl. logikai ÉS, VAGY stb.) műveleteket hajtja végre.*

Az aritmetikai egységben általában a következő két regiszter mindig megtalálható:

- *AC = Accumulator Register, mely a műveletvégrehajtásnál 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 **busz interfész egység** biztosítja a processzor kapcsolódását a külső sínrendszerhez.*

*A **címszámító és védelmi egység** feladata a programutasításokban található címek leképezése a főtár fizikai címeire és a tárolóvédelmi 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 (lásd 4.3. fejezet).*

3.1.2. A processzorok üzemmódjai és állapotai

A processzorok különböző üzemmódjaira, állapotaik megkülönböztetésére lényegében:

- a védelmi szempontból, a programfolyamatok biztonságos (pl. rendszerösszeomlás nélküli) működtetése miatt,
- az operációs rendszer és a felhasználói programok működtetésének elkülönítése miatt (részletesen kifejtve lásd 4.6. fejezet),
- a korábbi processzorokkal való bináris programkompatibilitás megtartása miatt van szükség.

Lássunk erre néhány példát!

- Megkülönböztetünk supervisor és felhasználói (user) állapotokat, melyek elkülönítik az operációs rendszerhez tartozó programokat a felhasználói programoktól. A privilegizált üzemmódban a processzor végrehajthat olyan utasításokat is, melyek felhasználói üzemmódban tilosak (azaz kivételt eredményeznek).

- *A mikroszámítógépek közül 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 azt „hazudja" a felhasználói programnak, 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 üzemmód*** *Ekkor 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.*

3.1.3. Regiszterkészlet

A processzor regiszterei a felhasználói programok szempontjából három kategóriába sorolhatók:

- *Rendszerregiszterek, melyek a felhasználói programok számára nem „láthatók", nem elérhetők.* Erre például szolgál 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.

Például a Pentium processzorcsaládban a leggyakrabban következő regiszterekkel találkozhat az assembly programozó:

- Általános regiszterek EAX, EBX, ECX, EDX, (32 bit)
- Címzéshez használt regiszterek: EBP, ESP, ESI, EDI (32 bit)
- Utasításszámláló regiszter: EIP (32 bit)
- Szegmensregiszterek: CS, DS, SS, ES, FS, GS (16 bit)
- Állapotjelző regiszter: EFLAGS (32 bit)

Meg kell jegyezni, hogy a napjainkban használatos mikroprocesszorok (például Pentium II és III, AMD K6 és K7 stb.) a CISC utasításokat elemi RISC utasításokra bontják fel, vagyis emulálják, és ehhez külön regiszterkészletet használnak. Ezért a fizikai regiszterkészlet különbözik a programokban felhasználható logikai regiszterkészlettől.

3.1.4. Utasításszerkezet, utasításkészlet és utasítástípusok

Utasításszerkezet alatt az elemi szintű (gépi kódú), a processzor által közvetlenül értelmezhető utasítások felépítését értjük. Ez határozza meg a processzor számára, hogy a gépi utasításokban szereplő bináris értékeket, azaz az utasítás egyes részeit hogyan értelmezze.

Az utasításszerkezet három leglényegesebb része

- ***műveleti kód*** *(opcod), mely meghatározza a processzor számára, hogy milyen műveletet kell végrehajtania,*
- ***operandus hivatkozások****, mely azt adja meg a processzor számára, hogy hol található, illetve milyen adatokkal kell a műveletet végrehajtani.*
- ***kiegészítő, módosító rész,*** *melyben olyan kiegészítő információkat adhatunk meg a processzor számára, melyek a műveleti jelrész illetve az operandus hivatkozások teljes körű (pontos) értelmezéséhez szükségesek.*

Az utasítások elvi felépítése a következő:

MŰVELETI JELRÉSZ	KIEGÉSZÍTŐ RÉSZ	OPERANDUS HIVATKOZÁSOK

27. sz. ábra
A gépi utasítás felépítése

Utasításkészlet

Egy processzor utasításkészlete alatt azoknak az elemi szintű gépi kódú utasításoknak az összességét értjük, melyek végrehajtására a processzor a legalsó, hardver szinten képes. Ez az a szint, amelyre a fordítóprogram a különböző programnyelveken megírt programokat lefordítja.

Egy processzor utasításkészletének értékelésekor a következő kérdésköröket célszerű elemezni:

- Mennyi az utasításkészletben az elemi utasítások száma és ezekkel milyen feladatokat lehet megoldani? Nyilvánvaló, hogy minél több feladatot képes a processzor elemi szinten megoldani, annál több területen lehet hatékonyan alkalmazni. Ugyanakkor azt is vizsgálni kell, hogy a processzor felhasználási területén melyek azok az utasítások, melyeket nagyon gyakran kell használni. Ha nagyon sok utasítás csak ritkán kerül felhasználásra, ezek feleslegesen terhelik az áramköri erőforrásokat. Ugyanakkor arra is érdemes odafigyelni,

hogy a processzor felhasználási területen nincs-e olyan sűrűn ismétlődő feladat, amelyet célszerű az utasításkészletben is tükröztetni kell. (Ilyen megfontolások vezettek – a multimédia alkalmazások terjedése miatt – a multimédiás képességű, azaz az MMX architektúrájú processzorok kifejlesztéséhez.)

- Milyen típusú adatokat és milyen pontossággal tudunk az utasításkészletben lévő gépi utasításokkal feldolgozni? (Például a lebegőpontos utasítások milyen pontossággal végezhetők el.)
- Milyen következetesség van az utasításkészlet kialakításában? Így például a címek kiszámítása minden utasításban azonos módon történik-e, függetlenül attól, hogy milyen utasítást kell végrehajtani, vagy bármelyik adattípus, címzési mód használható-e minden utasításban?

Utasítástípusok

Funkcionális feladatuk szerint az utasításkészletben a következő főbb utasítástípusokat tudjuk megkülönböztetni:

- *adatátviteli, adatmozgató utasítások,*
- *műveletvégrehajtó utasítások,*
- *vezérlő utasítások.*

Ezek döntő többsége a soros utasításvégrehajtást (azaz a következő végrehajtandó utasítás a megelőző utasítás után következik) nem befolyásolja. A vezérlő utasítások egy részénél (ugróutasítások, szubrutinhívás stb.) az utasításvégrehajtás nem a következő utasítással folytatódik.

Adatátviteli utasítások

Az adatátviteli utasításokkal különböző tárolók között lehet mozgatni, másolni az adatokat

- *memória, regiszter ↔ memória, regiszter (MOVE)*
- *memória ↔ háttértárolók, perifériák (IN, OUT)*
- *regiszter, memória ↔ verem (POP, PUSH).*

Általában az utasításkészletben megkülönböztetik

- az egy vagy több byte-os átvitelt és
- a szavas (2, 4, 8 byte-os) átvitelt.

Műveleti utasítások

- *aritmetikai utasítások* (ADD, SUB, MUL, DIV stb.),
- *logikai utasítások* (AND, OR, XOR stb.),
- *bitléptető/forgató utasítások* (SLL, SRL stb.),
- *bitműveletek, karakterlánc (string) műveletek (CMPB stb.),*
- *multimédiás és 3D grafikus műveletek (PADD, PADDS stb.).*

Az *aritmetikai utasítások* csoportjába tartoznak a bináris egész (fixpontos) számokkal, a lebegőpontos számokkal és a binárisan kódolt decimális számokkal végzett szokásos alapműveletek (összeadás, kivonás, szorzás, osztás).

Az aritmetikai műveletek végrehajtása után az eredménytől függően a processzor beállítja az állapot (flag) regiszter egyes jelzőbitjeit. Így például ezek alapján a programozó vizsgálhatja a következő eseteket és ez alapján ugró utasításokat vezérelhet:

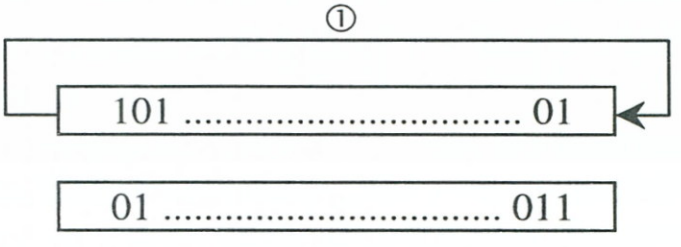
- *átvitel (carry) jelzőbit 1-esre lesz beállítva, ha a műveleti eredményben, a legmagasabb helyiértéken átvitel történik,*
- *nulla (zero) jelzőbit 1-esre lesz beállítva, ha a művelet eredménye 0,*
- *előjel (sign) jelzőbit 1-esre lesz beállítva, ha a művelet eredménye negatív,*
- *a túlcsordulás (overflow) jelzőbit 1-esre lesz beállítva, ha a művelet eredménye nagyobb mint a tárolható legnagyobb érték.*

A *logikai utasítások* a szokásos Boole algebrai műveletek (logikai ÉS, logikai VAGY, logikai NEM ... stb.) bitsorozatokon való elvégzését teszi lehetővé.

Például:

MEZŐ 1	1101 | 1100	
MEZŐ 2	1011 | 1101	Logikai ÉS
EREDMÉNY MEZŐ	1001 | 1100	

A *léptető és forgató utasítások* az operandus címén lévő bitsorozatot jobbra vagy balra léptetik (shift), illetve körbeforgatják (rotate). Utóbbi esetében például balra forgatáskor a legfelső helyiértéken a bitsorozatból kilépő bit jobboldalról a legalacsonyabb helyiértéken újra belép.



balra forgatás 1 bittel

A léptetés és forgatás legtöbbször (pl. Intel processzorok utasításkészlete) a carry flag közbeiktatásával valósul meg.

A *bitműveleti utasításokban* az operandusok bitsorozatok formájában vesznek részt a műveletben.

A karakterlánc műveletek

Ezekkel az utasításokkal karaktereket (betű, szám stb.) tartalmazó mezőkön lehet műveleteket végrehajtani (pl. összehasonlítás stb.).

Multimédiás és 3D grafikus műveletek

Ezekkel az utasításokkal a 3D grafikához és a képfeldolgozáshoz szükséges vektoros adatokat dolgozzák fel (SIMD utasítások).

Vezérlő utasítások

Ezekkel az utasításokkal egyrészt a program soros utasítás-végrehajtását lehet vezérelni, másrészt a processzor működését szabályozni. A program futását, az utasításvégrehajtás sorrendjét vezérlő utasítások:

- *feltétel nélküli vezérlésátadás* (JMP),
- *feltételes vezérlésátadás* (pl. JNE),
- *szubrutin hívás* (CALL),
- *visszatérés szubrutinból* (RET),
- *ciklus képző utasítás* (LOOP).

A processzor működését szabályozó utasítások közé tartoznak például a különböző programvezérelt megszakítást kérő utasítások (pl. INT).

3.1.5. A processzor utasításfeldolgozása

A processzorok utasításfeldolgozása az aritmetikai egység művelet-végrehajtásától, az utasításfeldolgozás műveleti vezérlésének különböző lehetőségeitől és az utasításfeldolgozás gyorsításától (pipelining) függ.

*Az **ALU működését** és „képességeit" a műveletekben értelmezett adattípusok alapján értékelhetjük.* Ezek:

- *fixpontos számok,*
- *lebegőpontos számok,*
- *binárisan kódolt decimális (BCD) számok,*
- *multimédiás (MMX) adatok,*
- *karakteres mezők,*
- *bitmezők.*

A processzor teljesítményében meghatározó jelentőségű a műveletvégrehajtó egységek száma és működése. Ezek lehetnek például fixpontos és lebegőpontos végrehajtó egységek. Ezeket az ún. „n"-utas szuperskalár processzorokban, mint már láttuk, többszörözik.

A **vezérlőegység működésének** értékelésében a legfontosabb szempont a műveleti vezérlés megoldása. Ez lehet:

- huzalozott (áramkörileg megvalósított) vagy
- mikroprogramozott (interpreter végrehajtású).

Az utasításvégrehajtást a pipeline, szuperskalár szervezéssel gyorsíthatjuk.

3.1.6. A processzorok tárolókezelése

Ennek értékelésénél a legfontosabb a címzési módok értékelése. Itt vizsgálni kell a relatív (bázis), közvetett (indirekt), közvetlen adat és indexelt címzés adta lehetőségeket, valamint a virtuális címekből a fizikai címek kiszámításának módszerét, valamint az ehhez felhasznált eszközöket (regiszterek, memóriatáblázatok) és a címszámítás hardver megoldását. Ezt az utóbbit az MMU = Memory Management Unit egység végzi, mely a processzoron belül (Intel processzorok AU = Adress Unit egysége) vagy a processzoron kívül (RISC processzorok pl. POWER PC) helyezkedhet el.

A processzorok tárolókezelési rendszerének egyik legfontosabb eleme a tároló védelmi rendszer, mely alapvetően befolyásolja a számítógép biztonságos működtetését.

3.1.7. Megszakítások és kivételek kezelése

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

- *a megszakítások (interrupt) amikor a program utasításainak átmeneti felfüggesztését a processzortól független, külső esemény idézi elő,*
- *a kivétel (exception), amikor a programot a programutasítás végrehajtása alatt a processzoron belül fellépő esemény miatt kell megszakítani.*

Példák:

- DMA megszakításkérés,
- Paritáshiba,
- Osztási hiba,
- Túlcsordulás
- BIOS rutin meghívása (programvezérlet megszakítás = INT).

A megszakítási rendszer értékelésének legfontosabb szempontjai a következők:

- A megszakításkiszolgálás lépései,
- A maszkolható és nem maszkolható megszakítások,
- A megszakításkezelés hardver megoldása,
- Vektoros és nem vektoros megszakításkezelés.

A megszakítási rendszert az 5.2. fejezetben fogjuk részletesen tárgyalni.

3.2. MŰVELETVÉGREHAJTÁS, AZ ARITMETIKAI EGYSÉG MŰKÖDÉSE

A processzor aritmetikai egysége (ALU = Arithmetic Logic Unit) hajtja végre a gépi utasításban meghatározott műveleteket, az utasításban meghatározott (megcímzett) adatokon.

Mielőtt az ALU működését részletesebben megvizsgálnánk, tekintsük át azokat a legfontosabb aritmetikai és logikai műveleteket, melyek végrehajtására az aritmetikai egységnek képesnek kell lennie.

3.2.1. Aritmetikai műveletek

Az aritmetikai műveleteket osztályozhatjuk aszerint, hogy a műveletben résztvevő operandusok milyen adattárolási formában találhatók meg a memóriában.

A numerikus adatok ábrázolásának két legfontosabb típusa az előjeles és a nem előjeles forma.

Előjeles számábrázolást alkalmazunk a szokásos matematikai számítások végrehajtásához. Ezek *legfontosabb formái*:

- *a fixpontos számábrázolás,*
- *a lebegőpontos számábrázolás és*
- *a decimálisan kódolt bináris számábrázolás.*

Nem előjeles számábrázolási formát használnak a processzorok a multimédiás (MMX) és a 3D grafikus műveletekhez.

3.2.1.1. Fixpontos műveletek

! *A **fixpontos számokon** végrehajtott műveleteknél az operandusok általában szó szervezésben kerülnek tárolásra. A szóban található bitsorozat kettes számrendszerben megadja az adott szám tényleges értékét, a legnagyobb helyiértékű bit pedig a szám előjelét határozza meg (0 = pozitív, 1 = negatív).*

Két fixpontos szám összeadása a kettes számrendszer szabályai szerint bitenként történik, két 1-es bit esetén átvitel (carry) képződik a következő helyiértéket (2 hatványt) jelentő bitre.

Operandus 1	...	1	0	1	1	1	0
Operandus 2	...	0	1	0	1	0	1
Átvitel	...	1 ←	1 ←	1 ←	1	0	0

! ***Két fixpontos szám kivonását** visszavezethetjük az összeadásra, ha a negatív szám tárolására a kettes komplemenskódot használjuk.* Emiatt az összeadás és kivonás azonos algoritmussal és azonos műveletvégrehajtó áramkörökkel elvégezhető.

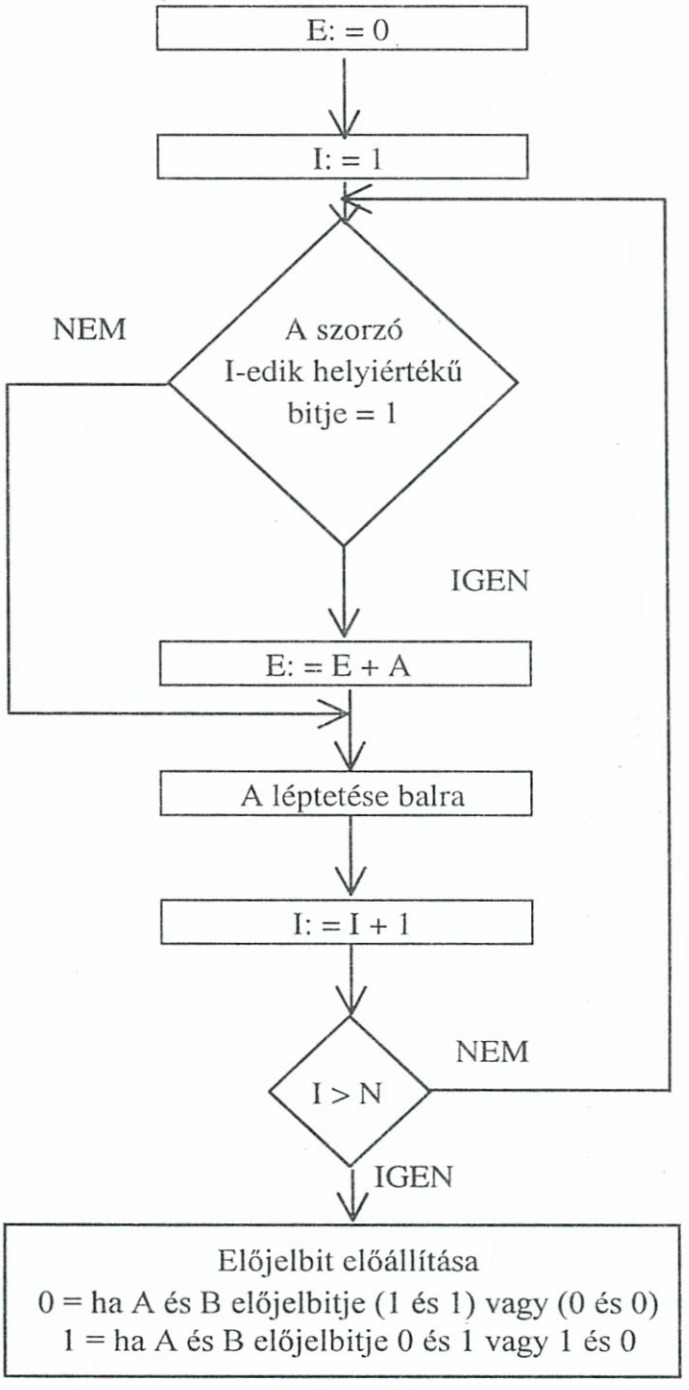
! ***Két fixpontos szám szorzását és osztását** viszont összeadások illetve kivonások sorozatára, illetve balra és jobbra történő 1-bites léptetésekre* (ez ugyanis 2-vel való szorzást illetve osztást jelent) *vezethetjük vissza.* Erre példaként nézzük meg a szorzás algoritmusát. Ehhez jelölje

E = a szorzatot (eredményt) tároló mezőt,

A = szorzandót,

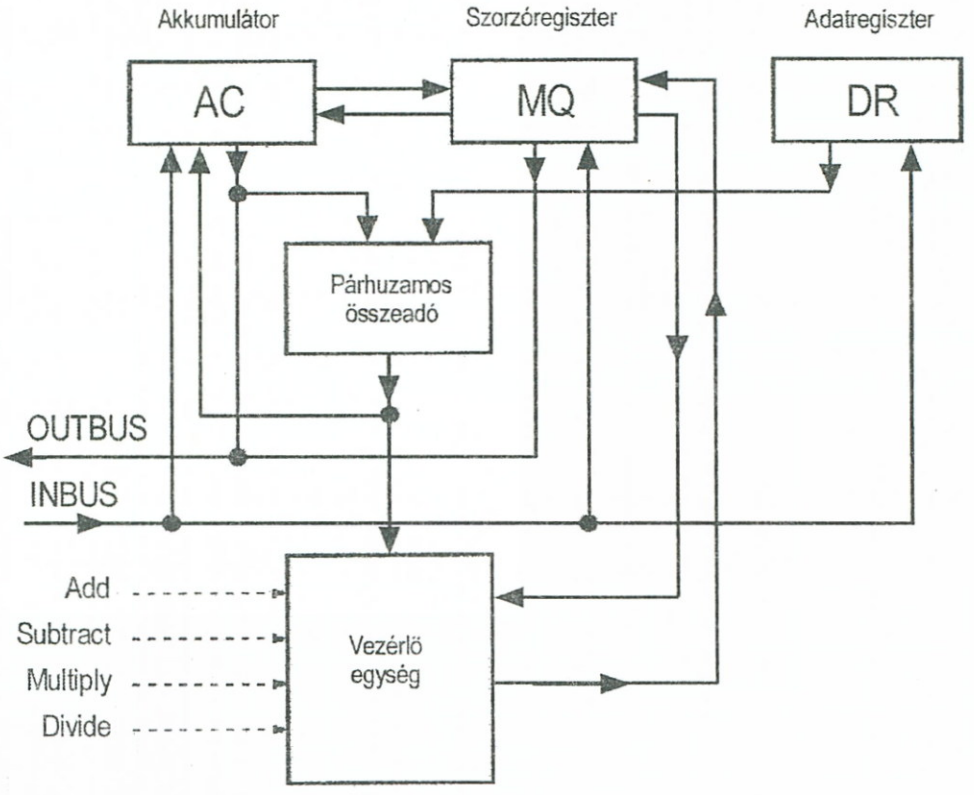
B = szorzót,

N = a szorzó helyiértékeinek számát.



28. sz. ábra
A fixpontos szorzás algoritmusa

A négy fixpontos aritmetikai műveletet „összeadással" végrehajtó hardver egység blokksémáját mutatja be a 29. sz. ábra.



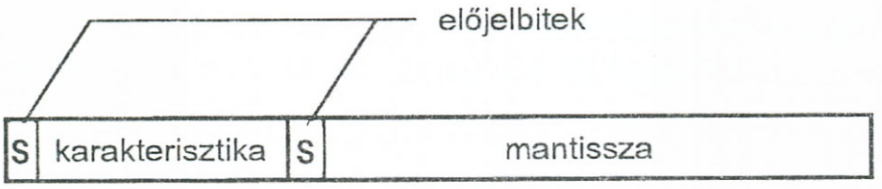
29. sz. ábra
Aritmetikai műveletvégző egység blokksémája

! *Levonhatjuk tehát a következtetést, hogy fixpontos számok aritmetikai műveleteit vissza tudjuk vezetni több lépésben végrehajtott bitenkénti összeadásra és shiftelésre.*

3.2.1.2. Lebegőpontos műveletek, az IEEE 754 és 854-es szabvány

! *A **lebegőpontos számábrázolást** általában olyan számításigényes algoritmusoknál használjuk, melyeknél a szükséges számtartomány (a legkisebb és legnagyobb érték eltérése) jelentős mértékű. A lebegőpontos műveletekhez a számokat $M*2^K$ alakban (M = mantissza, K = karakterisztika) ábrázoljuk.*

Ez lényegét tekintve azt jelenti, hogy egy lebegőpontos szám ábrázolásához két fixpontos szám – egyik a karakterisztika, másik a mantissza – tárolását kell megoldanunk, melyeket a számítógépen történő műveletvégrehajtáskor együtt vegyük figyelembe.



30. sz. ábra
A lebegőpontos számok ábrázolási formája

A számítástechnika történetének kezdeti időszakában többfajta lebegőpontos számábrázolást is használtak. *Az 1985-(86)-as évben került elfogadásra az IEEE 754-es (854) számú lebegőpontos számábrázolási szabvány, melyet a legfontosabb processzorgyártók is elfogadtak.*

A szabvány meghatározza a lebegőpontos számábrázolás normalizált formáját. Ezek szerint ennél:

- *a mantissza előjele* (fixpontos számként) *0 ha szám pozitív és 1 ha negatív,*
- *a mantisszában lévő fixpontos szám.1-re normalizáltan értendő, azaz 1. a formájú.*

A szabvány a lebegőpontos műveletvégrehajtáshoz többfajta pontosságot definiál (lásd 31. sz. ábra).

	Előjel	Mantissza	Karakterisztika
egyszeres pontosság = 32 bit	1 bit	23 bit	8 bit
dupla pontosság = 64 bit	1 bit	52 bit	11 bit
kiterjesztett pontosság = 80 bit	1 bit	64 bit	15 bit
négyszeres pontosság = 128 bit	1 bit	112 bit	15 bit

31. sz. ábra
A különböző pontosságú lebegőpontos számok ábrázolási formája

Bonyolultabb, igen sok lépésből álló számítások során előfordulhat, hogy a műveletek eredménye a legkisebb illetve legnagyobb mantisszájú és karakterisztikájú (a számítógépen még ábrázolható) számnál kisebb, illetve nagyobb. Ezt alulcsordulásnak illetve túlcsordulásnak nevezzük. Többek között az ilyen esetek kezelhetősége érdekében vezeti be a szabvány

- *a denormalizált számot,*

- *a ±0-t,*
- *a ± végtelen számot és a*
- *„nem számot" (lásd 32. sz. ábra).*

	Előjel	Karakterisztika	Mantissza
denormalizált szám	±	0	tetszőleges szám ≠ 0
nulla	±	0	0
végtelen	±	11...11	0
„nem szám"	±	11...11	tetszőleges szám ≠ 0

32. sz. ábra
Speciális formátumú lebegőpontos számok

Ezek közül a denormalizált szám az igen kis számértékek számítógépes kezelését segíti (alulcsordulás esetén nem kell a műveletvégrehajtást megszakítani hibajelzéssel). A túlcsordulásokat (igen nagy számok) a „végtelen" számmal kezelhetjük. Az ezekkel végzett műveleteket a szabványnak megfelelő lebegőpontos aritmetikai egység a matematikában szokásos módon fogja értelmezni. Így például:

- végtelen ± tetszőleges véges szám = végtelen
- tetszőleges véges szám / végtelen = 0 stb.

Az úgynevezett „nem számok" azt a célt szolgálják, hogy programunk akkor se álljon le, ha az elvégzett művelet (például végtelen/végtelen) matematikailag értelmezhetetlen.

A lebegőpontos műveletek visszavezetése fixpontos összeadásra

A lebegőpontos műveletek végrehajtását – mivel ez esetben két fixpontos számmal, a mantisszával és karakterisztikával kell végrehajtani aritmetikai műveleteket – szintén vissza tudjuk vezetni fixpontos összeadásra. Ehhez csak azt kell átgondolnunk, hogy

- *ha a művelet operandusainak karakterisztikái összeadás/kivonásnál nem egyeznek meg* (azaz a műveletet nem lehet elvégezni), *akkor ezt a problémát a mantissza jobbra léptetésével* (2-vel való osztás) *és a karakterisztika egyidejű balra léptetésével meg tudjuk oldani,*

- *lebegőpontos osztásnál, szorzásnál a mantisszákat fixpontos számként osztani, illetve szorozni kell, a karakterisztikák pedig fixpontos számként kivonásra illetve összeadásra kerülnek.*

!

3.2.1.3. Műveletek binárisan kódolt decimális (BCD) számrendszerben

A BCD kódban egy decimális számjegyet bináris értékének megfelelően tárolunk. Ezek szerint:

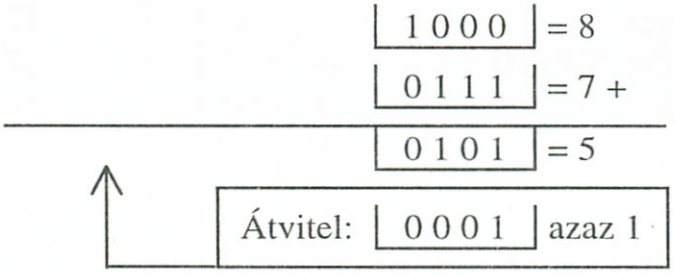
!

Tízes számrendszerbeli számjegy	Bináris négybites kódja
0	0000
1	0001
2	0010
3	0011
⋮	⋮
8	1000
9	1001

33. sz. ábra
Binárisan kódolt decimális számjegyek

A BCD kódú műveletvégzésnél a 4-bittel kódolt decimális számjegyeket számjegyenként – azaz 4 bites egységekben – *adjuk össze. Amennyiben a számjegyek összege 10 vagy annál több, akkor a következő tízes helyiértékre történő átvitelt a tízes számrendszer szabályai szerint képezzük.* Például:

!



34. sz. ábra
BCD kódú számjegyek összeadása

3.2.1.4. Multimédiás (MMX) és a 3D grafikus aritmetikai műveletek

A számítógépes képfeldolgozás és grafika fejlődése arra ösztönözte a hardver fejlesztőket, hogy a multimédiát és a 3D grafikát használó programok műveletigényét az utasításkészletben is érvényesítsék. Emiatt SIMD (egy utasításfolyam, több adatfolyam) típusú gépi utasításokkal egészítették ki az utasításkészletet. Emiatt jelentek meg vektor adattípusok, melyek kezelése új típusú feladatot jelentett.

Ha például két egybájtos értéket össze akarunk adni nem előjeles módon, a következő problémákkal kell szembenéznünk: az operandusok értéke 0–255-ig terjedhet, így összegük 0–510 közötti értéket vehet fel. Ezért az eredmény 1 byte-ban nem fog beleférni. Ekkor két lehetőségünk van:

- *eredménynek a maximális* 255-ös *értéket tekintjük, ezt nevezzük telített számítási formának,*
- *eredménynek csak az alsó 1 byt-on (8 biten képződő biteket) tekintjük, ezt nevezzük körbefűzéses formának* (ez a nevét onnan kapta, hogy ekkor „végigmegyünk" ismételten „körbe" a 0–255 biteken).

Az MMX és 3D grafikus adatokkal, a műveleteket a processzor vektorszámítógépként működve hajtja végre.

3.2.2. Logikai műveletek

A számítógépek műveletvégrehajtásában nagyon fontos szerepük van a logikai (Boole algebrai) műveleteknek. Ezek matematikailag bizonyítható módon visszavezethetők két alapműveletre. Így például:

Operandusok		Nem azonosság	ÉS = AND
A	B	C = A⊕B	C = A∧B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

35. sz. ábra
A logikai műveletek alaptípusai

Ezáltal a szokásos Boole algebrai műveleteket végző gépi kódú utasításokat (NEM, ÉS, VAGY, AZONOSSÁG stb.) áramköri szinten végre lehet hajtani a NEM AZONOSSÁG és az ÉS logikai műveleteket modellező áramkörökkel.

3.2.3. Az aritmetikai műveletek visszavezetése logikai műveleteket végrehajtó áramkörökre

Azt már láttuk, hogy az összes aritmetikai művelet végrehajtásának 2-es számrendszerbeli algoritmusa visszavezethető fixpontos számokkal végzett bitenként történő összeadásra és shiftelésre. Ezért ahhoz, hogy bármilyen aritmetikai művelet végrehajtható Boole algebrai műveleteket modellező áramkörökkel elég ezt megmutatni az 1-bites összeadás esetére.

Jelölje A és B a két 1-bites összeadandót, $Á_{-1}$ az összeadás előző helyiértékéről származó átvitelt, az összeadás eredményét pedig Ö. Ha az összeadás során újabb átvitel képződik, akkor ez legyen $Á_{+1}$. *Ekkor az 1-bites összeadó Boole algebrai igazságtáblázata a következő lesz:*

KIINDULÓ ADATOK			EREDMÉNY	
A	B	$Á_{-1}$	Ö	$Á_{+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	0	1
1	1	1	1	1

36. sz. ábra
Az 1-bites összeadó igazságtáblázata

Mivel Ö = 1 akkor és csak akkor, ha páratlan számú 1-es van A, B, $Á_{-1}$ között, ezért

$$Ö = \overline{A}\cdot\overline{B}\cdot Á_{-1} + A\cdot\overline{B}\cdot\overline{Á}_{-1} + \overline{A}\cdot B\cdot\overline{Á}_{-1} + A\cdot B\cdot Á_{-1} = (A\oplus B)\oplus Á_{-1}$$

Az $Á_{+1} = 1$ akkor és csak akkor, ha 1-nél több 1-es van A, B, $Á_{-1}$ között, így

$$Á_{+1} = \overline{A}\cdot B\cdot Á_{-1} + A\cdot\overline{B}\cdot Á_{-1} + A\cdot B\cdot\overline{Á}_{-1} + A\cdot B\cdot Á_{-1} = A\cdot B + A\cdot Á_{-1} + B\cdot Á_{-1}$$

Ennek technológiai realizációját általában tranzisztor mátrix (7 sor + 3 oszlop)-okkal oldják meg.

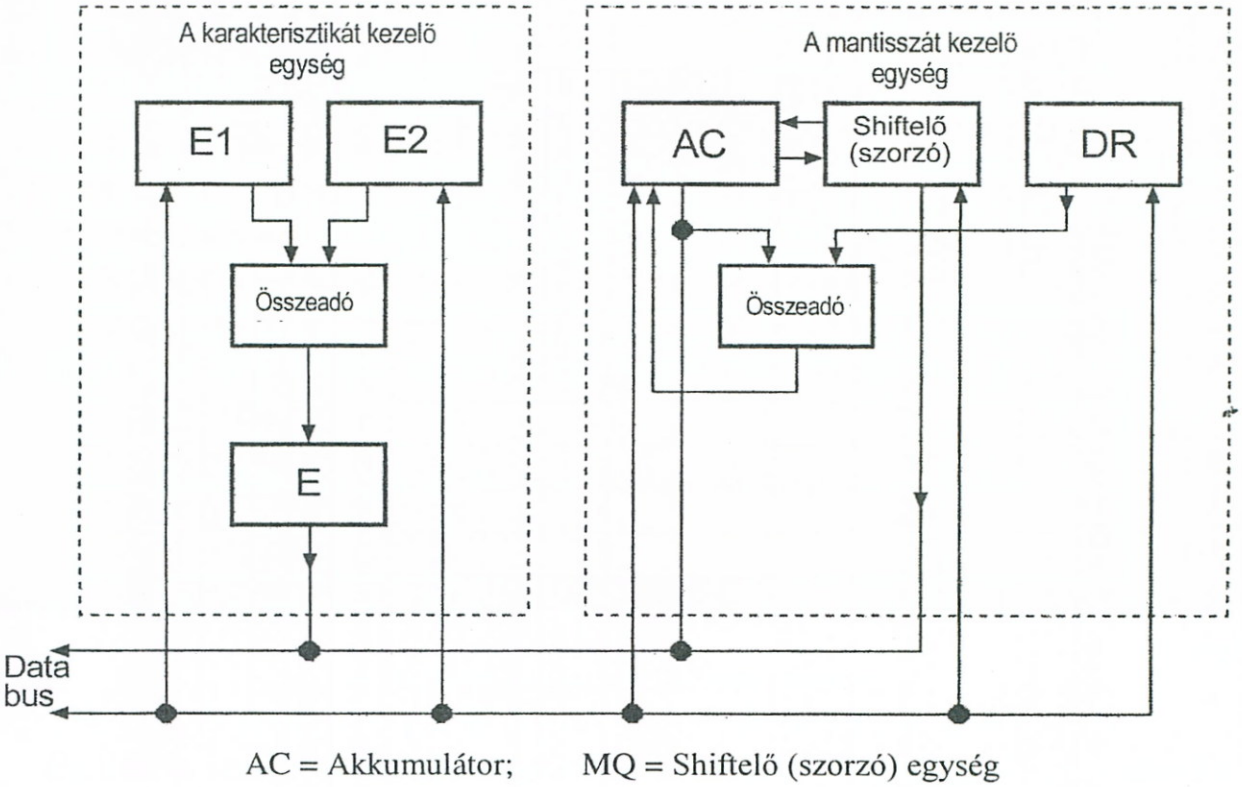
3.2.4. Az ALU áramköri felépítése

Az előbbiekben tárgyalt műveletvégrehajtási algoritmusokra is figyelemmel *az aritmetikai egység* lényegében a *következő logikai áramkörökből épül fel:*

- *összeadó egységekből, melyek alapeleme két 1 bites fixpontos adatelem összeadását végrehajtó áramkör,*
- *léptető áramkörökből, melyek az operandusokat tároló hardver regiszterek jobbra vagy balra léptetését hajtják végre,*
- *logikai műveleteket végrehajtó áramkörökből,*
- *az operandusokat illetve eredményeket tároló átmeneti tárolókból, melyek közül legfontosabb az aritmetikai egység akkumulátor regisztere.*

Az aritmetikai egység a műveleteket végrehajtva beállítja a jelző (FLAG) regiszter bitjeit, melyet a felhasználói illetve operációs rendszer programjai lekérdezhetnek és ezáltal információkhoz juthatnak a műveletek kimeneteléről.

A lebegőpontos ALU-nak a mantisszát és karakterisztikát áramköri szinten párhuzamosan kell kezelnie. A lebegőpontos műveletvégrehajtó egység blokksémáját a 37. sz. ábra ábrázolja.



37. sz. ábra
Az ALU lebegőpontos műveletvégző részének blokksémája

Megjegyzések a lebegőpontos ALU működéséhez

A 36. sz. ábra jelentősen leegyszerűsítve mutatja be a lebegőpontos ALU működését. Így például nem tartalmazza a 28. sz. ábrán bemutatott algoritmus hardver realizációját, a lebegőpontos regisztertárat stb.

A mikroszámítógépek esetében a lebegőpontos ALU a i80386-os processzorig, a processzortól fizikailag elkülönülő hardver egység volt (coprocesszor). az i80486-os processzortól viszont a lebegőpontos műveleteket végző áramköröket beépítik a processzorba.

3.3. MŰVELETI VEZÉRLÉS, A VEZÉRLŐEGYSÉG MŰKÖDÉSE

3.3.1. A műveleti vezérlés fogalma

A műveleti vezérlés részleteinek megértéséhez célszerű ismételten áttekinteni, hogy az egyes utasításokat milyen elemi lépésekben végzi el a processzor.

***1. lépés**: Utasításelőkészítés vagy lehívás.*

Ebben a fázisban a processzor az utasításszámláló (PC) tartalma alapján kikeresi a főtárból az utasítást és átviszi a vezérlőegység utasításregiszterébe (IR).

***2. lépés**: Az utasításszámláló regiszter tartalmának növelése.*

Az utasításszámláló regiszterhez hardver áramköri úton hozzáadásra kerül az aktuálisan végrehajtandó utasítás hossza, így az most már a következő végrehajtandó utasítás címét fogja tartalmazni.

***3. lépés**: Az utasítás dekódolása, műveleti kód és az utasításszerkezet értelmezése, az utasításban lévő operandusok címének számítása.*

***4. lépés**: A művelet végrehajtásához szükséges adatok kiolvasása a főtárból, előkészítése a végrehajtáshoz.*

***5. lépés**: A műveleti kód alapján értelmezett művelet végrehajtása az előkészített operandusokkal.*

(Vezérlésátadó utasítás esetén ebben a lépésben a következő utasítás címét írja be a processzor az utasításszámláló regiszterbe.)

***6. lépés**: A művelet eredményét a processzor beírja az utasításban előírt tárolóhelyre.*

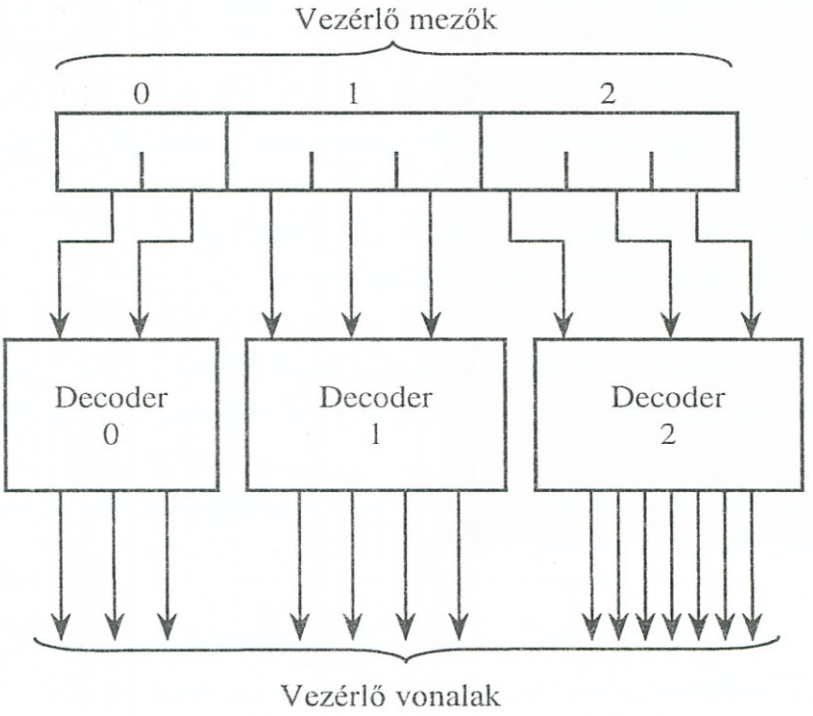
Ezeknek az elemi lépéseknek a végrehajtása az áramköri megvalósítás szintjén még további részekre bomlik fel. Így például az utasításlehívást (1 pont) részletezve:

- PC tartalom → memória címregiszterbe (MAR),
- a memória címregiszter tartalma alapján az utasítás kiolvasása a memóriából a memória adatregiszterbe (MDR),
- az utasítás átvitele az MDR-ből az IR-be.

Ezek végrehajtásához az adatútvonalak megnyitására, lezárására és például állapotjelzők beállítására van szükség. Mindezek összességét tekinthetjük a számítógép műveleti vezérlésének.

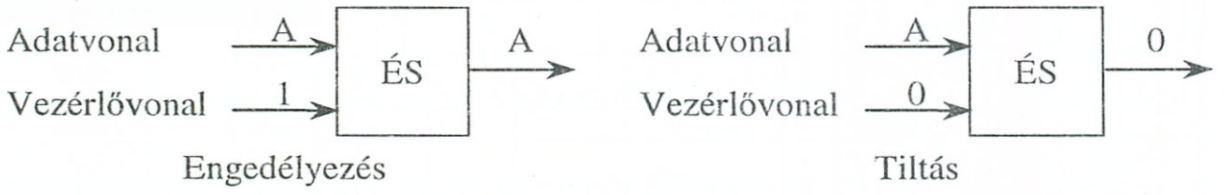
Ennek megfelelően műveleti vezérlésnek a gépi utasítások elemi lépéseinek végrehajtásához a számítógép összes, a műveletvégrehajtáshoz szükséges hardver részegységének a gépi utasítás alapján történő összehangolt irányítását nevezzük.

Az adatútvonalak lépésenkénti nyitása, zárása vezérlési pontokon valósul meg. Ekkor a vezérlési pontokhoz vezérlő mezők tartoznak és vezérlővonalakon keresztül jut el a vezérlő mezőkből a vezérlési információ a vezérlési pontokhoz (lásd 38. sz. ábra).



38. sz. ábra
A műveleti vezérlés vezérlő mezői és vonalai

A vezérlési pontokon az adatútvonal engedélyezését és zárását leegyszerűsítve egy „ÉS" kapuval szemléltethetjük:



3.3.2. A vezérlőegység vezérlő jelei és a vezérlési pontok

A vezérlőegység működése során vezérlőjeleket ad ki a teljes számítógép irányítására. Ezek lehetnek:

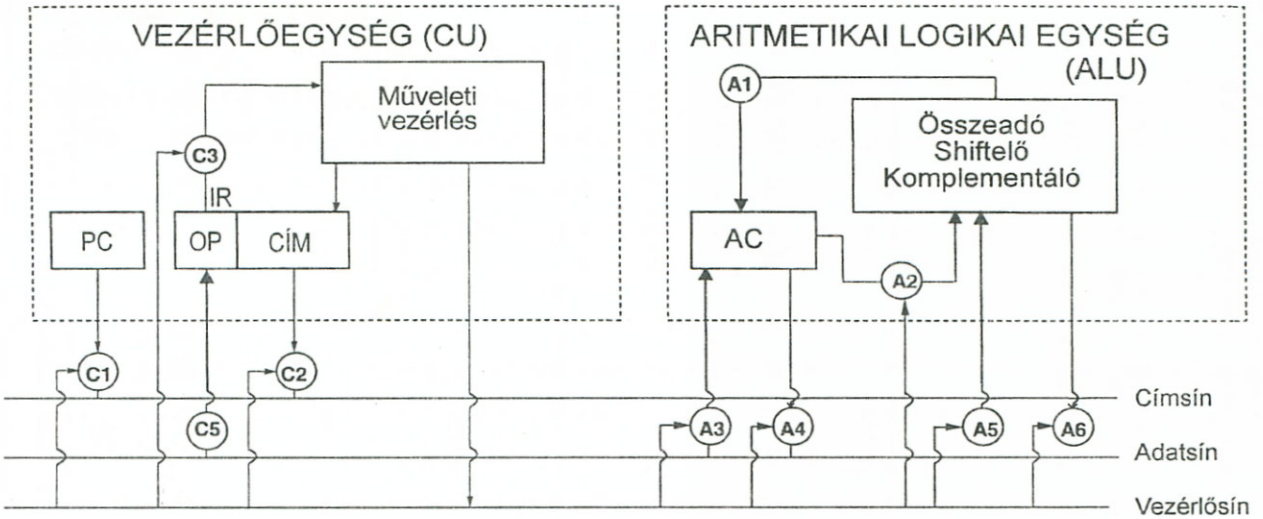
- *a processzor belső vezérlőjelei, melyek a processzoron belüli részegységek működését irányítják például az aritmetikai egység és a processzor regiszterei közötti adatátvitelt,*
- *a processzoron kívüli egységek irányítását szolgáló külső vezérlőjelek, melyek a processzor és a memória, a processzor és az input/output eszközök közötti adatátvitelt, illetve a megszakításkezelést és a sínvezérlést irányítják.*

Példák vezérlőjelekre:

Külső:
- MEMWR, I/O WR (memória és eszközpuffer írása)
- MEMRD, I/O RD (memória és eszközpuffer olvasása)

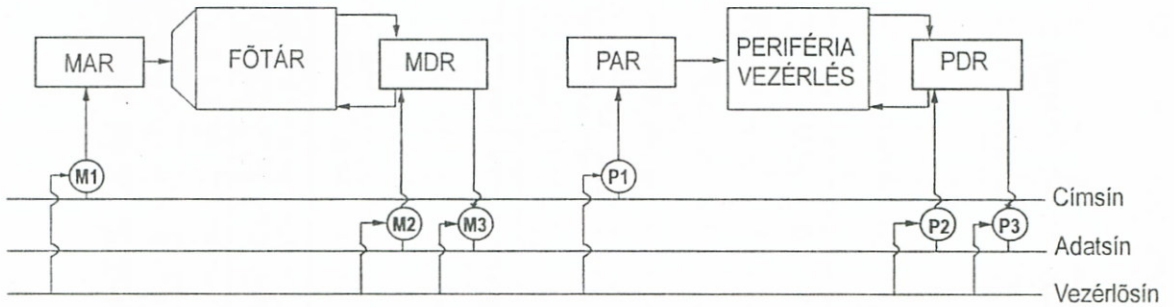
Belső:
- ALUOP, REG WR, REG RD (ALU művelet engedélyezése, regiszter írása, olvasása)

Az adat- és címvonalak nyitó/záró vezérlési pontjait egyszerűsített formában a 39. sz. ábra szemlélteti.



PC = Programszámláló regiszter
IR = Utasításregiszter
OP = Műveleti kód
AC = Akkumulátor

39/A. sz. ábra
Az adat és címvonalak vezérlési pontjai a processzoron belül



MAR = Memória címregiszter
MDR = Memória adatregiszter
PAR = Periféria címregiszter
PDR = Periféria adatregiszter

39/B. sz. ábra
Az adat és címvonalak vezérlési pontjai a processzoron kívül

Vizsgáljuk most meg egy egyszerűsített példán a műveleti vezérlésben érintett vezérlési pontokat egy memória, regiszter átviteli utasítás végrehajtása során.

Legyen a végrehajtandó utasítás MOV AX, 00A0h, azaz a 00A0h című memóriaterület kell átvinni az akkumulátorba. (A MOV utasítás a memóriában található, tehát először azt ki kell olvasni és dekódolni. Lásd 40. sz. ábra).

Elemi lépés	*Érintett vezérlési pontok*
Utasításszámláló tartalmának átvitele a sinen keresztül a memóriacímregiszterbe	C1, M1
A memóriaadatregiszter tartalmának (azaz az utasításnak) átvitele az utasításregiszterbe	M3, C5
Utasításdekódolás alapján a memóriacímregiszter feltöltése 00A0-al	C3, C2, M1
A memóriaadatregiszter tartalmának átvitele az akkumulátor regiszterbe	M3, A3

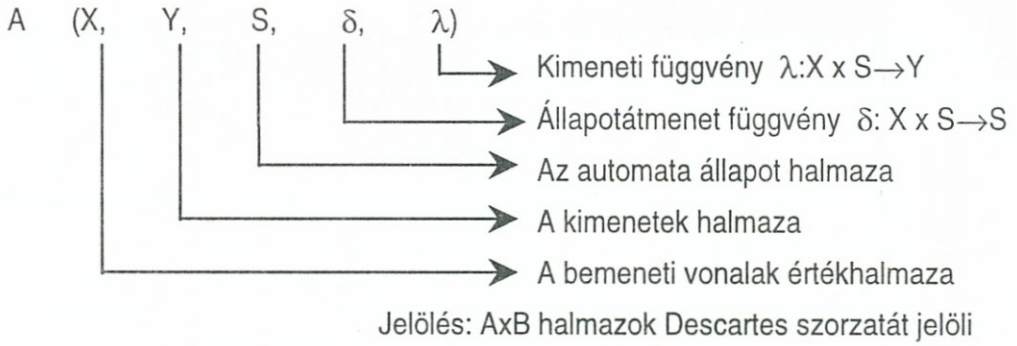
40. sz. ábra
Az utasítás végrehajtás során érintett vezérlési pontok

Az ábra szerint a MOV utasítás végrehajtása 4 elemi lépésben történhet meg. Minden egyes elemi lépés engedélyezéséhez (tiltásához) be kell állítani az érintett vezérlési pontokat. Ez kétféle módon történhet:

- a vezérlési pontokat áramkörökkel állítjuk be,
- a vezérlési pontokat a gépi utasítás kódja alapján a ROM tárolóból kiolvasott mikroprogram utasításai állítják be lépésről-lépésre.

Műveleti vezérlés modellezése

A műveleti vezérlés véges állapotú automatákkal (Huffmann modell FSM = Finite State Machine) modellezzük (lásd 41. sz. ábra).



41. sz. ábra
Véges állapotú automata

Az FSM egy meghatározott λ függvénnyel a bemeneti jel és állapota szerint állítja elő az automata kimenőjelét. Ehhez hasonlóan a bemeneti jel és az automata állapotának függvényében kerül meghatározásra az FSM következő állapota (δ függvény).

3.3.3. Huzalozott és mikroprogramozott műveleti vezérlés

A műveleti vezérlést a számítógépekben tehát kétféleképpen lehet megoldani:

- *huzalozott módon, azaz hardver áramkörökkel, melyek a processzorba beépítve irányítják a műveletvégrehajtás elemi lépéseit;*
- *mikroprogramozott módon, mely esetben az utasítás-végrehajtás elemi lépéseit egy (általában a firmware ROM tárolójában található) mikroprogram vezérli. Ebben az esetben a gépi kódú utasítás műveleti jelrésze (azaz a gépi kódú utasítás kódja) dekódoláskor egy mikroprogram lefutását kezdeményezi, mely a mikroprogramtárolóban található. Ennek a mikroprogramnak az utasításai sorban végrehajtva történik meg az utasítás elemi lépéseinek a vezérlése.*

Ez úgy is kifejezhető, hogy 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.*

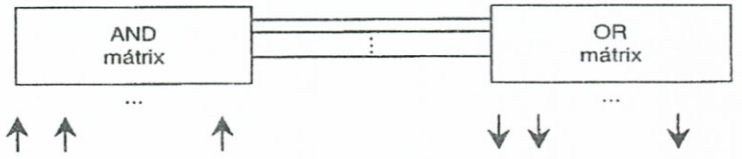
A mikroprogramozott műveleti vezérléshez szükséges adatokat, azaz a mikroutasításokat egy mikroprogram tárolóban helyezzük el. Ehhez szükséges:

- egy mikroutasításszámláló regiszter, mely mikroprogramutasítás címét tartalmazza
- mikroprogramutasítások, melyekkel
  - *a következő mikroutasítás címe,*
  - *a vezérlőjelek (kódolva vagy közvetlenül)*

előállíthatók.

A huzalozott műveleti vezérlést kombinációs, ill. sorrendi áramkörökkel valósítják meg. Ezek legtöbbször úgynevezett programozható áramkörök (PLA = Programmable Logic Array).

Áramköri megvalósításukat tekintve a PLA-k „ÉS" és „VAGY" áramkörökből felépített mátrixok (lásd 42. sz. ábra). Ennek matematikai alapját az adja meg, hogy minden Boole-függvény, így a véges állapotú automata λ és δ függvényei is előállíthatók diszjunktív normálformában.

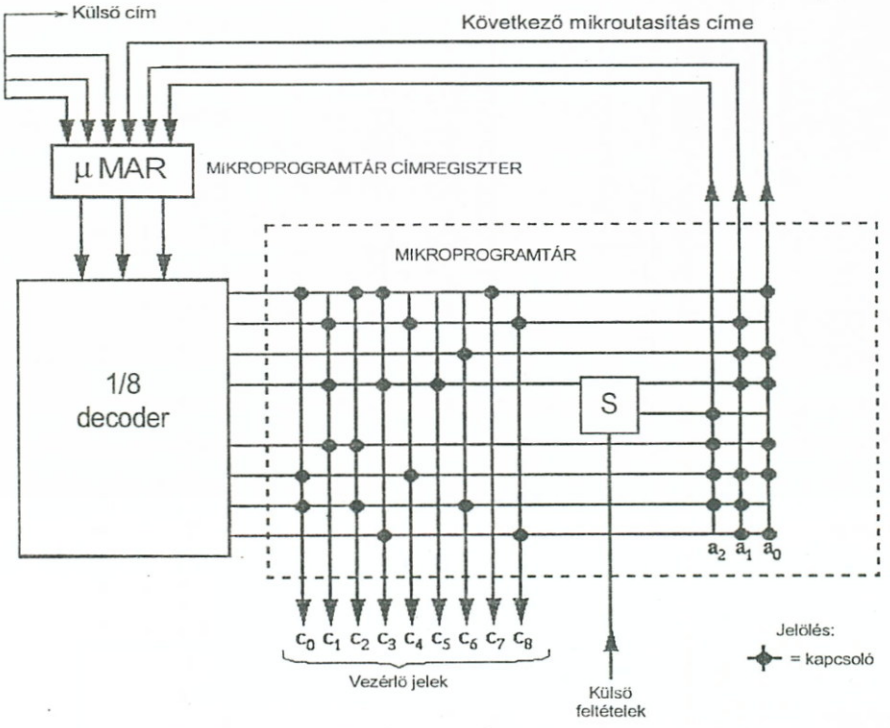


42. sz. ábra
A huzalozott műveleti vezérlés megvalósításának szemléltetése

3.3.4. A horizontális és vertikális mikroprogramozás

*A **horizontális mikroutasítás** egy vezérlési mezőt és a következő mikroutasítás címét tartalmazza (ha van ilyen). A vezérlési mező minden egyes bitje az áramkörök egy-egy vezérlési pontját állítja be.*

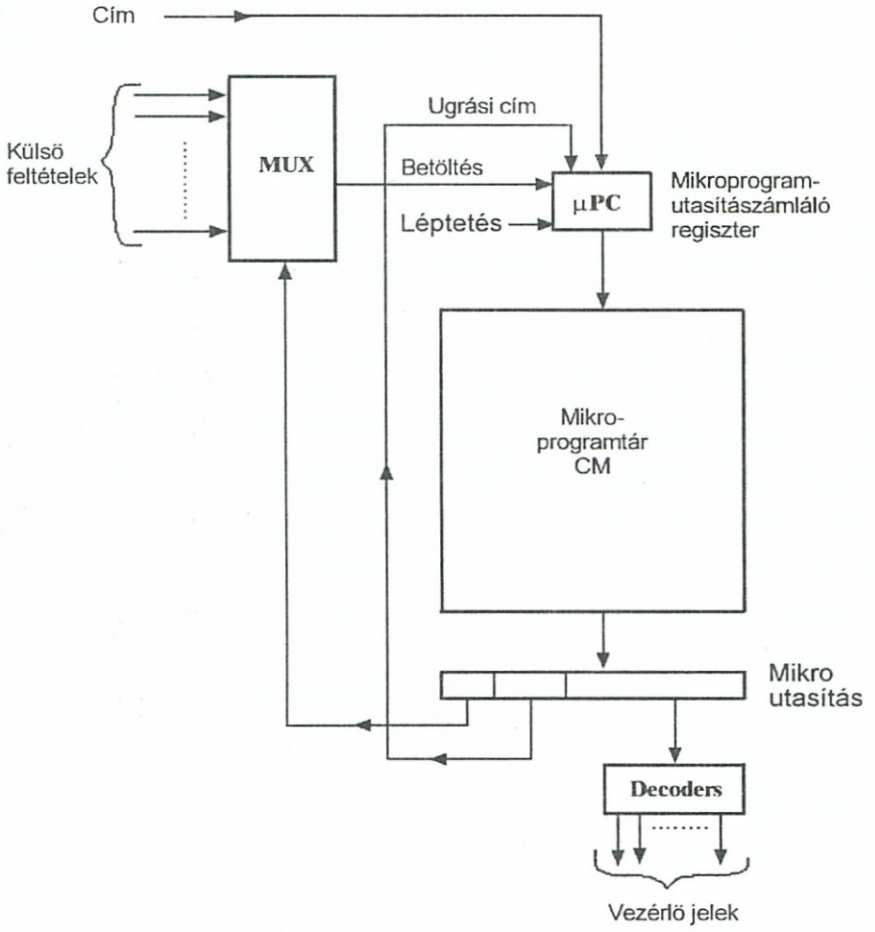
Ezt úgy lehet megvalósítani, hogy a műveleti vezérlést végző áramkörök ún. programozható logikával (PLA = Programmable Logic Array) rendelkeznek, azaz olyan ÉS illetve VAGY kapukból álló hálózatok, melyek működését a vezérlő bitekkel meghatározhatjuk (lásd 43. sz. ábra).



43. sz. ábra
A horizontális mikroprogramvezérlés vázlata

*A **vertikális mikroutasítás** lényegében úgy épül fel, mint egy gépi kódú utasítás.*

Ennek megfelelően a vertikális mikroprogramozásnál általában mikroutasítások sorozata kerül sorosan végrehajtásra (lehetnek például a mikroprogramban mikroszubrutinok is). Ehhez a mikro-programvezérlő és a mikroprogramtár mellett szükség van például mikroutasításszámláló regiszterre is és a mikroutasítások műveleti vezérlőrészét a gépi kódú utasításokhoz hasonlóan dekódolni kell (lásd 44. sz. ábra).



44. sz. ábra
Vertikális mikroprogramvezérlés vázlata

A mikroutasítások felépítése

A vertikális mikroutasítás felépítése hasonlít a gépi kódú utasítás szerkezetéhez:

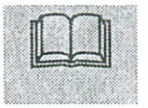
A mikroprogramtárra hivatkozó cím	Műveleti rész

Jellemzői: rövid mikroutasítások, dekódolni kell (lassúbb).

A horizontális mikroutasítás vagy a tényleges vezérlőbiteket tartalmazza, vagy speciális változata, a kódolt horizontális mikroutasítás esetén egy kódtáblázat egy sorának címét tartalmazza, ahol a vezérlőbitek megtalálhatók.

Vezérlési pontok bitjei	Következő utasítás

Jellemzői: hosszabb mikroutasítások.



Speciális eset: Intel Pentium Pro-tól kezdve az x86-os utasításokat felbontják RISC mikroutasításokra, melyek végrehajtása párhuzamosítható.

3.3.5. Mikroprogramvezérlés és a processzor architektúrája

A mikroprogramvezérlés és a processzor architektúrája között szoros összefüggés van:

- *a horizontális mikroprogramvezérlés (ennek szélsőséges esete a huzalozott műveleti vezérlés) a RISC processzorokra jellemző,*
- *a vertikális mikroprogramvezérlést viszont a CISC processzorokban alkalmazzák.*

Ez összefüggésben van azzal is, hogy a RISC processzorok csökkentett, egyszerűsített utasításkészlete nagyon hasonlít a CISC processzorok mikroprogramutasításaihoz. Emiatt a RISC processzorokat gyakorlatilag assembly nyelven nem szokták programozni. Ezért azt szokták mondani, hogy

MINDEN RISC PROCESSZOR ANNYIT ÉR,
AMILYEN HATÉKONY A
FORDÍTÓPROGRAM.

3.4. AZ UTASÍTÁSVÉGREHAJTÁS GYORSÍTÁSA PÁRHUZAMOSÍTÁSSAL (PIPELINING)

Mint már láttuk a számítógép fejlődéstörténetének meghatározó jellegzetessége volt az utasításvégrehajtás gyorsítására vonatkozó, állandóan növekvő igény.

A gyorsítás megoldható a gép működésének ütemezését meghatározó órajelfrekvencia növelésével, aminek viszont technológiai korlátai vannak. Ezért kerültek előtérbe azok a módszerek, melyek a számítógép gyorsabb műveletvégrehajtását rendszertechnikai eszközökkel biztosítják.

A rendszertechnikai gyorsítás egyik legfontosabb módszere az utasításvégrehajtás szintjén átlapolt feldolgozás, melyet pipeliningnek neveznek.

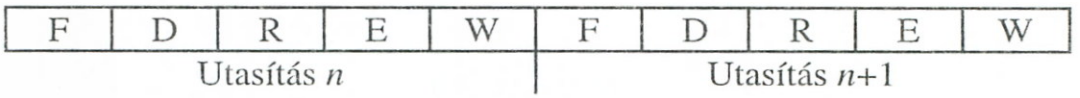
3.4.1. A pipelining lényege

A pipelining fogalmával már megismerhettük a 2.4.6. fejezetben. Most először a soros és a pipeline szervezésű utasítás végrehajtást hasonlítjuk össze.

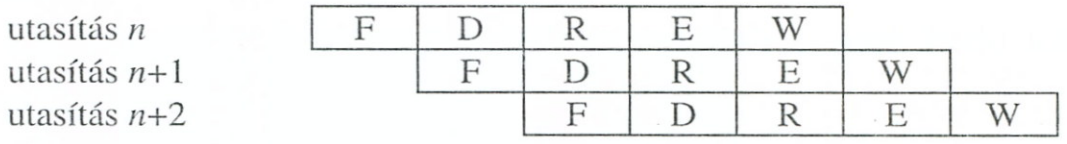
Tegyük fel, hogy a gépi utasítás elemi lépései a következők:

Utasításkiolvasás (F), dekódolás (D), operandus kiolvasás (R), végrehajtás (E), visszaírás (W). Ekkor a soros és a pipeline feldolgozás különbségeit a 45. sz. ábra mutatja be.

Soros feldolgozás



Pipeline feldolgozás

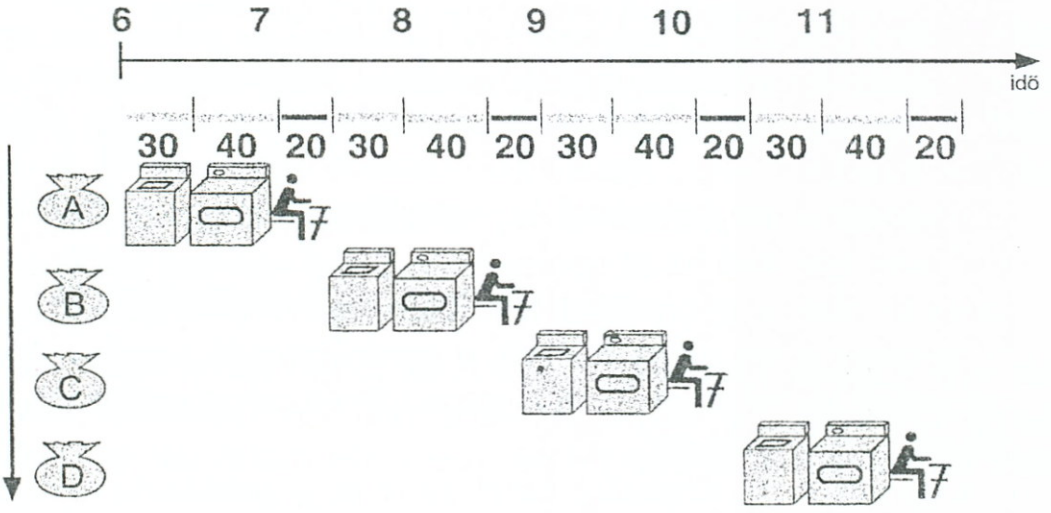


45. sz. ábra
A soros és a pipeline feldolgozás összehasonlítása

A pipelining lényegét (futószalag vagy szerelőszalag) egy gyakorlati példával is szeretnénk bemutatni. Legyen a végrehajtandó feladat a ruhatisztítás, melyet Attila, Béla, Csilla, Dávid végeznek. A ruhatisztítás három munkafázisból álljon:

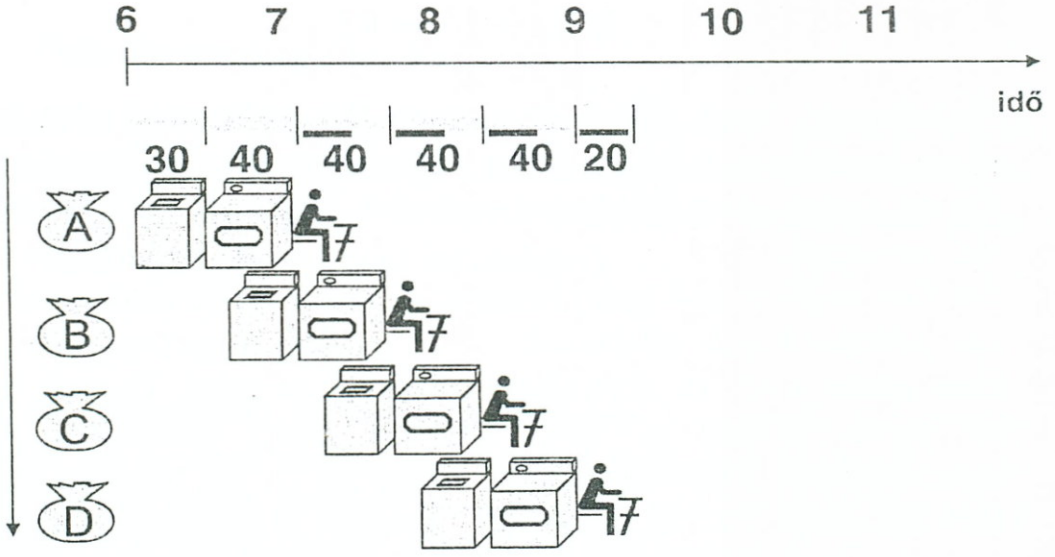
- mosás 30 perc
- szárítás 40 perc
- vasalás 20 perc.

Ekkor a soros munkavégzés időbeli lefutását a 46. sz. ábra mutatja be.



46. sz. ábra
Ruhatisztítás soros munkavégzéssel

A pipelining-gel okosabban lehet megszervezni a munkát (lásd 47. sz. ábra).



47. sz. ábra
Ruhatisztítás pipelining szervezéssel

Látható, hogy ugyanaz a feladat pipelining munkaszervezéssel 6 óra helyett 3,5 óra alatt végezhető el. Példánkban A, B, C, D megfelel az egymás után következő utasításoknak, a mosás, szárítás, vasalás a gépi utasítás elemi lépéseinek, a mosógép, szárító és vasaló pedig az elemi lépéseket egymástól függetlenül végrehajtó hardver erőforrásoknak.

3.4.2. A pipelining működés során fellépő problémák

Az előbbiekben a pipelining lényegét egy idealisztikus példán mutattuk be, mely a valóságban ilyen kristálytisztán sohasem megvalósítható. *A pipeline* szervezésben *megoldandó fontosabb problémák* a gyakorlatban a következők:

- *az utasítok elemi fázisainak végrehajtásához szükséges idő igen eltérő lehet* (például ha az operandus egy regiszterben található, akkor innen kb. tízszer gyorsabban lehívható, mint a főtár memóriájából),
- *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 utasításvégrehajtás 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:

$$1/\ R1 + R2 \Rightarrow R1$$
$$2/\ R1 + 10 \Rightarrow R3$$

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, vagy ún. „hazard"-nak (veszélyhelyzet) *nevezzük.*

- *hardver erőforrások igénybevétele során is előfordulhatnak ütközések, például buszkonfliktusok.*
  Erre példát a 45. sz. ábra alapján fogunk mutatni. Tegyük fel, hogy az ábrán látható n, n+1, n+2-vel jelölt utasítások és ezek operandusai a főtárban vannak. Ekkor az „n"-ik utasítás operandus kiolvasásához (R) és az n+2-ik, utasítás utasításkiolvasásához (F) egyaránt szükség van a memóriabuszra. E két buszművelet azonos időben nyilván nem hajtható végre.

3.4.3. A pipelining során fellépő problémák kezelésének módszerei

A 3.4.2. pontban bemutatott problémákat vagy sztatikusan, a program fordítása során vagy dinamikusan, azaz futás közben a hardverrel tudjuk kezelni. Módszerek:

- ***NOP utasítások** (fordítóprogram) **beiktatása a programba***

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

- ***Utasítás-átrendezés** (fordítóprogram)*

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)*

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.

- ***Data forwarding** (hardver)*

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).

- ***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 L1 szintű különálló adat és utasítás cache).

A vezérlésátadó utasítások kezelése

A vezérlésátadó utasítások kezelése a pipeline-ben 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.

Ennél *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:

- *sztatikus esetben a fordítóprogram értékeli ki az ugrási feltételeket és meghatározza a legnagyobb valószínűséggel előforduló ugrási címeket és ennek megfelelően szervezi a pipeline-t;*

- *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.*

A Pentium processzorcsalád korszerűbb tagjai a dinamikus előrejelzésre két gyors működésű cache tárolót használnak a processzorba beépítve:

- a BTB (Branch Target Buffer) tartalmazza a más végrehajtott ugróutasítások statisztikái szerint számított elágazási valószínűségeket,
- az RSB (Return Stack Buffer) a szubrutinból való visszatérési címeket tartalmazó gyorsítótár.

A sztatikus és dinamikus előrejelzések hasznosságát akkor érthetjük meg, ha például a ciklusutasításokat (melyek a programokban nagyon gyakoriak) vizsgáljuk. Például egy 100-as ciklus esetén az ugrás 99 esetben a ciklus elejére fog megtörténni, így ennek dinamikus és sztatikus előrejelzése is megfelelő hatékonysággal megtörténhet.

3.5. SZUPERSKALÁR PROCESSZOROK

Az elmúlt évtizedekben a processzorok fejlődésében három jól elkülönülő szakasz különböztethető meg:

- A hagyományos, Neumann elvű, soros utasítás feldolgozású, processzorok az 1980-as évekig uralták a processzorok piacát. Ezekre a szekvenciális utasításkibocsátás jellemző, azaz az utasításpufferből az utasítások sorban egymás után kerülnek átadásra az utasítás végrehajtó egységeknek.
- A RISC processzorokkal jelentek meg az utasítás szinten párhuzamos működésű ILP processzorok, (ILP = Instruction Level Parallel), melyek már pipelining szervezésűek. Szintén szekvenciális utasításkibocsátással működnek, azaz ezek még mindig skalár processzorok.
- Több pipeline szervezésű, párhuzamos működésre képes végrehajtó egységet tartalmazó processzorok az 1990-es években jelentek meg. Ezek lehetőségeit kihasználandó a szekvenciális utasításkibocsátást helyettesíteni kellett párhuzamossal. Ennek két módszere van:
  - VLIW (Very Large Instruction Word) architektúrák, több műveletet tartalmazó utasításokat bocsátanak ki,
  - a dinamikusan ütemezett szuperskalár processzorok, melyek ciklusonként több utasítás végrehajtására képesek, így ennek megfelelően több utasítást bocsátanak ki ciklusonként (A 90-es évekre a 4X-es kibocsátás a jellemző)

A szuperskalár processzorok tervezése során több speciális feladatot is meg kellett oldani, melyekkel a skalár processzoroknál még nem találkozhattunk. Ezek közül a fontosabbak:

- *a párhuzamos dekódolás, előrekódolás;*
- *a szuperskalár utasításkibocsátás* (ez azért kritikus feladat, mert a magasabb kibocsátási ráta felerősítheti a vezérlési és adatfüggőségek teljesítménykorlátozó hatásait). Az ehhez tartozó korszerű eljárások:
  - *utasítás várakoztatás,*
  - *regiszterátnevezés,*
  - *spekulatív elágazás-kezelés;*
- *párhuzamos utasításvégrehajtás*, melynek megszervezésekor azt a feladatot kellett megoldani, hogy az utasítások befejezési sorrendje eltérhet a soros utasításvégrehajtás befejezési sorrendjétől.

A szuperskalár utasításkibocsátás ötletét már az 1970-es években publikálták, de ennek az architektúrának pontos elméleti kidolgozására csak az 1980-as évek közepén került sor. Az 1980-as évek második felében kísérleti mintaprojektekben fejlesztették ki az első szuperskalár processzorokat. A kereskedelmileg forgalmazott első szuperskalár processzorok (POWER PC, RS/6000 1990, ALPHA, SUPER SPARC 1992) RISC architektúrájúak voltak.

A szuperskalár CISC processzorok csak jóval később jelentek meg a piacon (PENTIUM, MC 68060 1993). Ennek oka nagyobb bonyolultságukban keresendő. A szuperskalár CISC processzorok megtervezéséhez ugyanis meg kellett oldani:

- *a változó hosszúságú utasítások és*
- *a többfajta memóriautasítás*

kezelését.

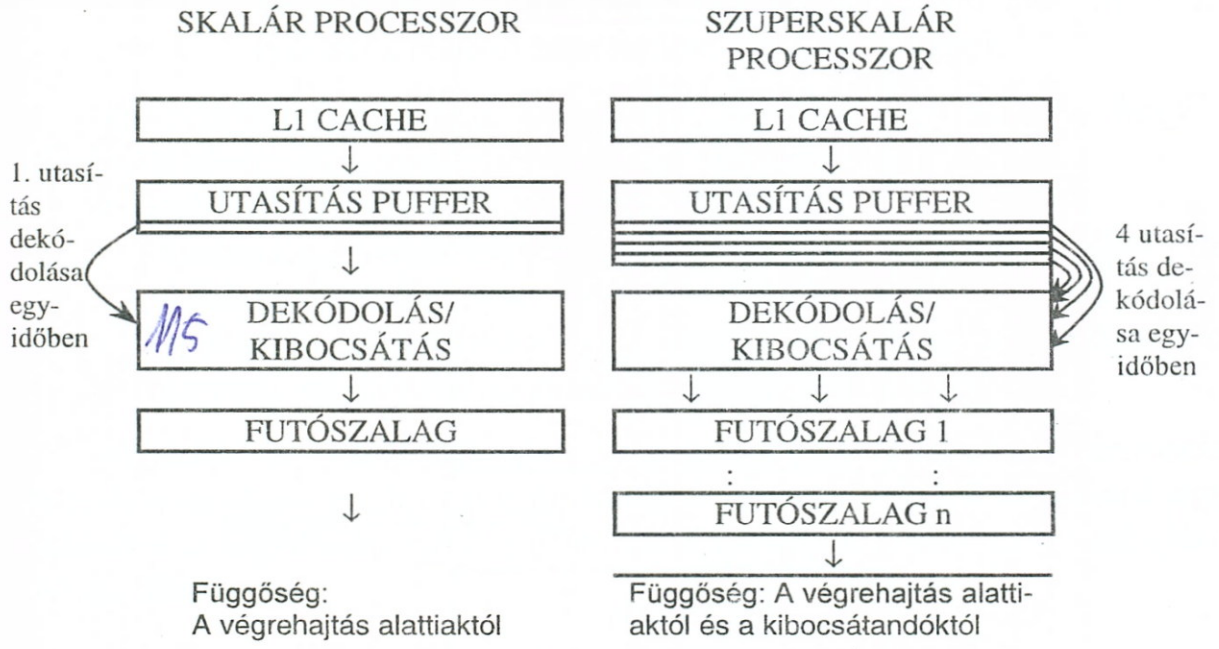
E problémára megoldást a x86-os utasításkészlet RISC emulációja jelentett, azaz a szuperskalár CISC processzorokat (Pentium Pro, Nx586, K5 stb.) egy szuperskalár RISC maggal valósítják meg. Ez azt jelenti, hogy ezek a processzorokban az x86-os utasításokat „RISC" jellegű mikroutasításokra bontja fel a processzor. Így többé már nem volt akadálya a szuperskalár RISC processzorokra jellemző architekturális megoldások átvételére. *Ezekben a processzorokban a RISC mag általában 4X kibocsátású, mely x86-os utasításokban 2X vagy 3X-os (Pentium Pro) kibocsátást jelent.*

Az elkövetkezendőkben a szuperskalár feldolgozást jellemző rendszertechnikai megoldásokat tekintjük át.

3.5.1. Párhuzamos dekódolás

Nyilvánvalóan 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.

Az ezzel kapcsolatos feladatok elemzéséhez hasonlítsuk össze egy skalár és egy 4X kibocsátású szuperskalár processzor működését (lásd 48. sz. ábra).



48. sz. ábra
A skalár és szuperskalár processzor dekódolása és utasításkibocsátása

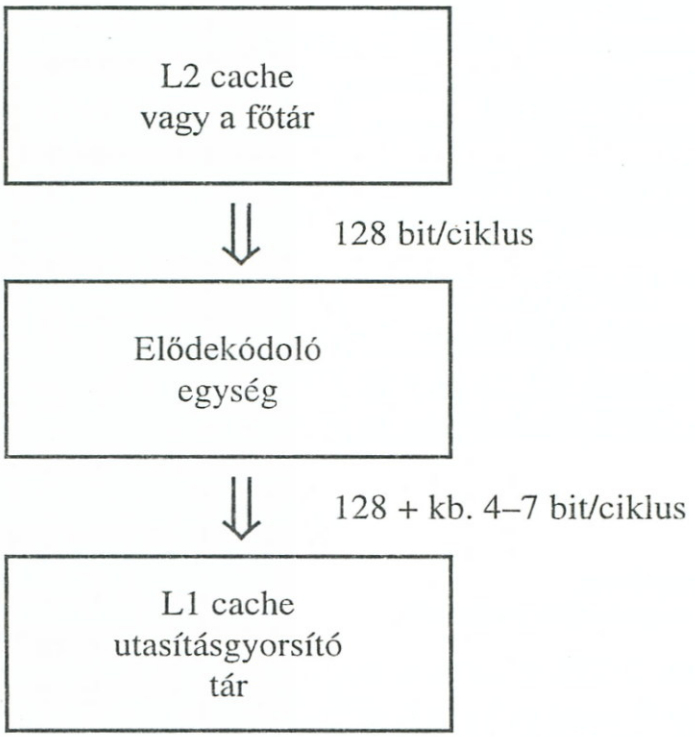
Az ábráról leolvashatók a különbségek:

- 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:*
  - *a 4 kibocsátásra váró utasítás függ-e a végrehajtás alatt állóktól,*
  - *a 4 kibocsátásra váró utasítás között vannak-e függőségek.*

Emiatt a szuperskalár processzorok esetében a függőségvizsgálatok jóval nagyobb feladatot jelentenek, mint a skalár processzoroknál.

Ez természetesen időt igényel, ezért *a processzor dekódolási feladatait az előkódolással próbálják meg csökkenteni. Ennek elvét a 49. sz. ábra mutatja be.*



49. sz. ábra
Az elődekódolás elve

Az elődekódolás során a dekódolás feladatainak egy része már akkor végrehajtásra kerül, amikor az utasításokat a másodlagos gyorsító tárból vagy a memóriából az L1 szintű gyorsítótárba írják. Ennek során ún. elődekódoló bitekkel egészítik ki az utasítást (pl. POWER PC 620 7 bit, AMD K5 bájtonként 5 bit).

Ezek a bitek CISC processzoroknál meghatározhatják például a műveleti kód helyét, az utasítás kezdetét és végét stb.

3.5.2. A szuperskalár utasításkibocsátás és várakoztatás

A szuperskalár utasításkibocsátásnak két összetevője van:

- *A kibocsátási ráta adja meg a processzor által ciklusonként kibocsátható vagyis a végrehajtó egységekhez továbbítható utasítások számát.*
- *A kibocsátási politika határozza meg a függőségek kezelését. Ezek lehetnek:*
  - *Adatfüggőségek a hivatkozott regiszterek között,*

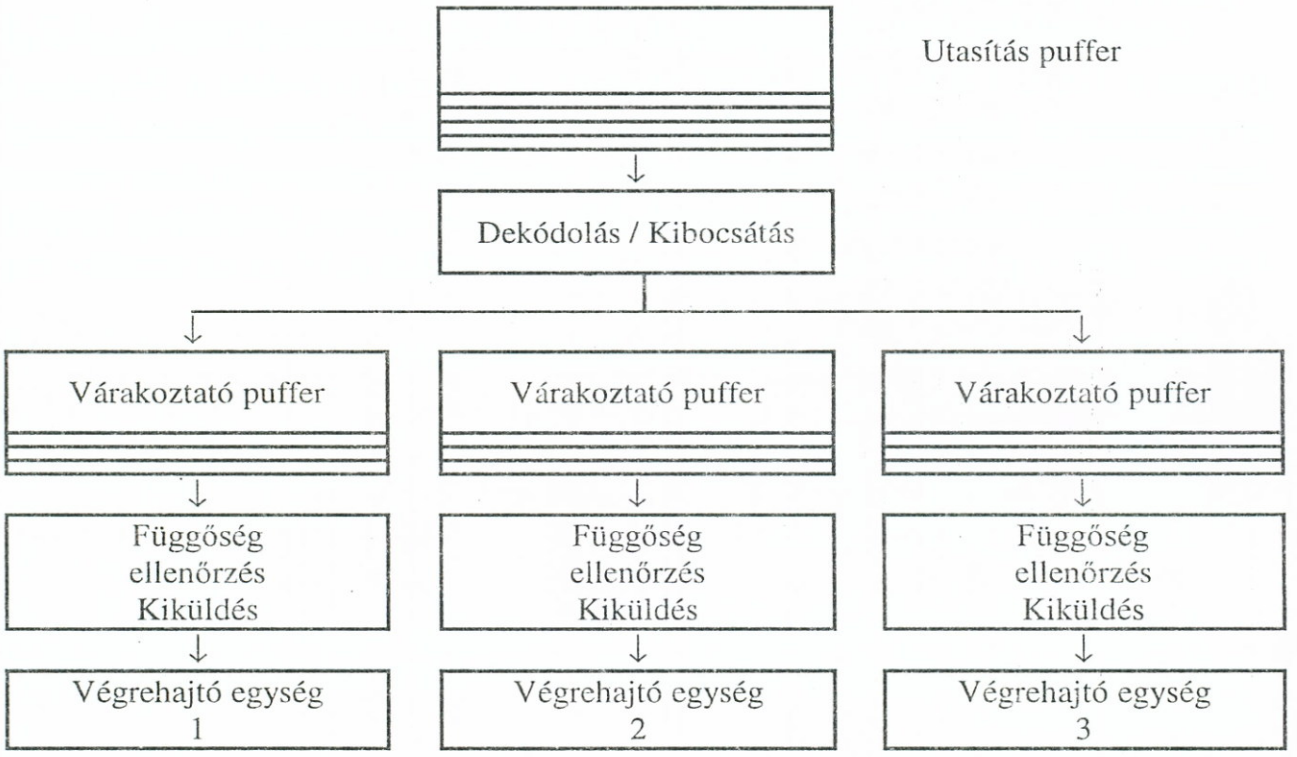
- *Vezérlésfüggőségek* (feltételes vezérlésátadó utasítások, melyeknél a feltétel kiértékelése nincs befejezve).

Az adatfüggőségeket a processzor kezelheti oly módon, hogy az áladatfüggőségeket regiszterátnevezéssel megszünteti.

A vezérlésfüggőségeket a processzor kezelheti:

- *blokkolással, azaz a processzor felfüggeszti az utasítás kibocsátást a feltétel kiértékeléséig;*
- *spekulatív elágazáskezeléssel, azaz a processzor valamilyen módszerrel, megbecsüli az elágazási utasítás kimenetét.*

Az adat- és vezérlésfüggőségek okozta blokkolódásokat a processzor kezelheti az utasításkibocsátás blokkolásával, vagy a függő utasítások pufferelésével a végrehajtó egység előtt, amit várakoztatásnak nevezünk (lásd a 50. sz. ábra).



50. sz. ábra
A várakoztatás alapelve

Várakoztatás esetén az utasítás a várakozó pufferben vár mindaddig, míg a megelőző utasítások végrehajtása feloldja a függőséget. Ekkor az utasítás kibocsátás alatt az utasítások várakozó pufferbe írását, és kiküldésnek a függőségmentes utasításnak a várakozó pufferből a végrehajtó egységekhez való továbbítását értjük.

Példák:

- PENTIUM:
  - nincs regiszter-átnevezés;
  - blokkolás.
- PENTIUM PRO (és az ezt követő processzorok):
  - regiszter-átnevezés;
  - várakoztató puffer (20 utasítás kapacitással).

3.5.3. A regiszter átnevezés

A regiszter átnevezés célja az ál-adatfüggőségek megszüntetése. Ál-adatfüggőségek jöhetnek létre például abból a helyzetből, hogy a programozónak rendelkezésére álló regiszterkészlet jóval kisebb annál, mint ahány regiszter a processzorba fizikailag beépítésre kerül.

Például az IBM kompatíbilis PC-k Assembly nyelvében 4 általános regiszter = EAX, EBX, ECX, EDX használható, ezzel szemben a Pentium II processzor 40 felhasználható általános regisztert tartalmaz. *Emiatt a programozó (fordítóprogram) esetenként – a címezhető regiszterek hiánya miatt – például a verembe történő mentésre kényszerül, amire a fizikailag rendelkezésre álló regiszterek száma miatt nem lenne szükség.*

Tekintsük például a következő utasítássorozatot:

```
MOV     EAX, 100
ADD     [mem1], EAX
MOV     EAX, 200
ADD     [mem2], EAX
```

Ezek – mivel az EAX regisztert használják – függnek egymástól. Ha megfelelő számú regiszter (R1, R2) van a processzorban, akkor ezt kiküszöbölhetjük:

```
MOV     R1, 100
ADD     [mem1], R1
MOV     R2, 200
ADD     [mem2], R2
```

A regiszter átnevezéssel tehát a processzor a függőséget okozó utasítás célregiszterét egy tartalékregiszterrel helyettesíti, amelyben tárolható a függőséget okozó utasítás eredménye.

A regiszter átnevezés megvalósítása szintén lehet

- *sztatikus (fordítóprogram hajtja végre);*
- *dinamikus, amikor az átnevezést futás közben a hardver hajtja végre. Ez utóbbi a jellemző a korszerű processzorokra.*

!

A regiszterátnevezés során a processzor egy leképezőtáblázattal hozzárendeli az architekturális regisztertárat az átnevező puffertárhoz.

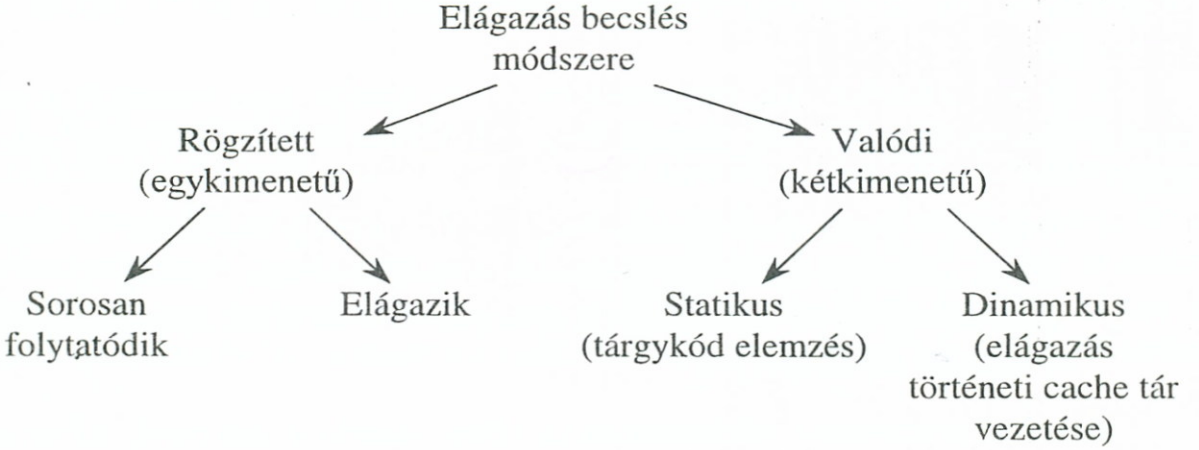
Az átnevezés folyamata a következő lépésekkel valósul meg:

- *az utasításkibocsátáskor a cél- és forrásregiszter átnevezése az átnevező puffertárral;*
- *a kiküldéskor a regiszter érvényességi bitek azaz a függőségek ellenőrzése;*
- *a végrehajtás után az átnevező puffertár aktualizálása.*

!

3.5.4. A processzor spekulatív elágazás-feldolgozása

Spekulatív elágazásfeldolgozásnak nevezzük azt az eljárást, amikor a processzor – a pipelining zavartalan működése érdekében – valamilyen módszerrel megkísérli az ugró utasítások kimenetét megjósolni. Ennek lehetséges változatait az 51. sz. ábra mutatja be.



51. sz. ábra
Az elágazás becslés módszerei

Láthatjuk, hogy *a feltételes elágazási utasítások lehetséges kimeneteinek becslése történhet:*

!

- *egykimenetűen, amikor a becslés rögzítetten mindig azonos (elágazik vagy nem);*

- *kétkimenetűen, melynél sztatikus esetben a fordítóprogram a tárgykód elemzése alapján, vagy dinamikus esetben a program futása során a processzor az egyes elágazások megvalósulásának statisztikája alapján „jósolja" meg az elágazást.*

3.5.5. Párhuzamos végrehajtás

Ha a processzor az utasításokat párhuzamosan hajtja végre, akkor az egyes végrehajtó egységek az utasítások eredményeit az eredeti utasítássorrendtől eltérően is előállíthatják. Ezt csak úgy lehet kezelni, ha az eredményadatok átmenetileg tárolódnak és végleges helyükre az eredeti utasítássorrendnek megfelelően kerülnek csak beírásra.

Ezért fontos:

- *a programutasítások soros végrehajtásának helyreállítása (processzor konzisztencia),*
- *a memória-hozzáféréseknek az eredeti utasítássorrend szerinti végrehajtása (memória konzisztencia).*

A processzor és memória konzisztenciát együttesen soros konzisztenciának nevezzük.

A soros konzisztencia biztosításának legfontosabb eszköze a szuperskalár processzoroknál az átrendező puffertár, a ROB (ROB = Re Order Buffer).

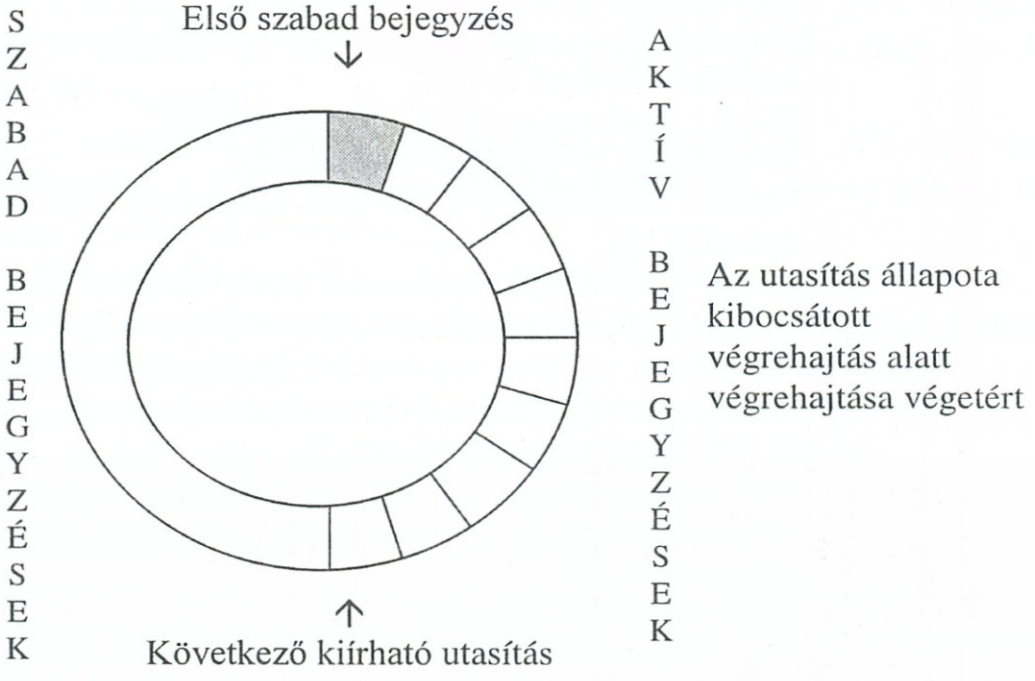
A ROB működésének megértéséhez figyelembe kell vennünk a párhuzamos utasításvégrehajtást. Emiatt meg kell különböztetnünk, hogy *egy utasítás*

- *végetért, azaz a megfelelő végrehajtó egység befejezte az utasítást és az eredményadat egy átmeneti tárolóba került, illetve*
- *befejezett, ha az eredményt a processzor beírta az utasításban előírt végleges tárolóhelyre.*

Mivel a processzor párhuzamosan több *utasítást* dekódol, kibocsát és végrehajt, ezek *egy időpillanatban a következő állapotokban lehetnek:*

- *dekódolt,*
- *kibocsátott* (várakozási pufferbe beírt)
- *végrehajtás alatti,*
- *végetért,*
- *befejezett.*

A soros konzisztencia megőrzéséhez a processzornak nyilván kell tartani az utasítások állapotát. Ezt a ROB-ba beírt bejegyzések biztosítják (52. sz. ábra).



52. sz. ábra
A ROB működési elve

A ROB segítségével eldönthető, hogy a processzor egy utasítás eredményadatait mikor írhatja be az átmeneti tárolást biztosító regiszterből az utasításban előírt végleges tárolóhelyre. Ez csak akkor történhet meg, ha a kérdéses utasítást a program eredeti sorrendjében megelőző összes utasítás befejeződött (ez azonos azzal, hogy a ROB-ban törlésre került). *Ekkor a vizsgált utasítás befejezettnek nyilvánítható és a ROB-ból törölhető (kiírható).*

3.5.6. Esettanulmány: a Pentium-II processzor

Technikai jellemzők:

- Processzor mag 7,5 millió tranzisztor.
- Belső órajel 233, 266, 300, 350 és 400 MHz.
- Külön utasítás és adatcache (L1) egyenként 16 KB, 4-utas csoport asszociatív (lásd 4.3. fejezet).
- 256, 512 KB L2 cache (kb. 31 millió tranzisztor).
- Rendszerbusz 66 és 100 MHz-es órajellel.
- Adatbusz szélessége 64 bit, címbusz szélessége 32 bit (ellenőrző-bitekkel 36 bit).

A processzor a x86-os gépi utasításokat elemi, RISC utasításokhoz hasonló mikroutasításokra bontja fel. Ezeket szuper-skalár processzorként egymással párhuzamosan is képes végrehajtani. Szuper pipeline architektúra, azaz a gépi utasításokat a processzor 14 elemi lépésre bontja fel.

További fontosabb szuperskalár jellemzők:

- Külön buszrendszer az adat és utasítás cache-hez (Dual Independent Bus). Az L2 cache busza a processzor órajelének felezésével működik.
- Szuper pipeline 14 fokozattal:
  - előrendezés (8 fokozat): utasításkihozatal a cache-ből, x86-os utasítások felbontása RISC mikroutasításokra, regiszter átnevezés, ROB írás,
  - végrehajtás (3 fokozat): mikroutasítások továbbítása a végrehajtóegységek várakoztató puffereihez, a mikroutasítások végrehajtása,
  - visszaírás (3 fokozat): a mikroutasítások eredményeinek rendezése a ROB szerint, a ROB, az x86-os regiszterek és a memória aktualizálása.
- A RISC mikroutasításokhoz 11 végrehajtó egység, melyek egyenként 5 utasítás kapacitású várakoztató pufferrel rendelkeznek.
- A spekulatív elágazásfeldolgozást a BTB (Branch Target Buffer) segíti, melynek kapacitása 512 bejegyzés. Minden bejegyzés egyrészt az ugrási címet, másrészt az ugrás bekövetkezésére vonatkozó számlálót (4 bit) tartalmaz.

ELLENŐRZŐ KÉRDÉSEK ?

69. Sorolja fel a processzor legfontosabb architekturális építőelemeit!
70. Mi a feladata a vezérlő egységnek és mit tartalmaz a PC és az IR?
71. Mi a feladata az aritmetikai egységnek és mit tartalmaz az AC és a FLAG regiszter?
72. Mi a feladata a busz interfész (BIU) a címszámító (AU) egységnek?
73. Definiálja a Pentium processzorcsalád valós üzemmódját, fejtse ki hogy miért van erre szükség!
74. Definiálja a Pentium processzorcsalád védett üzemmódját!
75. Mit jelent a Pentium processzorcsalád védett-valós és rendszermenedzselő üzemmódja?
76. Hogyan csoportosíthatók a processzor regiszterei?
77. Mit értünk a gépi utasítás szerkezete alatt?
78. Milyen részekből épül fel egy gépi utasítás?
79. Definiálja az utasításkészlet fogalmát!
80. Sorolja fel az utasításkészlet fontosabb utasítás típusait!
81. Mikor kerülnek beállításra az állapot (flag) regiszter fontosabb jelzőbitjei (carry, zero, sign, overflow)?
82. Ismertesse a vezérlő utasításokat!
83. Milyen adattípusokat dolgoz fel az ALU?
84. Milyen szempontok szerint értékeljük a processzorok tárolókezelő rendszerét?
85. Definiálja a megszakítás és a kivétel fogalmát!
86. A fixpontos számok milyen formában kerülnek tárolásra a számítógépen?
87. Ismertesse, hogy a fixpontos alapműveleteket milyen módon lehet visszavezetni összeadásra és shiftelésre!
88. Hogyan ábrázoljuk és milyen esetekben használjuk a lebegőpontos számokat?
89. Mit tartalmaz az IEEE 754-es szabvány?
90. Milyen pontossággal lehet a lebegőpontos számokat ábrázolni?
91. Az IEEE 754-es szabvány szerint mit jelent a denormatizált, a végtelen szám és a „nem szám”?
92. Hogyan lehet visszavezetni a lebegőpontos műveleteket fixpontos műveletekre?
93. Mit nevezünk BCD kódnak és hogyan hajtjuk végre a műveleteket BCD kódú számokkal?
94. Ismertesse az MMX aritmetikai műveleteket!

95. Indokolja meg, hogy miért hajtható végre az összes aritmetikai és logikai művelet kapuáramkörökkel!
96. Milyen áramkörökből épül fel az ALU?
97. A gépi utasítást milyen elemi lépésekben hajtja végre a processzor?
98. Mit nevezünk műveleti vezérlésnek?
99. Miért van szükség a műveleti vezérléshez vezérlő mezőkre és vonalakra?
100. Hogyan csoportosíthatók a vezérlőegység vezérlőjelei?
101. Mit nevezünk huzalozott és mikroprogramozott műveleti vezérlésnek?
102. Milyen processzorokra igaz az a kijelentés, hogy „számítógép működik a számítógépen belül" és ez mit jelent?
103. Mi a horizontális mikroprogramozás lényege?
104. Mi a vertikális mikroprogramozás lényege?
105. Milyen összefüggés van a mikrogrogramvezérlés és a processzor architektúrája között?
106. Hasonlítsa össze a soros és a pipeline utasításfeldolgozást!
107. Mutassa be a pipelining működés során fellépő problémákat!
108. Mit jelent a pipelining utasításvégrehajtásban a NOP utasítások beiktatása és a data forwarding?
109. Mit jelent a pipelining utasításvégrehajtásban az utasításátrendezés és a scoreboarding?
110. Hogyan lehet kezelni az utasításolvasás és adatolvasás/írás ütközéseit a belső sínen?
111. Mit jelent a pipeline leállítása illetve törlése a vezérlésátadó utasítások kezelésében?
112. Mit jelent a vezérlésátadó utasítások sztatikus és dinamikus kezelése pipelining feldolgozásnál?
113. Mit jelent a processzor spekulatív elágazás feldolgozása?
114. Milyen fontosabb problémákat kell kezelni a szuperskalár CISC processzszorok tervezésekor, és ezekre mi a megoldás?
115. Hasonlítsa össze a skalár és a szuperskalár processzorok utasításdekódolását és kibocsátását!
116. Miért van szükség szuperskalár processzoroknál előredekódolásra és ennek végrehajtása hogyan történik?
117. Mit jelent a szuperskalár utasításkibocsátási ráta és kibocsátási politika?
118. Milyen kezelési módjai vannak szuperskalár processzoroknál az adat és vezérlésfüggőségeknek?
119. Hogyan kezelheti az adat és vezérlésfüggőségeket a processzor? Mit nevezünk várakoztatásnak?
120. Ismertesse a regiszterátnevezés célját, megvalósítását és lépéseit!

121. Spekulatív elágazás feldolgozás esetén milyen módszerei vannak az elágazás becslésének?
122. Mit jelent a soros konzisztencia megőrzése szuperskalár processzoroknál?
123. Milyen állapotban lehetnek a programutasítások szuperskalár feldolgozásnál?
124. Mi a ROB és segítségével hogyan dönthető el, hogy egy utasítás befejeződött-e?

IV.

TÁROLÓKEZELÉS

4.1. A TÁROLÓKEZELÉS ALAPFOGALMAI

A számítógép különböző típusú adattároló-egységei a processzor mellett meghatározó jelentőségűek a hatékony műveletvégrehajtásban.

A tárolószervezésnek a legfontosabb elve: azokat az adatokat, melyekre a műveletvégrehajtás során gyakran van szükség, a processzor „közelében", gyors elérésű tárolókon kell elhelyezni.

Elérési idő: A memória megcímzése után az adatnak a tároló kimenetén való megjelenéséig eltelt idő.

Ciklus idő: Elérési idő + feléledési idő , azaz az az idő, amelynek elteltével egy címzés után a memória újra megcímezhető.

A nagyon gyors elérés drágább hardver elemeket igényel, ezért ezeket a tárolókat csak kisebb kapacitású egységekben lehet a számítógépekbe beépíteni.

Emiatt *a számítógépben lévő adattároló-egységek felépítése hierarchikus,* melyet az jellemez, 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.* A tárolóhierarchia egyes szintjén lévő tárolók hozzáférési idejét és tárolókapacitásuk nagyságrendjét mutatja be az 53. sz. ábra.

Tárolótípus	Regiszter-tárak	Gyorsítótár (CACHE)	Főtár	Háttértár (pl. mág-neslemez)
Jellemző hozzáférési idő	$2 \cdot 10^{-9}$	$4 \cdot 10^{-9}$	$20 \cdot 10^{-9}$	$10 \cdot 10^{-3}$
A tárolókapacitás nagyságrendje	10^2 bájt	10^4 bájt	10^7 bájt	10^9 bájt
	Növekvő adathozzáférési idő →			
	Növekvő tárolókapacitás →			
	Csökkenő tárolási költség →			

53. sz. ábra
Tárolótípusok, hozzáférési idejük és tárolókapacitásuk

A processzor tartalmazza a regisztereket, melyek egyenként általában 32, 16 bit 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 – már két nagyságrenddel nagyobb a regisztereknél, ugyanakkor az adatok kiolvasása kb. 2-szer annyi időt igényel, mint a regisztertáraknál.

A főtár – mely az aktuálisan végrehajtott programot és ennek adatait tárolja – már kb. 10-szer lassúbb a regisztereknél, ugyanakkor viszont adattárolóképessége 100.000-szer nagyobb.

A programvégrehajtáshoz aktuálisan nem szükséges adatokat tároljuk a háttértárolókon, ezek általában mágneslemezek. Ezek tárolókapacitása 10 milliószor nagyobb a regisztereknél, viszont 5 milliószor lassúbbak.

Az előzőekben megadott számértékek nagyságrendjét elég nehéz közvetlenül érzékelni, ugyanakkor csak ezek ismeretében érthetjük meg igazán a számítógépek architektúráját és működését. Ezért példaként tegyük fel, hogy a regiszterek hozzáférési ideje 1 másodperc és számítsuk ki, hogy ez esetben milyen időt igényel 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 kapjuk:

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

Ebből a táblázatból a következő következtetéseket lehet levonni:

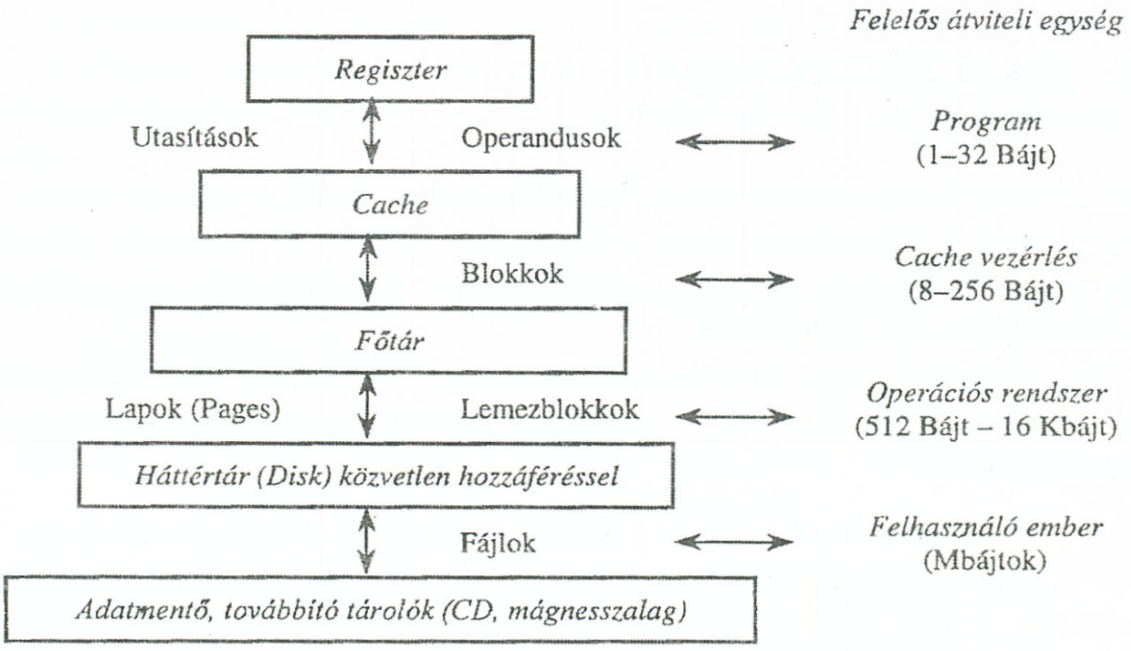
- ha a processzornak a főtárból kell adatot kiolvasni, akkor már komoly várakozásokra kényszerül;
- ha a processzornak a műveletvégrehajtáshoz olyan adatra van szüksége, melyet a háttértárolóról kell kiolvasni, akkor már „elviselhetetlen" ideig kell várakoznia;
- egy lassú periféria műveletvégrehajtási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

Ezért a háttértárakat és a lassú perifériákat önálló vezérlőegységekkel és átmeneti puffer (buffer) tárolókkal 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 műveletvégrehajtásához szükséges adatokat, összehangolva a tárolóhierarchia egyes szintjein lévő memóriaegységek működését.*

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

A tárolóhierarchia egyes szintjei közötti adatáramlás egységeit, és ezek mozgatását végző egységeket mutatja be az 54. sz. ábra.

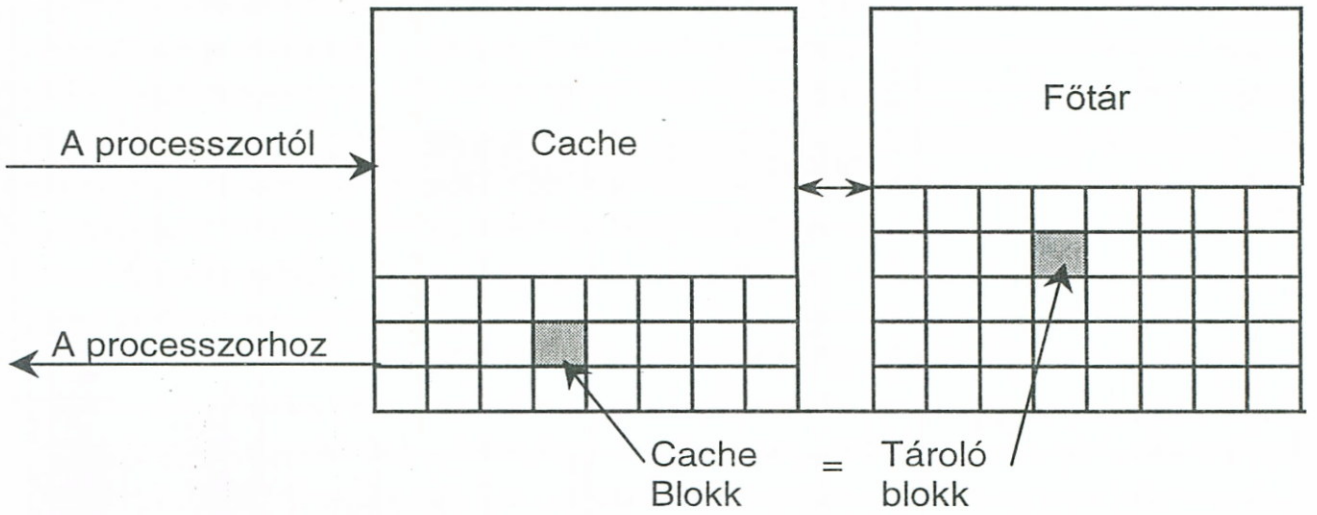


54. sz. ábra
A tárolóhierarchia szintjei közötti adatáramlás

Az ábra alapján a következő fontosabb következtetéseket vonhatjuk le:

- *Az adatok csak egymásra következő tárolóhierarchia szintek között mozgathatók egy lépésben (Nem lehet „kihagyni" egy szintet).*
- *Az egymásra következő tárolóhierarchia szintek között mozgatható adatok egysége azonos.*

Erre a gyorsítótár esetében mutat példát az 55. sz. ábra.



56. sz. ábra
Adatmozgatás a főtár és a cache között

4.2. 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.

Így alakultak ki a felhasználói programok (taszkok) által használható regisztertárak (elsőként a RISC processzorokban), melyek fontosabb típusai a következők:

- *a regiszterbank* (register banking) *esetében a regisztertömb fix méretű* (azaz meghatározott darabszámú regiszterből álló) *részekre van felosztva, melyek a felhasználói taszkokhoz kerülnek hozzárendelésre;*

- *az ablaktechnika* (register windowing) *esetében 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 blokktechnikánál* (register blocking) *a regisztertömb változó méretű átlapolható részekre van felosztva.*

Látható, hogy az ablaktechnika és a blokktechnika a különböző taszkok közötti, regiszterekben történő paraméterátadást segíti.

Fel kell hívni arra a figyelmet, hogy egyes *szuperskalár architektúrájú processzorok esetében a programok által látható (megcímezhető) regiszterkészlet a regiszterátnevezés technikájának alkalmazása miatt nem azonos a processzorba fizikailag beépített architekturális regiszterkészlettel.* Így például a Pentium Pro processzor 40 fizikai regisztert tartalmaz a regiszter átnevezéshez, de a programok az Intel Assembly-nek megfelelő ((EAX, EBX stb.) regiszterkészletet címzik.

4.3. GYORSÍTÓ (CACHE) TÁRAK

A számítógéparchitektúrák tervezése során az egyik legfontosabb megoldandó kérdés a tárolóhierarchiában elhelyezkedő, különböző sebességű adattároló eszközök közötti adatforgalomnak oly módon történő biztosítása, mely lehetővé teszi a processzor folyamatos, várakozás nélküli működését.

Ennek a célnak az elérését segítik:

- a processzor és a központi memória között, illetve
- a központi memória és a háttértárolók között

elhelyezett átmeneti tárolók, melyeket gyorsító tárnak vagy cache-nek nevezzük.

4.3.1. A processzor és a központi memória között elhelyezkedő cache tárak

4.3.1.1. A cache tárolók alkalmazásának szükségessége, a cache fogalmának meghatározása

A cache tár szerepének lényegét a következő gondolatmenet alapján érthetjük meg. Ha a processzornak korábban van szüksége egy adatra a központi tárból, mint azt a memória adni tudná, akkor a processzor ún. várakozó

állapotba kerül. 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.

Ahhoz például, hogy egy 66 MHz-es Pentium várakozás nélkül képes legyen a központi tár adatait olvasni 15 nsec-nál kevesebb elérési idejű memóriákra lenne szükség. Ilyen sebességet a központi memória DRAM áramköreivel már nem lehet elérni, ez a sebességtartomány az SRAM áramkörökre jellemző (a memóriatípusokat részletesebben a 4.3. fejezetben fogjuk vizsgálni). Ebből adódhatna a következtetés, hogy építsük fel a teljes központi memóriát SRAM áramkörökből. Ez viszont igen drága megoldás lenne, mivel egy azonos kapacitású DRAM áramkörhöz képest egy SRAM áramkör kb. tízszer annyiba kerül. Az előző problémára tehát az áthidaló megoldást a processzor és a főtár közé behelyezett gyorsabb, de drágább SRAM áramkörből felépített és főtárnál kisebb kapacitású puffertároló, a cache jelenti, ahová bemásoljuk a főtár blokkjait.

Az eddigieket összegezve, a cache tárolókat a következőképpen definiálhatjuk:

A cache vagy más néven gyorsító tárolók az utasítások és adatok átmeneti tárolására szolgáló, viszonylag kisméretű, gyors memóriapufferek, melyek a főtár blokkjainak másolatát tartalmazzák, önálló vezérléssel rendelkeznek és a felhasználó számára láthatatlanok (elérhetetlenek). A cache fogalmának meghatározásában *blokk alatt a főtár rekeszeinek* olyan *egymás utáni sorozatát értjük, melynek bemásolása a cache-be egy lépésben megtörténhet.*

A cache vezérlő egy „intelligens" áramkörökből felépített hardver egység. (Intelligencia alatt ez esetben az értendő, hogy például a vezérlő képes a különböző bemásolási és adatvisszaírási stratégiák, algoritmusok kezelésére.)

4.3.1.2. Az L1 és L2 cache és a cache memóriakapacitása

A cache tárolót közvetlenül beépíthetik a processzorba, vagy elhelyezhetik a processzoron kívül is.

Mikroszámítógépeknél a processzorba épített (on-chip) cache tárolót L1 (Level 1)-nek, azaz első szintűnek nevezik, míg a processzoron kívülit (off chip) L2 (Level 2) cache-nek hívják.

Az előző megállapításunk általánosan igaz, de a szabályt erősítő kivétel a Pentium Pro processzor, mely az L2 cache tárolót is tartalmazza. Ez az architekturális megoldás nem vált be.

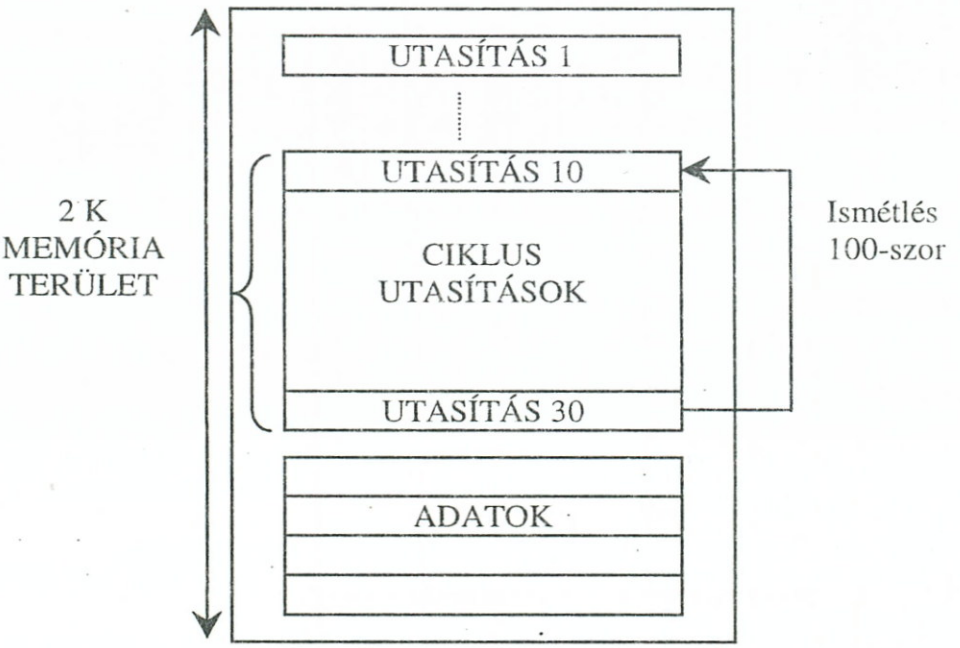
A cache tár kapacitása és a cache tárolóval gyorsítható főtárrész memóriakapacitása között is szoros összefüggés áll fenn. Ezt mutatja a következő táblázat:

Cache tároló kapacitása kbájtben	Gyorsítható főtároló memóriakapacitás Mbájtban
64	16
128	32
512	64

(A táblázat példaadatokat tartalmaz, az összefüggés különböző cache vezérlők esetén jelentősen eltérhet a táblázatban megadottaktól.)

4.3.1.3. A programozás és a cache összefüggései

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.



57. sz. ábra
A programlokalitás elve

Ez azt jelenti, hogy ha például az ábrán látható 2 K memóriablokkot bemásoljuk a cache-be, akkor a processzor több száz esetben innen tudja gyorsan lehívni az utasításokat és adatokat és nem szükséges a lassúbb, központi memóriához fordulnia. Az ún. objektumorientált programozás terjedésével a programoknak a fenti tulajdonsága tovább erősödött.

A példa alapján *megfogalmazhatjuk a program lokalitás elvét:*

A programok egy kis időintervallumban a címtérnek, memóriának csak egy relatíve kis részét veszik igénybe (Ez a programutasításokra és az ezek által felhasznált adatokra is igaz.). Ennek két oldala van:

- *Időbeli lokalitás:*
  Ha egy adatra vagy egy utasításra hivatkozás történik, akkor ez nagy valószínűséggel rövid időn belül ez újra megtörténik. (Ciklus)
- *Helyi lokalitás:*
  Ha egy adatra vagy egy utasításra hivatkozás történik, akkor ez nagy valószínűséggel a környezetében lévő címekre is megtörténik (gondoljunk a soros utasításvégrehajtásra).

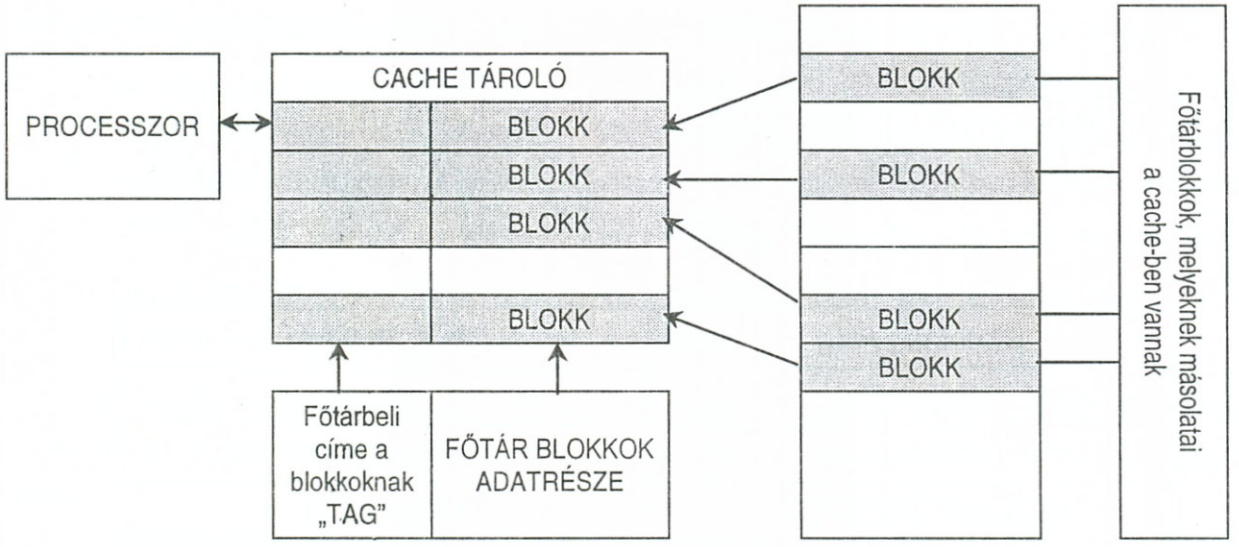
4.3.1.4. A cache működésének elve, cache hit és miss

Ha a processzor a központi tár egy rekeszét olvasni akarja, akkor a gyorsító tár vezérlője bemásolja ezt és még néhány rekeszt a cache tárolóba (lásd az 57. sz. ábra blokkjait). *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 gyorsítótárban rendelkezésre állnak.*

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 főtár és a cache-tár adategyezőségének biztosítása.*

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



58. sz. ábra
A főtárblokkok bemásolása a cache-be

Cache hit

Ha processzor olyan adatot igényel, mely a cache-ben megtalálható, akkor találatról vagy cache hit-ről beszélünk. A találatot a cache-vezérlő 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.*

Cache miss

Ha a processzor által igényelt adat nincs meg a cache-ben, ezt tévesztésnek vagy cache miss-nek nevezzük.

A találatok és tévesztések aránya határozza meg a cache teljesítményét. Az ezzel összefüggő legfontosabb adatok a következők:

HIT–RATE:	A tárolóhozzáféréseknek az a százaléka, amikor az adat a cache-ban megtalálható
MISS–RATE:	100–(HIT–RATE)
HIT hozzáférési idő:	Az adatok hozzáférési ideje a cache-ban (HIT-HF)
MISS javítási idő:	A hit/miss meghatározásának ideje + adatblokk kiolvasás a főtárból + adatblokk átadási ideje a processzornak (MISS-JAV)

137!

A processzor szempontjából a fenti adatok ismeretében az *adatokhoz való átlagos hozzáférési időt a következőképpen határozhatjuk meg:*

$$(\text{HIT}-\text{HF})\frac{[100-(\text{MISS}-\text{RATE})]}{100}+\frac{\text{MISS}-\text{RATE}}{100}\cdot(\text{MISS}-\text{JAV})$$

A cache tárolók jellemző értékeit tartalmazza az 59. sz. ábra.

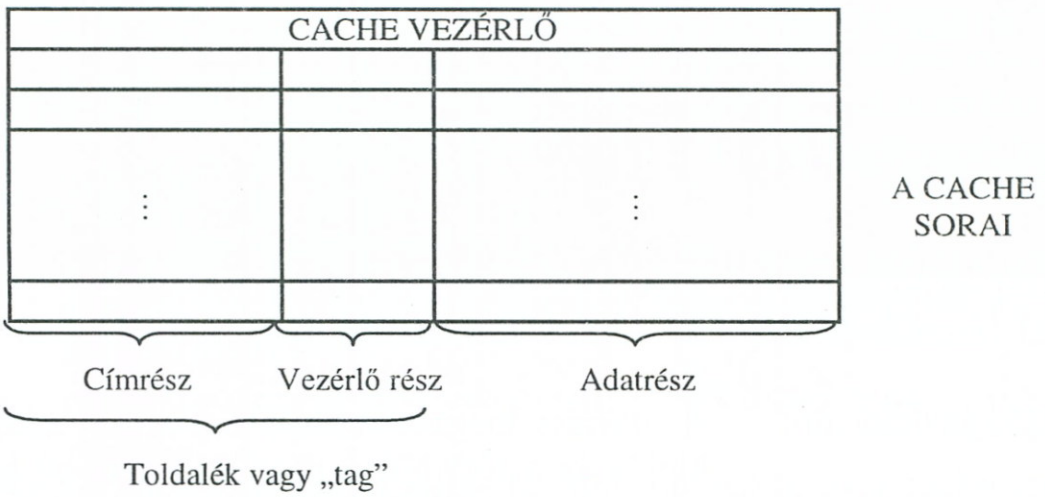
Blokknagyság	16–128 Bájt
HIT–HF	1–4 órajel
MISS–JAV	8–100 órajel
HOZZÁFÉRÉSI IDŐ	6–60 órajel
ÁTVITELI IDŐ	2–40 órajel
MISS–RATE	0,5–10%
CACHE NAGYSÁG	8 Kbájt – 4 Mbájt

59. sz. ábra
A cache tárolók jellemző értékei

! *Látható, hogy a találatok és tévesztések százalékos aránya a cache méretétől és a cache vezérlőtől függően általában 90% feletti érték.*

4.3.1.5. A cache tárak felépítése és típusai

A cache tárak általános felépítését a 60. sz. ábra mutatja be.



60. sz. ábra
A cache tárak általános felépítése

Látható, hogy a cache-be lemásolt főtárblokkok a cache soraiban kerülnek elhelyezésre. Minden sor két részből épül fel:

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

A cache tárak többsége részben, vagy teljesen asszociatív tárolóként működik. Ezért először röviden jellemezzük az asszociatív tárolók működését.

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.

Ez alapján arra gondolhatnánk, hogy ha az asszociatív tárolókban az adatok minden címzési eljárás nélkül visszakereshetőek, akkor célszerű volna a számítástechnikában általánosan ilyen előnyös tulajdonságú tárolóeszközöket használni. Természetesen az asszociatív tároló nem csodaszer, mivel a tárolt adatok mennyiségével arányos számú összehasonlító áramkör kell ahhoz, hogy a tartalom szerinti visszakeresés megvalósítható legyen. Ez pedig rendkívül drágává teszi ezeket az eszközöket.

Az asszociatív tárakat alkalmazó cache tárolók esetében az adatok, melyek tartalma szerint keresünk vissza, a főtárbeli blokkok címei, vagy részcímei.

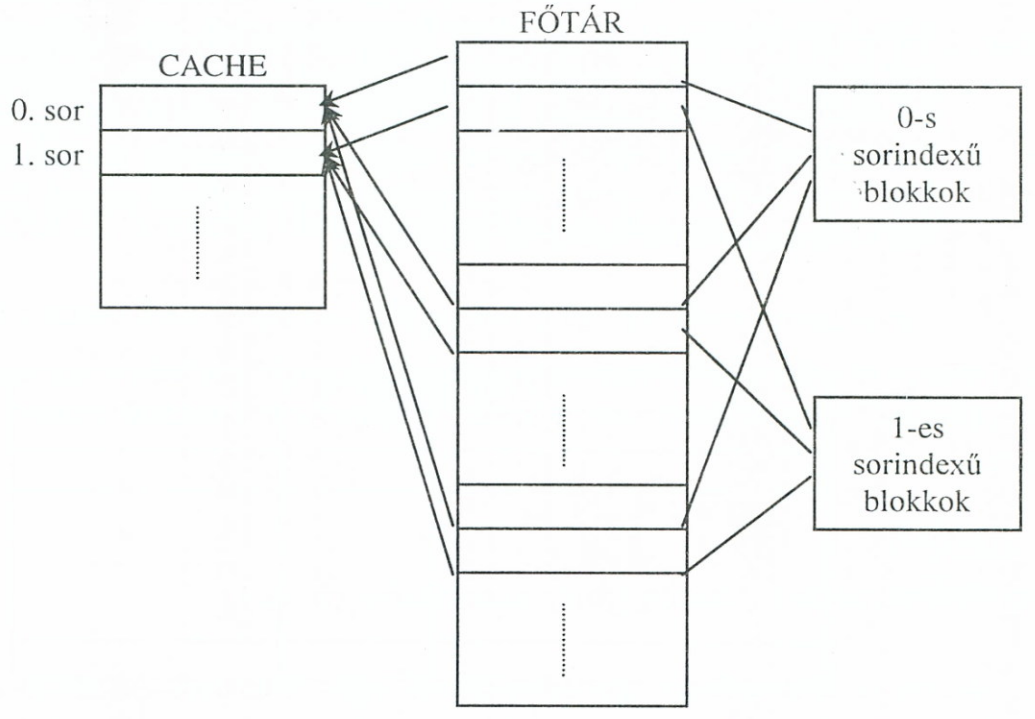
***Teljesen asszociatív cache** (fully associative cache)*

A teljesen asszociatív 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. Ehhez annyi összehasonlító áramkörre van szükség, amennyi sort tartalmaz a cache. Emiatt ezt a cache típust nagyon gyorsnak és egyúttal nagyon drágának jellemezhetjük.

Közvetlen leképezésű *(direct mapping)* **cache**

A közvetlen leképezésű 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 cacheben 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 61. sz. ábra).



61. sz. ábra
A közvetlen leképezésű cache-ben a főtár blokkok elhelyezése

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

n-utas csoport asszociatív cache *(n-way set associative cache)*

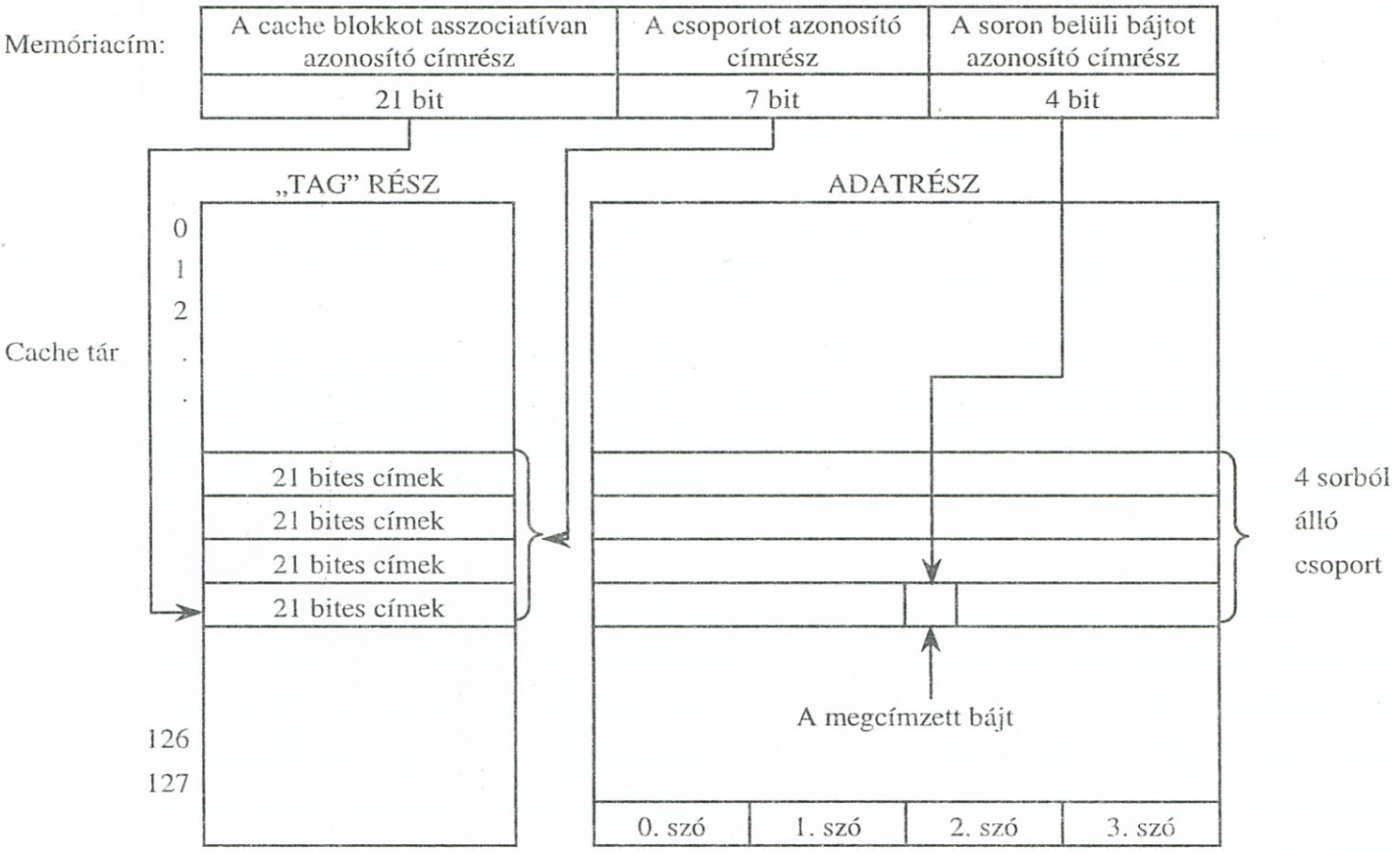
Ez a cache tárolótípus tulajdonképpen átmenetet képez a teljesen asszociatív és a közvetlen leképezésű cache tárolók között.

Az „n"-utas csoport asszociatív gyorsító táraknál a cache több, „n" sorból álló részre, csoportokra van osztva. Az egy csoporthoz tartozó cache tárolórész önmagában teljesen asszociatív tárolóként működik, azaz egy cache-be másolandó blokk a csoporton belül az „n" sor bármelyikébe bekerülhet.

Annak megállapítása, hogy egy cache írandó blokk melyik csoporthoz tartozik, a közvetlen leképezésű cache tárolóknál megismert módon, a memóriacímből képzett index-el kerül meghatározásra. Ezért az „n" utas csoport asszociatív cache tárolót úgy is elképzelhetjük, mint egy közvetlen leképezésű cache-t, amelyen belül (csoportszinten) több teljesen asszociatív cache tároló működik.

A csoport asszociatív cache tároló működését egy konkrét példán mutatjuk be. Tegyük fel, hogy egy 32-bites processzorunk van, ekkor a memóriacímek is 32-bitből állnak. A példánkban szereplő cache mérete legyen 8 Kbájt, egy sorának adatrésze tartalmazzon 16 bájtot, és egy csoporthoz tartozzon 4 sor. Példánkban tehát egy 4-utas csoport asszociatív cache-t fogunk vizsgálni. Mivel a sor adatrészében 16 bájt található, ezért a soron belüli bájtok címzésére 4 bitet ($16 = 2^4$) kell felhasználni a 32 bites memóriacímből. Egy sor 16 bájtból áll és 4 sor tartozik egy csoporthoz, ezért a 8 Kbájtos cache-ben 128 csoport található. (128 x 4 x 16 bájt = 8 Kbájt) Így a csoportot meghatározó index 7 bitből ($2^7 = 128$) fog állni (lásd 62. sz. ábra).

A példánkban szereplő cache tár esetében, ha a processzornak egy adatra van szüksége, akkor első lépésben a csoportot azonosító 7 bites index alapján a konkrét csoport kerül kiválasztásra. Ezt követően a letárolt 21 bites címek asszociatív azonosításával meghatározásra kerül a címnek megfelelő sor. Ezen belül egy konkrét bájt megcímzése a 32 bites cím alsó 4-bitjével történhet meg.

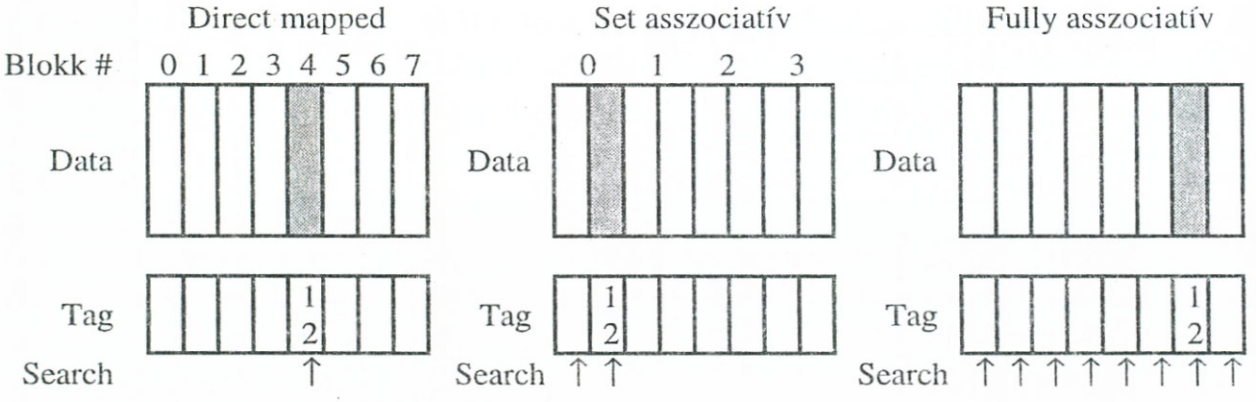


62. sz. ábra
4-utas csoport asszociatív cache

A csoport asszociatív cache tárolók rugalmasabbak, mint a közvetlen leképezésűek és relatíve kevesebb asszociatív összehasonlító áramkört tartalmaznak, és még viszonylag gyorsak. Ezért a gyakorlatban ez a cache típus terjedt el szélesebb körben.

4.3.1.6. Egy tárolóblokk kikeresése a cache-ben

Példaként keressünk egy tárolóblokkot, melynek 12 a címe a főtárban (63. sz. ábra).



63. sz. ábra
Főtárblokk kikeresése a különböző típusú cache tárolókban

Az a cache blokk, melynek 12 a címe a főtárban

- a direct mapping szervezési módban (12 modulo 8 = 4) a 4-es sorban található meg,
- 2-utas csoport asszociatív cache-ben (12 modulo 4 = 0) a 0-ik csoportban és
- teljesen asszociatív cache-ben tetszőleges sorban található meg.

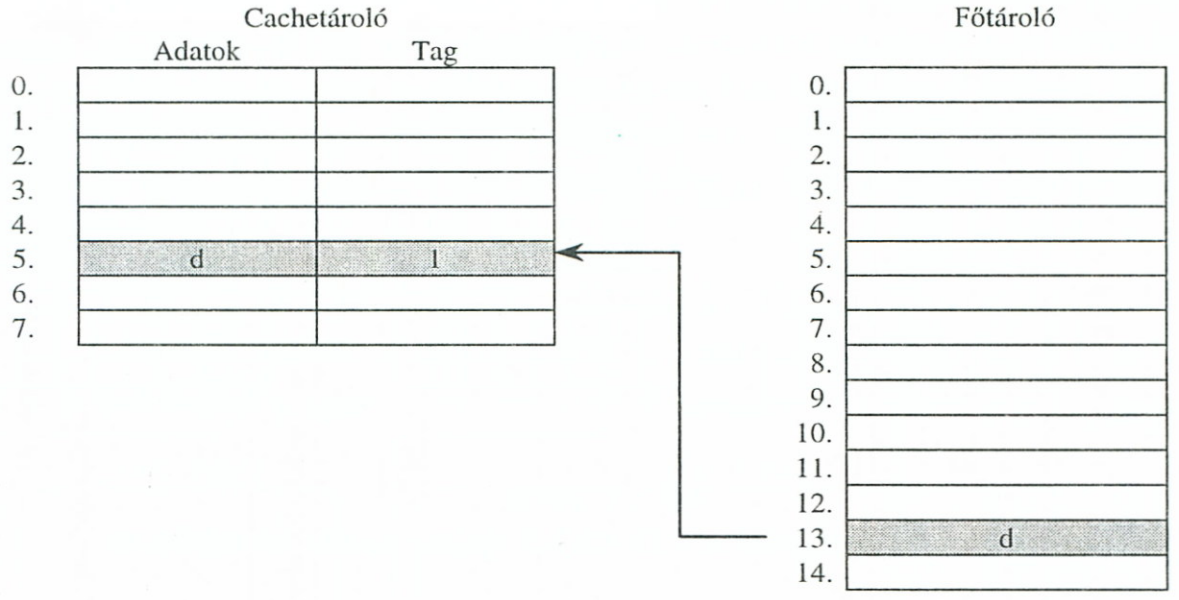
A direct leképezésű cache esetén tehát a következő szabályt fogalmazhatjuk meg:

- *A cache sorának címe = Főtárcím MOD (a cache bejegyzések száma)*
- *A „Tag" tartalma = Főtárcím DIV (a cache bejegyzések száma)*

Megjegyzés:

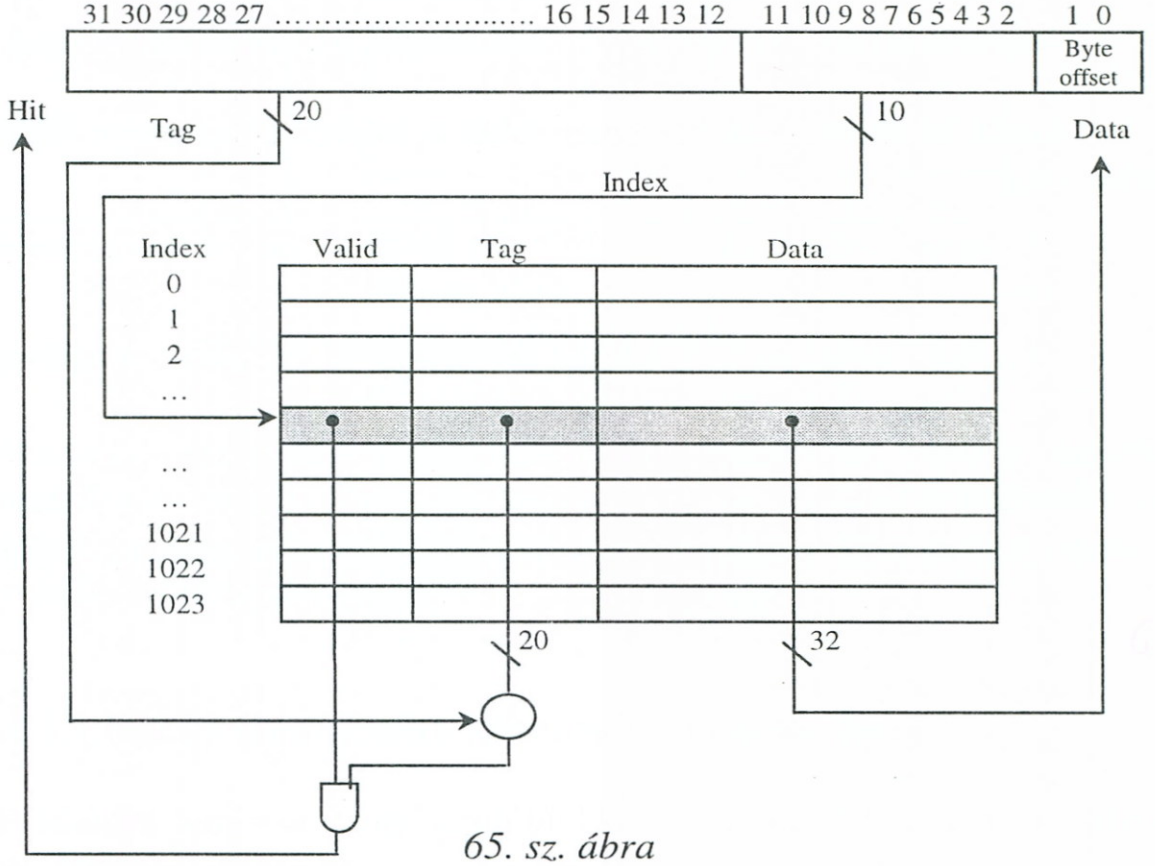
MOD-al jelöltük a maradék, DIV-el pedig a hányadosképzést. Például:

7MOD(3) = 1 7DIV(3) = 2



64. sz. ábra
A direkt leképezésű cache címképzése

Egy tárolóblokk kikeresésének folyamatát egy direkt leképezésű cache-ben a 65. sz. ábrán tanulmányozhatjuk.

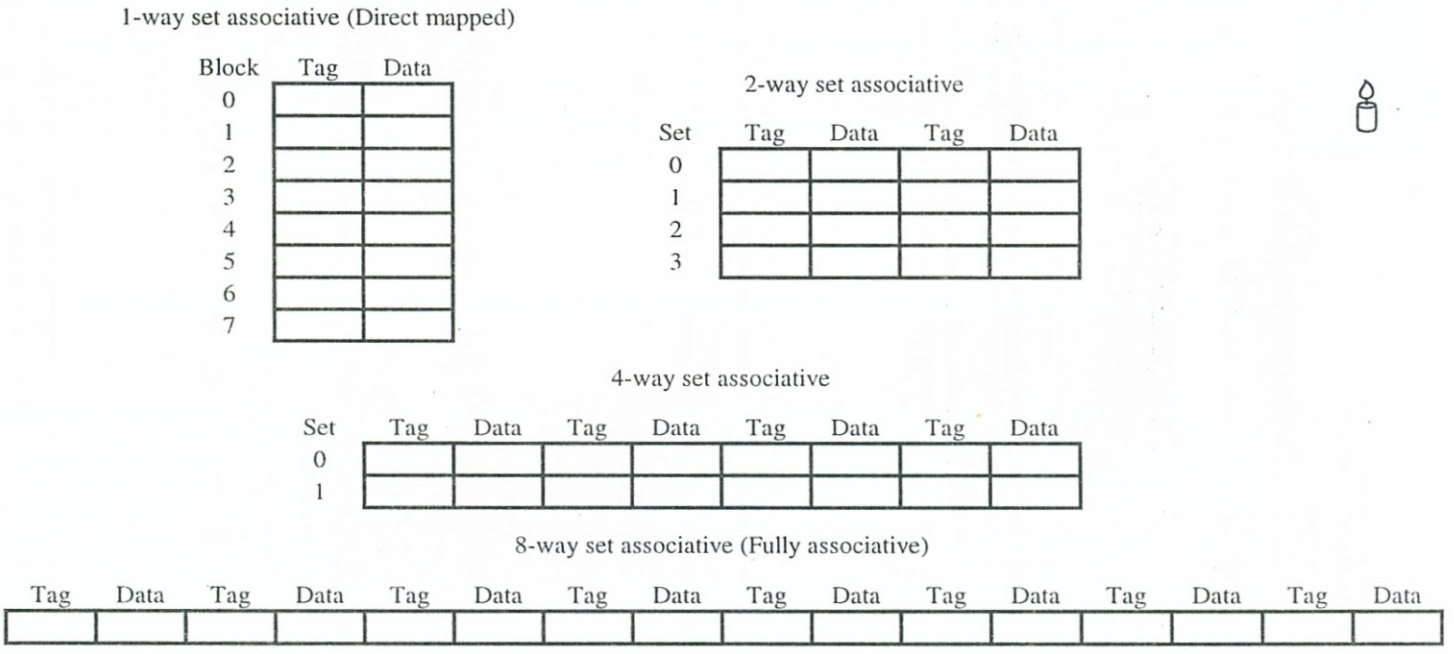


65. sz. ábra
Tárolóblokk kikeresése a direkt leképezésű cache-ben

Az „n"-utas csoport asszociatív cache tárolók kifejlesztéséhez a direkt leképezésű, cache tárolók működése során jelentkező problémák vezettek. A direkt leképezésű cache-ekben több főtárbeli blokk egy rögzített cache tárolósorra képződik le, ezért konfliktusok léphetnek fel, azaz egy cache miss-nél nincs igazán jó helyettesítési stratégia. A megoldást az jelentheti, ha több cache bejegyzés esik egy indexre.

Az „n"-utas csoport asszociatív cache-ben „n" cache bejegyzés esik egy indexre, azaz az index a cache sorok egy halmazát válassza ki. Ezen a halmazon belül a keresett blokk párhuzamos összehasonlítással (asszociatív kereséssel) kerül kiválasztásra.

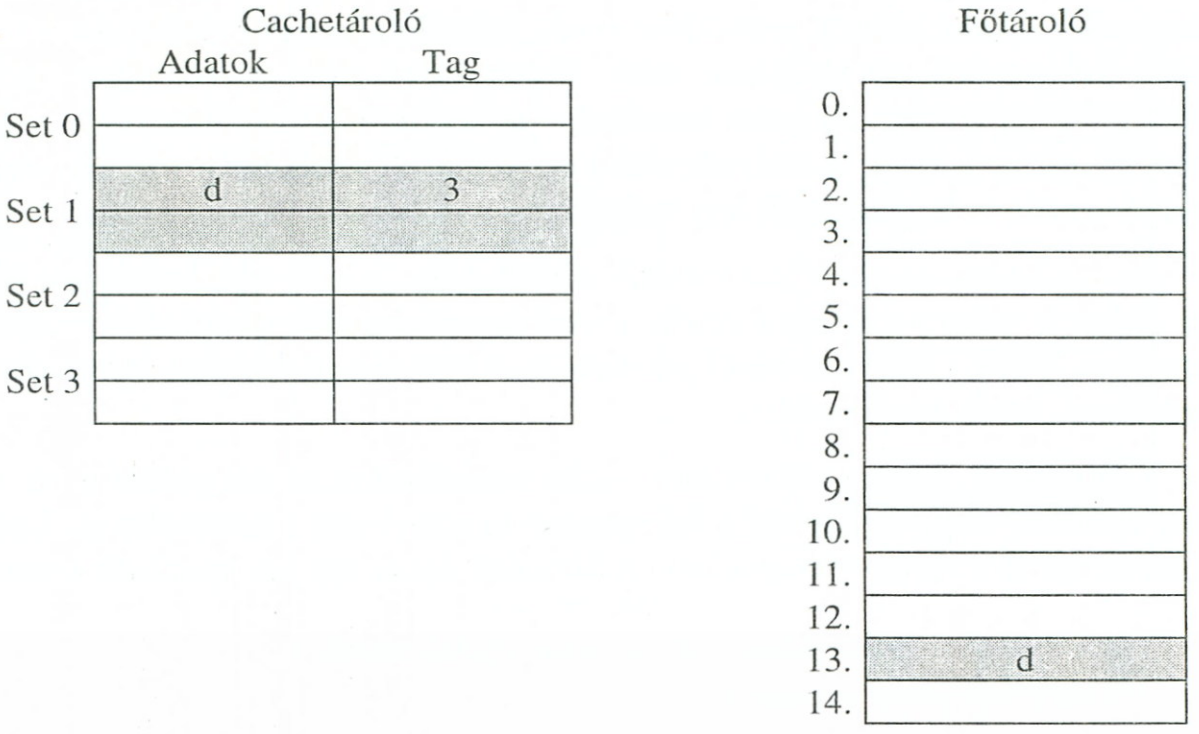
Különböző csoport asszociatív cache struktúrákra mutat példát a 66. sz. ábra. Látható, hogy a közvetlen leképezésű és a teljesen asszociatív működésű cache tár is a csoport asszociatív működésű cache tár két speciális esete.



66. sz. ábra
Különböző csoport asszociatív cache tárak

A csoport asszociatív cache tárolók esetén tehát a következő címképzési szabályt fogalmazhatjuk meg:

A cache sorának címe = Főtárcím MOD (a csoportok száma)
A „Tag" tartalma = Főtárcím DIV (a csoportok száma)



67. sz. ábra
A csoport asszociatív cache címképzése

4.3.1.7. Blokkbemásolás a cache-be, helyettesítési stratégia

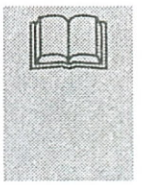
Ha egy főtárblokkot be kell másolni a cache-be, az első kérdés annak eldöntése, hogy a bemásolandó blokk adatai a cache melyik sorába kerülhetnek be, azaz az új blokk adataival, melyik más régebben a cache másolt blokk adatait lehet felülírni. Erre a célra a cache vezérlésében különböző *helyettesítési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



A legbonyolultabb blokkbemásolási eljárás a prefetching továbbfejlesztése, amikor a cache-vezérlő megfelelő stratégiával (algoritmussal) megpróbálja felderíteni, hogy melyik lesz az a főtárbeli blokk, amelyet a processzor igényelni fog és azt előre betölti a cache-be.

4.3.1.8. A cache-ben megváltoztatott adatok visszaírása a főtárba

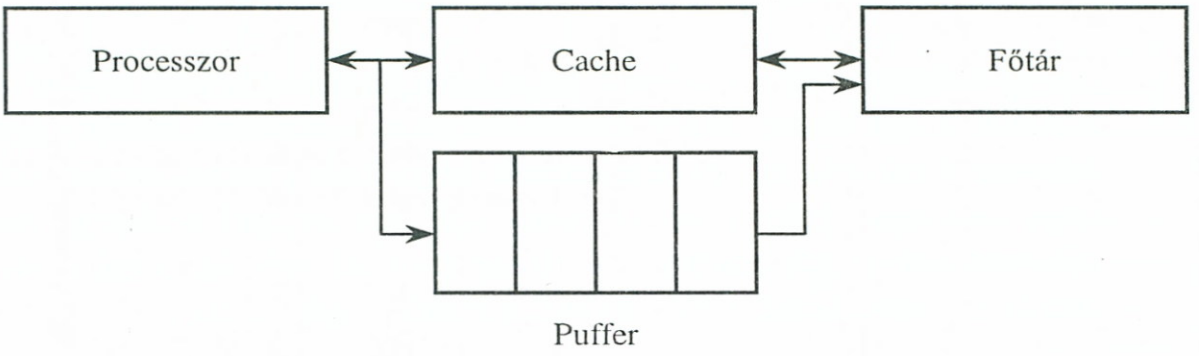
Ha a processzor egy műveletvégrehajtás során megváltoztat egy adatot a cache-ben, akkor igen rövid idő alatt a főtár tartalmát is módosítani kell, hogy a két memória tartalma azonos legyen.

Ez két eljárással történhet:

- *A **közvetlen átírás** vagy **write through** eljárásnál a gyorsítótár írásával együtt megtörténik a főtár írása is. Ennek alkalmazása esetén a főtár és a cache adategyezősége "automatikusan" biztosított, de a főtár írásműveleteinek végrehajtási idejét a cache alkalmazása nem javítja.*

  *A közvetlen írás módszerének hatékonyságát javítja a **pufferelt közvetlen átírás*** (buffered write through), *mely esetben a processzor 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 sebesség-

különbség miatt némi késleltetés is fellép. Így ez alatt az idő alatt a főtárba visszaírás alatt álló adatokra esetleg végrehajtott memóriaművelet hamis eredményeket szolgáltathat. Ennek kivédésére ellenőrző áramköröket szoktak használni. Emellett az is gondot jelent, ha a puffer megtelik, mivel ez esetben a processzor várakozásra kényszerül (lásd a 68. sz. ábra).

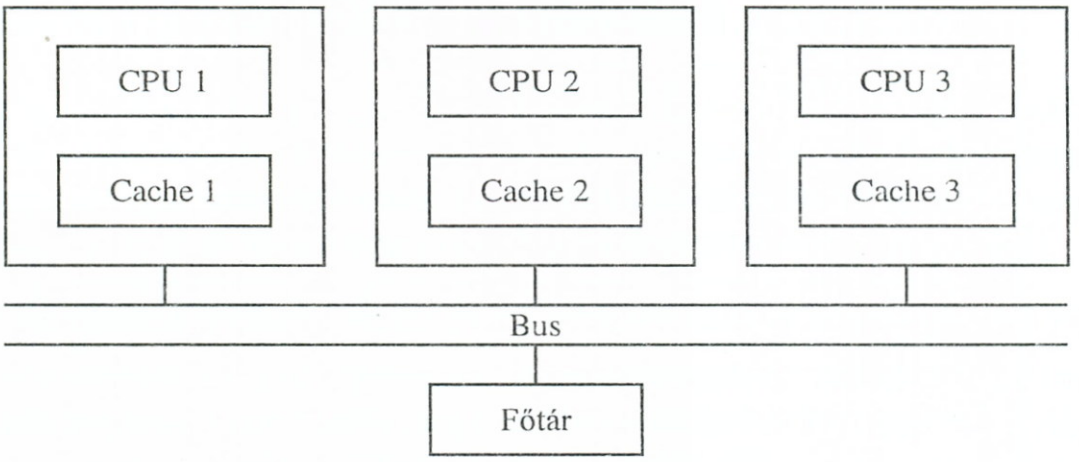


68. sz. ábra
A write through javítása pufferelléssel

- *A **visszaírás vagy write back** eljárásnál a gyorsítótárban módosított adat csak akkor kerül visszamásolásra a főtárba, ha a cache-nek a módosított adatot tartalmazó sorát 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 módosításra került-e. E célra soronként egy kiegészítő bit szolgál, melynek a "módosult" (alter) vagy "piszkos" (dirty) nevet adták. 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.

4.3.1.9. A MESI protokoll

Multiporocesszoros architektúráknál általában több cache tárolót alkalmaznak és a főtár meghatározott közös használata is lehetséges a processzorok között. Ezért ez esetben különösen fontosak azok a szabványeljárások, melyek *a főtár és a cache tárak azonosságát biztosítják* (68. sz. ábra).



69. sz. ábra
Cache tárolók multiprocesszoros rendszerben

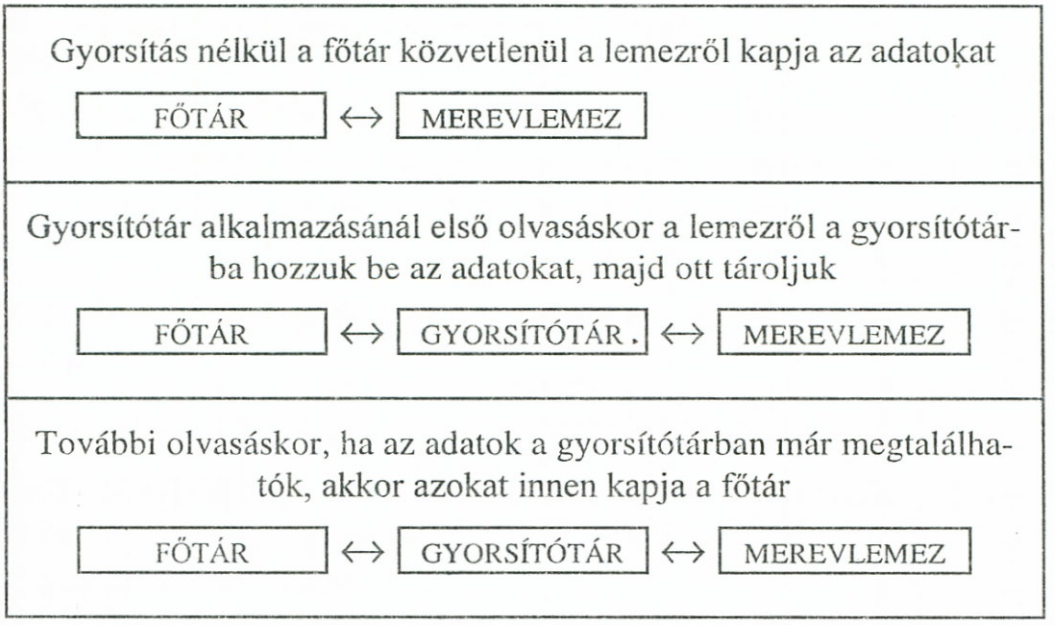
Egy ilyen szabványeljárás ***a MESI protokoll*** *(Modified, Exclusive, Shared, Invalid), mely szerint a cache tárak blokkjainak lehetséges állapotai:*

- *módosított,* mely esetben cache tár blokkja a főtárblokkhoz képest módosítva lett és most ez tartalmazza az aktuális adatokat,
- *kizárólagos,* ekkor a cache-blokkja megegyezik a főtárblokkal, és ez a blokk másik cache-ben nem található meg,
- *megosztott* vagy közös, ekkor a cache blokk a főtárral egyező érvényes adatokat tartalmaz, de ez a blokk több cache-ben is megtalálható,
- *érvénytelen,* amikor a blokk más nem aktuális adatokat tartalmaz (például egy másik cache-ben már módosították).

4.3.2. 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.

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.



70. sz. ábra
Mágneslemez gyorsító tár működésének elve

Két különböző típusa van a merevlemez gyorsításnak:

- *gyorsítás programmal,*
- *gyorsítás hardver eszközzel.*

A lemezgyorsító programok általában az operációs rendszer részét képezik. (Például DOS esetén SMART DRIVE.) Ezek esetében a lemezgyorsító tárrészt a főtárból kell kijelölni.

Nyilvánvaló ennél előnyösebb és hatékonyabb a hardver gyorsító alkalmazása. A korszerű merevlemez interfészek általában tartalmaznak egy 64 vagy 128 K gyorsítótárolót. Léteznek a merevlemez vezérlőelektronikájával egybeépített Mbájt nagyságrendű kapacitással rendelkező hardver gyorsítótárak is.

A hardver gyorsítók működésének a lényege a következő:

- *ha a processzor adatokat akar kiírni a lemezre, akkor a gyorsító vezérlője ezeket letárolja a cache-ben és 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 műveletvégrehajtást;*
- *ha a processzor adatokat akar olvasni a lemezről, 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.*

4.4. A SZÁMÍTÓGÉP BELSŐ MEMÓRIÁJA

A számítógép központi egységének főtárolója (ezt nevezik még: munkatárolónak, belső memóriának, központi tárnak, operatív tárnak, fizikai memóriának vagy csak egyszerűen memóriának is) az aktuálisan futó programfolyamatokhoz tartozó az adatok és programok átmeneti tárolására szolgál.

Működésének lényegét tekintve a belső memóriának két alaptípusa van:

- *a ROM (Read Only Memory = csak olvasható memória) gyárilag beprogramozott, csak olvasható és a felhasználó által megváltoztathatatlan memória. Tartalma a számítógép kikapcsolása után is megőrződik;*
- *a RAM (Random Access Memory = véletlen elérésű memória) a programok által írható és olvasható memória. Tartalma a számítógép kikapcsolása után visszavonhatatlanul elvész.*

4.4.1. A ROM különböző típusai

A ROM-ban tároljuk például az operációs rendszer fixen programozott részét (például PC-knél a BIOS-t = Basic Input Output System-t).

A hagyományos ROM-ok tartalmát gyártáskor határozzák meg (beégetik), ezek tartalma többé már nem módosítható.

A programozható ROM-okban tárolt adatok a hagyományos ROM-okkal szemben egyszer vagy többször megváltoztathatók. Ezek legfontosabb típusai:

- *PROM = programozható ROM. Ebbe a memóriatípusba a felhasználó egyetlen alkalommal beírhatja a számára állandóan megőrzendő adatokat vagy programokat. Ezt követően a PROM tartalma már nem változtatható;*
- *az EPROM = Erasable Programmable ROM tároló az adatokat elektromos töltés formájában őrzi meg. Ez a ROM tároló ibolyántúli (UV) fénnyel törölhető és újra írható.* Ehhez természetesen megfelelő készülék és szakértelem szükséges;
- EEPROM = Electrically Erasable Programmable ROM tárolók újraprogramozásuk előtt egy elektronikus impulzussal törölhetők. Ellentétben az EPROM-al e tárolótípus igen drága.

151 ! *A ROM tárolók elérési ideje* kialakulásuk idejében *a RAM tárolókéval* körülbelül azonos volt, ma már viszont *lényegesen lassúbbak. Ezért a gyakran használatos programrészeket* (pl. PC-knél a BIOS-t) *a mai gépekben átmásolják egy RAM területre. Ezt árnyék memóriának (shadow) nevezzük.*

Flash-memória

! Az 1980-as évek végén jelent meg a „nem felejtő" félvezető memóriáknak egy új típusa a flash memória. *A flash memóriák alapját az EPROM technológia szolgáltatja, de működésükben az a különbség, hogy elektronikusan törölhetők és blokkonként olvashatók. E memóriatípus előnye az is, hogy nincs szükségük tápellátásra, tartalmukat e nélkül is megőrzik.*

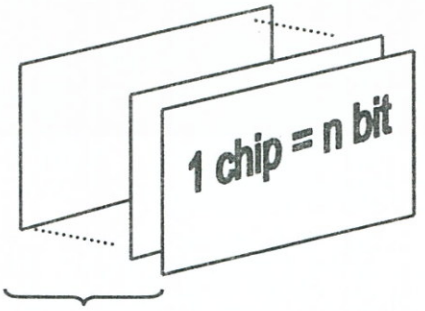
A flash memóriák általában 16 és 32 Mbites chipekkel készülnek, írás/olvasási ciklusuk 70–150 msec között van. *Mivel sebességük több nagyságrenddel kisebb a RAM-oknál, ugyanakkor nagy megbízhatóságúak, érzéketlenek a rázkódásra és kicsi az áramfelvételük, ezért elsősorban a hordozható számítógépekben háttértárakként,* 85x54 mm-es kártyák formájában *alkalmazzák* ezt a memóriatípust.

4.4.2. A RAM felépítése és működése

A számítógépekben a RAM tárolókat írható-olvasható munkamemóriának használjuk, nagyságuk és sebességük meghatározó jelentőségű a teljes számítógépes rendszer teljesítménye szempontjából.

! *A RAM chipek memóriakapacitását Mbit-ben szokás megadni. A ma általánosan elterjedt memóriák chipjei 4, 16 és 64 Mbit kapacitásúak, 1998-ban mutatták be az 1 Gbit-es memóriachipet.*

152 ! *Meghatározott, 2-hatvány darabszámú chip-ből kerülnek összeállításra a memóriamodulok. Ezek elvi felépítését mutatja be a 71. sz. ábra.*



$$1 \text{ memóriamodul} = \frac{n \cdot m}{8} \text{ bájt}$$

71. sz. ábra
A memóriamodul felépítése

Például az 1998-ban bemutatott 1 Gigabites (2^{30} bit) chipekből 16 és 32 képezhet egy memóriamodult. Ezért ezeknek a memóriamoduloknak a kapacitása (m = 32 esetén) $\frac{32 \cdot 2^{30}}{8}$ bájt = 4 Gbájt.

SIMM és DIMM memóriamodulok

A jelenleg alkalmazott memóriamodulok két típusba sorolhatók:

- *a SIMM* (Single Inline Memory Module) *32* bites (paritásbitekkel együtt 36 bites) memóriamodul, melynek 72 *kivezetése vagy lába van;*
- *a DIMM* (Dual Inline Memory Module*) 64 bites szervezésű memóriamodulok a Pentium processzoros gépekhez készültek, ezeknek 168 érintkezője van.* (A DIMM modulokat arról lehet megismerni, hogy a kártya foglalatában két érintkező sor van.)

Hibafelismerés és javítás

A memóriahibák felismerését a paritásellenőrzés szolgálja. Egy 32-bites SIMM modulnál minden bájt kiegészítésre kerül egy paritásbittel [4x(8+1) = 36 bit], *ami alapján a memóriavezérlő a hibás memóriaműveleteket felismeri.* Ez természetesen csak 1 bitnyi hibák esetén lehetséges, mert ha több bit változott meg, akkor elképzelhető, hogy a paritásellenőrzés ezt nem képes felismerni. *A paritásbit generálására önálló áramkört alkalmaznak a memóriamodulban.*

Ha olyan memóriát kívánunk alkalmazni, mely nem csak felismeri, hanem *ki is javítja például az 1-bites paritáshibákat, akkor ehhez ún. ECC = Error Correction Circuit hibajavító áramkörökre van szükség. Ezek hatékony hibavédelmet biztosítanak, de jelentősen megdrágítják a memóriát.* Ennek ellenére egyértelmű tendenciának tűnik a jövőben a hibajavító áramkörök alkalmazásának széles körű elterjedése. Már ma is a memóriamodulok egyre több típusa integráltan tartalmazza a hibajavító áramköröket.

Statikus és dinamikus RAM-ok

A statikus és dinamikus RAM-ok alkotják az írható/olvasható memóriák két legfontosabb típusát.

154

A statikus RAM-ok (SRAM) az adatokat félvezető memóriában (flip-flop) tárolják. Ennek állapota mindaddig fennmarad, amíg ezt újabb írással meg nem változtatjuk, vagy a tápfeszültség meg nem szűnik. Ez azt jelenti, hogy az SRAM-okat nem kell időközönként frissíteni, ciklusidejük lényegében megegyezik az elérési idővel.

A statikus chipek nagyon gyorsak, de nagyon drágák, ugyanakkor kevesebb energiát fogyasztanak. Egy memóriacella helyigénye a szilíciumlapkán a dinamikus RAM-okhoz képest lényegesen nagyobb. Gyorsaságuk és magas áruk miatt *az SRAM memóriát elsősorban a cache tárolókként használják.*

A dinamikus RAM-ok = DRAM elemi cellája egy igen kis méretű kondenzátor, néhány pF kapacitással. Ezek az elemek viszonylag nagy sűrűséggel helyezhetők el az integrált áramkörben, előállításuk relatíve olcsó. Az elemi cellák (kondenzátorok) töltése viszont egy idő után kisül, ezért ennél a memóriatípusnál *a memória tartalmát meghatározott időközönként fel kell frissíteni.* A frissítés cellacsoportonként történik, mely idő alatt a memória a processzor számára nem elérhető.

A DRAM memóriáknak kiolvasás után szükségük van egy „feléledési" időre, ezért ciklusidejük kb. a hozzáférési idő kétszerese.

A frissítést a régebbi architektúrákban az időütemezővel és a DMA vezérlővel oldották meg, mely egy „szimulált" kiolvasással feltöltötte a DRAM memória kondenzátorait. A korszerű memóriák már integráltan tartalmaznak egy frissítő áramkört, mellyel automatikusan, külső vezérlés nélkül fel tudják frissíteni tartalmukat.

4.4.3. A sztatikus RAM-ok típusai

Aszinkron SRAM

Nevének megfelelően *az aszinkron SRAM működése nincs teljesen összhangban a processzoréval, ezért a processzor esetenként várakozásra kényszerül.*

Szinkron SRAM

A szinkron csoportos (burst) statikus RAM, ha a processzor nem túl gyors olyan ütemben szolgáltatja az adatokat, ahogy azt a processzor várja. A memóriát a processzor szinkronizálja, átlagos hozzáférési ideje 8,5–12 nsec között van.

PB SRAM

A PB SRAM = Pipelined Burst Static RAM esetében a memória-áramkörök be/kimeneti regisztereket is tartalmaznak. E regiszterek feltöltése az első adatelérésekor több időt igényel, de ezt követően a feltöltött regiszterek alapján való adattovábbítás már sokkal gyorsabb. Az átlagos hozzáférési ideje a PB SRAM-nak ezért kb. 4,5 nsec.

4.4.4. A dinamikus RAM-ok típusai

DRAM

Régebben a számítástechnikában csak egyfajta RAM típust használtak a főtárolóban, ezt hívták egyszerűen RAM-nak. A statikus RAM-ok megjelenését követően ezt a RAM típust meg kellett különböztetni, ekkor kapta a dinamikus jelzőt, ami arra utal, hogy ennek a memóriatípusnak a tartalma adott időn belül megsemmisül, ha nem frissítjük fel.

A DRAM mátrixszervezésű azaz oszlopokból és sorokból épül fel, melyek „metszéspontjában" található egy memóriacella. (A memóriacellában az információt egy elemi kondenzátor tárolja.) Az elvi felépítést mutatja a következő ábra:

72. sz. ábra
A DRAM logikai felépítése

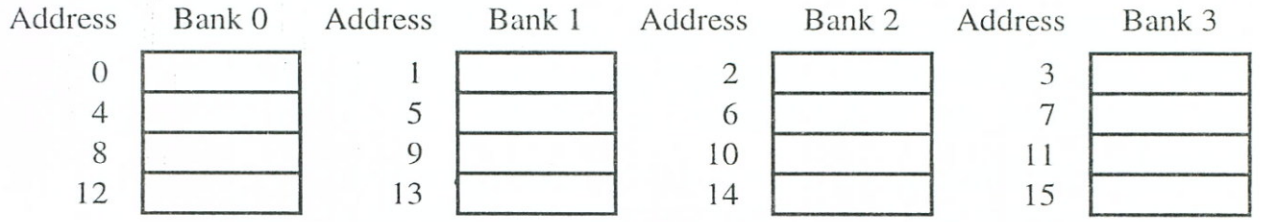
Ezek szerint a memóriacellát megcímezni két lépcsőben lehet:
- *először a mátrix egy sorát címezzük meg, ezt követően*
- *a sorból kiválasztjuk az adott oszlophoz tartozó cellát.*

Ebből következik, hogy a DRAM áramkörnek egyszerre nincs szüksége a teljes címre, hanem először a sorcímet, majd az oszlopcímet kell rendelkezésére bocsátani. Ezért a processzor által megadott címet egy hardver egységgel két részre, sor és oszlopcímre kell szétválasztani.

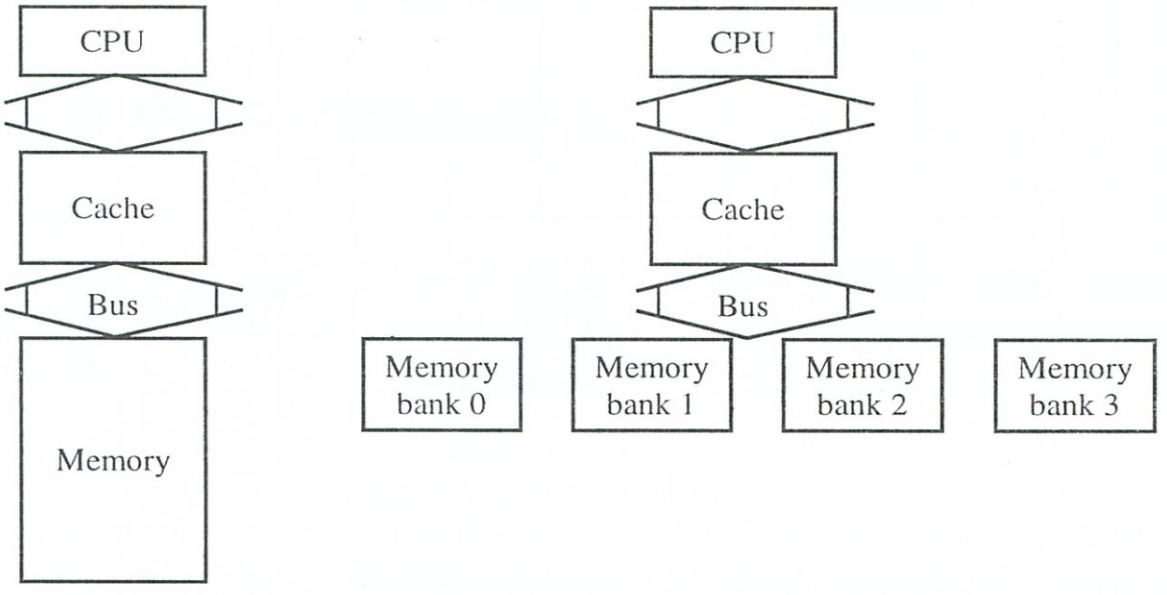
A processzorok teljesítménye az elmúlt évtizedben rohamosan nőtt, ezzel a memóriák egyre nehezebben tudtak lépést tartani. A processzorhoz képest lassú memória okozta problémát először az L1 és L2 cache tárolókkal próbálták megoldani, de rövidesen ez is kevésnek bizonyult. Ez vezetett az egyre újabb és újabb DRAM memóriatípusok kifejlesztéséhez.

Gyorsítás átlapolt memóriakezeléssel (Memory interleaving)

Ennek alapgondolata a következő: *a memóriát egymástól függetlenül címezhető és olvasható részekre, úgynevezett memóriabankokra osztjuk fel. Ennek elvét láthatjuk a következő ábrákon:*



73. sz. ábra
A memória címzése interleaved memory esetén



74. sz. ábra
A hagyományos és az átlapolt memóriafelépítés

Feltételezve, hogy a processzor általában a memóriát címfolytonosan olvassa, a 0-ik memóriabankból kiolvasott adat hozzáférése alatt az 1 memóriabankban lévő következő címen lévő adat már megcímezhető. Ez kissé leegyszerűsítve azt jelenti, hogy címfolytonos olvasás esetén az adatok kiolvasása kb. kétszeres sebességgel történhet.

!

FPM RAM

Az FPM RAM = Fast Page Mode RAM, *azaz a gyors lapmódú RAM működésének lényege, hogy ha a DRAM áramkörök megkapják a sor és oszlopcímet és a memóriarekesz kiolvasása megtörténik, akkor címfolytonos olvasásnál az áramköröknek elegendő megadni a következő oszlopcímet.*

!

EDO RAM

Az EDO = Extended Data Output *RAM a gyors lapmódú működés memóriáknál kb. 10–15%-kal gyorsabb. Ez azáltal érhető el, hogy az EDO RAM-nál az adat a memória kimenetén hosszabb ideig áll rendelkezésre egy másodlagos tároló alkalmazásával.* Jellemző működési sebessége 50 MHz.

!

SDRAM és BEDO RAM

Az SDRAM = Synchrones DRAM és a *BEDO* = Burst Extended Data Output *az EDO RAM-oknál is gyorsabb hozzáférést biztosítanak, melyet a „burst", azaz blokkos (csoportos) hozzáférési technika bevezetésével és a processzor és a memória működésének szinkronizálásával lehet elérni* (a RAM a memória írási és olvasási műveleteket a processzor működését meghatározó órajelhez szinkronizálja). *A burst üzemmódban az adatokat megfelelő áramkörökkel blokkokba szervezik.* (Ekkor a memóriavezérlés automatikusan generálja a következő memóriacellák címét.) További gyorsítást eredményez a pipeline alkalmazása, amikor a RAM már olvassa az adatokat, miközben még adatokat ad ki (két memóriabank alkalmazásával). Jellemző sebessége 100 és 133 MHz, hozzáférési ideje (burst) 10 nsec.

!

E memóriatípus újabb változata a DDR SDRAM (Double Data Rate SDRAM), melyek sebessége 300 és 400 MHz.

4.5. VIRTUÁLIS TÁRKEZELÉS

Egy program végrehajtásához a megfelelő programrésznek és az általa feldolgozott adatoknak a főtárban kell lenniük. Az aktuálisan nem futtatott programokat és a hozzájuk tartozó adatállományokat egy közvetlen elérésű

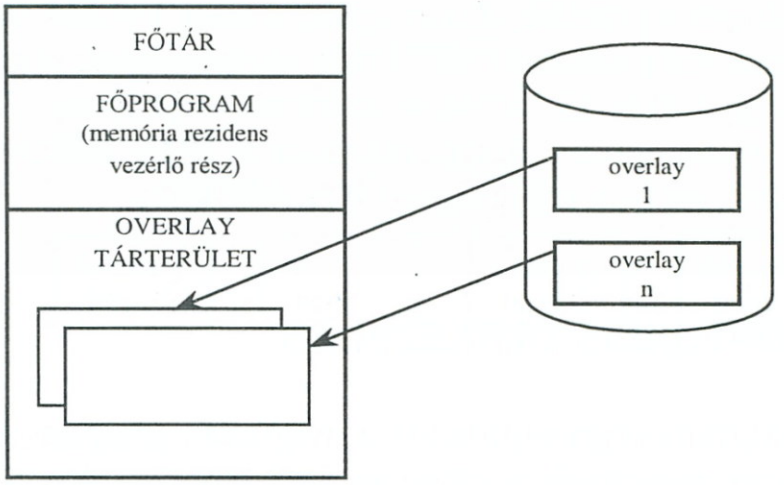
háttértárolón tároljuk és csak akkor töltjük be a központi memóriába, ha szükség van rájuk. Ezt a megoldást a főtár korlátozott memóriakapacitása mellett az is szükségessé teszi, hogy egy egységnyi információ tárolási költsége a háttértárolón nagyságrendekkel kisebb, mint a főtárban.

A központi memória a számítástechnika egész története alatt mindig az egyik legszűkösebb erőforrás volt. Az egyre komplexebb alkalmazásokhoz egyre nagyobb méretű programokra volt szükség, a kezelt adatállományok tárolóigénye is állandóan növekedett a fejlődés során. Különösen a multiprogramozott és multitasking üzemmód kialakulását követően vált jellemzővé, hogy a számítógépen „egyidejűleg" sok olyan nagy memóriaigényű programfolyamatot kell futtatni, melyek együttesen nem férnek be a főtárba.

Ha erre a problémára hatékony megoldást szeretnénk találni, a következőket célszerű figyelembe venni:

- *A programok végrehajtásuk során legtöbbször egy korlátos memóriaterületen belül dolgoznak*, igen ritka például az olyan ugró utasítás, mely a program elejéről a végére adja át a vezérlést. Ez a már tárgyalt *lokalitás elve a programvégrehajtásban.*
- *Az egyes programrészek között igen nagy eltérések lehetnek abból a szempontból, hogy a programvégrehajtás során milyen gyakran van rájuk szükség.* Erre jó példát szolgáltatnak az operációs rendszer különböző hibakezelő, analizáló rutinjai, melyek aktivizálására a teljes futási idő tíz ezreléke alatti gyakorisággal van csak szükség. *Ez a gyakoriság elve a programvégrehajtásban.*

A memóriakapacitás hiánya vezetett az 1960-as, 70-es években oda, hogy a programfejlesztők az ún. „overlay" technikát kezdték alkalmazni. Ennek elvét mutatja be a 75. sz. ábra.



75. sz. ábra
Az overlay technika

Az overlay technika alkalmazásakor csak a programrendszer általános, mindig szükséges vezérlő programja kerül betöltésre a főtárba és emellett a főtárban lefoglalásra kerül egy tárolóterület, melyre a ritkábban szükséges programmodulok (melyeket ebben az esetben overlay-nak neveznek) igény szerint (csak akkor, ha a program futása során szükség van rájuk) kerülnek betöltésre. A lefoglalt tárterület nagyságát ez esetben a legnagyobb méretű modul tárolóigénye határozza meg. Ennek a megoldásnak volt egy nagy hátránya: az overlay-k töltését, nyilvántartását stb. az egyes felhasználói programokban le kellett programozni.

Ezért optimálisabb az a megoldás, ha ezt a feladatot az operációs rendszerre bízzuk.

4.5.1. A virtuális tárkezelés alapfogalmai

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, az-az virtuális tárterületnek. A háttértárolón kijelölt virtuális tárolót osszuk fel blokkokra, melynek méretét a lokalitás elvének figyelembevételével határozzuk meg.

VIRTUÁLIS TÁR

0 blokk	1 blokk	3 blokk	....	n-ik blokk

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

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 főtárblokkot írhatjuk felül?*
  Ezt 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 tárkezelés jellemző adatai a következők:

- Blokkméret 4 kbájt – 64 kbájt
- „Hit Time", azaz a főtárban bennlévő blokkban található adat hozzáférési ideje 40 – 100 órajel
- Hibaelhárítás („Miss Penalty"), azaz ha a keresett adatot tartalmazó blokk nincs benn a főtárban 700'000 – 6'000'000 órajel
  Ezen belül a blokk hozzáférési ideje:
  - a háttértárban 500'000 – 4'000'000 órajel
  - A blokk átviteli ideje 200'000 – 2'000'000 órajel

- Hibaarány (Miss-Rate) 0.00001 – 0.001%
- Virtuális tár kapacitása 512 Mbájt – 8 Gbájt

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.

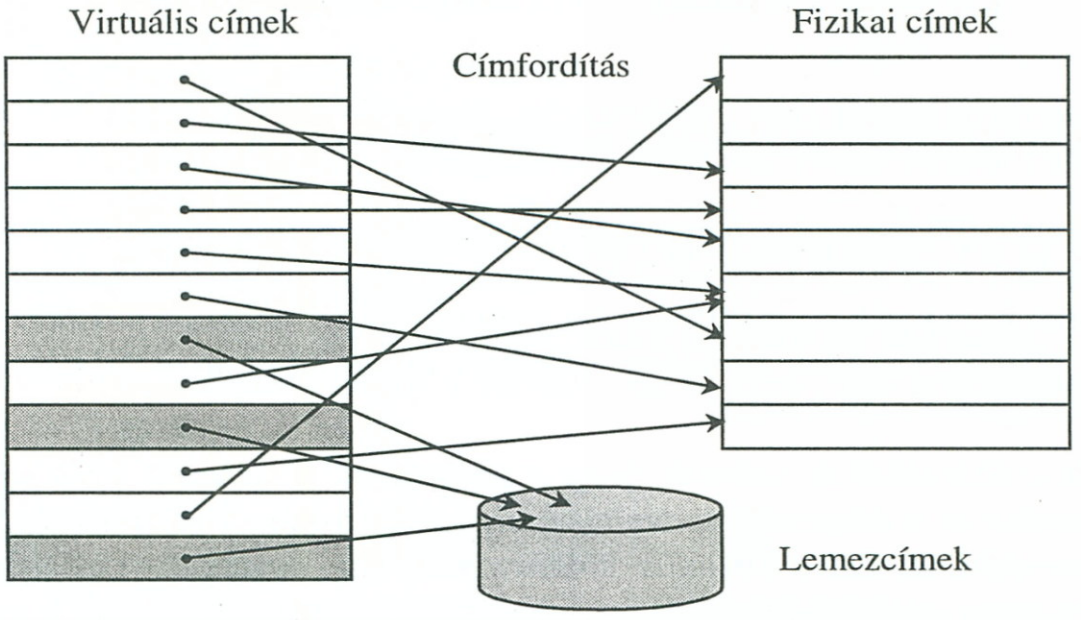
A virtuális cím leképezése fizikai címmé

Természetesen a processzornak a műveletvégrehajtás során a főtár valódi vagy másképpen nevezve fizikai címeire van szükség, 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.*

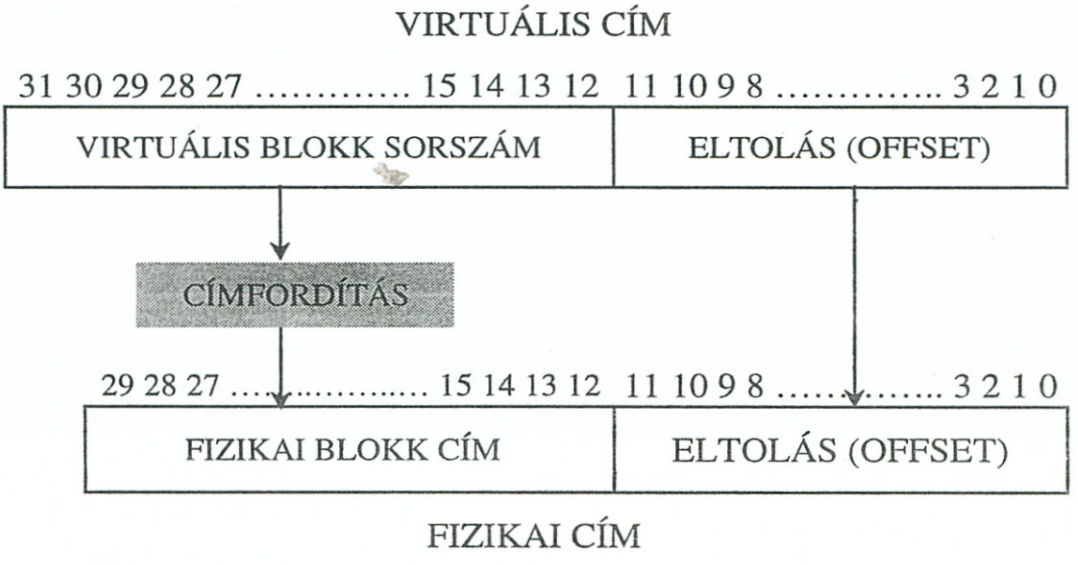
A virtuális címek fizikai címmé történő leképezését az első virtuális tárkezelést alkalmazó számítógépekben még az operációs rendszer végezte. Ma már kizárólagosan az jellemző, hogy *ezt a feladatot egy megfelelő címleképezési áramköröket tartalmazó hardver egység, az MMU (Memory Management Unit) végzi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

A virtuális címek és a fizikai címek viszonyát mutatja be a 76. számú ábra.



76. sz. ábra
A virtuális és a fizikai cím összefüggései

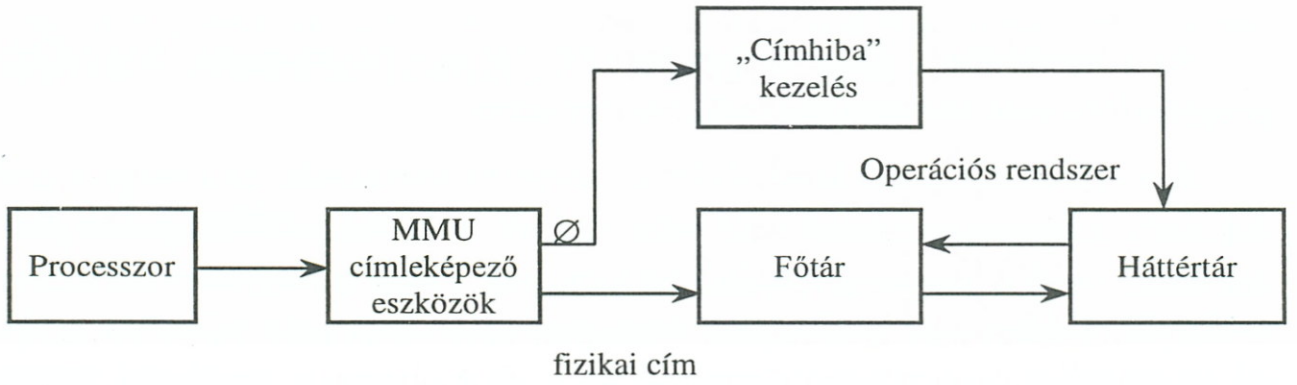
Ennek megfelelően a virtuális címek fizikai címmé történő leképezésének elve a következő ábra szerinti.



77. sz. ábra
A virtuális cím leképezésének elve

A címleképezést tehát a következők szerint lehet modellezni:

- Adott a V = {0,1 ... n-1} virtuális címtér és az F = {0,1 ... m-1} fizikai címtér. Itt n>>m.
- A címleképezést egy f : V → FU{∅} függvény valósítja meg. (Itt ∅ jelöli, hogy nincs megfelelő fizikai cím a főtárban.)



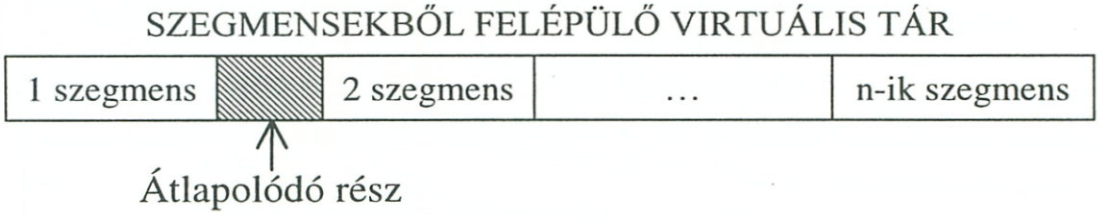
77. sz. ábra
A címleképezés folyamata

Az MMU tehát 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 megszakításkezelőrutinja 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 virtuális tárkezelésnek két alapvető formája van, a szegmentálás és a lapozás, ekkor virtuális tár blokkjait szegmensnek, illetve lapnak nevezzük.

4.5.2. 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.



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 = szegmens fizikai kezdőcíme + relatív cím összefüggéssel meghatározható.

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,

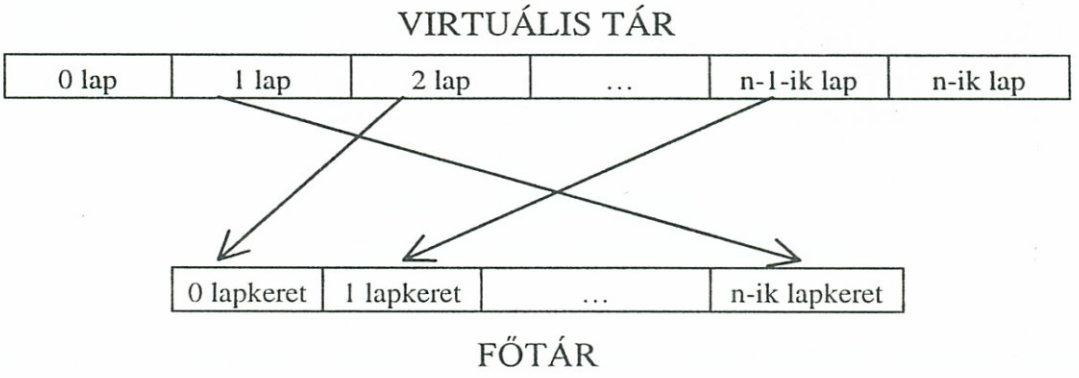
- az ún. „legrosszabb" helyre, amikor az a cél, hogy a betöltést követően a szegmens mellett a lehető legtöbb tárterület maradjon szabadon.

Összefoglalva az eddigieket, megállapíthatjuk, 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.

4.5.3. Lapozás

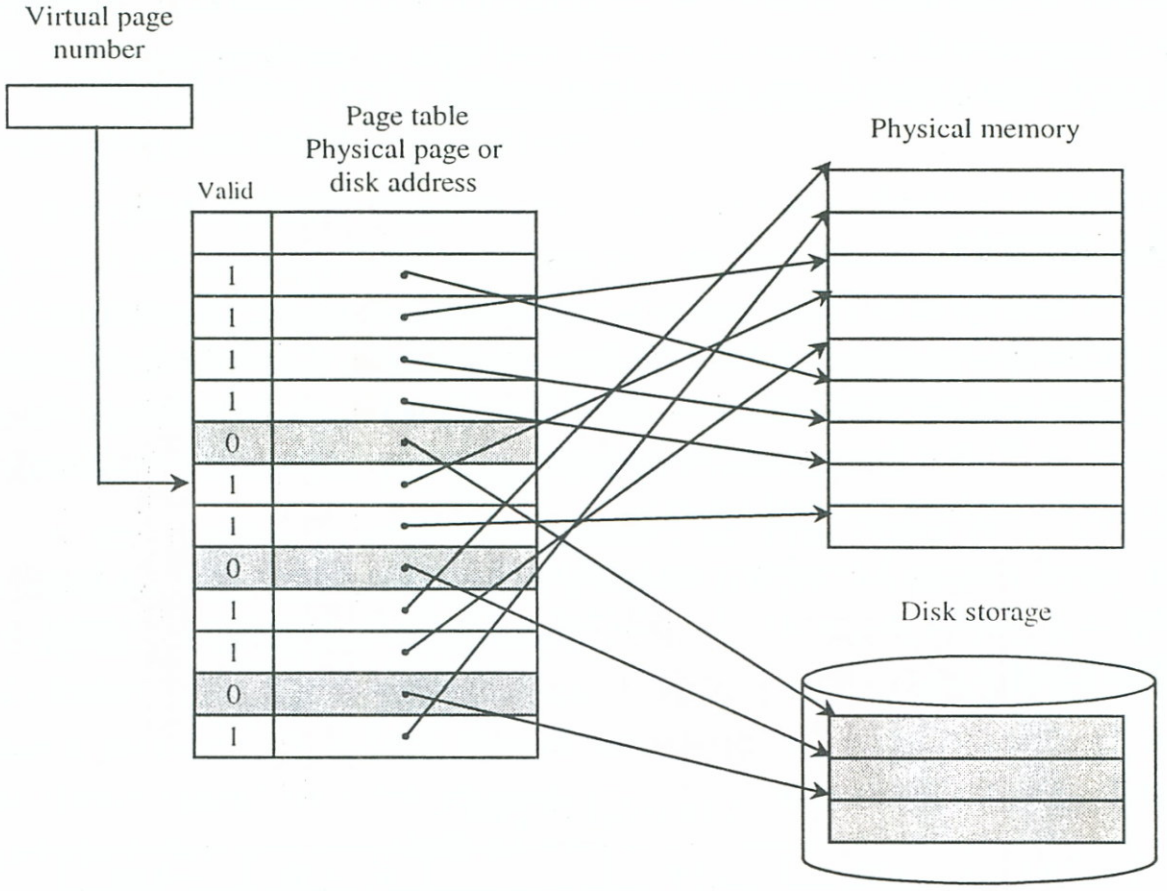
Ha a virtuális tár rögzített méretű nem átlapolható blokkokból áll, akkor ezeket lapoknak nevezzük, a virtuális tárkezelésnek ezt a formáját pedig lapozásnak. A főtár a lapmérettel megegyező nagyságú részekre van felosztva, ezeket lapkereteknek (frame) nevezik.

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



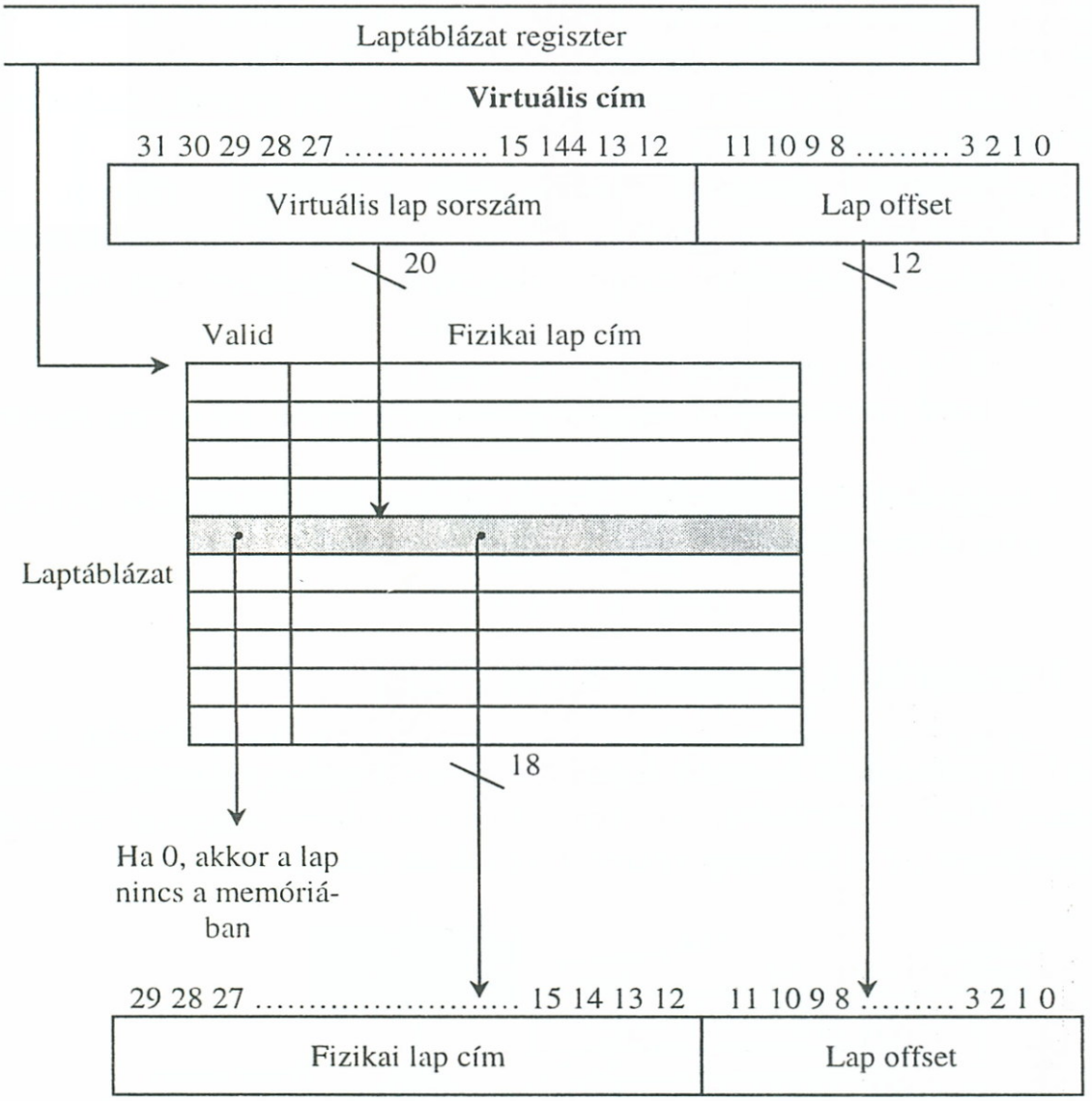
A lapozásnál a virtuális cím hasonlóan épül fel, mint azt a szegmentálásnál láttuk, azaz *a lap sorszámát és a megcímzett bájtnak a lap kezdetétől számított relatív címét tartalmazza.*

A központi memóriába beolvasott lap fizikai kezdőcímét a főtárban – azaz annak a lapkeretnek a címét, ahová a lap elhelyezésre került – a laptáblázatok tartalmazzák (lásd 79. sz. ábra). Általában szintén a laptáblázat tárolja a főtárban nem szereplő lapok lemezcímét.



79. sz. ábra
A laptáblázat tartalma

Minden programfolyamatnak (processznek) saját laptáblázata van, ennek kezdőcíme sokszor egy regiszterben található. A laptáblázatok segítségével a virtuális cím fizikai címmé történő átszámítása a 80. sz. ábra eljárása szerint történik.



79. sz. ábra
Címleképezés laptáblázatokkal

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.

A laptáblázatok nagysága igen jelentős is lehet. Ezt egy példán keresztül mutatjuk be. Tegyük fel, hogy 32 bites virtuális és fizikai címekkel és 4 Kbájtos lapnagysággal dolgoznak. Ez 4 bájt/lap bejegyzést jelent a

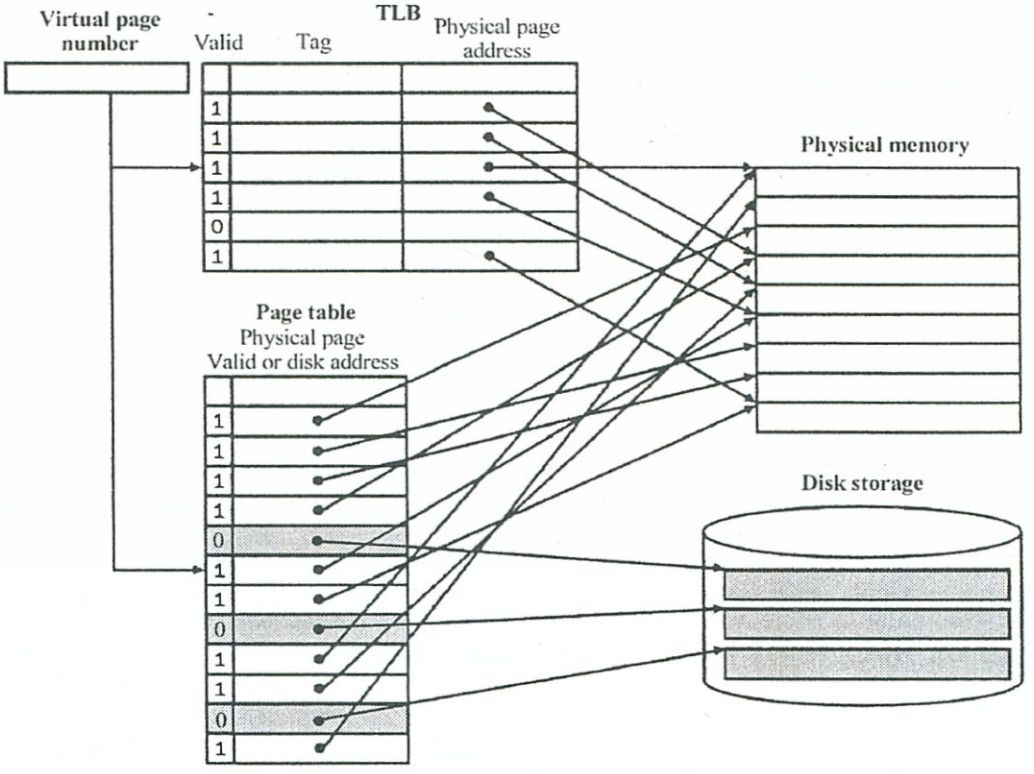
laptáblában. A laptáblabejegyzések száma a teljes címtérben $2^{32}/2^{12}=2^{20}$, így az ennek megfelelő maximális laptábla méret $2^{20}\cdot 2^2$=4 Mbájt.

Ezért fontos szerepe van a tárolóigény csökkentő és gyorsító technikáknak. Ezek közül a fontosabbak:

- Hierarchikus felépítés alkalmazása, például: főtábla ⇒ laptáblakatalógus ⇒ laptábla. Ebben az esetben a magasabb szintű tábláztok, a következő szintű táblázatok címadatait tartalmazzák. Például:
  - RISC processzorok 3-5 szint,
  - Intel processzorok szegmenscímzésen belüli kétszintű lapcímzés
- A leggyakrabban szükséges lapok adatait lapcímfordítást gyorsító cache tárban tárolják.

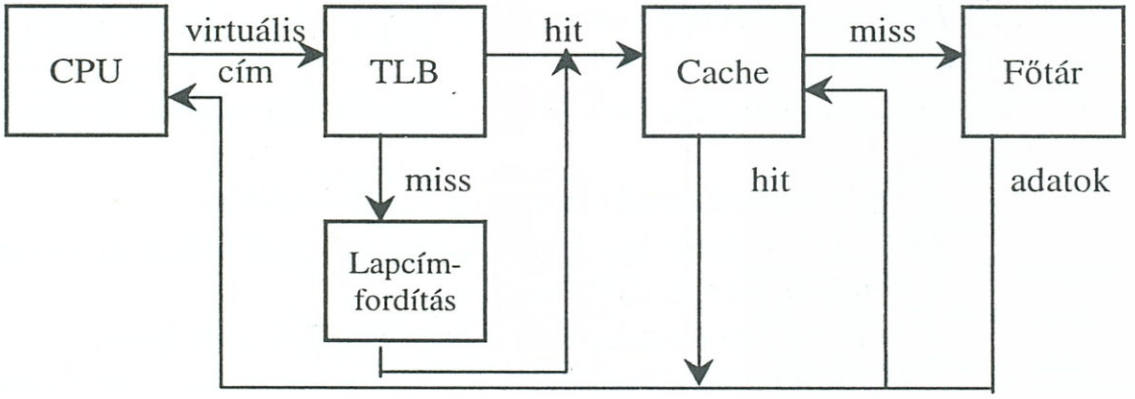
4.5.4. Translation Lookaside Buffer (TLB)

! *A TLB a leggyakrabban használt lapok lapcímfordításhoz szükséges adatait tartalmazza. Fully associative, set associative vagy direct mapped cache tároló, legtöbbször 32–256 bejegyzéssel (lásd 81. sz. ábra).*



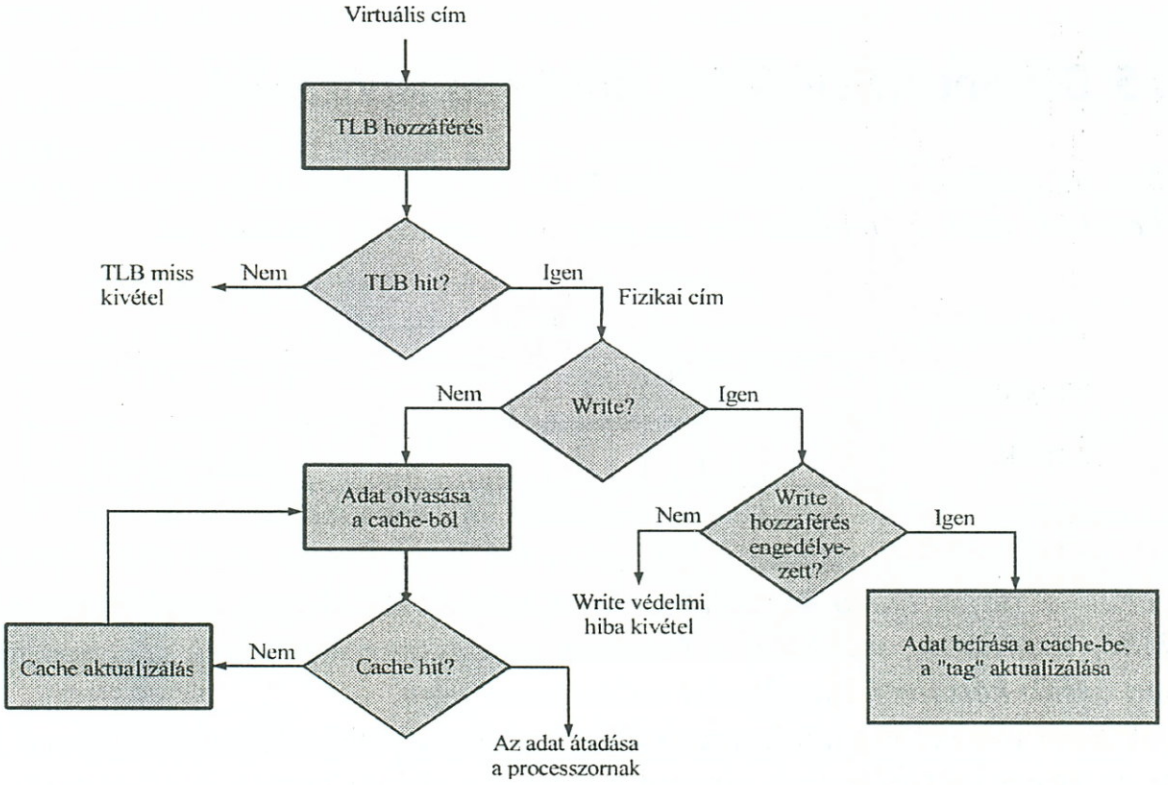
81. sz. ábra
A TLB és a virtuális tárkezelés

A TLB általában a processzor és a cache tároló között helyezkedik el, ekkor a tárolóhozzáférés folyamatát a 82. sz. ábra mutatja be.



82. sz. ábra
A TLB és a cache

A tárolóhozzáférés folyamatának logikai lefutását a TLB hit esetén a 83. sz. ábra mutatja be.



83. sz. ábra
A tárolóhozzáférés folyamata TLB hit esetén

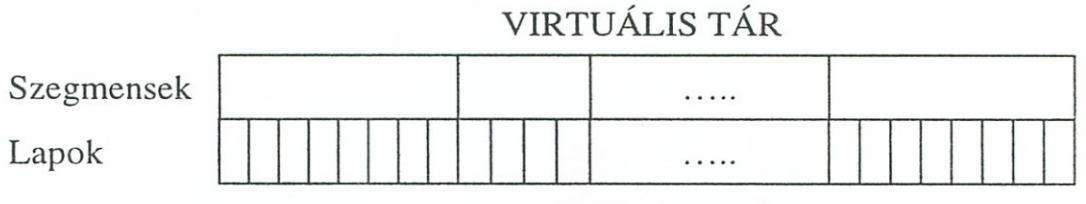
A TLB MISS esetében a laptáblázat érvényességi bitjét (Valid Bit) kell kiolvasni és az alapján:

- *Ha a lap a főtárban van, de nincs bejegyzés a TLB-ben (ez a „tiszta" TLB miss), akkor a megfelelő bejegyzést a laptáblából be kell írni a TLB-be.*
- *Ha a lap nincs a főtárban, akkor laphiba keletkezett. Ennek kezelése:*
  - *A lap címének meghatározása a háttértárban,*
  - *Egy „D" = dirty lap kiválasztása a laptáblában és visszaírása a háttértárba (pl. LRU szerint),*
  - *A hiányzó lap bemásolása a háttértárból a főtárba,*
  - *A laptábla aktualizálása,*
  - *A TLB aktualizálása.*

Lapozásos virtuális tárkezelésnél fregmentáció, azaz üres kihasználhatatlan memóriaterületek nem képződhetnek, mivel a lapok csak azonos méretű lapkeretek helyére kerülhetnek be a főtárba. Legfeljebb az fordulhat elő, hogy a nem használt lapkeretek üresek, azaz nem kerülnek feltöltésre lapokkal. (Ez az eset általában csak a rendszerindításkor fordul elő.)

4.5.5. Szegmentált virtuális tárkezelés lapozással

Ebben az esetben a szegmentáláson belül alkalmazzák a lapozásos virtuális tárkezelést, azaz a virtuális tár szegmensei lapokból épülnek fel.



Ezt a fajta virtuális tárkezelést használják védett üzemmódban a i80386, i80486-os, valamint a Pentium processzorok, melyek szegmenscímzésen belüli kétszintű lapcímzést valósítanak meg.

Ebben az esetben a szegmenstábla és a laptáblakatalógus a központi tárban, az egyes laptáblázatok viszont a virtuális tárban kerülnek elhelyezésre.

Végezetül nézzük meg két konkrét processzor a virtuális tárkezelésének jellemző adatait (84. sz. ábra).

Jellemző	Intel Pentium Pro	Power PC 604
Virtuális cím	32 bit	52 bit
Fizikai cím	32 bit	32 bit
Lap méret	4 KB, 4 MB	4 KB, 256 MB
A TLB szervezése	Külön TLB az adatokra és utasításokra 3-utas csoport asszociatív LRU stratégia Utasítás TLB: 32 db bejegyzés Adat TLB: 64 db bejegyzés TLB miss hardver kezelés	Külön TLB az adatokra és utasításokra 2-utas csoport asszociatív LRU stratégia Utasítás TLB: 128 db bejegyzés Adat TLB: 128 db bejegyzés TLB miss hardver kezelés

84. sz. ábra
A Pentium Pro és Power PC 604 processzorok virtuális tárkezelése

4.6. A TÁROLÓ VÉDELMI RENDSZER

A számítógép tárkezelő rendszerének egyik legfontosabb feladata a programok és adatok védelme, például a szándékos vagy véletlen felülírástól. Ennek különösen azért van jelentősége, mivel az operációs rendszer és a felhasználói programok fizikailag azonos memóriát használnak.

A védelmi rendszerrel minimálisan a következő feladatokat kell megoldani:

- *a hibás vagy nem létező címek kiszűrését,*
- *az operációs rendszer védelmét a felhasználói programoktól,*
- *a felhasználói programok egymástól történő védelmét,*
- *az adatokhoz, programokhoz történő hozzáférés jogosultságainak ellenőrzését.*

A védelmi rendszernek a multiprogramozott, illetve multitaszking üzemmód általános elterjedését követően lett kiemelkedő jelentősége, amikor a számítógép „egy időben" több programfolyamat végrehajtását végzi.

4.6.1. Tárolóvédelem privilégizálási szintekkel

A tárolóvédelem egyik módszere a programfolyamatok, vagy más néven taszkok privilégizált osztályokba történő besorolása.

Ez azt jelenti, hogy *minden egyes taszkhoz hozzárendelésre kerül egy privilégizálási szintszám, melyhez meghatározott jogosultságok, illetve tiltások kapcsolódnak.*

Magas jogosultság	0 privilégium szint
Alacsony jogosultság	1 privilégium szint

Taszk 1
Taszk 3

Taszk 2

Az ábrán látható példában *a magas jogosultsággal rendelkező 0-as privilégiumszinten futó taszk 1 és taszk 2 saját szintjén egymás rutinjait meghívhatja és megteheti ezt az alacsonyabb jogosultsági szinten futó taszk 3 rutinjaival is. Ugyanakkor a taszk 3 a taszk 1, illetve taszk 2 rutinjait csak speciálisan ellenőrzött módon* (úgynevezett kapukon keresztül*) képes meghívni.* A taszk 1 képes írni a taszk 3-hoz tartozó tárolóterületet, ez viszont megfordítva nem lehetséges.

A napjaink számítógépes rendszereiben minimálisan két privilégizálási szintet különböztetnek meg:

- *egy magas jogosultsági szintet az operációs rendszerhez tartozó programok számára,*
- *egy alacsonyabb jogosultsági szintet a felhasználói programok számára.*

Így például a RISC processzorok jellemzően két privilégizálási szintet használnak, míg a i80x86 processzorokra a 4 szintű védelem a jellemző.

Egyes számítógéparchitektúrákban privilégizált utasítások is megtalálhatók. Ez azt jelenti, hogy a processzor utasításkészletében vannak olyan speciális utasítások, melyeket a processzor csak az operációs rendszer privilégizálási szintjén futó taszkok esetén hajt végre. (Ezeknek az utasításoknak a használata a felhasználói programok privilégizálási szintjén „privilégizálási hiba" kivételt eredményez.)

A privilégizálási szintek közötti védelmet az is segíti, hogy minden egyes szint egymástól elkülönült önálló tárolóterülettel is rendelkezik, melybe a felfüggesztett taszkok legfontosabb adatai elmenthetők (i80x86 processzorok esetén ez a TSS, a taszk állapot szegmens).

4.6.2. Tárolóvédelem deszkriptorokkal

Az azonos privilégizálási szinten futó taszkok tárolóterületét is védeni kell egymástól. Ezért minden egyes taszkhoz speciális védelmi táblázatok kerülnek felépítésre, melyekben található információk *(deszkriptorok) meghatározzák a taszkhoz tartozó szegmensek és lapokhoz történő hozzáférési jogosultságokat. Ezek a következők:*

- *olvasási jog,* azaz a taszkhoz tartozó szegmensek vagy lapok adatait egy másik taszk olvashatja-e,
- *írási jog,* azaz a taszkhoz tartozó szegmensek vagy lapok adatait egy másik taszk átírhatja-e,
- *végrehajtási jog,* azaz a taszkhoz tartozó tárolóterületen található kódszegmens futtatását egy másik taszk leindíthatja-e.

Példaként az i80x86 processzorok védelmi rendszerét tekintjük át.

A védelmi rendszer az ún. *rendszerobjektumok védelmét biztosítja.* Ezek a következők lehetnek:

- *program és adatszegmensek,*
- *lapok,*
- *taszkok,*
- *táblázatok (például laptábla, a védelmi rendszer táblázatai stb.),*
- *kapuk, melyek „ajtaját" a programfolyamatok csak akkor nyithatják ki, ha jogosultsági kulcsuk illik a „zárba".*

A védelem a következő eszközökkel valósul meg:

- 4 szintű privilégizálási rendszer (PL, IOPL $\rightarrow$ 0, 1, 2, 3),
- deszkriptorok, melyek leírják a rendszerobjektumokat,
- kapuk (megváltozott PL esetén ellenőrzés),
- TSS = Taszk állapot szegmens, ahova taszkváltáskor a taszk legfontosabb paraméterei elmentésre kerülnek.

A védelemhez szükséges deszkriptorokat három típusba sorolható táblázatok tartalmazzák:

- *GDT = Globális Deszkriptor Tábla (rendszerszintű deszkriptorokat tartalmazza)*
- *LDT = Lokális Deszkriptor Tábla (taszkhoz tartozó deszkriptorokat tartalmazza)*
- *IDT = Megszakítás (kivétel) Deszkriptor Tábla*

A védelmi rendszer 4 fajta kaputípust kezel. Ezek a Call, a Megszakítás, a Trap és a Taszk kapu. A CALL kapuk a taszkok közötti paraméterátadásra szolgálnak, a megszakítás, trap és taszk kapukkal a hardver megszakításokat, a processzor által detektált kivételeket (faults, traps, aborts) és a programvezérelt megszakításokat (INT, IRET utasítások) lehet kezelni. A vezérlésnek kapukon történő „áthaladása" során a privilégizálási szint (PL) mindig ellenőrzésre kerül.

4.7. HÁTTÉRTÁRAK

A központi egységben található belső memóriák kapacitása nem elegendő az összes felhasználói program és az operációs rendszerhez tartozó rendszerprogramok, valamint az ezekhez tartozó adatok tárolására. Ezért a tárolóhierarchiában fontos szerepük van a nagy kapacitású háttértárolóknak is.

A külső tárolóknak a tárolóhierarchiában alapvetően két feladatuk van:

- *a belső memória kiterjesztését szolgáló virtuális tár kezelése, mely az aktív (futtatás alatt álló) taszkokhoz tartozó programokat és ezek adatait tárolja,*
- *az aktuálisan nem aktív programfolyamatok programjainak és az ezekhez tartozó adatállományoknak a tárolása. Ebbe beleértjük a programok és adatállományok biztonsági másolatainak tárolását is.*

Egy külső tárolóegysége a tárolóhierarchiának *a következő részegységekből épül fel:*

- *magából az adathordozóból, mely az adatokat és a programokat tárolja. Ezek az adatok rögzítésének fizikai megoldását tekintve lehetnek mágneses vagy optikai tárolók;*
- *az író/olvasó eszközből, mely az adatokat az adathordozóról beolvassa, vagy kiírja;*
- *a vezérlő egységből, mely a külső tároló működését szervezi, irányítja.*

A háttértárolók az adatok visszakeresése szempontjából két kategóriába sorolhatók:

- közvetlen hozzáférésű eszközök, melyek a keresett adatot tartalmazó memóriablokkot közvetlenül meg tudják címezni és az adatot a teljes tároló végigolvasása nélkül közvetlenül ki tudják olvasni vagy írni,

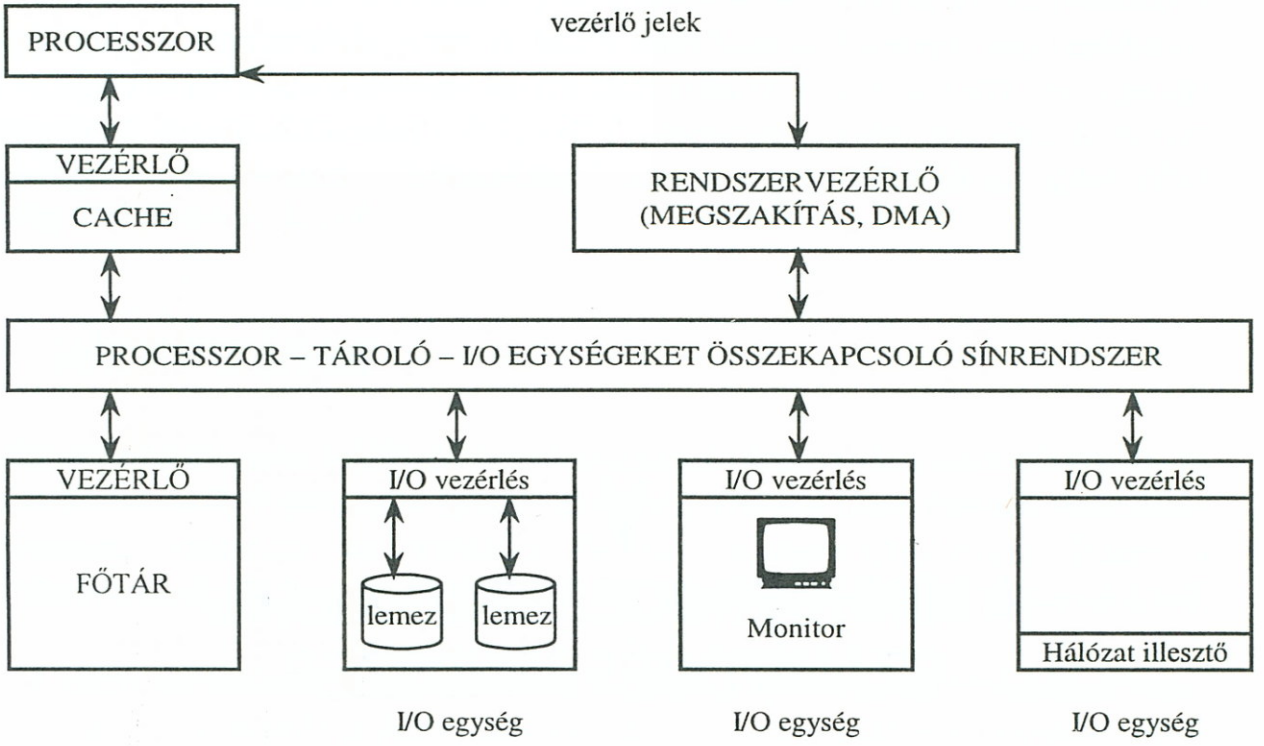
V.
A MIKROPROCESSZOR ALAPÚ SZÁMÍTÓGÉPRENDSZER

5.1. A SZÁMÍTÓGÉP LEGFONTOSABB RÉSZEGYSÉGEI

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:

- *a processzor,*
- *az L2 cache,*
- *a főtár,*
- *a megszakításrendszer,*
- *a közvetlen memóriahozzáférési (DMA) rendszer,*
- *a perifériák,*
- *a háttértárak,*
- *a busz vagy sínrendszer.*

Ezekből a részegységekből a számítógép architektúrájától függően egy konkrét gép központi egységébe egy vagy több is beépítésre kerülhet (lásd 87. sz. ábra).



87. sz. ábra
A mikroprocesszor alapú rendszer felépítése

A processzorral, a tárkezeléssel az előző fejezetekben már részletesen foglalkoztunk, az elkövetkezendőkben a többi részegységgel kapcsolatos alapfogalmakat, felépítésüket és működésük lényegét tárgyaljuk.

Ezek a részegységek lehetnek aktív vagy master (mester) eszközök, illetve működhetnek passzív vagy slave (szolga) eszközként. Az aktív eszköz a kezdeményező, a passzív eszköz csak fogadja és végrehajtja az aktív eszköztől származó vezérléseket.

5.2. A MEGSZAKÍTÁSI RENDSZER ÉS A MEGSZAKÍTÁS-VEZÉRLŐ

A számítógépek különböző részegységei működésének összehangolásában az egyik legfontosabb szerepe a megszakítási rendszernek van. Ha nem lenne megszakítási rendszer, akkor például a processzornak állandóan ellenőriznie kellene, hogy a felhasználó nem nyomott-e le egy billentyűt a bil-

lentyűzeten. Ez pedig, ha belegondolunk, hogy a processzor és az ember műveletvégző-sebessége több nagyságrenddel eltér, igen nagy pazarlás lenne a processzor teljesítményével.

5.2.1. Megszakítást kiváltó események és kezelésük

A megszakítási rendszer kiépítését az teszi szükségessé, hogy a számítógépes programok végrehajtása során *felléphetnek olyan események, melyek kezelése csak az utasításvégrehajtás „normális" menetének átmeneti felfüggesztésével lehetséges.* Ezek bekövetkezhetnek:

- *meghatározott programhibák esetén* (így például a program egy aritmetikai műveletben 0-val akar osztani);
- *meghatározott műveletek befejezésekor, melyek bekövetkezésére számítani lehet, de ezek időpontja pontosan nem tervezhető* (erre tipikus példa az, amikor egy periféria jelzi, hogy egy input vagy output műveletet befejezett);
- *szándékosan, azaz programvezérelt módon* (amikor például a programhibák felderítése céljából a programfutást lépésenként megszakítjuk annak érdekében, hogy megtekintsük a tárolótartalmakat);
- *teljesen véletlenszerűen és váratlanul (ilyen például egy súlyos hardver hiba vagy áramkimaradás).*

Az utasítások szabályszerű feldolgozását megszakító eseményeknek két alapvető típusa van:

- *a programfutáshoz képest külső eredetű megszakítások (interrupt)* (például egy I/O eszköz adatátviteli igényének jelzése);
- *az utasítások szabályszerű végrehajtását megakadályozó kivételek (exception), melyet a processzor egy utasítás végrehajtása során észlel.*

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

Mivel a megszakítások és kivételek fellépése a programvégrehajtás szempontjából véletlenszerű, nem tervezhető, ezért előfordulhat, hogy egy megszakítás kiszolgálásának ideje alatt szintén bekövetkezik egy megszakítást igénylő esemény. El kell tehát dönteni, hogy ebben az esetben a megszakítást a processzor engedélyezi vagy tiltja. Ebből a szempontból tehát *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 kivételek esetében a kiváltó eseményt kezelő különleges programrész lefutása után a processzor általában ismételten megkísérli a megszakított utasítás végrehajtását. Ezek általában több altípusra bonthatók. Például a *Pentium processzorcsaládban a processzor által detektált kivételek, lehetnek*

- *a hibák (faults), mely megengedi az utasítás újraindítását, a kivételkezelő rutin visszatérési címe a hibát kiváltó utasításra mutat;*
- *csapdák (traps), amelyet a processzor az utasításhatáron jelent, azt az utasítást követően, amelyben a kivételt észlelt;*
- *abort-ok (aborts), amely nem engedélyezi a kivételt kiváltó program folytatását;*
- *programozott kivételek, melyeket például az INT és az IRET utasítások generálnak.*

5.2.2. A megszakítások, kivételek kiszolgálása

A megszakításkiszolgálást gyakran aszinkron alprogram-feldolgozásnak is szokták nevezni. Ennek oka

- aszinkron a feldolgozás, mert időben nem megjósolható, hogy egy megszakítást kiváltó esemény (például egy billentyűzet leütése) mikor fog megtörténni,
- alprogram a feldolgozás, mert egy meghatározott típusú megszakítás mindig azonos megszakításkiszolgáló programegység („szubrutin") végrehajtását fogja eredményezni.

A megszakítások és kivételek kiszolgálásának lépéseit egy példán keresztül mutatjuk be, melyet *az IBM-PC kompatíbilis gépek* alkalmaznak. Ezek *a megszakításkérelmeket ún. vektoros módon dolgozzák fel, ami azt jelenti, hogy a megszakításkérelem a megszakításkiszolgáló rutin kezdőcímét egy vektor elemeként azonosítja a processzor számára.* (Azaz a megszakítást vezérlő a processzornak a vektor egy elemét meghatározó sorszámot ad át, mely a memóriában található megszakítási vektortáblában kijelöli a kiszolgáló rutin címét.)

Ebben az esetben a megszakítás kiszolgálásának lépései a következőkből állnak:

A hardver által

- *az eszközvezérlő beállítja a megszakításkérő vezérlő vonal 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 vektor elemének sorszámát;*
- *a processzor elmenti a verembe az utasításszámláló és az állapotregiszter tartalmát;*
- *a processzor a megszakítási vektor elemsorszáma alapján a megszakításkiszolgáló rutin kezdőcímét betölti az utasításszámláló regiszterbe és ezzel megkezdődik a megszakításkiszolgáló rutin végrehajtása.*

Az operációs rendszer által

- *a megszakított program adatainak elmentése verembe* (ha szükséges);
- *a megszakítás okának behatárolása;*
- *a kiszolgáláshoz szükséges adatok összegyűjtése;*
- *a megszakítást okozó esemény kezelése;*
- *a megszakított program adatainak visszatöltése;*
- *a megszakításkiszolgáló rutin befejezésének jelzése.*

A hardver által

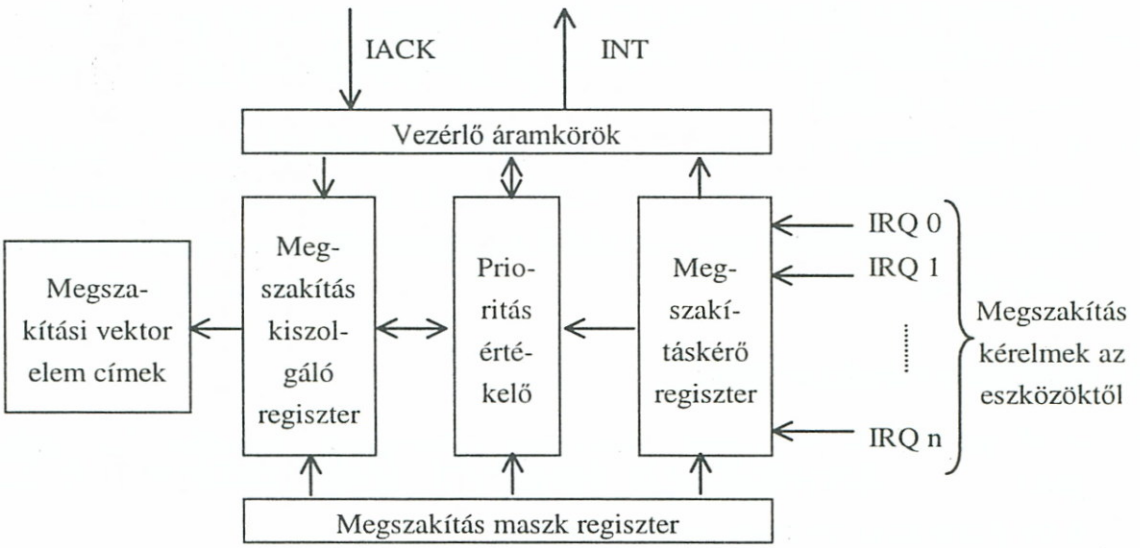
- *az elmentett állapot és utasításszámláló regiszter tartalmának visszatöltése és a megszakított program folytatása.*

5.2.3. A megszakításvezérlő felépítése és működése

A számítógépek megszakításvezérlő egysége végzi a megszakítás kiszolgálásához szükséges *legfontosabb hardver feladatokat:*

- *fogadja megszakításkérő vezérlővonalakon* (IRQ) *a megszakításkérelmeket,*
- *vizsgálja, hogy az igényelt megszakítás nincs-e maszkolással letiltva,*
- *vizsgálja és értékeli a megszakítás prioritását,*
- *az INT (Interrupt) vezetéken közli a megszakítás kérést a processzorral,*
- *ha az IACK (Interrupt Acknowlegment) vezetéken a processzor visszaigazolja, hogy kész a kérés fogadására, akkor a megszakításvezérlő átadja a processzornak a megszakításhoz tartozó megszakításvektor címet.*

A megszakítás vezérlő elvi felépítését mutatja be a következő ábra:

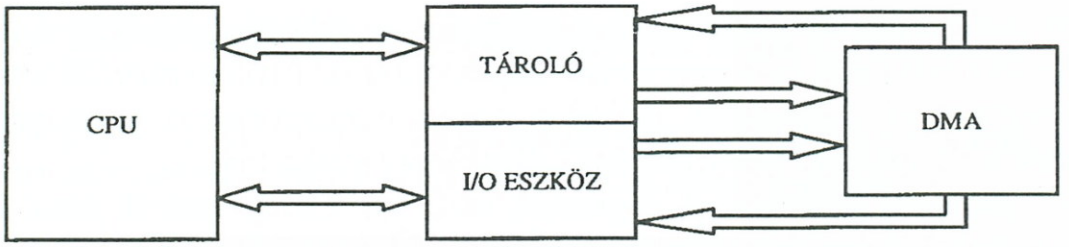


88. sz. ábra
A megszakításvezérlő blokksémája

Amennyiben egy számítógépben több megszakításvezérlő is megtalálható, akkor ezek legtöbbször master–slave kapcsolatban állnak egymással. (Ekkor például a master egy megszakításkérelmet átirányíthat a slave-nek.)

5.3. A KÖZVETLEN MEMÓRIAHOZZÁFÉRÉS (DMA) ÉS VEZÉRLŐJE

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



89. sz. ábra
A DMA működésének elve

DMA üzemmódban például 64 Kbájt nagyságú adatblokkok esetében az adatátvitel sebessége kb. hatszor akkora, mintha az I/O művelet a processzor irányításával kerülne végrehajtásra.

A processzor és a DMA vezérlő egymással a kapcsolatot a megszakítási rendszeren keresztül tartja, így például a DMA vezérlő egy I/O művelet befejezését egy megszakításkérelemmel jelzi.

A DMA átvitelt az I/O eszközök DREQ0, DREQ1 ... DREQn (DMA REQuesti) vezérlő vonalakon kezdeményezhetik. Ezekhez prioritás van hozzárendelve, ami szerint a DMA vezérlő rangsorolja az adatátviteli igények kiszolgálását. A DMA vezérlő a processzornak az adatátviteli igényt a HRQ (Hold ReQuest) vezetéken jelzi, melyet az a HLDA (HoLD Acknowledge) vezetéken engedélyez.

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

Ezenkívül 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.

5.4. AZ INPUT/OUTPUT ESZKÖZVEZÉRLŐK

Az adatbeviteli és kiviteli eszközök és vezérlésük rendkívül sokszínű, e fejezetben csak az általánosan érvényes jellegzetességeket fogjuk tárgyalni.

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.*

A processzor az eszközvezérlőket alapvetően két módon irányítja:

- ***közvetlen I/O utasításokkal*** (miután az állapotregiszter lekérdezésével megállapította, hogy az eszköz az utasításvégrehajtására képes állapotban van), *a parancsregiszter beállításával és a pufferregiszterek írásával vagy olvasásával,*
- ***közvetett módon, amikor a címzés úgy történik, mintha az I/O eszköz tárolója a főtár része lenne*** *(memory mapped addressing, ilyen például a grafikus memória).*

5.5. KOMMUNIKÁCIÓS KAPCSOLATOK A SZÁMÍTÓGÉP KÖZPONTI EGYSÉGEINEK RÉSZEI KÖZÖTT, SÍNRENDSZEREK

A számítógép részegységei közötti kommunikációs kapcsolatokat (adatok, címek, valamint a gép vezérléséhez szükséges információk átvitelét) a sín vagy buszrendszer biztosítja.

A sínrendszer a kapcsolatokat biztosító kommunikációs vezetékek mellett aktív és passzív elektronikai elemeket is tartalmaz, így hozzátartoznak a sínvezérlő áramkörök is, melyek meghatározott, szabványosított algoritmusoknak megfelelően működnek.

Ezért a sínrendszer fogalmába beleértjük a kommunikációs kapcsolatok szabványosított szabályait is, amit sínprotokollnak (bus protocol) nevezünk. A sínprotokollban meghatározott szabályrendszer fizikailag a buszvezérlő hardver egységben testesül meg.

5.5.1. 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, valamint*
  - *a sínvezérlőjelek* (például a sínhasználat kérése és ennek visszaigazolása),
  - *szinkronizációs jelek.*

5.5.2. A sínrendszerek típusai

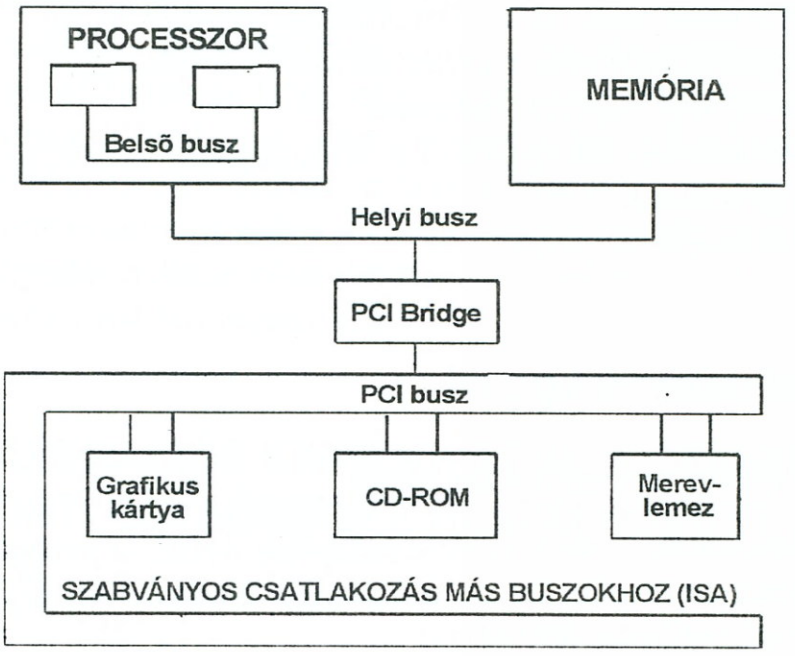
A sínrendszerek között megkülönböztetünk

- *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).*

Ezt az architektúrát továbbfejlesztette a kb. 180 gyártó által elfogadott *PCI (Peripheral Component Interconnector) kvázi szabvány. Ennél a helyi sín és a PCI busz közé egy processor-PCI Bridge-t iktattak be, mely lehetővé tette, hogy a buszrendszer a konkrét processzortól és annak sebességétől függetlenül is működőképes legyen. Ez azt jelenti, hogy a PCI busz nem kötődik szorosan egy processzorhoz, hanem többfajta processzorral is képes együttműködni (lásd 90. sz. ábra).*



90. sz. ábra
A PCI busz logikai felépítése

5.5.3. A sínrendszer működésével kapcsolatos alapfogalmak

A buszt, mint architekturális hardver építőelemet a rákapcsolt eszközök közösen használják. Ezzel kapcsolatosak a busz rendszer használatának előnyei és hátrányai:

- Előnyök:
  - Új egységek egy szabványos illesztőhelyen (pl. PC slot) könnyen hozzákapcsolhatók a buszhoz,
  - Költségtakarékos megoldás, mivel ugyanazt az átviteli utat több célra, több eszköz használja.
- Hátrányok:
  - A sávszélesség (időegység alatt átvitt bájtszám) korlátozott,
  - Az adatátviteli sebességet a busz hossza és a hozzá kapcsolt egységek száma korlátozza,
  - A busznak különböző sebességű eszközöket (tároló, monitor, egér stb.) kell kezelnie.

A következőkben a sínrendszerrel kapcsolatos legfontosabb fogalmak definícióját adjuk meg.

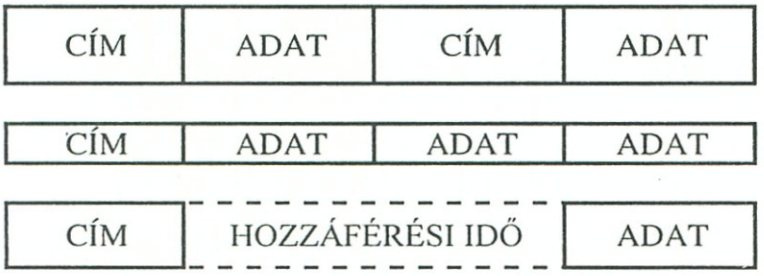
Buszciklus: *Egy adategység átviteléhez szükséges idő (a ciklusok végén jelennek meg a szükséges jelszintek).*

Busz tranzakció: *A buszigények sorozata, mely az adatátvitel igénylésétől, annak befejezéséig tart (több műveletből, buszciklusból állhat).*

Multiplexált tranzakció: *A címek és adatok egymás után ugyanazon a vezetékeken keresztül kerülnek átvitelre.* Egy busz tranzakció *lépései a következők* lehetnek:

- *sínhasználat igénylése* (REQUEST)
- *sínhasználat jogának odaítélése* (ARBITRATION)
- *címátvitel* (ADRESSING)
- *adatátvitel* (DATA TRANSFER)
- *hibafelismerés* (ERROR DETECTION)
- *a felszabadítása* (RELEASE)

! A buszrendszer segítségével megvalósított *adat átvitel fontosabb típusai*:



Egyszerű ciklikus átvitel (Single cycle transfers)

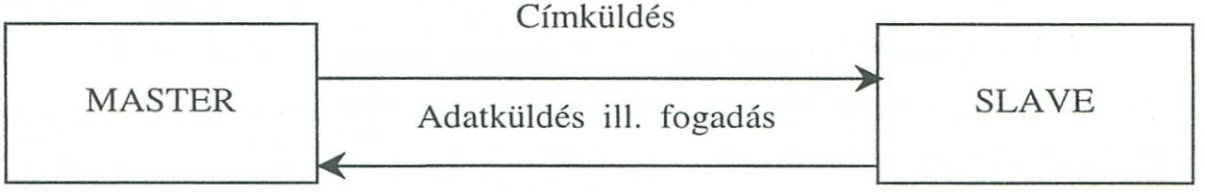
Burst átvitel (1 cím, „n" adat)

Multiplexált olvasás (read)

Master és slave

A sínt egyidőben csak egy eszközpár használhatja. Ezek közül a kezdeményező eszköz a master, a kapcsolatban résztvevő passzív eszköz pedig a slave eszköz. A master és slave közötti *munkamegosztás lényege* a következő:

- *MASTER:*
  - *Elindít és befejez egy busztranzakciót,*
  - *Címet küld.*
- *SLAVE:*
  - *Válaszol az igényekre és címekre,*
  - *A sínre teszi illetve fogadja az adatokat.*



91. sz. ábra
A master és slave eszközök feladatai

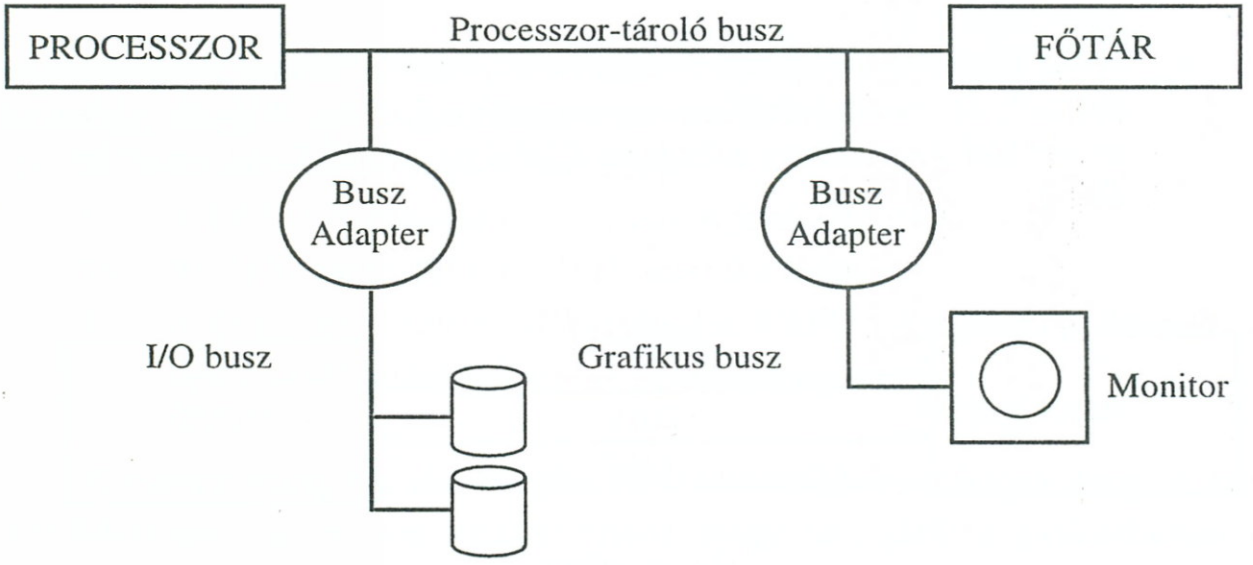
Busz arbitráció

! *Egyidejűleg több aktív (master) eszköz is igényelheti a sín használatát. Ekkor valamilyen eljárással el kell dönteni, hogy melyik eszköz lesz jogosult a sín használatára. Ezt az eljárást nevezzük sín arbitrációnak.*

Egyes busztípusok jellemzői:

- *Processzor – tároló busz*
  - *Architektúra specifikus*
  - *Rövid, nagy sebességű*
- *I/O busz*
  - *Ipari szabvány,*
  - *Hosszabb és lassúbb a processzor – tároló busznál és több különböző egységet szolgál ki. Busz adapterrel kapcsolódik a processzor – tároló buszhoz.*
- *Processzor – tároló – I/O eszközök egy buszon*
  - *Olcsó*
  - *Lassú (összességében)*
  - *Ma már ritka*

Ha a processzor – memória tranzakciókat el akarjuk választani az I/O műveletektől, akkor a processzor tároló buszt egy busz adapterrel kell illeszteni az I/O buszhoz (lásd 92. sz. ábra).



92. sz. ábra
I/O buszok csatlakoztatása a processzor-memória buszhoz

A sínhez csatlakozó eszközök címzése

Alapszabály, hogy mindig a master címzi a slave-t. Ez a címzés lehet:

- *helyfüggetlen (minden eszköznek egyértelmű logikai címe van),*
- *memory mapped addressing (az eszköz címzése úgy történik, mintha a főtár része lenne),*

- *broadcast (az összes slave megcímzése egyidejűleg pl. inicializáláskor).*

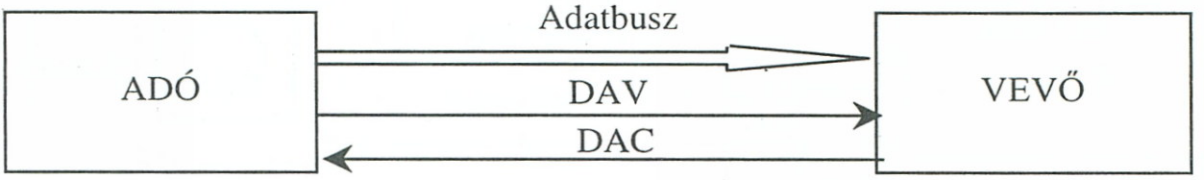
A mikroszámítógépeknél master általában a processzor vagy valamilyen DMA-t használó I/O eszköz lehet, a memória viszont mindig slave szerepkört tölt be a sínhasználat során.

Handshaking

A sínrendszeren történő adatátvitel hibátlan lefutásának biztosítása az egyik legfontosabb szempont. E témakörbe tartozik, hogy a buszon kommunikáló „ADÓ" egységnek valamilyen formában meg kell bizonyosodnia az elküldött adatcsomag megérkezéséről a „VEVŐ" egységhez. (Ez a szituáció hasonlítható ahhoz, amikor egy fontos levelet feladunk a postán és szeretnénk tudni, hogy azt a címzett rendben megkapta-e).

E célra szolgál az *„ADÓ" és „VEVŐ" egység közötti „kézfogás" vagy handshaking eljárás.* Ennek egyszerűsített változatának logikai *lépései a következők:*

- *az „ADÓ" egy vezérlőjellel informálja a „VEVŐ"-t, hogy az érvényes (elküldendő) adatokat az adatsínre helyezte* (DAV = data valid vezérlőjel = 1)
- *a DAV jel 1-es állapotát érzékelve a „VEVŐ" olvassa az adatbusz adatait, és ennek hibátlan megtörténtét egy DAC = Data Accepted vezérlőjel 1-es szintre állításával jelzi az „ADÓ"-nak* (lásd 93. sz. ábra).

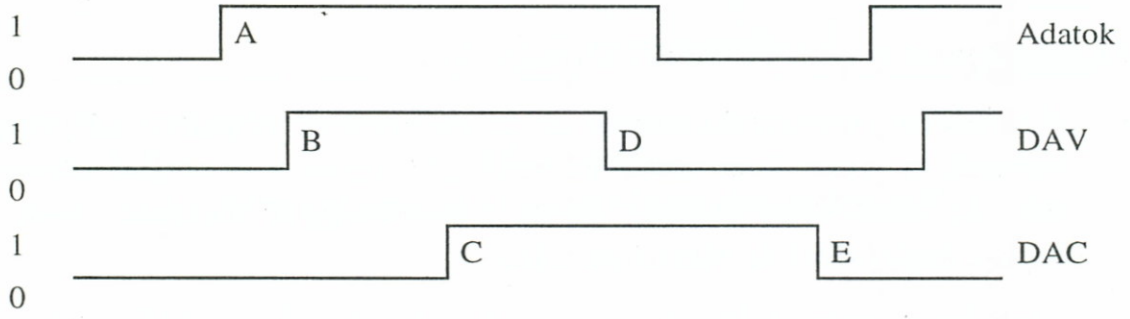


93. sz. ábra
Az egyszerű handshaking eljárás

Az egyszerűtett handshaking-nek az a hiányossága, hogy az „ADÓ" nem rendelkezik információval az eljárás folytathatóságáról (azaz arról, hogy sínre teheti-e a következő adatbájtot). Ezért a gyakorlatban legtöbbször az egyszerűsített handshaking továbbfejlesztését, a kétszeres kézfogás (fully interlocked handshaking) eljárását alkalmazzák.

Fully Interlocked Handshaking

Ennél az adatátvitel következő lépésére csak akkor lehet rátérni, ha az előző adatátvitel igazoltan, hibátlanul befejeződött. Ezt az jelzi, hogy a DAV és a DAC vezérlőjel egyidejűleg újra a 0-szintre áll be (lásd 94. sz. ábra).



94. sz. ábra
A fully interlocked handshaking jelszintjei

A kétszeres kézfogásos eljárás lépései tehát a következők:

A/ *Az ADÓ az első adatbájtot a sínre helyezi*
B/ *Az ADÓ a DAV jelet 1-re állítja*
C/ *A VEVŐ visszaigazolja az adatok átvételét (DAC = 1)*
D/ *Az ADÓ visszaállítja a DAV jelet 0-s szintre*
E/ *A DAC jel 0 szintre állításával a VEVŐ jelzi, hogy felkészült az újabb adat fogadására*

A handshaking eljárást megfelelő áramkörökkel valósítják meg. A kettős kézfogás algoritmusa viszont ahhoz, hogy az eljárás a gyakorlatban is működjön még önmagában még kevés. Gondoljunk arra az esetre, hogy *mi történik akkor, ha az ADÓ DAV vezérlőjelét a VEVŐ huzamosabb idő elteltével sem igazolja vissza. Ezt a problémát a korszerű számítógéprendszerekben úgy oldják meg, hogy a DAV jellel egyidőben egy számláló (Timer) is aktivizálásra kerül. Ezzel figyelhető, hogy a DAV jel visszaigazolása egy meghatározott időn belül megtörténik-e, és ha nem akkor ez egy megszakítást eredményez.*

5.5.4. A buszprotokoll és a sínvezérlés formái

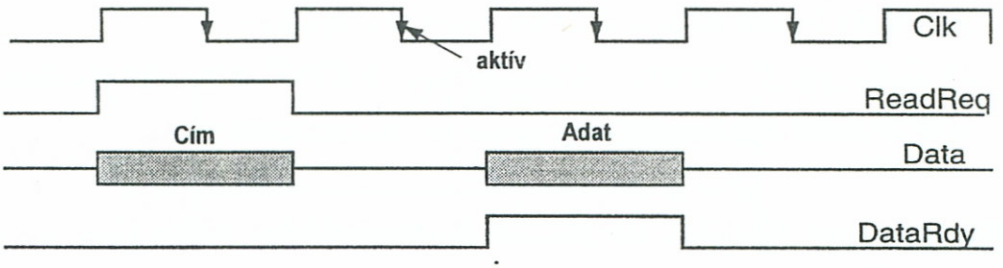
Az adatátvitelben résztvevő eszközöknek összehangoltan kell működniük. Ehhez megfelelő konvenciók és algoritmusok kellenek, melyek áramkörökben, elektromos jellemzőkben öltenek testet.

A sínen történő adatátvitelben résztvevő eszközök együttműködésének szabályait buszprotokollnak nevezzük.

Az adatátvitel vezérlésének két formája lehet:

- *A szinkron sínvezérlés esetén az eseményeknek rögzített időpontjaik vannak, a sínen kommunikáló eszközök azonos órajellel ütemezettek. Az adás-vétel mindig azonos sebességgel történik, nem kell kapcsolatfelvétel és visszaigazolás.*
- *Aszinkron sínvezérlésnél az események tetszőleges időpontban bekövetkezhetnek. Ezért a sínre csatlakozó eszközök zavartalan együttműködéséhez egy kapcsolatfelvétel és vétel visszaigazolási eljárás (handshake) szükséges.*

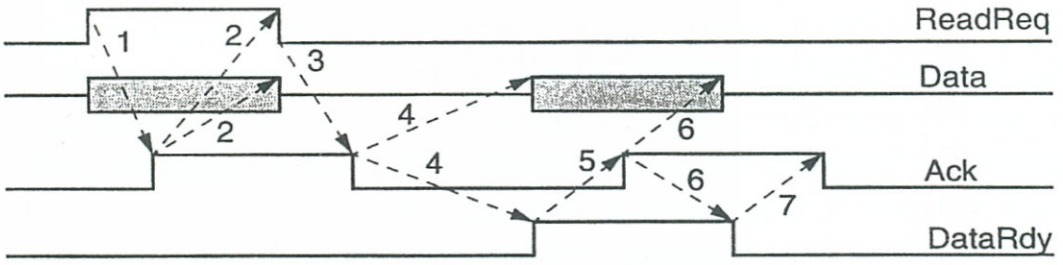
A szinkron sínvezérlés előnye gyorsasága, hátránya viszont, hogy közös órajelet kell biztosítani az összes sínre kapcsolt eszköz számára. Egy memóriából történő olvasási tranzakcióra szinkron sínvezérlésnél mutat példát a 95. sz. ábra.



95. sz. ábra
A memóriából történő olvasás szinkron sínvezérléssel

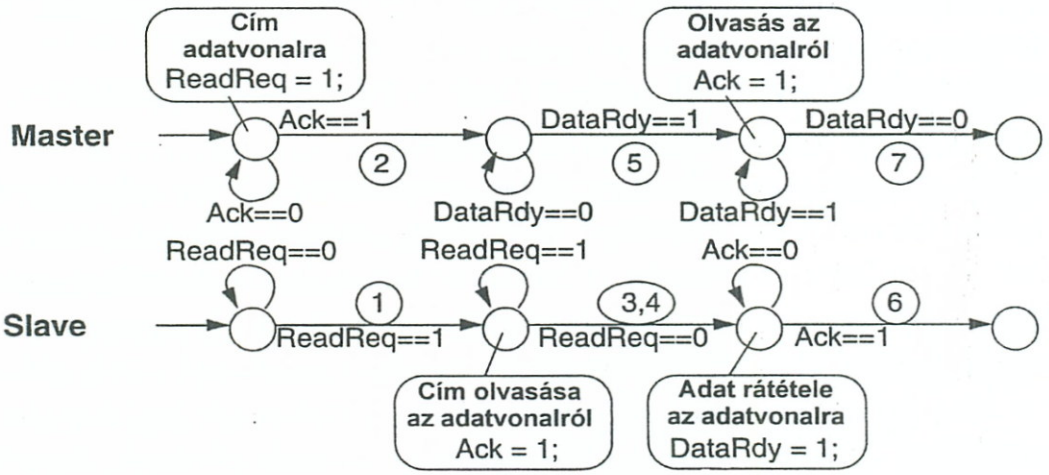
Az aszinkron sínvezérlés előnye, hogy nem kell azonos órajelet biztosítani az összes sínre kapcsolt eszköz számára, és nagyon eltérő sebességű eszközök kiszolgálását is lehetővé teszi. Hátránya viszont az aszinkron sínvezérlésnek, hogy a protokollba be kell építeni a relatíve bonyolult aszinkron „handshake" eljárást.

Egy olvasási tranzakcióra aszinkron sínvezérlésnél mutat példát a 96. sz. ábra.



96. sz. ábra
Olvasási tranzakció aszinkron sínvezérlésnél

Az aszinkron protokollba épített kérés-visszaigazolást (handshake) a 97. sz. ábra mutatja be.



97. sz. ábra
Az aszinkron sínvezérlés handshake eljárása

A buszon megvalósított adatátvitelnek két alapeljárása van:

- *Blokkátvitel*
  - *Egy tranzakcióban több szó átvitele történik meg*
  - *A címet csak egy alkalommal kell kiküldeni*
  - *A busz csak akkor kerül felszabadításra, ha az utolsó szó is átvitelre került*

- *Megosztott átvitel (példa egy I/O eszköz adatot olvas a tárolóból)*
  - *Az I/O eszköz küldi az adatátviteli igényt és a címet, azt a tároló visszaigazolja, azt követően a busz felszabadításra kerül.*
  - *A tároló jelzi, hogy az adatok átvitelre készen állnak. Az I/O eszköz visszaigazolja az adatok átvételét, majd a busz felszabadításra kerül.*

Fontos megjegyezni, hogy a korszerű buszokon a vezérlés protokollja, akár menet közben is megváltozhat. Példák erre:

- Dinamikus buszszélesség változtatás (pl.: 32 bites adat olvasása 8 bites portról 4 részletben),
- Megosztott és blokkos átvitel váltott alkalmazása.

5.5.5. Busz arbitráció

Ha csak egy buszmaster van, akkor ez az eszköz (legtöbbször a processzor) felügyeli a buszhasználatot, inicializálja és ellenőrzi a buszfoglalást. A slave eszközök válaszolnak az írási és olvasási igényekre.

Ha több buszmaster van, akkor csak arbitrációval dönthető el, hogy melyik eszköz kapja meg a buszhasználat jogát.

A buszhasználat jogának megosztása a masterek között a legegyszerűbb módon időosztással (time-sharing) történhet, amikor minden master meghatározott időszeletre megkapja a buszhasználat jogát (statikus módszer). Csak akkor hatékony, ha a masterek adatátviteli igénye kb. azonos.

Dinamikus buszhasználat megosztás esetén a masterek csak akkor kapják meg a sínt, ha azt igénylik. Ez esetben azt a problémát kell kezelni, hogy a buszhasználati igények azonos időpontban is jelentkezhetnek. Ennek módszerei lehetnek:

- *prioritások meghatározása a masterek számára* (ez azzal a veszéllyel jár, hogy a magas prioritású és gyors eszközök pl. processzor mellett a többi master „nem jut szóhoz"),
- *egyenletes buszhasználati jog elosztás* (például a masterek igényei jelentkezésük időpontjai szerint egy várakozó sorba lesznek besorolva). Ennek az a veszélye, hogy a fontosabb eszközök hosszú ideig várakozásra kényszerülnek.

Gyakorlatban a mikroszámítógépeknél előző két változatot kombináltan alkalmazzák.

A másik fontos kérdés, hogy ha egy master lefoglaljå a buszt, akkor mennyi időre kapja meg a buszhasználat jogát.

A lefoglalt busz felszabadításának legfontosabb módszerei a következők:

- *release on request: a master annyi időre lefoglalja a buszt (akkor is, ha adatot nem forgalmaz), amíg a buszt másik master nem igényli. Ezt alkalmazzák leggyakrabban mikroszámítógépek buszrendszereiben.*
- *release when done: a master egy tranzakcióra kapja meg a buszhasználat jogát, annak befejezése után felszabadítja a buszt.*
- *preemption: ha egy magasabb prioritású master jelentkezik, ez a tranzakciót megszakítja. Ezt blokküzemmódban van értelme alkalmazni.*

Dinamikus buszhasználat szétosztás csak akkor lehetséges, ha

- *minden master jelzi a buszfoglalási igényt,*
- *egy master csak akkor használhatja a buszt, ha igényének elfogadását visszaigazolják,*
- *a masternek jelzést kell küldenie, ha a tranzakció befejeződött.*

Ebből is nyilvánvaló, hogy léteznie kell *egy olyan hardver egységnek, mely a sínfoglalási kérelmeket fogadja, elbírálja és visszaigazolja. Ezt az eszközt busz arbiternek nevezzük.*

A busz arbitráció lehet:

- *központosított, ekkor a számítógéprendszerben csak egy arbiter van. Ez lehet egy önálló hardver egység, vagy egy másik hardver eszköznek* (legtöbbször a CPU-nak) *valamilyen részegysége.*
- *szétosztott, akkor a számítógéprendszerben több arbiter hardver egység található* (Ez a multiprocesszoros rendszerek esetén fordul elő.).

Központosított busz arbitráció esetében a buszhasználatot igénylő masterek egy „request" jellel jelzik az igényeiket az arbiternek, mely az arbitrációs algoritmus szerint kiválaszt egy mastert és az igény lefogadását a „grant" jellel igazolja vissza.

Ennek három módszere lehetséges:

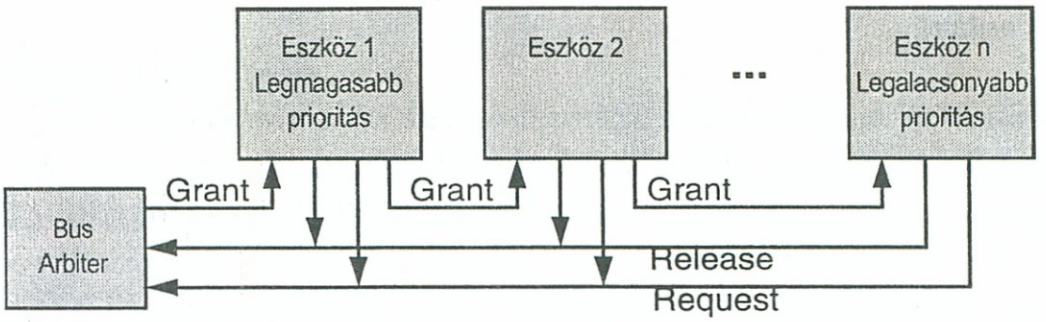
- *soros sínfoglalási eljárás, amikor egy közös „request" vezérlővonala van a mastereknek és ezek egy felfűzött* (daisy-chain) *„grant" vezérlővonallal rendelkeznek,*

- *párhuzamos sínfoglalási eljárás, mikor a masterek egymástól független „request" és „grant" vezérlővonallal rendelkeznek,*
- *az előző két módszer kombinált alkalmazása.*

A soros és párhuzamos sínfoglalás

Soros kiszolgálás esetén az eszközök sorba vannak kötve és sorrendjük határozza meg prioritásukat, azaz azt, hogy mikor kaphatnak jogosultságot a sín használatára.

A soros (daisy-chain) arbitrációra mutat példát a 98. sz. ábra.



98. sz. ábra
A soros buszfoglalás elve

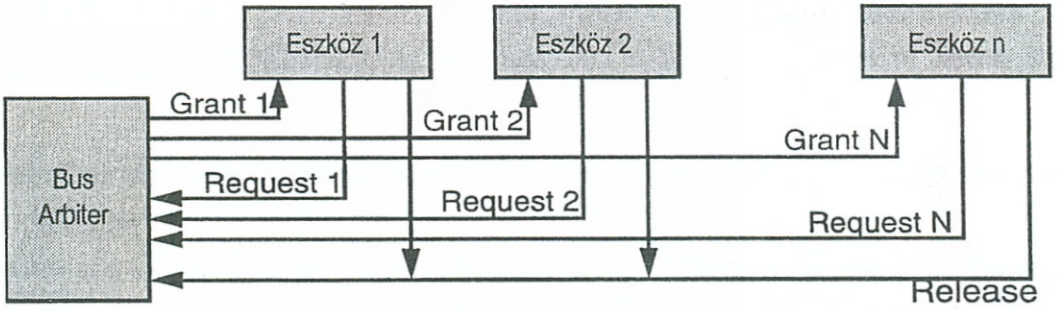
A buszprotokoll lefutása soros sínfoglalás esetén 1. Request; 2. Grant; 3. Bustransaktion; 4. Release.

A soros sínfoglalás előnye egyszerűségében van. Hátránya, hogy a kis prioritású eszközök sokáig várhatnak.

Párhuzamos kiszolgálás esetén minden sínhasználatért folyamodó eszköz önálló kérő és engedélyező vezérlővonallal rendelkezik és a sínvezérlés prioritás szerint engedélyezi a sín igénybevételét. A prioritás meghatározása különböző eljárások szerint történhet, például:

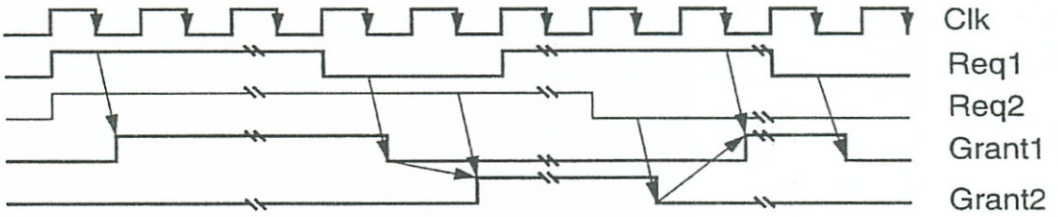
- *az egyszerű körbejáró eljárásnál minden sínhasználatot követően a korábban legalacsonyabb prioritású eszköz kapja meg a legmagasabb prioritást, az összes többi eggyel alacsonyabb prioritási szintre kerül,*
- *az LRU eljárásnál az az eszköz kapja meg a sínhasználat jogát, mely a sínt legrégebben vette igénybe.*

A párhuzamos (központosított) arbitrációra mutat példát a 99. sz. ábra.



99. sz. ábra
A párhuzamos sínfoglalás elve

A buszprotokoll (szinkron) lefutása a 100. sz. ábra szerint történik.



100. sz. ábra
Párhuzamos sínfoglalás protokollja szinkron sínvezérlésnél

5.5.6. Sínrendszerek a gyakorlatban

Először a sínrendszer teljesítménye és ára közötti összefüggéseket mutatjuk be röviden a 101. sz. ábra szerint.

Megoldás	Nagy teljesítmény és ár	Kisebb teljesítmény és ár
Szinkronizáció	Szinkron	Aszinkron
Buszszélesség	Elkülönített adat és címvezetékek	Multiplexált adat és címvezetékek
Adatátvitel	Sok bájt átvitele egy busz-ciklus alatt	Kevesebb bájt átvitele egy buszciklus alatt
Blokknagyság	Több szóátvitele tranzakciónként	Egy szó átvitele tranzakciónként
Master	Több master (arbitráció)	Egy master
Megosztott átvitel	Elválasztott sínfoglalás és adatátvitel	Sínfoglalás és átvitel egy tranzakcióban

101. sz. ábra
A sínrendszer ára és teljesítménye

Az elkövetkezendőkben néhány példát mutatunk be konkrét buszrendszerekre, ezek jellemző adatait a 102. és 103. sz. ábrák tartalmazza.

	S Bus	Micro-Channel	PCI	SCSI 2	
Adatvonalak	32 bit	32 bit	32/64 bit	8/16 bit	
Masterek száma	több	több	több	több	
Órajel	16–25 MHz	asynchron	33 MHz	10 MHz	asynchron
Adatátvitel (szavas)	33 Mbájt/s	20 Mbájt/s	33 Mbájt/s	20 Mbájt/s	6 Mbájt/s
Adatátvitel max	89 Mbájt/s	75 Mbájt/s	111 Mbájt/s	20 Mbájt/s	6 Mbájt/s

102. sz. ábra
I/O buszok

A CPU-Memória buszokra a 100. sz. ábra mutat be példákat.

	HP Summit	SGI Challenge	Sun XDBus
Adatvonalak	128 bit	256 bit	144 bit
Órajel	60 MHz	48 MHz	66 MHz
Masterek száma	több	több	több
Adatátvitel max	960 Mbájt/s	1200 Mbájt/s	1056 Mbájt/s

103. sz. ábra
CPU-Memória buszok

5.6. AZ I/O MŰVELETEK VÉGREHAJTÁSA MIKROSZÁMÍTÓGÉPRENDSZEREKBEN

5.6.1. Az I/O műveletekkel kapcsolatos alapfogalmak

Egy számítógéprendszer teljesítményének megítélésében nagyon fontos szempont, hogy *milyen jellegű periféria műveletek kiszolgálását kell biztosítania.* Az input/output igényeket a következő típusokba sorolhatjuk be:

- *Tranzakcioorientált I/O jellemzője a sok, kis változtatás nagy adathalmazokban* (pl.: bankautomaták, repülőgéphelyfoglalás). A számítógéprendszer *teljesítménye szempontjából a legfontosabb az időegység alatti hozzáférések száma.*
- *Fájl I/O jellemzője a soros hozzáférés nagy adathalmazokhoz. Emiatt a legfontosabb az időegység alatt átvitt adatmennyiség.*
- *Eseményorientált I/O esetében a számítógépes rendszernek külső eseményekre kell reagálnia* (pl.: egér helyzetének megváltoztatása, folyamatvezérlés). *Emiatt e rendszerekben a reakcióidő, valamint időegység alatti feldolgozható eseményszám a legfontosabb teljesítmény jellemző.*

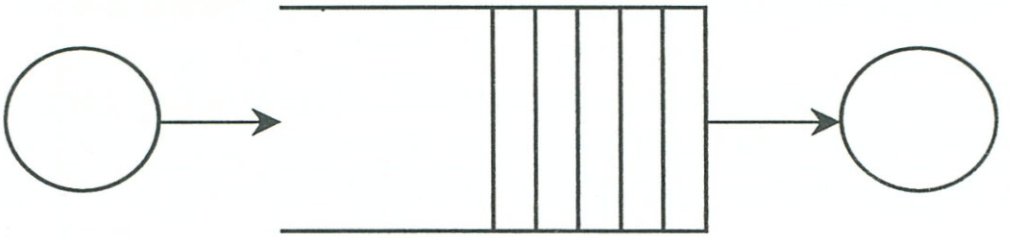
Az I/O egységeket a kapcsolat jellege és az I/O-ban a számítógéprendszer „partnerének" jellege szerint is osztályozhatjuk. Ezek szerint az I/O egységek típusai a következők lehetnek:

- A kapcsolat jellege szerint
  - csak input egységek,
  - csak output egységek,
  - I/O művelet végrehajtására egyaránt alkalmas egységek.
- Az I/O művelet résztvevője szerint
  - Ember által használt perifériák (terminál, egér),
  - Gép (Winchester, szenzor).

Az I/O teljesítménye nemcsak az I/O eszköztől függ. Ezt a processzor, a tárolókezelőrendszer, a kapcsolóhálózat (sínrendszer), az I/O vezérlés, az I/O egység, az I/O szoftver teljesítménye együttesen határozza meg. Az I/O rendszer teljesítményének legfontosabb mérőszámai

- *a válaszidő* (azaz egy I/O igényt a számítógéprendszer mennyi idő alatt képes kiszolgálni),
- *az adatátbocsátó képesség* (pl. Mbájt/sec-ben mérve).

! *A számítógéprendszerekben az I/O műveletek végrehajtása általában az igénylő – kiszolgáló modell szerint történik (lásd 104. sz. ábra). Gondoljunk például a PC nyomtatási várakozási sorára (SPOOL).*



104. sz. ábra
Az igénylő- kiszolgáló modell

A várakozó sorokat a processzor és az I/O eszközök közötti több nagyságrendű sebességkülönbség is szükségessé teszi (pl. egy lemezművelet végrehajtása több milliószor lassúbb a processzor utasításvégrehajtásánál).

! *Nyilvánvaló, hogy a teljesítmény szempontjából meghatározó jelentőségű a kiszolgáló feladat átbocsátóképessége, mely a kiszolgáló szerver által időegység alatt végrehajtott feladatok száma jellemez.*

Nagy átbocsátóképességet olyan „ideális" hardver architektúrákkal lehet elérni, melyek biztosítják, hogy

- *a szerver mindig dolgozzon,*
- *a várakozósor sohase legyen üres.*

5.6.2. Az operációs rendszer szerepe az input/output műveletekben

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

- *A védelmi funkció:*

  ! *A taszkok által közösen használt I/O eszközöket a felhasználói programok csak felszabadítást követően vehetik igénybe, ezt az operációs rendszernek ellenőriznie kell.* Általában multitaszking üzemmódban nem engedhető meg emiatt, hogy a felhasználói programok közvetlen kapcsolatba kerüljenek a perifériákkal . Például, ha az IBM PC Assembly-ben az I/O műveleteket programvezérelt megszakítással (INT) programozzuk, akkor az operációs rendszer eszközkezelő részének adjuk át a vezérlést.

- *Eszközspecifikus szolgáltatások.* (Így például kódkonverziók végrehajtása különböző hardver egységek között.)
- *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.

5.6.3. Az input/output eszközök címzése

A processzor az eszközöket két módon címezheti:

- *közvetlen I/O címekkel, ekkor az I/O utasítások a processzor utasításkészletében szerepelnek* (Ez például az INTEL processzorokra jellemző),

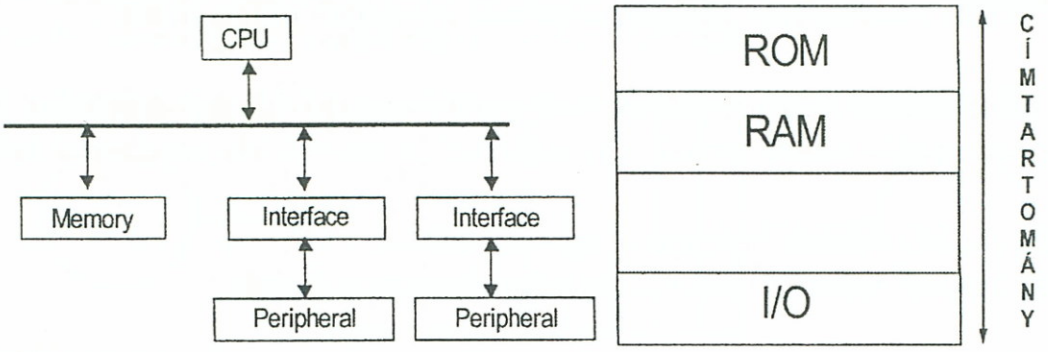
!

- *közvetett módon* (memory mapped addressing), *mely esetben a címzés úgy történik mintha az I/O eszköz tárolója a főtár része lenne* (Ez a RISC processzorokra jellemző). *A közvetett I/O eszköz címzés esetén:*

!

  - Ha az írási/olvasási cím az I/O címtartományba esik, ennek értelmezése automatikusan I/O művelet,
  - Az I/O címtartományt az operációs rendszer védi,
  - Az I/O eszközökkel a felhasználói programok az operációs rendszer közbeiktatásával kommunikálnak.

A közvetett periféria címzést mutatja be a 105. sz. ábra.



105. sz. ábra
A memory mapped címzés

5.6.4. Az I/O adatátvitel típusai

Az adatátvitel a mikroszámítógép és az I/O eszközök között a következő módon történhet meg:

- *Programozott I/O átvitellel*
  Ekkor az adatátvitel az I/O eszközök és a főtár között csak a processzor közbeiktatásával (vezérlésével) történhet meg.

- *Megszakításos I/O átvitellel, amikor a processzor jelzi az I/O eszköz számára az adatátviteli igényt, mely ha felkészült az adatátvitelre, akkor ezt egy megszakításkérelemmel jelzi a processzornak. Az adatátvitel befejezését az I/O eszköz szintén megszakításkérelemmel jelzi.* (Intel processzoroknál a megszakítási vektorban található meg az I/O eszközt kiszolgáló rutin címe.)

- *Közvetlen memóriahozzáféréssel, amikor az I/O eszköz és a főtár közötti adatátvitelt a processzortól „függetlenül" a DMA vezérlő irányítja.* A processzor feladata az I/O előkészítése (az I/O eszköz állapotának vizsgálata) és az I/O művelet hibátlan végrehajtásának ellenőrzése. (Ez esetben a processzor és a DMA vezérlő a kapcsolatot a megszakítási rendszer segítségével tartja fenn.)

- *I/O processzor alkalmazásával* (főleg mainframe-kre jellemző), *melynek a processzor átadja az I/O művelet végrehajtásához szükséges összes adatot és ezt követően az I/O processzor teljesen önállóan vezérli az I/O művelet végrehajtását.*

Az I/O műveletek végrehajtásának különböző típusait a telefonálással lehet leginkább szemléltetni. Képzeljük el, hogy egy fontos telefonhívást várunk, ekkor a következő esetek lehetségesek, ha az ember tölti be a processzor szerepkörét:

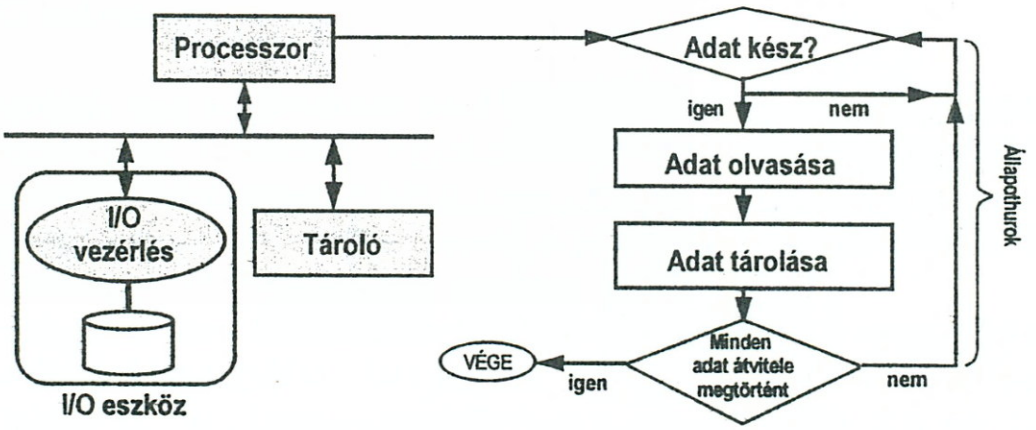
- elromlott a telefon csengője. Ekkor nem tehetünk mást, mint például azt, hogy időegységenként (pl. 10 másodpercenként) felvesszük a telefont és belehallgatunk. Ez felel meg a programozott I/O átvitelnek.
- működik a telefon csengője. Ekkor bármilyen más tevékenységet folytathatunk addig, amíg a telefon nem jelzi, hogy valaki hív bennünket. Ez felel meg a megszakításos I/O átvitelnek.
- valamilyen más fontos dolgunk akad, ami miatt nem akarunk foglalkozni a telefonnal, ezért azt üzenetrögzítőre kapcsoljuk. Ez felel meg jó közelítéssel a DMA és az I/O processzorral történő I/O átvitelnek.

5.6.4.1. Programozott I/O átvitel (Polling)

Ennek jellemzői:

- *A processzor teljeskörűen ellenőrzi és vezérli az I/O műveletet.*
- *A periféria állapot regiszter ciklikus lekérdezése folyamatosan terheli a processzort. Ezért ennek a módszernek csak akkor van értelme ha az I/O eszköz nagyon gyors.*

!

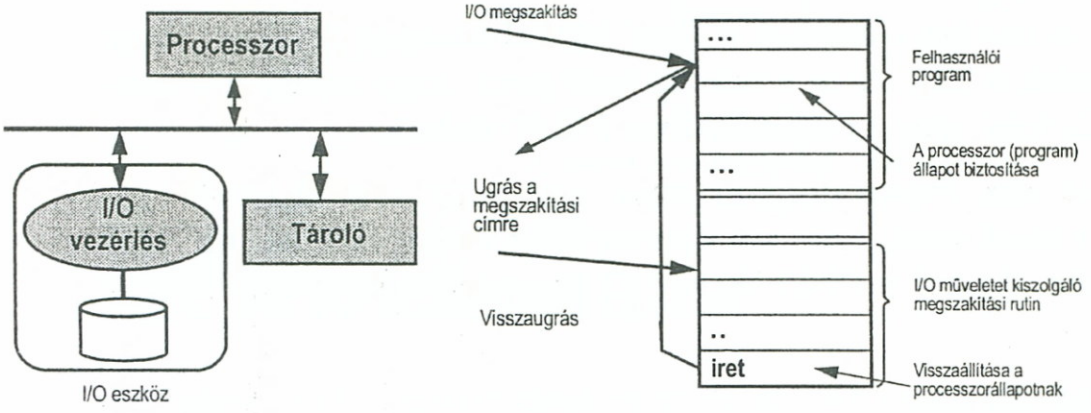


106. sz. ábra
Programozott I/O átvitel

5.6.4.2. Megszakításos I/O átvitel

Ennek jellemzői:

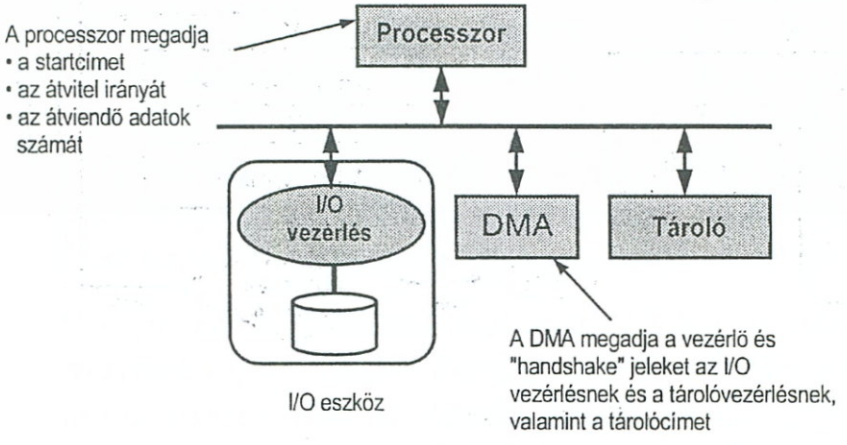
- *A felhasználói program a tranzakció idejére megállításra kerül,*
- *Speciális hardverigény: megszakítási kérelem generálása* (I/O eszköz), *megszakításfelismerés* (processzor, megszakításvezérlő), *megszakításkezelés* (processzor).



107. sz. ábra
Megszakításos I/O művelet

5.6.4.3. DMA I/O adatátvitel

A DMA által vezérelt I/O műveletek célja a processzor tehermentesítése. Ekkor a DMA a buszmaster és a processzortól „függetlenül" vezérli az adatátvitelt.



108. sz. ábra

A közvetlen memóriahozzáféréssel (DMA) végrehajtott I/O művelet lépései a következők:

- *a processzor ellenőrzi a perifériát, hogy tudja-e fogadni az átvitelt, ezt követően a DMA vezérlő részére átadja az átvitel paramétereit,*
- *a DMA buszfoglalási kérelmet jelez (DMA REQUEST), a processzor ezt visszaigazolja (DMA ACKNOWLEDGE),*
- *DMA masterként lefoglalja a buszt, végrehajtja az adatátvitelt,*
- *a DMA jelzést küld megszakítással a processzornak az átvitel befejezéséről,*
- *processzor ellenőrzi a végrehajtás hibátlan megtörténtét és a buszengedélyezést megszünteti.*

A buszhasználat megosztása a processzor és a DMA között a következő módszerekkel történhet:

- *Blokkátvitel (Blocktransfer) esetén a DMA vezérlő az adatátvitel teljes idejére lefoglalja a buszt.*
- *CiklusloPás (Cycle Stealing) esetén busz felszabadítása minden szó (bájt) átvitel után megtörténik. Ez átlapolt buszhasználatot jelent a processzor és a DMA vezérlő között. Ütközés esetén a DMA-nak prioritása van.*
- *Transzport Mode esetén a DMA vezérlő használja a buszt (például a processzor utasítás dekódolása és végrehajtása alatt) addig, amíg a processzor nem végez memóriahozzáférést. Ez a mainframe-kre jellemző.*

Összehasonlító példa a programvezérelt, a megszakításos és a DMA I/O műveletre

Vizsgáljuk meg a következő példát. Az I/O műveleti igény legyen 1000 átvitel, egyenként 1 kbájtos egységekben. Az átviteli sebesség legyen 10 Mbájt/sec (ekkor az átviteli idő 0.1 sec)

- programozott I/O átvitel esetén a processzor foglalt a teljes átviteli időben;
- megszakításos I/O átvitel esetén processzor csak a megszakítási kérelmek kiszolgálása alatt foglalt. Ez esetben a CPU terhelése (foglalása) 0.1 sec lesz, ha az interruptok egyenként kb. 2 μ sec hardver kezelési időt igényelnek és a megszakítás kiszolgálóprogramnak megszakításonként 98 μ sec futási ideje van. Ez 1000 interrupt esetén (1000 · 100 μ sec) összesen. 0.1 sec CPU időt jelent.

- DMA I/O átvitel esetén a processzor csak a megszakítások kiszolgálása alatt foglalt. Ha a DMA felprogramozása és az adatátvitel ellenőrzése 50 μ sec-et igényel és 1 interrupt 2 μ sec, 1 interrupt kezelőprogram kiszolgálás 98 μ sec (ekkor a megszakításkezelés 2 · 100 = 200 μ sec), akkor a processzor foglalása 300 μ sec = 0,0003 sec.

5.6.4.4. I/O adatátvitel I/O processzorral

A DMA vezérlő tehermentesíti a processzort az adatátvitel vezérlésétől, de ez nem vonatkozik az I/O eszközök vezérlésére (pl. a periféria állapotellenőrzése stb.). Az I/O processzorok (multiplex, szelektor és blokkmultiplex csatornák) viszont az I/O adatátvitel teljes vezérlését önállóan megoldják, tulajdonképpen társprocesszorként működnek. „Felprogramozásukat" a processzor végzi azáltal, hogy átadja az I/O eszközök „csatornaprogramját" az I/O processzor számára (CCW = Channel Command Word, CAW = Channel Address Word)

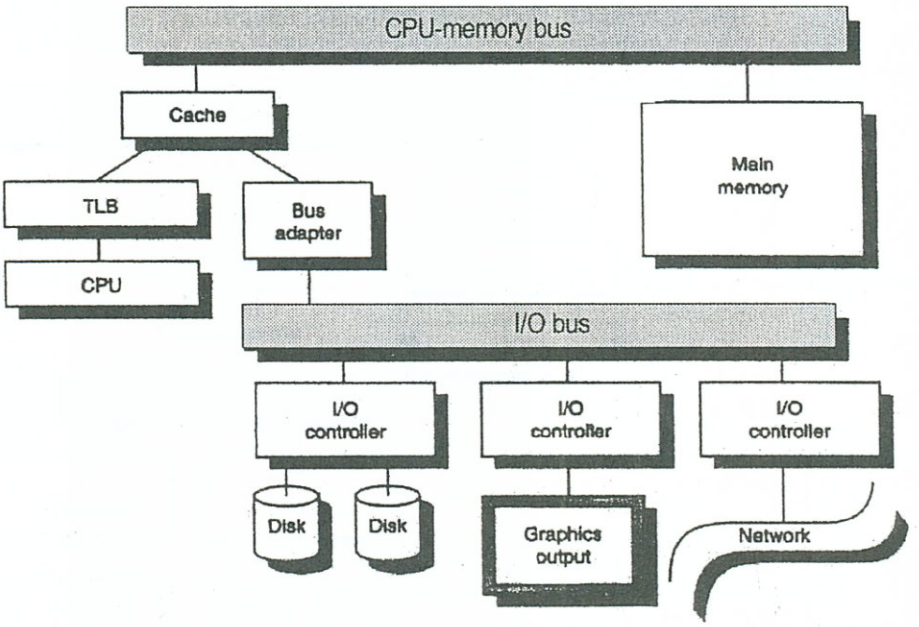
5.6.5. A cache, a virtuális tároló és az I/O műveletek összefüggései

Az alapproblémát az jelenti, hogy a virtuális tárkezelés és a cache alkalmazásával három másolatban létezhet ugyanaz az adat. Egyik változata a cache-ben, a másik a főtárban, a harmadik változata pedig a virtuális tárban. Ezeket az adatokat a CPU, az I/O eszközök és az operációs rendszer egymástól függetlenül módosíthatják.

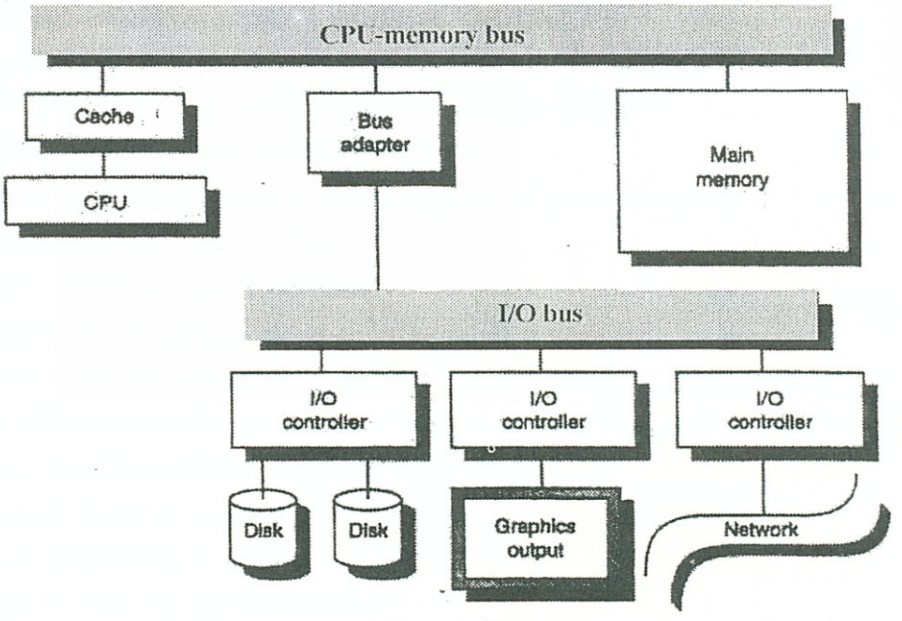
Emiatt az adatkonzisztencia megőrzése külön eljárásokat igényel. Erre két rendszertechnikai megoldás létezik:

- A CPU-memória busz a cache közbeiktatásával csatlakozik az I/O buszhoz. Ennek előnye, hogy a konzisztencia megfelelő áramkörökkel könnyen biztosítható. Ennek a megoldásnak a hátránya viszont az, hogy az I/O műveletek a cache közbeiktatásával kerülnek végrehajtásra, annak ellenére, hogy az I/O műveletek a cache-ben található adatokra csak ritkán hivatkoznak (lásd 109. sz. ábra).
- A CPU-memória busz közvetlenül csatlakozik az I/O buszhoz (lásd 110. sz. ábra). Ekkor a következő problémákra kell megoldást keresni:
  - Az I/O eszközök kiírásánál nem aktuális adatokat látnak a memóriában.

- Előfordulhat, hogy a CPU nem aktuális adatokat lát a cache-ben, mivel az I/O eszköz a tárolót már megváltoztathatta. Megoldást erre az jelenthet, ha a hardver vezérlés és az operációs rendszer felügyeli, hogy az I/O eszközök által írt vagy olvasott címtartomány adatai ne szerepelhessenek a cache-ben. (Az I/O pufferek az operációs rendszer rezidens területén kerülhetnek elhelyezésre.)



109. sz. ábra
Az I/O busz a cache közbeiktatásával csatlakozik a CPU-memória buszhoz

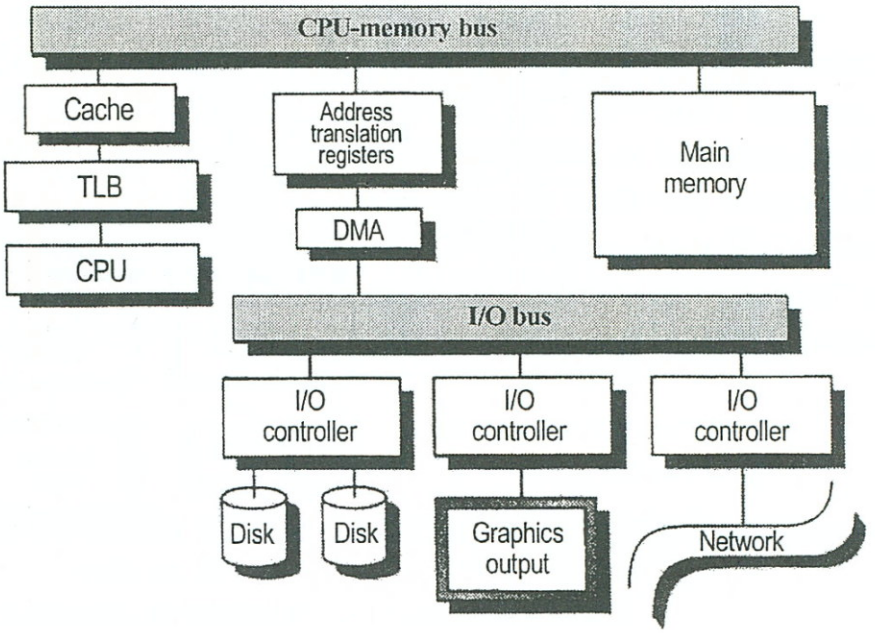


110. sz. ábra
Az I/O busz a cache kiiktatásával csatlakozik a CPU-memória buszhoz

A DMA és a virtuális tárkezelés együttes használata is problémákat okozhat.

- A DMA adatátvitel átlépheti a laphatárt (vagy a cím meg sem található a főtárban),
- Az operációs rendszer egy lapcserét végezhet a DMA átvitel közben

Erre megoldást az jelenthet, hogy egy hardver címfordító egységet (DMA lapregiszter) iktatunk be DMA és a CPU-memória busz közé (lásd 111. sz. ábra) illetve a DMA műveletek pufferterületét a virtuális tárkezelésen kívül (az operációs rendszer rezidens részében) jelöljünk ki.



111. sz. ábra
A DMA és a virtuális tár kapcsolata

ELLENŐRZŐ KÉRDÉSEK

?

180. Sorolja fel a számítógép fontosabb részegységeit!
181. Milyen típusú események kezeléséhez kell átmenetileg felfüggeszteni a program futását?
182. Mi a különbség a megszakítás és a kivétel között? Mit jelent a maszkolható és nem maszkolható megszakításkérelem?
183. Mi a lényege a vektoros megszakítási rendszernek?
184. Sorolja fel a megszakításkiszolgálás hardver és szoftver lépéseit!
185. Mi a feladata a megszakításvezérlőnek?
186. Mi a DMA lényege?
187. Ismertesse a DMA regisztereit és ezek feladatait!
188. Milyen regisztereket tartalmaznak az I/O eszköz vezérlők és ezeknek mi a feladata?
189. Hogyan irányítja a processzor az I/O eszközvezérlőket?
190. Mi a feladata a buszrendszernek és a buszvezérlőnek?
191. Az átvitt adatok jellege szerint milyen logikai részekből tevődik össze a sínrendszer?
192. Sorolja fel a sínrendszeren átvitt legfontosabb vezérlőjeleket!
193. Jellemezze a külső és belső sínrendszert, a helyi és rendszersínt!
194. Mi a PCI sín létrejöttének alapgondolata? Vázolja fel a PCI sín blokkvázlatát!
195. Mi a buszciklus, a busztranzakció és milyen lépései vannak?
196. Definiálja a master és slave eszközöket és legfontosabb feladataikat!
197. Mi a sín arbitráció, milyen típusú buszok vannak?
198. Hogyan lehet címezni a buszrendszerre rákapcsolt eszközöket?
199. Milyen lépésekből áll az egyszerűsített handshaking?
200. Mit jelent a fully interlocked handshaking?
201. Miért kell „timer"-t alkalmazni a kézfogásos algoritmusban?
202. Jellemezze a szinkron sínvezérlést!
203. Jellemezze az aszinkron sínvezérlést!
204. Mit jelent a blokkos, a megosztott adatátvitel a sínen?
205. Mit jelent a sztatikus és dinamikus buszhasználati jog megosztás?
206. A lefoglalt busz felszabadításának melyek a legfontosabb módszerei?
207. Mi a dinamikus buszhasználat megosztás feltétele?
208. Határozza meg a központosított és szétosztott buszarbitrációt!
209. Jellemezze a soros sínfoglalást!
210. Jellemezze a párhuzamos sínfoglalást!
211. Adja meg a tranzakció orientált I/O, a fájl I/O és az eseményorientált I/O fogalmát és teljesítményjellemzőit!

212. Mitől függ a számítógép I/O teljesítménye?
213. Milyen I/O műveletekkel összefüggő feladatai vannak az operációs rendszernek?
214. Az I/O eszközvezérlők milyen esetekben és formában adnak át információkat az operációs rendszernek?
215. Mutassa be az I/O eszközök címzésének lehetőségeit!
216. Definiálja az I/O adatátvitel típusait!
217. Jellemezze a programozott I/O átvitelt!
218. Jellemezze a megszakításos I/O átvitelt!
219. Jellemezze a DMA I/O átvitelt!
220. Milyen lépései vannak a DMA adatátvitelnek?
221. Hogyan történhet a buszhasználat megosztása a processzor és a DMA vezérlő között?
222. Mi az I/O processzor alkalmazásának lényege?

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

Arthur Dickschus: Egyszerűen PC ismeretek, Hardver. Panem 1998

Cserny László: Mikroszámítógépek. LSI 1996

Cserny László: RISC processzorok. LSI 1995

Haus-Peter Messmer: PC – Hardwerbuch. Addison-Wesley 1997

IBM Mikroelectronics: Power PC 750 Risc Microprocessor 233, 250, 266, 275 and 300 MHz 1998

IBM: Power PC 750™ SCM RISC Microprocessor 1998

Ila László – Sághi Balázs: Megjelenítők, háttértárolók, soros és párhuzamos interfész. Panem 1996

Intel Architekture Software Developer's Manual (I., II., III. kötet) 1996

Intel: Pentium III. Processzor Datasheet 1999

Intel: Pentium III. Xeon™ Processor at 500 and 550 MHz 1999

Intel: Pentium Pro Family Developer's Manual, Specifications (I, II., III. kötet) 1996

J. Henessy, P. Patterson: Computer Architekture – Quantitive Approach. Morgan Kaufmann 1996

Klaus Dembowski: PC táblázatok. Adatok és tények a PC hardverhez. Kossuth 1997

Kovács Magda: 32-bites mikroproceszorok. LSI 1994

Liebig/Flik: Rechnerorganisation. Springer-Verlag Berlin Heidelberg 2. Auflage 1993

Markó Imre: PC-k konfigurálása és installálása. LSI 1999

MPC 750 RISC Microprocessor User's Manual. Motorola Inc. 1997

Pentium II. Processor Developer's Manual 1997

Power PC Microprocessor Family: The Bus Interface for 32-Bit Microprocessors. Motorola Inc. 1997

Power PC Microprocessor Family: The Programming Enviroments. Motorola Inc. 1997

Rohon Coelho, Maher Hawash: DirectX, RDX, RSX and MMX technology. Addision Wesley Developers Press 1997

Sághi Balázs: Alaplapok, sínrendszerek, konfigurálás. Panem 1996

Sághi Balázs: PC építés, tesztelés, eszközkezelés. Panem 1997

Sima Dezső, Fountain Terence, Kacsuk Péter: Korszerű számítógép-architektúrák tervezésitér-megközítésben. SZAK 1998

S. Schulze, D. Monjau, K. Agsteiner: High-Level Entwurf mikroelektronischer Systeme unter Verwendung von Methoden des wissensbasierten Konfigurierens

Tabak: Advanced Microprocessors. McGraw-Hill 1995

Triebel, Walter: 80386, 80486 and Pentium Microprocessors; The Hardware, Software, and Interfacing. Prentice Hall 1997

Triebel, Walter: Effizienzbestimmung beim Parallelisieren von dinamischen Datenstrukturen. University Jena, 1997

Volker Lausch: Effiziente Programmierung moderner CPUS. Technische Universität Berlin 11/1996

Vörös Gábor: Bevezetés a neuronális számítástechnikába. LSI 1997

Win Chip C6 Processor Data Sheet. IDT 1997

Aktuális hardver hírek

http://www.coolcomputing.com/

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

Az Intel várható fejlesztései 2003-ig (Processzorok, chipsetek, alaplapok)

http://195.56.48.34/pcabc/hardver/intelfej.htm

Sebastian Loch: Dinamic Scheduling. Seminar "Moderne Rechnerarchitekturen" 1998, Universität Tübingen

Björn Köster: I/O-Grundlagen 1998

http://www.osgo.ks.he.schule.de/Lichtenberg/referate/MatAref98/iogrundlagen/io.html

Walter Waldner: Der intel Pentium II processor (P6-Familie) System-Architektur (1998)

http://www.htblo-klu.ac.at/lernen/hardware/pent2.htm

Digitaltechnik und Rechnerstrukturen

http://www.tik.ee.ethz.ch/tik/education/lectures/DRS/DRS.html

Future x86 CPU Brief 11.18.99

http://www.coolcomputing.com/features/futurecpu.html

Glossar zu Bus-systemen

http://dynamik.fb.10.tu-berlin.de/~borgmann/homeborg/bus-glos.html

HARDWARE HunPAGE

http://hardware.externet.hu/

Intel: Multiprocessor Specification Ver.1.4. (1997)

Mikroprocesszor fórum

http://www.chipanalyst.com/q/

Oliver Groht, Lars Lemcke: Speicherverwaltung der 80XXX/Pentium CPU FH-Hamburg (előadásvázlat)

http://141.22.17.5/~lehnart/Pentium/ppt/tsld001.htm

Paralleles Rechners

http://www.iib.bauing.tu-darmstadt.de/lehre/parallel/klars.html

Számítógépek, számítógéprendszerek (Jegyzet és előadásvázlatok)

http://www.iit.uni-miskolc/~vadasz/it02_szgpek/index.html

Universität-Gesamthochschule Padernborn Rechner architektur (skriptzuvorlesung)

http://hyperg.uni-padernborn.de/0x83ea6001_0x0001b332

Hadrver gyártók lapjai

http://www.usb.net

http://www.usb.org

http://www.intel.hu

http://www.sun.com

http://www.dell.com

http://www.hp.com
http://www.apple.com
http://www.tomshardware.com
http://www.ibm.com
http://www.mot.com
http://www.amd.com
http://www.asus.com
http://www.quantum.com
http://www.seagate.com
http://www.intel.com
http://www.developer.intel.com
http://www.computers.com
http://www.idt.com
http://www.cyrix.com
http://www.sgi.com

SZÓSZEDET

Nagykanizsa Vezető: *Németh Zoltán*
8800 Nagykanizsa, Ady Endre u. 74/a
Tel.: 06-93-312-383; 06-93-313-010/730; 06-30-9370-605;
Fax: 06-93-310-107; E-mail: nemethz.gdf@chello.hu

Nyíregyháza Vezető: *Méhész János*
4400 Nyíregyháza, Vasvári Pál u. 1.
Tel.: 06-42-407-027; 06-42-406-850; 06-20-3559-390;
Fax: 06-42-406-848; E-mail: szuvnyirseg@chello.hu
Honlap: www.freeweb.hu/gdf-nyhz

Pápa Vezető: *Gerics Erzsébet*
8500 Pápa, Budai Nagy Antal u. 1.
Tel.: 06-20-9583-058; Tel/Fax: 06-89-311-297;
E-mail: gerics@gege-comp.hu Honlap: www.gege-comp.hu

Pécs Vezető: *Kedves Vera*
7624 Pécs, Tiborc u. 38/c
Tel.: 06-72-213-412; Fax: 06-72-310-259;
E-mail: gdfpecs@axelero.hu

Pilisvörösvár **Szervezése Esztergomban**
Vezető: *Fűrész Tiborné*
2500 Esztergom, Bánomi út. 8.
Tel./Fax: 06-1-230-43-76; 06-20-3518-601;
E-mail: fureszt@axelero.hu Honlap: web.axelero.hu/fureszt

Salgótarján Vezető: *dr. Agócs József*
3100 Salgótarján, Kossuth u. 8.
Tel.: 06-32-416-833; Fax: 06-32-317-420;
E-mail: jozsef.agocs@ncsszi.hu

Sátoraljaújhely **Szervezése Mátészalkán**
Vezető: *Pethő Mária Terézia*
4700 Mátészalka, Seregély u. 17/A.
Tel.: 06-20-9743-125; 06-20-3663-669;
Tel./Fax: 06-44-417-889; E-mail: info@okito.hu

Siófok **Szervezése Székesfehérváron**
Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Sopron Vezető: *Dr. Molnár László*
9400 Sopron, Bajcsy-Zsilinszky u. 4.
Tel.:06-99-518-182; 06-30-4112-941;
Fax: 06-99-518-259; E-mail: molnarl@fmk.nyme.hu

Szeged Vezető: *Malincsák János*
6720 Szeged, Kígyó u. 4.
Tel: 06-20-9245-756; Tel./Fax: 06-62-423-258,
E-mail: malincsa@tvnetwork.hu, gdfszeged@tvnetwork.hu
Honlap: www.gdfszeged.hu

Szekszárd Vezető: *Doszkocs László*
7100 Szekszárd, Rákóczi u. 70.
Tel./Fax: 06-74-413-435; E-mail: gdf@kvantum.hu
Honlap: http://gdf.kvantum.hu/

Székesfehérvár Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Szolnok Vezető: *Nyáry László*
5000 Szolnok, Arany J. u. 4.
Tel.: 06-56-375-122; Fax: 06-56-511-262;
E-mail: titgdf@externet.hu

Szombathely Nyílt nap helye:
POTE Szhelyi Képzési Kp. Jókai Mór u. 14.
Vezető: *Csicseri Andrea*
9700 Szombathely, Kisfaludy u. 51.
Tel.: 06-94-501-894; 06-20-9910-357; Fax: 06-94-501-899;
E-mail: gdf_okt@flagnet.hu Honlap:www.flagnet.hu

Tatabánya Vezető: *Gimesi Lászlóné*
2800 Tatabánya, Vértanuk tere 13.
Tel./Fax: 06-34-510-460; 06-20-9625-047;
E-mail: gnkft@axelero.hu

Vác Vezető: *Dr. Molnár Lajos*
2601 Vác, Németh László u.4-6.
Tel.: 06-27-317-077; 06-27-317-886; Fax: 06-27-315-093;
E-mail: boronkay@vac.hu

Veszprém Vezető: *Nagyné László Mária*
8201 Veszprém, Egyetem u. 10.
Tel.: 06-88-422-022/4191; 06-20-5636-777;
Fax: 06-88-422-022/4452; E-mail: laszlo@almos.vein.hu

Zalaegerszeg Vezető: *Dr. Sárváryné Kiss Katalin*
8900 Zalaegerszeg, Rákóczi u. 4-8.
Tel.: 06-92-311-229; 06-20-9225-546; Fax: 06-92-510-590;
E-mail: zeggdf@szam.hu Honlap: www.szam.hu

Határon túli központok:

Románia

Erdélyi központok szervezése Kolozsváron (Cluj-Napoca)
Vezető: *Dr. Selinger Sándor*
Románia,3400 Cluj-Napoca str.: I. Budai Deleanu 64.
Levélcím: OFP 5. CP 737
Tel./Fax: 00-40/2-64-431-841; E-mail: gdf@gdf.ro
Honlap: www.gdf.ro

Csíkszereda (Miercurea Ciuc)
Kolozsvár (Cluj-Napoca)
Marosvásárhely (Târgu-Mureş)
Nagyvárad (Oradea)
Sepsiszentgyörgy (Sfântu-Gheorghe)
Szatmárnémeti (Satu-Mare)
Székelyudvarhely (Odorheiu-Secuiesc)

Szerbia-Montenegro

Szabadka (Subotica) Vezető: *Bóni László*
24000 Szabadka, Szegedi út 74. Szerbia-Montenegro
Tel.:00-381-24-546-067; 00-381-24-546-463;
Fax: 00-381-24-546-021; E-mail: imc@magnotron.co.yu

Szlovákia

Kassa (Košice) Vezető: *Dr. Kiss Imre*
04001 Košice, Moyzesova 58., Slovenská Republika,
Tel.: 00-421-55-727-4435, Fax: 00-421-55-727-4429;
E-mail: kiss@teledom.sk Honlap: www.teledom.sk

Diószeg Sládkovičovo Vezető: *Ladislav Spalek*
92521 Sládkovičovo, Fučikova ul. 269.
Slovenská Republika
Tel.: 00-421-31-788-1733; 00-421-31-788-1712;
Fax: 00-421-31-788-1710;
E-mail: ladislav.spalek@ivosr.sk, takacsatti@hotmail.com

ISBN 963-577-278-5
9 789635 772780

GÁBOR DÉNES FŐISKOLA
MŰKÖDÉSI HELYEI

Budapest **Informatikai Rendszerek Intézete:**
Vezető: *Dr. Kovács Magda*
1037 Budapest, Bécsi út 324.
Információ
Tel.:06-1-436-65-19; Fax:06-1-436-65-28
Honlap: www.gdf.hu

Informatikai Alkalmazások Intézete:
Vezető: *Dr. Zárda Sarolta*
1115 Budapest, Etele út 68.
Tanulmányi Osztály
Tel.: 06-1-203-02-83; 06-1-203-03-04/8900
E-mail: : tanoszt@gdf.hu Honlap: www.gdf.hu

Ajka Vezető: *Kovács Sándor*
Tel.: 06-88-453-948, 06-30-3902-576;

Baja Vezető: *Búcsú Lajos*
6500 Baja, Oltványi u. 14.
Tel./Fax: 06-79-426-427; E-mail: oktatas@fit.hu
Honlap: www.fit.hu

Balatonboglár Vezető: *Lakos István*
8630 Balatonboglár, Szabadság u. 41.
Tel.: 06-85-351-633; 06-30-2544-995;
Tel./Fax: 06-85-351-316 E-mail: lakist@mathiasz.sulinet.hu
Honlap: www.extra.hu/balatonboglar

Békéscsaba Vezető: *Kovács Zoltánné*
5600 Békéscsaba, Andrássy u. 73/1.
Tel.: 06-66-448-385; 06-30-3552-039;
Fax: 06-66-448-385; E-mail: gdf-bkk@nap-szam.hu
Honlap: www.gdf-bkk-03.nap-szam.hu

Cegléd Vezető: *Sági Ferenc*
2700 Cegléd, Kossuth F.u. 32.
Tel.: 06-20-9575-958;
Tel./Fax: 06-53-311-695; E-mail: sagif@infotars.hu
Honlap: www.extra.hu/gdf-cegled

Debrecen Vezető: *Kovács Jánosné*
4029 Debrecen, Maróthy Gy. u. 5-7.
Tel./Fax: 06-52-418-660; E-mail: gdf@cts.hu
Honlap: www.cts.hu

Dunaújváros **Szervezése Székesfehérváron**
Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Eger Vezető: *Csathó Csaba*
3300 Eger, Széchenyi u. 58.
Tel.: 06-30-9581-822; Tel./Fax: 06-36-411-811;
E-mail: sprinter@mail.agria.hu Honlap: http://sprinter.agria.hu

Esztergom Vezető: *Fűrész Tiborné*
2500 Esztergom, Bánomi út 8.
Tel./Fax: 06-1-230-43-76; 06-20-3518-601;
E-mail: fureszt@axelero.hu Honlap: web.axelero.hu/fureszt

Érd Vezető: *Budai Csaba*
2030 Érd, Széchenyi tér 1.
Tel./Fax: 06-20-9344-770; 06-1-283-79-79;
E-mail: budacsa@axelero.hu

Gyöngyös Vezető: *Gerják István*
3200 Gyöngyös Kőrösi Csoma S. u. 9.
Tel./Fax: 06-37-303-683; 06-30-9585-488
E-mail: gerjak@mail.datanet.hu, trivium@tvnetwork.hu

Győr Vezető: *Domonkos Gyuláné*
9021 Győr, Árpád u. 2.
Tel.: 06-96-550-200; Fax: 06-96-550-201
E-mail: etelka@gdfgyor.axelero.net Honlap: www.gdfgyor.axelero.net

Hódmezővásárhely Vezető: *Dr.Gál József*
6800 Hódmezővásárhely, Szántó K. J. u. 64.
Tel.: 06-62-535-536; 06-30-3135-873;
Fax: 06-62-535-530; E-mail: imsuli@delfin.hu
Honlap: www.gdf.vasarhely.hu

Isaszeg Vezető: *Keresztúri Jusztina*
2117 Isaszeg, Madách u. 1/a
Tel./Fax: 06-28-496-206; E-mail: gdfisa@mail.sziszi.hu

Kaposvár Vezető: *dr.Paál Jenő*
7400 Kaposvár, Guba S. u. 36-40.
Tel.: 06-82-314-155/147; 06-82-424-339;
Fax: 06-82-320-757; E-mail: posane@atk.kaposvar.pate.hu

Kecskemét Vezető: *Pósfayné Bakota Éva*
6000 Kecskemét, Hunyadi János tér 2.
Tel.: 06-76-411-494; 06-20-9762-100; 06-20-9935-735;
Fax: 06-76-411-041; E-mail: gdfkecsk@axelero.hu

Kisvárda Vezető: *Meskó Marianna*
4600 Kisvárda, Szent László út 27.
Tel.: 06-45-500-290; Fax: 06-45-500-291;
E-mail: info@felveteliiroda.hu Honlap:www.felveteliiroda.hu

Keszthely Vezető: *dr Vargáné Dugonics Rita*
8360 Keszthely, Deák F. u. 57.
Tel.: 06-83-312-330/262; 06-83-312-330/290; 06-30-2165-583
Fax: 06-83-314-334; E-mail: vdr@georgikon.hu

Mátészalka Vezető: *Pethő Mária Terézia*
4700 Mátészalka, Seregély u. 17/A.
Tel.: 06-44-417-889; 06-20-9743-125; 06-20-3663-669
Tel./Fax: 06-44-417-889;
E-mail: fic@okito.hu, info@okito.hu Honlap: www.okito.hu

Miskolc Vezető: *Dr Czap László*
3535 Miskolc, Árpád u. 2.
Tel.:06-20-3132-869; Tel./Fax: 06-46-414-429;
E-mail: czap@mazsola.iit.uni-miskolc.hu
Honlap: http://gdfm.homeip.net