Cserny László

MIKROSZÁMÍTÓGÉPEK

CSERNY LÁSZLÓ

MIKROSZÁMÍTÓGÉPEK

2. átdolgozott kiadás

Nyitott rendszerű képzés
– távoktatás – oktatási segédlete
Felsőoktatási tankönyv

LSI Oktatóközpont
A Mikroelektronika Alkalmazásának
Kultúrájáért Alapítvány

Készült a Művelődési és Közoktatási Minisztérium támogatásával.

I S B N 963 577 188 6

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

LIGATURA Kft - TEXT Kft

TARTALOMJEGYZÉK

ÁBRÁK JEGYZÉKE

ELŐSZÓ

A tisztelt Olvasó által most kézbe vett **Mikroszámítógépek** című könyv, reményeim szerint, jól fogja szolgálni a célját, azaz 'mindennapi' munkaeszközünk, a számítógép jobb megismerését, működésének jobb megértését.

A mikroszámítógépek(személyi számítógépek) széles körű elterjedésével, a számítástechnikai kultúra napjaink részévé vált. A számítógépet ma már nemcsak a műszaki, a tudományos-kutató munkát végző szakemberek használják, de az élet minden területére bevonult és sikerrel alkalmazzák a legkülönbözőbb képzettségű és végzettségű szakemberek is.

A gépek használói már ismerik a legfontosabb kezelési teendőket, a mindennapi munkájukhoz szükséges alkalmazói programok hasznát és használatát. Nem véletlen ezért, hogy a felhasználót közvetlenebbül érintő, a gépkezeléssel, a programismertetésekkel foglalkozó könyveknek van ma elsősorban sikere. Az utóbbi évek számítástechnikai kiadvány-dömpingjében azonban, kevés olyan mű jelent meg, amely a korszerű számítógépek felépítésével, működési alapelveivel foglalkozik. Nem jelent meg olyan könyv, amely az újabb processzorok nyújtotta lehetőségekkel és struktúrákkal(pl.: RISC struktúrával) foglalkozna.

Könyvemmel ezt a hiányt is szeretném némiképp pótolni, azon túlmenően, hogy a könyv elsődleges célja oktatási anyagul szolgálni a Gábor Dénes Műszaki Informatikai Főiskola hallgatói számára. Könyvemben a mikroszámítógépekkel, a mikroprocesszoros környezettel, nem egyetlen processzor típusra szorítkozva foglalkozom; igyekezve az általános és nem az egyedi megoldásokat megragadni és bemutatni.

Az oktatás segítése körülhatárolta és bizonyos mértékig megkötötte a könyv tartalmát és terjedelmét, ezért nem foglalkozhat azokkal a területekkel, amelyeket a számítógép-architektúrák nagyobb teljesítményű képviselői foglalnak el. Szándékaink szerint, a számítógép architektúrák témakörét egy másik kiadványban foglalnánk össze.

A **Mikroszámítógépek** könyv áttekintést kíván adni, a mikroszámítógépek mai helyzetének megfelelően, a gépek felépítésével, működésével kapcsolatos ismeretekről. A könyv olvasásánál számítunk valamilyen számítástechnikai tájékozottságra, alapfogalmak, működési elvek ismeretére, de ettől függetlenül, úgy vélem, bárki számára ajánlható a könyv, aki mélyebb ismereteket óhajt szerezni a mikroszámítógépek struktúrájáról, működéséről.

A könyv nem tudományos munka gyanánt készült, így nem tartalmaz elvi kifejtéseket, sem indoklásokat. A cél a közérthető, középiskolai tudásszint

mellett is elolvasható, megtanulható anyag összeállítása volt, amely ugyanakkor elegendő ismeretet ad arról - különösen egy informatikával foglalkozni kívánó leendő szakember számára -, hogy miből épül fel, hogyan működik egy mai mikroszámítógép. A könyv tartalmának tananyagkénti felhasználása, az ebből fakadó általánosító tárgyalásmód miatt, a konkrét eszközök vonatkozásában a közölt adatok nem tekinthetők hivatkozási alapnak, ahhoz a gyártó cégek dokumentációi adják meg a szükséges és megbízható információkat.

A könyv **1.**, bevezető **fejezete** azokat az alapfogalmakat, alapelveket tárgyalja, amelyek szükségesek a későbbi részek olvasásához, megértéséhez.

A számítógépek, azon belül a mikroszámítógépek általános jellemzőiről, felépítéséről szól a **2.fejezet**. A számítógépek struktúrájának tárgyalásakor a hagyományos, Neumann-elvű számítógépek szolgálnak kiindulási alapul, mivel a mikroszámítógépek és az univerzális célú nagyszámítógépek is erre a struktúrára épülnek ma még. Így a tárgy keretén belül, nem foglalkozunk az egyéb, nagyobb teljesítőképességű architektúrákkal.

A **3.fejezet** a központi egységek belső felépítésével, az adatok és az utasítások tárolási formáival, a művelet- és utasításvégrehajtás folyamatával foglalkozik, azokkal az ismeretekkel, amelyek a számítógépek működésével kapcsolatban, általánosan érvényesnek tekinthetők.

Külön érintjük az összetett utasításkészletű(CISC) és a csökkentett utasításkészletű(RISC) processzorok közötti különbségeket, valamint a teljesítménynövelés egyes strukturális eszközeit(pl.: az adatcsatornás[pipelining] feldolgozást).

A könyv **4.fejezete** a számítógépek egyik leglényegesebb erőforrásával, a tárolók használatával foglalkozik részletesen. A hatékonyságnövelő strukturális megoldásokról(tárhierarchia elemeiről és azok kapcsolatairól) és általánosító eljárásokról(virtuális címzési módról) olyan mélységben esik szó, hogy érthetővé váljék azok előnye és használatának lényege. Az anyag tárgyalásában a korszerű mikroprocesszorokat vesszük alapul és az azok által is alkalmazott megoldásokat vizsgáljuk általánosított értelemben.

A mikroszámítógép belső adatforgalmának lebonyolításával foglalkozó eszközöket(sínrendszer, megszakítások, I/O kapcsolatok) ismerteti az **5.fejezet**.

A **6.fejezet** a mikroszámítógépek leggyakoribb, illetve legjellemzőbb processzoraival foglalkozik részletesebben. Az itt közöltek a processzorok működését és jellemzőit kívánják bemutatni és ahogy ezt már említettük, nem helyettesíthetik a gyári adatlapokat, dokumentációkat, habár az adatok megadásához törekedtünk a pontos információk beszerzésére.

A fejezet *egyrészt* két, ismert CISC processzorcsaláddal, az INTEL és a MOTOROLA cég gyártmányaival, *másrészt* a RISC processzorok két, jellegzetes képviselőjével, a Sun Microsystems SPARC processzorával és a MIPS cég Rxx00-as processzoraival foglalkozik.

A könyv **7.fejezete** a háttértárakkal: a hajlékony- és merevlemezekkel, valamint az optikai tárolással foglalkozik áttekintő módon. Az alkalmazói oldali megközelítésnél, az IBM-PC/AT gépeket, mint a piacon legnagyobb számban jelenlévő mikroszámítógépeket tekintjük alapnak, ezért az eszközök BIOS, DOS szintű elérését is megemlítjük.

A **8.fejezet** a különböző monitorokkal, pontosabban azok vezérlési módjaival foglalkozik, szintén az IBM-PC/XT, AT gépeknél használt eszközök különböző típusainak bemutatásával.

Végül a **9.fejezet** a mikroszámítógépek egyéb beviteli/kiviteli eszközeivel (billentyűzet, egér, nyomtató) foglalkozik, azok használatához szükséges alapvető ismeretek tárgyalásával.

A könyv érdemi részét egy rövid **irodalomjegyzék** zárja, amely tájékoztatást kíván adni arról, hogy melyek azok a többnyire általános jellegű művek, amelyekből további ismeretek gyűjthetők a témakörhöz.

Bízom abban, hogy eltekintve az óhatatlanul meglévő hibáktól, a könyv hasznos segítője lesz nemcsak a Gábor Dénes Műszaki Informatikai Főiskola hallgatóinak a **Mikroszámítógépek** tárgy tanulásához, hanem bárkinek, aki szeretne bővebbet megtudni a számítógépek, különösen a mikroszámítógépek felépítéséről, működéséről.

Köszönettel tartozom mindenkinek, aki segítőkészségével, türelmével hozzájárult a könyv megírásához, továbbá a Gábor Dénes Műszaki Informatikai Főiskola azon hallgatóinak, akik a részükre tartott előadások révén hozzásegítettek az anyag tartalmának és terjedelmének kialakításához és "elszenvedték" a tankönyv hiányát.

Budapest, 1994.november 3.

A Szerző

Előszó a 2., javított kiadáshoz

Az elmúlt évek oktatási tapasztalatai meggyőztek arról, hogy tökéletesen megírt könyv nincs, de javítható igen. Ezért, szükségesnek tartottam a könyv mielőbbi többirányú javítását:

- *egyrészt*, néhány feleslegesnek ítélt rész lerövidítésével, vagy kihagyásával, a leírtak érthetőbbé tételével;
- *másrészt*, néhány rövid, új rész beiktatásával(a nagyobb teljesítményű architektúrák ismertetése, a fejezetvégi ellenőrző kérdések), amelyek, úgy hiszem, szükségesek a témakör áttekintéséhez és elsajátításához;
- *harmadrészt*, a rajzok szükséges javításával, finomításával, az óhatatlanul minden könyvben megmaradó helyesírási hibák lehetőséghez mért kigyomlálásával.

Remélem, hogy ezzel sikerült a kitűzött célt - az áttekinthető, tanulható és viszonylag korszerű tartalmú könyv megírását - jobban megközelíteni, mint az előző kiadással.

Természetes, hogy a gyorsan fejlődő terület legújabb eredményeit nem lehet folyamatosan nyomon követni, ezért egy-két év múlva szükségessé válik a könyv teljes átdolgozása is.

Budapest, 1996.július 15.

Szerző

1. FEJEZET || BEVEZETÉS

1.1.INFORMATIKAI BEVEZETŐ, FOGALMAK ÉRTELMEZÉSE

A számítógépek oly módon élnek az emberek tudatában, mint információfeldolgozó berendezések, ugyanakkor az emberek nem csak magának az információ fogalmának, hanem igen sok más, a számítástechnikában használt további fogalomnak a pontos tartalmával sincsenek tisztában. Ezért szükségesnek érezzük a leggyakrabban előforduló fogalmak előzetes, bevezető értelmezését.

a.)Információ, informatika

Az embert, mint élőlényt, környezetéből számtalan és igen sokféle hatás éri, amelyek befolyásolják tevékenységét, viselkedését. Az ember az őt ért hatások (tudatos, vagy nem tudatos) értékelése után dönt afelől, hogy mi legyen a reagálása, válasza ezekre a hatásokra. Azt mondjuk ilyenkor, hogy **döntést hoz**. Azok a 'valamik', amelyek kiértékelésére a döntést megelőzőleg sor kerül, azok alkotják az **információt**. Az információt mindig valamilyen információhordozó(fény, hang, szag, stb.) közvetíti az ember, a döntéshozó számára. Egy ilyen **közlemény** csak akkor ér valamit, csak akkor hasznos az ember számára, ha arra az adott helyzetben szüksége van, valamilyen új ismeretet szolgáltat, azaz egy adott helyzetben tulajdonképpen valamilyen mérvű bizonytalanságot, ismerethiányt csökkent. Ugyanakkor a közleményt, a beérkező jelsorozatot az embernek értelmezni is tudnia kell, azaz értenie kell a közlemény nyelvét.

Tehát azt lehet mondani, hogy **információ**nak tekintünk mindent, ami egy adott helyzetben bizonytalanságunkat csökkenti.

Hangsúlyozni kell, hogy az információ a közlemény, az üzenet tartalmi oldalát jelenti és nem pedig annak megjelenési formáját.

Így például, az iskolában nem a csöngetés hangja, mint hangfrekvenciás jelsorozat alkotja az információt, hanem annak tartalmi oldala, jelentése, amely szerint egy ilyen jelsorozat az óra végét, vagy kezdetét jelzi. De ugyanez a jelsorozat, egy másik környezetben, pl. a tűzoltóságnál, riasztást jelent.

Tehát ugyanaz a jelsorozat egy másik környezetben, egy másik 'döntési helyzetben' egészen más információt jeleníthet meg.

Hasonlóképpen egy számjegy sorozat, pl. a 43-as szám, egyik helyzetben jelentheti valakinek az életkorát, de egy másik helyzetben valakinek a lakcímében a házszám értéke lehet.

Az ember számára hasznos jelsorozatok, azaz amelyeknek az adott egyén számára információtartalma van, valahonnét kiindulnak, valamilyen közegen keresztül eljutnak az emberhez, aki azt értelmezi, felhasználja. Ezenközben az információ többször is átalakításra kerülhet, egyrészt formailag, másrészt tartalmilag is. Ez utóbbi azt jelenti, hogy új információ keletkezett a régiből.

Az **informatika** az a tudományág, amely az információk keletkezésével, továbbításával, feldolgozásával, hasznosításával foglalkozik a legszélesebb értelemben. Az informatika tárgyköre tehát nem szűkül le kizárólag csak a számítógépes feldolgozásokra, hanem ennél szélesebb értelmezésű.

b.)Adat, adatfeldolgozás

Az információk továbbításakor, feldolgozásakor a végrehajtott átalakítások, műveletek csak a közlemény formai oldalával kapcsolatosak. A végrehajtónak, pl. a számítógépnek, teljesen érdektelen a jelsorozat által képviselt információtartalom, azt csak az elvégzendő művelet(pl. összeadás) szabálya érdekli, azaz csak az, hogy a két összeadandó szám számjegyeit hogyan kell összeadni ahhoz, hogy az eredményt megkapja. Tehát a gép számára teljesen mindegy, hogy a két összeadandó érték két árucikk súlyát, vagy pedig valakinek két különböző helyről származó jövedelmének értékét jelenti-e. A műveletet mindkét esetben ugyanúgy hajtja végre. A gép számára(de az ember számára is) a jelentésétől megfosztott jelsorozat létezik csak ilyenkor.

Az információnak ezt az alakját **adat**nak nevezzük. Tehát az adat jelentésétől megfosztott jelsorozat.

Az információ és az adat fogalma a számítástechnikai gyakorlatban igen gyakran keveredik, nagyon sokszor nem teszünk határozott különbséget a kettő között. De mindenféleképpen tudni kell azt, hogy a két dolog nem ugyanaz. **Az információ a tartalmi, az adat a formai oldalát jelenti ugyanannak a közleménynek, jelsorozatnak.**

Az adatok megjelenési formája igen sokféle lehet(kép, hang, írott szöveg, stb.), azonban az emberi munkavégzés során a legfontosabb, leggyakoribb megjelenési mód az írott, nyomtatott jelsorozat. Ezek legkisebb eleme, alapeleme a **karakter**. A köznapi gyakorlatban használatos karaktereket három csoportba szokás osztani. Ezek:

- a betűk(A, B, C,..., a, b, c, ... , x, y, z),
- a számjegyek(0,1,2,3,...9), és
- az ú.n. speciális jelek(?,!,@,#,....+,-,*,/,=,....).

A csak számjegyekből álló adatokat **numerikus**, a betűkből állókat **alfabetikus**, a vegyes karaktersorozatból állókat **alfanumerikus** adatoknak nevezzük.

Az adatokon végzett átalakításokat, műveleteket, illetve ezek sorozatát **adatfeldolgozás**nak nevezzük. A gépi adatfeldolgozás többnyire a következő lépéseket foglalja magában:

- *adatelőkészítés*, melynek során az adatokat feldolgozásra, pontosabban adatbevitelre alkalmas formára hozzuk,
- *adatbevitel*, melynek során az előkészített adatokat betöltjük a számítógépbe és átalakítjuk arra a formára, amely a gép belső használata szempontjából a legmegfelelőbb,
- az *adatok feldolgozása* előre meghatározott lépéssorozat alapján,
- *adatkihozatal*, melynek segítségével a feldolgozás eredményét az ember számára értelmezhető, vagy a további feldolgozás számára megfelelő formára alakítjuk.

c.)Kódolás, kódrendszerek

Az adatfeldolgozások, adattovábbítások alkalmával az adatokat nagyon sokszor át kell alakítani annak érdekében, hogy a kívánt feladat a lehető leghatékonyabban, legkevesebb hibával legyen elvégezhető.

Így például, amikor egy hangversenyt közvetít a rádió, akkor a hangrezgéseket a mikrofon segítségével előbb elektromos jelekké alakítják, majd annak a hallgatóhoz való továbbítása érdekében, az elektromos jeleket az adóberendezés és az antenna elektromágneses hullámokká alakítja át. Ez jut el a vevőkészülékig, amely azt visszaalakítja, az ember számára értelmezhető hangrezgésekké.

A számítógépi feldolgozásokhoz, az adatbevitel alkalmával, a feldolgozandó számokat a berendezés átalakítja *egyrészt* tízes számrendszerből kettes számrendszerbe, *másrészt* karakteres formáról bináris(0-t és 1-est képviselő jelekből álló) formára.

Az előbbi példákkal bemutatott átalakításokat, amelyek a közlemény tartalmi oldalát nem érintik, **kódolás**nak nevezzük. A **kódolás** áttérést jelent valamely jelkészlet és szabályrendszer(együttesen: **kódrendszer**) használatáról egy másik jelkészlet és szabályrendszer használatára.

Azt a jelkészletet(pl. a 0, 1 számjegyeket, vagy a magyar ábécé betűit és írásjeleit), amelynek elemeiből egy közleményt fel lehet építeni, **kódábécé**nek nevezzük.

d.)Algoritmus, program

Egy feladat, egy probléma megoldásakor, az eredményt többnyire különböző műveletek, átalakítások, feltételek kiértékelése sorozatán keresztül tudjuk csak elérni. A feladatok elemzésével meghatározhatók azok a lépések(részfeladatok), amelyek végrehajtásával(megoldásával) a végeredményt elérhet-

jük. A feladatoknak íly módon felbontott és megkapott, a megoldáshoz vezető lépéssorozatát **algoritmusnak** nevezzük.

Egy útbaigazítás(pl.: "menjen az első kereszteződésig, ott forduljon jobbra, majd a második saroknál ismét jobbra ..."), amelynek alapján megtalálunk egy keresett helyet, algoritmust képez. Ugyanígy algoritmus a szakácskönyv előírása is, amely megadja, hogy mikor, mit, hogyan kell az edénybe tenni, meddig kell melegíteni, főzni ahhoz, hogy a kívánt étel elkészüljön.

Az algoritmusban megfogalmazott elemi lépések, amelyeket már nem akarunk további részfeladatokká felbontani, **utasítások** formájában írhatók elő a feladatot megoldó számára. Az utasítások egymásutánisága, sorozata alkotja a **programot**.

A számítógép számára nem írhatunk elő tetszőleges tartalmú utasításokat, hanem csak az alkalmazott programozási nyelv, végső soron a gép saját, ú.n. gépi nyelve által megengedett utasításkészlettel írhatjuk le a kívánt algoritmust. Ez lényegesen szűkebb, mint a beszélt nyelvek által megengedett lehetőségek köre és a programozási nyelv szabályrendszere, nyelvtana is sokkal szigorúbb, mint a természetes nyelveké, de egyelőre a számítógépek működési feltételei ennél többet nem tesznek lehetővé.

e.)Hardver, szoftver

Az információfeldolgozás leghatékonyabb, leguniverzálisabb eszköze a számítógép. A **számítógép** egy olyan (elektronikus) eszköz, amely tárolt programutasítások alapján dolgozik, azokat a tárolás sorrendjében végrehajtva és aritmetikai, logikai műveletek automatikus elvégzésére alkalmas.

A számítógépnek ez a megfogalmazása, mint kiindulási alap kezelendő, mivel a számítógépek új struktúrái ettől eltérő működésmódot, felépítést is lehetővé tesznek. A fenti megfogalmazás az ú.n. **Neumann-féle számítógép** alapelveit foglalja magában. A számítógép mind a feldolgozandó adatokat, mind pedig a feldolgozást vezérlő utasítássorozatot, a programot tárolja. A program utasításait a tárolás sorrendjében sorra végrehajtja, hacsak magában a programban nem írunk elő más végrehajtási sorrendet.

A könyvben tárgyalt számítógép ú.n. univerzális, digitális számítógép, ami *egyrészt* azt jelenti, hogy a számítógépet tetszőleges feladat megoldásához használhatjuk, *másrészt* az adatok tárolási formája számjegyes formájú, azaz minden információ kódolása számjegyes. Ez utóbbi, - elvben -, a számok esetében tetszőleges pontosságot megenged, de természetesen technikai és gazdasági okok miatt ez mindig csak meghatározott mértékű lehet.

A feladat megoldását szolgáló algoritmus gépi nyelven leírt utasításait a számítógép áramkörei, minden további átalakítás nélkül, közvetlenül végre tudják hajtani. A számítógép elektronikus áramköreit, mechanikus berendezéseit, kábeleit, csatlakozóit, perifériáit együttesen **hardver**nek(hardware-nek) nevezzük.

Azonban a számítógép hardvere önmagában nem sokat ér, mert nem működőképes a felhasználó szempontjából. Ahhoz, hogy a felhasználó feladatát meg lehessen oldani, a feladat algoritmusát át kell tenni a gép számára értelmezhető program formájába, majd ezt a számítógépen végre kell hajtatni, azaz a programot 'futtatni' kell. A számítógépet működőképessé tevő programok összességét(beleértve ebbe a programokhoz tartozó dokumentációkat is) **szoftver**nek(software-nek) nevezzük.

Vannak olyan számítógépi feladatok, elsősorban a gép általános vezérlésére szolgáló algoritmusok programjai, amelyek állandóak a felhasználó szemszögéből, de célszerű fejlesztési okokból lehetővé tenni azok esetleges cseréjét. Ezek a programok olyan kisebb tárakban vannak elhelyezve, amelyek csak olvasást engednek meg(ROM=Read Only Memory) és a tár cseréjével együtt, megoldható a tárolt program cseréje is. Ezt a megoldást, eszközt nevezik **förmver**(firmware)-nek.

Az előbbiek alapján érezhető, hogy a hardver és a szoftver határvonala nem húzható meg egyértelműen, ami annál is inkább igaz, mert minden programutasítás által megvalósított műveletet meg tudunk oldani megfelelő áramkörökkel és fordítva is igaz, mert minden áramkörrel megoldott műveletet is szimulálni tudunk programmal.

1.2.A SZÁMÍTÓGÉPEK ÁLTALÁNOS MŰKÖDÉSI ELVE

A számítógépeket a szellemi munka automatizálására tervezték és ott is elsősorban a számítási munkák megkönnyítésére. Az aritmetikai műveletek visszavezethetők néhány logikai művelettel megvalósított műveletsorra. Hasonlóképpen, más adatfeldolgozási problémák elvégzendő műveletei is megvalósíthatók elemi logikai műveletekkel.

A számítógépek algoritmus alapján, a programutasítások által vezérelve működnek. Alapvetően nagyon primitiv berendezések, mert csak és kizárólag csak azt hajtják végre, amire az ember által készített programok előírást adnak számukra. Ebben az értelemben olyan igen bonyolult automatákról kell beszélnünk, amelyek vezérlő algoritmusa cserélhető.

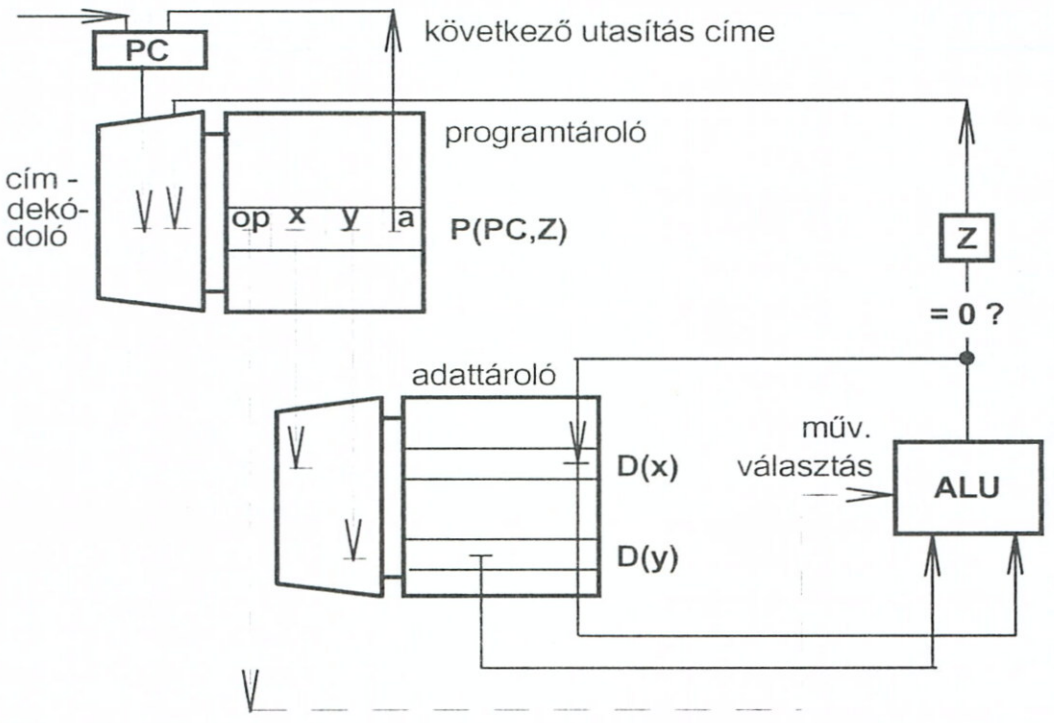
A számítógéptől, mint automatától megköveteljük, hogy olyan berendezés legyen, amely programozható módon aritmetikai és logikai műveletek végrehajtására képes és rendelkezik olyan lehetőséggel, amelynek révén a végrehajtandó feladatokhoz az induló adatokat kívülről átveheti és az eredményeket is át tudja adni a környezetének. **Tehát olyan automata, amely bemenettel és kimenettel is rendelkezik.**

A megfogalmazott célnak megfelelő programozható automata többféle is létezik, de a gyakorlati igényeknek az ú.n. **'logikai gép'** felel meg leginkább. Elméleti jelentőségű a programozható automaták körében az ú.n. Turing-gép, amely egyszerűsége miatt kiválóan használható automataelméleti vizsgálódásokhoz.

A számítógép céljára szolgáló 'logikai gép'-től az alábbi feltételek teljesülését várjuk el:

- közvetlen elérésű tároló(tehát, amelynek bármely tárolóhelye azonos idő alatt elérhető) alkalmazása,
- olyan műveletvégrehajtó egység, amely minimálisan a következő műveletek(utasítások) elvégzésére alkalmas:
  - = 'és'(and) művelet,
  - = 'kizáró-vagy'(exclusive or) művelet,
  - = 'eltolás'(shift) művelet, azaz amely az eredeti értéket a hatványalap értékével(többnyire 2-vel) egyszer, vagy többször szorozza, vagy osztja,
  - = 'átvitel'(move) művelet, amely egy adatot az egyik tárolóhelyről egy másik tárolóhelyre visz át.
- tárolóban elhelyezett program, amelynek automatikus végrehajtása vezérli a gép működését.

Az előzőekben körülírt tulajdonságokkal rendelkező gép logikai működési elvét mutatja be az 1-1.ábra.



1-1.ábra: *Logikai gép elvi vázlata*

A gép tárolt programja olyan utasításokból áll, amelyek az elvégzendő művelet kódja mellett, a művelethez szükséges két adat, valamint a következő utasítás tárolóbeli helyét(címét) is tartalmazzák. A művelet eredménye mindig az egyik adat helyére kerül. Ahhoz, hogy a **logikai gép** bonyolultabb, feltételektől is függő algoritmusok feldolgozására is alkalmas legyen, szük-

séges az, hogy az utasítások végrehajtási sorrendjét módosíthassuk. Ezért a következő végrehajtandó utasítás helyét módosíthatja az előző művelet eredménye is. Erre szolgál a Z(zero) jelzőbit(flag) használata oly módon például, hogy ha a művelet eredménye nem nulla(így Z=0), akkor a következő utasítás címét az utasításbeli cím határozza meg(azaz PC ⇐ a), ha pedig a művelet eredménye nulla(Z=1), akkor a soronkövetkező utasítás helye az aktuális utasítást követő tárolóhely legyen(azaz PC ⇐ PC+1). Az elvégzendő műveletet az utasítás műveleti kódja jelöli ki(az 'és', 'kizáró-vagy', 'eltolás', 'átvitel' parancsok valamelyikét).

Ahhoz, hogy a 'logikai gép'-ből igazi számítógép válhasson, további részegységek bevezetése, használata szükséges. Az elvégzendő teendők(funkciók) szempontjából, az **adatok bevitelét/kihozatalát biztosító egységeken** kívül, a következő funkcionális egységekre van szükség egy számítógépen belül:

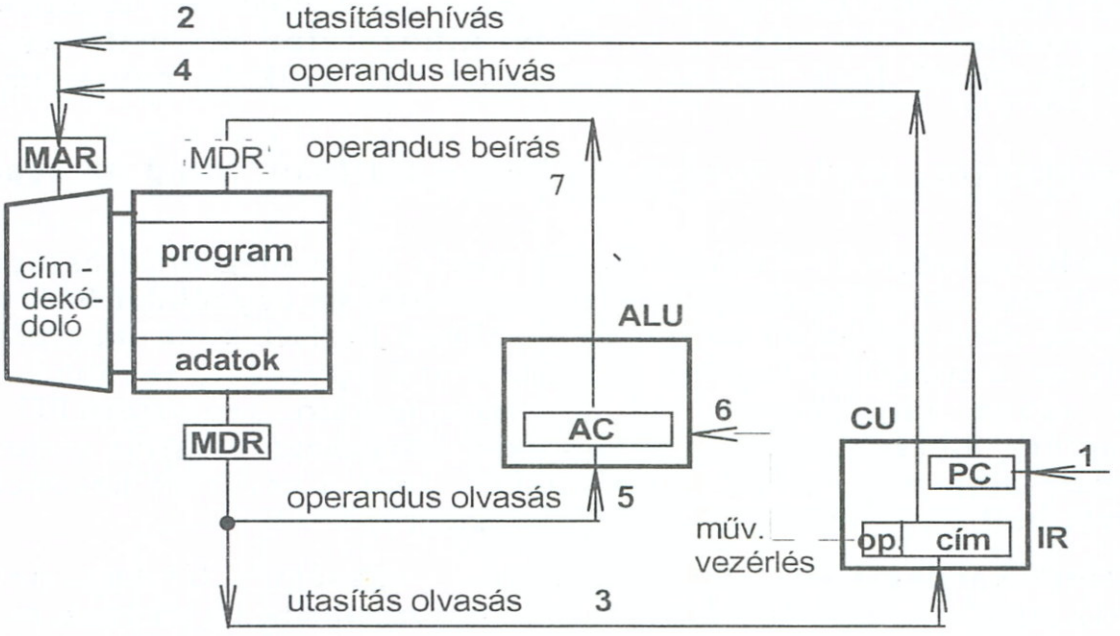
- **tárolóegység**, amely mind a program utasításait, mind az adatokat tárolja;
- **vezérlő egység**(CU=Control Unit), amely az egység utasításregiszterébe (IR=Instruction Register) a program utasításait egyenként befogadja, értelmezi azok műveleti kódját és az abban foglaltaknak megfelelően vezérli az aritmetikai-logikai műveletvégző egységet; továbbá tartalmaz egy olyan tárolóhelyet, az utasításszámláló regisztert(PC=Program Counter), amelynek tartalma mindig a következő utasítás tárbeli helyét (címét) adja meg. A PC kezdőértékét, azaz a program kezdőcímét, kívülről kell megadni, betölteni a regiszterbe;
- **aritmetikai-logikai műveletvégző egység**(ALU=Arithmetic-Logic Unit), amely a műveletvégzéshez az operandusok egyikének és az eredménynek az időleges tárolásához az akkumulátor regisztert (AC=Accumulator Register) használja fel; a művelethez szükséges másik operandus tárbeli címét a műveletet előíró utasítás jelöli ki.

Az említett egységek logikai kapcsolatát egy utasítás végrehajtása során az 1-2.ábra vázlata mutatja be.

Megállapítható, hogy az utasítások feldolgozása az alábbiakban részletezett fázisokra bontható fel. Természetesen egy-egy utasítás konkrét végrehajtása nem ennyire leegyszerűsített módon történik, de alapjaiban a működés logikája az elmondottaknak felel meg:

- a **program kezdőcímének** megadása(1.lépés) után,
- **utasításelőkészítés, -lehívás**: ebben a fázisban, az utasításszámláló regiszter(PC) tartalma alapján, a gép kikeresi a tárból a soronkövetkező utasítást(2.lépés) és átviszi a vezérlő egység utasításregiszterébe(IR) (3.lépés),
- **utasításszámláló regiszter tartalmának növelése**: ez a lépés a PC tartalmának növelésével a következő utasítás tárolóbeli helyének címét állítja elő; ez a PC tartalmának 1-gyel(pontosabban 1 utasítás hosszának megfelelő értékkel) való növelését jelenti,

- **műveleti kód értelmezése és az operandus címének meghatározása**: ezalatt a fázis alatt történik *egyrészt* a műveleti jelrész értelmezése, azaz annak meghatározása, hogy mit kell csinálnia a gépnek az utasítás hatására, *másrészt* a művelethez szükséges operandus(ok) címének a meghatározása, kidolgozása,
- **végrehajtás**: ebben a fázisban történik a kijelölt művelet végrehajtása (6.lépés) a kijelölt operandussal(ez magában foglalja az operandusnak a tárolóból történő kikeresését(4.lépés) és átvitelét(5.lépés) is),
- **visszaírás**: az eredmény előírt helyre történő írása(7.lépés).



1-2.ábra: *Neumann-elvű gép soros utasításfeldolgozásának vázlata*

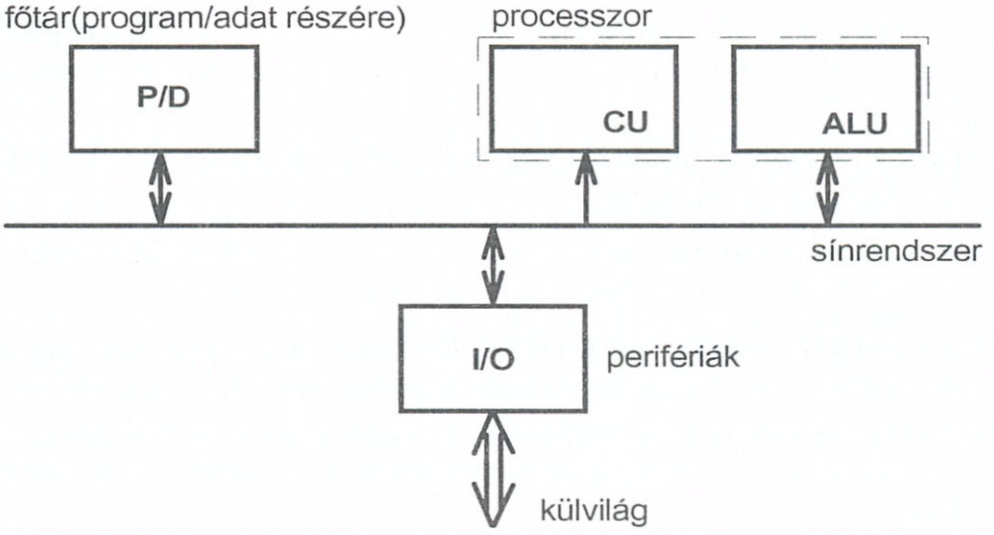
Az utasítások elmondottak szerinti feldolgozását követő számítógépnek, a még szükséges részekkel, tárolóhelyekkel kibővitett logikai vázlata látható az 1-2.ábrán. Ez az ábra az ú.n. **soros utasításfeldolgozás** elvét követi, amely szerint még a jelenlegi(Neumann-elvű) gépek többsége dolgozik. Ez azt jelenti, hogy az utasítások és adatok ugyanabban a tárban találhatók és emiatt először az utasítást kell onnét kikeresni, majd ezt követően a szükséges adato(ka)t.

A gép memóriájához tartozik két regiszter is; az egyik a címregiszter (MAR=Memory Address Register), amelyik a kiválasztandó tárolóhely címét fogadja be a kiválasztó dekódoló vezérléséhez; a másik regiszter az adatregiszter(MDR=Memory Data Register), amelyik a tároló bemenete/kimenete gyanánt szolgál, az adatok ezen keresztül kerülnek a tárolóba, illetve kiolvasáskor ezen keresztül jutnak el a gép más részeibe.

Az utasításregiszterbe kerülő utasítás műveleti jelrésze vagy az aritmetikai egység működését határozza meg, vagy a vezérlő(ugró) utasítások esetében, az utasításszámláló regisztert(PC) tölti fel az utasításban lévő memóriacím-

mel. Alaphelyzetben a PC tartalmát automatikusan 1-gyel(pontosabban 1 utasítás hosszának megfelelő értékkel) kell növelni minden egyes utasítás végrehajtása alkalmával.

Az aritmetikai egységek általában, a mindenkor megtalálható akkumulátor regiszter(AC) mellett, tartalmaznak még egy, vagy több további tárolóhelyet is, a második operandus ideiglenes befogadására.



1-3.ábra: *Számítógépek funkcionális egységeinek kapcsolatai*

Az 1-2.ábrán bemutatott elvi séma továbbra is egy nagyon leegyszerűsített vázlatát adja a számítógépeknek. A számítógépek bonyolultsága következtében további részegységek bemutatására is szükség lenne. A fő funkcionális egységeket és azok leegyszerűsített kapcsolatát mutatja be az 1-3.ábra blokkvázlata.

1.3.SZÁMÍTÓGÉPEK CSOPORTOSÍTÁSI LEHETŐSÉGEI

A számítógépek összehasonlításához, csoportosításához ki kell választani azokat a legjellemzőbb tulajdonságokat, amelyek alapján a többé-kevésbé hasonló gépek egy csoportba kerülhetnek. Igazából nem lehet egyértelmű módon egyik, vagy másik csoportba sorolni a számítógépeket.

A számítógépek legfontosabb tulajdonságai, amelyek lehetőséget kínálnak a csoportosításra, a gépek értékelésére:

- a gépek **műveleti sebessége**, azaz az az utasítás(művelet) szám, amelyet átlagosan egy időegység alatt dolgoz fel a gép; az így mért sebesség mértékegysége a MIPS(millions of instructions per second), vagy MOPS (millions of operations per second). Használják még a lebegőpontos aritmetikai műveletszámra vonatkozó MFLOPS(millions of floating point operations per second) mértékegységet is.

- a számítógép **órajel frekvenciája**; a gépek órajel sorozata szinkronizálja az egyes részek működését, biztosítja a párhuzamos folyamatok egymásmellettiségét és megszabja a számítógép működési sebességének felső korlátját. A ma(1996-ban) használatos átlagos mikroszámítógépek szokásos órajelfrekvenciája 50-100 MHz között mozog.
- a használt **áramköri egységek technológiája**; a gép logikai áramköreinek előállítási technológiája nagymértékben befolyásolja azok működési sebességét.
- a belső, illetve külső **sínrendszer**(bus system) **szélessége**, azaz annak értéke hogy, hány bináris jelet tud egyidőben, párhuzamosan továbbítani.
- az **utasítások, illetve műveletek** végrehajtásakor, azok **átlapolhatóságának** lehetősége(pipelining).
- a használt **szóhosszúság**, azaz azon bináris jelsorozathossz, amit az utasítások végrehajtásakor a gép egy egységként kezel.
- a **memória adatátviteli sebessége** [Mbyte/sec]-ban mérve; ennek nagyságát a memória ú.n. ciklusideje és a sínrendszer szélessége szabja meg.
- a **perifériális egységek adatátviteli sebessége** [Mbyte/sec]-ban mérve, amely a perifériák működési sebességétől és a periféria vezérlők kapacitásától függ.

A számítógépek **sebességük, teljesítőképességük alapján**, szokásosan három csoportba: nagy-, közép- és kisgépek körébe sorolhatók. Ezek a csoportok természetesen nem különíthetők el élesen egymástól, mert pl. a kisgépek nagy teljesítményű tagjai megfelelnek a középgépek kisteljesítményű típusainak.

A nagygépek, vagy **'mainframe' gépek**(super computer, mainframe computer) jellemzője a nagy műveleti sebesség, a nagy kapacitású tárolók és nagy teljesítményű perifériák használata. Ezeket a gépeket elsősorban nagyvállalati adatfeldolgozásokhoz, tudományos számításokhoz használják, ahol vagy nagyon sok adattal kell dolgozni, vagy igen számításigényes feladatokat kell megoldani. A gépeket ugyancsak nagy teljesítményű, sokoldalú működtető programrendszerrel(operációs rendszerrel) látják el, amelyek a gépek használatát nagy számú felhasználónak teszik lehetővé egyidőben. Mint nagy teljesítményű berendezések, általában a központi gép szerepét játsszák a többgépes rendszerekben.

A **középgépek**(minigépek - mini computer) csoportjába olyan gépek tartoznak, amelyek memóriája kisebb kapacitású, sebessége alacsonyabb és általában véve a teljesítőképességük kisebb, mint a nagygépeké. Alkalmazási területük szűkebb, azaz kisebb adatmennyiséggel dolgoznak, kisebb felhasználói kört szolgálnak ki. Többnyire folyamatvezérlési, termelésirányítási feladatok, mérésfeldolgozó rendszerek kiszolgáló gépeiként alkalmazzák azokat. Viszonylag nagy számú felhasználó kapcsolódhat rá saját eszközeivel(termináljával). Nagyobb teljesítményű, önállóan használt változatait bonyolult grafikai feladatok(pl. térinformatikai rendszerek, mérnöki tervezőrendszerek-CAD/CAM, stb.) kezeléséhez,

számításigényes feladatok megoldásához használják ú.n. munkaállomásként(work station).

A **kisgépek**(small computer), **mikrogépek**(microcomputer) kategóriájába tartozó gépek teljesítménye viszonylag alacsony, többnyire önállóan, személyi számítógépként, vagy hálózatba kötve használják őket. Az alacsony teljesítményt elsősorban a gépekhez használt perifériák(nyomtatók, tárolók) eredményezik, mert a csoport nagy teljesítményű tagjai mint munkaállomások, vagy mint hálózati kiszolgáló(server) gépek alkalmazhatók.

Másfajta csoportosítási lehetőséget ad a számítógéppel feldolgozott utasításfolyam és adatfolyam vizsgálata. Ennek a csoportosítási módnak, melyet M.J.Flynn javasolt 1966-ban, a bonyolultabb számítógép architektúrák besorolásánál van jelentősége. A **kezelt folyamatok száma** alapján a következő négy csoport alkotható meg:

SISD(Single Instruction Stream Single Data Stream), azaz egyetlen utasításfolyam és egyetlen adatfolyam feldolgozása. Az ilyen gépek egy vezérlő egységgel és egy aritmetikai egységgel rendelkeznek, egyidőben egyetlen utasítás végrehajtására alkalmasak. Ebbe a csoportba tartoznak a hagyományos, Neumann-elvű számítógépek.

SIMD(Single Instruction Stream Multiple Data Stream), azaz egyetlen utasításfolyam, többszörös adatfolyam feldolgozása. Ezek a gépek egy vezérlő egységgel és több aritmetikai egységgel rendelkeznek és egyidőben egy és ugyanazt az utasitást hajtja végre több adaton. Ebbe a körbe sorolhatók a vektor- és tömbprocesszoros gépek.

MISD(Multiple Instruction Stream Single Data Stream), azaz több utasításfolyam alapján egyetlen adatfolyam feldolgozása. Ilyen típusú gépek tulajdonképpen nincsenek, bizonyos esetekben ide sorolják az ú.n. pipeline feldolgozást alkalmazó számítógépeket.

MIMD(Multiple Instruction Stream Multiple Data Stream), azaz több utasításfolyam és több adatfolyam feldolgozása. Ebbe a csoportba tartozó gépek a különböző multiprocesszoros számítógépek.

A számítógépek csoportosítási lehetőségeit, tulajdonságait figyelembe véve, a nagygépek és a mikrogépek közötti különbségeket abban lehet meghatározni, hogy

- a nagyszámítógépek(szuperszámítógépek) belső és külső, perifériális teljesítménye lényegesen nagyobb(néhány GFLOPS nagyságrendű esetleg), mint a mikroszámítógépeké;
- a mikrogépek központi egysége, pontosabban processzora(vezérlő egység és aritmetikai egység), egyetlen integrált áramköri egységet alkot (egytokos mikroprocesszor), míg a nagyszámítógépek központi egységének magját nem egyetlen mikroprocesszor alkotja;

- a mikrogépek mellett alkalmazott perifériális eszközök(elsősorban nyomtatók, tárolók) teljesítménye és megbízhatósága kisebb, mint a nagygépek mellettieké;
- a mikrogépek összeszerelt állapotukban, általában egyetlen házban, dobozban vannak elhelyezve, többnyire asztali méretben, míg a nagygépek mérete még ma is ennél nagyobb, egy, vagy több szekrény nagyságú.
- mindegyik gépnél jellemző már azonban, az igen nagy integráltsági fokú (VLSI) áramkörök használata.

1.4.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Mit értünk információ alatt?
2. Mi az informatika?
3. Mit értünk adat alatt?
4. Mi a különbség az információ és az adat között?
5. Mi a kapcsolat az információ és az adat között?
6. Milyen részekre bontható fel az adatfeldolgozás folyamata?
7. Mit értünk kódolás alatt?
8. Mi a kódrendszer?
9. Mit nevezünk algoritmusnak?
10. Mi a program?
11. Mit értünk számítógép alatt?
12. Mit értünk hardver alatt?
13. Mit értünk szoftver alatt?
14. Mi a förmver(firmware)?
15. Milyen számítógép a CISC gép?
16. Milyen számítógép a RISC számítógép?
17. Vázolja fel a logikai gép részeinek kapcsolatait.
18. Milyen alapműveletek szükségesek és elégségesek a működőképes automatához?
19. Miért vezethetők vissza a számítógépi műveletek a logikai műveletekre?
20. Miért szükséges a feltétel figyelembe vétele a logikai gép működéséhez?
21. Melyek az utasításfeldolgozás lépései? Sorolja fel őket!
22. Mi az oka a soros utasításfeldolgozásnak?
23. Milyen részegységek szükségesek egy logikai automata működéséhez?
24. Mi a feladata a vezérlő egységnek?
25. Mi a feladata a műveletvégző egységnek?
26. Mi a feladata a tárolóknak?
27. Mi a feladata a beviteli/kihozatali egységnek?
28. Mi a feladata a sínrendszernek?
29. Vázolja fel a logikai automata legfontosabb egységeinek kapcsolatait tükröző blokkvázlatot!
30. Soroljon fel öt jellemzőt, amelyek alapján a számítógépek csoportosíthatók!
31. Mit értünk az órajelfrekvencia alatt?

32. Mit értünk a számítógépek műveleti sebessége alatt?
33. Mit értünk sínszélesség alatt?
34. Milyen kategóriákat használnak a teljesítmény szerinti csoportosításban?
35. Milyen csoportokat különböztet meg a folyamatok szerinti csoportosításkor?
36. Mit értünk a SISD kategória alatt?
37. Milyen architektúrájú számítógépek sorolhatók a SISD típusú gépek körébe?
38. Hová tartoznak a Neumann-elvű számítógépek?
39. Mit értünk a SIMD kategória alatt?
40. Milyen struktúrájú számítógépek tartoznak a SIMD típusú gépek körébe?
41. Mit értünk a MISD csoport alatt?
42. Mit értünk a MIMD rövidítés alatt?
43. Mely gépek tartoznak a MIMD kategóriájú számítógépek közé?

2. FEJEZET

SZÁMÍTÓGÉPEK JELLEMZŐ FELÉPÍTÉSE, STRUKTÚRÁJA

Az előző fejezetben már be lett mutatva a számítógép általános működési elve és azok a jellemzők, amelyekkel a mikroszámítógépek és a nagyszámítógépek között különbség tehető. A mikroszámítógépek egyik lényeges jellemzője az egytokos processzor alkalmazása. A könyv további részében tehát(kivéve a következő, 2.1.pontot), csak olyan számítógépekkel foglalkozunk, amelyek egyetlen mikroprocesszor köré vannak felépítve.

2.1.HAGYOMÁNYOS FELÉPÍTÉSŰ SZÁMÍTÓGÉP

A hagyományos felépítésű számítógép alapelveit, első mechanikus, kisérleti változatát már a 19.században megalkották. Charles Babbage által 1834-ben megalkotott **Analytical Engine** már mindazon részegységeket tartalmazta, amelyeket a mai számítógépek is magukban foglalnak. Mechanikus tárolású, lyukkártyás program- és adatbeviteli egységgel, nyomtatott és lyukkártyás kimeneti lehetőséggel rendelkező gép volt. Decimális tárolási móddal és aritmetikai műveletvégzővel dolgozott. Sajnos a gépet igazából nem sikerült Babbage-nek befejeznie, az akkori gyártási technika korlátai miatt.

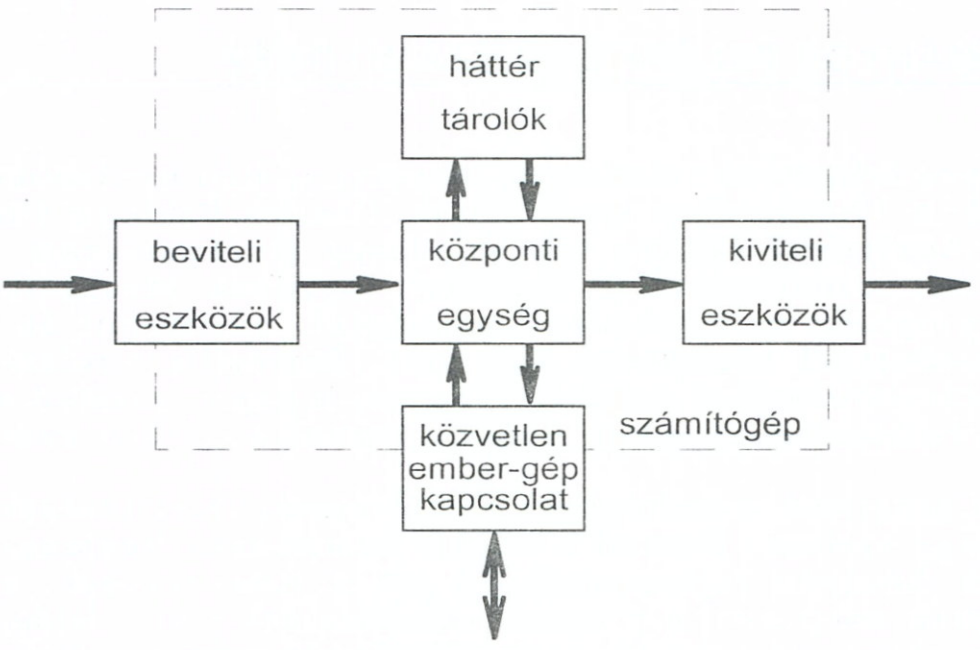
A mai, tárolt programú számítógépek alapelveinek, felépítésének kidolgozásában igen nagy szerepet játszott a magyar származású **Neumann János**. Az 1940-es évek közepén fejlesztették ki azt a számítógépet, amely tárolt program alapján dolgozott, a program utasításainak soros feldolgozási elve szerint. A mai, korszerű számítógépek többsége is ezen elv alapján működik, de utalni kell arra, hogy a nagyteljesítményű, multiprocesszoros feldolgozásoknál más utasításfeldolgozási módot is alkalmaznak. Pl.: az adatvezérelt számítógépeknél nem a tárolás sorrendjében történik az utasítások feldolgozása, míg a neurális számítógépek(neural computer) nem is rendelkeznek tárolt programmal.

2.1.1.A hagyományos számítógépek jellemzői

A Neumann-elvű, azaz hagyományos számítógépek legfontosabb jellemzőiként, összefoglalóan a következőket lehet felsorolni:

- a gép tartalmaz egy **közös tárolót**, amely egyaránt tárolja a végrehajtandó program utasításait, valamint az utasítások által feldolgozandó adatokat is; a program és az adatok kódolására egyaránt bináris(két jelből álló) kódrendszert alkalmaznak;
- a gép **vezérlő egysége** a tárolt program utasításait egyenként sorra véve oldja meg a kívánt feladatot; az automatikus programvégrehajtás egyszerűsítése végett a vezérlő egységben egy utasításszámláló regiszter tárolja a soronkövetkező utasítás tárolóbeli helyének címét;
- a program utasításai által megkívánt aritmetikai és logikai műveletek elvégzésére egy önálló egység, az **aritmetikai és logikai műveletvégző egység** szolgál; a bináris kódolás alapján minden művelet visszavezethető elemi logikai műveletekre, így tulajdonképpen az aritmetikai egység csak logikai műveleteket végez igazából;
- az adatok és a program **bevitelére/kihozatalára önálló egységek** szolgálnak.

A számítógép funkcionális felépítését, struktúráját mutatja be a 2-1.ábra.



2-1.ábra: *Számítógépek funkcionális felépítése*

A közös adat- és programtároló alkalmazásának jelentős következménye az, hogy a program utasításait a végrehajtás során át lehet írni, ugyanúgy, mintha az is adat lenne. Ezt a lehetőséget, amelyre szintén Neumann János hívta fel a figyelmet, azonban ma már nem célszerű kihasználni. Ugyanis *egyrészt* a rendelkezésre álló tárolóhely nagysága már többnyire elegendő és így nincs szükség az ilyen megoldásokra, *másrészt* a programok bonyolultsága olyan fokú lett, hogy az ellenőrizhetőség(tesztelhetőség) miatt nem lehet megengedni az ilyesfajta 'trükkös' megoldásokat, *harmadrészt* a

programok nagyüzemi gyártástechnológiája sem teszi lehetővé az ilyen időigényes és megbízhatalan működésű fejlesztéseket.

2.1.2.A számítógépek erőforrásai

Az automatizált gépi munka hatékonyságnövelésének első lépcsője a különböző feladatok elvézésére önálló egységek létrehozása a munkamegosztás elve alapján. A mikroszámítógépek esetében ez az elkülönítés már igen jól látható, mivel az egyes egységek vagy egy-egy integrált áramköri egységben (IC-ben), vagy egy-egy cserélhető, nyomtatott áramköri lapon(kártyán) találhatók. A számítógép funkcionális felépítését, amely az adatfeldolgozás folyamatát követi, a 2-1.ábra mutatja be

Az ábrán látható funkcionális egységek, amelyeket az alábbiakban részletesebben is bemutatunk, a következők:

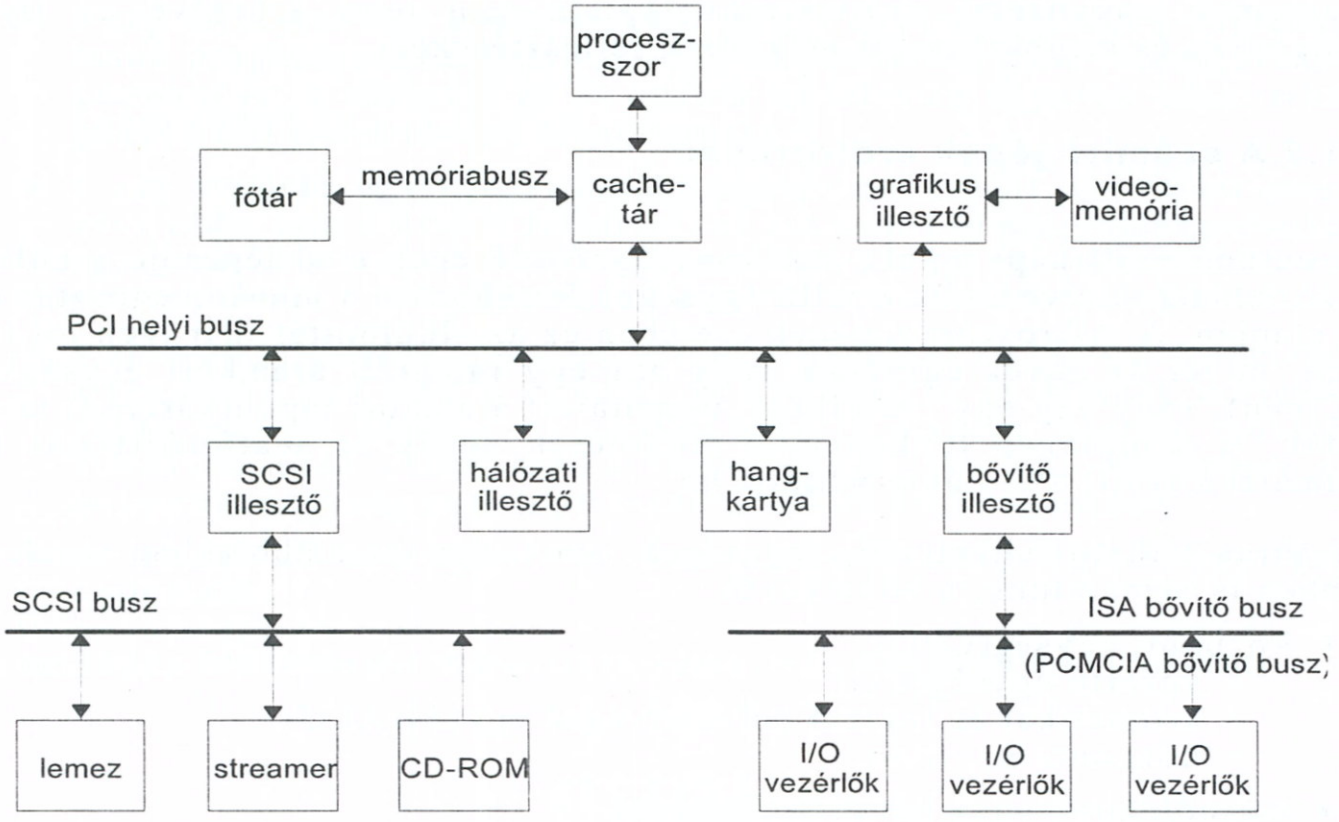
- központi egység(CPU)
  = vezérlő egység(CU),
  = aritmetikai és logikai műveletvégző egység(ALU),
  = központi tár, főtár,
- másodlagos-, vagy háttértárolók,
- perifériák(I/O)
  = beviteli egységek(input units),
  = kiviteli egységek(output units),
  = ember-gép-kapcsolat eszközei;

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

- sínrendszer(bus system)
  = címsín,
  = adatsín,
  = vezérlősín.

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

A Neumann-elvű számítógépek mai, korszerű változatának egy lehetséges rendszertechnikai felépítését mutatja be az alábbi, 2-2.ábra. Az egyes részegységek közötti kapcsolatokat a különböző szintű sínrendszerek hozzák létre. A sínrendszerek típusjelöléseinek(ISA, PCI, SCSI, PCMCIA) magyarázatára a későbbiekben, a sínrendszerek tárgyalásakor térünk vissza.



2-2.ábra: *Számítógépek rendszertechnikai felépítése*

a.)Központi egység(CPU), processzor

A számítógép központi egysége a gép 'lelke'. A vezérlő egység irányítja a tárolt program alapján a műveletek végrehajtását és így, elsősorban az aritmetikai egység munkáját is. A vezérlő egység feladata a processzor és az I/O eszközök közötti adatátvitel irányítása is. (Kivéve egyes gyors perifériák saját vezérlőit, amelyek önállóan végzik az adattovábbítást, a központi vezérlő egység indítása után.) A perifériák és a processzor közötti adatátvitelre a gép **sínrendszere** (buszrendszere) szolgál.

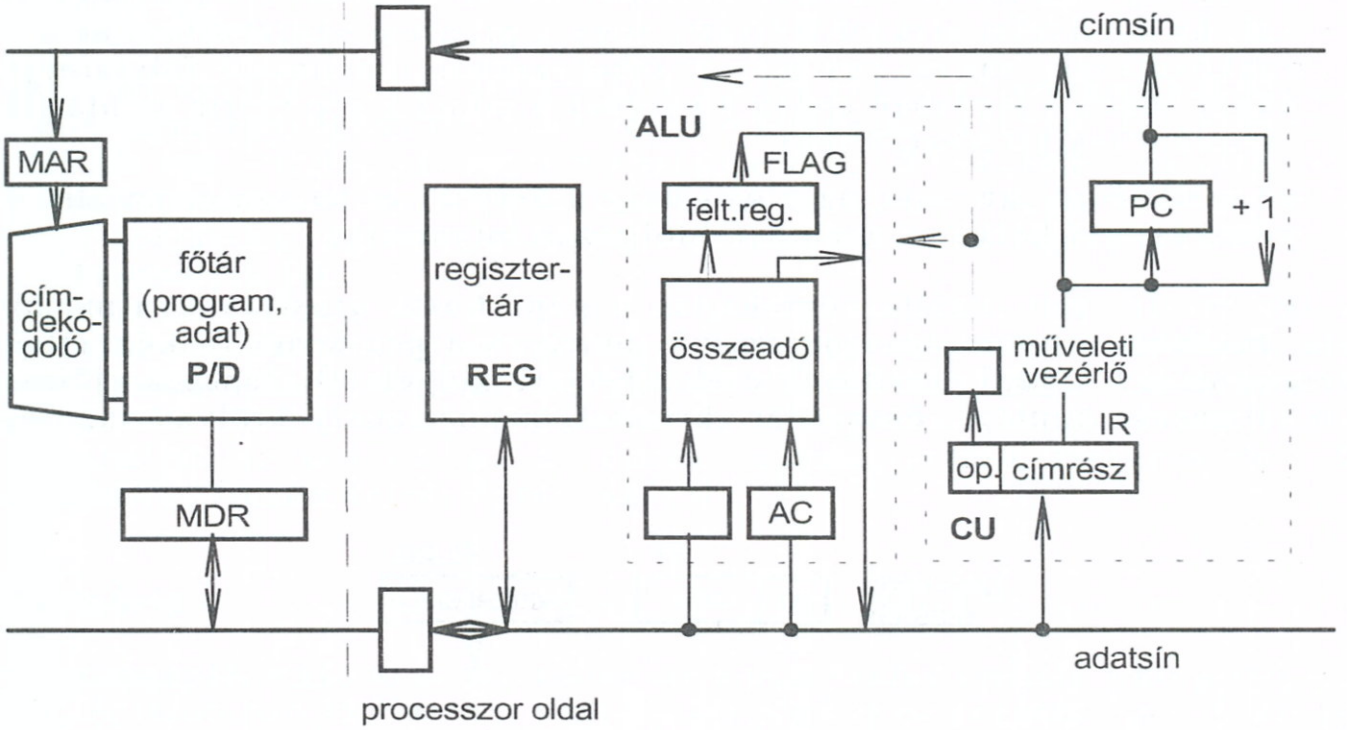
A processzor részletesebb rajzát az adat- és címkapcsolatokkal együtt a 2-3.ábra mutatja be.

A processzor a vezérlő egység és az aritmetikai egység működéséhez, kisebb-nagyobb számban használ speciális, vagy általános célú regisztereket egy-egy adat tárolására.

A **regiszterek** egy-egy adat átmeneti befogadására szolgáló 16-32-64 bináris helyiérték(bit=binary digit) hosszúságú, gyors működésű tárolóhelyek.

A vezérlő egység két legfontosabb regisztere: az **utasításszámláló regiszter** (PC=Program Counter, vagy IP=Instruction Pointer), amely a tárolt program soronkövetkező utasításának memóriabeli címét tárolja és az **utasításregisz-**

ter(IR=Instruction Register),amely a tárolóból előkeresett, végrehajtandó utasítást fogadja be a feldolgozás időtartamára.



2-3.ábra: *Központi egység felépítése*

A vezérlő egység, illetve a számítógép fontos regisztere még az aktuális processzorállapotot visszatükröző és vezérlési előírásokat is tartalmazó **vezérlő/állapotjelző regiszter**(Control/Status Register, Flag Register).

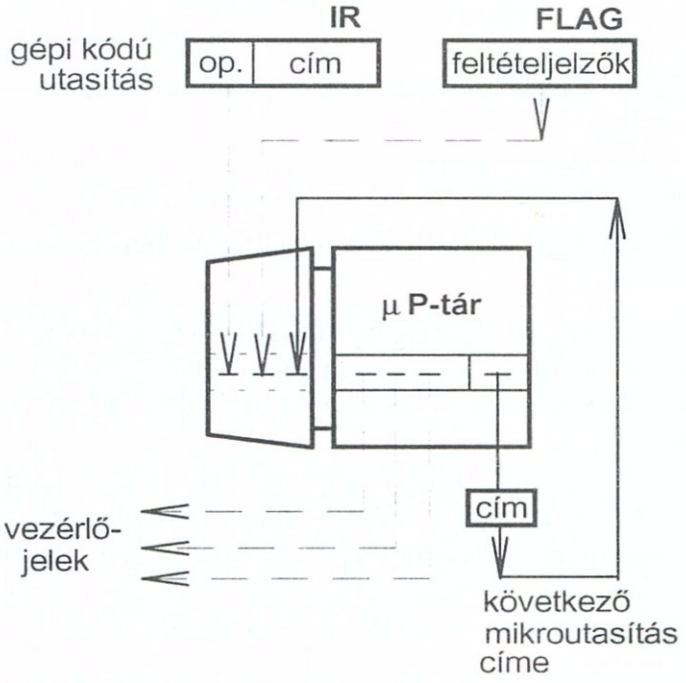
Korábban már említésre került, hogy a program utasításai, hacsak magában a programban nem adunk meg ettől eltérő értelmű sorrendet, a tárolás szigorú sorrendje szerint kerülnek végrehajtásra. Ezért a PC tartalmát minden utasítás végrehajtása alkalmával, automatikusan 1-gyel növeli a gép, hacsak az említettek miatt, a program maga nem ír elő más értéket.

Az utasítások feldolgozásának folyamata a következő lépésekre bontható:

- a PC tartalma - az ott lévő memóriacím - alapján, a feldolgozandó(végrehajtandó) utasítás kikeresése a tárban és átvitele az utasításregiszterbe(IR-be);
- a PC tartalmának növelése 1-gyel(pontosabban 1 utasításhossznak megfelelő tárolóhely számmal), hogy a PC a soronkövetkező utasítás helyét adja meg a tárolóban;
- a lehívott utasítás értelmezése, dekódolása, azaz annak meghatározása, hogy milyen feladatot kell a gépnek elvégeznie az utasítás hatására; ha az utasításban előírt művelet adat(operandus) felhasználását követeli meg, akkor meghatározza az operandus memóriabeli pontos helyét(címét), egyébként pedig a végrehajtási fázisra tér át;

- a szükséges adatot előkészíti a művelet elvégzéséhez, azaz a cím alapján kikeresi a memóriából és átviszi az utasítás által meghatározott helyre, amely legtöbbször az aritmetikai egység akkumulátor regisztere (AC);
- az utasításban előírt művelet végrehajtása az előkészített operandusokkal; ha a művelet arra ad előírást, hogy a program végrehajtása ne a soronkövetkező, hanem egy másutt elhelyezkedő utasítással folytatódjék, akkor ebben a lépésben ennek a helynek a címét a gép beírja az utasításszámláló regiszterbe(PC-be);
- az eredményt, ha volt, az előírt helyre helyezi és visszatér az első lépésre és újra kezdi az utasításfeldolgozás folyamatát.

Az így részletezett utasításvégrehajtás a **gépi kódú utasításokra**(machine code, machine instruction) vonatkozik, amelyek a gép számára közvetlenül értelmezhetők. A gépi utasítások csak egyszerű műveletek, lépések előírására alkalmasak; mint pl. összeadás, kivonás, memóriahely kiolvasása, írása, stb.



2-4.ábra: *Mikroprogramozott műveleti vezérlés elve*

A vezérlő egység az utasítás műveleti előírásának értelmezése(dekódolása) után, az abban előírtaknak megfelelően vezérli a gép egészét, a különböző helyeken lévő áramköri egységeket. Ez a vezérlés, a **műveleti vezérlés** történhet *közvetlenül*, ú.n. **'huzalozott logikával'**, amikor is a dekódolás hatására, a vezérlő egység egy bonyolult áramköri rendszer segítségével állítja be a gép egyes részeinek állapotát és történhet a vezérlés *közvetett*, **mikroprogramozott módon** is. A mikroprogramozott vezérlés, amelyet először M.V.Wilkes javasolt már 1951-ben, azt jelenti, hogy az utasítás műveleti kódja elindít egy kisméretű, elemi vezérlési lépéseket tartalmazó mikroprog-

ramot és ennek segítségével szabályozza a gép egyes részeinek állapotát. A mikroprogram a mikroprogramtárban található, amely csak-olvasható, ROM tároló.

A 2-4.ábra a mikroprogramozott műveleti vezérlés egyszerűsített logikai elvét vázolja fel.

A központi egység, a processzor másik fontos eleme az **aritmetikai-logikai műveletvégző egység(ALU)**, amely a műveletvégzés eszköze. Az aritmetikai egységben egy, esetleg két regiszter található: az akkumulátor regiszter(AC) és a második operandust befogadó regiszter; az újabb processzorok akkumulátor és operandus regiszter gyanánt is az általános célú regiszterek valamelyikét használják fel.

A processzorok fontos regisztere még a **veremtároló**(stack) használatát biztosító **'veremmutató' regiszter**(SP=Stack Pointer Register). A veremtároló egy speciális tároló, amely elsősorban az alprogramok kezelését segíti elő, de használják az aritmetikai műveletek végrehajtásának szervezéséhez is, operandusok, részeredmények tárolásával.

b.)Memória

A számítógépek legfontosabb erőforrása a processzor mellett a központi memória. A tárolóban található a végrehajtás alatt lévő program és a feldolgozásban felhasznált adatok is.

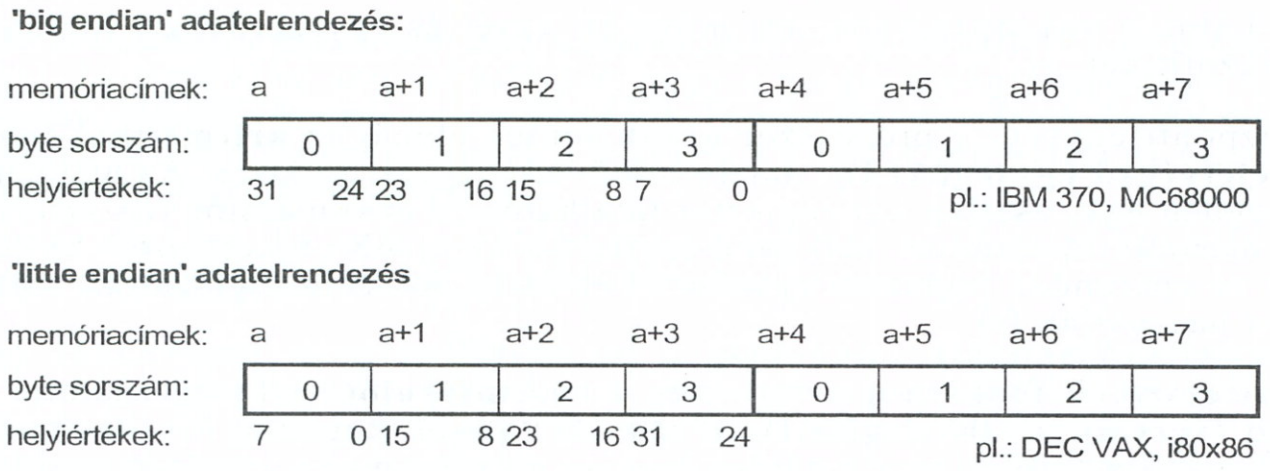
A memória legkisebb tárolási egysége az egy bináris jel(0, vagy 1) tárolására szolgáló elemi rész. A tárolók ezen legkisebb egységét **bit**nek nevezik. A processzor által fizikailag egy egységként kezelhető legkisebb memória terület(ritka kivételtől eltekintve) ennél azonban nagyobb. A fizikailag legkisebb egységként kezelt tárolóterületet **rekesz**nek(location, cell) nevezzük. Ennek mérete az egyes géptípusoknál más és más lehet, de a ma forgalomban lévő gépek többsége esetében, ez az érték egységesen **1 byte**(8 bit).

Minden fizikailag önállóan kezelhető tárolóhely(rekesz), **cím**mel(address) rendelkezik, amely alapján a tárolóhelyet a processzor ki tudja választani és abban adatot tud elhelyezni, vagy adatot tud onnan kiolvasni. A rekeszek címeit 0-val kezdődő szigorúan növekvő sorszámok alkotják.

A címzés szempontjából, ha nem byte-onként, hanem például 4 byte-os szavanként kell elérni a tárolót, nagyon lényeges, hogy a szóhoz tartozó byte-ok milyen sorrendben kerülnek tárolásra. Két változat használatos (2-5.ábra):

- a **'big endian'** tárolási formánál, a szó legmagasabb helyiértékű byte-ja kerül a legalacsonyabb című tárolóhelyre és a szó címzése a legalacsonyabb memóriacímmel történik(például az MC68000 processzorcsalád, az IBM 370-es gépek);
- a **'little endian'** tárolási formánál a szó legalacsonyabb helyiértékű byte-ja kerül a legalacsonyabb című tárolóhelyre és a szó címzése a legalacsonyabb memóriacímmel történik(például az i80x86 processzorcsalád, DEC VAX gépek).

A korábbi processzorok többnyire egyik, vagy másik tárolási és címzési módot használták, de az újabb processzorok általában választható módon használják az egyik, vagy a másik formát.



2-5.ábra: *'Big endian' és 'little endian' tárolási-címzési mód*

Lényeges jellemző a cím lehetséges mérete, azaz az, hogy hány bináris helyiértéket lehet felhasználni a cím értékének leírására. Ha ez pl. 16 bit lehet, akkor a maximális tárolóhely sorszám, azaz cím, $.2^{16}-1=65535$ lehet. Tehát ennél több tárolóhelyet közvetlen módon nem tud kezelni a processzor, ez alkotja a tárolóhelyek **címezhető tartományát**. Az íly módon értelmezett címtartomány alkotja egyúttal a **fizikai címek tartományát** is és a mögötte lévő, létező(installált), vagy nem létező tárolóeszköz a **fizikai tároló**, amely többnyire a főtár.

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

Tárolóeszközként kétféle típusú tárolót lehet megkülönböztetni:

- **Írható-olvasható tárolók**, amelyek általános tárolási célra használhatók. Írható-olvasható tárak a regiszterek, a cache-tárak, a főtár, a háttértárak, mert ezek esetében követelmény az adatbeírási és -olvasási lehetőség. A háttértárak kivételével, amelyek mágneses, vagy optikai elven működnek, a tárolók építőeleme a félvezető alapú RAM(Random Access Memory) tároló.

  A RAM tárolók egyik típusa az ú.n. dinamikus RAM(DRAM) tároló, amely alacsony teljesítményigényű, de tartalmát rövid idő alatt elveszti, ezért annak tartalmát ciklikusan fel kell újítani(refresh); a tárolók másik típusa a statikus RAM(SRAM) tároló, amely nem igényli az állandó adatújítást és magasabb működési sebességű, egyszerűbb időzítési megoldásokat igényel.

- **Csak olvasható tárolók(ROM-Read Only Memory)**, amelyek tartalmát a felhasználó közvetlenül nem tudja módosítani. Ezek között vannak olyanok, amelyek csak a gyártás során tölthetők fel(ROM) és vannak olyanok, amelyek a felhasználó által egyszer feltölthetők (PROM=Programmable ROM); illetve vannak speciális módon(ultraibolya fénnyel, vagy elektronikusan) törölhető, majd újraprogramozható tárak(EPROM=Erasable Programmable ROM).

A központi tárak működésének fontos jellemzője még az **elérési idő**(access time) és a **ciklusidő**(cycle time). Elérési idő az az időtartam, amely a kiolvasás megkezdése és az adatnak a tároló kimenetén való megjelenése között eltelik. A ciklusidő ennél valamivel hosszabb időtartam, mert magában foglalja a kiolvasás utáni **feléledési időt**(recovery time) is, amelyre egyes memóriáknak szüksége van a következő memóriához fordulást megelőzően. A statikus RAM(SRAM) tárolók elérési ideje és ciklusideje közel azonos értékű, míg a dinamikus RAM(DRAM) ciklusideje körülbelül kétszerese az elérési időnek.

A tárakba történő íráshoz, vagy olvasáshoz meg kell adni a keresett tárolóhely címét, amit a tárolóhoz tartozó **címregiszter**(MAR=Memory Address Register) fogad be és ennek tartalma vezérli a memória kiválasztó áramköreit. Az adatok számára a tároló bemenete/kimenete az **adatregiszter** (MDR=Memory Data Register), amely a beírandó, vagy kiolvasott adatot ideiglenesen befogadja.

c.)Másodlagos tárak, háttértárak

A számítógépek munkájához igen nagy mennyiségű adatot kell időnként tárolni. Ezek elhelyezésére a központi tár nem elegendő, illetve nem mindig célszerű minden adatot mindig a memóriában tartani. Ezért már igen korán igénybe vettek olyan, többnyire mágneses elven működő tárolóeszközöket, háttértárolókat(secondary memory, backing store), amelyeken nagy mennyiségű adat helyezhető el.

Mágnesszalagtár

A mágnesszalag(magnetic tape) az egyik legrégebbi másodlagos tárolóeszköz, amely kinézetében és tárolási elvében hasonló a közönséges magnószalaghoz, de lényeges különbség van az alkalmazott adatrögzítési módban. A számítógéphez használt mágnesszalagnál digitális jelrögzítést használnak, szemben a normál hangszalagok analóg jelrögzítési módjával.

Mágnesszalagtárat mikroszámítógépek mellett nem használnak, legfeljebb egyszerűbb személyi számítógépeknél kazetta formájában. Mikroszámítógépes környezetben mágnesszalag egyetlen alkalmazási területe az adatállományok archiválására szolgáló eszköz(streamer) adathordozójaként.

Nagygépek mellett is csökken a jelentősége, de egyszerűsége, olcsósága, viszonylagosan magas kapacitása, kis helyigénye miatt még alkalmazzák.

Mágneslemeztár

A mágneslemeztárak(magnetic disk) többféle változata alakult ki az idők során, de alapelvét tekintve, mindegyik hasonló módon működik.

A mágneslemezek esetében a mágnesezhető réteg egy lemez felületén van elhelyezve, amely lemez lehet merev, fémből készült(disk, harddisk, Winchester-disk), vagy készülhet műanyagból, hajlékony formában(floppy disk). Az adattárolás a lemez felületén, koncentrikus **sáv**ok (track-ek) mentén történik. A merevlemezek esetében általában több lemez alkot egy csomagot, amelyek felületén az egymás alatt elhelyezkedő sávokat együttesen **cilinder**nek(cylinder) nevezzük. A sáv kisebb adatterületekre, **szektor**okra(sector-ra) van felbontva, amelyek egyenként 512 byte hosszúságúak többnyire.

A **merevlemezek** állandó sebességgel forognak és az adatok írása/olvasása a felülettől néhány mikron távolságra lévő fej('repülő fej') segítségével történik. Minden lemezfelülethez tartozik egy író/olvasófej. A mikroszámítógépekben kizárólagosan használt merevlemez típus a pormentesen, zárt tokban elhelyezett merevlemez, az ú.n. winchester lemez. Mind a nagygépek, mind a mikroszámítógépek esetében használnak nem cserélhető, fixen beépített merevlemezeket és cserélhető lemezeket.

A merevlemezek kapacitása 20 Mbyte - 10 Gbyte között mozog.

A **hajlékonylemezek**(floppy disk) mágnesezhető réteggel ellátott vékony műanyag lemezek vékony, papir, vagy műanyag védőtokban elhelyezve. A lemezek írásakor/olvasásakor az író-olvasófej hozzáér a lemez felületéhez, ezért a lemezt csak akkor forgatja a meghajtó, amikor adatírás, vagy -olvasás zajlik. Így a lemez elérési ideje is lényegesen magasabb, mint a merevlemezeké.

A hajlékonylemez elsősorban a mikroszámítógépek adatbeviteli, -kihozatali adathordozója. A ma használt floppy disk-ek mérete 5.25", illetve 3.5" átmérőjű és kapacitásuk 360 Kbyte - 1.44(2.88) Mbyte között van. Használnak 21 Mbyte kapacitású, 3.5"-es lemezeket is.

A mikroszámítógépes környezetben használt mágneslemeztárak részletesebb ismertetésére a 7.fejezetben térünk vissza.

Optikai lemezek

Az optikai lemezek(optical disk) elterjedése az utóbbi időben következett be nem csak a hangrögzítés, de az adatrögzítés területén is. Valódi háttértárolókénti használatát még hátráltatja egyrészt az írható/olvasható változat kiforratlansága, másrészt a nagyobb elérési idő és kisebb adatátviteli sebesség, összehasonlítva a mágneslemezekkel.

Az optikai lemez felületén az adatrögzítés a hanglemezéhez hasonlóan, spirális pálya mentén történik. Az adatrögzítés az ú.n. mesterlemezre történik, amelyre lézersugár segítségével, a rögzítendő adatsornak megfelelően kicsiny lyukakat égetnek. Erről a lemezről készítik a nyomólemezt, illetve annak segítségével az adatlemezt. A műanyag lemez felületére egy igen vékony

fényvisszaverő fémréteg kerül, amelyet egy újabb, áttetsző műanyag réteggel védenek a sérülésektől.

A törölhető, azaz az írható/olvasható lemezek technológiája magneto-optikai elveket alkalmaz. A lemezek felülete egy vékony, speciális ritkafémmel készült, mágnesezhető réteggel van bevonva, amelyet lézerrel helyileg felmelegítenek és egyúttal mágneseznek is. A fém lehülte után a mágnesezettség megmarad, így az általa rögzített adatsor azután bármikor visszaolvasható.

A lemezek kapacitása igen nagy, 550-650 Mbyte között van.

A préselt, csak olvasható lemezek a **CD-ROM**-ok(Compact Disk ROM), míg a felhasználó által egyszer írható optikai lemezek a **WORM**(Write Once Read Many) lemezek.

d.)Perifériák

A számítógép külvilággal való kapcsolattartásának eszközei alkotják együtt a **perifériákat**(mint például: monitor, billentyűzet, nyomtató, stb.). A processzor a perifériális eszközöket egy-egy vezérlő egységen(controller) keresztül éri el és irányítja. Ezek a vezérlők vagy magában a perifériában találhatók és a központi egységgel egy szabványos csatlakozón keresztül tartanak kapcsolatot, vagy önálló egységként helyezkednek el a központi egységhez kapcsoltan.

A perifériák egyre több feladatot, vezérlési funkciót látnak el saját vezérlőjük irányítása alatt, azaz egyre 'intelligensebbek' lesznek. Ez esetben elegendő a szabványos csatlakozási felület(interface) kialakítása a központi egység és a periféria között.

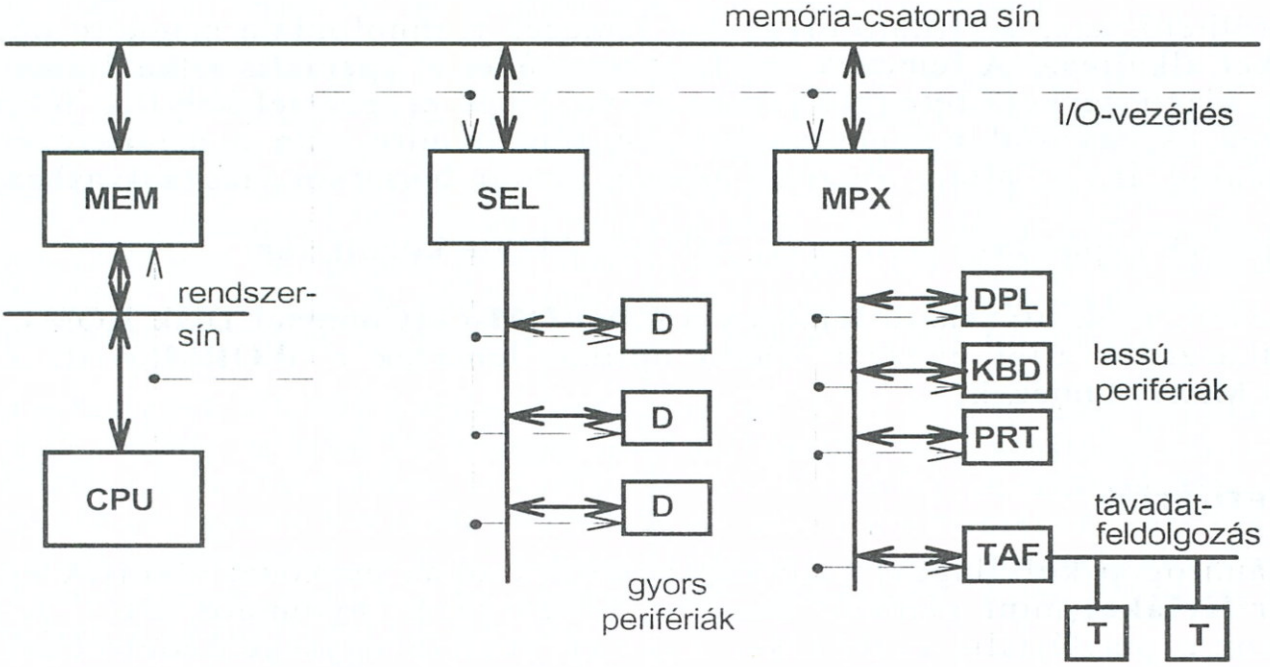
A perifériák kapcsolata a központi egységgel más a nagygépeknél('mainframe' számítógépek) és más a mikroszámítógépek esetében.

A **nagyszámítógépek** a csatornarendszert(channel system) használják fel a kapcsolat kialakítására. Az adatcsatornák önálló vezérlő egységgel rendelkeznek és megfelelő program alapján, önállóan irányítják az adatátvitel folyamatát a processzor munkájától függetlenül, a memória és a periféria között. Az adatátvitel indítása előtt a processzor az átvitel programját átadja a csatornavezérlőnek, majd elindítva azt, a továbbiakban a csatorna már önállóan dolgozik tovább, amíg be nem fejezi a kijelölt átviteli feladatot. Ekkor jelzést küld a processzornak, amely további igény esetén, ismét elindít egy adatátviteli folyamatot.

A gépek kétféle csatornát használnak:

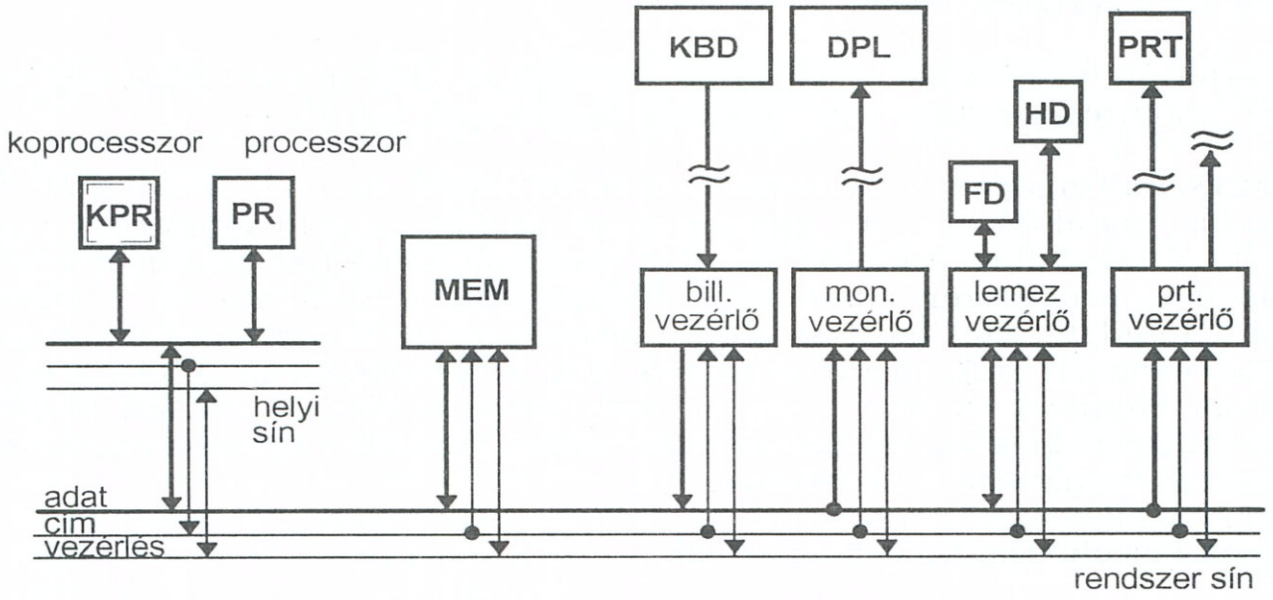
- **szelektor csatornát**, amely nagy sebességű perifériákat(háttértárak) kapcsol a processzorhoz és egyidőben csak egy perifériával tud kapcsolatot tartani;
- **multiplexor csatornát**, amely a lassú perifériákat(nyomtató, terminálok, stb.) köti össze a processzorral, mégpedig időosztásos üzemmóddal, azaz felváltva létesít kapcsolatot az egyes perifériákkal.

A nagygépes perifériakezelés módját mutatja be a 2-6.ábra.



2-6.ábra: *Csatorna-elvű perifériavezérlés*

A **mikroszámítógépek** perifériái a közös sínrendszerre(buszrendszer-re) csatlakozó vezérlőkön keresztül kapcsolódnak a processzorhoz. Ennek vázlata látható a 2-7.ábrán.



2-7.ábra: *Mikroszámítógép sínrendszerre épülő struktúrája*

A számítógépek környezetében leggyakrabban használt perifériákat az alábbiakban soroljuk fel, nem véve számításba a másodlagos tárakat, amelyek

ugyancsak a számítógép perifériájaként kezelendők. A felsorolt perifériális eszközök egy részéről a későbbi fejezetekben részletesebben is szó lesz, ezért itt csak a leglényegesebb jellemzőiket említjük meg.

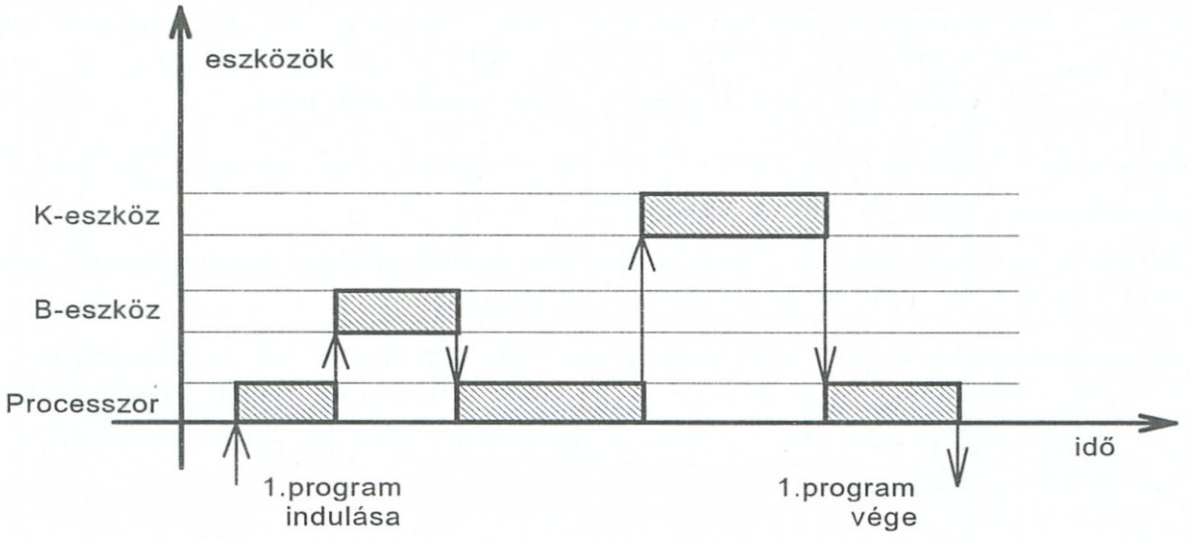
- képernyős kijelző(monitor, display), amely az adatok grafikus, vagy szöveges megjelenítésére szolgál;
- billentyűzet(keyboard), amely kisebb mennyiségű adat bevitelére, valamint a gép közvetlen vezérlésére szolgál;
- nyomtatók(printer), rajzgépek(plotter), amelyek az eredmények szöveges, vagy grafikus megjelenítésére használatosak nyomtatott formában;
- digitalizáló berendezés, amely rajzok pontjainak egyenkénti bevitelére, digitalizálására szolgál;
- szkenner(scanner), amely rajzok, grafikák, fényképek raszteres(bittérképes) bevitelét teszi lehetővé;
- egér(mouse), trackball, amelyek segítségével a grafikus képernyőn mozgatott 'kurzor'-ral, rámutatással kijelölhetők a műveletek, így a program, a gép irányítása billentyűzet nélkül történik;
- fényceruza(light pen) segítségével a képenyőre látszólag közvetlenül írhatunk, amelyet a gép érzékel és feldolgoz; ez a lehetőség különösen a tervező rendszerek esetében jelent nagy kényelmet a programok kezelésében;
- távközlési csatoló, modem, amelyen keresztül a számítógép közvetlenül kapcsolódhat a távközlési hálózatokhoz(kapcsolt telefonhálózat, bérelt adatvonalak, csomagkapcsolt hálózat) és ezeken keresztül más számítógépes rendszerekhez;
- hálózati csatoló, amellyel helyi számítógép hálózathoz(LAN=Local Area Network) kapcsolható a számítógép.

2.1.3.Hagyományos számítógépek korlátjai

A hagyományos felépítésű számítógép *eredeti formájának* legnagyobb hátránya, hogy a rendelkezésre álló erőforrásokat(processzor, memória, beviteli, kihozatali eszközök) igen rosszul használja ki, egy időben csak egy program végrehajtása lehetséges és az erőforrások nem megoszthatóak. A számítógép 'szűk keresztmetszete' a processzor és a memória közötti adatátvitel sebessége, a memória-sávszélesség.

A hagyományos számítógépek feldolgozási sebességének strukturális gyorsítására kevés lehetőség kínálkozik. Az erőforrások használatának mérsékelt párhuzamosítása lehetséges csak, mint például

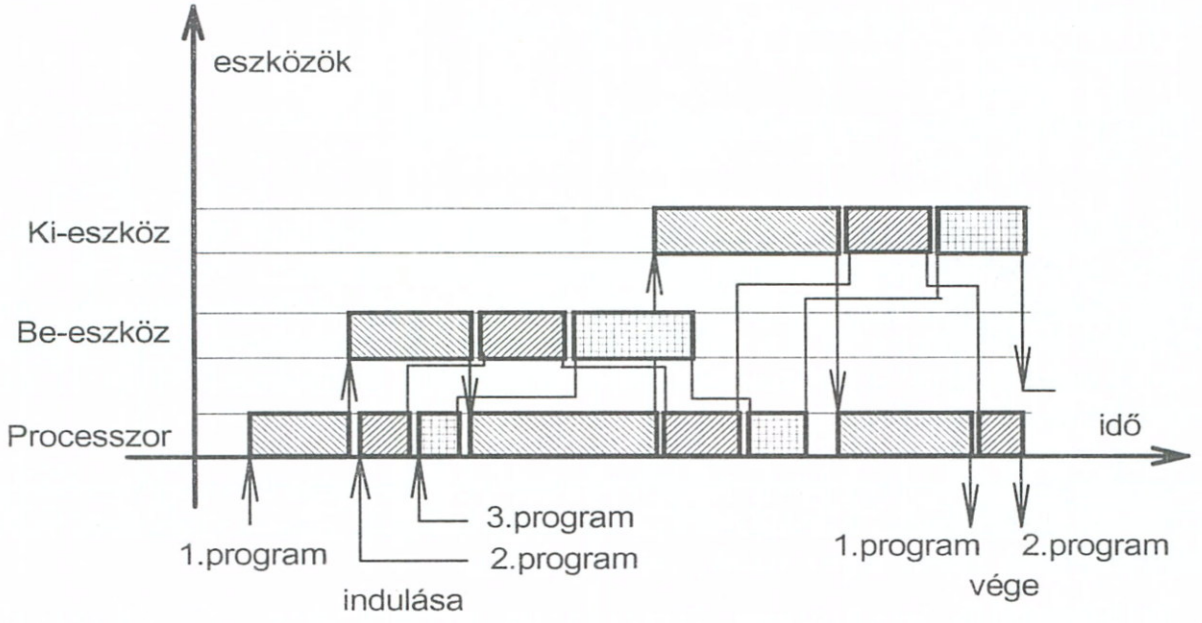
- a felhasználói programok látszólagos párhuzamos feldolgozása(multiprogramming),
- a felhasználói programok végrehajtásának és a beviteli/kiviteli műveleteknek az egyidejű elvégzése(I/O spooling),
- a processzor funkcionális egységeinek a többszörözése,
- egyes végrehajtási folyamatok átlapolt feldolgozása(pipelining).



2-8.ábra: *Kötegelt(soros) feldolgozás terhelési diagramja*

Az említett párhuzamosítások közül, itt csak a multiprogramozott működési forma által nyújtott gyorsítási lehetőséget vizsgáljuk meg röviden. Az erőforrások időbeni igénybevételét, egyszerűsített terhelési diagramját mutatja be egy program feldolgozása alatt a 2-8.ábra.

Az ábra vízszintes sávjai egy-egy erőforrás időbeli leterhelését mutatják, amely alapján látható, hogy azok kihasználtsága igen alacsony. Különösen hátrányos ez a nagy működési sebességű processzor esetében. Ezt a fajta feldolgozásmódot **kötegelt**, vagy soros **feldolgozásnak**(batch processing) nevezzük.



2-9.ábra: *Multiprogramozott feldolgozás terhelési diagramja*

A rendelkezésre álló eszközök jobb leterhelését megfelelő szervezéssel, amelyet egy bonyolult vezérlő programrendszer, az **operációs rendszer** valósít meg, javítani lehet. Ennek segítségével a szabad időtartamok felhasználhatók más, következő programok feldolgozására. Ez egy látszólagos, párhuzamos feladatmegoldást erdményez, azonban a valóságban csak az erőforrások váltakozó használata valósul meg az operációs rendszer vezérletével.

A feladatoknak, programoknak ilyen megoldású párhuzamos feldolgozását **multiprogramozásnak**(multiprogramming), vagy multitaszkos(multitasking) feldolgozásnak nevezzük. Ezt mutatja be a 2-9.ábra.

Látható, hogy evvel a módszerrel az erőforrások jobb leterhelése valósítható meg, azonban ennek a feltétele *egyrészt*, az ezt megvalósító operációs rendszer, *másrészt* egy olyan processzor struktúra, amely lehetővé teszi a feladatváltásokat, az egyes programok által használt erőforrások védelmét a többi programtól, állapotjelzők előállítását és tárolását. A korszerű processzorok már rendelkeznek ezekkel a funkciókkal, így a többfeladatos feldolgozásra lehetőséget biztosítanak.

A multiprogramozott üzemmód többféle változata használatos, amelyek alapvetően az erőforrások szétosztási módjában különböznek egymástól. Eszerint a következő elosztási módszereket alkalmazzák:

- **prioritásos elv**; az egyes feladatok fontossága különböző lehet és a szabad erőforrást, amennyiben több feladat is igényli egyidejüleg, a legmagasabb prioritású feladat kapja meg;

- **időosztásos üzemmód**(time-sharing); ennél a feldolgozási módnál a feladatok egyenlő időközönként férnek hozzá a processzorhoz, míg az egyéb erőforrások használata részben prioritásos elvű;

- **időazonos**, vagy **valós idejű**(real-time) **feldolgozás**; ez esetben egy-egy program kiemelt fontosságú, ami azt eredményezi, hogy amennyiben ez a program kéri a processzor és a többi erőforrás használatát, akkor azonnal át kell engedni azokat ennek a kiemelt feldolgozásnak; ilyen jellegű feldolgozás pl. minden folyamatirányító feladat(gyártósorok, gyártási folyamatok vezérlése).

A hagyományos felépítésű gépek teljesítőképességét igazából csak strukturális változtatásokkal lehet növelni.

2.2.MIKROGÉPEK SZOKÁSOS FELÉPÍTÉSE

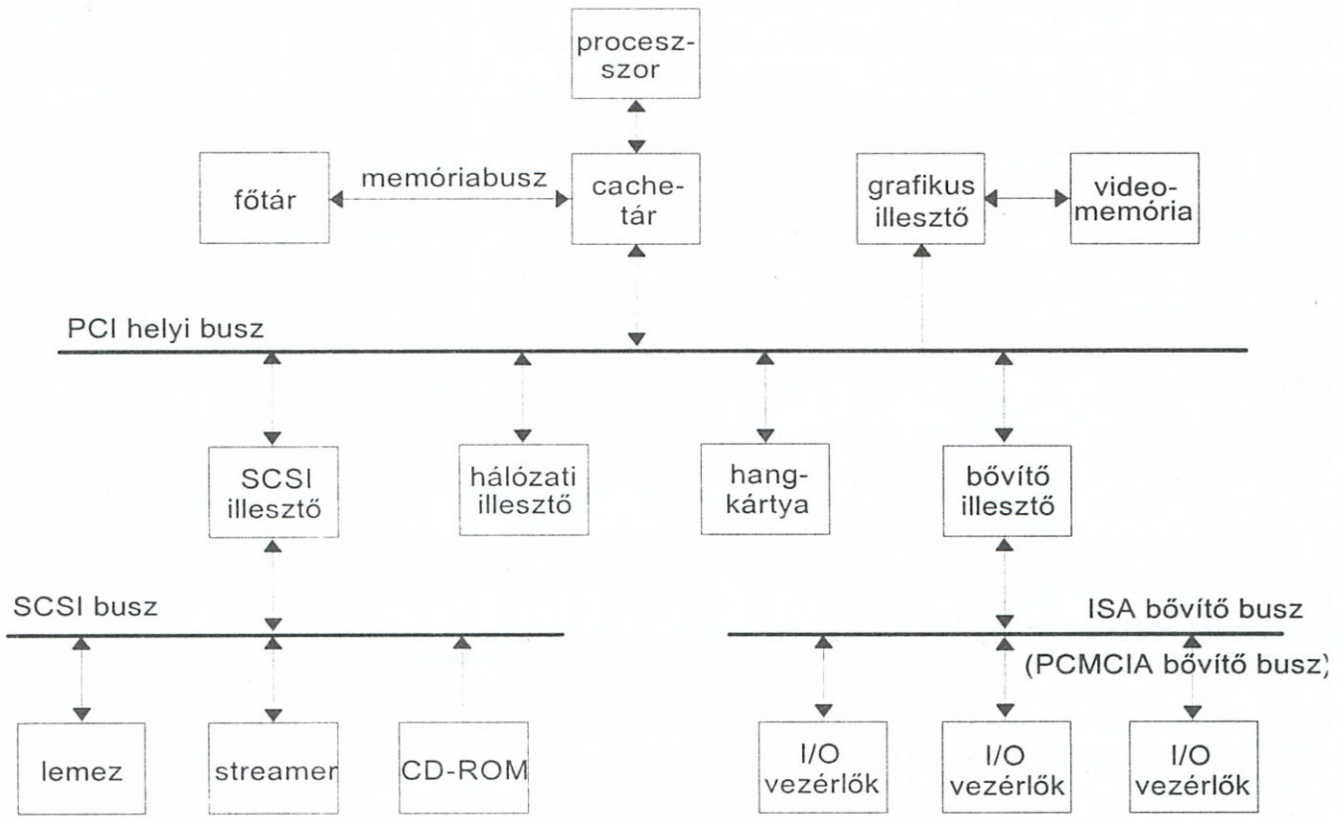
A számítógépek általános, bevezető áttekintése lezárásaként ebben a részben a mikroszámítógépek felépítésére jellemző, szokásos eltérésekre kívánunk röviden utalni. A további fejezetekben az egyes részegységeket részletesen tárgyaljuk.

A számítógépek funkcionális felépítését, amely a mikroszámítógépekre ugyanúgy érvényes, korábban, a 2-1.ábrán már bemutattuk. A mikroszámítógépek szokásos rendszertechnikai felépítését a 2-10.ábra mutatja be.

A mikroszámítógépek legfőbb rendszertechnikai jellemzője a **sínrendszer** (bus system). A mikroszámítógép sínrendszere a gép funkcionális egységeit kapcsolja össze egy szabványositott(általánosan elfogadott) vezetékrendszerrel. A vezetékek száma és funkciója meghatározott és ez a gyártók által elfogadott, így a gép sínrendszeréhez könnyen csatlakoztatható bármilyen gyártótól származó részegység.

A sínrendszert három sín alkotja, amelyek mindegyikének jellemzője, hogy hány vezetéket foglal magában, azaz mekkora a **sín 'szélessége'**:

- az **adatsín**, amelyen az adatokat továbbítja a gép, általában a processzor és a gép más egységei között; szélessége: 16-32 bit, ma inkább már 32 bit;
- a **címsín**, amelyen keresztül a memória, vagy más eszközök címeit továbbítja a processzor; szélessége meghatározza a memória címezhető tartományát, a ma használatos processzorok címsín szélessége 32 bit többnyire;
- a **vezérlősín**, amelyen át a vezérlő jeleket küldi (és fogadja) a gép különböző egységei számára a processzor.

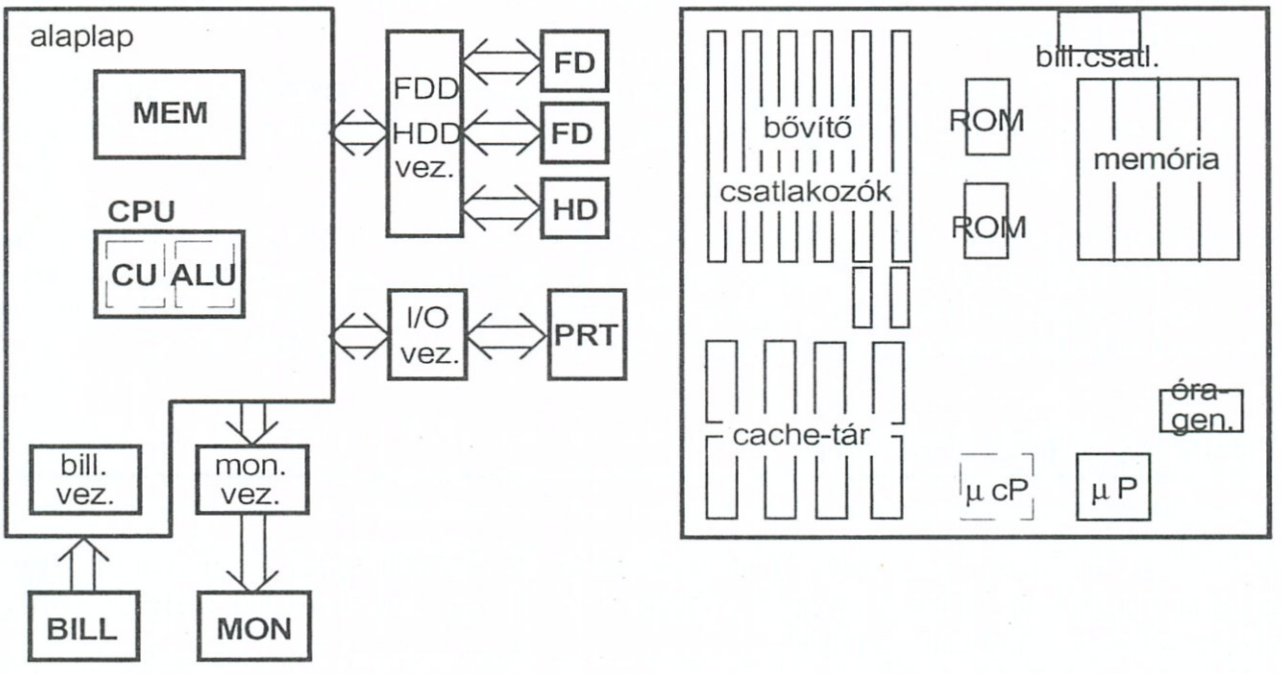


2-10.ábra: *Mikroszámítógép sínrendszerre épülő rendszertechnikai felépítése*

A processzor közvetlenül az ú.n. **'lokális busz'**-ra, helyi sínre kapcsolódik, amelyen gyorsabb adattovábbítás valósítható meg, mint a rendszersínen keresztül. A lokális sínre kapcsolódik a lebegőpontos műveleteket végrehajtó társprocesszor is, ha van egyáltalán a gépben.

A mikroszámítógépek részegységei többnyire egyetlen dobozban vannak elhelyezve, amelynek csatlakozóin keresztül kapcsolódhatnak a géphez a külső perifériák(monitor, billentyűzet, egér, nyomtató, modem, stb.). A processzor, a memória és néhány egyszerűbb vezérlőegység, ugyanazon nyomtatott-áramköri lapon, az **alaplapon**(motherboard, main board) van elhelyezve, amelyen **bővítő csatlakozók** is találhatók, a további vezérlő kártyák(monitor kártya, lemezvezérlő kártya, I/O kártya, hálózati kártya, stb.) csatlakoztatására.

A mikroszámítógépek szokásos rendszerépítési megoldását mutatja be a 2-11.ábra. Az ábrán látható egy alaplapi elrendezés nagyvonalú, beültetési vázlata is.



a.) alaplap rendszerkapcsolatai b.) alaplap elrendezése

2-11.ábra: *Mikroszámítógép alaplapja és kapcsolatai*

Az alaplapon található egységek tehát:

- a **processzor**, amely többnyire Intel i286/386/486/Pentium-os(IBM-PC/AT kompatibilis gépek), vagy Motorola MC68020/68030/68040-es (Apple Macintosh, Sun Microsystems, stb) processzor;
- a **memória**; az alaplapon elhelyezhető memória nagyság 4,16,32,64 Mbyte, aminek következtében ma már nem használnak memória bővítő-kártyát;

- **kiegészítő vezérlők**; amelyek speciális feladatokat látnak el, vagy egyszerűbb, kiforrott, egységesített jellemzőkkel rendelkező perifériális eszközöket kapcsolnak a processzorhoz:
  - = billentyűzet vezérlő,
  - = beépített hangszóró vezérlője,
  - = óraáramkör,
  - = közvetlen memóriaelérés(DMA) vezérlője,
  - = megszakításvezérlő.
- **bővítő csatlakozók**, amelyeken keresztül további vezérlőkártyák kapcsolhatók a sínrendszerhez; ezek a vezérlők:
  - = *monitorvezérlő kártya*, amely a felhasználók eltérő igényeiből eredő, legkülönbözőbb monitorok használatát teszi lehetővé;
  - = *lemezegység*(merevlemez, hajlékonylemez) *vezérlője*, amelyen együtt található a kétféle lemezegység vezérlőegysége; mivel a mai lemezegységek már olyan saját vezérlővel rendelkeznek, amelyek már egyszerűen programozhatók, ezért többnyire az alaplapi vezérlő egy kártyán található a párhuzamos és soros kimeneteket vezérlő áramkörökkel;
  - = *(multi) I/O kártya*, amely a gép párhuzamos(nyomtató csatlakoztatási helye) és soros kimeneteit(egér, modem, botkormány, stb. csatlakoztatási helye) vezérli.

A mikroszámítógépek igen sokféle formában kerülnek piacra, a nagyteljesítményű munkaállomásoktól(work station) kezdve, az asztali(desktop) gépeken, táskagépeken(laptop), könyvgépeken(notebook) keresztül egészen a menedzserkalkulátorokig bezárólag.

2.3.TIPIKUS STRUKTÚRÁK

A következő néhány pontban a legfontosabb számítógép struktúrák lényegi jellemzőit foglaljuk össze annak érdekében, hogy áttekintést adhassunk azok legjellemzőbb tulajdonságairól.

2.3.1.Neumann-elvű számítógépek

A Neumann-elvű, azaz hagyományos számítógépek legfontosabb jellemzőit a 2.1.pontban már ismertettük, ezért itt csak megismételjük azok lényegét:

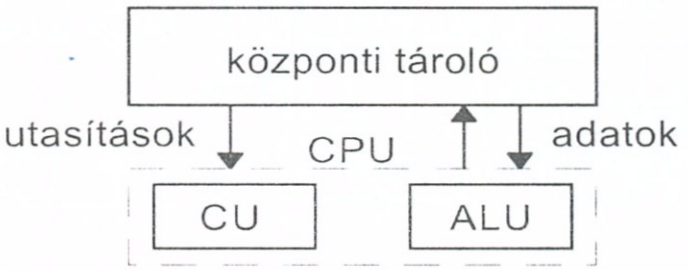
- a gép vezérlése **tárolt program** alapján történik;
- a gép irányítása ú.n. **'vezérlésáramlásos'**(control-flow, control-driven) rendszerű, azaz a számítógép vezérlő egysége a tárolt program utasításait egyenként sorra véve oldja meg a kívánt feladatot; az automatikus programvégrehajtás egyszerűsítése végett a vezérlő egységben egy utasításszámláló regiszter tárolja a soronkövetkező utasítás tárolóbeli helyének címét;

- a gép tartalmaz egy **közös tárolót**, amely egyaránt tárolja a végrehajtandó program utasításait, valamint az utasítások által feldolgozandó adatokat is; a program és az adatok kódolására egyaránt bináris(két jelből álló) kódrendszert alkalmaznak;

- a program utasításai által megkívánt aritmetikai és logikai műveletek elvégzésére egy önálló egység, az **aritmetikai és logikai műveletvégző egység** szolgál;

- az adatok és a program **bevitelére/kihozatalára önálló egységek** szolgálnak.

A teljesítmény növelésében mindenkor akadályként jelentkezik a processzor és a tároló közötti adatátviteli sebesség korlátozott volta, amelyet még hangsúlyosabbá tesz a közös program- és adattárolás ténye, azaz az utasítások soros feldolgozásának szükségszerűsége. A processzor és a tároló közötti kapcsolatot vázolja fel a következő, 2-12.ábra.



2-12.ábra: *Neumann-elvű gépek tárolókapcsolata*

A Neumann-elvű gépek a folyamatok kezelése szempontjából(M.J.Flynn csoportosítása szerint - 1.3.pont), az *'egy utasításfolyam egy adatfolyam'*(SISD) csoportba tartoznak. A működés gyorsításához, a teljesítmény növeléséhez, a tevékenységek párhuzamosítására kevés lehetőség van. Ilyen párhuzamosítási lehetőségek:

- a számítógép erőforrásainak(processzor és egyéb eszközök) egyenletesebb leterhelésére a **multiprogramozott üzemmód** alkalmazása;

- a funkcionális egységek(pl.: műveletvégző egységek) többszörözése, **multifunkciós processzor** kialakítása;

- a processzor tevékenységének és az I/O műveleteknek átlapolt végrehajtása(**'spooling' technika** alkalmazása);

- az utasítások és az aritmetikai műveletek átlapolt feldolgozása, **'pipelining' feldolgozás**.

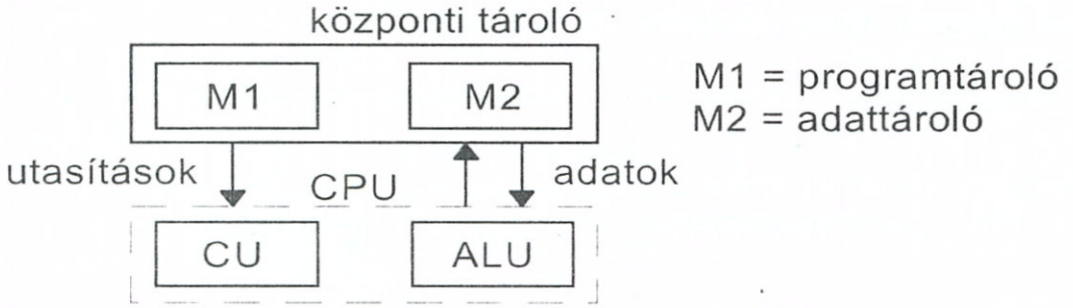
A Neumann-elvű, hagyományos számítógépek működésének részletes tárgyalására a 3.fejezetben kerül sor.

2.3.2.Harvard struktúrájú számítógépek

A Harvard struktúrájú számítógépek felépítése ugyanaz, mint a Neumann-elvű gépeké, avval a különbséggel, hogy külön program- és külön adattárolót használ a processzor. A két tárolási funkció szétválasztásával csökken a közös tároló és a közös sínrendszer használatából eredő szűk keresztmetszet hatása, így növelhető a gép teljesítménye.

A folyamatok kezelése szempontjából a Harvard struktúrájú számítógépek is a SISD csoportba tartoznak.

A külön tárolók használata egyúttal megszünteti a programutasítások felülírásának lehetőségét is, így a futó program nem tudja módosítani önmagát. A létrejött strukturális változást mutatja be a 2-13.ábra.



2-13.ábra: *Harvard struktúrájú számítógépek tárolókapcsolata*

2.3.3.Vektorszámítógépek

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

A folyamatok kezelése oldaláról, ez tulajdonképpen az *'egy utasításfolyam, több adatfolyam'*(SIMD) besorolásnak felel meg.

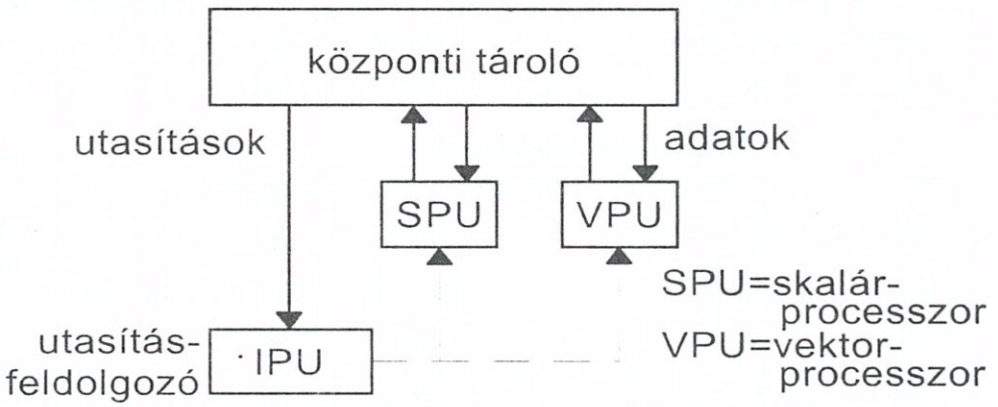
A teljesítmény növelését *egyrészről* az segíti elő, hogy ugyanazt a műveletet (utasítást) kell végrehajtani az egymást követő vektorelemeken és ehhez elegendő az utasítást csak egyszer lehívni a tárolóból; *másrészről* az segíti, hogy a vektor egymást követő elemeinek feldolgozása átlapolható és így egy adatpipeline alakítható ki. A vektorműveletek előírását külön vektorutasítások használata teszi lehetővé.

A vektorprocesszorok teljesítménye függ a kialakított adatpipeline hosszától. Minél több elem feldolgozását lehet átlapolni, annál inkább növekszik a processzor teljesítőképessége. Ugyanakkor meg kell oldani a vektorelemek tárolóból történő lehívásának és visszatöltésének hatékony módját is, mivel

a tároló elérési ideje jelentősen csökkentheti a számítógép teljesítményét. Ennek megakadályozására *egyrészről* a műveletvégző egység előtt és után ú.n. regiszterláncot alkalmaznak az adatok ideiglenes tárolására, *másrészről* a tárolóterületet több részre felosztva(memory interleaving), a vektorelemeket sorra az egymást követő memóriablokkokból készíti elő a processzor.

A vektorprocesszorok teljesítőképességét erősen rontja az, ha skalár mennyiségekkel kell dolgoznia, mert áthaladásuk a feldolgozó láncon időveszteséget okoz. Ezért a vektorprocesszor mellett, a skalár mennyiségekkel való műveletvégzéshez külön processzor áll rendelkezésre.

A processzorok tárolókapcsolatát a 2-14.ábra vázlata tükrözi vissza.



2-14.ábra: *Vektorszámítógépek tárolókezelése*

A vektorszámítógépek közé tartoztak az 1970-es évek végén elkészült Cray-1 (Cray Research Inc.), valamint CDC Cyber 205(Control Data Corporation) típusú szuperszámítógépek.

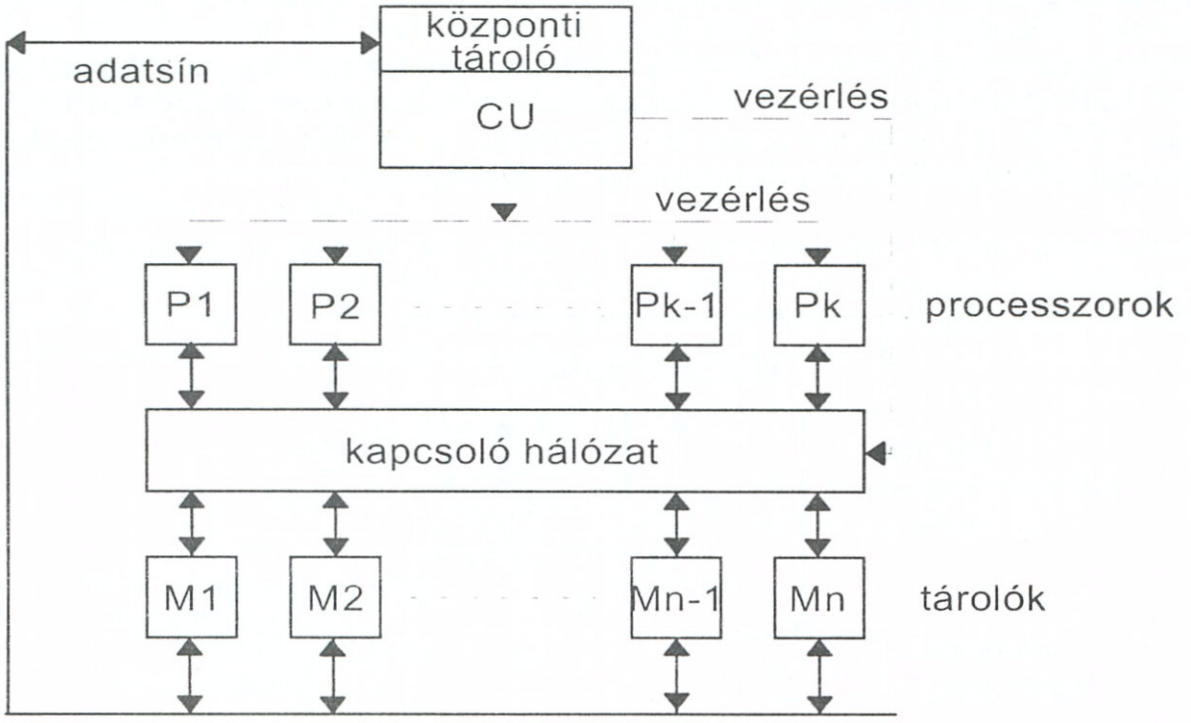
2.3.4.Tömbprocesszoros számítógépek

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

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

A processzorokat és a memóriamodulokat egy vezérelhető kapcsolóhálózat köti össze, amely lehetővé teszi bármelyik processzor összekapcsolását bármelyik memóriamodullal. A számítógépek strukturális felépítését a 2-15.ábra rajza mutatja be.

A teljesítőképesség maximumának elérése érdekében a processzorok és a memória, valamint a központi vezérlő egység és a processzorok között nagysebességű adatátvitelt kell megvalósítani.



2-15.ábra: *Tömbprocesszoros számítógépek tárolókezelése*

A tömbprocesszoros számítógépek három fő csoportja különböztethető meg:

- a **tömbprocesszoros** számítógépek alapformájánál('array computer'), a műveletvégző egységek egymástól független módon dolgoznak ugyanazon utasítás végrehajtásán és adataikat a kapcsolódó memóriamodulból veszik ki és ugyanoda teszik vissza, szükség esetén a kapcsolóhálózat segítségével továbbítva azokat valamely más egységhez(2-14.ábra); ebbe a körbe sorolhatók a Borroughs ILLIAC-IV(1972), a Borroughs Scientific Processor(BSP, 1977-1980), valamint az ICL DAP(1980) számítógépek;

- az **asszociatív**('associative processor') tömbszámítógépek esetében, a hagyományos, cím szerinti adatvisszakeresés helyett a tartalom szerinti címzést(content addressing) alkalmazzák; az ilyen típusú tárolás és visszakeresés az összehasonlítási, hasonlósági feladatok megoldásánál, alakfelismerési problémáknál alkalmazható előnyösen; ilyen típusú számítógépek a STARAN(1971), a PEPE(1976) szuperszámítógépek(Goodyear Aerospace Corporation);

- a **szisztolikus**('systolic array') tömbszámítógépek processzorai csak a szomszédos egységekkel tartanak kapcsolatot és műveleteik elvégzése után, eredményeiket a szomszédos egységeknek adják át további feldol-

gozásra; a külvilággal csak a processzortömb szélső elemei tartanak kapcsolatot, azaz csak ezek alkalmasak az I/O feladatok elvégzésére.

2.3.5.Üzenetátadásos számítógépek

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

- *egyrészt*: milyen módon használják a processzorok a tárolót; ugyanazt a memóriát használja-e megosztott módon mindegyik egység(shared memory), vagy minden processzor számára önálló memória használata lehetséges(disjoint, distributed memory);

- *másrészt*: milyen módon létesítenek egymással kapcsolatot a processzorok; megosztott sínhasználat(shared bus system) révén, avagy közvetlen kapcsolat kiépítésének lehetőségével (interconnection network).

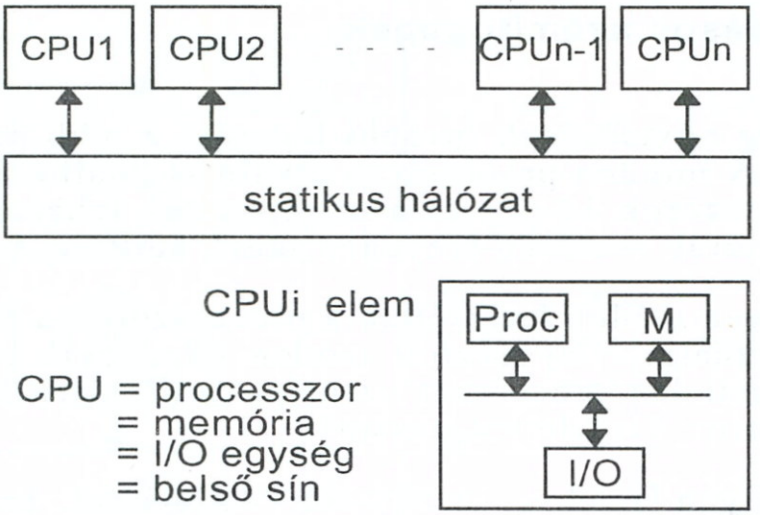
A multiprocesszoros gépek, a folyamatok kezelése szempontjából(M.J. Flynn csoportosítása szerint), a *'több utasításfolyam, több adatfolyam'*(MIMD) kategóriájú számítógépek közé tartoznak.

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

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

Elosztott memória esetén(2-16.ábra) a statikus kialakítású hálózatok segítségével létrehozott egy-, két-, három-, vagy többdimenziós(hiperkocka) csomópont-elrendezések jelentősége kiemelkedő. Mivel a csomóponti processzorok azonos kialakításúak, mindegyikhez kapcsolható I/O csatorna és így a rendszer I/O átvitele is magas lehet. A hiperkocka architektúra jellegzetes képviselői a Cosmic Cube(California Institute of Technology - 1983), az Intel iPSC/1, iPSC/2(1985, 1986) számítógépek. Külön megemlítendő, az INMOS cég által ilyen célra kifejlesztett egyetlen 'chip'-en elhelyezett RISC-elvű processzorból, memóriamodulból, belső sínrendszerből és DMA-beviteli/kiviteli vezérlőegységből álló komplexuma, a T212, vagy T414 típusú **transzputer**(transputer), amelyből tetszőleges számú egység építhető össze és együttesen igen nagy teljesítményre képesek.

Multiprocesszoros, statikus kapcsolati rendszerrel rendelkező architektúra vázlatát mutatja be a 2-16.ábra.



2-16.ábra: *Multiprocesszoros architektúra statikus kapcsolati rendszerrel*

Megosztott memóriahasználat mellett, a megfelelő teljesítőképesség csak több, párhuzamosan kiépíthető kapcsolat esetében érhető el. Az egyes egységek(processzor, memória) között, a feladattól függő módon kialakítható, dinamikus átviteli rendszer léte szükséges.

A különböző igények kielégítésére, az egyszerűbbtől a bonyolultabbakig, többféle hálózati kapcsolati rendszert fejlesztettek ki az idők során(így pl.: az 'omega', a Banyan, a 'perfect shuffle', a Benes, a 'crossbar', stb. típusú kapcsolóhálózatokat).

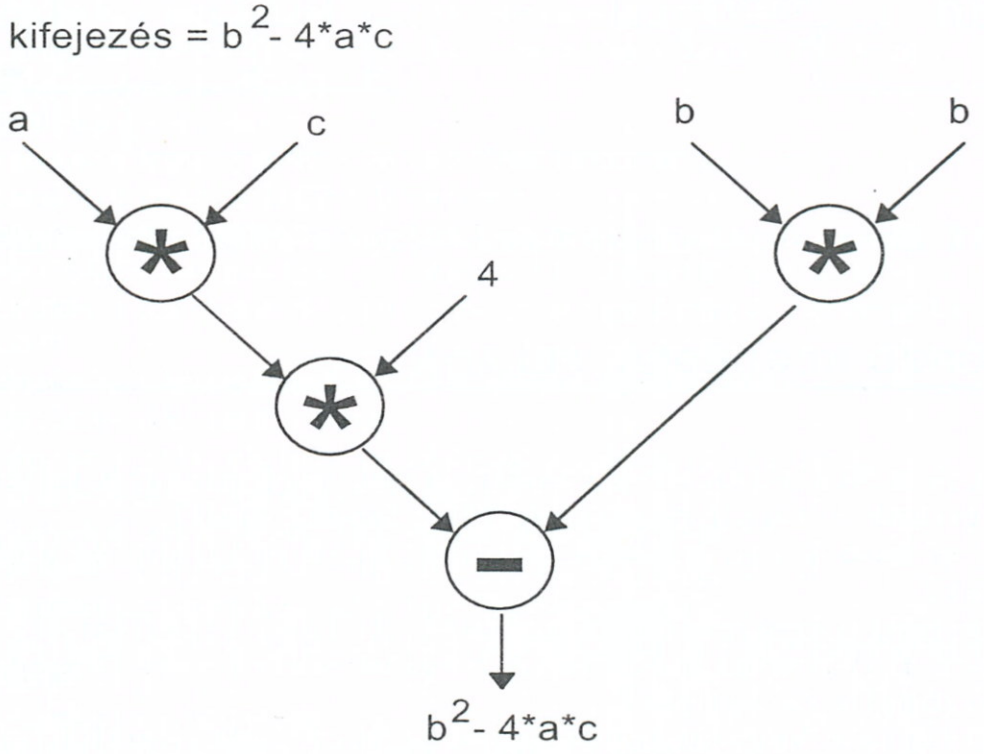
2.3.6.Adatvezérelt számítógépek

Az adatvezérelt(data-flow, data driven) számítógépek architektúrája alapjaiban különbözik a Neumann-elvű gépekétől. A vezérlésáramlásos(hagyományos, Neumann-elvű gépek) számítógépek soros utasításfeldolgozása helyett, a feladatokhoz szükséges adatok rendelkezésre állásától függ a műveletek végrehajtása.

A számítógép logikai struktúráját az elvégzendő műveletek egymáshoz kapcsolódását leíró **adatáramlási gráf** határozza meg. Az adatáramlási gráf csomópontjaihoz(nodes) vannak hozzárendelve az elvégzendő műveletek(utasítások). Egy-egy ilyen hozzárendelés elemi aritmetikai és logikai műveletek elvégzését, több befutó adat közüli választást(merge), egy adat valamely irányba történő irányítását(switch) eredményezi. A gráf éleihez a csomópont által igényelt adatok(token-ek), illetve a művelet eredményeként keletkező adat(ok) van(nak) hozzárendelve.

A másodfokú egyenlet megoldásánál használt gyök alatti mennyiség kiszámítását mutatja be a 2-17.ábra nagyon egyszerű adatáramlási gráfja.

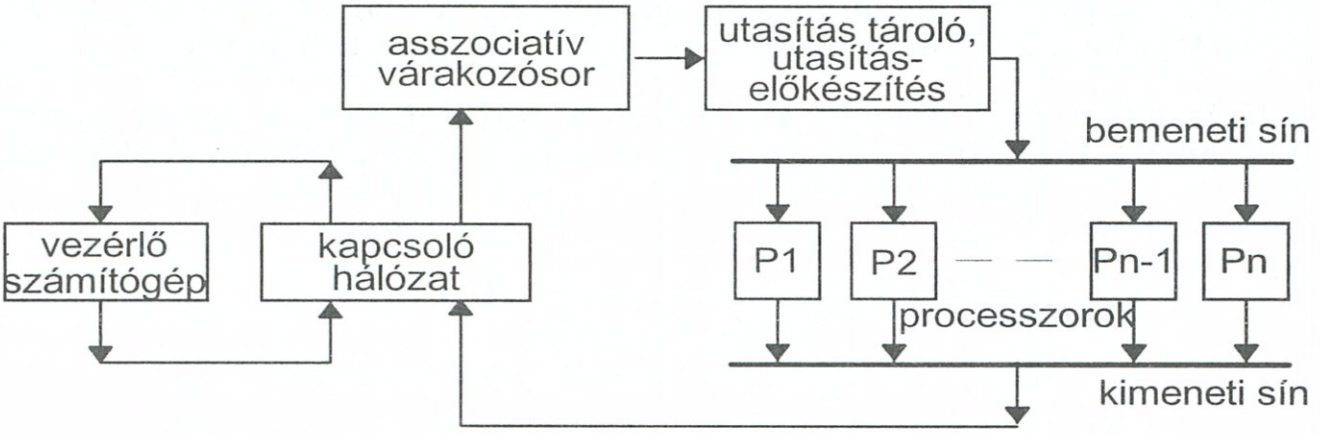
Az adatáramlási gráf alapján készíti el a fordítóprogram annak programgráfját, amely kijelöli a műveletet és a kapcsolódó adatcsomagokat(tokeneket).



2-17.ábra: *Adatáramlási gráf*

Az adatvezérelt számítógépek multiprocesszoros rendszerek, amelyben a processzorokat egy kapcsolóhálózaton keresztül kötik egymáshoz és minden ütemben az adatok(tokenek) egy-egy processzor-pár között mozognak a token-ekben előírt módon(2-18.ábra).

Minden körben az input-adatok és az eredményadatok kiegészítésre kerülnek a következő elvégzendő utasítás azonosítójával, majd a megjelölt utasítás alapján, a rendszer összepárosítja avval a tokennel, amelynek utasításazonosítója ugyanaz. Ezt követően, a programtárolóból kikeresi az utasítás műveleti kódját és hozzáteszi az adatcsomaghoz, majd ezt még kiegészíti az eredmény következő felhasználási helyének címével. Az így összeállított csomag kerül a processzorba, ahol az előírt utasítás végrehajtása megtörténik. A kapott eredmény a kapcsolóhálózat segítségével jut tovább a következő végrehajtási lépés előkészítési fázisába, vagy kerül ki a gépből.



2-18.ábra: *Adatvezérelt számítógép működési vázlata*

2.4.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Melyek a Neumann-elvű számítógépek legfontosabb jellemzői?
2. Mit értünk a 'tárolt programú' tulajdonság alatt?
3. Milyen hatása van a közös tárolóeszközű program- és adattárolásnak?
4. Mit értünk vezérlésáramlásos(control-flow) vezérlési elv alatt?
5. Milyen folyamatot modellez a Neumann-elvű számítógépek funkcionális felépítése?
6. Rajzolja le a hagyományos számítógépek funkcionális felépítését!
7. Melyek a hagyományos számítógép funkcionális egységei?
8. Vázolja fel egy mai számítógép rendszertechnikai felépítését.
9. Párosítsa össze a következő rövidítéseket a nekik megfelelő fogalmakkal! CPU, CU, ALU, PC; utasításszámláló regiszter, műveletvégző egység, központi egység, vezérlő egység.
10. Párosítsa össze a következő rövidítéseket a nekik megfelelő fogalmakkal! IR, CPU, AC, RAM; utasításregiszter, akkumulátor regiszter, közvetlen elérésű tároló, központi egység
11. Milyen főbb egységei vannak a központi egységnek?
12. Vázolja fel a központi egység logikai vázlatát!
13. Melyek a központi egység legfontosabb regiszterei? Sorolja fel ezeket!
14. Mire szolgál az utasításszámláló regiszter(PC)?
15. Mire szolgálal az utasításregiszter(IC)?
16. Mire szolgál az akkumulátor regiszter(AC)?
17. Mire szolgál az állapotjelző regiszter?
18. Sorolja fel az utasításfeldolgozás elemi lépéseit!
19. Adja meg az utasításfeldolgozás elemi lépéseinek helyes sorrendjét! (műveleti előírás értelmezése, művelet végrehajtása, utasítás előkészítése, adatok előkészítése a műveletekhez, következő utasítás címének a meghatározása, eredmény visszaírása)

20. A műveleti vezérlésre milyen módszereket használnak és mi ezek lényege?
21. Melyik vezérlési mód adja a gyorsabb utasításfeldolgozást, a huzalozott, vagy a mikroprogramozott?
22. Mit értünk bit alatt?
23. Mit értünk byte alatt?
24. Mit értünk cím alatt?
25. Mit értünk a tárolók címtartománya alatt?
26. Mi a különbség a 'little endian' és a 'big endian' címzési-tárolási forma között?
27. Mit értünk szó alatt?
28. Mi a különbség a RAM és a ROM tárolók között?
29. Mire szolgálnak a ROM tárolók?
30. Milyen típusú ROM tárolókat ismer?
31. Milyen főbb csoportjai vannak a tárolóeszközöknek?
32. A központi tár megvalósításához milyen tárolóeszközöket használnak?
33. Mit értünk a tárolóeszközök elérési ideje alatt?
34. Mit értünk ciklusidő alatt?
35. Milyen regiszterek kapcsolódnak a központi tárhoz?
36. Mi a feladata a címregiszternek?
37. Mely eszközöket soroljuk a perifériák körébe?
38. Ismertesse a mágneses tárolóeszközök jellemzőit!
39. Ismertesse az optikai adattárolás lényegét!
40. Milyen kapcsolati rendszert alkalmaznak a nagyszámítógépek körében?
41. Mi a csatorna lényege?
42. Mit értünk szelektor csatorna alatt?
43. Mit értünk multiplexor csatorna alatt?
44. Milyen eszközökkel létesít kapcsolatot a szelektor csatorna?
45. Milyen eszközökkel létesít kapcsolatot a multiplexor csatorna?
46. Milyen erőforrás megosztási elvet használnak a multiplexor csatorna esetében?
47. Párosítsa össze a perifériákat és a csatorna típusokat! (szelektor csatorna, multiplexor csatorna; lassú perifériák, gyors perifériák)
48. Milyen perifériális kapcsolati rendszert alkalmaznak a mikro-számítógépek esetében?
49. Soroljon fel legalább ötféle perifériális eszközt.
50. Mire szolgálnak a modemek?
51. Mire szolgálnak a hálózati csatolók?
52. Mi képezi a hagyományos számítógépek használatának szűk keresztmetszetét?
53. Milyen módszerekkel növelhető a hagyományos számítógépek kihasználtságának mértéke?
54. Mit értünk kötegelt feldolgozás(batch processing) alatt?
55. Mit értünk multiprogramozás alatt?
56. Több feladat feldolgozásánál melyik módszer nyújt jobb kihasználtságot: a kötegelt feldolgozás, vagy a multiprogramozás?
57. Multiprogramozásnál milyen módszereket használnak a processzor idejének a szétosztására a különböző feladatok között?
58. Mit értünk a prioritásos multiprogramozási módszer alatt?

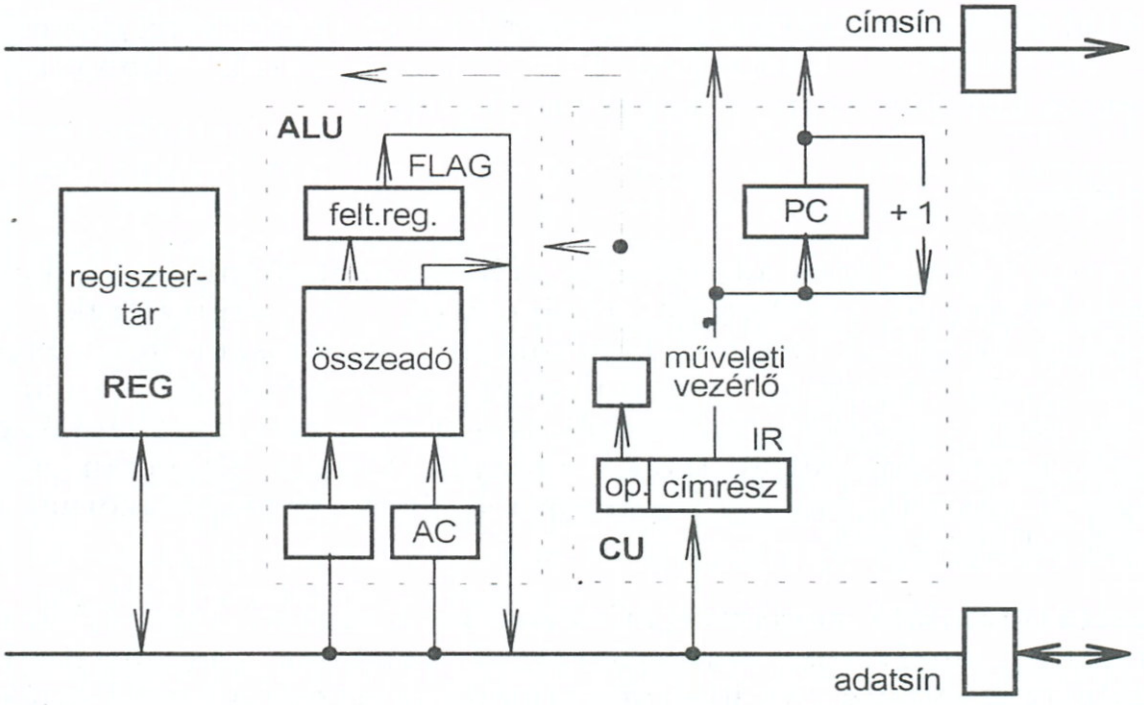
59. Mit értünk az időosztásos(time-sharing) multiprogramozási módszer alatt?
60. Mit értünk a valós idejű(real-time) multiprogramozási módszer alatt?
61. Ismertesse a mikroszámítógépek szokásos felépítését, részeit!
62. A folyamatok kezelése szempontjából milyen csoportba sorolhatók a Neumann-elvű számítógépek?
63. Melyek a Neumann-elvű számítógépek legfontosabb jellemzői?
64. Miben különbözik a Neumann-elvű számítógép struktúrája a Harvard-struktúrától?
65. Milyen lehetőségei vannak a folyamatok párhuzamosításának?
66. A folyamatok kezelése szempontjából milyen csoportba sorolhatók a Harvard-struktúrájú számítógépek?
67. Miben egyezik meg a Harvard-struktúrájú és a Neumann-elvű számítógépek működése?
68. A folyamatok kezelése szempontjából milyen csoportba sorolhatók a vektorszámítógépek?
69. Mi jellemzi a vektorszámítógépek működését?
70. Miért szükséges a vektorszámítógépekben a skalár-processzor használata?
71. A folyamatok kezelése szempontjából milyen csoportba sorolhatók a tömbprocesszoros számítógépek és milyen típusait ismeri?
72. Műveletvégzés szempontjából mi jellemzi a tömbproceszszoros számítógépeket?
73. Vázolja fel a tömbprocesszoros architektúrák processzor-memória kapcsolatát tükröző ábrát.
74. Nevezzen meg néhány tömbprocesszoros számítógépet.
75. A folyamatok kezelése szempontjából milyen csoportba sorolhatók az üzenetátadásos számítógépek?
76. Vázolja fel az üzenetátadásos számítógépek processzor-memória kapcsolatát tükröző rajzot.
77. Milyen memóriahasználat a jellemző az üzenetátadásos számítógépekre?
78. Fogalmazza meg röviden, hogy miért kapta az elnevezését az üzenetátadásos számítógép? Mi a működésének a lényege?
79. Milyen eszköz a transzputer?
80. Mitől függ az utasítás végrehajtása az adatvezérelt számítógépben?
81. Miben különbözik az adatvezérelt számítógép a Neumann-féle számítógéptől? Rajzolja fel a működési vázlatát!

3. FEJEZET

KÖZPONTI EGYSÉG, PROCESSZOR

3.1.A PROCESSZOR RÉSZEI

A processzorok alapvető működési elveiről, jellemzőiről a 2.1.pontban már néhány dolgot elmondtunk. A processzorok legfontosabb építőelemeit mutatja be a 2-3.ábra alapján a következő, 3-1.ábra. A mikroprocesszorok felépítése hasonló a nagygépek központi egységének felépítéséhez és habár sokféle mikroprocesszor struktúra ismert, mindegyikre jellemző a három fő egység, a *vezérlő egység*, az *aritmetikai egység* és a különböző funkciójú *regiszterek* megléte. Az egyes részegységek feladatai az alábbiakban foglalhatók össze.



3-1.ábra: *A processzor építőelemei*

Vezérlő egység

A processzor **vezérlő egységének**(CU=Control Unit) feladata a program utasításai, vagy külső kérések(periféria megszakítási kérelme, sín igénybevételi kérése) alapján, vezérlő jelek segítségével a gép részeinek irányítása. Ez *egyrészt* az aritmetikai egység műveleteinek irányítását,

valamint az egyes adatútvonalak nyitását/zárását(kapuzását), *másrészt* a külső egységek, a memória, az I/O eszközök vezérlését jelenti.

Az utasítások végrehajtása, ezen keresztül a gép vezérlése, legtöbbször mikroprogram alapján történik; azaz minden egyes utasítás műveleti kódja egy kis kapacitású ROM tárban, a mikroprogramtárban elhelyezkedő programot indít el, amelynek minden lépése a vezérlő jeleknek egy sorát eredményezi.

Aritmetikai-logikai műveletvégző egység

A processzor másik lényegi egysége az **aritmetikai egység**(ALU=Arithmetic Logic Unit), amely az utasításokban előírt aritmetikai, vagy logikai műveleteket hajtja végre. A műveletvégzéshez néhány regisztert is magában foglalhat az aritmetikai egység.

Az aritmetikai egység bináris műveletek elvégzésére alkalmas és a kettes számrendszer alapján történő műveletvégzés mellett, többnyire a decimális aritmetika szerint is képes műveleteket végrehajtani. Amennyiben az elvégzendő aritmetikai műveletek száma nagy, vagy ú.n. lebegőpontos(hatványkitevős) formátumú számokkal kell műveleteket végezni, akkor a főprocesszor mellé elhelyezhetünk egy erre a célra szolgáló matematikai társprocesszort(koprocesszort) is. A nagyobb teljesítményű(pl.: i486-os, MC68040-es, különböző RISC) processzorok, többnyire már a lebegőpontos koprocesszort is magukban foglalják.

Regiszterek

A processzorok ideiglenes adattárolási céljaira szolgálnak az általános célú, vagy meghatározott funkciójú **regiszterek**. A regiszterek a belső sínrendszeren keresztül tartanak kapcsolatot a processzor más részeivel. A regiszterek vagy szorosan kapcsolódnak a vezérlő és az aritmetikai egységhez, vagy egy több regiszterből álló tömb egyik egységét képezik. A regiszterek gyors működésű tárak, amelyek hossza általában az adatsín szélességével egyezik meg. A legfontosabb, legtöbb processzornál meglévő regiszterek a következők:

Utasításszámláló regiszter(PC=Program Counter, vagy IP=Instruction Pointer), amely a soronkövetkező utasítás memóriabeli címét tartalmazza mindig. Az utasításszámláló regiszter a kezdő értékét, azaz a program első utasításának tárbeli helyét, kívülről(pl. az operációs rendszeren keresztül) kapja, a program indítása előtt.

Amennyiben a program utasításai szigorú egymásutánban követik egymást a tárolóban, akkor a PC tartalma mindig a következő utasítás tárbeli címét adja meg; ha a programban elágazás következik be, akkor a vezérlő egység új értékkel, az elágazási utasításban lévő címmel tölti fel a PC-t.

Utasításregiszter(IR=Instruction Register) a vezérlő egység fontos része, amely a tárból kikeresett, 'lehívott' utasítást fogadja be arra az

időre, amíg a vezérlő egység az utasítás műveleti jelrésze(opcode) alapján meghatározza az elvégzendő műveletet és ennek alapján elindítja a végrehajtást vezérlő mikroprogramot. Az utasítás címrésze(az operandus(ok) tárbeli helyét adja meg) alapján ugyanakkor az operandus pontos tárbeli címe is kidolgozásra kerül.

A korszerű processzorok utasításfeldolgozási módszere(pipeline technika) miatt, a mai processzorokban az utasításregiszter ebben a formájában már nem létezik.

Bázis(cím)regiszter(base register) az operandusok címzéséhez felhasznált regiszter, amely nem általános használatú, azaz nem minden processzornál található meg, vagy más néven használatos. A báziscím egy alapcím, amelyhez viszonyítva adhatjuk meg az utasításban az operandus helyét.

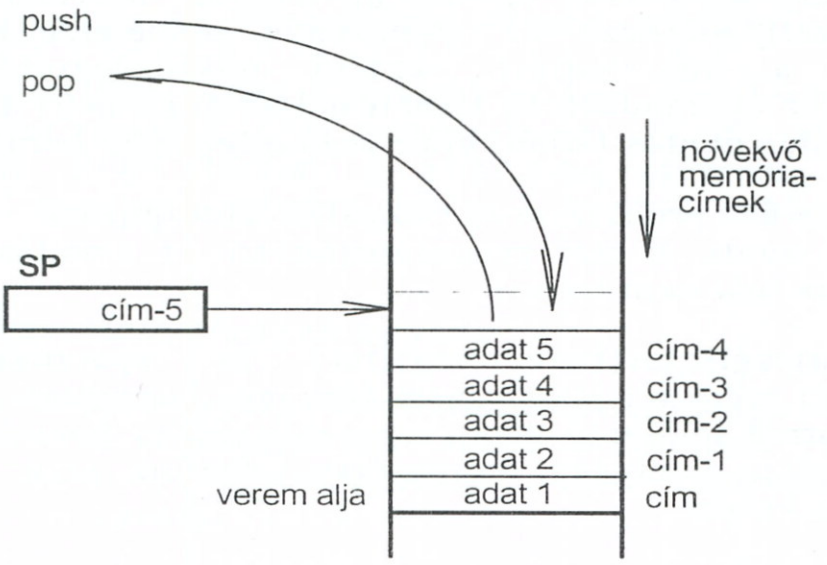
Hasonló feladatot látnak el - a több processzornál is alkalmazott - szegmensregiszterek, amelyek a felhasznált tárolóterület kezdetének címét, helyét tárolják.

Indexregiszter(ek)(index register) szintén az operandusok címzését segítik elő, különösen adatsorozatok feldolgozásánál használhatók előnyösen. Ugyancsak nem minden processzornál van külön erre a célra szolgáló regiszter.

Állapotregiszter(ek), vezérlő regiszter(ek)(status register, flag register, control register), amelyek egy, vagy több regiszteren belül tárolnak vezérlő és ellenőrző jeleket. A műveletek végrehajtásának eredménye alapján bekövetkező állapot jellemzőit tükrözi vissza a regiszter egy-egy helyiértéke, bitje. Ilyen jellemző pl. az eredmény nulla volta(zero flag), a keletkezett számérték túl nagy értéke(overflow flag), átvitel keletkezése(carry flag), stb. Az állapotregiszternek lehetnek olyan bitjei, amelyek valamilyen vezérlési előírást tárolnak, mint pl. valamely részegység használatának engedélyezése, vagy tiltása, memória-lapozás engedélyezése, megszakításkérés kiszolgálásának letiltása, stb. Ha a különböző funkciójú bitek száma nagy, akkor önálló vezérlő regiszter(control register) és állapotjelző regiszter(status, vagy flag register) használata a szokásos.

Veremmutató regiszter(SP=Stack Pointer), amely egy speciális tároló, a veremtároló legfelső elemét jelöli ki. A veremtároló általában a főmemóriában van kialakítva, annak egy lefoglalt területét használja. Adatokat csak a verem tetejére lehet tenni és csak onnét lehet levenni is. Ezt a tárolókezelési módot nevezik LIFO(Last-In-First-Out) módszernek. A veremmutató(stack pointer) mindig arra a tárolóhelyre mutat, annak a tárolóhelynek a címét tartalmazza, ahová a következő adatot elhelyezi(push), vagy ahonnét a következő adatot leveszi(pop) a processzor. A veremmutató használatának vázlatát a 3-2.ábra mutatja be.

A veremmutató *push*, *pop* utasításokkal kapcsolatos automatikus növelésének, vagy csökkentésének módja az egyes processzoroknál eltérő lehet.



3-2.ábra: *Veremtároló használata*

Pufferregiszter(ek)(buffer registers), amelyek a processzor belső adat- és címsínjét választják el a külső sínrendszertől, vagy más ideiglenes célú tárolásra szolgálnak. Nem minden processzornál találhatók meg.

Sínrendszer

A központi egység részét képezi az egyes egységeket(vezérlő egység, aritmetikai egység, regiszterek, memória) összekötő **sínrendszer**(bus system) is. A sínrendszer funkcionálisan, három sínt foglal magában: az adatsínt(data bus), a címsínt(address bus) és a vezérlősínt(control bus). A processzoron belül és kívül egyaránt használt kapcsolatteremtő eszköz. A mikroprocesszorok belső sínrendszere nem tartalmaz elkülöníthető vezérlési sínt, az elosztott formában jelenik meg benne. A külső sínrendszer többnyire két részre van osztva, van a processzorral közvetlen összeköttetésben lévő lokális sínrendszer(local bus) és van az ettől áramkörileg elválasztott rendszersín(system bus). Az elválasztás oka az, hogy a processzor önmaga nem alkalmas arra, hogy a számítógép minden egységét 'meghajtsa'. Erre az elválasztó, sínvezérlő áramkörök szolgálnak.

A sínrendszer részletes tárgyalására az 5.1.pontban térünk vissza.

3.2.ADAT- ÉS UTASÍTÁSTÁROLÁSI FORMÁK

A Neumann-elvű számítógépeknél mind a programokat, mind az adatokat ugyanabban a tárolóban helyezzük el és a tárolás jellege mindkét esetben ugyanaz. Ez azt eredményezi, hogy **a tárolt jelsorozat értelmezésétől függ csak az, hogy a tárolóhely tartalmát utasításnak, vagy adatnak kell-e tekinteni**. Egy utasítás mindig értelmezhető adatként, de ugyanez fordítva nem mindig tehető meg. Ez utóbbinak az a magyarázata, hogy egy adatot képviselő jelsorozatnak nem mindig feleltethető meg érvényes utasítás;

ugyanis lényegesen több jelkombináció képezhető az adott tárolóterületen, mint amennyit utasításként felhasználunk.

Az értelmezési és átírási problémák megszűnnek külön adat- és külön programtároló, az ú.n. Harvard-struktúra használata esetén.

3.2.1.Adatok tárolási formái

Az adatok tárolásánál ma használatos módszerek többé-kevésbé azonosak a különböző gyártók esetében. A két fő célkitűzés, amit a tárolási formák kialakításánál számításba vesznek:

- a tárolás lehetőleg hatékony megoldása, az egyértelmű és könnyű értelmezhetőség fenntartása mellett;
- numerikus adatok esetében, a műveletvégzés legegyszerűbb és leggyorsabb módú megvalósításának elősegítése(pl.: helyiértékes, kettes számrendszer használata; pozitív, negatív számok megfelelő kódolása; decimális műveletvégzéshez a legmegfelelőbb decimális kód használata, stb.).

a.)A számok szokásos írásmódja

A gépi adattárolási formák, a kialakult matematikai módszereknek, a kézi számításoknál megszokott formáknak felelnek meg, a tárolásból eredő szempontok(pl. a véges tárolóhely használatának kényszere) figyelembe vételével. Ezért *először a szokásos számírási módokat tekintjük át, hogy annak alapján egyértelmű legyen a gépi tárolási forma kialakítása.*

A mindennapi gyakorlatban számításainkat 10-es alapú, helyiértékes számrendszerben végezzük. Ez azt jelenti, hogy a számjegyek alaki értékei az elválasztó jeltől(tizedes vesszőtől) jobbra és balra, helyiértékenként más és más értéket képviselnek és ezen alaki értékeknek megfelelő számban kell venni 10 valamely hatványát, a helyiértéktől függően.

A számok szokásos írásmódja az alábbi:

$$A \equiv (\pm a_{-m} a_{-m+1} a_{-m+2} \ldots a_{-1} a_0, a_1 a_2 \ldots a_n)$$

ahol $a_{-m}, a_{-m+1}, \ldots, a_{-1}, a_0, \ldots, a_n$ az egyes helyiértékeken szereplő számjegyek alaki értékei.

Ha a számrendszer, amelyben a fenti jelsorozatot értelmezzük, alapszáma **r**(radix), akkor a leírt jelsorozat a következő számértéket jelenti:

$$A = \pm \sum_{i=-m}^{n} a_i r^{-i} \qquad \text{és ahol} \qquad 0 \le a_i < r \qquad \text{minden i-re.}$$

Például az $A \equiv (7346)$ jelsorozat értelmezése tízes(r=10), illetve nyolcas (r=8) számrendszerekben a következő B számokat jelenti:

$$B_{10} = 7346_{10} = 7 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 6 \cdot 10^0 = 7000 + 300 + 40 + 6 = 7346_{10}$$
$$B_8 = 7346_8 = 7 \cdot 8^3 + 3 \cdot 8^2 + 4 \cdot 8^1 + 6 \cdot 8^0 = 3584 + 192 + 32 + 6 = 3814_{10}$$

A számok fenti írásmódjában a jobb alsó index a számrendszer alapszámára utal.

Fixpontos írásmód

A számok, előzőekben is használt, alábbi írásmódját fixpontos (fixvesszős a magyar írásmódnak megfelelően) írásmódnak nevezzük:

$$\pm a_{-m} a_{-m+1} a_{-m+2} \ldots a_{-1} a_0 , a_1 a_2 \ldots a_n$$

Az elválasztójeltől(a számítógépes gyakorlatban az angolszász gyakorlat alapján a **pont**ot használják) balra találhatók az **egészrész** számjegyei, attól jobbra a **törtrész** számjegyei.

A leírt jelsorozat értelmezése, számértéke:

$$A = \pm \sum_{i=-m}^{n} a_i r^{-i} \qquad \text{és ahol} \qquad 0 \le a_i < r \qquad \text{minden i-re.}$$

A fixpontos írásmódot a kisebb, kevesebb számjegyből álló számok írásakor alkalmazzuk.

Lebegőpontos írásmód

A számok lebegőpontos, vagy hatványkitevős írásmódját a matematikai, tudományos és műszaki gyakorlatban használják, elsősorban a nagy és a kis számok leírására. Formáját tekintve az alábbi módon szokásos írni:

$$\pm a \cdot r^{\pm p}$$

ahol
- a a **mantissza**, amely önmagában egy tetszőleges fixpontos írásmódú szám,
- p a **karakterisztika**, a hatványkitevő, amely önmagában egy fixpontos egész szám,
- r a számrendszer **alapszám**a, radixa.

Az így leírt jelsorozat értelmezése, számértékének kiszámítása a következő:

$$A=\left(\pm\sum_{i=-m}^{n}a_i r^{-i}\right)\cdot r^{\pm p} \qquad \text{és ahol} \qquad 0\le a_i<r \qquad \text{minden i-re.}$$

Ilyen lebegőpontos leírású szám például: $-654.187\cdot 10^{7}$

Normalizált írásmód

A számítások során gyakran használunk különféle, egységes formátumú írásmódokat. Ezek közül hangsúlyozott fontosságú az ú.n. **normalizált alak**. A számítógépes gyakorlatban, a gépi tárolási módok miatt, a nullára, illetve kisebb mértékben az egyes(ek)re normalizált forma kap jelentőséget.

A normalizálás azt jelenti, hogy a számokat olyan hatványkitevős(tehát lebegőpontos) alakra hozzuk, amelyben az egészrész 0 értékű, vagy csak az egyeseknek megfelelő helyiértéken található értékes(azaz nem nulla) számjegy.

Így például nullára normalizáltak a következő számok:

$$(\mathbf{0},45298\cdot 10^{-5})_{10}$$

$$(\mathbf{0},1101011\cdot 2^{4})_{2}$$

(A kettes számrendszer használatakor, igazából a hatványalapot és a hatványkitevőt is kettes számrendszerben kellett volna felírni. Ezt, az áttekinthetőség miatt nem tettük meg.)

Egyesekre normalizáltak az alábbi számok:

$$(-\mathbf{6},28\cdot 10^{13})_{10}$$

$$(\mathbf{2},763\cdot 8^{4})_{8}$$

$$(\mathbf{1},01101\cdot 2^{-3})_{2}$$

A nullára normalizált, $\pm a\cdot r^{\pm p}$ alakú számoknál, a normalizálásból eredően, a mantisszarészre mindig teljesül az, hogy

$$\frac{1}{r}\le a<1$$

és a szám értelmezése a már megismert módon a következő:

$$A=\left(\pm\sum_{j=1}^{l}a_j r^{-j}\right)\cdot r^{\pm p} \qquad \text{és ahol} \qquad 0\le a_j<r \qquad \text{minden j-re.}$$

A számok mindig átalakíthatók normalizált formába.

Véges pontosság problémája

Kézi számolásnál, az eredmény pontossága és a használt papír nagysága között nem látunk közvetlen kapcsolatot. A papírra gyakorlatilag annyi számjegyet írhatunk le amennyit akarunk, így annak korlátozó hatását nem érezzük. Ha az eredmény pontosságához 4-5 értékes jegyre van szükségünk, akkor a részeredményeket 6-7 jegy pontossággal számolva, nem találjuk szemben magunkat a papír nem kellő szélességének problémájával.

Más a helyzet azonban a számítógépnél, amelynek tárolóhelyei véges hosszúságúak. Programozástechnikai megoldásokkal lehet növelni a felhasznált tárolóhely méretét, de ez az alapproblémán nem változtat.

Ha a rendelkezésre álló tárolóhelyen pl. 3 decimális számjegyet tudunk tárolni(**fixpontos) előjel nélküli egész számok** formájában, akkor a tárolható számok tartománya **000-999** között mozog. Az elvégezhető műveletek (összeadás, kivonás, szorzás, osztás) nem mindig vezetnek olyan eredményre, amelyet a rendelkezésre álló tárolóhelyen el tudunk helyezni. Így az összeadás és a szorzás eredménye ugyan egész szám lesz, de nem biztos, hogy 3-jegyű szám lesz. A kivonás során az eredmény lehet negatív is, amit nem tudunk megjeleníteni ebben a megoldásban. Hasonlóan az osztás eredménye sem mindig egészszám és így nem tudjuk tárolni sem. Erre mutat példát az alábbi néhány eset:

700+450=1150	ami túl nagy lesz(nem 3-jegyű),
400-500=-100	a negatív számokat nem tudjuk megjeleníteni,
064*064=4096	ami ismét túl nagy szám,
035/014=2.5	ami nem egészszám, ezért ugyancsak nem tudjuk tárolni.

A műveletek elvégzése során az *egyik hibajelenség* az, hogy a keletkezett eredmény túl nagy(overflow=túlcsordulás), vagy túl kicsi(underflow=alulcsordulás) ahhoz, hogy a rendelkezésre álló tárolóban megjeleníthessük; a *másik hibajelenség* pedig az, hogy az eredmény nem tartozik az operandusok halmazába(pl. egészszámokkal végzett művelet során nem egészszámot kapunk eredményül).

A véges pontosságból fakadó előző hibák, az előírt műveletsor elvégzési sorrendjét is megszabhatják, elkerülendő a részeredmények helytelen nagyságát. Pl. az alábbi zárójeles kifejezések más és más eredményre vezetnek, pedig matematikailag mindegyik meghatározható értékű.

500+(400-600)=500+(-200)	a második tagbeli negatív szám(400-600) miatt, nem kapunk eredményt,
(500+400)-600=900-600=300	ebben a sorrendben elvégezhető a műveletsor,

700+(400-800)=(700+400)-800 egyik formában sem kiszámítható, holott az eredmény(300) az tárolható lenne, de a részeredmények nem tárolhatók.

Hasonló pontossági kérdések a **lebegőpontos számoknál** is felmerülnek. A műszaki, tudományos számításoknál igen gyakran kell egyidőben igen nagy és igen kis számértékekkel dolgozni. Ezt a fixpontos irásmód mellett nem tudjuk jól megoldani; ebben segít a lebegőpontos írásmód, amely lehetővé teszi azt, hogy mindig ugyanannyi értékes számjeggyel dolgozhassunk.

Az ábrázolható számok esetében meg kell különböztetnünk egymástól a **tárolt számok pontosságát** és a **tárolható számok tartományát**. A számok pontosságát a mantisszában szereplő számjegyek száma, a tárolható számok tartományát a karakterisztikában lévő számjegyek száma határozza meg.

Ha a nullára normalizált lebegőpontos számok elhelyezésére az előző példában szereplő tárolási lehetőséggel rendelkezünk, kiegészítve a karakterisztikával is, azaz a mantisszát 3 decimális számjeggyel, a karakterisztikát 2 decimális számjeggyel írhatjuk le, továbbá kiegészítve az előjel jelzésével, akkor a tárolható számok tartománya, amely 199 nagyságrendnyi, az alábbiak szerint alakul:

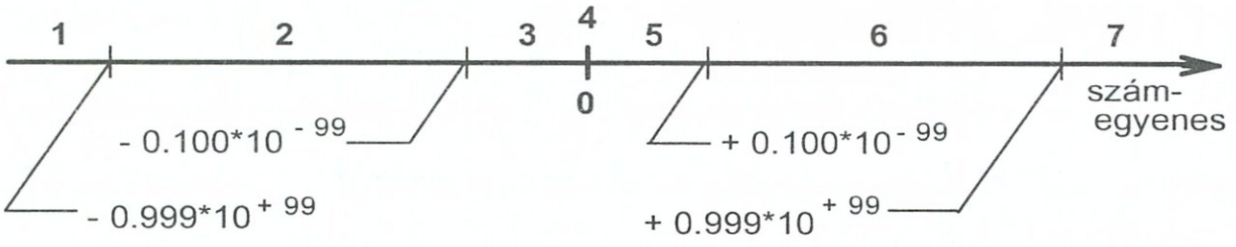
$-0.\mathbf{999}*10^{+\mathbf{99}} \leq A \leq -0.\mathbf{100}*10^{-\mathbf{99}}$ a negatív számok körében, illetve

$+0.\mathbf{100}*10^{-\mathbf{99}} \leq A \leq +0.\mathbf{999}*10^{+\mathbf{99}}$ a pozitív számok körében, valamint az

$A=+0.\mathbf{000}*10^{+\mathbf{00}}$ nulla számérték.

A tárolandó számok tartományát a számegyenesen elhelyezve, amelyet a 3-3.ábra mutat be, a következő hét tartományt különböztethetjük meg:

1. a $-0.\mathbf{999}*10^{+\mathbf{99}}$ -nél kisebb, igen nagy abszolút értékű negatív számok tartománya;
2. a $-0.\mathbf{999}*10^{+\mathbf{99}}$ és $-0.\mathbf{100}*10^{-\mathbf{99}}$ közötti, ábrázolható negatív számok tartománya;
3. a $-0.\mathbf{100}*10^{-\mathbf{99}}$ -nél nagyobb, igen kis abszolút értékű negatív számok tartománya;
4. a nulla;
5. a $+0.\mathbf{100}*10^{-\mathbf{99}}$ -nél kisebb, igen kis pozitív számok tartománya;
6. a $+0.\mathbf{100}*10^{-\mathbf{99}}$ és $+0.\mathbf{999}*10^{+\mathbf{99}}$ közötti, ábrázolható pozitív számok tartománya;
7. a $+0.\mathbf{999}*10^{+\mathbf{99}}$ -nél nagyobb, igen nagy pozitív számok tartománya.



3-3.ábra: *Lebegőpontos számábrázolás tartományai*

A véges tárolási hossz, valamint a normalizált tárolási előírás miatt, az 1., 3., 5. és 7.tartományba eső számértékeket nem tudjuk ábrázolni, tárolni. Ilyen eredményt adó műveletvégzés esetében ez vagy **túlcsordulási**(overflow) **hibát**(az 1. és a 7.tartomány esetében), vagy **alulcsordulási**(underflow) **hibát**(a 3. és 5.tartomány esetében) eredményez. Az alulcsordulás kisebb problémát jelent, mert a 3. és 5.tartományba eső igen kis abszolút értékű számok általában minden gond nélkül helyettesíthetők a 0 számértékkel.

További probléma, hogy a véges tárolási hosszúságú lebegőpontos számok nem egyenletesen adják vissza a valós számokat. Tároláskor az ábrázolható tartományok véges sok diszkrét értéke ábrázolható csak, míg a valós számok folytonosan, végtelen számban helyezkednek el az adott tartományban. Tehát az ábrázolható számok, mint diszkrét pontok helyezkednek el a számegyenesen. Ebből következik, hogy lehetséges olyan kiszámított érték, amely az ábrázolható számok 2., vagy 6.tartományába esik, mégsem tárolható, mert a kiszámított pontos érték két tárolható érték közé esik. Ilyenkor a legközelebbi tárolható számmal helyettesítjük számot. Ez a lépés a **kerekítés**.

A tárolható számértékek sűrűsége változó a számegyenes mentén, a kisebb abszolút értékű számok körében nagyobb, mint a nagyobb számok körében.

A lebegőpontos szám mantisszájának növelése, a helyiértékek számának növelése a tárolható számok sűrűségét növeli, azaz a számábrázolás pontossága nő. A karakterisztika hosszának növelése az ábrázolható számok tartományának alsó, felső határát csökkenti, illetve növeli.

b.)Numerikus adatok tárolása

A numerikus adatok tárolt formája elsősorban a műveletvégzés elősegítésére szolgál. Ezért a tároláshoz olyan bináris kódrendszereket használnak, amelyek lehetővé teszik az aritmetikai műveletek egyszerű és gyors végrehajtását. A számítógépekben elsősorban a kettes számrendszert használják, illetve a látszólag tízes számrendszerben dolgozó gépeknél(pl.: zsebszámológépek) különböző bináris kódrendszereket(BCD-, Gray-, Stibitz-, Aiken-kód) alkalmaznak.

Kettes számrendszer szerinti tárolás

Az előző alpontban említettük, hogy a tárolt adatok véges hosszúságú tárolóhelyen vannak elhelyezve és ennek a tárolóhelynek a hosszától, valamint az alkalmazott tárolási formától(fixpontos, vagy lebegőpontos) függ a számadatok lehetséges pontossága. Először az előjel ábrázolásától eltekintünk, majd bemutatjuk azokat a kódolási formákat, amelyek segítségével a negatív számok is tárolhatók a számítógépeken belül.

1.Fixpontos, lebegőpontos tárolási forma

Fixpontos számtárolási formánál, a szám kettes számrendszerbeli együtthatóit kell elhelyezni a rendelkezésre álló véges(2-4 byte) hosszúságú rekeszbe. Elvileg a kettedes pont helyét tetszőlegesen választhatnánk meg, de célszerűségi okok miatt(túlcsordulás, alulcsordulás elkerülése végett) minden gépnél rögzített ennek helye, mégpedig vagy a legértékesebb(legnagyobb helyiértékű) bit előtt, vagy a legalacsonyabb helyiértékű hely után. Műveletvégzéskor, *a gép a tárolt jelsorozatot az első esetben, mint törtszámot, a második esetben mint egészszámot értelmezi, kezeli.* Ezt mutatja be a 3-4.ábra.

Látható, hogy a fixpontos tárolásnál a kezelhető számtartomány erősen korlátozott és a tárolóhely hosszától függ. Az ábrázolható számtartomány, attól függően, hogy fixpontos egész-, vagy törtszámról van szó, az alábbi:

fixpontos egésznél: $0 \le A < 2^n$

fixpontos törtnél: $2^{-n} \le A < 1$

ahol n a tárolócellák(bitek) száma.

Ugyanakkor, a programozónak a feladat megoldása közben (mivel ritka eset, hogy csak kizárólag egészszámokkal, vagy kizárólag csak törtszámokkal kell dolgozni) állandóan figyelemmel kell kísérnie azt, hogy a kettedes pont valójában hová kerülne, azaz a hatványkitevő aktuális értékét külön tárolnia kell. Ennek kiküszöbölésére, valamint az ábrázolható számtartomány kibővítésére használjuk a lebegőpontos formát a számítógépen belül is.

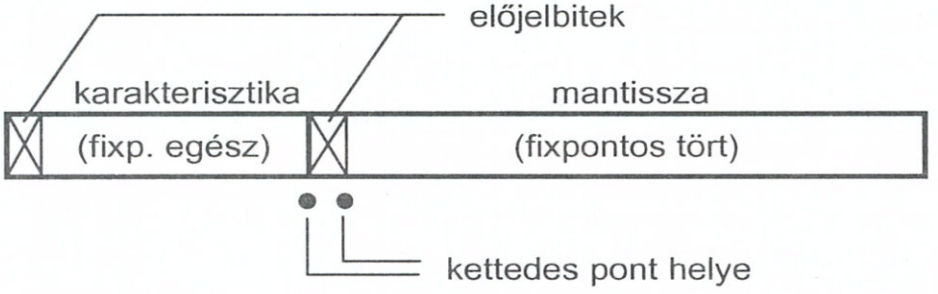
3-4.ábra: *Fixpontos egész- és törtszám tárolási formája*

A **lebegőpontos számtárolási formánál** a 3.2.1.a.pontban már megismert írásmódhoz hasonlóan, a számok tárolása a hatványkitevős formának megfelelően, többnyire nullára normalizált alakban történik.

Mivel a számrendszer alapszáma rögzített(ez általában r=2), ezt tárolni nem kell, elegendő csak

- a mantissza és
- a karakterisztika tárolása.

Tehát a lebegőpontos adattárolási forma mindig egy számpár, a mantissza és a karakterisztika együttes tárolását és kezelését jelenti; mégpedig önmagukban fixpontosan, azaz a mantisszát fixpontos törtszámként, a karakterisztikát fixpontos egészszámként kezeli a gép. Ezt mutatja be a 3-5.ábra rajza.



3-5.ábra: *Lebegőpontos számok tárolási formája*

A számok elegendő pontosságú tárolása miatt az együttes felhasznált tárolóterület 4-6-8 byte. Ennek egy részét értelmezi karakterisztikaként a számítógép.

A tárolható számok tartománya, ha eltekintünk egyelőre az előjelektől:

$$\frac{1}{2} \le A < 1 \cdot 2^{2^m - 1}$$

ahol m a karakterisztika tárolására szolgáló bitek száma.

2.Pozitív és negatív számok megkülönböztetése

Az előző részben eltekintettünk a számok előjeleinek feltüntetésétől a könnyebb érthetőség miatt. Azonban nyilvánvaló, hogy a negatív számokat is jelölni kell valamilyen módon. Erre a célra - az előjelezésre - a tárolt jelsorozat legértékesebb, legmagasabb helyiértékű bitjét használjuk fel(3-6.ábra) és ennek értéke:

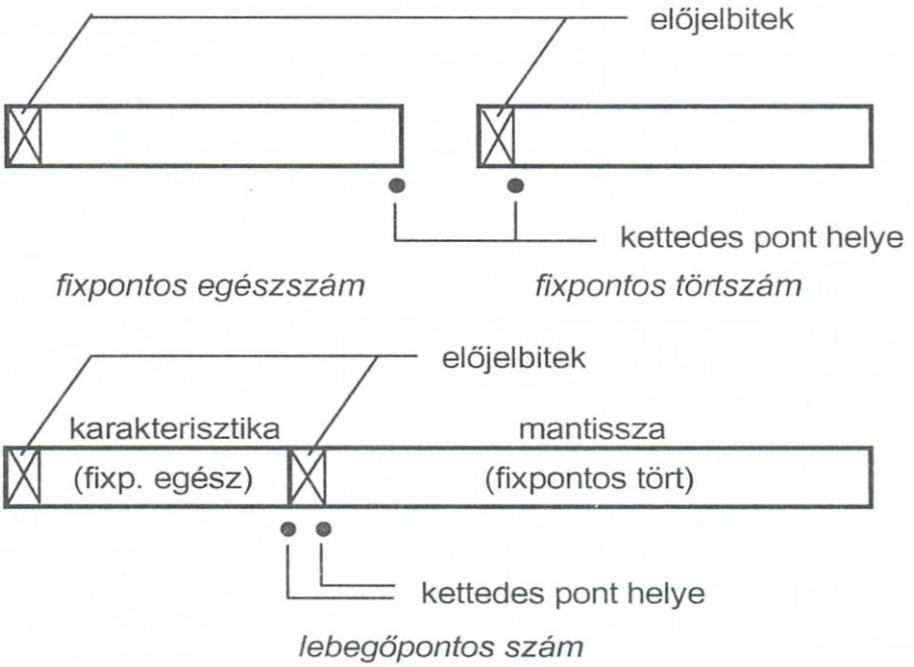
0 a pozitív számok és

1 a negatív számok esetében.

A negatív számok tárolásához négyféle módszert használnak a számítógépeknél, mégpedig:

- az előjeles, abszolútértékes,
- az 1-es komplemens kódú,
- a 2-tes komplemens kódú és
- a 2^{m-1} többletes kódolású tárolási módszert, ahol m a karakterisztika tárolási bitjeinek száma.

Leggyakrabban a kettes komplemensű, vagy a 2^{m-1} többletes tárolási módszert alkalmazzák, kiegészítve az előjeles, abszolútértékes tárolási móddal.



3-6.ábra: *Előjelbitek helye*

Az adatok tárolásánál lényeges a tárolóhely hossza, mert ez meghatározza egyrészt az ábrázolható legnagyobb és legkisebb számot, másrészt azt a számot, amelyre komplemensképzéskor ki kell egészíteni a tárolandó számértéket.

- ***Előjeles, abszolútértékes tárolásmód*** esetében, az előjel biten a szokásos módon jelöljük a szám pozitív vagy negatív voltát, a többi helyiértéken pedig a szám abszolútértékét tároljuk.(n bites tárolóhely esetén n-1 biten. - 3-7.ábra)
- ***Egyes komplemenskódú tárolás*** esetén, az n-1 számbiten az előjeltelen tárolandó számértéket az adott tárolóhosszon ábrázolható legnagyobb számra, a $2^{n-1}-1$ -re kiegészítő szám jelsorozatát tároljuk.(3-7.ábra)

  Mechanikus komplemensképzésnél ez azt jelenti, hogy a tárolandó adat minden helyiértékén a 0-t 1-esre, az 1-est 0-ra változtatjuk. Az előjelbit értéke 1-es lesz.

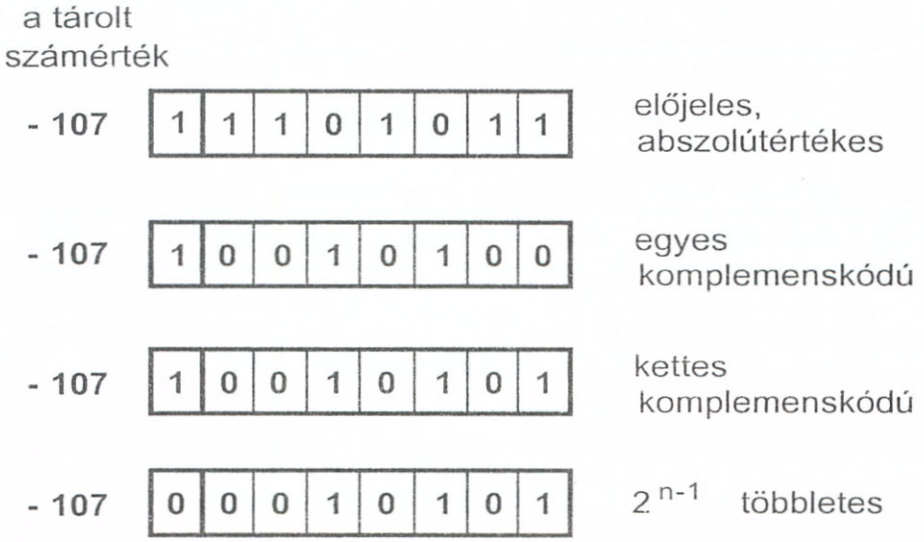
- ***Kettes komplemenskódú tárolás*** esetén, az előjeltelen számadatot az adott tárolóhosszon ábrázolható legnagyobb számnál 1-gyel nagyobb számra, azaz a 2^{n-1}-re kiegészítő számot tároljuk.(3-7.ábra)

  Előállítása mechanikusan úgy történik, hogy jobbról-balra haladva az első 1-esig bezárólag változatlanul leírjuk a számjegyeket, majd a továbbiaknál az 1-est 0-ra, a 0-t 1-esre változtatjuk. Az előjelbit értéke 1-es lesz.

- ***2^{n-1} többletes kódú tárolás*** választása esetén, a tárolandó számértékhez az adott tárolóhosszon ábrázolható számnál eggyel nagyobb számnak (2^{n-1}-nek) megfelelő értéket adunk hozzá, annak érdekében, hogy mindig pozitív számot tárolhassunk.(3-7.ábra)

  Ezt az eljárást általában a lebegőpontos számok karakterisztikájának tárolásánál használják.

  Formailag az előállítása azonos a kettes komplemensű számértékkel, avval a különbséggel, hogy az előjelbit értéke mindig 0 értékű.



3-7.ábra: *Negatív számok tárolási formái*

A különböző pozitív és negatív számtárolási módoknál problémát okozhat, hogy kétféle, pozitív és negatív 0 érték lehet a tárolható számok között. Ez a gond az előjeles, abszolútértékes és az egyes komplemenskódú tárolásnál merül fel, azaz egyszerre létezhet pozitív és negatív nulla is. Egyes változatoknál ezt az értéket pl. az igen nagy negatív szám jelölésére használják.

3.ANSI/IEEE 754 -es lebegőpontos számábrázolási szabvány

A gépgyártás korai időszakában többféle lebegőpontos formátum is használatos volt, némelyike még hibás eredményeket is szolgáltatott. A nyolcvanas években az amerikai villamosmérnökök szervezete(IEEE=Institute of Electrical and Electronics Engineers) kidolgozta azt a szabványt, amelyet azóta a nagy gyártók processzorai(Intel, Motorola, SPARC, MIPS) is használnak. Ez

a szabvány nem csak az adatformátumokat definiálja, hanem általában a lebegőpontos műveletek, átalakítások körét is.

Az egyesekre normalizált számok értelmezési formája:

$$A = (-1)^s \cdot (1.a) \cdot 2^{\pm p+e}$$

alakú, ahol

s a mantissza előjele; 0, ha pozitív és 1,ha negatív,

a az egyesekre normalizált mantissza törtrésze,

p a karakterisztika(hatványkitevő) eredeti értéke,

e az eltolás(a többlet) értéke, amelynek nagysága $2^{m-1}-1$, ahol m=8, 11, 15.

Megemlítjük, hogy a szokásos szóhasználatban **mantissza** alatt, a hatványkitevős számábrázolási forma kitevőrészt szorzó tényezőjét(jelen esetben (**1.a**) értékét) értjük, holott a matematikában ez a fogalom a számok logaritmusának felírásakor, a törtrészt jelenti. Ennek a különbségnek az egyértelmű jelzésére, az IEEE 754-es szabvány a mantissza fogalom helyett a **szignifikandus**(significand) elnevezést használja. Az említettek ellenére, a mantissza fogalmat a továbbiakban is az említett 'szokásos' értelmezés szerint használjuk.

A szabvány több lebegőpontos formát definiál, amelyek közül a processzorok számára az egyszeres pontosságú adatformátum használata a minimum követelmény:

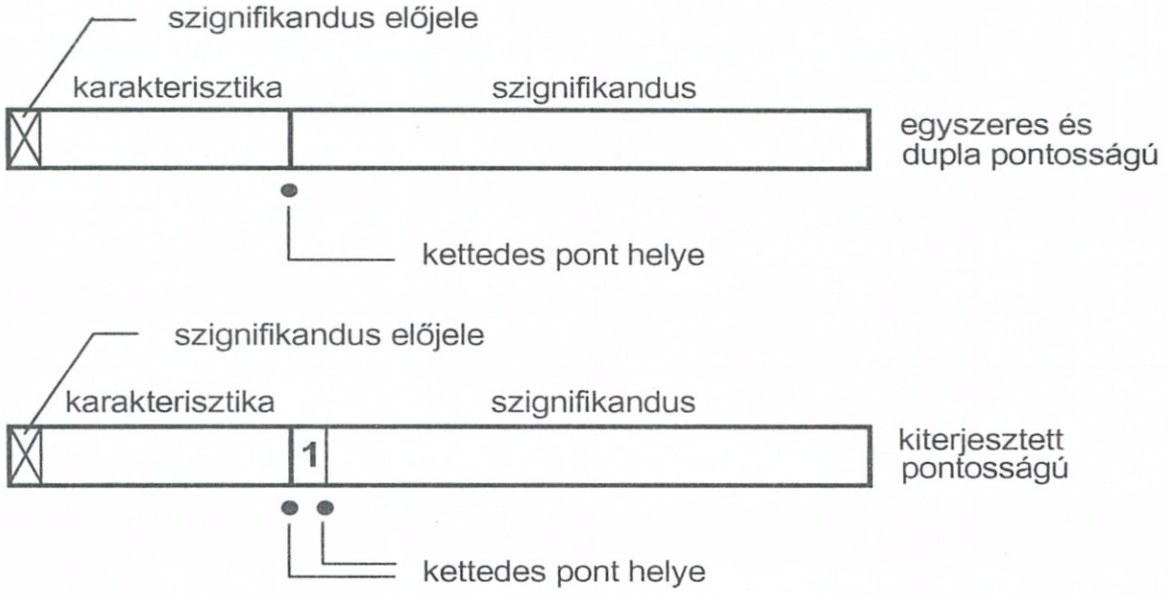
- egyszeres pontosságú(single precision), amely 32 bit hosszúságú,
- dupla pontosságú(double precision), amely 64 bit hosszúságú,
- kiterjesztett pontosságú(extended precision), amely 80 bit hosszúságú,
- négyszeres pontosságú(quadrupled precision), amely 128 bit hosszúságú.

A kiterjesztett pontosságú formátum a 16 bites sínszélességgel rendelkező lebegőpontos processzorok számára előnyös, mivel így öt részre(5x2 byte) bontva minimális tárciklus alatt átvihető a teljes adat. A 32 bites adatsínnel rendelkező processzorok esetében(így a korszerű CISC és RISC processzorok esetében is) azonban ez nem megfelelő, ezért ezeknél az eszközöknél a négyszeres pontosságú adatformátum használatos helyette.

A tárolt adatformák mindegyike azonos szerkezetű és három részből áll (3-8.ábra):

- szignifikandus **előjelbit**je(1 bit), amely 0 értékű a pozitív számoknál és 1 értékű a negatív számoknál;
- **karakterisztika**(8, 11, 15 bit), amelynél a számábrázolási mód a $2^{m-1}-1$ -es többletű(m=8, 11, 15) forma; a karakterisztika legkisebb(0) és legnagyobb(255, 2047, 32767) értékeit speciális célokra használják fel, mégpedig a 0 értékű karakterisztikát a nulla és a denormalizált számok, a maximális értékű karakterisztikát pedig a végtelen és az ú.n. 'nem-szám' adatformátumok ábrázolásához alkalmazzák;

- **szignifikandus**(23, 52, 64, 112 bit), amely fixpontos egyesekre normalizált törtszám, azaz az egészrésze mindig 1 értékű, a törtrésze pedig tetszőleges értékű, így az M mantissza $1 \leq M < 2$ közé eső érték; mivel az egészrész mindig 1-es értékű, ezért ezt megállapodás szerint nem tárolják, csak a törtrészt(kivéve a kiterjesztett pontosságú tárolásmódot); megkülönböztetésül a matematikában használt mantissza fogalomtól, a szabvány a számnak ezt a részét szignifikandusnak(**significand**) nevezi.



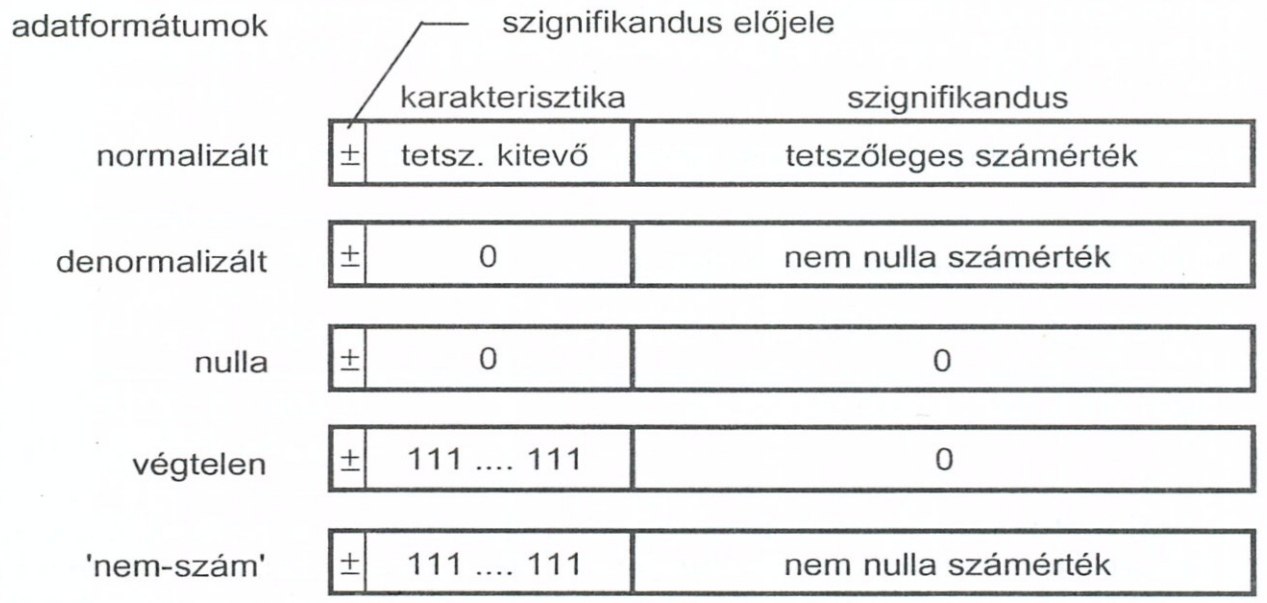
3-8.ábra: *IEEE 754 szerinti lebegőpontos számtárolási formák*

A lebegőpontos számokkal végzett műveletek problémája a **túlcsordulás és az alulcsordulás** megfelelő **kezelése**. Ezeknek az eseteknek a megoldására a szabvány a normalizált adatformátum mellett, további adatformátumokat (3-9.ábra) határoz meg, mégpedig:

- normalizált adatformátum,
- denormalizált(denormalized) adatformátum,
- nulla számérték,
- végtelen értékének adatformátuma,
- nem meghatározott számérték, 'nem-szám'(NaN=Not a Number) adatformátum; a 'nem-szám' adatformátumon belül a szabvány különbséget tesz a:

  – 'jelző' nem-szám(signaling NaN) és a
  – 'csendes', egyszerű nem-szám(quiet NaN) között.

  A 'jelző nem-szám', ha lebegőpontos műveletek operandusaként szerepel, akkor kivételt okoz, míg az 'egyszerű nem-szám' nem eredményez kivételt a művelet végrehajtásakor. Ez utóbbi esetben az eredmény szintén NaN adatformátumú lesz.



3-9.ábra: *IEEE 754 szerinti adatformátumok*

Az egyes adatformátumok esetében ábrázolható, tárolható számok tartományát foglalja össze a következő táblázat.

3-1.táblázat: *Lebegőpontos számok tárolható tartománya*

megnevezés	single	double	quad
s előjel mérete [bit]	1	1	1
k kitevő(**e** többlettel növelt érték) mérete [bit]	8	11	15
f mantisszarész(törtrész) mérete [bit]	23	52	112
e kitevőtöbblet értéke	127	1023	16383

megnevezés	előjel	kitevőrész	ábrázolható számtartomány; mantisszarész
normalizált érték	s	0<k<2e+1	$(-1)^s \times 2^{k-e} \times 1.f$
denormalizált érték	s	k=0	$(-1)^s \times 2^{k-(e-1)} \times 0.f$
nulla	s	k=0	$(-1)^s \times 0$
negatív végtelen(-∞)	1	k=2e+1(max)	f=.0000...00
pozitív végtelen(+∞)	0	k=2e+1(max)	f=.0000...00
jelző 'nem szám'(sNaN)	x	k=2e+1(max)	f=.0xxx...xx
egyszerű 'nem szám'(qNaN)	x	k=2e+1(max)	f=.1xxx...xx

A táblázatban a nem meghatározott értékeket az 'x' betű jelzi.

Az aritmetikai műveletek során keletkezett igen kis számértékek, amelyek **alulcsordulást** eredményeznek, kezelésére kétféle megoldási lehetőség kínálkozik: (1) a számértéket nullával helyettesíteni és a számításokat továbbfolytatni, vagy (2) alulcsordulási hibakezelő eljárás elindítása és a számítások leállítása. Ezek egyike sem a legmegfelelőbb, ezért a szabvány bevezeti a **denormalizált adatformátum** használatát.

A denormalizált adatformátum esetében a karakterisztika értéke nulla, és a mantissza egy tetszőleges, 0 egészrésszel rendelkező törtszám(ez nem nullára normalizált alak!).

Egyszeres pontosságú(single precision) formátumot vizsgálva ez azt jelenti, hogy

- a legkisebb normalizált számérték: $1.0*2^{-126}$
- a legnagyobb denormalizált számérték: $0.999...9*2^{-127}$
- a legkisebb denormalizált érték: $0.000...1*2^{-127} = 2^{-23}*2^{-127}=2^{-150}$

Tehát a legkisebb normalizált formátumú számérték és a nulla közé eső tartományban is ábrázolhatók az eredmények, de egyre csökkenő pontossággal, mivel a mantisszarész baloldalán(a magasabb helyiértékeken) növekszik a 0 számjegyek száma és evvel az értékes számjegyek száma csökken. Így a legkisebb számérték esetében csak a mantissza legutolsó bitje 1-es értékű, azaz a mantissza által leírt számérték 2^{-23} nagyságú és ezt szorozva a karakterisztika értékével adódik a legkisebb kezelhető számértéknek: $\mathbf{2^{-150}}$.

A 3-9.ábrán látható, hogy a szabvány két (±) **nulla értéke**t enged meg, amelyek mindegyikénél mind a mantissza, mind a karakterisztika 0 értékű.

A **túlcsordulások**at eredményező igen nagy számokat a szabvány minden esetben **végtelen**ként kezeli és az evvel az adatformátummal végzett műveleteknél a végtelenre alkalmazott szabályokat veszi érvényesnek. Így

$$\infty \pm a = \infty,$$

$$a/\infty = 0,$$

$$a/0 = \infty$$

A végtelen ábrázolásához a mantissza 0 értékű, a karakterisztika minden bitje pedig 1-es értékű.

A végtelen osztása végtelennel(∞/∞) nem-meghatározott(undefined) számértéket eredményez, amelyre külön adatformátumot, a **'nem szám'**(NaN=Not a Number) **formátumot** határozta meg a szabvány. Az ehhez tartozó adatformátum a végtelenéhez hasonló, avval a kiegészítéssel, hogy a mantissza tetszőleges értékű lehet.

A három számtárolási forma jellemző értékeit a 3-2.táblázat tartalmazza.

A lebegőpontos műveletek fontos problémája az eredmények tárolásakor jelentkező **kerekítési**(rounding) **igényhez** a helyes eljárás megválasztása. Ha a kiszámított pontos érték elhelyezhető a választott tárolási formában, akkor a pontos érték kerül a tárolóba. Ha a pontos érték nem helyezhető el a választott tárolási formában(például a tárolóhely hossza kisebb, mint a

pontos szám értékes jegyeinek a száma), akkor valamilyen módszer alkalmazásával a pontos értéket helyettesíteni kell egy ábrázolható, de nem pontos számértékkel.

3-2.táblázat: *IEEE 754 adatformátumok jellemző értékei*

Jellemző	Egyszeres pontosságú	Dupla pontosságú	Kiterjesztett és négyszeres pontosságú
előjelbit[bit]	1	1	1
karakterisztika[bit]	8	11	15
mantissza[bit]	23	52	64/112
teljes hossz[bit]	32	64	80/128
kitevő többlet	127	1023	16383
kitevő tartománya	-126 - +127	-1022 - +1023	-16382 - +16383
lekisebb normalizált szám	2^{-126}	2^{-1022}	2^{-16382}
legnagyobb normalizált szám	$\sim 2^{+128}$	$\sim 2^{+1024}$	$\sim 2^{+16384}$
decimális számok tartománya	$\sim 10^{-38}$ - 10^{+38}	$\sim 10^{-308}$ - 10^{+308}	$\sim 10^{-4932}$ - 10^{+4932}
legkisebb denormalizált szám	$\sim 10^{-45}$	$\sim 10^{-324}$	$\sim 10^{-4935}$

A szabvány által elfogadott, általában választható módszerek:

- kerekítés a legközelebbi ábrázolható számértékre; ha a távolság egyforma, akkor a páros értékű számra(a legalacsonyabb helyiérték 0 értékű) történik a kerekítés;
- kerekítés a 0 érték felé;
- kerekítés $+\infty$ felé;
- kerekítés $-\infty$ felé.

Ha a kerekítéskor előállított számérték nem ábrázolható a választott tárolási formában, akkor a processzor ú.n. kivételt(exception) kezdeményez (5.2.pont), amely lehetőséget nyújt a felhasználó számára, hogy egyedi módon adjon megoldást a problémára.

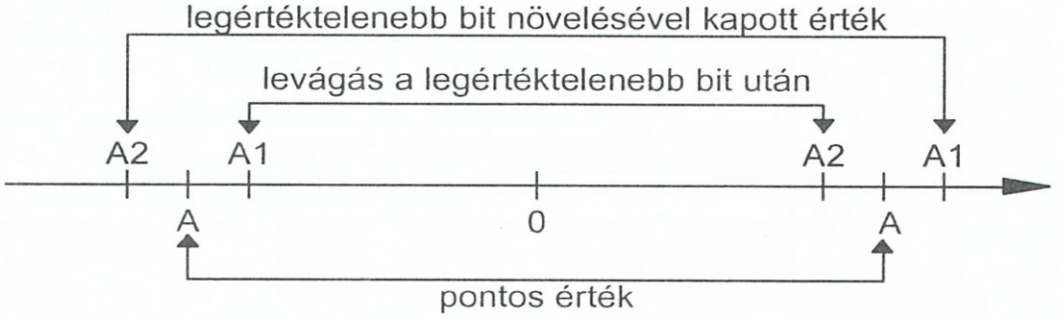
A választandó kivétel típusát a 3-10.ábra alapján határozhatjuk meg. Jelölje a pontos, A számértékből - a kerekítés során - kapott legközelebbi, ábrázolható értékeket A1 és A2 a 3-10.ábrának megfelelően. Az abszolút értékben *nagyobb*, legközelebbi számértéket a legalacsonyabb helyiértékű bit (LSB=least significant bit) növelésével, az abszolút értékben *kisebb*, legközelebbi értéket az LSB utáni rész elhagyásával(truncating) kaphatjuk.

Ha a kerekítendő, pontos A szám maximális érték(nincs nála nagyobb, ábrázolható számérték), akkor

- ha A>0, akkor A1 már nem ábrázolható $\Rightarrow$ túlcsordulás következik be;
- ha A<0, akkor A2 már nem ábrázolható $\Rightarrow$ túlcsordulás következik be.

Ha a kerekítendő, pontos A szám minimális érték(nincs nála kisebb, ábrázolható számérték), akkor

- ha A>0, akkor A2 már nem ábrázolható ⇒ alulcsordulás következik be;
- ha A<0, akkor A1 már nem ábrázolható ⇒ alulcsordulás következik be.



3-10.ábra: *Lebegőpontos számok kerekítési lehetőségei*

Tízes számrendszer szerinti tárolás

A kettes számrendszer szerinti adattárolás elsősorban akkor előnyös, ha aritmetikai műveleteket kívánunk végezni a számokkal, tehát a műszaki-tudományos feladatok, számításigényes problémák megoldáskor elsősorban. Ha a számítási munka egyszerűbb(pl. ügyviteli feladatok esetében), vagy nagytömegű adatot kell beolvasni, vagy kiíratni(így konvertálni tízes számrendszerből, valamely belső kódrendszerre), akkor célszerű a tízes számrendszer használatát utánozni különböző bináris kódokkal.

Ezekben a kódrendszerekben a tízes számrendszerbeli számok számjegyeit egyenként konvertáljuk át az alkalmazott bináris kódba. Mivel a tízes számrendszerben tízféle számjegyet használunk, ezek ábrázolásához legkevesebb 4 bit(1 tetrád) szükséges. A leggyakrabban használt kódrendszerek bináris kódszavai láthatók az alábbi táblázatban:

3-3.táblázat: *Decimális kódrendszerek*

Decimális számjegy	BCD kód	Gray kód	Aiken kód	Stibitz kód	2 az 5-ből kód
0	0000	0000	0000	0011	11000
1	0001	0001	0001	0100	00011
2	0010	0011	0010	0101	00101
3	0011	0010	0011	0110	00110
4	0100	0110	0100	0111	01001
5	0101	0111	1011	1000	01010
6	0110	0101	1100	1001	01100
7	0111	0100	1101	1010	10001
8	1000	1100	1110	1011	10010
9	1001	1101	1111	1100	10100

A számítógépi feldolgozásokban a legáltalánosabban használt és a legkönnyebben előállítható kód a 8-4-2-1 súlyozású BCD(Binary Coded De-

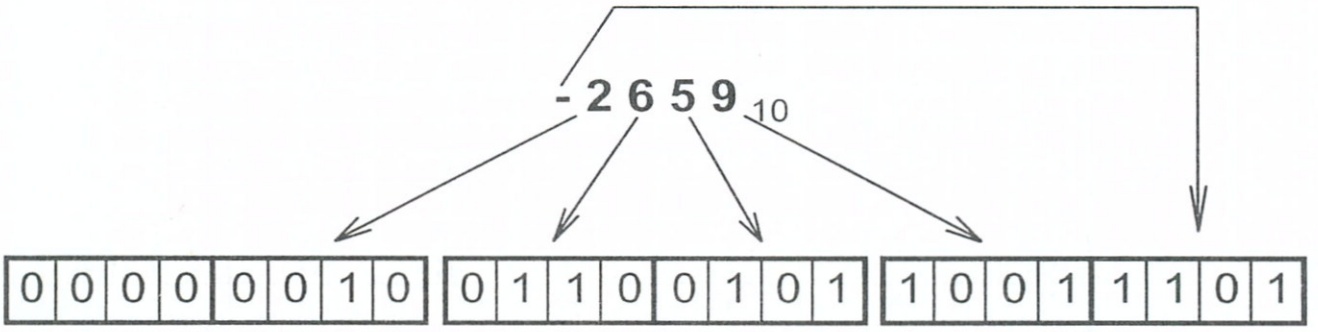
cimal) kód, vagy egyenes kód, ahol a számjegyeknek azok kettes számrendszerbeli értéke felel meg, 4 helyiértékre kiegészítve.

A pozitív és negatív számok megkülönböztetésére, a 4 helyiértéken képezhető 16 jelkombinációból fennmaradó 6 jelsorozat valamelyikét kell kiválasztani.

Az egyik alkalmazott tárolási formánál(EBCDIC kódrendszer tömörített számábrázolási formája, 3-11.ábra) a szám előjelét az utolsó tetrádban őrzik és értéke megállapodás szerint:

1100 a pozitív számok esetében és

1101 a negatív számok esetében.



3-11.ábra: *BCD számábrázolási forma*

Az Intel lebegőpontos processzorok(i8087, i80287, i80387) által alkalmazott BCD adatformátum(amely fixpontos egész típusú) esetében, az előjelet a legmagasabb helyiértéken álló byte legfelső bitje tárolja a szokásos értékekkel; a byte további bitjei 0 értékűek. A tárolt decimális jegyek száma 18.

c.)Alfanumerikus adatok tárolása

Azokban az esetekben, amikor a tárolt adatokkal nem kívánunk aritmetikai műveleteket végezni, csak tárolni és nem aritmetikai jellegű műveleteket (mint pl.: rendezés, válogatás, kigyűjtés, adatcsere, stb.) elvégezni, a numerikus adatok kódolására használt módszerek nem akalmasak az általában nagy tömegű adat kezelésére.

Az alfanumerikus(betű, számjegy, speciális jelek vegyesen) adatok tárolásához néhány kódrendszert használnak csak, amelyek közül a legelterjedtebbek a mini-, mikroszámítógépeknél alkalmazott **ASCII**(American Standard Code for Information Interchange), illetve az elsősorban IBM nagygépeknél használt **EBCDIC**(Extended Binary Coded Decimal Interchange Code) kódrendszerek. Az ASCII kód 7 bites(128 elemű) kódja megegyezik az ISO(International Organization for Standardization) 7 bites kódjával, pontosabban az ASCII kód az **ISO-7** kódnak egyik nemzeti változata. Az ISO rendszerben néhány karakter cserélhető az egyes nemzeti abc-k egyedi karaktereinek megfelelően.

Az ASCII(ISO-7) kódrendszereket általában 8 bites formában használják, kiegészítve a kódot egy, a hibafelfedést elősegítő ellenőrző bittel, a paritás bittel. Az ASCII(ISO-7) 7 bites kódrendszer táblázata látható a következő 3-4.táblázatban. A táblázat sorai a kódszó alsó tetrádját, míg az oszlopai a kódszó felső bitjeit jelölik ki.

A mikroszámítógépek többsége az ASCII kód 8 bites, kibővített(256 elemű, paritásbit nélküli) változatát alkalmazza, amely különböző nemzeti karakterek mellett, néhány grafikus és matematikai karaktert is tartalmaz.

3-4.a.táblázat: *7-bites ASCII kódrendszer*

bin	felső	000	001	010	011	100	101	110	111
alsó	hex	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	(sp)	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O		o	(del)

3-4.b.táblázat: *ASCII kódrendszer vezérlő karaktereinek funkciója*

hex	jel	vezérlő karakterek funkciója	hex	jel	vezérlő karakterek funkciója
00	NUL	Null	10	DLE	Data link escape
01	SOH	Start of heading	11	DC1	Device control 1
02	STX	Start of text	12	DC2	Device control 2
03	ETX	End of text	13	DC3	Device control 3
04	EOT	End of transmission	14	DC4	Device control 4
05	ENQ	Enquiry	15	NAK	Negative acknowledge
06	ACK	Acknowledge	16	SYN	Syncronous idle
07	BEL	Bell	17	ETB	End of transmission block
08	BS	Backspace	18	CAN	Cancel
09	HT	Horizontal tab	19	EM	End of medium
0A	LF	Line feed	1A	SUB	Substitute
0B	VT	Vertical tab	1B	ESC	Escape
0C	FF	Form feed	1C	FS	File separator
0D	CR	Carriage return	1D	GS	Group separator
0E	SO	Shift out	1E	RS	Record separator
0F	SI	Shift in	1F	US	Unit separator

Az alfanumerikus kódrendszerek(EBCDIC, ASCII) használatakor, a számjegyek esetében, többnyire kétféle tárolási lehetőséget biztosítanak a gépek. **Laza** tárolási formánál a számjegyeket a 8 bites kódok sorozatával tárolják

és a szám előjelét a legalacsonyabb helyiértékhez tartozó byte felső 4 bitjében(az ú.n. *zónarész*ben) helyezik el. A **tömörített** tárolási forma megegyezik az előző részben ismertetett BCD kódú számábrázolási formával. Az előjel elhelyezése is azonos az ott említettel, azaz a legalacsonyabb helyiértéket képviselő tetrád utáni 4 bitben található. (Megjegyzendő, hogy egyes esetekben, az előjel jelzésének módja ettől eltérő is lehet.)

d.)Egyéb adattárolási módok

Az eddigiekben ismertetett adattárolási formák közös jellemzője, hogy magából a tárolt jelsorozatból nem lehet következtetni annak tartalmára, azaz arra, hogy milyen típusú adatot(fixpontos, lebegőpontos, vagy más?) képvisel. Ez nem mindig előnyös, ezért alkalmaznak olyan tárolási formákat is(pl. Motorola, SPARC processzoroknál), amelyek az adatot képviselő bitek mellett, további kiegészítő rész(eke)t is tárolnak az adat típusának jelölésére. Ezt a tárolási formát **öndefiniáló adatformá**nak nevezzük. Ezek az összetett tárolási módok a bonyolultabb adatstruktúrák közvetlen, esetleg hardver szintű kezelését teszik lehetővé. Az öndefiniáló adatstruktúrák legjellemzőbb formái:

- **Jelölt adattárolás**(tagged storage), amelynek jellemzője, hogy a tárolt adat kiegészül olyan információkkal, amelyek az adatfelhasználás módját befolyásolják és meghatározzák. Pl. a Borroughs Bxx00 gépeknél alkalmaznak olyan 52 bites adatformát, melynek 3 bitje az adat típusát határozza meg és további 1 bit paritásbit gyanánt szolgál.

  A jelölt adattárolási forma előnye a hibafeltárási lehetőség, a közvetlen hardverszintű és irányítású konverzió lehetősége; hátránya a többletinformációból adódó kiértékelési időtöbblet, a merevebb feldolgozási struktúra, külön költségek.

- **Deszkriptoros tárolási forma**(data descriptor), amely egyszerűbb adatstruktúrák kezelését biztosítja kiegészítő információk(hozzáférési jogok, felhasználási cél, tárolási hely, stb.) hozzáadásával. A deszkriptorban tárolt információk egy része közvetlen hardver kezelésű(pl. az elérési jogok), más része pedig szoftver úton dolgozható fel. Ilyen deszkriptoros tárolási formát használnak az Intel processzorok az adattáblázatok és más objektumok kezeléséhez, tárolásához.

  A deszkriptorok használatának előnye a hibafeltárás és a védelem hardverszintű megoldási lehetősége, ugyanakkor hátránya a többlet tároló- és végrehajtási időigény.

- **Összetett strukturális forma**(object-oriented, capability addressing schemes) alkalmazása elsősorban a szoftverszintű adatkezelést segíti, de hatékonnyá csak akkor válik igazán, ha ehhez megfelelő hardverszintű megoldások is társulnak.

  Egy-egy összetett adatstruktúra objektum formában történő megközelítése kapcsán, az elérhető előnyök:

  – a kapcsolódó információkat egy egységben tárolja,
  – az objektum létrehozása/törlése nem külön-külön adattörléssel valósul meg,

- az objektumhoz kapcsolódó összes adat felhasználása, elérése egyidőben lehetséges,
- konverziója speciális utasítással oldható meg,
- az elemeire vonatkozóan, további öndefiniáló információkat tartalmazhat,
- külön elérési jogokkal rendelkezhet.

3.2.2.Utasítások tárolási formái

A számítógépek tárolóiban, az adatokon kívül, a feladatok műveleteinek végrehajtását vezérlő program utasításait is el kell helyezni. Ezek a tárolt utasítások adatként mindig értelmezhetők, de az adatok nem mindig értelmezhetők utasításként, ahogy azt már a fejezet elején említettük. A következő részekben az utasítások felépítését, típusait ismertetjük, figyelemmel a RISC processzorok jellemzőire is.

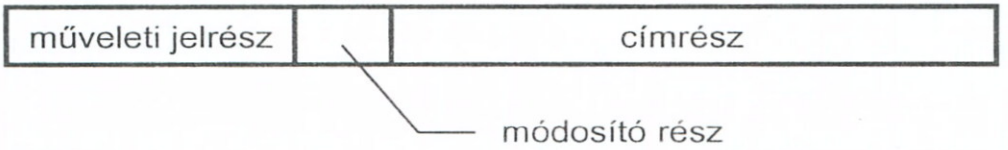
a.)Utasításszerkezet

A számítógépek utasításainak hossza egyes processzoroknál rögzített, fix érték, más típusoknál változó, 2-6 byte hosszúságú. Különösen a mikroprocesszorok körében jelentek meg az igen hosszú(pl. 17 byte az i386-os processzornál) utasítások is.

A változó és gyakran igen hosszú utasítások a CISC processzorokra jellemzőek, míg a redukált utasításkészletű, RISC processzorok utasításai rögzített hosszúságúak és rövidek.

Az **utasítás szerkezete** megszabja azt, hogy a processzornak az utasítás mely részét hogyan kell értelmeznie.

Egy utasításnak feltétlenül tartalmaznia kell azt, hogy annak hatására a processzornak milyen műveletet és mely operandusokkal kell elvégeznie. Pontosabban, az utasításban az operandusok memóriabeli helyének címét találja meg a processzor. Egyes esetekben az operandus maga található az utasításban.



3-12.ábra: *Utasítások általános szerkezete*

Az utasítások értelmezés szempontjából, alapvetően három fő részre bonthatók(3-12.ábra), melyek a következők:

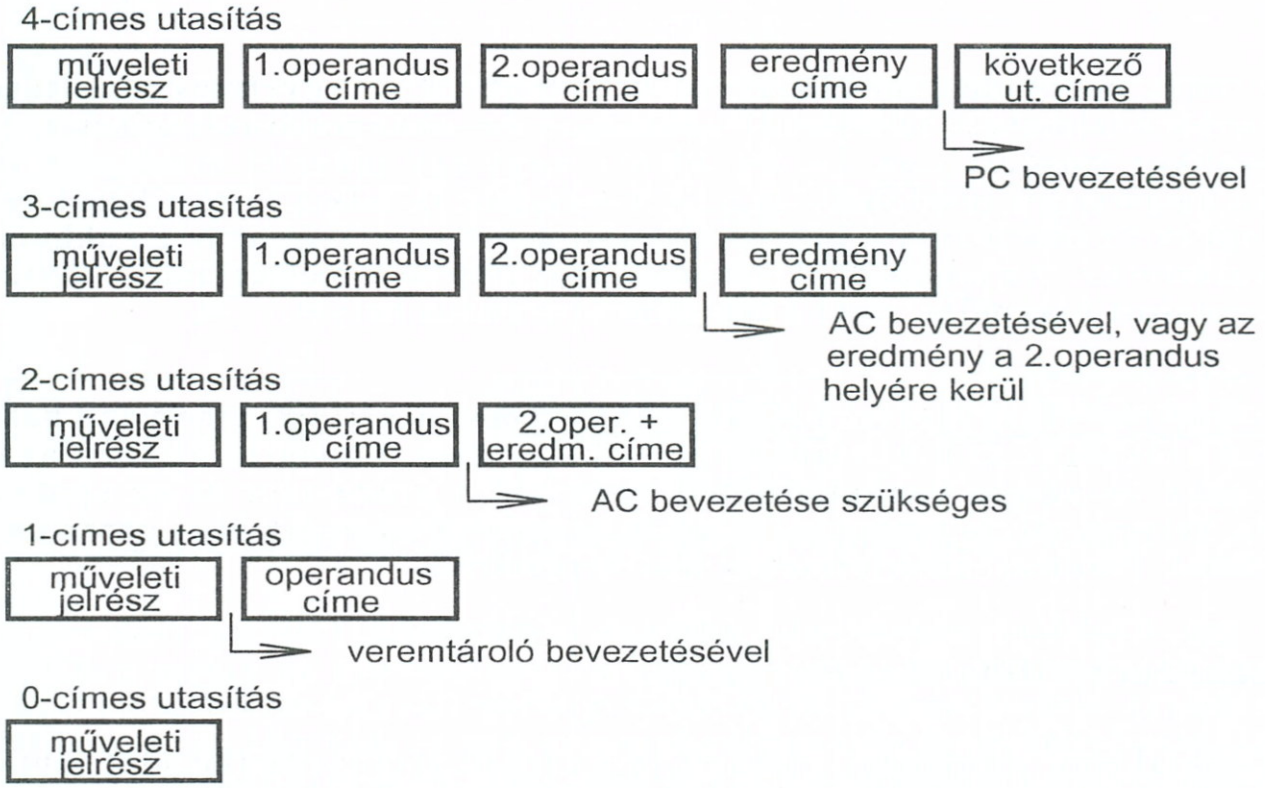
- **műveleti jelrész**(operation code, opcode), amely az elvégzendő művelet, feladat fajtáját adja meg a processzornak,

- **címrész**(address field), amely a művelet végrehajtásához szükséges adat(ok) - operandus(ok) - memóriabeli helyének címeit tartalmazza,
- **kiegészítő, módosító rész**, amely a műveleti jelrész értelmezéséhez, vagy a címek pontos meghatározásához ad módosító előírást.

A program futása közben - egy-egy utasítás végrehajtásakor - a legáltalánosabb esetben, a processzornak négy (memória)címre van szüksége:

- az első operandus címe,
- a második operandus címe,
- az eredmény címe,
- a soronkövetkező utasítás címe.

Attól függően, hogy az utasítás címrésze hány címet tartalmaz, beszélhetünk négy-, három-, két-, egy- illetve nullacímes utasításszerkezetről(3-13.ábra).



3-13.ábra: *Utasításszerkezetek*

A **négycímes utasítás**szerkezet használata, annak ellenére, hogy kevés utasításból álló programot eredményez visszaszorult, mivel igen hosszú utasításokat használhat és így a tárolóbeli helyfoglalása és feldolgozása problémákat okozhat. (Pl. 64KB-os nem túl nagy memóriát feltételezve, annak címzésére 16 bit hosszúságú címek kellenek. Így, egy 4-címes utasítás minimálisan 64 bit hosszúságú.)

A négycímes utasítás címeinek számát, az utasításszámláló regiszter (PC) bevezetésével lehet csökkenteni, mégpedig a soronkövetkező utasítás címének elhagyásával. Az így kapott forma a **háromcímes utasítás**szerkezet. A soronkövetkező utasítás címét az utasításszámláló regiszter tartalmazza annak az automatizmusnak a bevezetésével, hogy minden utasítás végrehajtásakor a processzor automatikusan növeli a PC tartalmát eggyel(pontosabban 1 utasításhossznyi értékkel). Ennek feltétele, hogy a program utasításai a végrehajtás sorrendjében kerüljenek elhelyezésre a tárolóban és így sorra elővehetők legyenek az utasításszámláló regiszter tartalmának növelésével. Ez ugyanakkor szükségessé teszi olyan, vezérlésátadó utasítások használatát, amelyek megbontják az utasítások szigorú soros feldolgozását, esetleg valamilyen feltétel teljesülése következményeként.

A harmadik cím, az eredmény címének elhagyásával keletkező **kétcímes utasítás**forma esetében, az eredményt a processzor automatikusan visszaírja vagy az egyik operandus helyére (természetesen törölve evvel az eredeti értéket), vagy egy erre a célra szolgáló regiszterbe, többnyire az aritmetikai egységhez tartozó akkumulátor regiszterbe (AC=Accumulator Register).

A második operanduscím elhagyásával jutunk el az **egycímes utasítás**hoz. Ennek használatához feltétlenül szükséges az akkumulátor regiszter, mint automatikus második cím. Ez azt jelenti, hogy egy művelet mindig az utasításban megcímzett tárolóhely és az akkumulátor tartalma között kerül végrehajtásra és az eredményt a processzor automatikusan visszaírja az akkumulátor regiszterbe. Az adatátvitel a memóriába, illetve a memóriából, szintén az akkumulátort használja, mint a második operandus helyét.

Az összes cím elhagyásával kapjuk a **nullacímes utasítást**, amelynek használatához szükséges az ú.n. veremtároló alkalmazása. Ebben az esetben az operandusokat mindig a veremtárolóból veszi elő a processzor és az eredményt is oda helyezi el. A veremtároló(stack memory) használatát a veremmutató (SP=Stack Pointer) ismertetésekor már bemutattuk.

Processzorok utasításszerkezetei

Az alábbiakban, mintaként, néhány processzor utasításszerkezetét mutatjuk be tájékoztatásul. A CISC processzorok többségénél a két-, illetve az egycímes utasításszerkezet használata a szokásos; míg a RISC processzorok általában háromcímes utasításokkal dolgoznak. Háromcímes például az AT&T WE3210 processzora, a Sun Microsystems Inc. SPARC processzora.

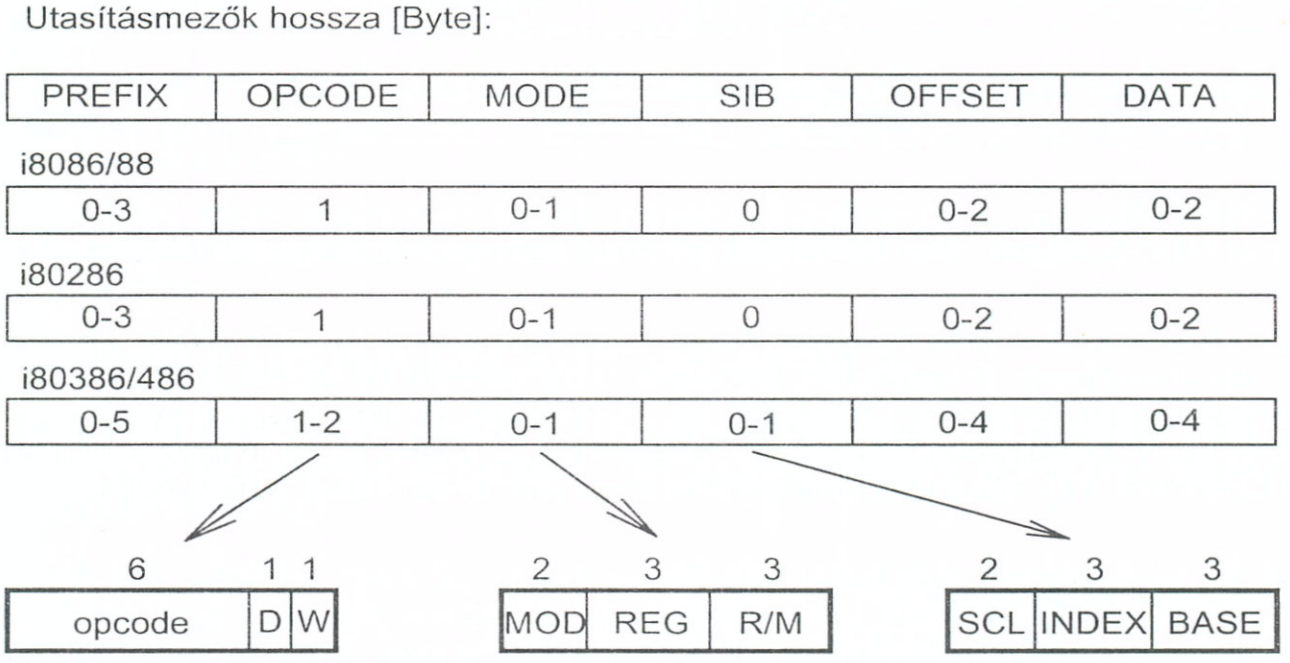
Az utasításszerkezetek kialakítása igen változatos:

1. Vannak olyan processzorok, melyek utasításszerkezete egyértelmű, az egyes részek feldolgozása mindig ugyanúgy történik; a címek kidolgozásának módja minden utasításnál azonos, azaz nem függ a műveleti jelrésztől.

Az ilyen utasításszerkezeteket, utasításkészletet **ortogonális utasításkészletnek** nevezik.

2. Vannak olyan processzorok, amelyek utasításszerkezete nem túl egyszerű kialakítású, de kellően áttekinthető, rendszerezett, mint például az **Intel i8086/286/386/486** processzoroké(3-14.ábra).

   Az utasítások *műveleti jelrésze*(OPCODE) 1-2 byte-os; a 8088/86-os és a 286-os processzoroknál 1 byte-os, míg a 386/486-os processzoroknál már 2 byte-os hosszúságú. A műveleti jelrészhez kapcsolódik két bit, amelyek egyike(D) a memóriabeli operanduscímre vonatkozóan megadja, hogy az adatot onnét, vagy oda kell-e szállítani, míg a másik(W) az operandus méretére(byte/word) vonatkozóan ad információt.



3-14.ábra: *INTEL processzorok utasításszerkezete*

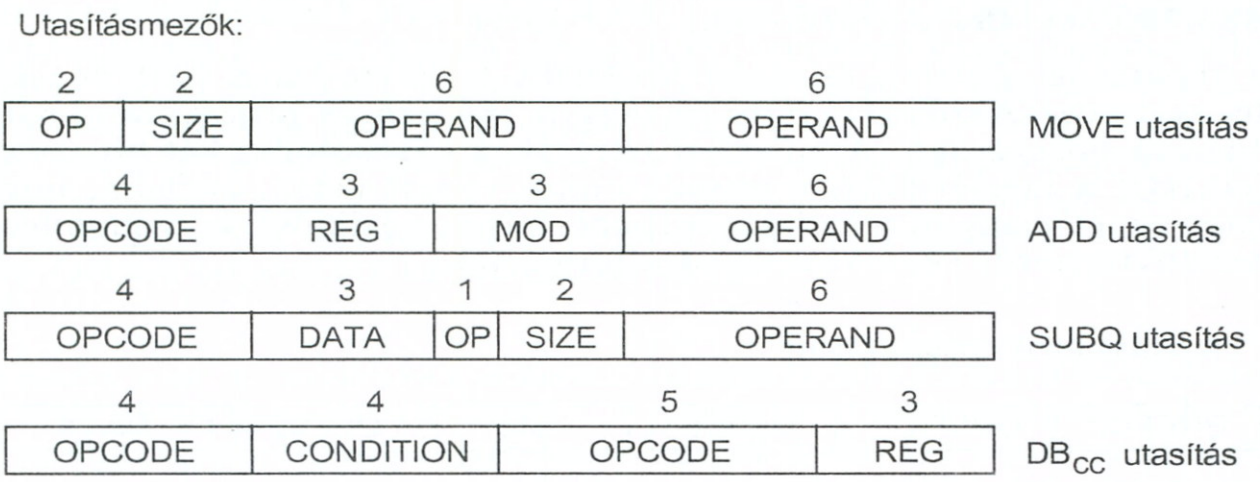
A műveleti jelrészt követően a legtöbb utasításban található egy byte (MODE), amely az *operandusok helyét adja meg*. Az operandusok egyikének(vagy mindkettőjének) mindig valamelyik processzorbeli regiszterben kell lennie. A 386/486-os processzoroknál további byte(SIB=Scale, Index, Base) szolgálhat kiegészítő információval az operandusok helyét illetően.

Ezt követően további 1, 2, vagy 4 byte(DISPLACEMENT, OFFSET) található, amelyek az operandus memóriabeli címét tartalmazzák.

Végül további 1, 2, vagy 4 byte(DATA, IMMEDIATE) szolgálhat egy számkonstans utasításbani közvetlen elhelyezésére.

Az Intel processzorok utasításai hat részből állnak össze, amelyek mindegyike több byte hosszúságú lehet egyenként. Az i8088/86-os és i286-os processzorok utasításainak hossza 1-9 byte között mozog, míg az i386/486-os processzoroké már 1-17 byte közötti lehet.

3. A **Motorola MC68000/68020/68030**-as processzorok utasításszerkezete igen bonyolult módon szervezett. Ezen processzorok néhány utasításának első 2 byte-ját mutatja be a 3-15.ábra.



3-15.ábra: *MOTOROLA processzorok néhány utasítása*

Az ábra alapján már látható, hogy igen bonyolult rendszerben alakították ki az utasítások felépítését, amelyet még növelnek az MC68020/68030-as processzorok további utasításai is. A bonyolult utasításszerkezet egyúttal bonyolult fordítóprogramot is eredményez, mivel igen nehéz optimálisan elkészíteni egy magasszintű nyelven megírt programot gépi kódban.

b.)Címzési eljárások I.

Az utasításszerkezet vizsgálatakor világossá vált, hogy az utasítások végrehajtásához ismerni kell az operandusok(feldolgozandó adatok) tárolási helyét is. Az operandusok címzésére többféle módszert is használnak, annak érdekében, hogy a különböző feladatokat könnyebben lehessen megoldani. Ebben a fejezetrészben a használatos címzési eljárások alapelveit, alapvető és általánosnak tekinthető megoldási módjait tárgyaljuk meg, nem kötve a bemutatott formákat egyetlen processzorhoz sem. A címzési lehetőségek kiterjesztéséről, a virtuális címzésről a 4.fejezetben lesz szó.

A címzési módok bemutatásához a legegyszerűbb, egycímes utasításszerkezetet használjuk fel, de a tárgyalt címzési módok a többcímes utasításokra vonatkozóan ugyanolyan módon értelmezendők.

Az utasítások címrésze általában nem az operandusok pontos(abszolút) címét tartalmazza, hanem azt a processzornak a címzés módjára utaló kiegészítő információk alapján, meghatározott módon ki kell számítania. A pontos címek kiszámítására, meghatározására alkalmazott eljárásokat nevezik **címmódosítási eljárásoknak**. A címmódosítás oka lehet:

- az utasítás címrésze nem elegendően hosszú, hogy minden memóriacím leírható legyen azon,

- adatsorozat elemein kell, általában azonos műveletet végrehajtani,
- ciklikus műveletsort kell elvégeztetni az adatokkal,
- az elkészült program áthelyezhetőségét kell biztosítani, stb.

Az operandusok tárolási helyének kijelölésére használt **alapvető címzési módok** a következők:

- **közvetlen**(direkt; direct) címzés, amely esetében az utasításban maga a tárolóhelycím található; ez a cím vonatkozhat a memóriára, vagy a processzor valamelyik regiszterére;
  a közvetlen címzési módon belül megkülönböztethetünk **abszolút** és **relatív** címzési módot is; az első címzési mód esetében a tényleges címet tartalmazza az utasítás, a második esetben pedig valamilyen alapcímhez viszonyított címet tartalmaz az utasítás;
- **közvetett**(indirekt, vagy címhelyettesítéses; indirect) címzés, amelynél az utasításban megcímzett tárolóhelyen nem az operandus található, hanem annak a címe;
- **literális**, vagy álcímzés(literal, pseudo, immediate), nevezik **közvetlen adatcímzés**nek is, amely esetben magában az utasításban található az operandus.

A címmódosítási eljárások közé sorolandó továbbá az

- **indexelés**(indexing), amely segítségével a megcímzett tárolási hely címértéke folyamatosan növelhető, így egy tárolóhely sorozaton egyesével végighaladhatunk.

A különböző címzési módok, címmódosítások általában együttesen is használhatók és így egészen összetett címkidolgozási előírásokat is kialakíthatunk. (Persze kérdés az, hogy mikor van ezekre szükség?) Legtöbbször valamelyik címzési mód(abszolút, relatív, indirekt) és az indexelés együttes használata szokásos és megengedett. A különböző címkiszámítási változatok alkalmazhatósági területe, azaz, hogy mely utasítások esetében használhatók, általában korlátozott.

Közvetlen(abszolút és relatív) címzési módok

A közvetlen, **abszolút címzési mód** esetén, ahogy az az előzőekben rögzítésre került, az utasítás címrészében az operandus valódi, pontos címe található.

A közvetlen cím vonatkozhat a memóriára(memory direct), vagy a processzor valamelyik regiszterére(register direct). Ez utóbbi esetben, a **regisztercím** megadásához kevesebb helyre van szükség az utasításban, így az ilyen utasítás hossza kisebb, mint a memória hivatkozásos utasításé.

Memóriacímzés esetén a tároló méretétől függően 16-32 bit hosszúságú cím megadására van szükség, míg regisztercímzésnél 3-6 bit hoszszúságú címrészre van csak szükség.

Az abszolút címzés használata nem mindig előnyös, mivel az így elkészült program és a kapcsolódó adatok a memóriában nem helyezhetők át, mert akkor a program összes címét módosítani kellene. Az abszolút cím elsősorban a vezérlésátadó utasításokban okoz gondot, mivel a program áthelyezésével az ugrás célpontja, annak címe is megváltozik. De ugyanígy az adatok helye is megváltozhat, így az ezekre történő hivatkozásokban is módosítani kell a címet.

Relatív címzés esetében az utasítás címrésze az operandus valamilyen alapcímhez(báziscímhez) viszonyított címét tartalmazza. *Alapcím*ként szolgálhat

- egy kijelölt regiszterben, a báziscímregiszterben(base register) elhelyezett érték(ez az adatmező kezdőcíme, egy társzegmens kezdőcíme, stb. lehet),
- a program kezdetének címe,
- magának a végrehajtás alatt lévő utasításnak tárolóbeli helye, amelynél az alapcímet az utasításszámláló regiszter(PC) szolgáltatja.

Az előbbieknek megfelelően beszélhetünk *bázisrelatív*, *programrelatív*, illetve *utasításrelatív* címzésről. Az utasításrelatív címzéshez, többnyire önálló műveleti jelrésszel rendelkező külön utasításfajta használható.

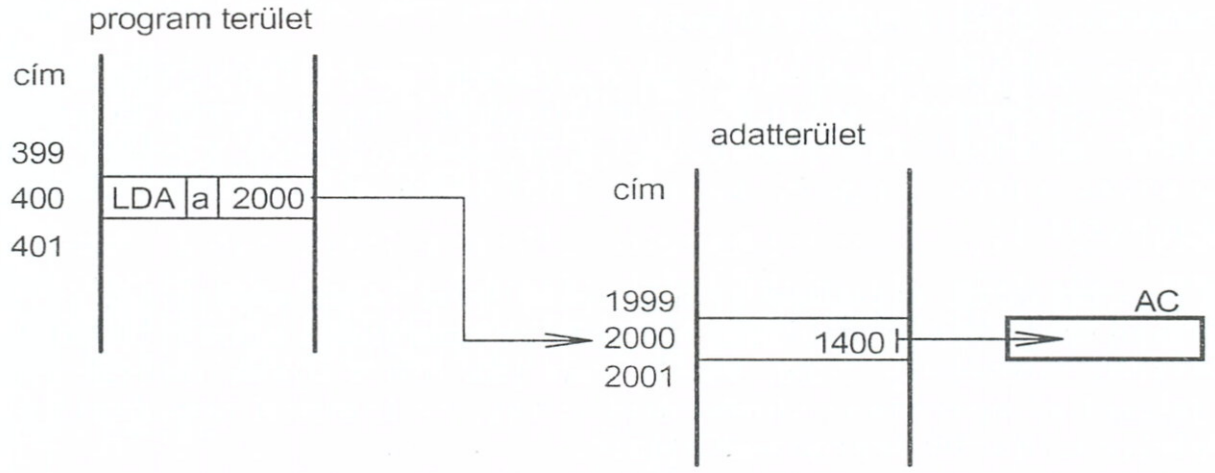
A tárolóhely **pontos címe** két címérték összeadásával, a báziscím és az utasítás címrészében lévő cím összegeként adódik ki.

A *bázisrelatív címzést* legtöbbször a tárolóterület szegmentálására használjuk fel. Azaz annak érdekében, hogy az utasításbeli cím rövidebb lehessen, a memóriát kisebb részekre(szegmensekre, modulokra) osztjuk fel és ezen részek kezdőcímét helyezzük el a bázisregiszterben. Egy-egy ilyen szegmens például, tárolhat csak adatokat, vagy csak programutasításokat. Ezt a megoldást találjuk pl. az Intel processzorok esetében is.

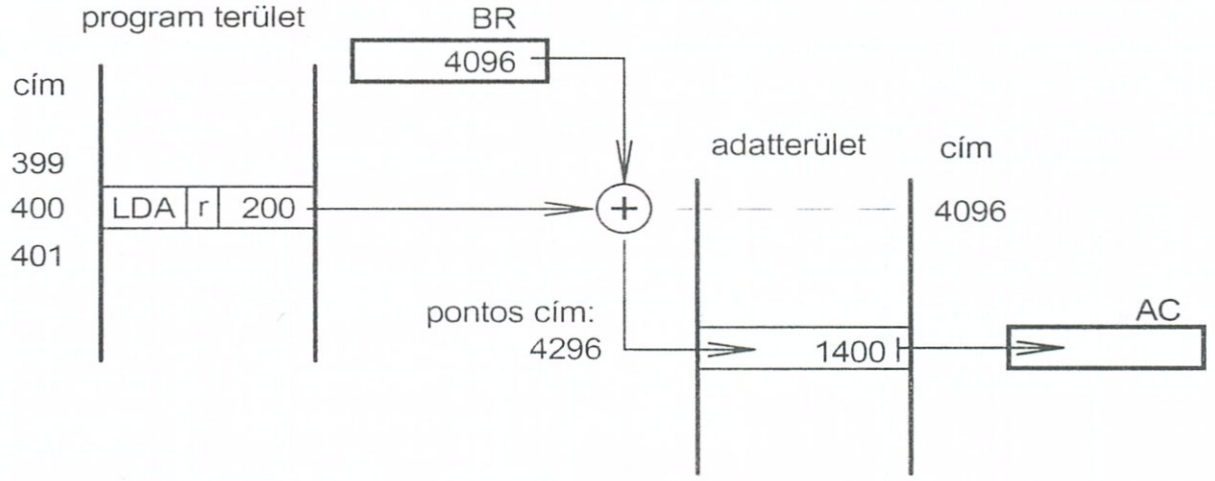
A *programrelatív címzés*(amikor a programbeli címek a program kezdetéhez viszonyítva vannak megadva) talán az egyik legfontosabb alkalmazási területe a relatív címzésnek, mert ez teszi lehetővé a programok memórián belüli áthelyezhetőségét, azaz bárhová betöltve a programot, beállítva az alapcímet a program kezdetére, az futtatható.

Az *utasításrelatív címzés* használatakor, az aktuális, végrehajtás alatt lévő utasítás tárolóbeli helyének szűkebb környezetét lehet elérni(többnyire ± 255 byte-nyi távolságon belül), ezért leggyakrabban relatív ugró (vezérlésátadó) utasításokban találjuk meg ezt a megoldást.

A közvetlen, abszolút és relatív címzést mutatja be a következő, 3-16.ábra példája. Az ábrán egy egyszerű töltő(load) utasítás kapcsán mutatjuk be az egyes címzési módok hatását. Az utasítás a kiszámított című tárolóhely tartalmát tölti az akkumulátor(AC) regiszterbe. Az utasításban külön rész szolgál a címzési mód jelölésére(a=abszolút, r=relatív).



a.) abszolút címzési mód



b.) relatív címzési mód

3-16.ábra: *Abszolút és relatív címzési módok*

Az *abszolút címzési mód* példájában, a 400_{10}-as tárolóhelyen található utasítás végrehajtásakor, a processzor a 2000_{10}-es tárolóhely tartalmát viszi át az akkumulátor regiszterbe, azaz az utasítás hatása:

$$[2000] \Rightarrow AC.$$

A *relatív címzési mód* példájában a 4K-s(4096 byte-os) modulokra osztott memória 4296_{10}-os tárolóhelyének tartalmát kell átvinni az akkumulátorba. A tárolómodul(szegmens) kezdőcíme a bázisregiszterbe (BR) kerül, azaz $4096_{10} \Rightarrow$ BR és az utasításban az ehhez viszonyított címet(200_{10}) kell csak megadni. Tehát az utasítás formája, LDA r 200 és hatása:

$$\text{pontos cím} = [400_{\text{címrész}}] + [BR] = 200 + 4096 = 4296$$

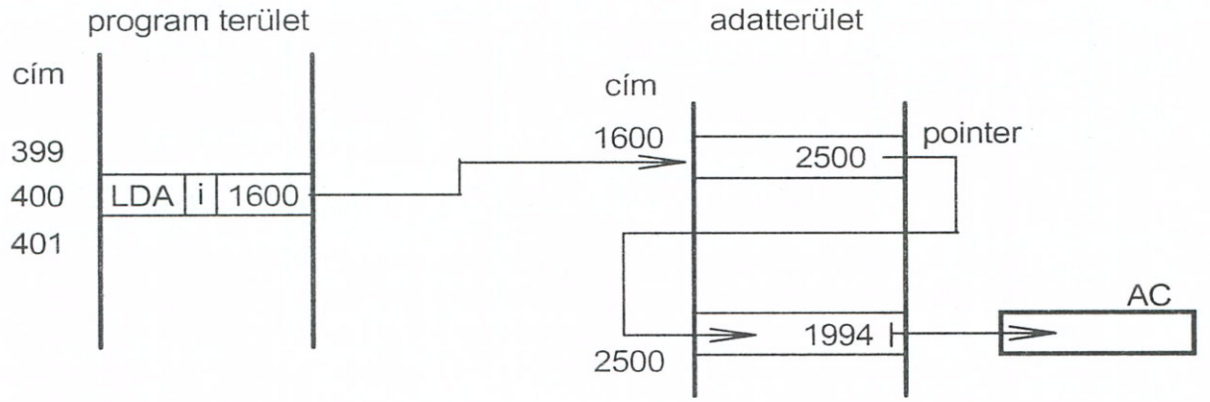
$$[4296_{10}] \Rightarrow AC$$

Közvetett(indirekt) címzési mód

A közvetlen címzésű utasításnál, az operandus memóriabeli címét, vagy annak egy összetevőjét találjuk az utasításban. Az indirekt címzésnél, az utasításban található cím nem magának az operandusnak a címét adja meg, hanem annak a tárolóhelynek a címét, ahol aztán az operandus címét megtalálja a processzor.

Egyes processzorok esetében ez a címzési mód lehet többszintű is, azaz az utasításban megcímzett tárolóhelyen nem az operandus címe található, hanem egy további tárolóhely címe, amely már (ha nincs újabb indirekt címzésre utaló jelzés) az operandus címét tárolja; így, az indirekt címzésnél a pontos cím kidolgozása több lépésben történik.

Az indirekt címzés történhet valamely memória tárolóhely felhasználásával(memory indirect addressing), vagy a processzor valamelyik regiszterének(register indirect addressing) segítségével. Az ezeken a tárolóhelyeken tárolt címeket, amelyek az operandusok címei, 'mutatók'-nak, **pointer**eknek nevezik.



3-17.ábra: *Indirekt címzési mód*

A memória indirekt címzésre mutat példát a 3-17.ábra. A példában a 2500_{10}-as tárolóhely tartalmát szeretnénk az AC-ba átvinni oly módon, hogy az operandus címe(a **pointer**) az 1600-as című tárolóhelyen található.

Az ábra alapján az utasítás végrehajtása egyenértékű a következő utasítás végrehajtásával:

$$\text{LDA } \mathbf{i}\ 1600_{10} \Leftrightarrow \text{LDA } \mathbf{a}\ 2500_{10}$$

és az utasítás hatása:

$$[2500_{10}] \Rightarrow \text{AC}$$

Gyakran alkalmazzák az indirekt címek csoportos tárolását, az ú.n. **vektortáblázatok**at a gyakran használt rutinok(rendszerint rendszerprogramok) kezdőcímeinek elhelyezésére.

Közvetlen adatcímzés, álcímzés

Ennek a címzési módnak(immediate addressing) a használatakor, maga az operandus található az utasítás címrészében. A használható operandusnagyság erősen korlátozott, az utasítás címrészének hossza szabja meg az így tárolható operandusok legnagyobb értékét.

Ezt a címzési módot kisebb értékű konstansokkal való munkához lehet felhasználni, mivel így nagyon könnyen lehet egyes regiszterekbe, tárolóhelyekre konstansokat betölteni. A közvetlen adatcímzés(álcímzés) használatára mutat be egy egyszerű példát a 3-18.ábra.

Az utasítás hatása:

$$639 \Rightarrow AC$$

Indexelés

A feldolgozások széles körében van szükség arra, hogy adatsorozatokon kelljen elvégezni valamilyen műveletet. Ezekben az esetekben azt kell megoldani, hogy valamilyen egyszerű formában, utasítással ciklikusan sorra elő tudjuk venni az egyes tárolóhelyeken lévő adatokat.

Az indexelt utasításokat adatsorozatokon végzett műveletekkor, ciklusokban tudjuk előnyösen használni. Az adatsorozat első elemének tárolási címét tartalmazza az utasítás címrésze és az indexregiszterben(IX) található az ettől való eltérés, azaz hogy hányadik elemet kell a sorozatból feldolgozni. Az indexregiszter tartalmának folyamatos növelésével(vagy csökkentésével) végig tudunk haladni az összes adaton, tárolóhelyen.

Mivel egy-egy adat előkeresése után mindig növelni(csökkenteni) kell az indexregiszter tartalmát, ezért alkalmaznak olyan megoldásokat is, ahol ez a növelés automatikusan megtörténik. Ezt a lehetőséget nevezik **autoindexelés**nek.

Az indexelés alkalmazására mutat példát a 3-18.ábra, amelyben a 3000_{10}-es tárolóhelytől kezdődően tárolt adatsor elemeit kell az akkumulátor regiszterbe tölteni.

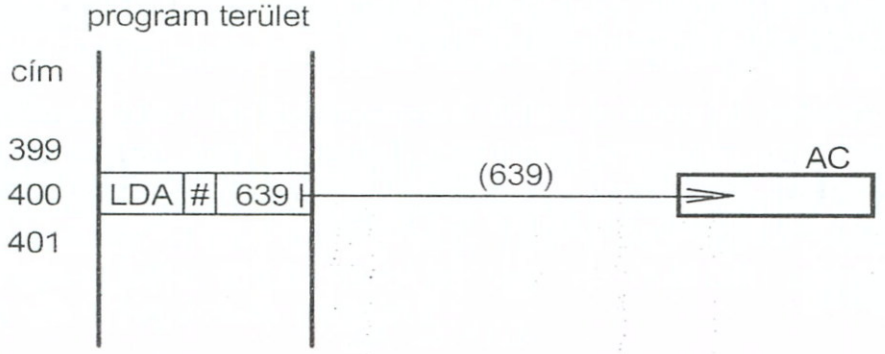
A **tárolóhely pontos címe** az utasítás címrésze tartalmának(mint alapcímnek) és az indexregiszter tartalmának összeadásával jön létre. Az IX indexregiszter tartalmát folyamatosan növelve, végig lehet járni a teljes adatsort. Az ábra jelöléseit figyelembe véve, az utasítás hatása:

$$\text{pontos cím} = [400_{\text{címrész}}] + [IX] = 3000 + 5 = 3005$$

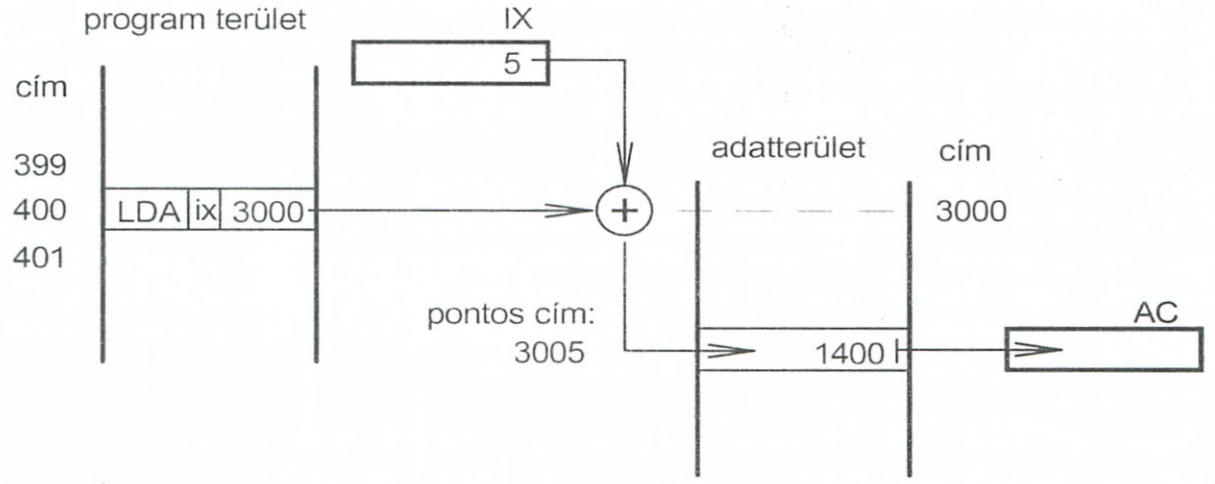
$$[3005_{10}] \Rightarrow AC$$

Az indexelt utasítás és a relatív címzési mód igen hasonlít egymáshoz, mert mindegyiknél egy alapcímtől való eltérés adja meg a pontos tárolóhelycímet. Azonban lényeges különbség van a kétféle címzési lehetőség között. A relatív címzésnél az alapcím egy regiszterben van és ennek értékét nem változ-

tatjuk folyamatosan, az indexelt utasítások esetében az alapcím az utasításban van és a folyamatosan változó rész az indexregiszterben található.



a.) közvetlen adatcímzési mód



b.) indexelés

3-18.ábra: *Közvetlen adatcímzés és indexelés*

Intel és Motorola processzorok címzési lehetőségei

A címzési módok általános megismerése után, az Intel és a Motorola processzorcsalád tagjainak címzési jellemzőit foglaljuk össze a következőkben, amelyekről a 6.fejezetben lesz szó részletesebben. Egyik processzorcsalád esetében sem lehet tiszta, racionális megoldásról beszélni.

Az **Intel processzorok** címzési módjait, az utasításszerkezet ismertetése alkalmával már említett MODE byte írja le. Az egyik operandus helyét a MODE byte MOD és R/M mezői határozzák meg, míg a másik operandus helyét, amely mindig valamelyik regiszterben található, a REG mező tartalma határozza meg.

Az *i8088/86, i80286 processzorok*(és az i80386/486-os processzorok valós üzemmódjának) címzési módjai ugyanazok. A lehetséges címzési módok közül(3-5.táblázat), csak egy használható közvetlen címzésre (direct addressing). Az utasításbani operandus használatára, az 'immediate' címzésre és az autoindexelésre nincs lehetőség. A címzések többségében egy, vagy több regiszter(BX, BP, SI, DI) tartalmának és az utasításban megadott eltolásnak(displacement=d8, d16) az összege adja meg a memória azon helyét, amelynek tartalmát a műveletvégzéshez fel kell használni. Tulajdonképpen ezek a címzési módok indirekt, vagy indirekt relatív címzési módok. A MOD mező [MOD]=11 tartalma mellett, mindkét operandus valamelyik regiszterben található.

3-5.táblázat: *INTEL processzorok címzési lehetőségei valós üzemmódban*

	MOD			
R/M	00	01	10	11
000	m[BX+SI]	m[BX+SI+d8]	m[BX+SI+d16]	AX vagy AL
001	m[BX+DI]	m[BX+DI+d8]	m[BX+DI+d16]	CX vagy CL
010	m[BP+SI]	m[BP+SI+d8]	m[BP+SI+d16]	DX vagy DL
011	m[BP+DI]	m[BP+SI+d8]	m[BP+SI+d16]	BX vagy BL
100	m[SI]	m[SI+d8]	m[SI+d16]	SP vagy AH
101	m[DI]	m[DI+d8]	m[DI+d16]	BP vagy CH
110	direkt címzés	m[BP+d8]	m[BP+d16]	SI vagy DH
111	m[BX]	m[BX+d8]	m[BX+d16]	DI vagy BH

3-6.táblázat: *INTEL processzorok címzési lehetőségei védett üzemmódban*

	MOD			
R/M	00	01	10	11
000	m[EAX]	m[EAX+d8]	m[EAX+d16]	EAX vagy AL
001	m[ECX]	m[ECX+d8]	m[ECX+d16]	ECX vagy CL
010	m[EDX]	m[EDX+d8]	m[EDX+d16]	EDX vagy DL
011	m[EBX]	m[EBX+d8]	m[EBX+d16]	EBX vagy BL
100	SIB	SIB+d8	SIB+d16	ESP vagy AH
101	direkt címzés	m[EBP+d8]	m[EBP+d16]	EBP vagy CH
110	m[ESI]	m[ESI+d8]	m[ESI+d16]	ESI vagy DH
111	m[EDI]	m[EDI+d8]	m[EDI+d16]	EDI vagy BH

A 3-5.táblázat a lehetséges címzési kombinációkat tartalmazza. A táblázatban az m[....] jelölés a zárójelben leírt kifejezés értéke, mint **pointer** által kijelölt memóriahely tartalmát jelenti. A [MOD]=01 oszlop címzési módjaiban 1 byte-os eltolással(d8), a [MOD]=10 oszlop címzési módjaiban 2 byte-os eltolással(d16) kell számolni.

Az *i80386/486-os processzorok* a 32 bites, védett üzemmódjukban, eltérő módon határozzák meg az operandusok helyét. Az ehhez tartozó táblázat a

3-6.táblázat. A címek kiszámításában a MODE byte-on kívül részt vesz a SIB(Scale, Index, Base) byte is.

A **Motorola processzorok** címzési módjait az utasítások MODE és REG mezői határozzák meg. A kialakított rendszer hasonlít a DEC PDP-11 típusú minigépének cimzési módjaihoz, de attól bizonyos mértékig elmarad, kevesebbet valósít meg. Az MC68000-es processzor 12 címzési lehetőséggel rendelkezik, amelyet a későbbi processzoroknál, az MC68020, MC68030-as változatoknál kibővítettek néhány újabb változattal, így 18-ra növelve a címzési módok számát. Az alkalmazható címzési eljárásokat, módokat a 3-7.táblázat tartalmazza(dpl=eltolás).

3-7.táblázat: *MOTOROLA processzorok címzési lehetőségei*

Címzési mód	Címkiszámítás
regiszter direkt a **D**(ata) regiszterekkel	Dn
regiszter direkt az **A**(ddress) regiszterekkel	An
regiszter indirekt az **A** regiszterekkel	m[An]
regiszter indirekt az **A** regiszterekkel, az A regiszter utólagos növelésével(postincrement)	m[An]+
regiszter indirekt az **A** regiszterekkel, az A regiszter előzetes csökkentésével(predecrement)	-m[An]
regiszter indirekt, az **A** regiszterekkel, eltolással	m[dpl+An]
indexelt regiszter indirekt, (8 bites) eltolással	m[dpl+An+Xn]
indexelt regiszter indirekt, báziseltolással	m[dpl+An+Xn]
memória indirekt, utóindexeléssel(post-indexed)	m[m[dpl1+An]+Xn+dpl2]
memória indirekt, előindexeléssel(pre-indexed)	m[m[dpl1+An+Xn]+dpl2]
utasításszámláló(PC) indirekt, eltolással	m[dpl+PC]
indexelt PC indirekt, (8 bites) eltolással	m[dpl+PC+Xn]
indexelt PC indirekt, báziseltolással	m[dpl+PC+Xn]
PC-n keresztüli memória indirekt, utóindexeléssel	m[m[dpl1+PC]+Xn+dpl2]
PC-n keresztüli memória indirekt, előindexeléssel	m[m[dpl1+PC+Xn]+dpl2]
abszolút, rövid címmel	cím
abszolút, hosszú címmel	cím
immediate	konstans

c.)Utasítástípusok, utasításkészlet

A 3.2.2.a.pontban a gépi utasítások szerkezetét részletesen bemutattuk. A tárgyaltak szerint, az utasítások lényegi részét alkotja az utasítások **műveleti jelrésze**(opcode), valamint az operandusok helyét meghatározó **címrésze**. Az értelmezhető műveleti jelrészek összessége határozza meg egy processzor,

egy gép utasításkészletét. A rendelkezésre álló utasítások köre fontos jellemzője a processzoroknak.

Utasítástípusok

Az utasítások műveleti jelrésze alapján az utasítások alábbi csoportjai alakíthatók ki:

- **átviteli utasítások**, amelyek a gép két része(pl. processzor és memória, processzor és veremtár, processzor és periféria) közötti adatátvitelre szolgálnak;
- **műveleti utasítások**, amelyek körébe az aritmetikai és logikai műveleteket végrehajtó utasítások tartoznak;
- **vezérlő utasítások**, amelyek a program végrehajtását, vagy a gép működését befolyásolják.

1.Átviteli utasítások

Az átviteli utasítások köre az adatátvitel cél(forrás)területe alapján három csoportra bontható, amelyek a következők:

- tároló(regiszter, memória) utasítások,
- veremkezelő utasítások,
- periféria utasítások.

Az átviteli utasítások egy, vagy több byte átvitelére szolgálnak. Az átvihető byte-ok száma a processzor típusától és az utasításkészlettől függ. A rögzített utasításhosszal rendelkező processzorok általában külön utasítással rendelkeznek a byte-os és külön utasítással a szavas(2, 4, esetleg 8 byte-os) átvitelre; míg a változó hosszúságú utasításokkal dolgozó processzorok esetében, magában az utasításban kell megadni, hogy byte-os, vagy szavas átvitelt kell-e végrehajtani. Gyakran lehetőség van olyan átviteli utasítás használatára, amelyben csak azt kell megadni, hogy honnan hová kell az adatokat továbbítani, de az átviendő adatmennyiséget nem kell meghatározni. Az adatátvitel addig tart, amíg az adatsor-vége jelet eléri a processzor.

Az átviteli utasítások végrehajtásakor, az átvitt adat az eredeti helyén továbbra is megtalálható, megmarad; ezért ilyen értelemben célszerűbb lenne inkább másolási, mint átviteli utasításról beszélni.

A tároló hivatkozású átviteli utasítások esetében, az átvitel szinte kizárólag memória-regiszter, regiszter-memória párok között bonyolítható csak le. Memória-memória közötti közvetlen átvitel nem használatos.

Az átviteli utasítások külön csoportját képezik a **periféria utasítások**(I/O utasítások). A perifériákat a mikroszámítógépek háromféle módon érhetik el (nagygépeknél más módszer, a 'csatorna' használata is szokásos):

- programozott átvitellel,
- megszakításos átvitellel,
- közvetlen memóriaelérés(DMA) használatával.

Programozott átvitel alkalmazásakor karakterenként történik az adatok továbbítása, figyelve a periféria állapotát, azaz azt, hogy a következő karakter átvihető-e, vagy sem. Ezt a lehetőséget az egyszerűbb processzorok, számítógépek esetében alkalmazzák.

Megszakításos rendszerű átvitelnél, a periféria kezdeményezi az átvitelt akkor, amikor az az átvitelre készen van. Ez a rendszer is karakterenkénti átvitelre ad lehetőséget, de gyorsabb a programozott átvitelnél.

A *közvetlen memóriaelérés*(DMA=Direct Memory Access) használata adja a legjobb eredményt, elsősorban karaktersorozat átvitelekor.

Ezeket a lehetőségeket az 5.fejezetben ismertetjük részleteiben.

Az átviteli utasítások különleges utasításai a **veremkezelő utasítások**(push, pop). Ezek az átvitelt a verem teteje és egy meghatározott regiszter, általában az akkumulátor regiszter között bonyolítják le. A verem tetejét a veremmutató(stack pointer) jelöli ki.

2.Műveleti utasítások

A műveleti utasítások egy, vagy két operandussal dolgoznak, azokkal végeznek el valamilyen aritmetikai, vagy logikai jellegű műveletet. A műveleti utasítások között a következőket különböztethetjük meg:

- aritmetikai műveleti utasítás,
- logikai műveleti utasítás,
- léptető/forgató(shift/rotate) utasítások,
- bitműveleti utasítás,
- karakterlánc(string) műveleti utasítás.

Az **aritmetikai utasítások** közé soroljuk, a minden processzornál rendelkezésre álló bináris egészek, illetve ezt kiegészítően, BCD számok összeadására, kivonására szolgáló utasításokat, továbbá a 16, 32 bites processzorok esetében rendelkezésre álló szorzási, osztási utasításokat. 32 bites processzorok esetében többnyire ez kiegészül a lebegőpontos műveleti utasítások használatával is.

Az aritmetikai műveletek végrehajtásakor, az eredménytől függően, az állapotregiszter egyes jelzőbitjeit(flag-jeit) a processzor beállítja. Ezeknek a tartalmát megvizsgáltatva, a programozónak lehetősége van a programvégrehajtás vezérlésére, elágazások kialakítására. A műveletek eredményét visszatükröző legfontosabb jelzőbitek a következők:

- átvitel(**carry**) jelzőbitje: ha az eredmény legmagasabb helyiértékén átvitel keletkezik, 1-es értéket vesz fel;
- nulla(**zero**) jelzőbitje: ha az eredmény nulla értékű, 1-es értéket vesz fel;
- előjel(**sign**) jelzőbitje: ha az eredmény negatív, akkor az értéke 1-es lesz;
- túlcsordulás(**overflow**) jelzőbitje: ha az eredmény nagyobb, mint a tárolható legnagyobb érték, akkor értéke 1-es lesz.

A **logikai utasítások** néhány elemi logikai művelet végrehajtását teszik lehetővé. Ezek a műveletek leggyakrabban a 'logikai ÉS'(AND), a 'logikai VAGY'(OR), a 'kizáró VAGY'(EXOR) és a 'tagadás'(NOT) művelete. Ezek a műveletek, amelyekről részletesen a következő, 3.3.pontban lesz szó, mindig bitenként értendők, azaz az operandusok összes bitjén egyenként kerülnek végrehajtásra.

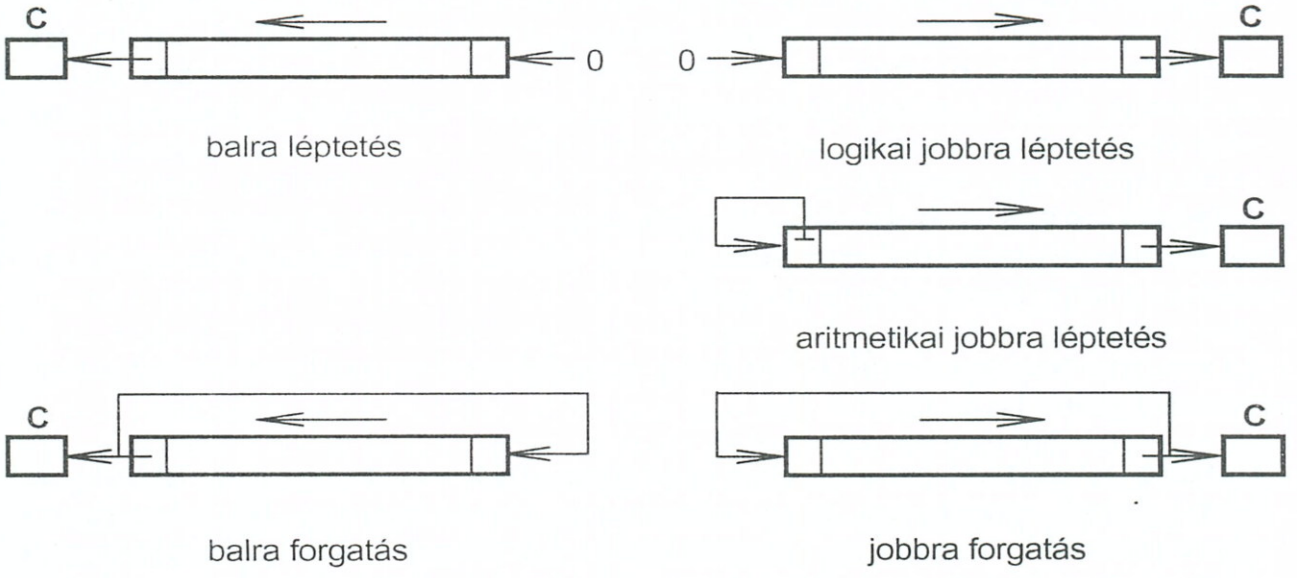
A **léptető és forgató utasítások** az operandusok bitjeit jobbra, vagy balra egy, vagy több helyiértékkel eltolják, körbeforgatják, azaz az egyik oldalon kilépő bitet a másik oldalon belépteti. Az utasítások végrehajtásakor a kilépő bitek egyúttal az átvitel(carry) bitbe is belépnek(3-19.ábra).

Balra léptetéskor(ú.n. aritmetikai, vagy logikai léptetéskor) a legalsó helyiértékre 0 lép be, a kilépő legfelső bit az átvitel(carry) bitbe íródik.

'Logikai' jobbra léptetéskor baloldalon, a legmagasabb helyiértékre 0 lép be és a legalsó helyiértéken kilépő bit az átvitel bitbe íródik. *'Aritmetikai' jobbra* léptetéskor baloldalon mindig a legfelső helyiérték bitje lép be ismétlődően és a legalsó helyiértéken kilépő bit az átvitel bitbe íródik.

Balra forgatáskor a legfelső helyiértéken kilépő bit a legalacsonyabb helyiértéken, jobbról belép, ugyanakkor az értéke az átvitel jelzőbitbe is beíródik.

Jobbra forgatáskor a legalacsonyabb helyiértéken kilépő bit a legfelső helyiértéken, balról belép, ugyanakkor az értéke az átvitel jelzőbitbe is beíródik.



3-19.ábra: *Léptető és forgató utasítások*

A **bitműveleti utasítások** segítségével az operandusok egyes bitjeinek értékét ellenőrízhetjük(bit test), vagy beállíthatjuk 0, vagy 1 értékre. Ha a processzor nem rendelkezik ilyen utasítással, akkor ugyanezek a feladatok a logikai műveletek és a léptető/forgató utasítások segítségével is elvégezhetők.

A **karakterlánc-műveleti utasítások** segítségével, amennyiben ezekkel az utasításokkal egyáltalán a processzor rendelkezik, karakterláncban *kereshetünk* egy megadott másik rövidebb-hosszabb karaktersort, *összehasonlíthatunk* két karakterláncot, illetve megadott helytől kezdődően, megadott helyre megadott hosszúságú karakterláncot *vihetünk át*.

3. Vezérlő utasítások

A processzor vezérlő utasításai *egyrészt* a program feldolgozási folyamatát irányítják, feltételes, vagy feltétel nélküli vezérlésátadó('ugró') utasításokkal, *másrészt* a processzor működését szabályozzák.

A **program futását** befolyásoló, vezérlő utasítások közé tartozik:

- **Feltétel nélküli vezérlésátadó(ugró) utasítás**, melynek végrehajtása után a processzor a program egy másik pontjától folytatja az utasítások feldolgozását; ezeknél az utasításoknál az utasításszámláló regiszter (PC) tartalmát tölti fel a gép az utasításban szereplő címmel, amely a következő végrehajtandó utasítás címe lesz.

- **Feltételes vezérlésátadó(ugró) utasítás**, amelynek hatására a processzor valamilyen, az utasítás műveleti jelrésze által előírt feltétel teljesülését ellenőrzi és amenyiben az teljesül, akkor az utasításban megadott című utasítással folytatódik a program végrehajtása, egyébként pedig a soronkövetkező utasítással.

- **Szubrutin(alprogram)hívó utasítás**, amelynek eredményeként, a vezérlés joga átkerül az utasításban megadott címen kezdődő szubrutin első utasítására és a processzor mindaddig az itt lévő utasításokat hajtja végre, amíg egy visszaugró(return) utasítást nem talál. Ekkor a program végrehajtása az eredeti helyre tér vissza. A szubrutinhívó utasítás tulajdonképpen egy speciális, feltétel nélküli ugró utasítás, amely egyúttal megőrzi annak a helynek a címét, ahonnét ez az elugrás történt(pontosabban az ezt követő utasítás címét, ahonnét a visszatérés után, a program végrehajtását folytatni kell).

  Egyes processzoroknál lehetőség van feltételes szubrutinhívó utasítások használatára is.

- **Return utasítás**, amely egy alprogramból visszaugrat a kiindulási helyet követő utasításra. Ez a viszatérés történhet egy feldolgozást végző alprogramból, vagy egy megszakítási kérelmet kiszolgáló alprogramból. Általában a két *return* utasítás működésében különbözik egymástól.

- **Ciklusutasítás**, amely a programutasítások ciklikus végrehajtását segíti elő egy feltétel teljesüléséig.

A **gép működését szabályozó utasítások** közé tartoznak a következők:

- **Megszakítást engedélyező és tiltó utasítások**, amelyek segítségével programból meg lehet tiltani a perifériák által kezdeményezett - a programfutást felfüggesztő - kérelmek elfogadását, illetve újra engedélyezni

lehet azok elfogadását. (A megszakításokról az 5.2.fejezetrészben lesz szó részletesen.)

- Processzor működését **leállító utasítás**(halt), amelynek végrehajtása után a processzor megszünteti a program további feldolgozását, többnyire egy speciális megszakítási kérelem(reset) beérkeztéig.

Utasításkészlet

A processzorok sokoldalúságának egyik kifejezője az alkalmazható utasítások körének feladat szerinti, funkcionális felosztása, amelyet az előző részben, az utasítástípusok tárgyalásakor ismertettünk. A processzor további jellemzőit adja magának a teljes utasításhalmaznak a vizsgálata.

Egy processzor **utasításkészlete** alatt értjük azoknak az elemi(gépi kódú) utasításoknak a összességét, amelyek végrehajtására legalsó, hardver szinten a processzor alkalmas.

Ez az a szint, amit adott esetben egy programozó felhasználhat, vagy amelyre a fordítóprogram egy programot lefordít. Ebből a szempontból a legalsó szint az jelenti, hogy az utasítás végrehajtása vagy 'huzalozott' módon, áramköri szinten valósul meg, vagy az elemi lépéseket vezérlő mikroprogram segítségével. Az utasításvégrehajtás említett módjairól az evvel foglalkozó fejezetrészben, a 3.4.pontban részletesen is szó lesz.

A processzorok utasításkészlete kapcsán felmerül azok értékelési problémája is: milyen jellemzők alapján történjen ez és milyen tulajdonságokat kell megkövetelni egy 'jó' utasításkészlettől.

Az **utasításkészlet jellemzésénél** általában a következő szempontokat szokták figyelembe venni.

- A rendelkezésre álló **elemi utasítások száma és az utasítások tartalma**, azaz, hogy milyen feladatot oldanak meg végrehajtásukkor. Nyilván minél többféle feladatot tud megoldani a processzor elemi szinten, annál sokoldalúbban lehet azt alkalmazni. Ugyanakkor nagyon sok utasítás csak ritkán kerül felhasználásra, és így feleslegesen köti le az áramköri lapka(chip) területét a hozzá tartozó elektronika, vagy mikroprogram-tároló. Az utasításokba zsúfolt feladatok ugyan növelik az egy utasítással végrehajtható műveletek számát és evvel csökkentik a program hosszát, ugyanakkor egy-egy utasítás feldolgozásának időtartama megnövekszik és így végül is lelassul a feldolgozás.

- Az utasításokkal **kezelhető feltételek száma**, hasonlóan az utasításszámhoz, fontos jellemzője egy utasításkészletnek. Ha sokféle feltétel vizsgálata lehetséges önálló utasításokkal, az egyszerűbbé teheti a lefordított programot, ugyanakkor az elvégzendő elemzési lépések miatt, maga a fordítóprogram válik bonyolultabbá. Ugyancsak kérdéses az egyes feltételvizsgálatok felhasználói programbeli használatának gyakorisága.

- Az utasítások **jellemzőinek következetes használati lehetősége**. Ez azt jelenti, hogy pl. egy címzési mód használata, egy feltétel kezelése minden esetben, minden utasításnál, ahol erre lehetőség van, ugyanúgy történjék. Azaz a kivételek száma minimális legyen.

- Az utasításkészlet kialakítása mennyire **nyújt hatékony támogatást**
  - a programíráshoz,
  - a programok fordításához,
  - az ellenőrizhetőséghez.

  Ez azt jelenti, hogy az utasítások által végrehajtott feladatok tegyék egyszerűbbé a programozó munkáját, azaz a gyakran előforduló feladatokhoz legyen megfelelő gépi utasítás; az utasítások kialakítása legyen egyszerű, hogy a fordítóprogramnak minél kevesebb elemzést kelljen elvégeznie; legyenek olyan utasítások, amelyek lehetővé teszik a gép egyes részeinek, a program futási állapotának ellenőrzését, védelmi szintjeinek kialakítását.

Az **utasításkészlet jóságának** meghatározására nincsen egzakt eljárás, vagy szempontrendszer. W.A.Wulf(1981) a következőkben jelölte meg az utasításkészlet jóságának elbírálási szempontjait(elsősorban a fordítás szempontjából):

- Az utasításkészlet minden utasítása azonos módon működjön, azaz a gép erőforrásait(pl. memóriát) azonos módon használja, tehát legyen egy szabályrendszer, amely általánosan érvényes az eszközhasználatra.

- Az utasításkészlet legyen 'ortogonális', azaz az utasítások egyes részeinek feldolgozása mindig ugyanúgy történjék; a címek kidolgozásának módja minden utasításnál azonos legyen, azaz ne függjön a műveleti jelrésztől.

- Minden utasítás mellett bármelyik címzési mód, adattípus használható legyen; ezek ne legyenek különbözőek az egyes utasításcsoportok esetében.

- Ha valamilyen feladat megoldására többféle lehetőség is van, akkor ezek bármelyike használható legyen.

- Egyszerű utasítások kialakítása, azaz egy-egy utasítás ne tartalmazzon bonyolult műveleti előírásokat és címzési módokat.

- A címzési módok tetszőleges használata ne legyen korlátozva egyes utasításoknál valamilyen egyedi szempont szerint.

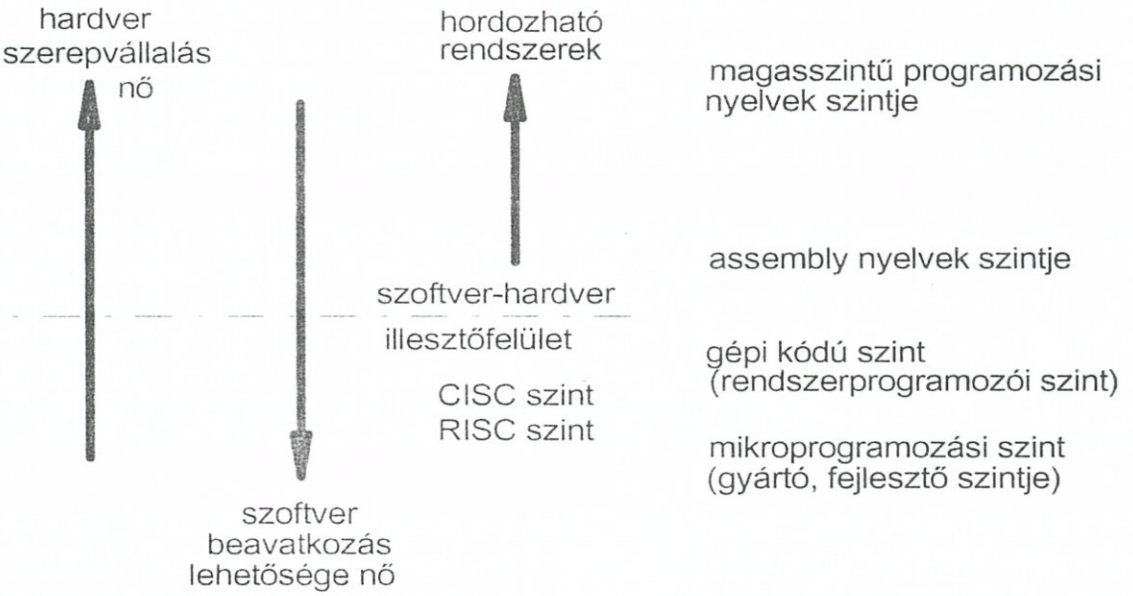
- Támogassa a környezeti feltételek kialakítását.

- Eltérés az alapelvektől csak abban esetben legyen, ha az az eltérés megvalósítás-, kivitelezésfüggetlen, tehát nem függ attól, hogy milyen gépi környezetbe kerül a processzor.

Az **utasításkészlet kialakítását** nagymértékben befolyásolja az, hogy a tervező milyen feladatokat bízott az elektronikára, a hardverre és milyen feladatok megoldásában teszi szükségessé a programozástechnikai, szoftver megoldásokat.

A korszerű hardvermegoldások egyre több feladat megoldását tudják átvállalni, azaz olcsó, kivitelezhető módon, 'huzalozva' valósítanak meg összetett műveleteket. Ugyanakkor a programozási oldal is egyre mélyebben tud beavatkozni a hardver működésébe, azaz programozott módon lehet az elektronika egyes részeit használatba venni, vagy letiltani azok használatát.

A felhasználói, programozástechnikai oldal és az elektronika között az egyensúlyi helyzet megtalálása igen nehéz feladat. A programok hordozhatóságát nagymértékben elősegítené, ha kialakítható lenne egy egységes csatlakozási felület(**software-hardware interface**) a programozástechnikai oldal és az elektronikai oldal között. Az elmondottak bemutatására a 3-20.ábra szolgál, felvázolva a két oldal közötti átmenet egyes szintjeit.



3-20.ábra: *Szoftver-hardver átmenet*

A magasszintű programozási nyelvekben használt összetett utasítások(feltételes vezérlésátadó utasítások, ciklusutasítások, eljáráshívó utasítások) többnyire nem rendelkeznek gépi szintű megfelelőkkel, azaz ezek feldolgozásához több, egyszerűbb utasítás egymásutáni igénybevétele szükséges. Ennek, a szoftver és a hardver közötti távolságnak(semantic gap) a csökkentése elsődleges cél.

Összetett(CISC) és csökkentett(RISC) utasításkészlet

A számítógépek utasításkészlete a kezdeti időszakban, a '60-as évek elejéig, igen egyszerű és kevés számú utasításból állt. Az adatok elérésére, memóriából való kikeresésére kis számú címzési mód volt használható. Ennek oka a hardver igen drága volta és a fordítóprogramok hiánya volt. A programozók-

nak gépi szinten kellett programjaikat megírni. Bár M.V.Wilkes már 1951-ben javasolta az utasítások műveleti vezérlésének megoldásához a mikroprogram(lásd 2.1.2.a.pontban, illetve 3.4.2.pontban) használatát, erre nem kerülhetett sor a '60-as évek előtt, annak technikai megoldhatatlansága miatt. Ebben az értelemben ezek a korai gépek csökkentett utasításkészletű, RISC gépek voltak.

1964-ben, az IBM 360-as gépcsalád bevezetésével jelent meg a kereskedelmi forgalmazású gépekben a mikroprogramozott műveleti vezérlés az utasítások végrehajtására. Ennek következtében növekedett az utasítások bonyolultsága, azaz egyre több funkciót biztak egyetlen utasításra, emellett a használható utasítások száma is növekedett(több, mint 200-250 utasítás). Az utasítások bonyolultsága azért növekedett különösen, mert a ROM tárolóban elhelyezett mikroprogram végrehajtása gyorsabb volt, mint az akkor használt ferritgyűrűs központi tárak által biztosított sebesség. Célszerűbb volt csökkenteni a tárhoz fordulásokat(ez növelte az utasítások komplexitását), hogy a feldolgozás gyorsasága növekedjen. Egy másik tényező, amely a bonyolult gépi utasítások kialakulása felé vezetett, a magasszintű programozási nyelvek széleskörű elterjedése. Az ezekben használt összetett utasítások (feltételes ugró utasítások, ciklusutasítások, stb.) miatt, meg kellett alkotni ezek gépi szintű megfelelőjét, amelyeket összetett mikroprogramok, mikroeljárások segítségével valósítottak meg. Ekkor alakultak ki az *összetett utasításkészletű* **CISC**(Complex Instruction Set Computer) **számítógépek**.

A '70-es évek technológiai fejlődésének eredményeként megjelenő, gyors működésű, félvezetős RAM tárak nem tették már szükségessé a központi tárhoz fordulások erőteljes csökkentését, sőt, a kialakuló igen bonyolult mikroprogramok(gépi utasítások) lelassitották a feldolgozás gyorsaságát. Emellett egyre nehezebb lett az egyre bonyolultabb mikroprogramok hibátlan előállítása, tesztelése. Ezek a tények vezettek el az egyszerűsítés szükségességének felismerése felé.

Az utasítások használatának gyakoriságára vonatkozó vizsgálatok azt mutatták, hogy csak néhány utasításfajta az, ami igen sokszor használatos egy-egy programban. Tehát nem feltétlenül szükséges minden utasítást megtartani. Ez egyúttal a fordítóprogram munkáját is egyszerűsítette. Mivel a mikroprogram a hozzátartozó mikrovezérlővel együtt, tulajdonképpen egy számítógép a számítógépen belül, felmerült a lehetősége annak, hogy a felhasználói programokat közvetlenül a mikroutasítások szintjére kellene fordítani és elhagyni az egész mikrovezérlő rendszert. Ez az elképzelés a **RISC**(Reduced Instruction Set Computer) **processzorok** alapgondolata, azaz egyszerű(mint a mikroutasítások) és viszonylag kevés számú utasítás használata, elhagyva a mikroprogramot az utasítások végrehajtásából.

A RISC processzorok gyakorlati megvalósítását azonban az segítette elő, hogy olyan fordítóprogramokat(fordítási technikákat) sikerült kifejleszteni, amelyek mikroprogram szinten is hasonló minőségű programot voltak képesek előállítani, mint amilyeneket egy jó programozó tud elkészíteni.

Azt lehet mondani, hogy a RISC processzorokkal a gyártás visszatér a kiindulási ponthoz, természetesen egy sokkal magasabb technológiai szinten.

CISC processzorok utasításkészletének főbb jellemzői:

- Az utasítások bonyolult műveletsor végrehajtását eredményezik és ehhez több végrehajtási ütemet(gépi ciklust) használnak fel.
- Sokféle utasítás(100-300) és címzési mód(8-20) használatának lehetősége. Pl.: az i386/486-os processzornál 111 utasítás, 8 címzési mód; az MC68020-as processzornál 109 utasítás, 18 címzési mód használható.
- Sokféle, tárolót közvetlenül igénybevevő, megcímző utasítás használati lehetősége.
- Mikroprogramvezérelt utasításvégrehajtás; a mikroeljárások igen bonyolultak és összességükben nagyon sok tárolót igényelnek(pl. a DEC VAX gép esetében 456 Kbyte-ot).
- Az utasítások változó hosszúságúak; a gyakrabban használt utasítások rövidebbek. Pl.: a DEC VAX 11/780-as gépnél 2-57 byte, az i386/486-os processzornál 1-17 byte közötti hosszúságúak.

RISC processzorok utasításkészletének főbb jellemzői:

- Kevéssé bonyolult utasítások; a felhasználói terület elemzése alapján a bonyolult utasítások elhagyása. Az utasítások végrehajtásához egy gépi ciklust használnak fel.
- Kevés utasítás(<128) és címzési mód(2-4) használata.
- Memóriahasználatra csak 2 (LOAD és STORE) utasítás áll rendelkezésre.
- Az utasítások végrehajtásához nincs mikroprogram; az igen bonyolult fordítóprogram állítja elő a végső formát.
- Az utasítások rögzített hosszúságúak.

3.3.MŰVELETVÉGZÉS, ARITMETIKAI EGYSÉG

A számítógép egyik legfontosabb egysége az aritmetikai egység (ALU=Arithmetic Logic Unit), amely az utasításban előírt műveleteket az adatokon végrehajtja. Az utasítástípusok tárgyalásakor megismert műveletek (aritmetikai, logikai, léptetési/forgatási, bit- és karakterlánc műveletek) közül az alapvető fontosságú aritmetikai és logikai műveletvégzést vizsgáljuk meg részletesebben.

Az aritmetikai egység az **aritmetikai műveleteket**(összeadás, kivonás, szorzás, osztás) általában a kettes számrendszer szerint hajtja végre, de szükség szerint valamelyik binárisan kódolt decimális kód(pl.: BCD kód) felhasználásával tízes számrendszer szerint is dolgozhat a számítógép. A műveletek többnyire akár fixpontos, akár lebegőpontos formában elvégezhetők.

Az aritmetikai egység feladata néhány **logikai művelet**(logikai ÉS, VAGY-művelet, komplementálás, stb.) elvégzése is, amelyet, ahogy arról már szó volt, bitenként hajt végre a processzor.

3.3.1.Aritmetikai műveletek

Az aritmetikai műveletek végrehajtásának módját elsősorban a kettes számrendszerbeli számok körében vizsgáljuk, külön a fixpontos adatábrázolási formában és külön a lebegőpontos számábrázolás szerint.

a.)Műveletek kettes számrendszerben

Műveletek fixpontos számokkal

Kettes számrendszerben az aritmetikai műveleteket ugyanolyan módon lehet elvégezni, mint a tízes számrendszerben, figyelembe véve azt, hogy a számrendszer alapszáma más. Az alapműveletek elvégzésére többféle algoritmus is használatos, könyvünkben a legegyszerűbb változatokat, amelyek a kézi számításhoz leginkább hasonlóak, mutatjuk be.

1.Összeadás

A feldolgozások legfontosabb és legalapvetőbb aritmetikai művelete az összeadás, mivel a többi elemi művelet(kivonás, szorzás, osztás) is visszavezethető erre.

A helyiértékes számrendszerekben az összeadás helyiértékenként történik, így a kettes számrendszerben is. Két, egyjegyű bináris szám, A és B összeadása és az átvitel képzése az alábbi táblázat szerint történik. A táblázat sorai A és B lehetséges értékeihez adja meg az összeg(S) és az átvitel(C) értékét.

3-8.táblázat: *Két, egyjegyű bináris szám összeadása*

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ennek alapján, két fixpontos egész- és két fixpontos törtszám összeadása, a következő példák szerint történik. (A példákban az egyszerűség kedvéért 8 bites tárolóhelyeket tételezünk fel, amelynek legértékesebb bitje az előjelbit, a többi 7 bit szolgál a számok tárolására.)

A számok összeadásakor túlcsordulás léphet fel, ha az eredmény nagyobb, mint az adott tárolóhelyen ábrázolható legnagyobb szám; jelen példánk esetében az $1111111_2=127_{10}$ -es, illetve a $0,1111111_2 = 127/128_{10} = 0,9921..._{10}$ -es számérték.

	fixpontos egészként		fixpontos törtként	
	dec.	bináris	decimális	bináris
A operandus:	22_{10}	**0** 0010110	$0{,}6015_{10}$	**0** 1001101
B operandus:	90_{10}	**0** 1011010	$0{,}3359_{10}$	**0** 0101011
Eredmény:	112_{10}	**0** 1110000	$0{,}9375_{10}$	**0** 1111000

2.Kivonás

A kivonás művelete visszavezethető az összeadásra, ha a negatív számok tárolására a 3.2.1.b.pontban megismert komplemenskódot használjuk. Különböző okok miatt, a leggyakrabban a kettes komplemenskódot használják, ezért a következő példában is ezt mutatjuk be. A példában csak a fixpontos egészszámokkal végzett műveletet mutatjuk be, mivel a törtek esetében is ugyanaz az eljárás.

	decimális	bináris
A operandus:	99_{10}	**0** 1100011
B operandus:	-66_{10}	**1** 1000010
B komplemense:		**1** 0111110
Eredmény(A-B):	33_{10}	*1* **0** 0100001

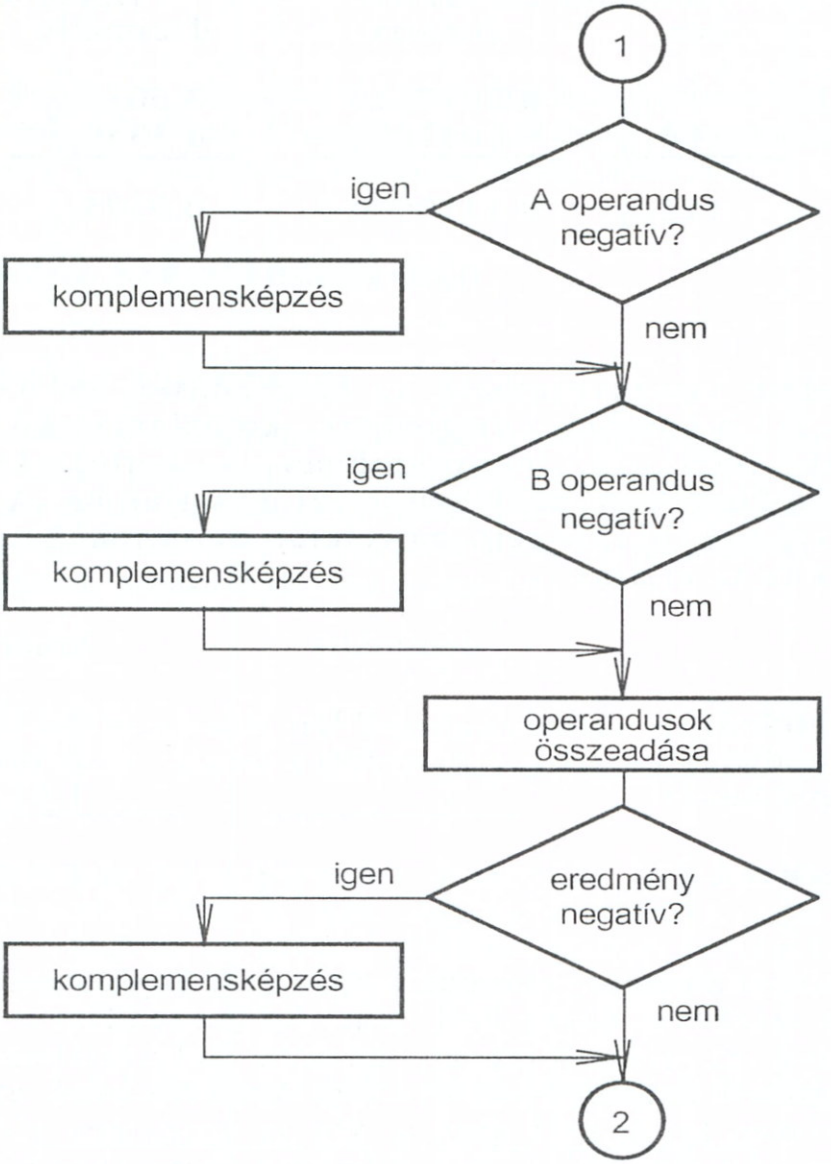
A két operandus kivonásakor, a negatív számot komplementálni kell, majd a másik operandus értékével össze kell adni. Az összeadást a tárolóhely teljes hosszán, beleértve az előjelbitet is, minden helyiértéken ugyanúgy kell elvégezni. A legfelső helyiértéken esetlegesen keletkező átvitelt(*1*), amely kicsordult, figyelmen kívül kell hagyni. Ha az eredmény előjele 1-es értékű lesz, azaz a szám negatív, akkor az eredményt kettes komplemens formában kapjuk meg, ezért azt vissza kell alakítani, vissza kell komplementálni.

Az összeadás és a kivonás azonos algoritmus alapján végezhető el, amelynek a folyamatábráját a 3-21.ábra mutatja be.

A különböző előjelű összeadandók esetében túlcsordulás nem léphet fel, de azonos előjelek esetén valódi túlcsordulás - amikor az eredmény nem fér be a tárolóhelybe - továbbra is felléphet.

3.Szorzás

A szorzás elvégzésére szolgáló algoritmusok közül a legegyszerűbbel foglalkozunk, azzal, amely sorozatos összeadással és léptetéssel - a regiszterek tartalmának eltolásával egy, vagy több helyiértékkel jobbra, vagy balra - állítja elő a szorzatot.

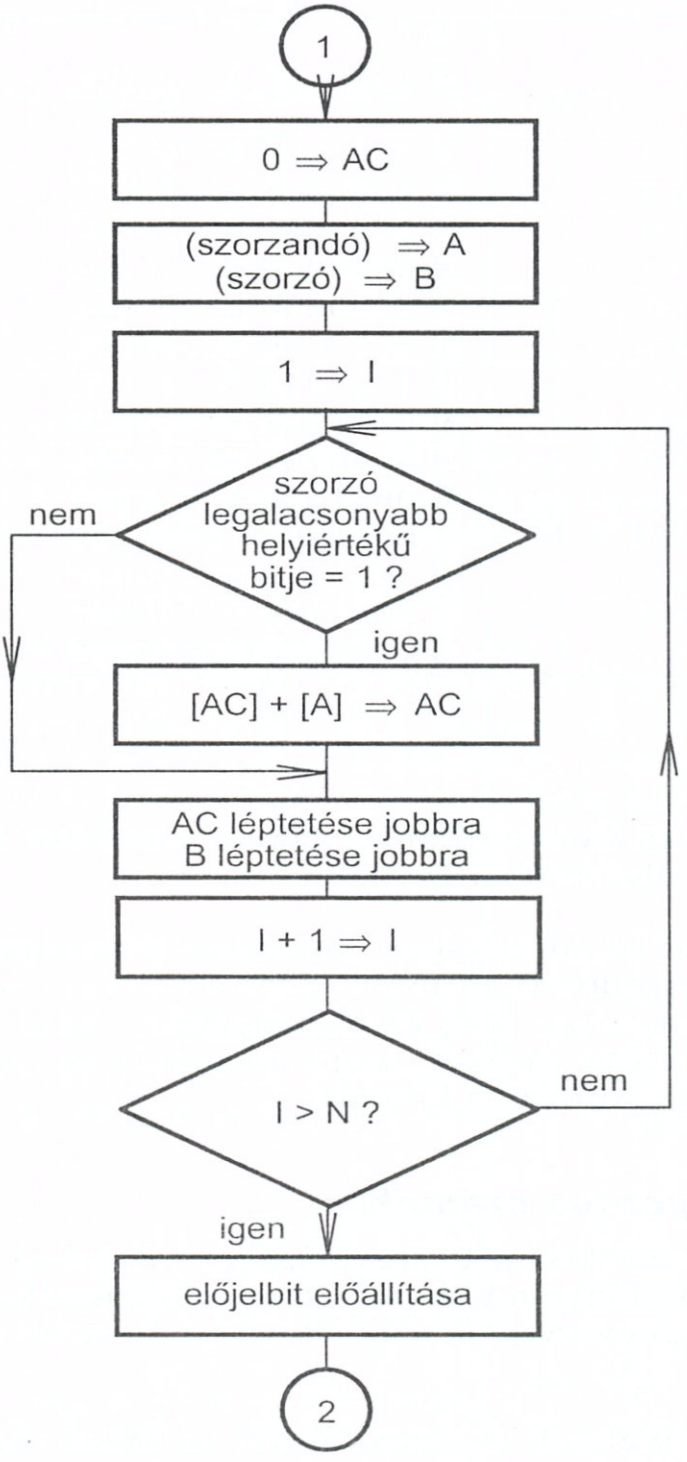


3-21.ábra: *Összeadás, kivonás folyamatábrája*

A szorzás elvégzésekor az eredmény hosszúsága kb. kétszerese az operandusok hosszának és ezért az eredményt befogadó akkumulátor(vagy más, erre a célra használt) regiszternek kétszeres hosszúságúnak kell lennie. (Az eredményt többnyire két regiszter együttesen fogadja be, így hozva létre a kétszeres tárolási hosszt.) Ezenkívül szükséges még egy további regiszter(B), amely a szorzót fogadja be. Célszerű a szorzandót is egy külön regiszterben (A) tárolni, mert ez a végrehajtást leegyszerűsíti.

Az adatok tárolási formáit vizsgálva, megállapítható, hogy a fixpontos törtszámkénti ábrázolás előnyösebb, mint az egészszámkénti ábrázolás, mert az eredményt befogadó akkumulátor regiszter felső helyiértékein rögtön megkapjuk az eredmény felhasználható, tárolható részét és az értéktelenebb helyiértékek vesznek el a további feldolgozásból. Egészszámkénti tároláskor ez nem ilyen egyszerű, mert az eredmény tárba írásakor értékes jegyek veszhetnek el. Lebegőpontos ábrázoláskor az előzőekben vázolt problémák nem me-

rülnek fel, mert *egyrészt* a mantissza törtszámként kerül ábrázolásra, *másrészt* az eredmény memóriába történő visszaírása előtt, azt még normalizálja a processzor, aminek következtében mindig a legértékesebb helyiértékek kerülnek a tárolóba. A szorzás folyamatábrája a 3-22.ábrán látható. (Az ábrában az N változó a szorzó helyiértékeinek számát jelenti.)



3-22.ábra: *Szorzás folyamatábrája*

A következő példa a szorzás lépéseinek részeredményeit mutatja be. Az eredményt a kétszeres méretű AC regiszter fogadja be, de megemlítendő, hogy az eredmény alacsony helyiértékű bitjeinek tárolására a kiürülő szorzó regiszter (B) is felhasználható lenne.

Lépések	Szorzandó(A) és eredmény(AC)	Szorzó(B)
Regisztertartalmak a kezdés előtt	**0** 1101010 **0** 0000000 00000000	**0** 1101000
1.lépés: léptetés előtt	**0** 0000000 00000000	**0** 1101000
1.lépés: léptetés után	**0** 0000000 00000000	**0** 0110100
2.lépés: léptetés előtt	**0** 0000000 00000000	**0** 0110100
2.lépés: léptetés után	**0** 0000000 00000000	**0** 0011010
3.lépés: léptetés előtt	**0** 0000000 00000000	**0** 0011010
3.lépés: léptetés után	**0** 0000000 00000000	**0** 0001101
4.lépés: léptetés előtt	**0** 1101010 00000000	**0** 0001101
4.lépés: léptetés után	**0** 0110101 00000000	**0** 0000110
5.lépés: léptetés előtt	**0** 0110101 00000000	**0** 0000110
5.lépés: léptetés után	**0** 0011010 10000000	**0** 0000011
6.lépés: léptetés előtt	**1** 0000100 10000000	**0** 0000011
6.lépés: léptetés után	**0** 1000010 01000000	**0** 0000001
7.lépés: léptetés előtt	**1** 0101100 01000000	**0** 0000001
7.lépés: léptetés után	**0** 1010110 00100000	**0** 0000000

4.Osztás

Az osztás algoritmusa bonyolultabb, mint a szorzásé és felépítése erősen függ az alkalmazott adattárolási formától.

Az osztás legegyszerűbb formájában, sorozatos kivonással, az ú.n. **visszaállító**(restoring) **algoritmussal** történik, azaz amennyiben a kivonás eredménye negatív - az osztó nincs meg az osztandóban - akkor vissza kell állítani az eredeti, kivonás előtti értéket, majd az osztót egy helyiértékkel jobbra léptetve, folytatható a sorozatos kivonás.

Műveletek lebegőpontos számokkal

A numerikus adatok tárolásának tárgyalásakor világossá vált a lebegőpontos adatábrázolási forma előnye a számításigényes feladatok megoldásánál. Ezért az alapműveletek legegyszerűbb megvalósítási formáit ebben az esetben is megnézzük.

Mivel a fixpontos műveletek végrehajtásáról az előző részben szó volt, ezért a lebegőpontos számok esetében a mantisszák, karakterisztikák összeadásával, kivonásával, szorzásával, osztásával nem foglalkozunk újra. Jelölje

$$A = m_a \cdot 2^{k_a}$$

$$B = m_b \cdot 2^{k_b}$$

a két lebegőpontos számot, ahol m_a, m_b a normalizált mantisszák és k_a, k_b a karakterisztikákat jelöli.

1.Összeadás, kivonás

A lebegőpontos számok összeadása, kivonása a fixpontos számokéhoz hasonló, csak valamivel összetettebb, mivel a hatványkitevős forma miatt, a karakterisztikákat előbb azonos értékre kell hozni.

Ha **a karakterisztikák megegyeznek**, azaz $k_a = k_b$, akkor

$$A + B = m_a \cdot 2^{k_a} + m_b \cdot 2^{k_b} = (m_a + m_b) \cdot 2^{k_a} = m_{(a+b)} \cdot 2^{k_{(a+b)}}$$

ahol

$$m_{(a+b)} = m_a + m_b \qquad \text{és} \qquad k_{(a+b)} = k_a$$

Tehát az eredmény karakterisztikája a közös, azonos karakterisztika értéke lesz, míg a mantisszákat össze kell adni és a kapott értéket normalizálni kell ismét. Ez egyúttal az új karakterisztikának a módosítását(csökkentését) is jelenti akkora számértékkel, mint ahány helyiértékkel balra kellett léptetni a mantisszát.

Ha **a karakterisztikák nem egyeznek meg**, azaz $k_a \neq k_b$, akkor a karakterisztikákat azonos értékre kell hozni. Ez azt jelenti, hogy valamelyik operandus mantisszájának jobbra léptetésével(a normalizált alak miatt balra nem léptethető a túlcsordulás miatt), azaz 2-vel történő osztásával és a karakterisztika értékének növelésével a két karakterisztikát azonos értékre hozzuk. Ennek alapján megállapítható, hogy a kisebb kitevőjű szám mantisszáját kell léptetni.

Az eredmény karakterisztikája:

$$k_{(a+b)} = \max\{k_a, k_b\}$$

Ha, tegyük fel, $k_a > k_b$, és így $n = k_a - k_b$, akkor a B operandus mantisszáját osztani kell a

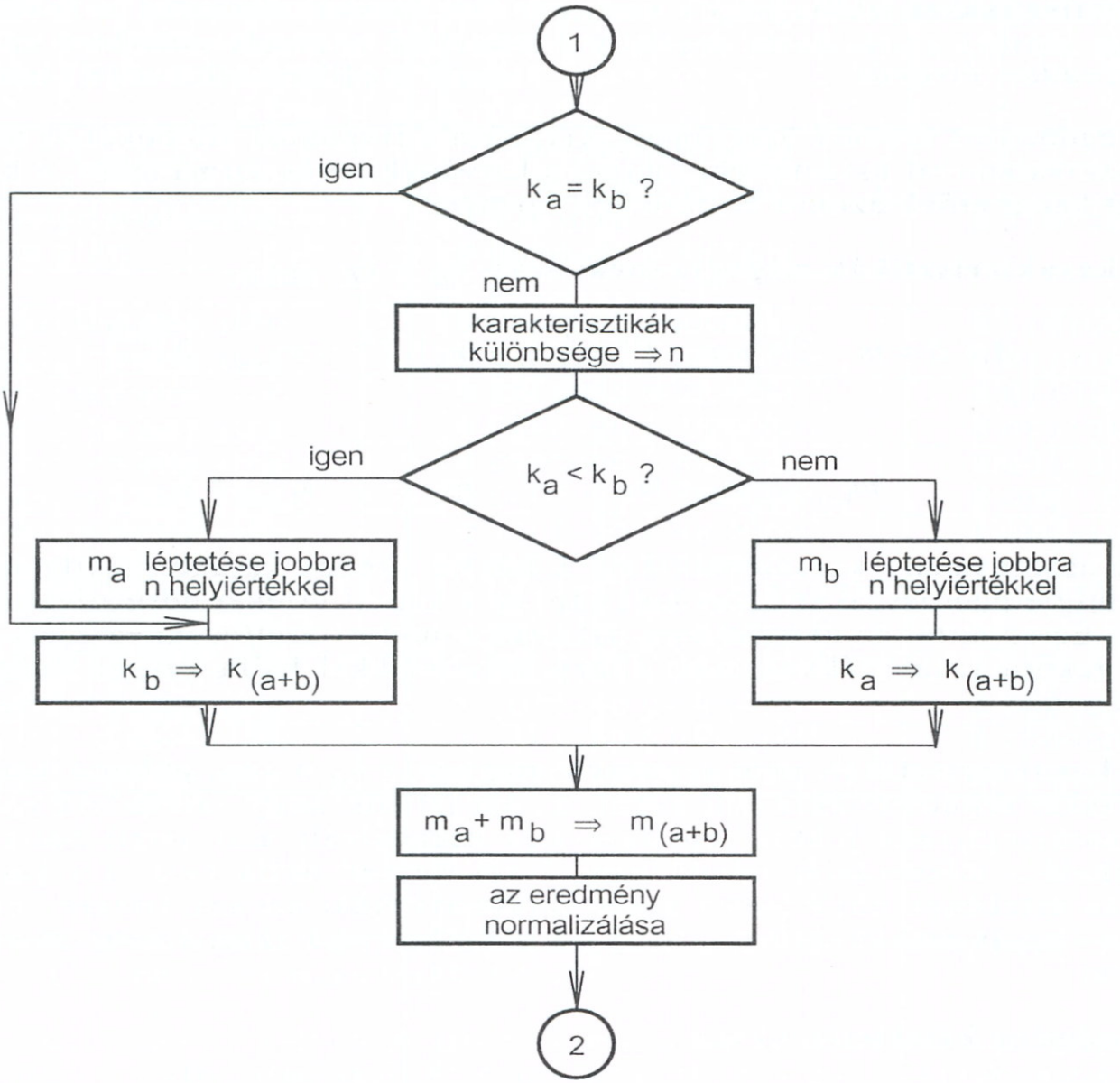
$$2^n = 2^{k_a - k_b}$$

értékkel és a karakterisztikát ennek megfelelően növelni kell, azaz

$$m_{bu} = \frac{m_b}{2^n} = \frac{m_b}{2^{(k_a - k_b)}} = m_b \cdot 2^{-(k_a - k_b)}$$

és

$$B = m_{bu} \cdot 2^n \cdot 2^{k_b} = m_{bu} \cdot 2^{(k_a - k_b)} \cdot 2^{k_b} = m_{bu} \cdot 2^{k_a}$$



3-23.ábra: *Lebegőpontos számok összeadása, kivonása*

Így, ekkor az összegre azt kapjuk, hogy

$$A + B = m_a \cdot 2^{k_a} + m_b \cdot 2^{k_b} = m_a \cdot 2^{k_a} + m_{bu} \cdot 2^{k_a} = (m_a + m_{bu}) \cdot 2^{k_a} = m_{(a+b)} \cdot 2^{k_{(a+b)}}$$

ahol

$$m_{(a+b)} = m_a + m_{bu} \qquad \text{és} \qquad k_{(a+b)} = k_a$$

Az összeadás és a kivonás együttes folyamatábrája látható 3-23.ábrán.

2.Szorzás, osztás

A szorzás és az osztás végrehajtásakor a mantisszákkal a műveletvégzés a fixpontos számokéval megegyező módon történik, ugyanakkor a karakterisztikák összeadásra, vagy kivonásra kerülnek.

b.)Műveletek tízes számrendszerben

Tízes számrendszer szerinti műveletvégzés kissé bonyolultabb, mint a sima kettes számrendszerbeli műveletvégrehajtás és az aritmetikai egység felépítése is eltér a kettesben dolgozóétól.

A műveletek közül csak az összeadást mutatjuk be BCD kódú számokkal. Amint az az adatábrázolási formák tárgyalásánál már ismertté vált, egy-egy decimális számjegyet bináris formában legalább 4-4 helyiértéken(tetrádon) kell kifejezni. BCD kódban a tízes számrendszerbeli számjegyeknek, azok kettesbeli értéke felel meg, kiegészítve mindegyik jelsorozatot 4 bitre.

A műveletvégzés a tetrádon belül kettes számrendszer szabályai szerint történik, míg a tetrádok között a tízes számrendszer szerint kell képezni az átvitelt. Az eredmény előjelének megállapítása külön történik.

A kivonást tízes rendszeren belül is el lehet végezni komplemens kódban, mégpedig 9-es, vagy 10-es komplemensben.

Két BCD kódú szám összeadását a következő példa segítségével mutatjuk be.

	Dec. szám	BCD szám			előjel
1.operandus:	+357	0011	0101	0111	1100
2.operandus:	+218	0010	0001	1000	1100
részeredmény:		0101	0110	1111	
átvitel:					
korrekció(+6):				0110	
átvitel(mod16):			1 ↵		
eredmény:	+575	0101	0111	0101	1100

A BCD kódú műveletvégzésnél, ha az eredmény számjegy nagyobb 9-nél, akkor korrigálni kell, azaz csak a 10-zel csökkentett érték írható le. Ezt az értéket kapjuk meg, ha a 9-nél nagyobb számokhoz 6-ot hozzádva, képezzük az eredményt(mod16 szerinti maradékot) és az átvitelt, azaz csak a 15 feletti értéket őrizzük meg mint eredményt és átvitelt képezünk a következő helyiértékre.

3.3.2.Logikai műveletek

A számítástechnikában fontos szerepe van a logikai műveleteknek, a logikai algebrának; mind a számítógépek tervezésekor, mind az alkalmazások során.

A logikai algebra állításokkal, illetve azok kapcsolatával foglalkozik. Jelölje A, B és C a következő mondatokban lévő állításokat:

A := 'Süt a nap.'

B := 'Van pénzem.'

C := 'Elutazom nyaralni.'

Minden állítás vagy igaz, vagy hamis; tehát két értéket vehet fel:

- lehet IGAZ(TRUE) értékű és
- lehet HAMIS(FALSE) értékű.

Az előbbi állításokkal a következő egyszerű összefüggéseket írhatjuk le:

1. Elutazom nyaralni, ha nincs pénzem.
2. Elutazom nyaralni, akár ha süt a nap, akár ha van pénzem.
3. Elutazom nyaralni akkor, ha süt a nap és van pénzem.

Ezeket a mondatokat le tudjuk írni a logikai alapműveletek segítségével is:

1. $C = \overline{B}$
2. $C = A + B$
3. $C = AB$

A bemutatott három alapművelettel(NEM-, VAGY-, ÉS-művelet) tetszőleges, összetett logikai kifejezés is leírható.

a.)Alapműveletek és törvények

Az alapműveletek és törvények tartalmának értelmezéséhez, azt kell áttekinteni, hogy az egyes műveletek milyen logikai feltételek mellett, milyen eredményeket adnak. Ennek egyszerű kezelésére, a számítástechnikai gyakorlatban szokásos összerendeléseket alkalmazzuk, azaz a logikai IGEN(igaz, TRUE) értékhez az 1-est, a logikai NEM(hamis, FALSE) értékhez a 0-át rendeljük hozzá. Az egyes műveletek, kifejezések értelmét, tartalmát az ú.n. 'igazságtáblázat' segítségével mutatjuk be.

Az 'igazságtáblázat' baloldala az egyes változók lehetséges értékkombinációit sorolja fel, míg a jobboldalán az eredményváltozóknak ezekhez a kombinációkhoz tartozó logikai értékeit találjuk meg.

Alapműveletek

Az alábbiakban, először az említett három alapművelet, a NEM(NOT)-, a VAGY(OR)- és az ÉS(AND)-művelet műszaki gyakorlat szerinti és matematikai jelölésmódját adjuk meg, majd táblázatosan az egyes műveletek 'igazságértékeit'.

A NEM-művelet egyoperandusos művelet, míg a VAGY- és az ÉS-művelet kétoperandusos.

	NEM-művelet	VAGY-művelet	ÉS-művelet
műszaki írásmód	$C=\overline{A}$	$C=A+B$	$C=AB$
matematikai írásmód	$C=\neg A$	$C=A \vee B$	$C=A \wedge B$

Az említett műveletek igazságtáblázata található a 3-9.táblázatban.

3-9.táblázat: *Alapműveletek igazságtáblázata*

operandusok		NEM művelet		VAGY művelet	ÉS művelet
A	B	$C=\overline{A}$	$C=\overline{B}$	$C=A+B$	$C=AB$
0	0	1	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	1

A számítógépes alkalmazásokban, ahogy azt már említettük, a logikai műveletek mindig bitenként értelmezendők, azaz egy tetszőleges tárolóhely tartalmán végzett logikai művelet, a tárolóhely minden egyes bitje esetében ugyanúgy kerül végrehajtásra.

A *NEM(NOT)-műveletet*, mint egyoperandusos műveletet a tárolóhelyek tartalmának ellenkező értékre állítására(0-t 1-esre, 1-est 0-ra) lehet felhasználni, ami nem más mint az adott érték egyes komplemenskódú értéke.

A *VAGY(OR)-művelet* értelmezése nem azonos a köznapi szóhasználat szerinti értelmezéssel, az egymást kizáró feltételek vagylagos kezelésével, mert a két állítás(feltétel) egyidejű 'igaz' volta is megengedett. A gépi gyakorlatban két tárolóhely tartalom 'összeadására', egyesítésére használják, azaz az eredményben azokon a biteken lesz 1-es érték, amely helyiértékeken bármelyik, vagy esetleg mindkét operandusban 1-es volt.

Az *ÉS(AND)-művelet* akkor ad 'igaz' értéket, ha mindkét állítás(feltétel) egyidejűleg teljesül. A művelet segítségével lehet kijelölni, 'maszkolni' egy tárolóhely tartalmának tetszőleges részét, amit az eredménybe átviszünk. Ezzel lehet megoldani a bitenkénti adatkezelést is, ha szükséges és nem áll rendelkezésre erre a célra szolgáló bitmanipulációs utasítás.

Alaptörvények

A továbbiakban, az összetett logikai kifejezések kiértékelésekor, illetve azokkal való műveletvégzéskor szükséges legfontosabb alaptörvényeket soroljuk fel. Ezekre az összefüggésekre különösen a bonyolultabb logikai kifejezések egyszerűsítésekor van szükség. A felsorolt azonosságok igaz volta az 'igazságtáblázat'-ok segítségével ellenőrizhető.

1. $A+\overline{A}=1$
2. $A\overline{A}=0$
3. $\overline{\overline{A}}=A$ kettős tagadás, azaz a tagadás tagadása maga az állítás
4. $\overline{A+B}=\overline{A}\cdot\overline{B}$
5. $\overline{AB}=\overline{A}+\overline{B}$

   } De - Morgan azonosságok
6. $A+0=A$
7. $A\cdot 0=0$
8. $A+1=1$
9. $A\cdot 1=A$
10. $A+AB=A(1+B)=A$

b.)Összetett műveletek és logikai függvények

A három alapműveleten kívül, néhány további - az előzőekből felépíthető - összetettebb műveletet szokás önálló műveletként kezelni és külön névvel illetni. A számítástechnikában az alábbiak tekinthetők a legfontosabbaknak:

- ekvivalencia(azonosság) művelete(jele: $\odot$, illetve $\equiv$),
- antivalencia művelete(jele: $\oplus$, illetve $\not\equiv$),
- NEM-VAGY(NOR) művelet(Pierce-művelet),
- NEM-ÉS(NAND) művelet(Sheffer-művelet).

Az első kettőnek a feldolgozásokban van fontos szerepe - tárolóhelyek tartalmának összehasonlításakor -, míg a második kettő a gép tervezésekor, az áramkörök leírásakor játszik szerepet, mivel a megvalósítható alapáramkörök ezek szerint működnek. Ezért szokás a NEM-, a VAGY- és az ÉS-műveletek helyett a NEM-, NEM-VAGY-, NEM-ÉS-műveletekből felépíteni a leíró kifejezéseket. Ez utóbbi három művelet segítségével ugyanúgy egyértelműen leírható egy kifejezés, mint az előző pontban megismert három alapművelettel.

A felsorolt négy művelet igazságtáblázata az alábbiakban látható:

3-10.táblázat: *Összetett műveletek igazságtáblázata*

A	B	ekvivalencia	antivalencia	NEM-VAGY	NEM-ÉS
		$A \odot B$	$A \oplus B$	$\overline{A+B}$	$\overline{AB}$
0	0	1	0	1	1
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	0	0

Megjegyzendő, hogy az antivalencia művelete valósítja meg a köznapi értelemben használt ***vagy*** szó tartalmát, ezért ezt KIZÁRÓ-VAGY műveletnek is nevezik.

Az alapműveletekkel a táblázat összetett kifejezései a következőképp néznek ki:

- ekvivalencia: $C = \overline{A} \cdot \overline{B} + A \cdot B$
- antivalencia: $C = \overline{A} \cdot B + A \cdot \overline{B}$
- NEM-VAGY: $C = \overline{A+B} = \overline{A} \cdot \overline{B}$
- NEM-ÉS: $C = \overline{AB} = \overline{A} + \overline{B}$

Logikai függvények

A logikai függvények a logikai műveletekkel felírt kifejezéseket jelentik, amelynek értéke a kifejezést alkotó logikai változók értékétől függ. A logikai függvények hasznát és felépítését az alábbi példa segítségével mutatjuk be.

> *Határozzuk meg egy 1 bites (teljes) összeadó összeg-kimenetének(S) és átvitel-kimenetének(C) logikai függvényét, ha az összeadásban az előző helyiértékről származó átvitelt(D) is figyelembe vesszük.*
>
> Jelölje a két összeadandót A és B, az előző helyiértékről származó átvitelt D; az összeget S és a keletkező átvitelt C. Ekkor figyelembe véve A, B és D összes lehetséges értékét, az alábbi igazságtáblázatot írhatjuk fel:

3-11.táblázat: *Teljes összeadó igazságtáblázata*

A	B	D	S	C
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	1	0	0	1
1	1	1	1	1

Tehát az S oszlopa tartalmazza azokat az eseteket, amikor az A+B+D összeg értéke 1_2, vagy 11_2 lesz; C oszlopa pedig azokban az esetekben tartalmaz 1-est, amikor az A+B+D összegzés eredményeként átvitel keletkezik, azaz 10_2 és 11_2 esetében. Ezek alapján felírható a két kimenet logikai függvénye:

$$S=\overline{A}\cdot\overline{B}\cdot D+\overline{A}\cdot B\cdot\overline{D}+A\cdot\overline{B}\cdot\overline{D}+A\cdot B\cdot D=(A\ B)\cdot D+(A\oplus B)\cdot\overline{D}=(A\oplus B)\oplus D$$

$$C=\overline{A}\cdot B\cdot D+A\cdot\overline{B}\cdot D+A\cdot B\cdot\overline{D}+A\cdot B\cdot D=A\cdot B+(A\oplus B)\cdot D$$

A végeredményeket a lehetséges összevonások figyelembevételével kaptuk meg.

A logikai függvények fő alkalmazási területe az egyes elketronikus elemek, logikai áramkörök működésének leírásakor van, amely függvények alapján a konkrét áramköri kapcsolások könnyen összeállíthatók.

A logikai műveletek megfelelői a gépi utasításkészletben is megjelennek, általában az alábbi lehetőségeket biztosítva:

- **ÉS**(AND)-kapcsolat két tárolóhely tartalma között, maszkoláshoz,
- **VAGY**(OR)-kapcsolat két jelsorozat összefésüléséhez, egyesítéséhez,
- **antivalencia**(EXOR)-kapcsolat, vagy **ekvivalencia**-kapcsolat két tárolóhely-tartalom különbözőségének, egyezőségének vizsgálatához,
- léptetések, túlcsordulások, stb. figyeléséhez.

3.3.3.Aritmetikai egység(ALU)

A mikroprocesszorok aritmetikai-logikai egysége(ALU), mint a műveletvégzés eszköze, a legegyszerűbb esetben is néhány fő részegységet biztosan magában foglal. Ezek a következők:

- összeadó-egység, amely két operandus összeadására szolgál,
- léptető áramkörök, amelyek a regiszterek tartalmát műveletvégzés közben jobbra, vagy balra léptetik, azaz tulajdonképpen osztják, vagy szorozzák azt,
- logikai áramkörök a logikai műveletek megoldásához,
- regiszterek, az adatok ideiglenes tárolására; ezek lehetnek az ALU részét képező, kizárólagos használatú regiszterek, mint pl. többnyire az akkumulátor regiszter(AC), vagy a processzor általános célú regiszterei közül egy, vagy több.

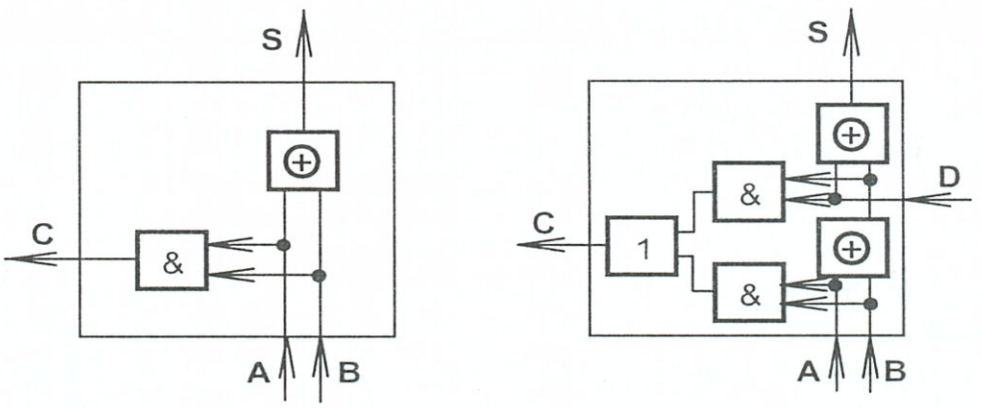
Az aritmetikai műveletek végrehajtásakor, az eredménytől függően, az állapotregiszter egyes jelzőbitjeit(flag-jeit) a processzor beállítja. A műveletek eredményét visszatükröző legfontosabb jelzőbitek a következők:

- átvitel(**carry**) jelzőbitje: ha az eredmény legmagasabb helyiértékén átvitel keletkezik, 1-es értéket vesz fel;
- nulla(**zero**) jelzőbitje: ha az eredmény nulla értékű, 1-es értéket vesz fel;
- előjel(**sign**) jelzőbitje: ha az eredmény negatív, akkor az értéke 1-es lesz;
- túlcsordulás(**overflow**) jelzőbitje: ha az eredmény nagyobb, mint a tárolható legnagyobb érték, akkor értéke 1-es lesz.

Az aritmetikai egység minimálisan a következő műveletek közvetlen elvégzésére('huzalozott' módon) alkalmas: **A+B**, **A·B**, $\overline{\mathbf{A}}$. Mint, ahogy ez az előző 3.3.1. és 3.3.2.pontokban már tárgyalásra került, ezekkel az alapműveletekkel a többi aritmetikai és logikai művelet is elvégezhető.

Két bináris számjegy összeadásának 3-8.táblázata, illetve az előző helyiértékről származó átvitelt is figyelembe vevő ú.n. teljes összeadó táblázata (3-11.táblázat) alapján, felvázolhatók az ezek megvalósítására szolgáló egységek(félösszeadó, teljes összeadó) logikai kapcsolási vázlatai, amelyek a 3-24.ábrán láthatók.

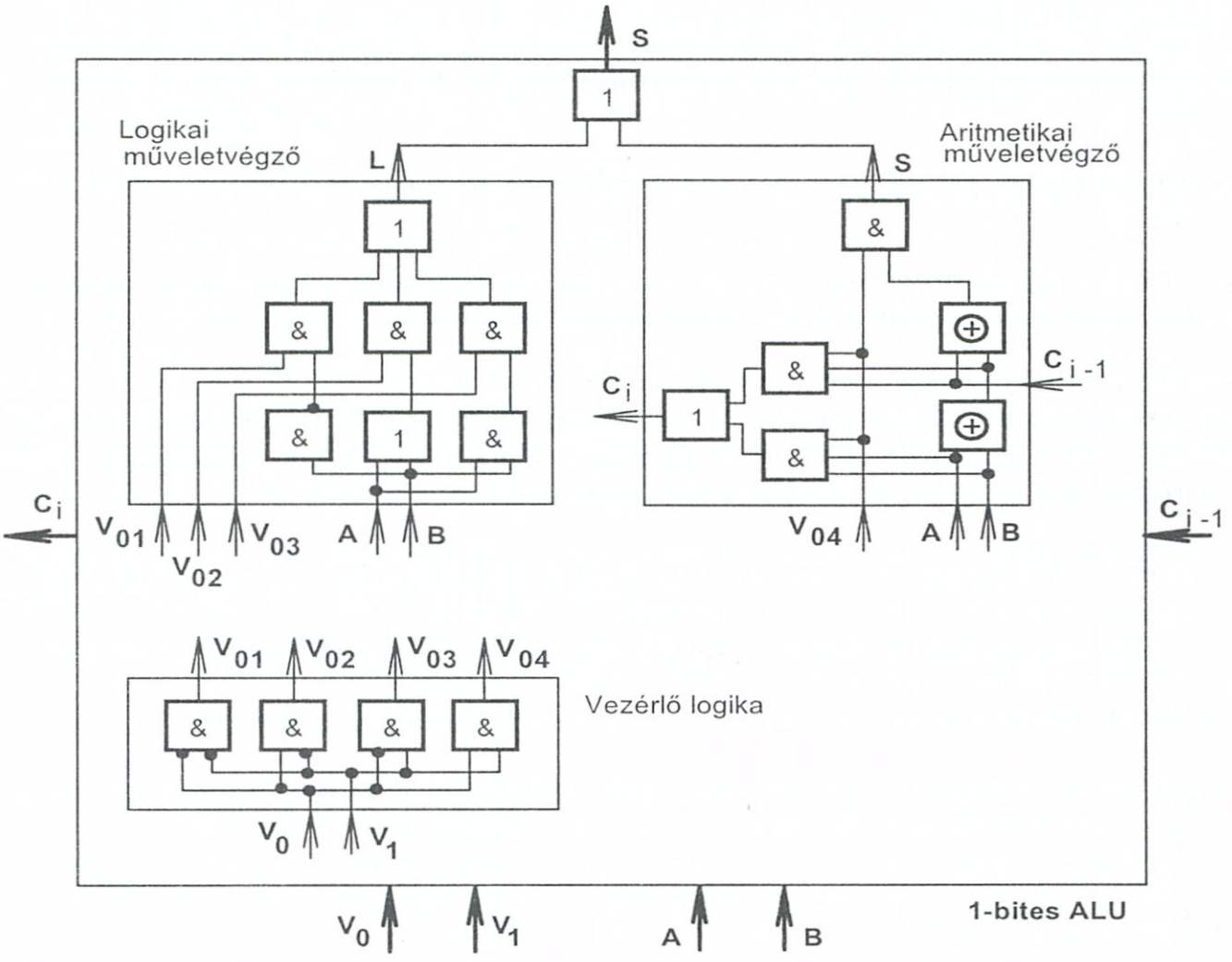
A logikai műveletek végrehajtására is alkalmas 1 bites, elemi aritmetikai egységet mutatja be a 3-25.ábra, amely az összeadón, a logikai műveletvégzőn kívül egy vezérlő-dekódoló részt is tartalmaz, amely utóbbi szolgál a vezérlő jelek feldolgozására. Ahhoz, hogy az elvégezni kívánt művelet kiválasztható legyen, két vezérlő jel, V_0, V_1 szükséges.



a.) félösszeadó (3-9.táblázat szerint)

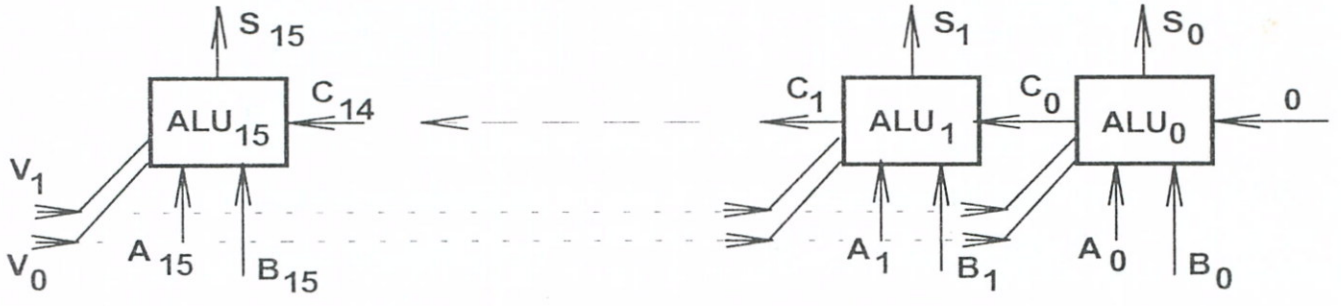
b.) teljes összeadó (3-12.táblázat szerint)

3-24.ábra: *Összeadók logikai vázlata*



3-25.ábra: *1-bites ALU felépítése*

Ilyen elemi, egybites teljes összeadókból épülhet fel egy teljes szó(16-32 bit) összeadására szolgáló összeadó egység. Ennek vázlatát mutatja be a 3-26.ábra.



3-26.ábra: *16-bites összeadó egység vázlata*

3.4.UTASÍTÁSVÉGREHAJTÁS, VEZÉRLŐ EGYSÉG

A processzor működésének lényegét a programutasítások feldolgozása, végrehajtása adja. Ennek a folyamatnak a kellő mélységű megismerése vezet el a számítógép működésének teljes értékű megértéséhez. Az utasításvégrehajtás lépésekre történő felbontása egyrészt lehetővé teszi az annak a vezérlésére felhasznált mikroprogramozás lényegének megértését, másrészt megadja az utasítások **átlapolt végrehajtásának** (pipelining - 3.5.pont) **magyarázatát** is.

3.4.1.Utasításvégrehajtás lépései

Korábban már több ízben részleteztük az utasítások feldolgozási lépéseit a számítógépek működésének általános bemutatásakor(1.2. és 2.1.2. pontokban). Az utasítások elemi lépésekre bontása után megvizsgáljuk azt, hogy milyen strukturális változásokat jelent a feldolgozás soros, vagy párhuzamos módú megvalósítása, megközelítése.

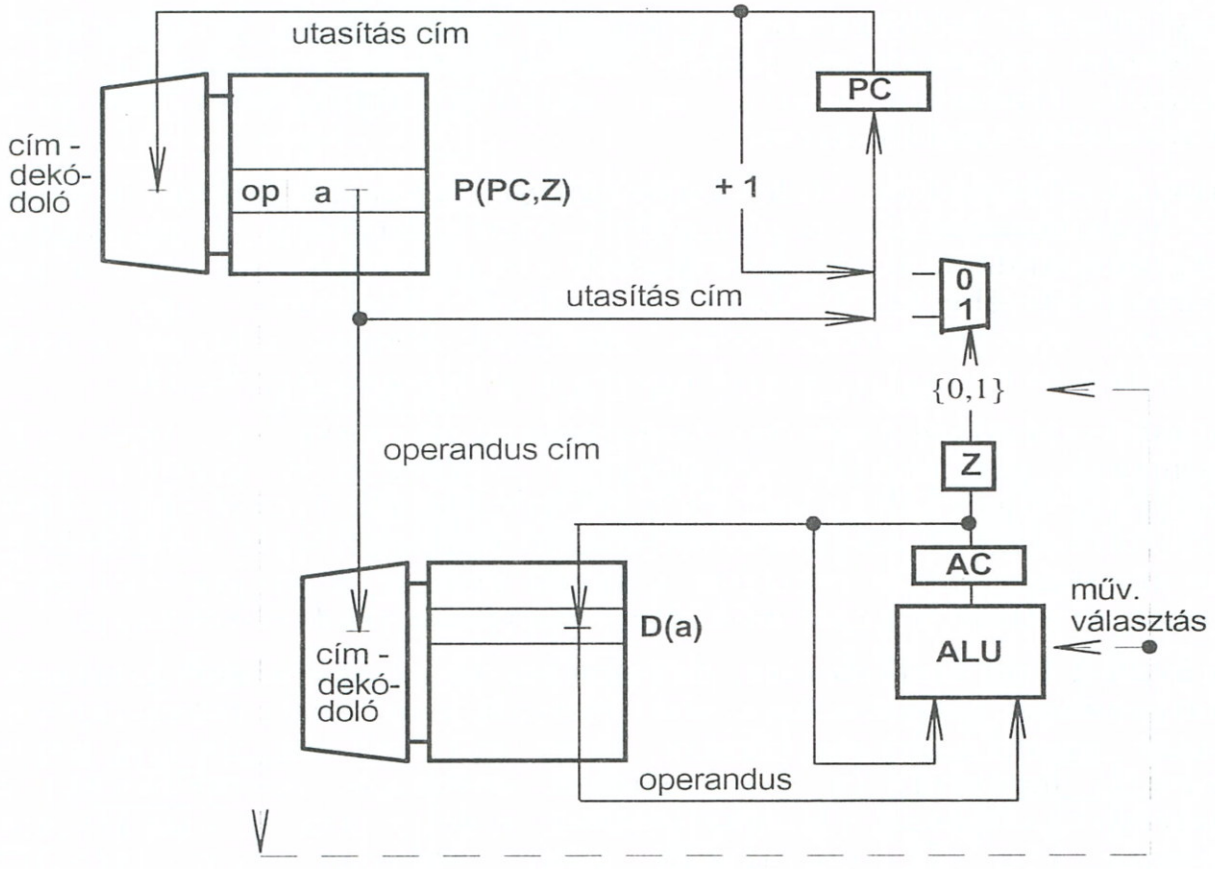
Egy-egy utasítás feldolgozása az utasítás előkeresését, értelmezését és az előírt feladat végrehajtását jelenti lényegében. A végrehajtási folyamatot funkcionális lépésekre bontva, egy utasítás feldolgozása alapvetően az alábbi részekből tevődik össze:

- **utasításelőkészítés, -lehívás(fetching)**: ebben a fázisban a processzor az utasításszámláló regiszter(PC) tartalmát(amely a soronkövetkező végrehajtandó utasítás memóriabeli címe) átviszi - ha van-, a memória címregiszterébe(MAR=Memory Address Register) vagy a címsínt vezérlő cím-pufferregiszterbe, amelynek alapján kikeresi a tárból az utasítást és átviszi a vezérlő egység utasításregiszterébe(IR),
- **utasításszámláló regiszter tartalmának növelése**: ez a lépés a PC tartalmának automatikus növelésével a következő utasítás tárolóbeli helyének címét állítja elő; ez a PC tartalmának 1-gyel (pontosabban az utasításhossznak megfelelő tárolóhely számmal) való növelését jelenti; az utasításszámláló regiszter a kezdő értékét, a program kezdetének a címét, kívülről, a gépet működtető operációs rendszertől kapja,
- **műveleti kód értelmezése, dekódolása és az operandus címének meghatározása**: ezalatt a fázis alatt történik a műveleti jelrész értelmezése (azaz annak a meghatározása, hogy mit kell csinálnia a gépnek az utasítás hatására) és a művelethez szükséges operandus(ok) pontos helyének, címének a meghatározása, kidolgozása az utasítás címrésze alapján; ha az utasításban előírt művelet nem igényli adat felhasználását(pl. vezérlő utasítások), akkor a végrehajtási fázisra tér át;
- **művelethez szükséges adat(ok) előkészítése**: a művelet elvégzéséhez, az előző lépésben kidolgozott cím alapján, kikeresi az operandus(oka)t a memóriából és átviszi az utasítás által meghatározott helyre, amely

legtöbbször az aritmetikai egység akkumulátor regisztere(AC), vagy más speciális regiszter;

- **végrehajtás(executing)**: ebben a fázisban történik a kijelölt művelet végrehajtása a kijelölt és előkészített operandusokkal; ha a művelet arra ad előírást, hogy a program végrehajtása ne a soronkövetkező, hanem egy másutt elhelyezkedő utasítással folytatódjék, akkor ebben a lépésben ennek a helynek a címét a gép beírja az utasításszámláló regiszterbe(PC-be);
- **az eredmény elhelyezése**: az eredményt, ha volt, az előírt helyre helyezi (ez többnyire az akkumulátor regiszter(AC)) és visszatér az első lépésre, újrakezdeni az utasításfeldolgozás lépéseit.

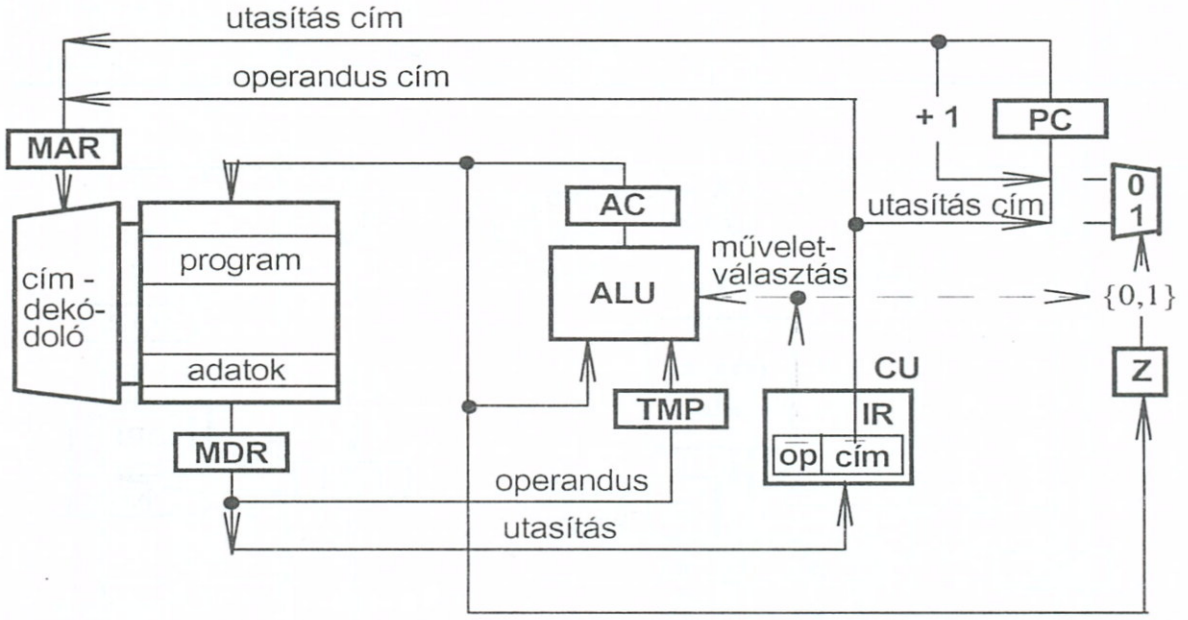
A felvázolt utasításvégrehajtási folyamat logikai vázlatát mutatják be a következő ábrák. Az első ábra(3-27.ábra) azt az ideálisnak tekinthető változatot mutatja be, amelynél az utasítás előkészítése, az adatok előkészítése és a művelet végrehajtása gyakorlatilag **párhuzamosan** történik, azt lehet mondani, hogy egy utasításfeldolgozási ciklusban történik az utasítás teljes feldolgozása.



3-27.ábra: *Párhuzamos szervezésű utasításfeldolgozás*

Az ilyen értelmű párhuzamosított utasításfeldolgozás feltétele a külön program- és adattároló léte, továbbá szükséges a két tárolóhoz két külön sínrendszer is. Amennyiben nincs külön program- és adattároló (Neumann-elvű számítógép), akkor ugyanabból a tárolóból kell előkeresni az utasítást és az adato(ka)t is. Ez csak egymást követően, **sorosan** történhet és ennek felel meg a második ábra(3-28.ábra) logikai vázlata.

Az ábrákon, a Z bit által vezérelt kétállapotú dekódoló a feltételes elágaztatás logikai megoldását mutatja be. Z értékétől függően, a PC tartalmának vagy automatikus növelése, vagy az utasításbani címmel való feltöltése következik be.

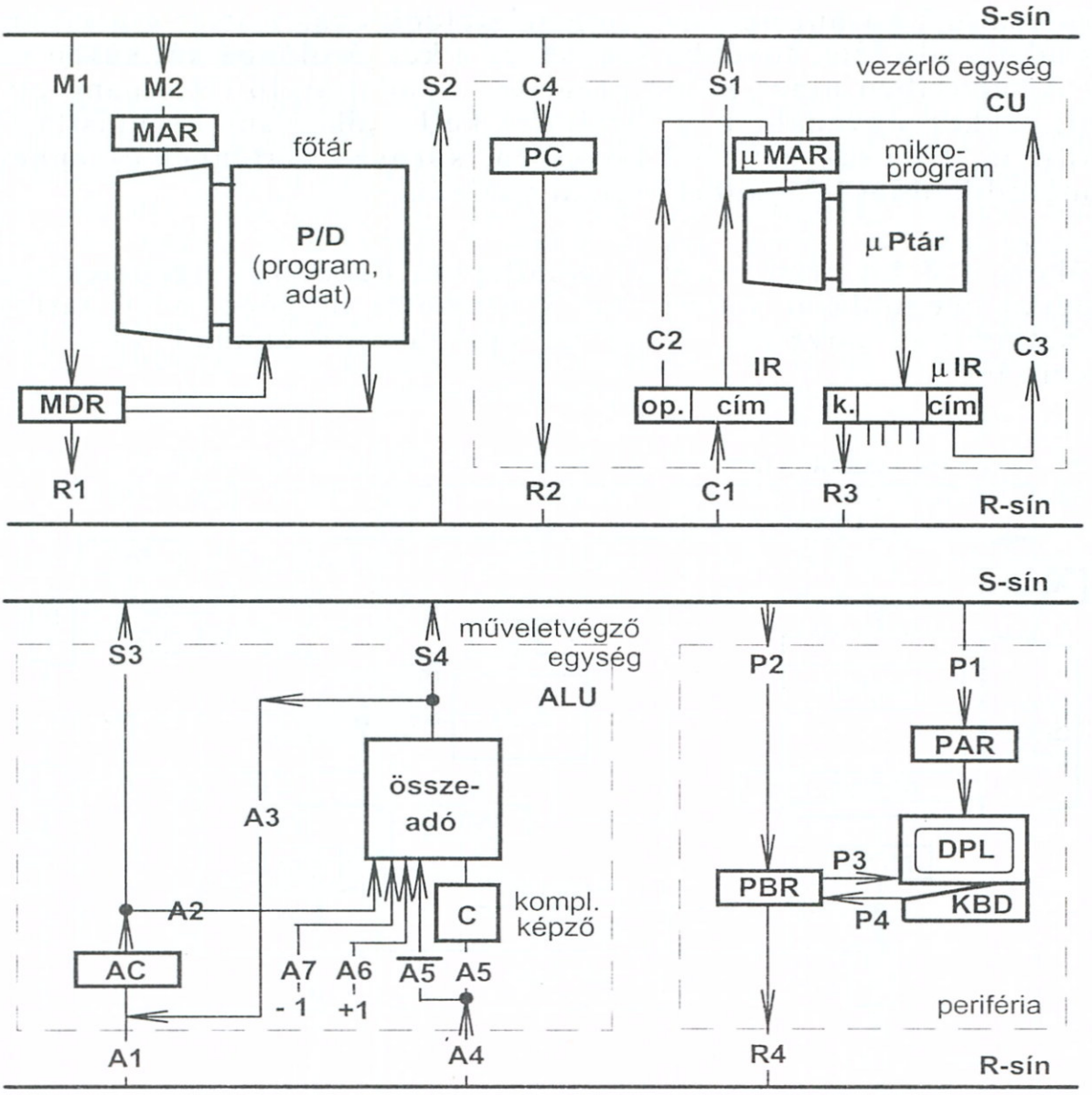


3-28.ábra: *Soros szervezésű utasításfeldolgozás*

3.4.2.Műveleti vezérlés

Az utasítások végrehajtásának ismertetett lépései, a működés szempontjából, további elemi műveletek ütemezett egymásutániságát jelentik. Ezek az elemi lépések adatútvonalak megnyitását, vagy lezárását (kapuzását), bizonyos állapotok beállítását, vagy törlését eredményezik (3-29.ábra). A műveleti jelrész alapján történő vezérelhetőség feltétele:

- a lépésenkénti vezérlési és választási lehetőség,
- az adatutak engedélyezési és tiltási lehetősége.



3-29.ábra: *Műveleti vezérlés alapproblémája*

a.)Alapprobléma

Egy-egy utasítás elemi lépéseinek vezérlése a gép igen sok pontjának egyidejű, vagy valamilyen sorrendben vezérelt működtetését jelenti. A számítógép igen primitív, elvi kapcsolási vázlatát mutatja be a 3-29.ábra, a legfontosabb kapcsolási(vezérlési) pontokkal. Ezek azok a pontok, amelyek megfelelő vezérlésével, az utasítás műveleti jelrésze által előírt feladat elvégezhetővé válik.

A 3-29.ábra feltüntetett vezérlési pontjait és azok funkcióját, azaz azt, hogy mi történik az adott kapu működtetésekor, foglalja össze a 3-12.táblázat. (Az ábrán az S-sín a tárolóba írásra(Store), az R-sín a tárolóból olvasásra(Read) szolgáló sínt jelöli.)

3-12.táblázat: *Vezérlési pontok és azok funkciója*

vezérlési pont	funkció		
R1	MDR	⇒	R-sín
R2	PC	⇒	R-sín
R3	konstans	⇒	R-sín
R4	PBR	⇒	R-sín
S1	IR(címrész)	⇒	S-sín
S2	R-sín	⇒	S-sín
S3	AC	⇒	S-sín
S4	összeadó	⇒	S-sín
C1	IR	⇐	R-sín
C2	IR(op.kód)	⇒	µMAR
C3	µIR(címrész)	⇒	µMAR
C4	PC	⇐	S-sín
M1	MDR	⇐	S-sín
M2	MAR	⇐	S-sín
A1	AC	⇐	R-sín
A2	AC	⇒	összeadó
A3	összeadó	⇒	AC
A4	összeadó	⇐	R-sín
A5	kompl.képző		
A6	+1	⇒	összeadó
A7	-1	⇒	összeadó
P1	PAR	⇐	S-sín
P2	PBR	⇐	S-sín
P3	PBR	⇒	DPL
P4	PBR	⇐	KBD

Az elemi műveletek és a vezérlési pontok kapcsolatának bemutatására egy nagyon egyszerű példát használunk fel.

Legyen a végrehajtandó utasítás egy töltő utasítás, amely a memória egy tárolóhelyének(X) tartalmát viszi át az ***akkumulátor*** *regiszterbe (AC). Az utasítás egycímes szerkezetű és a következő formájú:*

LDA X *és hatása* ***[X] ⇒ AC***

Nagyon leegyszerűsített formában, az utasításvégrehajtás elemi lépéseit és az érintett vezérlési pontokat az alábbi táblázat tartalmazza:

3-13.táblázat: *LDA X utasítás elemi lépései*

elemi lépések	érintett vezérlési pontok	megjegyzések
[PC] ⇒ R-sín ⇒ S-sín ⇒ MAR	R2, S2, M2	utasításcím átvitele
címdekódolás		
[mem(MAR)] ⇒ MDR		utasítás kikeresése
[MDR] ⇒ R-sín ⇒ IR	R1, C1	utasítás átvitele
[IR(op.kód)] ⇒ μMAR	C2	művelet értelmezése
[IR(címrész)] ⇒ S-sín ⇒ MAR	S1, M2	operanduscím átvitele
címdekódolás		
[mem(MAR)] ⇒ MDR		operandus kikeresése
[MDR] ⇒ R-sín ⇒AC	R1, A1	operandus átvitele

Az elemi műveletek vezérlésére(amelyek a programozó számára nem elérhetőek) kétféle lehetőség kínálkozik:

- **Huzalozott(hardver) módon**, amely esetben az utasítás feldolgozását jelentő elemi tevékenységek sorrendjének vezérlését bonyolult sorrendi, kombinációs áramkörrel oldják meg, amely a szükséges sorrendben és időzítéssel adja ki az egyes kapuk működtetésére szolgáló vezérlő jeleket. Ez ugyan gyors működést eredményez, de megvalósítása merev és költséges megoldású.

  A bonyolult felépítést ú.n. **programozható logikával**(PLA=Programmable Logic Array) egyszerűbben kezelhetővé lehet tenni. (A programozható logikák ÉS- és VAGY-kapukból álló hálózatok, amelyek ÉS-kapuit, vagy VAGY-kapuit, vagy esetleg mindkettőt lehet programozni.) A bonyolultság és a költségesség miatt a huzalozott logikát célgépekben, vagy olyan számítógépekben érdemes csak alkalmazni, ahol a gyors működés kiemelkedő fontosságú, mint például a RISC processzoros gépekben. A vezérlés logikáját tükrözi vissza a 3-32.ábra is, amely párhuzamos utasításfeldolgozás feltételezése mellett vázolja fel a huzalozott vezérlés lényegét.

- **Mikroprogramozott(szoftver) módon**, amely esetben az elemi tevékenységek sorrendjét egy(többnyire ROM tárban) tárolt program, a **mikroprogram** utasításai vezérlik. A mikroutasítások két részből állnak: a különböző kapukat engedélyező vezérlőbitekből és a következő mikroutasítás címéből. A makroszintű, gépi kódú utasítás műveleti jelrésze adja meg a műveletvezérlő mikroprogram kezdőcímét, amelynek a mikroutasításait sorra véve, történik az utasítás feldolgozása. Tehát azt lehet mondani, hogy a mikroprogramvezérelt műveletvégrehajtás tulajdonképpen **számítógép a számítógépben.** A mikrovezérlő, mint egy kisméretű Neumann-elvű számítógép irányítja az utasításvégrehajtás folyamatát(3-33.ábra).

Elvileg a mikroprogramot cserélni is lehet(ekkor nyilván RAM tárban kell elhelyezni) és ez lehetőséget ad arra, hogy ugyanazon a hardveren egy másik utasításkészlettel rendelkező számítógépet hozzunk létre, vagy emuláljunk. A programozható utasításvégrehajtás tehát hajlékonyabb és olcsóbb, mint a huzalozott változat.

A működési elvnek megfelelően vannak fix utasításkészlettel(mikroprogramokkal) rendelkező gépek és léteznek olyanok is, amelyek mikroprogramjait be lehet tölteni.

Az elmondottakból következik, hogy a mikroprogramozott műveletvezérlés esetében a felhasználónak elvileg lehetősége van a beavatkozásra a gép működését tekintve. Azonban a mikroprogramszintű beavatkozásnak igen nagyok a veszélyei is, mivel

- a gép operációs rendszere működésképtelenné válik,
- a magasszintű nyelvek fordítóprogramjai használhatatlanok lesznek,
- a hardver karbantartás(hibafeltárás) igen nehézzé, bizonytalanná válik.

Ezért az ilyen szintű változtatások csak a fejlesztések alkalmával célszerűek azok számára, akik a hardvert tökéletesen ismerik.

b.)Műveleti vezérlés megoldási módjai

A processzorok műveleti vezérlése kétféle vezérlési struktúrában oldható meg, amelyek között az átmenet minden szintje megtalálható. Ez a két forma:

- a horizontális és
- a vertikális vezérlési struktúra.

A horizontális és a vertikális vezérlési struktúrák közötti különbség az utasításfeldolgozás egyes elemi lépéseinek lehetséges párhuzamosítási fokában mérhető. Az utasításfeldolgozás párhuzamosságának szempontjából a következő három szakasz párhuzamosságát lehet vizsgálni:

- tárolóhoz fordulás(utasításelőkészítés, adatelőkészítés),
- műveleti jelrész kiértékelése(dekódolása),
- művelet(µműveletek) végrehajtása.

A műveleti vezérlés előző részben említett kétféle megoldási módjának részletesebb vizsgálatát és kapcsolatuk elemzését megelőzően, előbb a mikroprogramozott műveleti vezérlés lényegét mutatjuk be a további részek magyarázatának előkészítéséül.

Mikroprogramozott vezérlés lényege

Mikroprogramozott vezérlésről beszélünk akkor, amikor egy-egy gépi kódú utasítás végrehajtásának vezérlésére programozott vezérlőegységet használnak fel. A vezérléshez felhasznált programot nevezzük **mikroprogramnak**, amelynek egy-egy utasítása a **mikroutasítás**. A mikroprogramot általában csak-olvasható formában tárolják a vezérlőegységen(CU) belüli ROM tároló-

ban. A vezérlőegység azon részét, amely a műveleti vezérlést oldja meg, ezekben az esetekben **mikrovezérlő**(egység)nek nevezik.

A mikrovezérlő a mikroprogram végrehajtásakor, a program mikroutasításai alapján vezérlőjeleket ad ki, amelyek a számítógép vezérlési pontjaihoz kapcsolódó elemi műveletek, tevékenységek engedélyezésére szolgálnak. Tulajdonképpen a mikroutasítások feladata: adatutak engedélyezése/tiltása, állapotjelzők beállítása/törlése.

Bonyolult utasítások végrehajtásának vezérlésére szolgáló mikroprogramok feltételes vezérlésátadó és eljáráshívó mikroutasításokat is tartalmaznak. A feltételes vezérlésátadó(ugró) utasításokhoz külső feltételek teljesülését is vizsgálhatja a mikrovezérlő egység.

Mikroutasítások szerkezete

A mikroutasítások felépítése némileg eltér a gépi kódú utasításokétól, de alapjaiban hasonló ahhoz. Az utasítások szerkezete függ a kialakított vezérlési struktúrától, azonban tartalmilag két részből áll általánosan, amelyek:

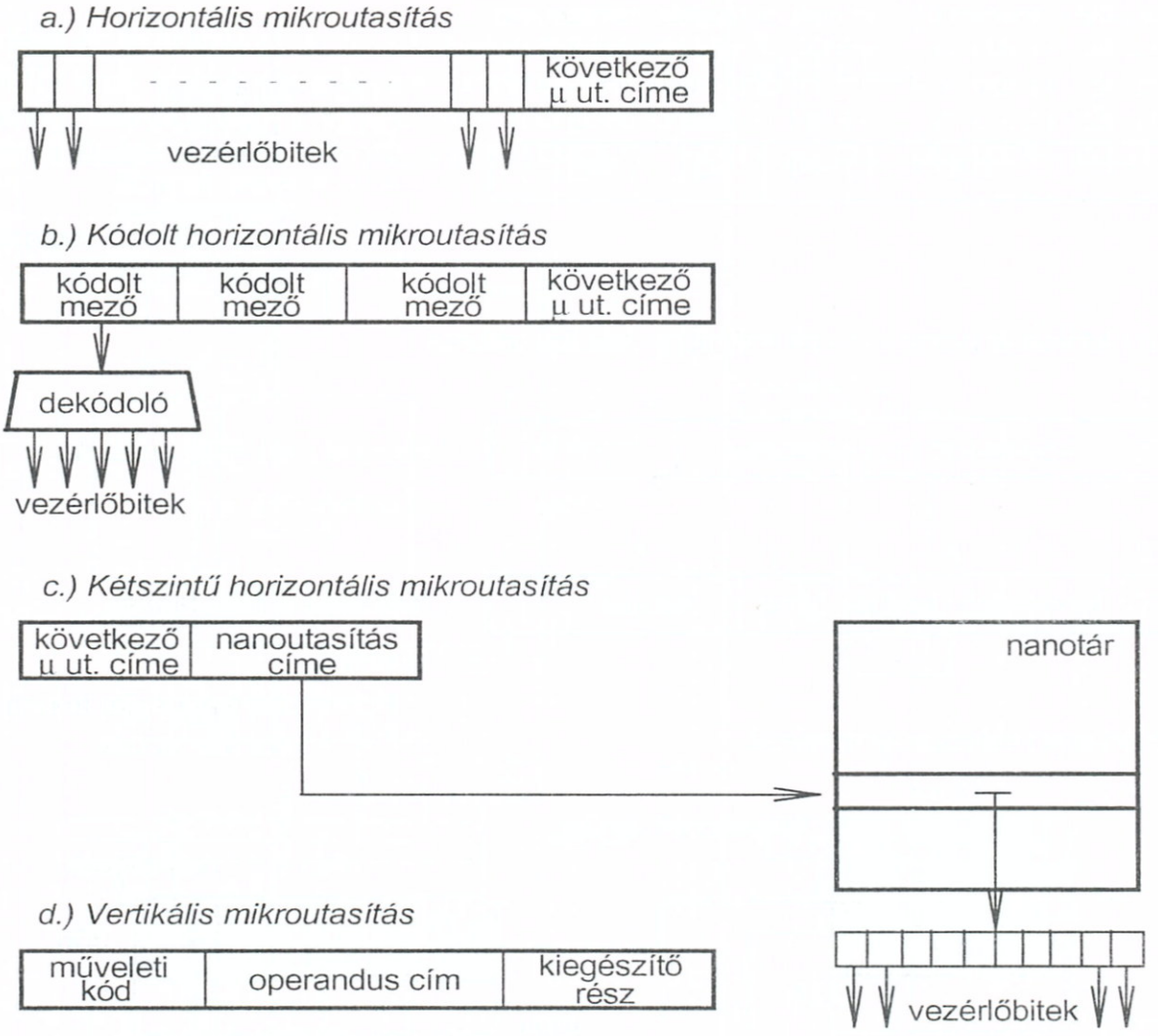
- a **következő mikroutasítás** mikroprogramtárbeli **címe**, amely nem minden változatnál található meg;
- a **vezérlési mező**, amely közvetlenül, vagy közvetve meghatározza az engedélyezett vezérlési pontokat; a vezérlési mező esetleg további részekre bontható, az alkalmazott vezérlési formától függően.

A műveleti mikrovezérlő felépítését alapvetően meghatározza az alkalmazott mikroutasítás-struktúra.

A **horizontális mikroutasítás**ban a vezérlési mező minden bitje önálló funkciót tölt be, egy-egy vezérlési pont engedélyezésére/tiltására, vagy állapotjelző beállítására/törlésére szolgál. Sok esetben azonban nem elegendő a vezérlési mező hossza minden vezérlési pont elérésére, ezért alkalmazzák az ú.n. kódolt (packed) mezőket, amelyek a vezérlési pontok egy-egy csoportját fogják össze és a mező további dekódolása után kapjuk meg a pontonkénti vezérlő jeleket. A szokásos mikroutasítás formák a 3-30.ábrán láthatók.

A mikroutasítások másik formája a **vertikális** vezérlési struktúrához tartozó **utasításformátum**(3-30.ábra). Ennél az utasításszerkezetnél az utasítás műveleti előírást és operandusra utaló címrészt tartalmaz. Egy-egy utasítás csak egyetlen, vagy kisszámú elemi műveletet vezérel.

Az utasításszerkezetet befolyásolja az alkalmazott **tárolási forma** is. A mikroprogramok tárolására szolgáló memóriaterület csökkentése érdekében gyakran alkalmazzák a kétszintű tárolási módot. Ebben az esetben a mikroutasításban a vezérlési mező helyén egy cím található, a vezérlési mezőnek az ú.n. *nanotárolóbeli címét*. Tehát a vezérlési biteket tartalmazó mezőt külön(ROM) tárolóban, a nanotárolóban helyezik el.

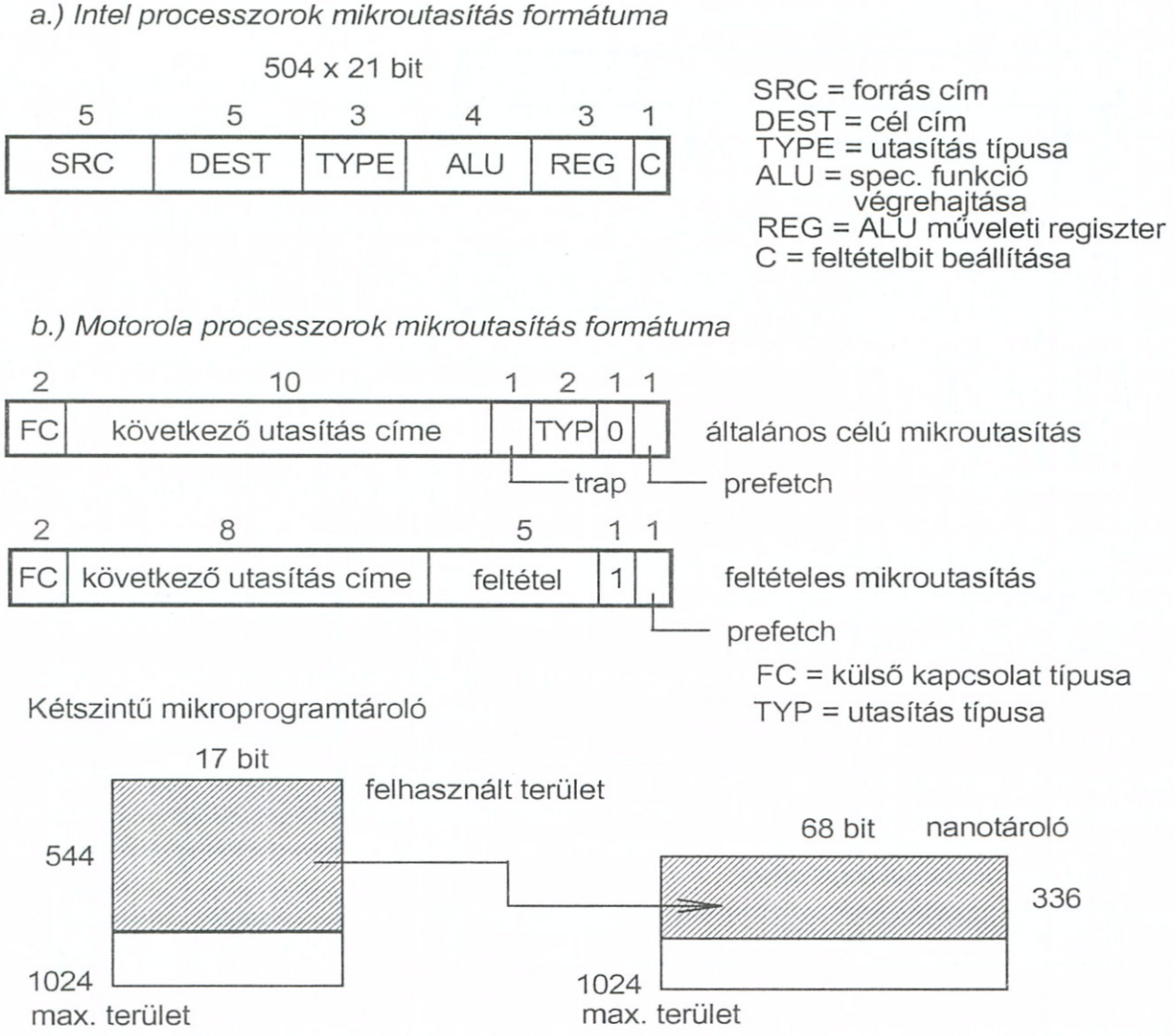


3-30.ábra: *Mikroutasítás szerkezetek*

A kétszintű tárolási forma előnye az, hogyha több mikroutasítás is ugyanazt a vezérlési jelsorozatot használja, akkor azt elegendő csak egy példányban tárolni a nanotárolóban és mindegyik utasítás ugyanerre a helyre hivatkozhat akkor. Ez bizonyos feltételek teljesülése mellett, összességében tárolóhely megtakarításhoz vezet. Hátránya a megoldásnak a hosszabb utasítás kiértékelési időtartam.

A különböző utasításformákra mutat **konkrét példák**at a 3-31.ábra.

A Motorola MC68000 processzor-család kétszintű, horizontális mikroprogramozást alkalmaz, összesen két utasításformátumot használva; egy általános célú és egy ugróutasítás céljára szolgáló formát. Az Intel 8088/8086 processzorok a vertikális mikroprogramozási módot alkalmazzák, alapvetően hasonló felépítésű, többféle célú utasítással dolgozva.



3-31.ábra: *Intel és Motorola mikroutasítások*

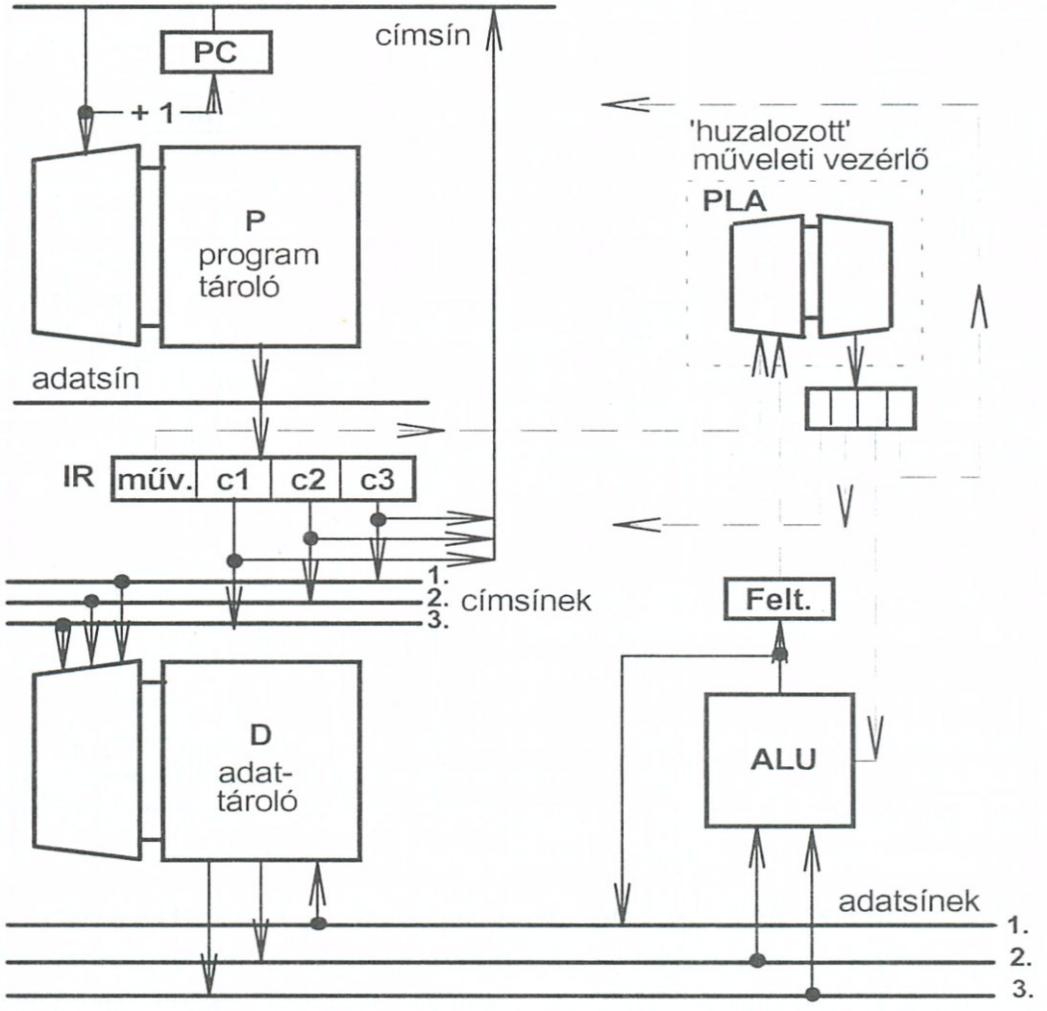
Műveleti vezérlés szintjei

Az előző részben említett két mikroprogramozási struktúra között számtalan átmenet létezik. A maximális párhuzamos feldolgozástól('huzalozott vezérlés', RISC processzorok) a teljesen soros feldolgozásig(CISC processzorok, pl.: DEC VAX 11/780, Intel processzorok) terjedő strukturális megoldások egyaránt használatosak.

A két vezérlési struktúra két szélső esetet tükröz vissza: az utasításfeldolgozás lehetséges, maximális párhuzamosítását(pl. huzalozott vezérlés, horizontális mikroprogramozás) és a lehetséges, maximális soros elvégzését(vertikális mikroprogramozás).

1.Huzalozott műveleti vezérlés

A huzalozott vezérlést a mikroprogramozás egyik szélső esetének tekinthetjük, amelyben a részfeladatok lehetséges maximális párhuzamosítását valósítjuk meg, megfelelő strukturális megoldással(3-32.ábra). A vázolt struktúrában, mindhárom szakasz párhuzamossága, kvázi-egyidejűsége biztosítható.

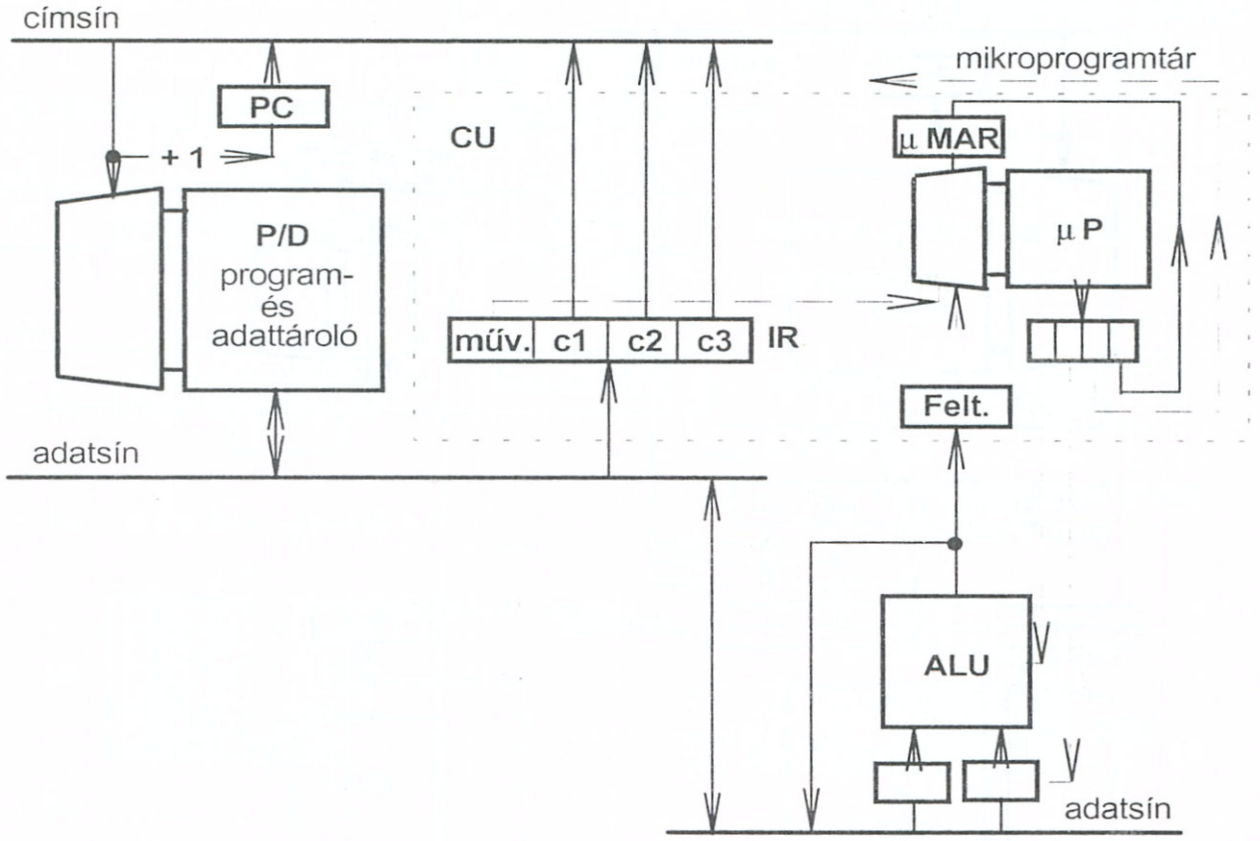


3-32.ábra: *Huzalozott, horizontális vezérlési struktúra*

Az utasításelőkészítés és az adatelőkészítés párhuzamos végrehajtását a külön program- és adattároló teszi lehetővé. Evvel egyidejűleg történik a műveleti jelrész dekódolása és a vezérlési pontok engedélyezése, vagy tiltása. Ez utóbbinak a megvalósítója a műveleti jelrész dekódoló, amelynek a működése ebben a formában felfogható egy olyan **mikroutasítás** végrehajtásának, amely **csak vezérlőbiteket tartalmaz**. Tehát a részfolyamatok teljes párhuzamosítása jön létre, a gépi utasitás végrehajtása egy ciklusban történik.

2.Horizontális mikroprogramozás

Horizontális mikroprogramozás esetében a vezérlési mező minden egyes bitje egy-egy vezérlési pontot engedélyezhet, vagy tilthat. Ha a gépi utasítás feldolgozása több lépésben oldható meg csak, akkor a mikroutasítás tartalmazza a következő utasítás címét is.



3-33.ábra: *Mikroprogramozott vezérlési struktúra*

Ha a párhuzamosítás mértéke csökken, amiatt például mert ugyanazt a tárolót használjuk a programutasítások és az adatok tárolására is, akkor kétszer kell a tárolóhoz fordulni, először a programutasítás lehívásához, másodszor az adat(ok) előkészítéséhez. Ebben az esetben csak a műveleti kód kiértékelése és annak műveleti vezérlése oldható meg kvázi-egyidejűleg. A 3-33.ábrán ennek logikai vázlata látható.

Az utasításfeldolgozás több lépcsőt igényel, emiatt a műveleti jelrész által elindított mikroprogram több mikroutasítás végrehajtását jelenti egyben. A mikroutasításoknak **a vezérlőbiteken kívül, a következő mikroutasítás címét is tartalmazniuk kell**. Így a mikrovezérlő egy címregisztert(µMAR) is tartalmaz.

A szélsőségesen horizontális(amely logikailag a 'huzalozott' műveleti vezérlésnek felel meg) és az egyszerű horizontális vezérlési struktúrák meglehetősen merevek és kevéssé változtathatók; egy adott hardver felépítéshez alkalmazkodók.

3.Vertikális mikroprogramozás

Vertikális mikroprogramozás az a fajta vezérlési struktúra, amelyhez tartozó mikroutasítások csak egy-egy elemi művelet végrehajtását engedélyezik, így a gépi kódú utasítás feldolgozása több mikroutasítás egymásutáni végrehajtását igényli. A mikroprogram tartalmazhat mikroeljárást, mikroszubrutint hívó utasítást is, azaz felépítése teljesen hasonlóvá válik a makroszintű, gépi kódú programokéhoz. Ebben az esetben, a mikroprogram automatikus végrehajtása miatt, a mikrovezérlőben mikroutasításszámláló regiszterre(μPC) is szükség van.

Az utasításvégrehajtás további részműveletekre bontása és ezek soros feldolgozása mikroutasítások által, a mikrovezérlő bonyolultságát növeli. A párhuzamosítás lehetősége csak a mikroműveletek végrehajtásában marad meg, mivel a tárolóhoz fordulás is és a mikroutasítások dekódolása is sorosan oldható csak meg(3-33.ábra).

A gyorsabb működés miatt a vezérlőegység az adatok tárolására regisztertárakat tartalmaz és a mikroutasítások ezeknek a tartalmával dolgoznak elsődlegesen. A mikroutasítások felépítésében **néhány bit, mint műveleti jelrész értelmezett**, amely megadja azt, hogy a mikroutasításban lévő vezérlési mezők(operanduscímek) előírásait milyen módon kell értelmeznie a mikrovezérlőnek.

A vezérlési mezők(kódolt mezők) tartalma további dekódolást igényelhet, mivel azok nem a vezérlőbiteket tartalmazzák közvetlenül. Ez egyúttal a mikroutasítás végrehajtási idejét is megnöveli.

A mikroprogram bonyolultsága növekszik, feltételes vezérlésátadó és szubrutinhívó mikroutasítások feldolgozására is szükség van. Emiatt a mikrovezérlő egy utasításregisztert(μIR), valamint a mikroprogram automatikus végrehajtásához egy utasításszámláló regisztert(μPC) is tartalmaz.

c.)Vezérlés módja és a processzor felépítése közötti kapcsolat

A processzorok struktúrájának kialakítását nagymértékben befolyásolja a választott vezérlési struktúra. A komplex utasításkészletű(CISC) processzorokra a mikroprogramokkal megvalósított műveleti vezérlés a jellemző. Ezekre viszonylag könnyű megfelelő fordító programot készíteni.

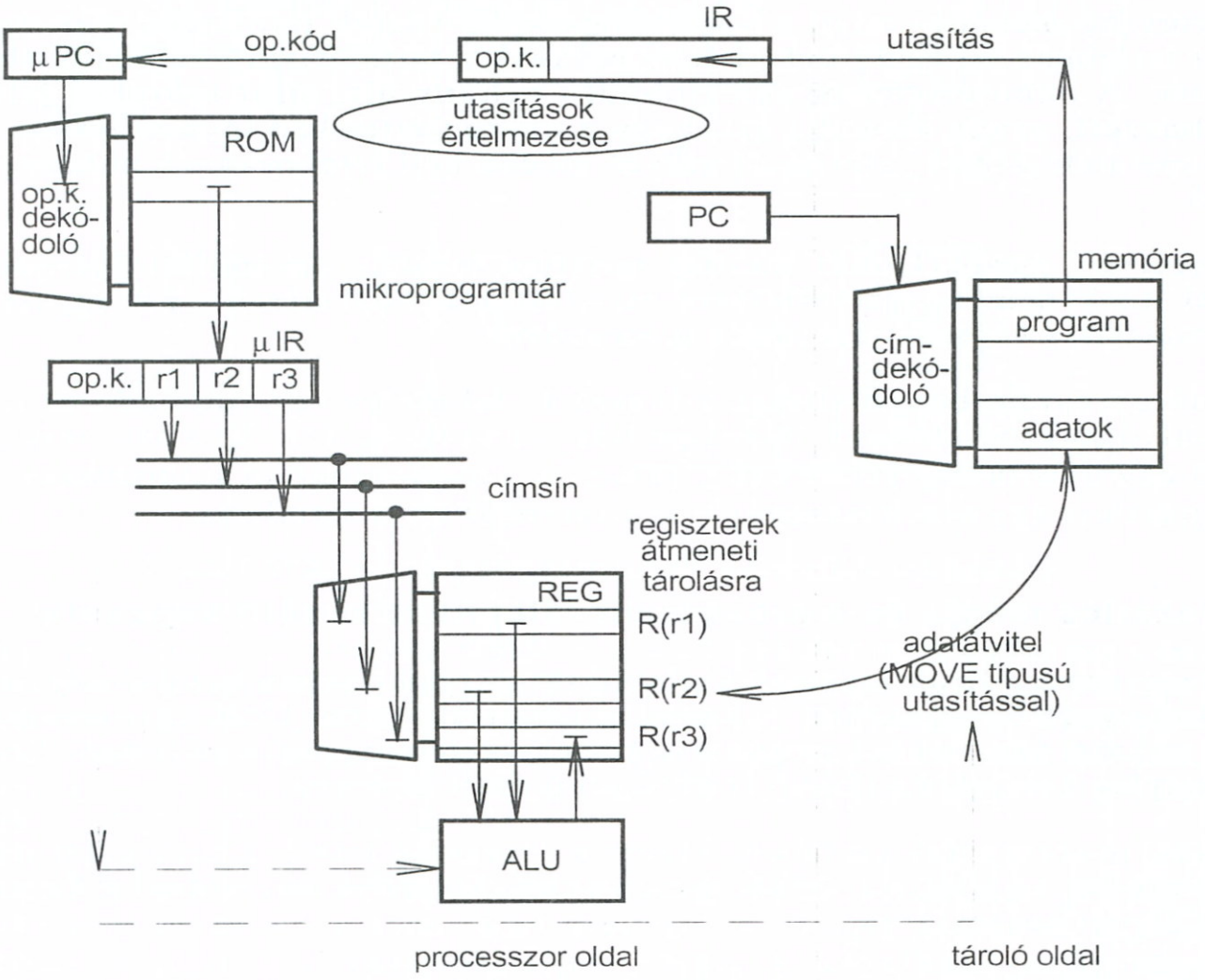
A RISC processzorok esetében viszont a gyorsaság miatt, a huzalozott műveleti vezérlés az elterjedt, habár azért vannak kivételek is; például a Fairchild cég Clipper processzora.

Az elmondottakat a 3-34. és 3-35.ábra logikai vázlatai tükrözik vissza. Az ábrák szerint, a műveleti vezérlés megoldása abban különbözik a CISC és a RISC processzorok esetében, hogy

- a CISC processzoroknál, a magasszintű programozási nyelv segítségével készült felhasználói programot az összetett utasításokkal rendelkező gépi nyelvre fordítjuk le, majd az igy előállt gépi utasításokat a mikro-

programok segítségével az adott hardveren értelmezzük, végrehajtjuk; tehát az utasítások feldolgozása, végső soron egy mikroutasításokkal megvalósított **értelmező(interpreter) rendszer igénybevételével** történik; a műveletekhez szükséges adatok mozgatása legtöbbször a memória és a regiszterek között történik;

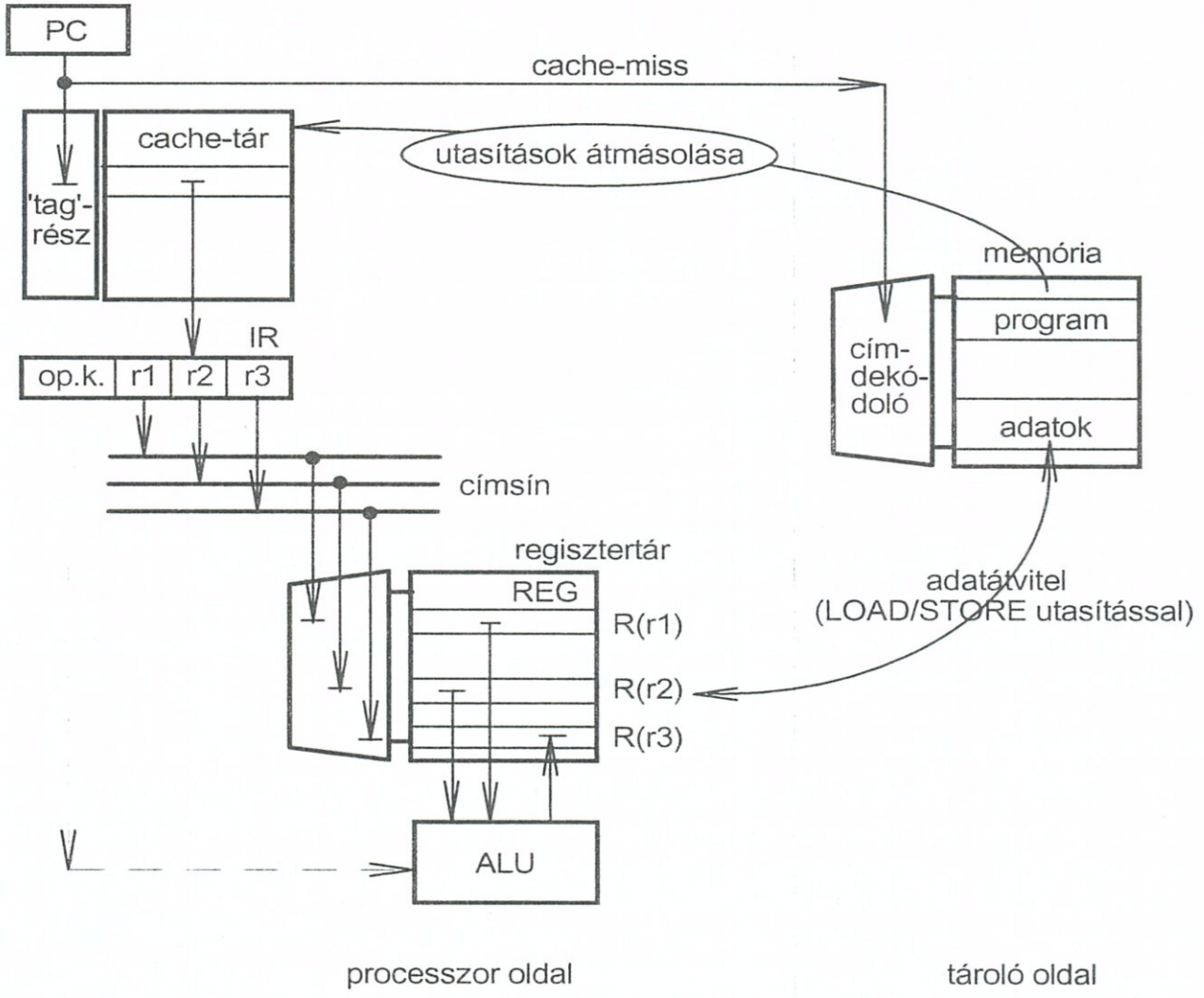
- a RISC processzorok esetében a magasszintű programozási nyelven elkészített felhasználói programot egy olyan **egyszerű utasításokból álló gépi kódra fordítjuk le**, amely nem igényli már a mikroutasításokkal megvalósított értelmező rendszert, mert utasításai közvetlenül végrehajthatók; tehát ez esetben, a lefordított programot a főtárból letöltve egy erre a célra szolgáló ú.n. cache-tárba, a végrehajtás ebből történik, a mikroprogramokhoz hasonló formában; az adatok mozgatása többségében a regiszterek között történik, a tárhoz fordulás gyakorisága igen alacsony.



3-34.ábra: *CISC processzorok műveleti vezérlési elve*

A CISC és RISC processzorok vezérlési struktúrájából adódó különbségek alapján azt lehet mondani, hogy a RISC processzorok(gépi kódú) programozása nagyon közel van a CISC processzorok mikroprogramozásához. Emiatt a

RISC processzorokat nem túl kényelmes gépi kódban programozni és nem is szokás. A gépi kódú program, az optimális kód előállítását, a fordítóprogramra kell bízni. *Minden RISC processzor olyan jó, amilyen jó a hozzátartozó fordítóprogram!!*

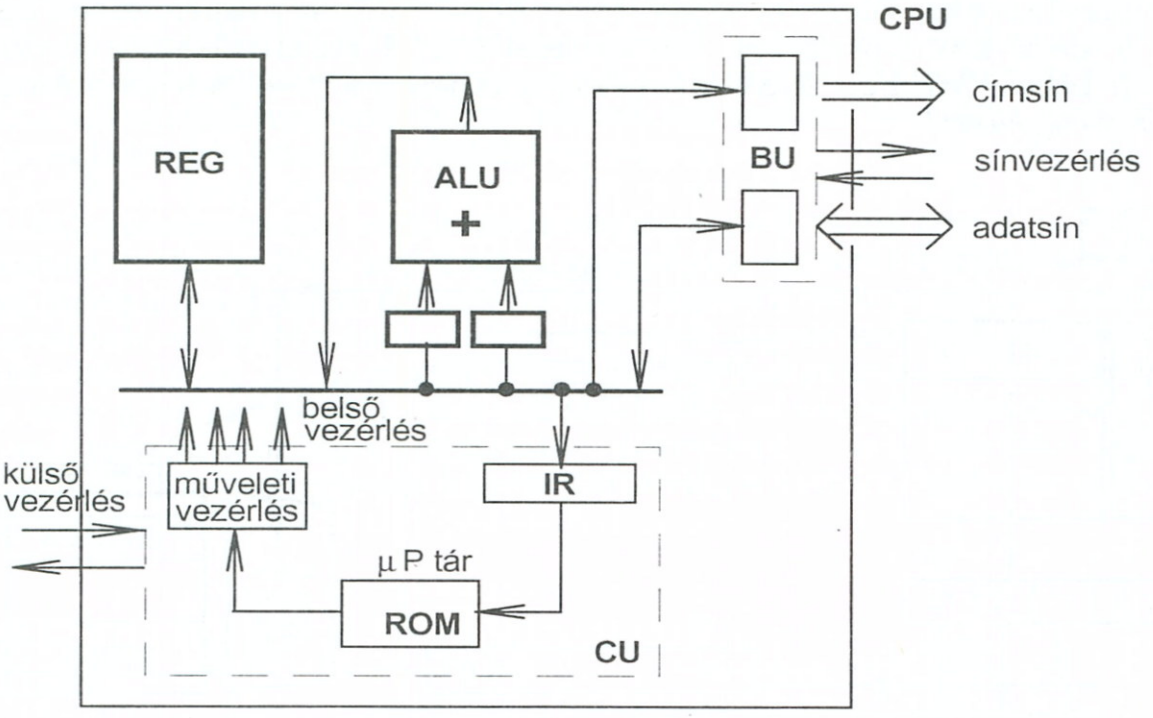


3-35.ábra: *RISC processzorok műveleti vezérlési elve*

3.4.3.Vezérlő egység

Az utasításvégrehajtás irányítására szolgáló vezérlő egység csak a processzor egészével együtt vizsgálható érdemben. A következő, 3-36.ábra a (CISC) mikroprocesszorok leegyszerűsített vázlatát mutatja be az elemek legfontosabb kapcsolódási lehetőségeivel.

A vezérlő egység, az utasításregiszterben(IR) megjelenő utasítás értelmezésével, vezérlő jeleket ad ki a processzor belső és a számítógép külső irányítására egyaránt. A vezérlési jelek kiadása, a CISC processzorok esetében, mikroprogram segítségével történik.



3-36.ábra: *Mikroprocesszorok főbb részei*

A **belső vezérlő jelek** az aritmetikai egység működését és a regiszterek közötti adatutak nyitását/zárását irányítják.

A **külső vezérlő jelek**

- *egyrészt* a processzor és a memória, illetve I/O eszközök közötti adatátvitelt irányítják(memória-periféria választás [M/IO*], olvasás-írás választás[R/W*], adathossz választás[W/B*], cím sínretétele[AS=Address Strobe], adat sínretétele[DS=Data Strobe]),
- *másrészt* a megszakításkezelést intézik(megszakításkérés [INTREQ], megszakításkérelem visszaigazolása [INTACK], nem maszkolható megszakításkérelem [NMI]),
- *harmadrészt* a sínvezérlést oldják meg(sín-kérelem[BREQ], visszaigazolás[BG], foglalás[LOCK]).

A vezérlő jelek elnevezését csak tájékoztató jelleggel adtuk meg, mivel *egyrészt* az egyes processzorok esetében ezek nem mindegyike létezik, *másrészt* az elnevezések is mások lehetnek.

3.5.UTASÍTÁS- ÉS MŰVELETVÉGREHAJTÁS GYORSÍTÁSA(PIPELINING)

A RISC, de a CISC processzorok teljesítőképességét is alapvetően befolyásolja a folyamatok párhuzamosítása. A nagyfokú párhuzamosítás, különösen az utasítások feldolgozása vonatkozásában, a processzorokra általánosan jel-

lemző, ezért ennek megoldásait a súlyának megfelelően, részletesen megvizsgáljuk a következőkben.

3.5.1.A párhuzamosítás lényege

A számítógép működésének gyorsítása *egyrészt* az alapütemet megszabó órajel frekvenciájának növelésével, *másrészt* a gépen futó folyamatok párhuzamosításával oldható meg. Az órajel frekvenciája nem független az adott időszak technológiai lehetőségeitől, nem növelhető tetszőlegesen; ezért a folyamatok párhuzamosítása fontos formája a teljesítmény növelésének.

A feldolgozási folyamatok a programok utasításainak végrehajtását, a benne előírt műveletek elvégzését jelentik. E kettő ugyan összefügg egymással, de érdemes külön vizsgálni

- egyrészt az utasításvégrehajtás lépéseit,
- másrészt az (aritmetikai) műveletek elvégzésének lépéseit.

A folyamatok párhuzamosítási lehetősége azok részfázisokra való bonthatóságából adódik. A folyamatot olyan lépésekre kell bontani, amelyek mindegyike önálló részt képez, más és más erőforráshoz kapcsolódik. Így, amint felszabadul valamelyik erőforrás, azt a másik folyamat hasonló feladatot elvégző fázisa igénybeveheti. Az egyik fázis eredménye a következő fázis induló adatát képezi(3-37.ábra). Ez az átlapolódó megoldás azt eredményezi, hogy egy-egy feldolgozási folyamat végrehajtási időtartama ugyan nem változik, de ugyanannyi idő alatt lényegesen több folyamat fejezhető be. Gyakorlatilag az elemi fázisok végrehajtási idejének megfelelő időközönként fejeződik be egy-egy folyamat.

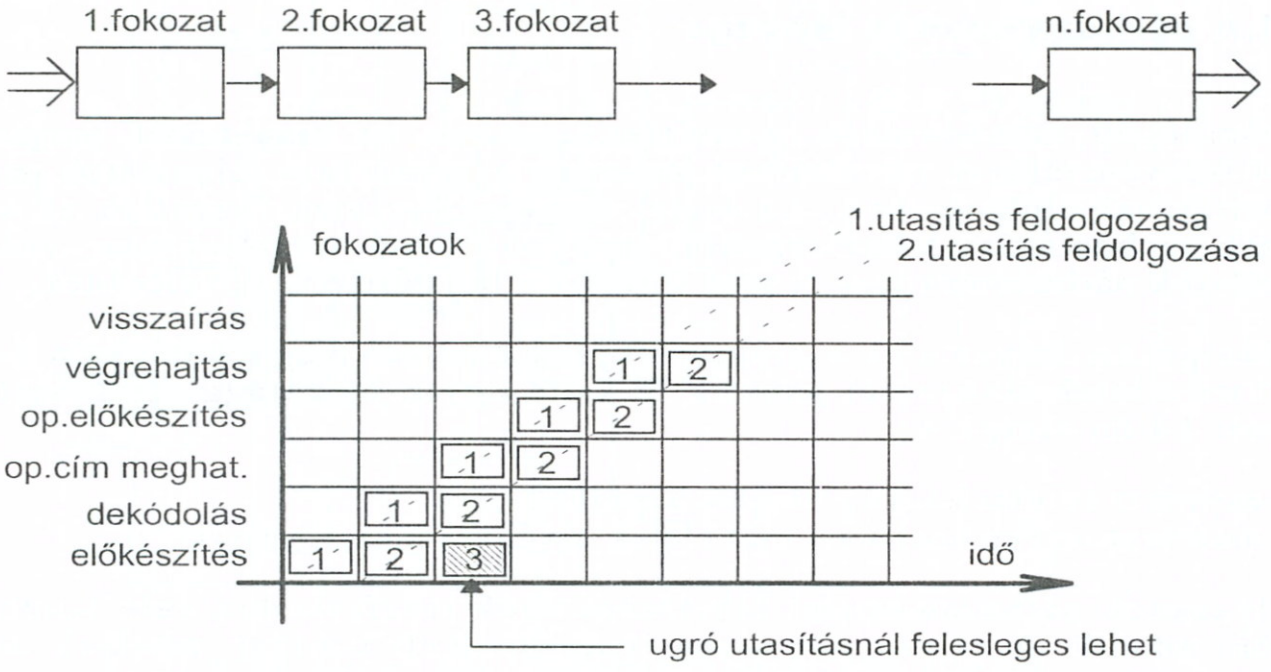
Ezt az átlapolt feldolgozási módszert nevezik **pipelining**(adatcsatornás, futószalag) **feldolgozás**nak.

Az ilyen módon átlapolt folyamatok feldolgozásához tartozó egységek működtetése kétféle módon történhet:

- **aszinkron ütemezés**sel, amely esetben az egymást követő fokozatok jelzik egymásnak elemi feldolgozási lépésük elkészültét, illetve azt, hogy készek fogadni a következő utasítást az arra az egységre előírt feladat elvégzéséhez; mindegyik egység amint befejezte tevékenységét, továbbadja a feldolgozást a következő egységnek, azaz a feldolgozás továbbhaladása folyamatos az adatcsatornán keresztül;
- **szinkron ütemezéssel**, amelynél az egyes fokozatok azonos időben kezdik feldolgozási lépéseiket, a feldolgozási folyamat ütemezését mindig a legtöbb időt igénybe vevő egység szabja meg.

A folyamat szinkronizálásában a legnagyobb problémát az jelenti, hogy az egyes szakaszok időszükséglete igen nagy mértékben eltérhet egymástól. Ez különösen a bonyolult utasításokkal rendelkező CISC processzoroknál gond. A RISC processzorok esetében az utasítások egyszerűbbek, így könnyebb megoldani az azonos időtartamú lépések kialakítását. Az adatáramlás, a

feldolgozás kiegyenlítésére szolgálnak az egyes fokozatok közé beiktatott átmeneti tárolók, pufferek(latch).



3-37.ábra: *Pipelining feldolgozás vázlata és ütemezése*

A 3-37.ábra az adatcsatornás(pipelining) feldolgozás vázlatát és az ütemezés időbeli lefutását mutatja be.

Az ábrán látható terhelési diagram alapján, többféle pipeline-használat különböztethető meg. Ha egy-egy feladat végrehajtása más és más leterhelési diagramot eredményez, azaz az egyes feladatok(utasítások) nem ugyanazokat az erőforrásokat használják fel, akkor **multifunkcionális,** egyébként pedig **egyfunkciós** pipeline-ról beszélhetünk.

Ugyancsak különbség tehető aszerint, hogy a pipeline-on végighaladó feldolgozások minden erőforrást csak egyszer használnak-e fel, vagy több ízben is. Az első esetben **lineáris** pipeline-ról beszélünk.

3.5.2.Az utasításvégrehajtás gyorsítása

Az utasításvégrehajtási folyamat párhuzamosítását nehezítik az egymást követő utasítások egymásrahatása. Ez a következő területeken befolyásolja a feldolgozás folyamatát:

- az utasítások egymásutánisága, az utasításfolyam feldolgozása,
- az egymást követő utasítások által használt adatok rendelkezésre állása,
- a hardver erőforrások igénybevétele.

A megoldandó problémák gyakorlatilag három témakörbe sorolhatók:

- strukturális(memóriaelérés, az ú.n. LOAD/STORE architektúra),
- vezérlési(program elágaztatás kezelése),
- adatkezelési problémák körébe.

A nehézségek megoldására két úton van lehetőség:

- **szoftver úton**, azaz a program elkészítésekor, vagy fordításakor figyelembe venni a processzor működéséből adódó megoldási lehetőségeket, követelményeket; így pl. programozáskor a strukturált programírás elveit figyelembe venni, vagy a RISC processzorok esetében, a fordítóprogramra bízni a programutasítások optimalizált sorrendjének, a lehetőségeknek az elemzését;
- **hardver úton**, amely ugyan gyorsabb, de bonyolítja a processzor felépítését.

a.)Az utasításfolyam feldolgozása

Az utasításvégrehajtás folyamatát többféle módon lehet elemi fázisokra felbontani. Az egyes végrehajtási lépések lehetséges tartalma, a RISC processzorok jellegzetességeire is tekintettel:

- utasítás előkészítése(fetching), amely az utasítás, többnyire cache-tárbóli kiolvasását jelenti;
- utasítás dekódolása(decoding), a műveleti előírás értelmezése;
- operanduscímek kiszámítása, meghatározása;
- operandusok előkészítése a tárból,
- művelet végrehajtása(executing), amely még általában további fázisokra bontható;
- az eredmény visszaírása az előírt tárolóhelyre;
- (a következő utasítás címének meghatározása).

Az egyes végrehajtási fázisok tartalma attól is függ, hogy milyen utasítást (pl. regiszter hivatkozású műveleti, vagy memória hivatkozású utasítást) kell-e végrehajtani. A soronkövetkező utasítás címének kiszámítása, normál soros feldolgozásnál, logikailag az utasításelőkészítéssel egyidőben, vagy azt követően már megtörténhet, de van ettől eltérő megközelítés is.

Az utasításfeldolgozás lépéseinek kialakításakor a kérdés az: milyen felbontási mélységig érdemes elmenni, mi szab határt a felbontásnak? Ennek megoldásában a legfontosabb megfontolandó szempontok az alábbiak.

- *Egyes fokozatok erőforrás igénye:* komoly gondot jelent a különböző fázisok által igényelt erőforrások egyezősége. Így például az utasítások előkészítése tovább bontható az alábbi lépésekre:
  - az utasításszámláló(PC) tartalmának átvitele,
  - az utasítás továbbítása a dekódoláshoz.

  Mindkét lépés ugyanazt az adatutat igényli, azaz csak a tár két, önálló hozzáférésével oldható meg a keletkezett ütközési probléma. Tehát a fázisok rövidítése az erőforrások többszörözését kívánhatja meg.

- *Időszükséglet:* a feldolgozási fázisok időszükséglete is határt szab a felbontásnak, mert például a (cache-)tárak elérési ideje(kb. 25 ns) alsó határ lehet az azonos időtartamú lépések kialakításában.
- *Áramköri problémák:* a pipeline vezérléséhez tartozó logika bonyolultsága miatt, olyan késleltetések alakulhatnak ki, amelyek a végrehajtási idő szempontjából, ugyancsak alulról korlátozzák a kialakítható fokozatokat.

Az utasítások folyamatos feldolgozását

- a tárolóhivatkozású utasítások,
- a feltétel nélküli, a feltételes vezérlésátadó(ugró) és ciklusutasítások, valamint
- a megszakítások, kivételek(exceptions)

zavarják meg leginkább. Mivel ezek száma meglehetősen nagy, a mérések szerint az utasítások 30%-a vezérlésátadó, 40-45% a nagyrészben tárolóhivatkozású utasítást eredményező adatátviteli, értékadó utasítás, a feldolgozás gyorsítása szempontjából fontos, hogy ezek kezelésére megfelelő eljárásokat használjanak a tervezők. Mindegyik esetben a folyamatos feldolgozást meg kell szakítani, fel kell függeszteni, aminek következtében a feldolgozásban lyukak (bubbles, holes) keletkeznek és ezek csökkentik a pipeline hatékonyságát.

A memóriautasítások időtartama a tárolóhoz fordulás miatt hosszabb, mint a többieké, a vezérlésátadó utasítások esetében pedig nem a soron következő utasítással kell folytatni a feldolgozást, hanem valamely más memóriacímen lévő utasítással. Ez viszont csak a 2., vagy 3.fázisban derül ki és ez esetben az adatcsatornába már betöltött utasítások feldolgozását meg kell szüntetni. Ugyancsak meg kell szüntetni a feldolgozást a megszakítások kezelése esetében is.

Memóriautasítások

A pipeline folyamatos működtetésének szervezése szempontjából akadályozó tényező a tárolóhoz fordulás nagyobb időigénye, valamint az is, hogy a beolvasott adat emiatt nem áll rendelkezésre azonnal. Gondot okozhat az is, ha nincs külön utasítás- és adattároló(Harvard-struktúra), hogy egyidőben kell egy adatot mozgatni és egy utasítást előkészíteni, mivel mindkét fázis ugyanazt az adatutat használja.

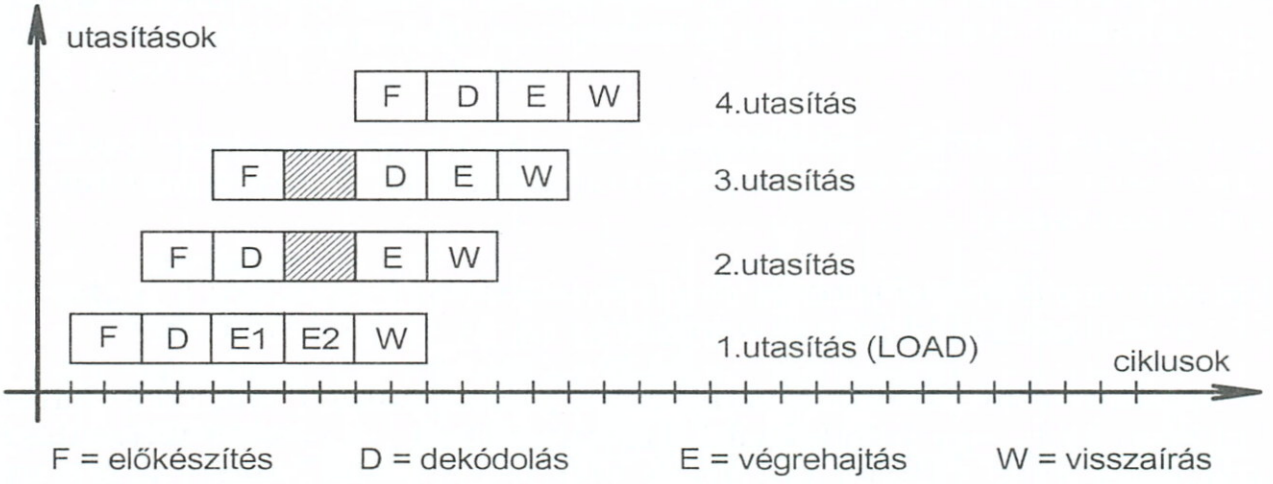
1. Váróciklusok alkalmazása

Az egyik megoldási lehetőség a memóriautasítást követő utasítások feldolgozásának a felfüggesztése ú.n. **váróciklusok**(wait cycle) **beiktatásával** (3-38.ábra), amelynek eredményeként a pipeline-ban lyukak(bubble) keletkeznek és ha sok a tárolóhoz fordulás, akkor emiatt jelentősen lecsökkenhet a feldolgozás hatékonysága. Az ábrán a hosszabb memóriához fordulási időtartam lehetséges hatását mutatjuk be. (Célszerűbb azonban azonos fokozatszámot kialakítani minden utasításnál, például a fokozatidők megnövelésével, kiegyenlítve az időbeli eltéréseket.)

2.Késleltetett memóriautasítás

Késleltetett memóriautasítás(delayed LOAD) alkalmazására akkor kerülhet sor, ha a LOAD által betöltött adatra az utána következő utasítás(ok)ban szükség lenne már. A késleltetés azt jelenti, hogy a pipeline munkájának várakoztatása helyett, a LOAD utasítást követően egy olyan utasítást kell beiktatni, amely független a beolvasandó adattól, azaz azt nem használja fel a benne előírt művelethez. Ez az utasítás lehet az üres utasítás(NOP) is, de akkor a hatékonyságot evvel csökkentjük. Például, ha eredetileg a végrehajtandó utasítássorozat és annak hatása a következő:

```
load R1,mem(A)        [mem(A)] ⇒ R1
load R2,mem(B)        [mem(B)] ⇒ R2
add R3,R1,R2          [R1]+[R2] ⇒ R3
```



3-38.ábra: *Memóriautasítás hatása a feldolgozásra*

akkor látható, hogy a harmadik utasításban szükség lenne az R1 és R2 regiszterek tartalmára, amelyből az R2 nem áll még akkor rendelkezésre, amikor az összeadó utasítás azt igényelné. A megoldást az utasítások átrendezése adja, legegyszerűbb esetben egy NOP utasítás beiktatásával.

```
load R1,mem(A)        [mem(A)] ⇒ R1
load R2,mem(B)        [mem(B)] ⇒ R2
nop
add R3,R1,R2          [R1]+[R2] ⇒ R3
```

A szükséges átrendezést a **fordítóprogram végzi el**, azaz ez a megoldási mód szoftver megoldású. Ha a késleltetés igénye több, mint egy fokozat, akkor növekszik a **késleltetési rés**(delay slot) hasznos utasításokkal való kitöltésének a nehézsége.

Elágazások kezelése

A pipeline folyamatos működtetésében a programban lévő vezérlésátadó utasítások(feltétel nélküli, feltételes ugró utasítások, ciklusutasítások) okozzák a legtöbb nehézséget. Ennek *egyrészt* az az oka, hogy a feltétel teljesülése, az ugrási cím, csak az utasítás részbeni, vagy teljes feldolgozása után válik csak ismertté; *másrészt* ha az ugrási címtől kell folytatni a végrehajtást, akkor az adatcsatornában(pipeline-ban) lévő utasításokat törölni kell. Ha ezek az utasítások már módosították valamelyik tárolóhely tartalmát('mellékhatások'), akkor a törléssel együtt vissza kell állítani az eredeti állapotot is.

A vezérlésátadó utasítások feldolgozásakor jelentkező problémák megoldására, többféle eljárást is alkalmaznak.

A ***feltétel nélküli vezérlésátadó(ugró) utasítások*** esetében, az egyik megoldás szerint a belépő utasításokat mindaddig várakoztatja a processzor, amíg az utasítás kiértékelése, az ugrási cím kidolgozása megtörténik. Ez természetesen kihagyást, üres fázisokat eredményez a feldolgozási folyamatban, hiszen utána ismét el kell kezdeni feltölteni az adatcsatornát a feldolgozandó utasításokkal.

Másik megoldás szerint, változatlanul folytatódik az utasítások betöltése az adatcsatornába és az ugrási cím meghatározása után a processzor törli a felesleges utasításokat.

A ***feltételes vezérlésátadó(ugró) és ciklusutasítások*** esetében a legfontosabb teendő valamilyen módon meghatározni az ugrás várható irányát.

- A **statikus** előrejelzési módszer a program fordítási időszakát használja fel arra, hogy egy feltétel várható teljesülését, vagy nem teljesülését figyelembe véve állítson elő olyan utasítássorozatot, amely a folyamatos feldolgozást legnagyobb valószínűséggel biztosítja.(Pl.: ciklusutasításban lévő feltételes ugró utasítás az esetek legnagyobb részében a ciklus elejére lépteti vissza a végrehajtást; egy rendszerhiba bekövetkezését vizsgáló utasításnál feltételezhető, hogy nem következik be a hiba, stb.)

- A **dinamikus** előrejelzési módszer futás közbeni vizsgálatokat jelent. Az alkalmazott módszerek avval a feltételezéssel élnek, hogy a feltételes ugró utasítások ismétlődnek, mint például a ciklusokban.

  A dinamikus *módszerek egyike* szerint a processzor egy táblázatban tárolja az ugró utasítások címeit és figyelemmel kiséri azt, hogy az utasításban foglalt feltétel teljesült-e, vagy sem és a következő alkalommal ezt az eredményt figyelembe véve készíti elő az adatcsatornában a feldolgozandó utasításokat.

  Egy *másik módszer* hasonló elvet követ. Egy cache-tárban(átmeneti tárolásra szolgál; részletesen 4.fejezetben kerül ismertetésre) az ugró utasítások címét és az ugrás helyének címét tárolja a processzor és amikor ismételten ugyanazt az utasítást kell feldolgozni, akkor az ugrási címet a cache-tárból veszi elő a processzor.

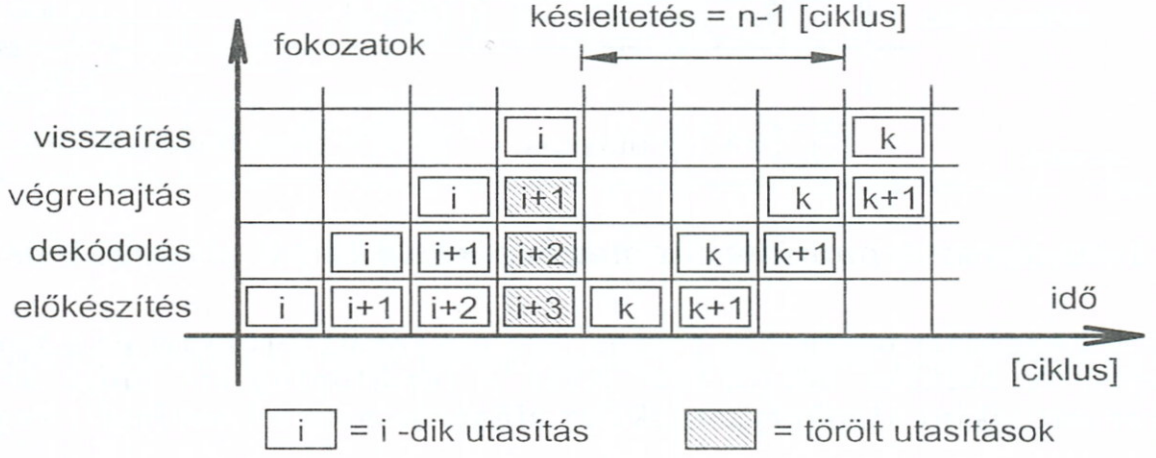
Az elágazások kezelésére használt legfontosabb módszerek az alábbiak:

- alapmódszer, a pipeline törlése(flushing, squashing),
- leállítás(pipeline freezing),
- elágazás előrejelzés(branch prediction),
- utasítássorrend átrendezés, ezen belül például késleltetett elágazás(delayed branch), elágazás összevonás(branch folding),
- pipeline többszörözés.

A hatékonyabb feldolgozást segíti a fordítóprogram megfelelő kialakítása, amellyel a magasszintű nyelv szintaktikájától eltérő módon történhet a fordítás, ami viszont a feldolgozás szempontjából folyamatossá teszi a végrehajtást.

1.Alapmódszer

Az alapmódszer a legegyszerűbb megoldást adja, azaz a proceszszor folyamatosan feltölti a pipeline-t mindaddig, amíg ki nem derül, hogy elágazási utasítás következik és az elágazást kell folytatni. Ekkor a már betöltött, felesleges utasításokat törli a pipeline-ból és elkezdi feltölteni azt az elágazás utasításaival(3-39.ábra).



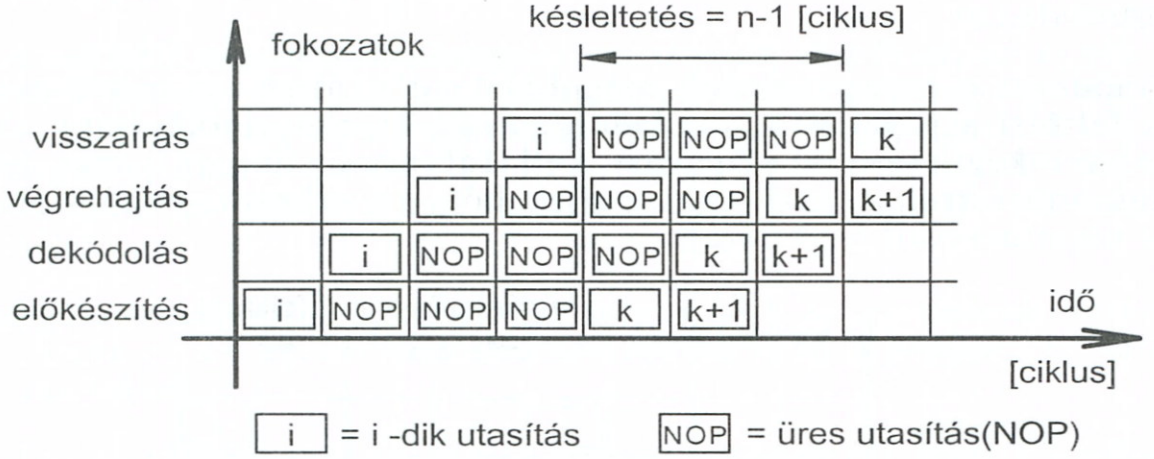
3-39.ábra: *Pipeline törlése*

A módszer alkalmazásának problémája a törölt utasítások által már elvégzett műveletek hatása(mellékhatás - side effect), amelyet a törléssel semlegesíteni kell, azaz vissza kell állítani az eredeti állapotot.

2.Késleltetett elágazás(delayed branch)

A késleltetett elágazás, a késleltetett memóriautasítás mellett, a leggyakrabban alkalmazott módszerek egyike. Mivel az elágazás ténye és iránya csak a feldolgozás későbbi, vagy utolsó fázisában válik egyértelművé, az elágazási utasítást közvetlenül követő helyekre más, az elágazástól függetlenül végrehajtható,vagy üres utasításokat helyezve, a pipeline-ban nem keletkezik üres hely (bubble). A 3-40.ábra mutatja be az alaphelyzetet, amikor a teljes késleltetési rés NOP utasításokkal van kitöltve és az elágazás csak az utasítás teljes feldolgozása után kezdődik.

Természetesen kérdés az, hogy mely utasítások helyezhetők át az elágazási utasítás mögé, amelyek végrehajtására mindig sor kerül függetlenül az elágazás irányától. Ha más lehetőség nincs, akkor a fordítóprogram üres utasításokat(NOP) helyez el a 'késleltetési rés'(delay slot)-ben. A késleltetési rés mérete a pipeline-tól függ, azaz attól, hogy mikor kezdi a processzor az elágazás célutasítását a pipeline-ba tölteni, de értéke $n-1$ ciklusnál nem több. Ha a processzor nem az elágazási utasítás teljes feldolgozása után indítja a célutasítást, hanem az ugrás tényleges címének a kidolgozása után, akkor a késleltetési rés értéke kisebb lesz. Minél nagyobb n értéke, annál nehezebb megfelelő számú, áthelyezhető utasítást találnia a fordítóprogramnak. Ezekben az esetekben NOP utasítással tölti ki a késleltetési rést.



3-40.ábra: *A pipeline ütemezése késleltetett elágazás esetén*

Ennél a megoldási módnál is fontos, hogy a fordítóprogram és a hardver azonos előfeltevésekkel dolgozzon. Azaz a fordítóprogram tudja azt, hogy a hardver mit tekint alaphelyzetnek: az elágazás nem teljesülését, vagy teljesülését.

b.)Adatok felhasználása

A folyamatos és párhuzamos utasításfeldolgozás nagy problémája az utasítások igényütközése a feldolgozandó adatok(operandusok) kapcsán. Előfordulhat, hogy egy sorrendben előrébb álló utasítás, egy őt követő utasítás által már módosított adatot akar olvasni és módosítani. Ezeket az ütközéseket, 'veszélyhelyzeteket' nevezik **hazard**-oknak.

Ilyen helyzetet okoz például a következő utasításokkal leírt programrészlet:

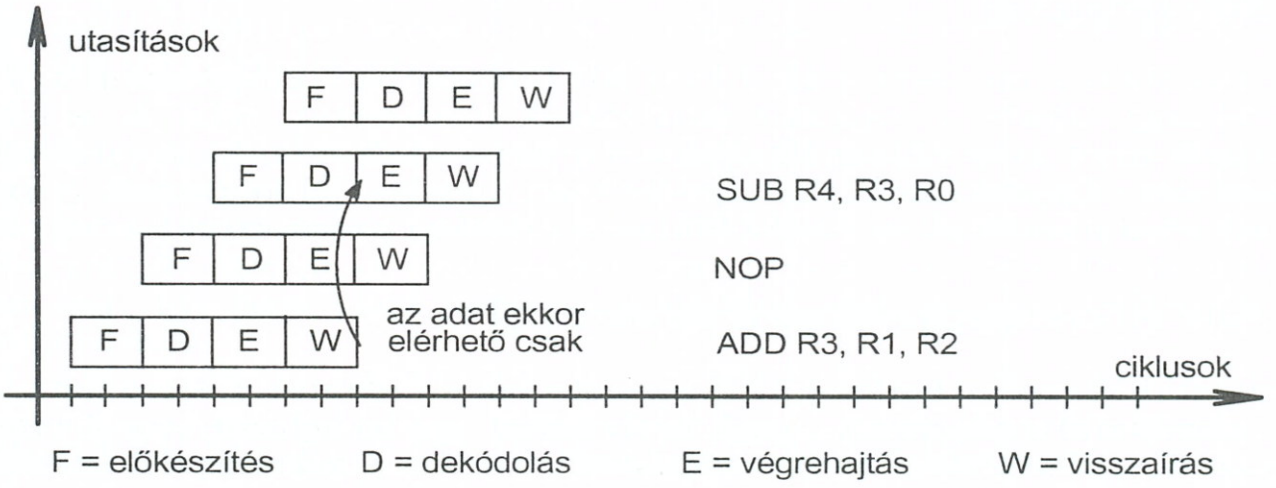
```
add R3,R1,R2        [R1]+[R2] ⇒ R3
sub R4,R3,R0        [R3]-[R0] ⇒ R4
```

Az első utasítás által feltöltött R3 regiszter tartalma, a második utasításban még nem áll rendelkezésre a szükséges fázisban. Az utasítások feldolgozása közben kialakuló helyzetet mutatja be a 3-41.ábra.

A problémát üres utasítások(NOP) beiktatásával, vagy utasítássorrend módosítással lehet feloldani. Az így előállt utasítássorozat:

```
add R3,R1,R2        [R1]+[R2] ⇒ R3
nop
sub R4,R3,R0        [R3]-[R0] ⇒ R4
```

Hasonló adatütközést okozhat az adatok memóriába töltése és annak azt követő felhasználása is. Mivel a tárolóutasítások eredménye azonnal nem áll rendelkezésre, ezért itt is üres utasítások, vagy más, a betöltéstől független hasznos utasítások beiktatásával lehet biztosítani a pipeline folyamatos működését.



3-41.ábra: *Adathasználat ütközése egymás utáni utasításokban*

Az ütközések típusai

Négyféle hazardot lehet megkülönböztetni(az elnevezések az utasításoknak és nem maguknak az elemi műveleteknek az időbeli sorrendjére utalnak):

- **írás utáni írás**(write after write), amely esetben a sorrendben második utasítás adatmódosítása, írása után módosítaná az adatot az első utasítás;
- **írás utáni olvasás**(read after write), amelynél a 2.utasítás azt megelőzően olvas ki egy adatot, hogy az 1.utasítás az adatmódosítást már elvégezte volna;
- **olvasás utáni írás**(write after read) esetében a 2.utasítás azt megelőzően írja felül az adatot, hogy azt az 1.utasítás kiolvasta volna;
- **olvasás utáni olvasás**(read after read) féle hazard nem okoz általában problémát, mivel a kiolvasás nem módosítja az adatot, így az olvasás sorrendje lényegtelen.

Szokásos megoldások

A 'hazard'-ok előfordulásának megakadályozására a legegyszerűbb módszer, az utasítás adatcsatornába való belépését megakadályozni, vagy továbbhaladását felfüggeszteni(interlocking), ha az utasítássor elemzése erre utal, azonban ez jelentős mértékben csökkenti a hatékonyságot.

1.Utasítás átrendezés(instruction scheduling)

Az adatütközés feloldásának egyik legegyszerűbb módszere az utasítások sorrendjének olyan átrendezése - ha lehet! -, vagy üres utasítások(NOP) beiktatása, amely után a kellő időpontban, fázisban a kívánt adat rendelkezésre áll. Erre szolgáltat példát a 3-41.ábra esete is.

2.Scoreboarding

Jobb megoldás a processzor által használt regisztereket kiegészíteni egy jelzőbittel(tag), amely a regiszter tartalmának módosítását jelzi és a követő utasítások csak akkor használhatják a regiszter tartalmát, ha a jelzőbit ezt engedélyezi. Egyébként késleltetni kell az utasítás feldolgozását. (Ezt az eljárást alkalmazzák pl. az MC88100-as, az i860-as processzorok esetében.)

3.Adat előreengedés(internal forwarding)

Az egymást követő utasítások által ugyanannak az adatnak a használata, nem csak utasítás átrendezéssel, vagy késleltetést eredményező 'scoreboarding' módszerrel oldható meg, hanem megfelelő hardver struktúrával biztosítható az adatok azonnali továbbítása a processzoron belül, a pipeline egyes fokozatai között.

A módszer előnye, hogy nem kell megvárnia egy követő utasításnak azt, hogy a megelőző utasítás az adatot tárolja valamelyik regiszterben, majd azt onnét kiolvashassa, hanem azonnal rendelkezésére áll az a feldolgozáshoz.

Az utasítások feldolgozása közben előálló helyzeteket aszerint lehet csoportosítani, hogy milyen műveletet kellene a két utasításnak egymás után elvégeznie. Így például előfordulhat, hogy a tárolóba vitel után ismételten ki kell olvasni egy adatot(*store-fetch*), vagy egymás után kétszer kell kiolvasni(*fetch-fetch*), vagy visszaírni(*store-store*) ugyanazt az adatot. Mindezeket el lehet kerülni a forwarding technikával.

Problémát jelent a bonyolult vezérlő logika, mégis egyre gyakrabban alkalmazzák, különösen a RISC processzorok esetében, mint például a SPARC, MIPS, vagy Amd29000-es processzoroknál.

4.Memóriautasítás

A memóriautasítások által okozott adatütközési probléma *egyik* lehetséges megoldási módját a bevezető részben, a 3-38.ábra segítségével már bemutat-

tuk. Ennek során a memóriautasítás utáni késleltetési résbe üres, vagy hasznos utasításokat helyez a fordítóprogram.

A probléma *másik* lehetséges megoldását adja a követő utasítás késleltetése a megfelelő időpontig. Ez hardver módszerrel(hardware interlocking) oldható meg, azaz a memóriautasítás belépése után, a processzor leállítja a pipeline feltöltését mindaddig, amíg a betöltött adat a következő utasítás által megszabott időpontban rendelkezésre nem áll.

c.)Erőforrások igénybevétele

Az erőforrások használatának problémája abból származik, hogy a párhuzamosított folyamatok igen gyakran ugyanazt az erőforrást kívánják igénybevenni feldolgozásukhoz(pl. leggyakrabban a memória okoz ilyen gondot). Ennek egyik lehetséges megoldása az erőforrások többszörözése, amely nyilvánvalóan többletköltséget eredményez.

A memóriának, mint erőforrásnak a használatát javítja az a megoldás, amit **'memory interleaving'**-nek(4.5.pont) neveznek. Ennek lényege az, hogy a memóriát elérés, címzés szempontjából több részre osztják és az egymást követő(egyidejű) tárolóhoz fordulásokat más és más tárolórészhez irányítja a processzor. Így nem okoz gondot egyidejűleg több utasítás előkészítése a feldolgozáshoz.

Az igények sorbaállítása egy másik lehetőség az azonos erőforrások iránti igények kiszolgálására. Ez azonban nem gyorsítja a feldolgozást.

3.5.3.Műveletvégzés gyorsítása

Az utasítások végrehajtási folyamata párhuzamosításának egy önálló részét képezi a műveletek végrehajtásának gyorsítása. Ennek elsősorban akkor van jelentősége, ha adatok sorozatán(vektorokon) kell ugyanazon, vagy különböző műveleteket elvégezni. Ez a helyzet, az ilyen típusú feladatokhoz készült gépeknél, az ú.n. vektorszámítógépeknél(vector computer), elsődlegesen megoldandó feladat.

Normál mikroszámítógépeknél is célszerű kihasználni a lehetséges párhuzamosításokat, ha számsorozatokkal kell dolgozni, ha több funkciós műveletvégző egységeket használhatunk.

Mind a fixpontos, mind a lebegőpontos műveleteknél lehetőség van a folyamat elemi lépésekre bontására és ezeknek adatcsatornába (pipeline) szervezésére. Pl. az összeadásra bitenként szervezett adatcsatorna segítségével az átvitelképzésből származó késleltetéseket csökkenteni lehet; vagy a regisztertartalom többszörös léptetésének egylépéses megoldása a *barrel-shifter*-rel.

3.6.RISC PROCESSZOROK

A RISC processzorok kialakulásának körülményeit a 3.2.2.c.pontban már bemutattuk, ezért itt csak a legfontosabb okokat ismételjük meg röviden.

A kezdeti időszak számítógépeinek utasításkészlete a '60-as évek elejéig, igen egyszerű és kevés számú utasításból állt. Az adatok elérésére, memóriából való kikeresésére is kevés számú címzési mód volt használható. A jóminőségű fordítóprogramok hiánya miatt, a programozóknak programjaikat gépi szinten kellett megírni. Az utasítások műveleti vezérlését huzalozott módon oldották meg. Ebben az értelemben ezek a korai gépek csökkentett utasításkészletű, RISC gépek voltak.

A '60-as évek közepétől növekedett az utasítások bonyolultsága, azaz egyre több funkciót bíztak egyetlen utasításra, emellett a használható utasítások száma is növekedett(több, mint 200-250 utasítás). Ennek az oka, többek között, a mikroprogramozott műveleti vezérlés elterjedése volt. Egy másik tényező, amely a bonyolult gépi utasítások kialakulása felé vezetett, a magasszintű programozási nyelvek széleskörű elterjedése. Ekkor alakultak ki az összetett utasításkészletű CISC(Complex Instruction Set Computer) számítógépek.

A '70-es évek közepére, a kialakuló igen bonyolult mikroprogramok(gépi utasítások) már akadályozták a feldolgozás gyorsítását. Emellett egyre nehezebb lett az egyre bonyolultabb mikroprogramok hibátlan előállítása, tesztelése. Ezek a tények vezettek el az egyszerűsítés szükségességének felismerése felé.

Felmerült a lehetősége annak, hogy a felhasználói programokat közvetlenül a mikroutasítások szintjére kellene fordítani és elhagyni az egész mikrovezérlő rendszert. Ez az elképzelés végül is a RISC processzorok alapgondolata, azaz egyszerű és viszonylag kevés számú utasítás használata, elhagyva a mikroprogramot az utasítások végrehajtásából.

A RISC processzorok gyakorlati megvalósítását a fordítóprogramok(fordítási technikák) fejlődése is segítette, amelyek már a mikroprogram szintjén is hasonló minőségű programot voltak képesek előállítani, mint amilyeneket egy jó programozó tud elkészíteni.

A RISC processzorok kifejlesztésének célja végső soron a feldolgozás gyorsítása volt. A gyorsítást az alábbi területeken végzett fejlesztések segítették elő:

- a processzor számára a lehető legegyszerűbb hardver kialakítása a maximális műveleti sebesség elérése érdekében;
- a felhasználói alkalmazási területek, módszerek elemzése alapján, a ritkán igénybevett utasítások elhagyása az utasításkészletből, a bonyolult utasítások egyszerűsítése;
- a feldolgozási, végrehajtási adatútvonalak elemzése alapján, a kapcsolódó utasítások optimális kialakítása;

- az áramköri lapka kialakításához a VLSI(Very Large Scale Integration) technológia használata; a komplex utasítások elhagyásával, a lapkán hely szabadul fel, ezt regisztertárak, cache-tárak kialakítására lehet felhasználni;
- a szilicium(Si) alapú lapkák helyett gallium-arzenid(GaAs) alapú lapkák használata, amelyek kisebb integráltsági fok mellett ugyan, de nagyobb sebességet eredményeznek;
- optimalizáló fordítók kidolgozása és alkalmazása.

A RISC processzorok **leglényegesebb jellemzői** D.Tabak(1986) értékelése alapján:

- A CISC processzorokkal szemben kis számú utasítás(<128) és a címzési módok igen szűkített mértékű(2-4) használata.
- Az utasítások rögzített hosszúságúak, szemben a CISC processzorok változó hosszúságú utasításaival(pl.: az i386/486-osnál 1-17 byte között).
- Közvetlen memóriahasználatra csak két(LOAD és STORE) utasítás áll rendelkezésre, amelyek 1, 2, vagy 4 byte-ot mozgatnak egyszerre.
- Az utasítások végrehajtásához nincs mikroprogram; a gépi kódú utasítások a CISC processzorok vertikális mikroutasításainak megfelelő bonyolultságúak.
- Az utasítások végrehajtásához egy adatútvonal ciklust(kiolvasás regiszterből - műveletvégzés az aritmetikai egységben(ALU) - visszaírás a regiszterbe) használnak fel, szemben a CISC processzorok több gépi ciklusból álló végrehajtási időtartamával.
- Az utasítások átlapolt, adatcsatornás(pipelining) feldolgozása; a LOAD és a feltételes ugró utasítások esetében késleltetett(delayed) végrehajtás, ami azt jelenti, hogy a LOAD utasítás hosszabb végrehajtási időtartama miatt, az adatcsatorna továbbhaladását felfüggesztik az utasítás befejeztéig, vagy ezt az időtartamot más lépésekkel tölti ki a fordítóprogram.
- Nagy regisztertárak kialakítása a memóriához fordulás gyakoriságának csökkentésére.
- A gépi utasítások primitívsége és az optimális sorrend kialakítása miatt, a fordítóprogram igen bonyolult lesz, sok elemzést kell elvégeznie és emiatt a fordítás időtartama megnő a CISC processzorokhoz tartozó fordítóprogramokkal szemben. A fordítóprogram feladata a késleltetett LOAD és ugró utasítások mögötti helyeket kitölteni olyan lépésekkel, amellyekkel meg lehet akadályozni az adatcsatorna továbbhaladásának felfüggesztését.
- A processzor és a gép struktúrájának gyorsítást segítő kialakítása. Az adatok gyorsabb elérését külön program- és adattároló, továbbá a processzoron belül olyan regisztertár kialakítása segíti, amely háromcímes utasításokkal elérhető.

D.Tabak által megfogalmazott jellemzők némelyike a mai processzorok esetében már nem érvényesül teljes mértékben.

A CISC és RISC processzorok leglényegesebb jellemzőit, az összehasonlíthatóság végett, a 3-14.táblázatban foglaljuk össze:

3-14.táblázat: *CISC - RISC processzorok összehasonlító táblázata*

	CISC processzorok	RISC processzorok
1	Összetett utasítások, melyek végrehajtása több gépi ciklust igényel	Egyszerű utasítások, melyek végrehajtása 1 gépi ciklust igényel
2	Bármely, erre alkalmas utasítás igénybe veheti a tárolót	Csak a LOAD/STORE utasítások fordulhatnak a memóriához
3	Az adatcsatornás(pipelining) feldolgozás kismértékű	Erőteljes adatcsatornás(pipe-lining) feldolgozás
4	Utasításvégrehajtás mikroprogram által vezérelt	Huzalozott utasításvégrehajtás
5	Változó hosszúságú utasítások	Rögzített utasításhossz
6	Sokféle utasítás és címzési mód	Kevés utasítás és címzési mód
7	Bonyolult mikroprogram	Bonyolult fordítóprogram
8	Kis számú regiszter	Nagy méretű regisztertár

Az előnyök mellett vannak olyan jellemzők is, amelyek a RISC processzorok hátrányául róhatók fel.

- Az utasításkészlet kialakítása a felhasználói terület elemzése után alakul ki és ez a feladatra orientáltság miatt hátrányos.
- Az egyszerűbb utasítások miatt a lefordított program hossza nagyobb, mint egy hasonló CISC processzorhoz tartozó programé.
- A működéssel kapcsolatos megbízhatósági, védelmi kérdések megoldását a hardver nem segíti, azt szoftver úton kell megvalósítani.

3.7.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Vázolja fel a központi egység elemei közötti logikai kapcsolatot tükröző rajzot!
2. Milyen részei vannak egy hagyományos struktúrájú számítógép központi egységének? Ismertesse a központi egység lényeges elemeit!
3. Mi határozza meg, hogy egy tárolt jelsorozat utasítás-e, vagy adat?
4. Hogyan értelmezzük az $A=(a_{-m}\ a_{-m+1}\ \ldots\ a_{-1}\ a_{-0},\ a_1\ a_2\ \ldots a_n)$ jelsorozatot, ha az egy számot ír le?
5. Milyen írásmód a fixpontos forma? Adjon példákat a fixpontos írásmódú számokra!
6. Milyen írásmódú a lebegőpontos forma? Adjon példákat a lebegőpontos írásmódú számokra!
7. Mit értünk normalizált írásmód alatt? Adjon példákat a normalizált formájú számadatokra!
8. Milyen írásmódú szám a következő: 254.678 ?

9. Értelmezze a következő jelsorozatot nyolcas számrendszerben: (5314)
10. Értelmezze a következő jelsorozatot tizenhatos számrendszerben: (5314)
11. Miért használják a számadatok tárolásához a kettes számrendszer szerinti kódolási formát?
12. Mit értünk túlcsordulási, illetve alulcsordulási hiba alatt? Mi az oka ezek bekövetkezésének?
13. A számadatok milyen tárolási formáit alkalmazzák a számítógépeknél?
14. Rajzolja fel, hogy a tárolt jelsorozat hogyan értelmezhető fixpontos egészszámként!
15. Rajzolja fel, hogy a tárolt jelsorozat hogyan értelmezhető fixpontos törtszámként!
16. Rajzolja fel, hogy a tárolt jelsorozat hogyan értelmezhető lebegőpontos számadatként!
17. Mit értünk mantissza alatt?
18. Mit értünk karakterisztika alatt?
19. Helyezze el a **12617_{10}** -es számot, a tízes számrendszer szabályai szerint, egy 8+4 helyiértékes(mantissza helyfoglalása=7 adat- +1 előjelhelyiérték; a kitevő helyfoglalása =3 adat- +1 előjelhelyiérték) tárolóhelyen lebegőpontos forma szerint!
20. Helyezze el egy 3 byte-os tárolóhelyen(2 By a mantissza+előjel részére; 1 By a kitevő+előjel részére) nullára normalizált formában, **kettes** számrendszerben, **előjeles, abszolútértékes** alakban a következő számot: $-17.675_{10}\ 10^{1}$
21. Miért nem célszerű a negatív számok tárolásához az előjeles, abszolútértékes tárolási módot használni?
22. A negatív számok milyen tárolási formáit alkalmazzák?
23. A negatív számokkal végzett műveletekhez miért használják a kettes komplemenskódot ?
24. Alakítsa át a következő, kettes számrendszerben ábrázolt előjeles, abszolútértékes számadatot egyes komplemenskódú formába, 8 helyiérték figyelembevételével: A=(1 110 0010)
25. Alakítsa át a következő, kettes számrendszerben ábrázolt előjeles, abszolútértékes számadatot kettes komplemenskódú formába, 8 helyiérték figyelembevételével: A=(1 110 0010)
26. Helyezze el egy 3 byte-os tárolóhelyen(2 By a mantissza+előjel részére; 1 By a kitevő+előjel részére) nullára normalizált formában, **kettes** számrendszerben, **kettes komplemens** kódban a következő számot: $-15.675_{10}\ 10^{1}$
27. Helyezze el egy 3 byte-os tárolóhelyen(2 By mantiszsza+előjel részére; 1 By kitevő részére, m=8) nullára normalizált formában, **kettes** számrendszerben, **előjeles, abszolútértékes,** a kitevőt **2^{m-1} többletes** alakban, a következő számot: -196.675_{10}
28. Milyen formában értelmezi az IEEE 754-es szabvány a számokat?
29. Rajzolja fel egy jelsorozat értelmezési módját az IEEE 754-es lebegőpontos szabvány szerint. Milyen részei vannak a jelsorozatnak?
30. Milyen lehetséges adatméretekkel foglalkozik a szabvány?

31. Milyen adatformátumokat határoz meg az IEEE 754-es szabvány?
32. Mit értünk denormalizált adatformátum alatt?
33. Mit értünk 'nem-szám' adatformátum alatt?
34. Mikor használják a számadatok tárolására(kódolására) a tízes számrendszert utánzó kódrendszereket?
35. Milyen módon kell előállítani a BCD kódú jelsorozatot?
36. Írja fel a tízes számrendszerbeli 3764-es számérték BCD kódú formáját!
37. Alfanumerikus adatok tárolásához milyen kódrendszereket alkalmaznak?
38. Mi a lényege az adatok ASCII kódrendszer szerinti tárolási(kódolási) formájának?
39. Milyen egyéb adattárolási formákat ismer és mi ezek lényege?
40. Milyen részekből áll egy gépi kódú utasítás? Mi az egyes részek funkciója?
41. Mire szolgál a gépi kódú utasítás műveleti jelrésze?
42. Mire szolgál a gépi kódú utasítás címrésze?
43. Mit befolyásol az utasítások hossza?
44. Mit befolyásol az utasításkészlet?
45. Mely processzorok esetében használnak változó hosszúságú utasításokat?
46. Mely processzorok esetében használnak azonos hosszúságú utasításokat?
47. Általános helyzetben, milyen címek ismerete szükséges egy utasítás végrehajtása során?
48. Milyen megoldásokkal csökkenthető a négycímes utasítás címeinek száma?
49. Az utasításszámláló regiszter(PC) bevezetése milyen programtárolási következménnyel jár?
50. Milyen megoldással vezethető be a nullacímes utasításszerkezet?
51. Soroljon fel néhány okot, amely miatt címmódosítási eljárásra van szükség!
52. Milyen címzési módokat használnak?
53. Milyen címzési mód az abszolút címzés?
54. Adjon példát az abszolút címzési módra(rajzzal)!
55. Miért nem jó az abszolút címzési mód használata?
56. Milyen címzési mód a relatív címzés?
57. Adjon példát a relatív címzési módra(rajz segítségével)!
58. Miért használják a relatív címzési módot?
59. Milyen változatait használják a relatív címzésnek?
60. Milyen címzési mód az indirekt címzés?
61. Adjon példát az indirekt címzésre(rajz segítségével)!
62. Milyen címzési mód a közvetlen adatcímzés?
63. Milyen esetekben célszerű használni a közvetlen adatcímzést?
64. Adjon példát a közvetlen adatcímzésre(rajz segítségével)!
65. Milyen címmódosítási eljárás az indexelés?
66. Milyen esetekben célszerű az indexelést alkalmazni?
67. Adjon példát az indexelés alkalmazására(rajzzal)!
68. Mi a különbség az indexelés és a relatív címzés között?
69. Milyen utasításcsoportokat ismer?

70.Milyen utasítások tartoznak az átviteli utasítások közé?
71.Mely utasítások alkotják a műveleti utasítások körét?
72.Milyen utasítások sorolhatók a vezérlő utasítások közé?
73.Mit értünk utasításkészlet alatt?
74.Milyen jellemzőkkel értékelhető egy processzor utasításkészlete?
75.Milyen utasításkészlet jellemző a CISC processzorokra?
76.Milyen utasításkészlet jellemző a RISC processzorokra?
77.Milyen műveletek végrehajtására alkalmas a processzor?
78.Mért vezethetők vissza az aritmetikai műveletek a logikai műveletekre?
79.Milyen számrendszerben ábrázolva célszerű végrehajtani az aritmetikai műveleteket?
80.Számítsa ki az A=21+17 összeget kettes számrendszerben, 1 byte-os(8 bites) tárolóhelyet és fixpontos egész tárolási módot feltételezve!
81.Miért használjuk a kivonások elvégzéséhez a komplemens-kódot?
82.Számítsa ki az A=21+(-17) összeget kettes számrendszerben, 1 byte-os(8 bites) tárolóhelyet és fixpontos egész tárolási módot feltételezve!
83.Rajzolja fel a fixpontos számok összeadásának és kivonásának közös algoritmusát!
84.Milyen műveletekre vezethetők vissza a szorzási és az osztási műveletek?
85.Lebegőpontos számok összeadásakor, ha a karakterisztikák nem egyeznek, melyik lesz az eredmény karakterisztikája, a kisebb, vagy a nagyobb értékű?
86.Lebegőpontos számok összeadásakor, ha a karakterisztikák nem egyeznek, melyik számnak a mantisszáját kell léptetni és milyen irányban?
87.Rajzolja fel a lebegőpontos számok összeadásának és kivonásának közös algoritmusát!
88.Hogyan történik az átvitelképzés két, tízes számrendszer alapján kódolt, BCD kódú szám összeadásakor?
89.Számítsa ki a B=325+189 összeget, a számokat BCD kódban ábrázolva!
90.Milyen logikai alapműveleteket ismer?
91.Írja fel a logikai ÉS(AND)-, a logikai VAGY(OR)- és a logikai NEM(NOT)-műveletek 'igazságtáblázat'-át!
92.Melyek a leggyakrabban használt összetett logikai műveletek?
93.Mire használják a NEM-VAGY és a NEM-ÉS műveleteket? Írja fel ezek 'igazságtáblázat'-át!
94.Írja fel az antivalencia(EXOR)- és az ekvivalancia-művelet 'igazságtáblázat'-át!
95.Írja fel az ekvivalencia-, és az antivalencia-műveletet az alapműveletek segítségével!
96.Mire használhatók a logikai függvények?
97.Írja fel a teljes összeadó 'igazságtáblázat'-át és kimeneteinek logikai függvényét az alapműveletek segítségével!
98.Milyen fő részei vannak az aritmetikai-logikai egységnek?

99. Melyek a legfontosabb állapotjelző bitek és melyik mire szolgál?
100. Vázolja fel a teljes összeadó logikai sémáját!
101. Vázolja fel egy 1-bites ALU logikai vázlatát!
102. Melyek az utasításfeldolgozás elemi lépései és mi ezek sorrendje?
104. Vázolja fel a hagyományos, Neumann-elvű számítógépek(soros) utasításfeldolgozásának logikai vázlatát!
105. Mit értünk a műveleti vezérlés alatt és milyen lehetséges formáit ismeri?
106. Mit értünk huzalozott műveleti vezérlés alatt?
107. Mit értünk mikroprogramozott műveleti vezérlés alatt?
108. Mi a mikroprogram és a mikroutasítáoknak milyen utasításszerkezetei használatosak?
109. Írja fel az LDA B(hatása: [B] ==> AC) utasítás feldolgozásának elemi fázisait.
110. Milyen szerkezetű a horizontális mikroutasítás?
111. Milyen utasítás a vertikális mikroutasítás?
112. Milyen tárolási mód a mikroutasítások kétszintű tárolása?
113. Vázolja fel a mikroprogramvezérelt gép logikai vázlatát!
114. Mi a kapcsolat az utasításkészlet, utasításszerkezet és a processzorok struktúrája között? Milyen a CISC és a RISC processzorok vezérlési elve?
115. Milyen lehetőségeket ismer a számítógépek folyamatainak gyorsítására?
116. Miből adódik az utasításfeldolgozási folyamat párhuzamosítási lehetősége?
117. Mit értünk a 'pipelining' fogalom alatt?
118. Milyen problémák jelentkeznek az utasításfeldolgozás folyamatának párhuzamosítása során?
119. Vázolja fel az utasítások pipeline feldolgozásának idődiagramját!
120. A 'pipeline' milyen működtetési(vezérlési) módjait ismeri és mi ezek lényege?
121. Mely utasítások végrehajtása zavarja meg leginkább a pipeline folyamatos működését?
122. A memóriahivatkozású(load/store) utasítások esetében milyen módszereket alkalmaznak a feldolgozás folyamatosságának biztosítására?
123. Vázolja fel a pipeline terhelési diagramját a memória utasítások esetében, ha váróciklusokkal biztosítják a folyamatosságot!
124. Pipeline feldolgozásnál, mit értünk késleltetett 'load' utasítás alatt?
125. Vázolja fel a pipeline terhelési diagramját a késleltetett 'load' utasítás alkalmazása esetére!
126. Mit értünk 'késleltetési rés' alatt és miért van rá szükség?
127. Milyen utasításokkal tölthető fel a késleltetési rés?
129. Miért okoz gondot az elágazási utasítások feldolgozása a pipeline működésének folyamatosságában?
130. Mit értünk 'mellékhatás'-ok alatt az elágazási utasítások feldolgozása kapcsán?

131. A feltétel nélküli vezérlésátadó utasítások feldolgozásánál milyen módszereket alkalmaznak leginkább?
132. A feltételes elágazási utasítások feldolgozásánál milyen módszereket alkalmaznak a pipeline működésének folyamatossága érdekében?
133. Mit értünk a statikus előrejelzési módszerek alatt és milyen változatait ismeri?
134. Mit értünk a dinamikus előrejelzési módszerek alatt és milyen változatait ismeri?
135. Pipeline feldolgozásnál, milyen módszer a 'késleltetett elágazás' módszere?
136. Milyen utasítással tölthető fel az elágazási utasítás késleltetési része?
137. Vázolja fel a pipeline terhelési diagramját a késleltetett elágazási utasítás feldolgozására!
138. Mit értünk a pipeline feldolgozás kapcsán, az adatütközés fogalma alatt és milyen típusait ismeri?
139. Mit értünk 'hazard' alatt?
140. Miből adódik az adatok használatának ütközése a pipeline feldolgozás során?
141. Az adatütközések feloldására milyen módszereket alkalmaznak?
142. Milyen tárolási módszer a 'scoreboarding' ?
143. Milyen módszer a 'forwarding', vagy adatelőreengedés?
144. Hogyan lehet párhuzamosítani a műveletvégrehajtást? Milyen lépésekre bonthatók a lebegőpontos műveletek?

4. FEJEZET TÁROLÓKEZELÉS

A tárolók, a processzoron kívül, a számítógépek legfontosabb erőforrásainak számítanak, ezért azok használatának hatékony formáit külön célszerű vizsgálni. A teljesítmény növelése szempontjából, különösen a processzorközeli tárolóhelyek(regisztertárak, cache-tárak) játszanak kiemelt szerepet. A főtár és a háttértárolók fizikai megvalósításával, adattárolási módszereivel itt nem foglalkozunk, azok külön fejezet tárgyát képezik.

4.1.TÁROLÁSI ALAPFOGALMAK

A számítógépek teljesítményének növekedésével, a használt tárolók jobb kihasználása és strukturális illesztése a megnövekedett teljesítőképességhez, elsődleges problémává váltak.

Lényeges szempont az utasítások és az adatok használati gyakoriságuknak megfelelő elhelyezése, azaz minél gyakrabban van szükség rájuk, a processzorhoz annál közelebb és a processzor által annál rövidebb idő alatt elérhetőknek kell lenniük. Fordítva is igaz, hogy ha nincs szükség valamilyen programrészre, vagy adatra, akkor az kerüljön ki a processzor közvetlen közeléből, átadva a helyet a nála fontosabb adatnak.

Ugyanakkor a feldolgozásra váró adatok mennyisége és a feldolgozást végző program mérete miatt, egyre több tárolóhelyre van szüksége a gépnek.

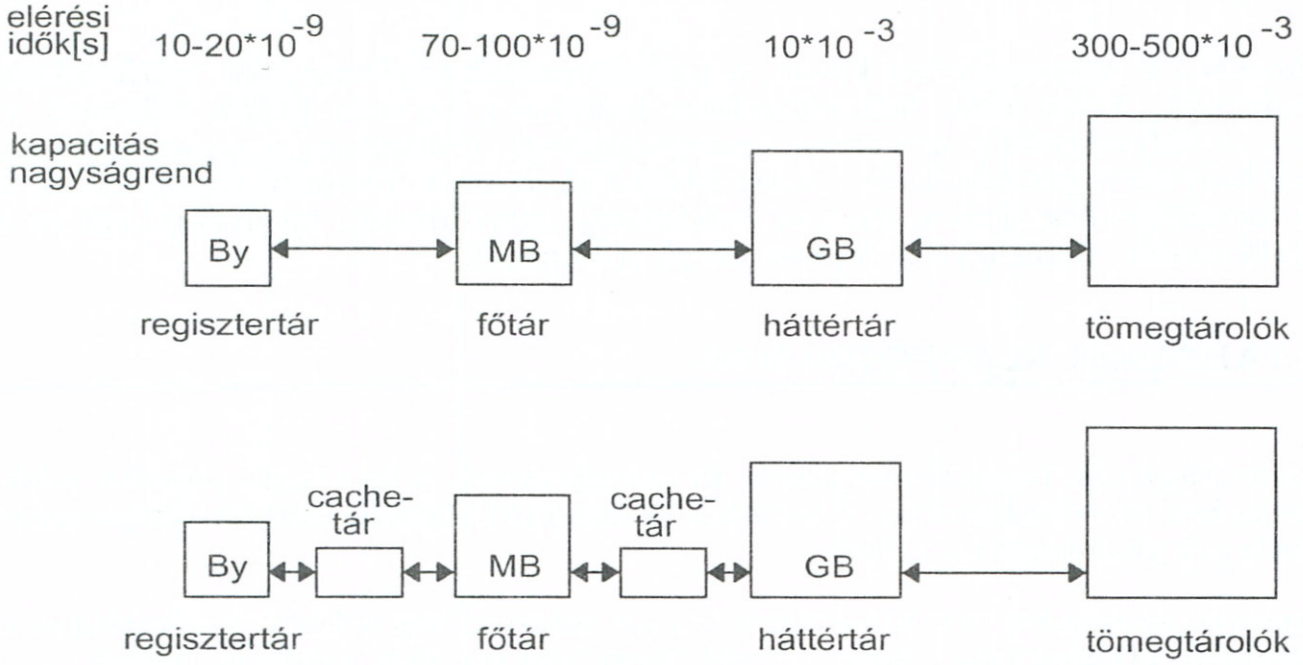
A nagy kapacitású tárolók elérési ideje általában nagyságrendekkel nagyobb, mint a processzor közvetlen közelében lévő tárolóhelyeké, ezért az adatok megfelelő elhelyezése és elérése a különböző tárolóeszközökön, életbevágóan fontos.

Az adatok kellő időben és idő alatt való elérésének és elhelyezésének strukturális eszköze a **tárhierarchia**. A különböző tárolókezelési eljárások, a növekvő tárigény mellett, a processzor köré kiépített tárhierarchia hatékony kezelését vannak hivatva megoldani.

A processzorhoz legközelebb a **regiszterek** vannak, amelyek viszonylag kevés(max. néhány 100 byte) adat befogadására alkalmasak, ugyanakkor az elérési idejük a legkisebb(10-30 nsec; 1 nsec=10^{-9} sec). A program végrehajtásához közvetlenül szükséges programrészek és adatok tárolására a **főtár**, központi memória szolgál, amelynek kapacitása néhány Mbyte és elérési ideje 70-100 nsec. Az éppen nem szükséges adatok tárolását a nagykapacitású **háttértárolók** biztosítják több Gbyte-os tárolóképességgel és 10-30 msec elérési idővel. Ezt egészítik ki az úgynevezett **tömegtárolók**(mágneslemez-, mágnesszalagegységek, optikai tárolók, stb.), amelyek összkapacitása végül

is csak az igényektől és a lehetőségektől függ. Ezek elérési ideje nagyon változó, néhány 100 msec-tól néhány percig is terjedhet.

Az adatok csak a tárolóhierarchián végighaladva kerülhetnek feldolgozásra a processzorban(4-1.ábra).



4-1.ábra: *Mikroszámítógépek tárolóhierarchiája*

Az adatáramlás folyamatosságának biztosítására, pufferelési céllal, az egyes tárolási szintek közé kisebb kapacitású, a felhasználó számára 'láthatatlan', gyors működésű tárolók kerülnek. Ezeket a tárakat nevezik **cache-tárak**nak, gyorsító táraknak.

A főtár processzor által közvetlenül címezhető tárolóterületének címei alkotják a **fizikai címek tartományát** és a mögötte lévő, létező (installált), vagy nem létező tárolóeszköz a **fizikai tároló**(2.1.2.b.pont).

A számítógépek által használt tényleges tárolóterület nagysága azonban lényegesen nagyobb, mint a processzor által az előbbiekben meghatározott módon megcímezhető tárolóterület. Azért, hogy a számítógép körüli teljes tárolótartomány azonos módon legyen címezhető, a programok a teljes tárolóterületet, vagy legalábbis annak nagy részét lefedő címtartománnyal dolgozhatnak, amelyet **logikai**, vagy **virtuális címtartománynak** nevezünk. A programutasításokban használt különböző címmódosító eljárások alapján kiszámított címet **tényleges címnek**(effective address) szokás nevezni, amely végső soron egy virtuális cím. A program által használt virtuális címek és a processzor által használható fizikai címek közötti kapcsolatot a tárkezelő rendszernek kell megteremtenie.

Az előbbiek alapján és azt kiegészítve, a **tárolókezelés feladata** általában:

- *egyrészt* az ismertetett struktúrájú tárhierarchia leghatékonyabb működtetése; virtuális címek fizikai címekké konvertálása, lapozás, szegmentálás;
- *másrészt*, a rendelkezésre álló memória szétosztása a feldolgozandó feladatok(task-ok) között;
- *harmadrészt*, a kellő védelem biztosítása az adatok biztonságos feldolgozásához, a nem legális hozzáférési kisérletek megakadályozásához; ez a védelem az alábbi területekre terjed ki:
  - a rendszer(programok) védelme a felhasználótól, azaz attól, hogy szándékosan, vagy véletlenül a gép működtető rendszerében módosításokat végezzen a felhasználói program;
  - az egyik felhasználó feladatának védelme egy másik felhasználó feladatától;
  - a futó program moduljainak, eljárásainak egymástól való védelme, megakadályozandó a modulok, eljárások által használt memóriaterületek átírását;
  - a felhasználók és a programok elválasztása, de annak is a biztosítása, hogy közös eljárásokat is használhassanak.

A tárolókezelés feladatainak megoldására szolgál a processzorok **tárolókezelő egysége**(MMU=Memory Management Unit), amely lehet a processzorba beépített(on-chip) és azon kívüli is(off-chip). Az Intel80286, i386/486-os processzorok beépített, a Motorola MC68020-as processzor önálló MMU-val rendelkezik. A tárolókezelő egység címképzési munkáját segíti a **lapozó cache-tár(**TLB=Translation Lookaside Buffers).

4.2.REGISZTERTÁRAK

A processzorokban egy-egy szónyi(2-4 byte-nyi) adat tárolására mindig használtak gyors működésű tárakat, regisztereket. Ezek

- egyrészt a *felhasználó által közvetlenül elérhető*(user visible register), speciális célú, funkcionális tárolók(pl.: utasításszámláló regiszter[PC], utasításregiszter[IR], indexregiszterek[IX], veremmutató regiszter[SP], feltételregiszter[FLAG], stb.), amelyek mellett egyre több, általános célú, felhasználású regiszter jelent meg. A mai processzorokban és különösen a RISC processzorokban, általában több tucat általános célú regiszter található, amelyek az ú.n. **regisztertárat** alkotják.

  Már itt különbséget kell tenni a regisztertárak és a cache-tárak között. A regisztertárak egy-egy tetszőleges adat befogadására alkalmasak, a regiszterek tartalmát egyenként cserélhetjük, lokális változókként működnek. A regiszterek használatát a fordítóprogram határozza meg. A cache-tárak tartalmát a processzor valamilyen stratégia szerint, de automatikusan és mindig blokkosan, azaz több byte-ját egyszerre mozgatva, cseréli.

- másrészt a *felhasználó által közvetlenül nem*, vagy egyáltalán nem elérhető regiszterek(system registers, control-, statusregisters, stb.), amelyek tartalmát csak a processzor használja feladataihoz.

Az említett regisztertárak, regisztertömbök egyre nagyobb szerepet kapnak a processzorok teljesítményének növelésében. A regisztertárakkal szemben támasztott követelmények:

- az adatforgalom csökkentése a memória és a processzor között, például lokális változók, szubrutin visszatérési címek elhelyezésével;
- minél nagyobb méret(100-500 regiszter);
- 3-címes elérési lehetőség, azaz egyidőben, egy utasításon belül meg lehessen adni a két operandus és az eredmény címét is;
- általános felhasználási célú legyen.

A regisztertömbök tartalmának kezelésére különböző technikákat dolgoztak ki és használnak. A technikák mindegyikére(vagy legalábbis a nagy többségére) jellemző, hogy a regisztertömböt **ciklikus tárolónak** tekinti, azaz a legmagasabb című regiszter után a legalacsonyabb, 000-ás című regiszter következik. A jellemző megoldások:

- nagyobb méretű regisztertár, átlapolódó ablakokkal(**register windowing**); például Sun SPARC processzorok;
- nagyobb méretű regisztertár, ablaktechnika nélkül, regiszter bankokkal (**register banking**); például az i80960 processzorok;
- nagyobb méretű, lineáris regisztertár, például az Amd29000 processzoroknál;
- kisebb méretű, lineáris regisztertár, például MIPS R-sorozat, MC88000 processzorok.

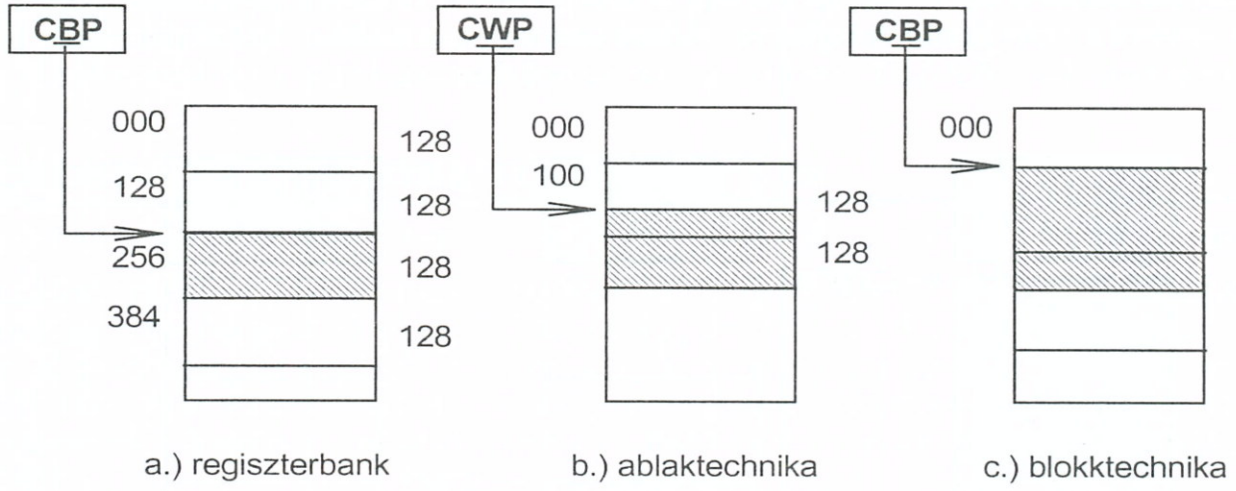
A processzor számára egy feladat(task) feldolgozása közben, általában a regiszterek egy része látható, használható csak. Ennek az 'ablaknak' a kialakítása, kezelése az, amiben az egyes technikák különböznek (4-2.ábra). Az alábbiakban részletesebben is bemutatandó technikák közül, az ablaktechnika (register windowing) használata az, amely szélesebb körben elterjedt.

- A **regiszterbank**(register banking) alkalmazása esetében a regisztertömb nem átlapolódó, azonos méretű részekre, ú.n. 'bank'-okra van felosztva. Egy-egy ilyen bank mérete 2 valamely hatványa. A processzor számára az aktuálisan használt bank kezdetét a bank-mutató (CBP=current bank pointer) jelöli ki.

- Az **ablaktechnika**(register windowing) alkalmazásakor a regisztertömb egy-egy azonos méretű, de átlapolható része látható és használható a processzor számára. Az 'ablak' mérete mindig azonos és 2-nek valamely hatványa. Az aktuális ablak kezdetét az ablak-mutató(CWP=current window pointer) jelöli ki.

- A **blokktechnika**(register blocking) használatakor a regisztertömb tetszőleges méretű, átlapolható részekre van felosztva. A processzor által használt aktuális blokk kezdetét a blokk-mutató(CBP=current block pointer) jelöli ki.

A processzor által használt rész kezdetét kijelölő mutatók utasításokkal elérhetők, módosíthatók; ez egyúttal arra is lehetőséget ad, hogy a regisztertömbben kvázi-veremtárolót(stack tárolót) alakítsunk ki.



4-2.ábra: *Regisztertárak kezelési formái*

a.)Ablaktechnika

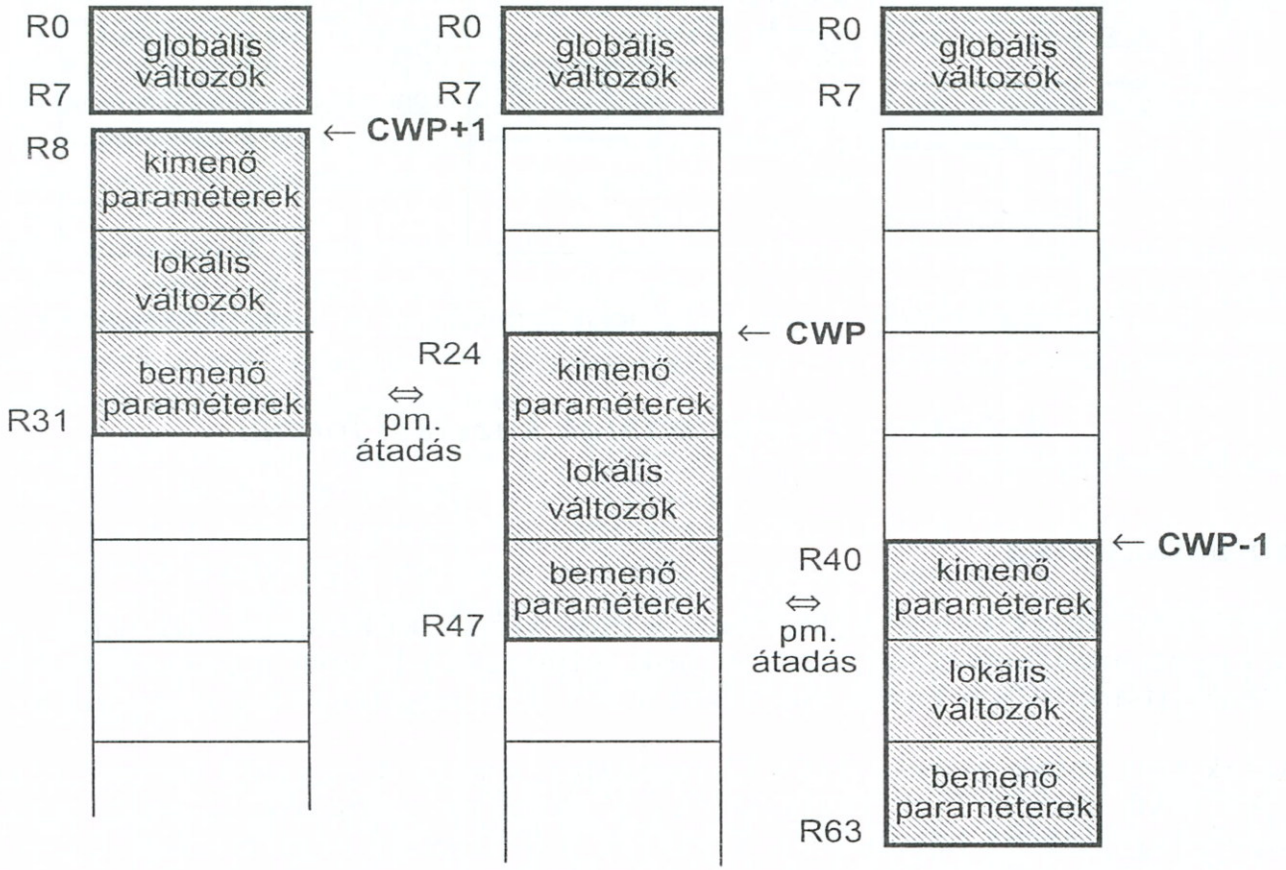
Az említett ablaktechnika(register windowing) különösen alkalmas arra, hogy egyes programrészek(szubrutinok, eljárások) között a paraméterátadást megkönnyítsük. A szokásos módszert a 4-3.ábra segítségével mutatjuk be.

A kialakított ablak mérete 4x8=32 regiszter, amely 4 egyenlő méretű részből áll. Az első rész(0-7-es regiszterek) a globális változók tömbje, amelyet minden programrész használhat. Gyakran a 0-ás című regiszterben fixen, huzalozott módon a 0 számkonstans található meg.

Az ablak 2., 3. és 4.része egy-egy programrész, szubrutin saját változóit tartalmazza, mégpedig oly módon, hogy a 2.tömb 8 regisztere az input változók, paraméterek helye, a 3.tömb 8 regisztere a helyi, lokális változók tárolóhelye, míg a 4.tömb 8 regisztere a kimenő, output változók elhelyezési területe. Ez a megoldás azt eredményezi, hogy miután az egyes programrészek, feladatok ablakai átlapolódnak, az adatok átvétele(a 2.rész regisztereiben) és átadása(a 4.rész regiszteriben) adatmozgatás nélkül történik, az ablakmutató(CWP) értékének átállításával. Ha az átadandó paraméterek száma nagyobb a rendelkezésre álló 7 regiszternél(1 regiszter a visszatérési címet tárolja), akkor a memóriabeli veremtárolóban helyezi el a processzor a további adatokat.

A regiszterek logikai címe(sorszáma) mindig 0-7, 8-31 közé esik, függetlenül az ablak konkrét helyétől(CWP értékétől). Szokásosan R0 tartalma fixen, huzalozott módon 0 értékű, a kimenő paraméterek regiszterei közül az R15-ös, illetve a bemenő paraméterek regiszterei közül az R31-es a (szubrutin) visszatérési címet tárolja. (Egy eljárás hívásnál az R15-ös regiszterből az R31-es logikai című regiszter lesz és fordítva is, az eljárásból való visszatéréskor.)

A regisztertömbben 104-520 db regiszter található és az ablakok száma 6-32 között mozog. A fenti **register windowing** technikát alkalmazzák a Sun Microsystems Inc. SPARC mikroprocesszoraiban is.



4-3.ábra: *Regiszter-ablaktechnika használata*

b.)Blokktechnika

Az ablaktechnika előnyös tulajdonságai mellett nagy hátránya, hogy rögzített ablakméretekkel dolgozik, így időnként kevés a hely, időnként pedig túl sok. Az eljárások egy részénél nincs szükség a bemenő, illetve a kimenő változók tárolására.

A mérések szerint az eljárások tárolóhely igénye átlagosan 4.6-5.7 tárolóhely közé esik, ezért célszerű változó méretű blokkokkal dolgozni.

A regisztertár kezelése ebben a formában('register block' technika) erősen hasonlít a veremtárak használatára, mert a processzornak figyelnie kell a tár alul-, illetve túlcsordulására. Mivel nem azonos méretűek a blokkok, a ciklikus tárolóban nem elegendő figyelni a következő elmentendő, vagy visszatöltendő ablak helyének mutatóját, hanem vizsgálni kell a rendelkezésre álló helyet is, hogy megállapítható legyen a kimentés, vagy a visszatöltés szükségessége, vagy lehetősége.

Megállapítható, hogy a tárolóhasználat hatékonyságának növelése mellett, sokkal bonyolultabb ablakmozgatási rendszerre van szükség a túlcsordulás és az alulcsordulás figyelése miatt.

4.3.CACHE-TÁRAK(GYORSÍTÓ-TÁRAK)

A tárolóhierarchia ismertetésekor már említésre került a cache-tárak fontos szerepe az adatforgalom gyorsítása és egyenletessé tétele szempontjából. Az ebben a pontban tárgyaltak elsősorban a processzor és a központi memória közötti tárolóra vonatkoznak, de elveit, jellegét tekintve, a központi tár és a háttértárolók közötti adatátvitelnél használt cache-tároló is ugyanúgy működik.

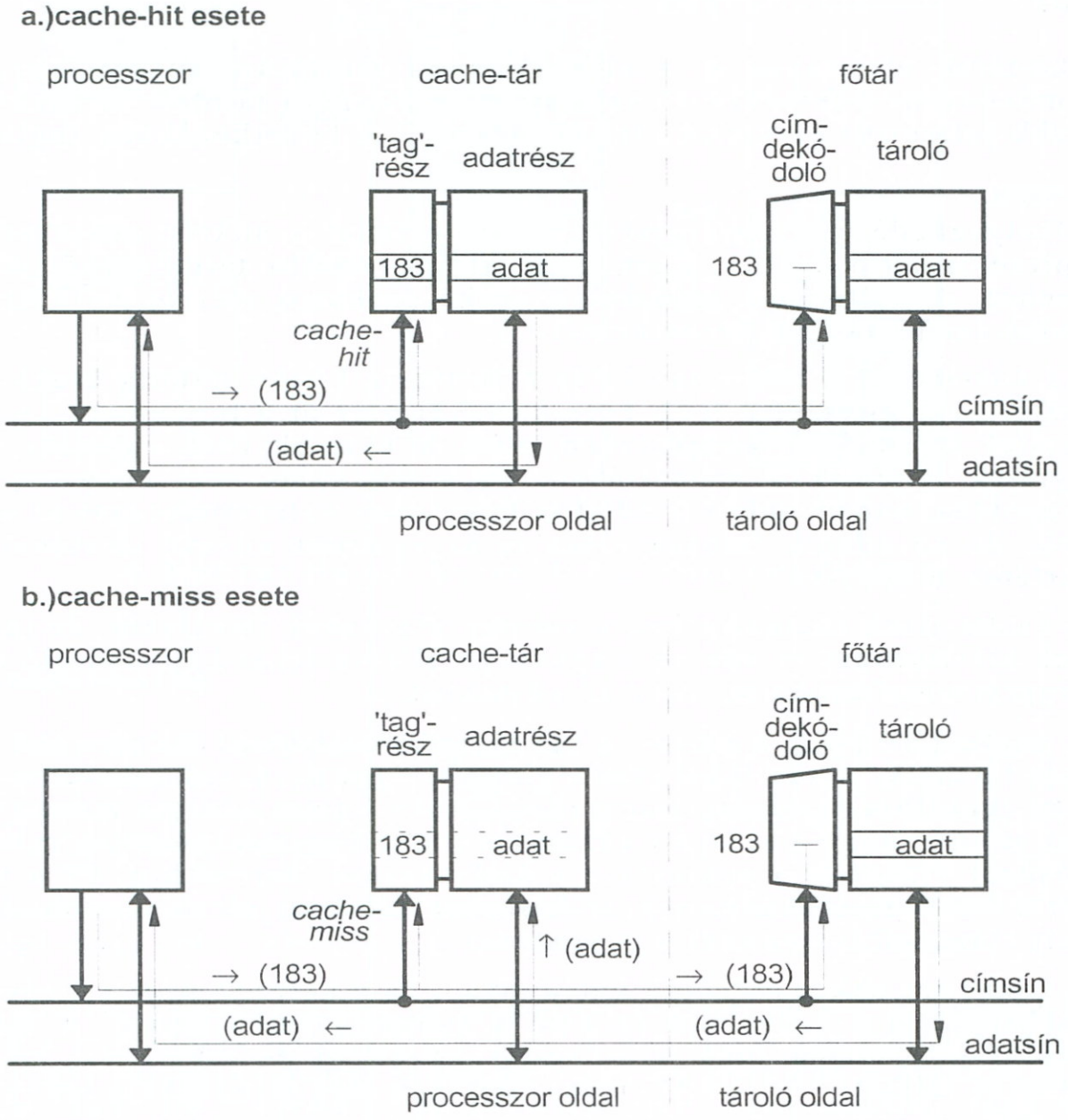
A cache-tárak utasítások és adatok átmeneti tárolására egyaránt szolgáló, gyors működésű, a felhasználó számára nem elérhető tárolók. A cache-tárak fontosabb jellemzői és kialakítási szempontjai a következőkben foglalhatók össze:

- Elhelyezkedése szempontjából, a tároló lehet a mikroprocesszorba beépítve(on-chip cache), vagy azon kívüli, önálló tárolóeszköz(off-chip cache). Az általános célú belső cache szokásos mérete 8-32 Kbyte, a külső cache 64-256 Kbyte körüli.
- Az adatátvitel a cache-tároló és a memória között mindig blokkos formájú, azaz egyszerre több byte-ot(4-32) visz át a processzor. Az adatátvitel formájából következően, ezek a byte-ok csak egymást követő byte-ok lehetnek a memóriában. Ez a megoldás nem kényszer szülte megoldás, hanem abból is adódik, hogy nagy valószinűséggel az utasítások, de az adatok felhasználása is az egymást követő tárolóhelyekről történik többnyire.
- A cache-tárak szolgálhatnak kizárólag utasítások tárolására; de tárolhatnak utasításokat és adatokat együtt is, illetve lehet külön tároló mind a programutasításoknak, mind az adatoknak is.
- A cache-tárolóban a memória egyes egymást követő rekeszeinek tartalmát tároljuk, a tárolóbeli hely címével együtt. A visszakeresés módja ú.n. **tartalom szerinti**(asszociativ, CAM=content address memory), ami azt jelenti, hogy a vizsgált adatnak a cache-ben tárolt adattal való egyezőségét vizsgálja a processzor a kiolvasáskor, kereséskor. Ez a vizsgálat a keresett adat címének az összehasonlítását jelenti a cache-ben tárolt címekkel, vagy azok egy részével.
- A cache-tár akkor működik hatékonyan, ha a keresett adat a kiolvasások többségében a cache-ben és nem a memóriában található. A találatok (cache-hit) száma függ a cache-tár méretétől és szervezési módjától. 10%-os találati hiba(cache-miss) általában még elfogadható arány.
- A cache-tároló tartalmának cseréjekor, a találati arány fenntartása érdekében, lényeges a megfelelő helyettesítési stratégia(replacement policy) kiválasztása.

- Lényeges szempont a cache-tár tartalmának és a központi tár azonos részei tartalmának egyezőségét biztosítani.
- Kívánatos, hogy a processzor és a cache-tár működési sebessége azonos legyen.

A processzor, a cache-tár és a memória közötti adatmozgatást mutatja be találat(cache-hit) és nem-találat(cache-miss) esetére a következő, 4-4.ábra.

Ha a keresett adat a cache-tárban található(cache-hit, read/write-hit), akkor azt onnét veszi ki a processzor. Ha nincs a cache-tárban a keresett adat (cache-miss, read/write-miss esete), akkor az, a memóriából kiolvasva, egyúttal a cache-tárba is beírásra kerül.



4-4.ábra: *Adatmozgatás cache-találat és -hiba esetében*

4.3.1.Cache-tárak típusai

A bevezető részben említésre került már, hogy a cache-tárak tartalom szerinti visszakeresést tesznek lehetővé. A **visszakeresés a keresett adat címe alapján történik**, ezért azt valamilyen módon a cache-tárban el kell helyezni. A cím tárolásakor, annak csak egy részét szükséges magában a cache-ben elhelyezni, még pedig csak akkora részt, amelynek alapján közvetlenül(tehát a tárolt értékből), vagy közvetve(a tárolt értékből és annak cache-beni helyéből, sorából) a blokk kezdőcíme meghatározható. A címnek azt a részét, amelyet a cache-tárban elhelyez a processzor és amelynek összehasonlításával történik a választás, **'tag'**-nek nevezik. A cache-ben 'tag'-ként tárolt címrész származhat a virtuális címből, vagy a fizikai címből, attól függően, hogy a cache-tár a processzor és a címfordító egység(MMU), vagy címfordító egység és a főtár között helyezkedik-e el.

A választást meghatározó cím mellett, a tárolt adatok állapotára vonatkozó információkat is tárol a cache-tár. Ezek a vezérlést és a helyettesítési eljárást kiszolgáló bitek, amelyek mindegyike nem található meg mindig a cache-tárban. A két legfontosabb vezérlő bit:

- **V**(valid bit), a cache-tár tartalmának(blokk, sor, byte) az érvényességét jelzi, azaz azt, hogy az adat a megadott című tárolóhelyhez tartozik és az aktuálisan érvényes adat. A cache-tár törlésekor(flushing, reset), minden V-bit 0-ra lesz beállítva és egy új adat betöltésekor lesz V=1 értékű. Minden blokkhoz legalább egy V-bit tartozik, de van olyan megoldás is, ahol minden egyes byte-hoz 1-1 V-bit tartozik.
- **D**(dirty bit), a blokk valamely részének a módosítását, felülírását jelzi. Az ilyen blokk helyére(ha D=1) nem lehet betölteni új blokkot, előbb a régit ki kell vinni a főtárba.

A jelzőbitek közül a legfontosabbak az érvényességet(V) és a módosítást(D) jelző bitek, amelyek használata a cache-tár működtetéséhez elengedhetetlen.

A cache-tárak legfontosabb jellemzőiként az alábbiak adhatók meg:

- a *cache-tár mérete*(cache-size), amely 8-256 KB között mozog, attól függően is, hogy belső(on-chip), vagy külső(off-chip) cache-tárról van szó;
- *blokk-méret*(block-size, block refill-size), amely megadja a főtár és a cache-tár között az egy egységben mozgatott adatmennyiséget; ez utasításoknál nagyobb, adatoknál kisebb érték szokott lenni, 1-8 szó(4-32 byte) nagyság között;
- *sorméret*(line size), az az adatmennyiség, amely egy-egy összehasonlítással kijelölhető és amelynek mérete a blokk méreténél kisebb, de általában avval megegyező;
- *helyettesítési algoritmus*(replacement policy), amely meghatározza a módot, ahogy a felesleges(kicserélhető) blokkot a cache-tárban kiválasztjuk egy-egy új blokk betöltésekor;
- *adataktualizálási módszer*(write strategy), az az eljárás, amellyel a módosítandó adatot a cache-tárba és a főtárba írjuk;

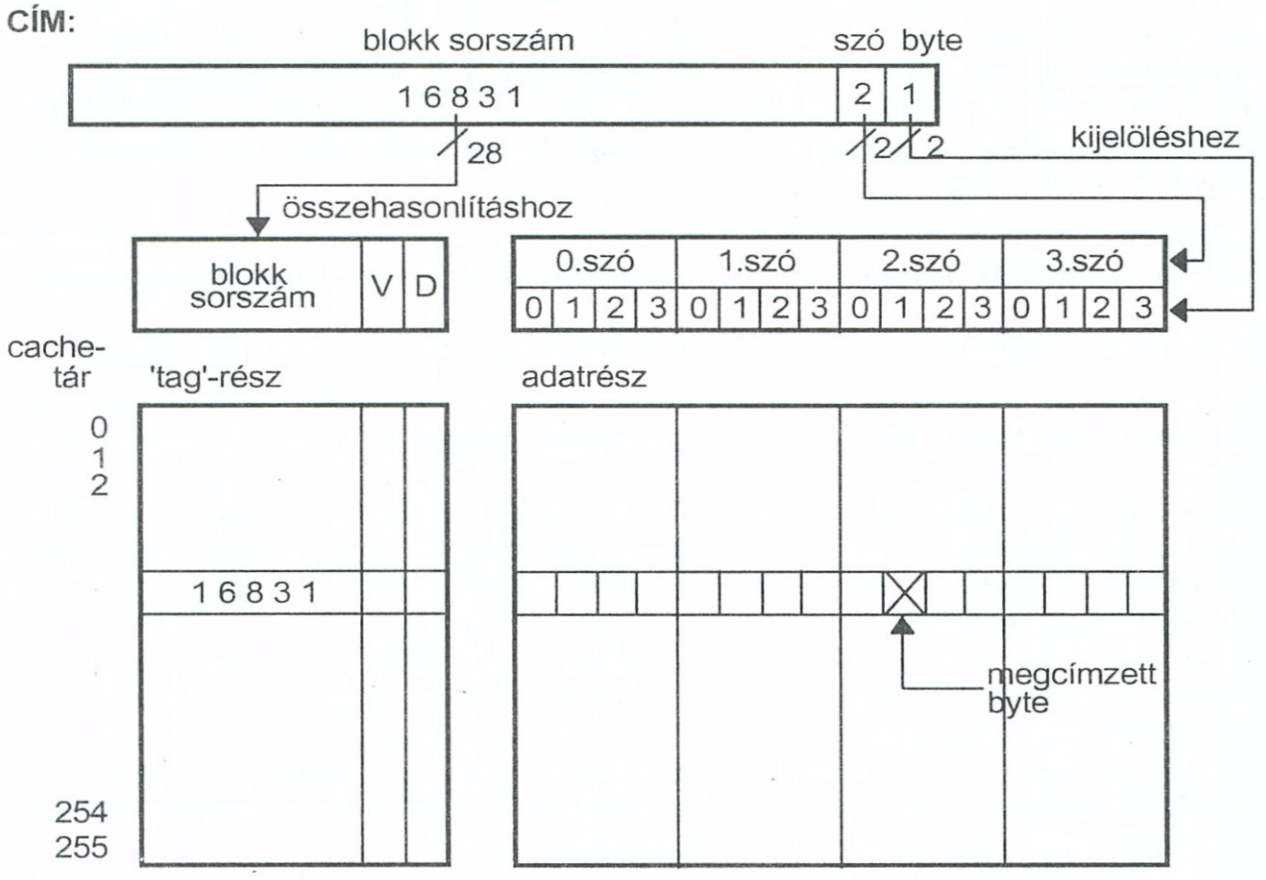
- *adategyezőség-biztosítási mód*(coherency mechanism), amely meghatározza azt a módszert, amellyel biztosítani lehet a főtár és a cache-tár(ak) tartalmának az egyezőségét.

A különböző tárolási módszerek bemutatásához használt példában avval a feltételezéssel élünk, hogy az adatátvitelnél a blokkok mérete: 4 szó=16 byte és a cache-tár 256 ilyen blokk + cím + jelzőbitek befogadására alkalmas, azaz 256 cellája, sora(line) van. A tárolóhelyek virtuális, vagy fizikai címe 32 bites. Mivel a blokk hosszúsága 16 byte, ezért csak 16-tal osztható címek jöhetnek szóba, mint kezdőcímek. A cím alsó 4 bitje ugyanis a blokkon belüli byte-ot jelöli ki. (*A konkrét esetekben ezek a méretek a megadottaktól eltérőek lehetnek!*)

A cache-tár fizikai megvalósításában külön részt képez az adattároló rész és külön részt a címet('tag'-et) tároló rész.

a.)Teljesen asszociatív(fully associative) cache

A teljesen asszociativ tárban(amelyet másképpen, a jelenlegi példa 256 sora alapján, *256-way set associativ cache*-nek is nevezhetünk) a beolvasott blokk bárhová elhelyezhető, bármelyik sorba kerülhet. Az elhelyezés sorát a helyettesítési algoritmus határozza meg.



4-5.ábra: *Teljesen asszociatív cache-tár*

A beolvasott blokk 16 byte-ja mellett, a virtuális, vagy fizikai memóriacím egy része is('tag' gyanánt) tárolásra kerül. Az ábrán bemutatott esetben ez egy 28 bites blokksorszám(4-5.ábra). A cím alsó 4 bitje a szó(4 byte) és azon belül a byte helyét határozza meg.

Amikor a processzor egy adatot keres a cache-ben, akkor a memóriabeli cím felső 28 bitjét(blokksorszám) összehasonlítja a cache-beli blokksorszámokkal (tags). Ez az összehasonlítás az összes sorban egyidőben történik, azaz a cache-hez egy olyan áramkör tartozik, amely a jelen esetben 256 párhuzamos, összehasonlító áramkört tartalmaz.

Ha a keresés sikeres(cache-hit[read-, vagy write-hit]), akkor a cím alsó 4 bitje alapján kijelöli az adott sorbeli byte-ot; ha a keresés sikertelen(cache-miss [read-, vagy write-miss]), akkor a memóriában kikeresi a kívánt byte-ot (szót) és beolvassa, vagy módosítja. A cache tartalmának átírása, vagy eredeti állapotban hagyása az alkalmazott aktualizálási eljárástól függ (4.3.2.pont).

A teljesen asszociativ cache-tár előnye a rugalmasság a betöltésnél(bármelyik blokk bárhová kerülhet), de a visszakereséshez ugyanannyi összehasonlító áramkör kell, mint ahány sora a cache-tárnak van és ez költséges. Emiatt az ilyen cache-táraknál, általában, 64 sornál többet nem alkalmaznak. Előnye továbbá az igen jó találati arány lehetősége, ugyanakkor hátrány a helyettesítési eljárás alkalmazásának a szükségessége.

A cache-tárban minden blokkhoz(esetleg minden byte-hoz) használnak egy **érvényességi jelzőbit**et(V=valid bit), amely arról ad információt, hogy a tár adott sora ténylegesen az aktuálisan érvényes adatokat tartalmazza-e, vagy sem. (Egyrészt, ugyanis a byte-ok átvitele az adatsín mérete miatt nem egyszerre történik és így lehetséges, hogy a sor egy része egy másik blokkhoz tartozó byte-okat foglal magában, másrészt a főtár azonos helyen lévő adatait felülírhatja valamilyen más memóriaművelet.)

A másik jelzőbit a **módosítási jelzőbit**(D=dirty bit), amely arról informál, hogy történt-e módosítás a blokk valamelyik byte-ja esetében.

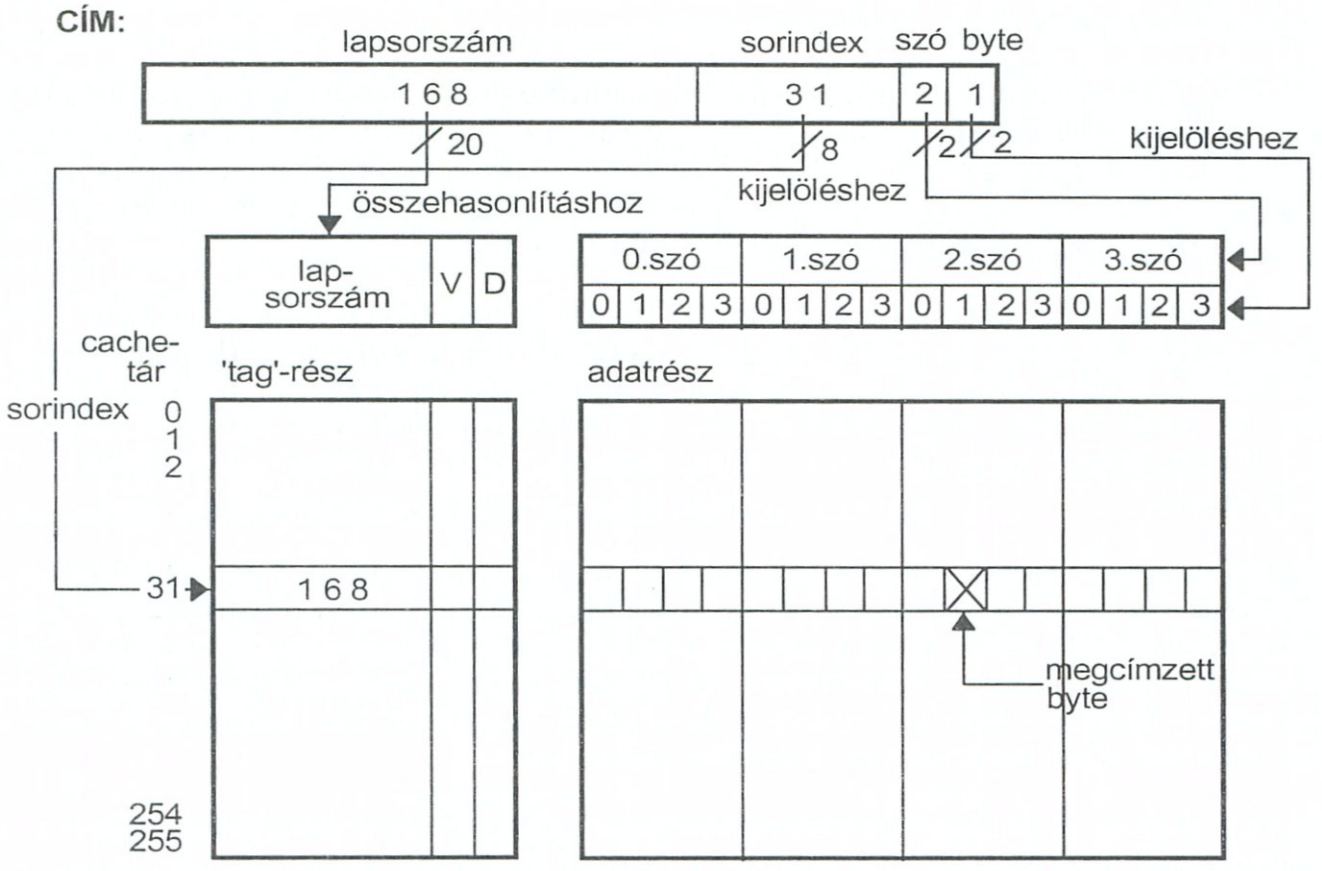
b.)Közvetlen leképzésű(direct mapping) cache

A közvetlen leképzésű(direct mapping) cache-tárnál egy-egy blokk csak meghatározott helyre kerülhet. (Emiatt *1-way set associative cache*-nek is nevezik.) A blokk helyét a cache-ben a blokksorszám(28 bit) alsó 8 bitje(mivel 2^8=256 sor van a példánkban), mint sorindex(line index) határozza meg. Tehát ebben a megoldásban a blokk csak abba a sorba kerülhet, amelyet a sorindexe meghatároz(4-6.ábra).

A korábbi 28 bites blokksorszám itt egy 20 bites lap(vagy tartomány)sorszámra és egy 8 bites sorindexre bomlik. A cache-ben a 16 adatbyte mellett a lapsorszám 20 bitje, mint 'tag', valamint a jelzőbitek kerülnek tárolásra.

Amikor a processzor egy adatot keres a cache-ben, akkor a virtuális, vagy fizikai memóriacímből előállított sorindex alapján kijelöli a keresett sort,

majd a felső 20 bitet(lapsorszámot) összehasonlítja a cache-beli lapsorszámmal('tag'). Ez így egyetlen összehasonlítást eredményez.



4-6.ábra: *Közvetlen leképzésű cache-tár*

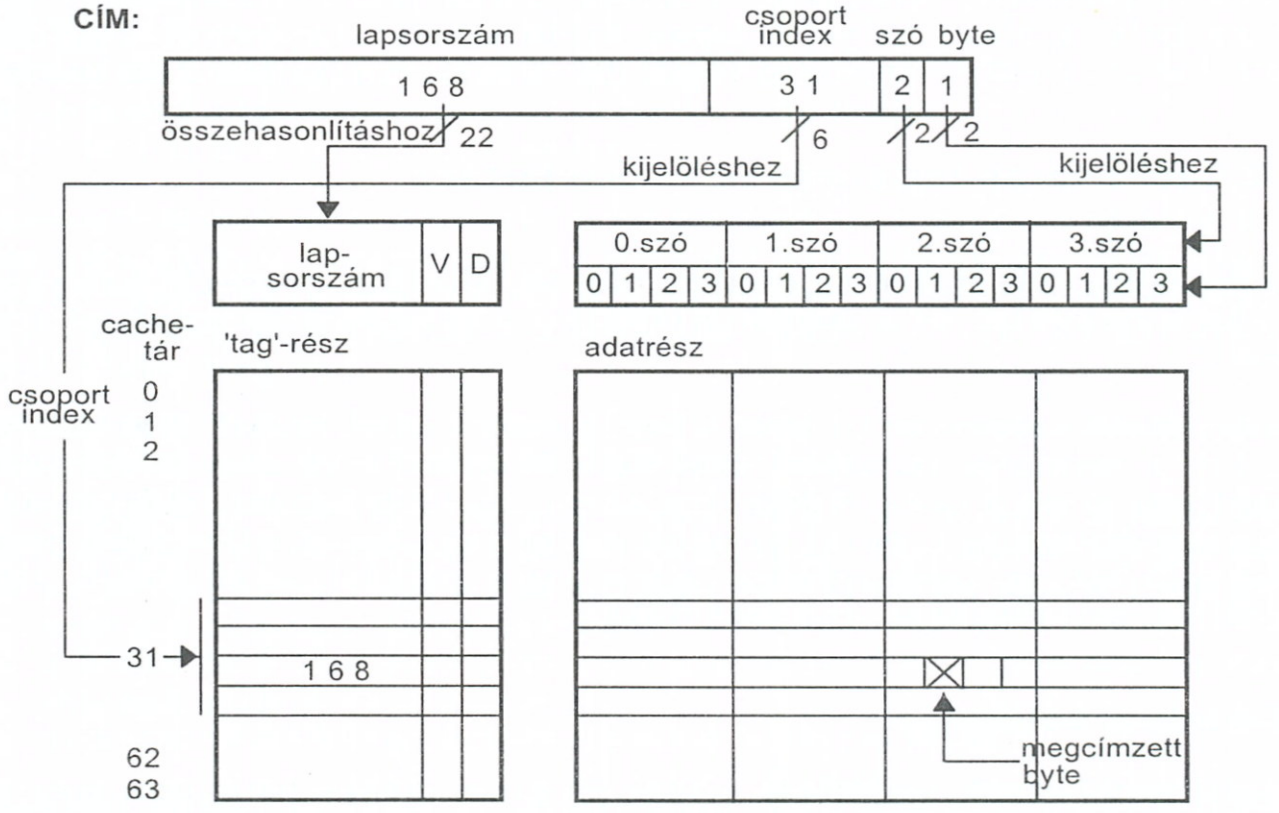
Ha a keresés sikeres(cache-hit[read-, vagy write-hit]), akkor a cím alsó 4 bitje alapján kijelöli az adott sorbeli byte-ot; ha a keresés sikertelen(cache-miss[read-, vagy write-miss]), akkor a memóriában kikeresi a kívánt byte-ot (szót) és beolvassa, vagy módosítja. A cache tartalmának átírása, vagy eredeti állapotban hagyása az alkalmazott aktualizálási eljárástól függ (4.3.2.pont).

A közvetlen leképzésű cache-tároló előnye a rövidebb 'tag', a gyors visszakeresés, az egyszerűség és olcsóság, ugyanakkor igen nagy hátránya a kötött blokkhelyek miatti merevsége, amelyből adódóan az azonos indexű helyek adatainak elérése lelassul és a találati arány alacsonyabb, mint az asszociatív tárnál. Minél nagyobb a tároló, annál jobb lesz a találati arány.

c.)Csoport asszociativ(set associative) cache

A csoport asszociatív(set associative) cache-tár(amelyet *n-way set associative cache*-nek neveznek) átmenet a teljesen asszociativ és a közvetlen leképzésű cache-tár között. A cache nagyobb, ***n*** sorból álló csoportokra van osztva, amelyek önmagukban, mint teljesen asszociativ tárolók működnek, azaz a csoporton belüli ***n*** hely bármelyikére kerülhet a blokk. Az elhelyezését a választott helyettesítési algoritmus határozza meg.

Példánkban 4-blokkos csoportméretet választva, a csoportok száma $256/4=64=2^6$ lehet. Így a beolvasott 16 adatbyte mellett, a korábbi blokksorszám 28 bitje helyett, most a felső 22 bitből képzett lapsorszámot tároljuk, valamint a jelzőbiteket. A 28 bit alsó 6 bitje alkotja a csoport kijelölésére szolgáló csoportindexet, -sorszámot(set index)(4-7.ábra).



4-7.ábra: *Csoport asszociatív cache-tár*

A csoport-asszociativ cache esetében, amikor a processzor egy adatot keres a cache-ben, akkor a memóriabeli címből képzett csoportindex alapján kijelöli az indexnek megfelelő csoportot, majd a memóriacím felső 22 bitjét(lapsorszámot) összehasonlítja a cache-beli lapsorszámokkal('tag'). Ez az összehasonlítás a csoport összes sorában egyidőben történik, azaz a cache-hez egy olyan áramkör tartozik, amely jelen esetben 4 párhuzamos, összehasonlító áramkört tartalmaz.

Ha a keresés sikeres(cache-hit[read-, vagy write-hit]), akkor a cím alsó 4 bitje alapján kijelöli az adott sorbeli byte-ot; ha a keresés sikertelen(cache-miss[read-, vagy write-miss]), akkor a memóriában kikeresi a kívánt byte-ot (szót) és beolvassa, vagy módosítja. A cache tartalmának átírása, vagy eredeti állapotban hagyása az alkalmazott aktualizálási eljárástól függ (4.3.2.pont).

A csoport asszociativ cache rugalmasabb, mint a közvetlen leképzésű tároló és ugyanakkor kis számú összehasonlító áramkör kell hozzá és viszonylag gyors.

d.)Szektor leképzésű(sector mapping) cache

A csoport asszociatív cache-tárhoz hasonló, köztes megoldást képező és ma a mikroprocesszorokban ritkábban használt megoldás a szektor leképzésü cache-tár.

Ennél a változatnál a processzor a csoport helyét jelöli ki asszociatív módon és azon belül a blokk helye, a lapon belüli elhelyezkedésének megfelelően kötött. Tehát az előző megoldás fordítottja a szektor leképzésű cache-tár.

4.3.2.Cache-tárak tartalmának karbantartása

A cache-tárak csak abban az esetben tudnak teljesítménynövelő hatást kifejteni, ha a tartalmuk és a memória tartalma megegyezik és megfelel a valóságnak.

Utasítástárolás esetében ez nem okoz gondot, mivel annál csak kiolvasás van, így a memória tartalmát nem kell felújítani. Az adatok esetében ez viszont már gondot okoz. Különösen akkor, ha több feladat feldolgozása fut a gépen(multitasking), mivel az adott memóriarészhez esetleg több program is hozzáférhet és módosíthatja. Ezen segít például a különálló tárolók használata, ami általában nem megoldható.

a.)Tartalom betöltése

A cache tartalmának betöltését a futó program igénye határozza meg többnyire.

A betöltés egyik változata(demand fetching) az, amelyik az **aktuális igény felmerülésekor**(nincs a keresett adat a cache-ben) keresi ki a memóriából a kért byte-ot tartalmazó blokkot. A blokk cache-be töltésével párhuzamosan, a processzor azonnal megkapja a keresett byte-ot(load-through, read-through). Ez a betöltési eljárás a legegyszerűbb és a leggyakrabban ezt is használják.

Egy másik eljárásnál avval a feltételezéssel élünk, hogy egy blokk használata nagy valószinűséggel maga után vonja a rákövetkező blokk használatát is. Ezért az n-dik blokk használatakor, automatikus **előkészítéssel**(prefetching) betöltődik a rákövetkező n+1-dik blokk is. Ezt a módszert kis blokkméretnél célszerű alkalmazni, a hiba esetén szükséges betöltés gyorsítása végett. Célszerű valamilyen előretekintési stratégiát(look ahead policy) alkalmazni a következő blokk kiválasztására.

A **szelektív előkészítést**(selective fetching) alkalmazó módszernél az írható adatokat a memóriában tartjuk és a cache-be csak olyan adatok(utasítások) kerülnek, amelyeknél nincs szükség az átírásra. Természetesen ez a megkötés erős korlátozó tényező a módszer alkalmazásával szemben.

b.)Aktualizálás

A cache-tár tartalmának módosítása után, a lehető legrövidebb időn belül a memóriában is aktualizálni kell a tartalmat. Ha ez nem történik meg a szükséges gyorsasággal, gyakorisággal, akkor előfordulhat az, hogy a processzor egy másik feladat(taszk) kapcsán, nem az aktuális memóriatartalmakkal dolgozik. A cache-tár és a főtár tartalma egyezőségének biztosítására használt módszerek:

- az azonnali beírás, átírás(write through) és
- a visszamásolás, visszaírás(write back, copy back),
- egyszeri átírás(write-once) módszere.

Az **azonnali átírás**(write through) alkalmazásakor, a módosított byte azonnal beírásra kerül a memóriába, függetlenül attól, hogy az adott blokk a cache-ben van-e, vagy sem. Az átírás gyorsítható 1-2 byte-os puffer használatával, amelyben sorbanállnak a memóriába írandó byte-ok. Az adatmódosításkor két lehetőség van:

- a módosított byte-hoz tartozó blokk a cache-ben van(cache-hit, write-hit); ekkor a cache tartalma is aktualizálódik;
- a módosított byte-hoz tartozó blokk nincs a cache-ben(cache-miss, write-miss); ez esetben az alábbi lehetőségek valamelyikét alkalmazzák:
  - az átírást követi egy visszaolvasás(read-miss) és ennek eredményeként a módosított blokk betöltése,
  - a blokk betöltése után aktualizálja a cache-t és a memóriát is (write through with write allocation),
  - az átírást nem követi betöltés(posted write through).

Az azonnali átírási módszer alkalmazásakor, ha cache-hiba(cache-miss) következik be, akkor általában az aktualizálást nem követi a blokk cache-tárba töltése(posted write through).

A **visszaírási, visszamásolási**(write back, copy back) eljárás alkalmazásakor a következő lehetőségek közül választ a processzor:

- az adott byte-hoz tartozó blokk a cache-ben van(write-hit); csak a cache-ben aktualizál, a főtár tartalmát csak a blokk cseréjekor módosítja,
- az adott byte-hoz tartozó blokk nincs a cache-ben(write miss);
  - csak a memóriában aktualizál, a blokk cache-be töltése elmarad;
  - a blokkot beolvasás után(fetch first) aktualizálja, visszaírás csak a blokk cseréjekor.

Cache-hiba esetén, a visszaírási módszer általában betölti a keresett blokkot a cache-be és ott aktualizál. A visszaírási módszernek hátránya, hogy a cache- és a főtár egyezősége nem mindig biztosítható, viszont az átírás elmaradása miatt működése gyorsabb és a sínterhelés csökken.

Az **egyszeri beírásos, átírásos módszer**(write-once) alkalmazásakor, a módosított byte első cache-be írásakor, a főtár tartalmát is aktualizálja a pro-

cesszor, majd a továbbiakban már csak a cache-tár tartalmát módosítja a processzor. A főtár aktualizálása csak a blokk cseréjekor történik meg.

Előnye az egyszeri átírásos módszernek, hogy a sínrendszer terhelése kisebb, mint az átírásos módszeré, ugyanakkor a főtár aktualizálása is elegendő gyakoriságú lesz, gyakoribb, mint a visszaírásos technika alkalmazásakor.

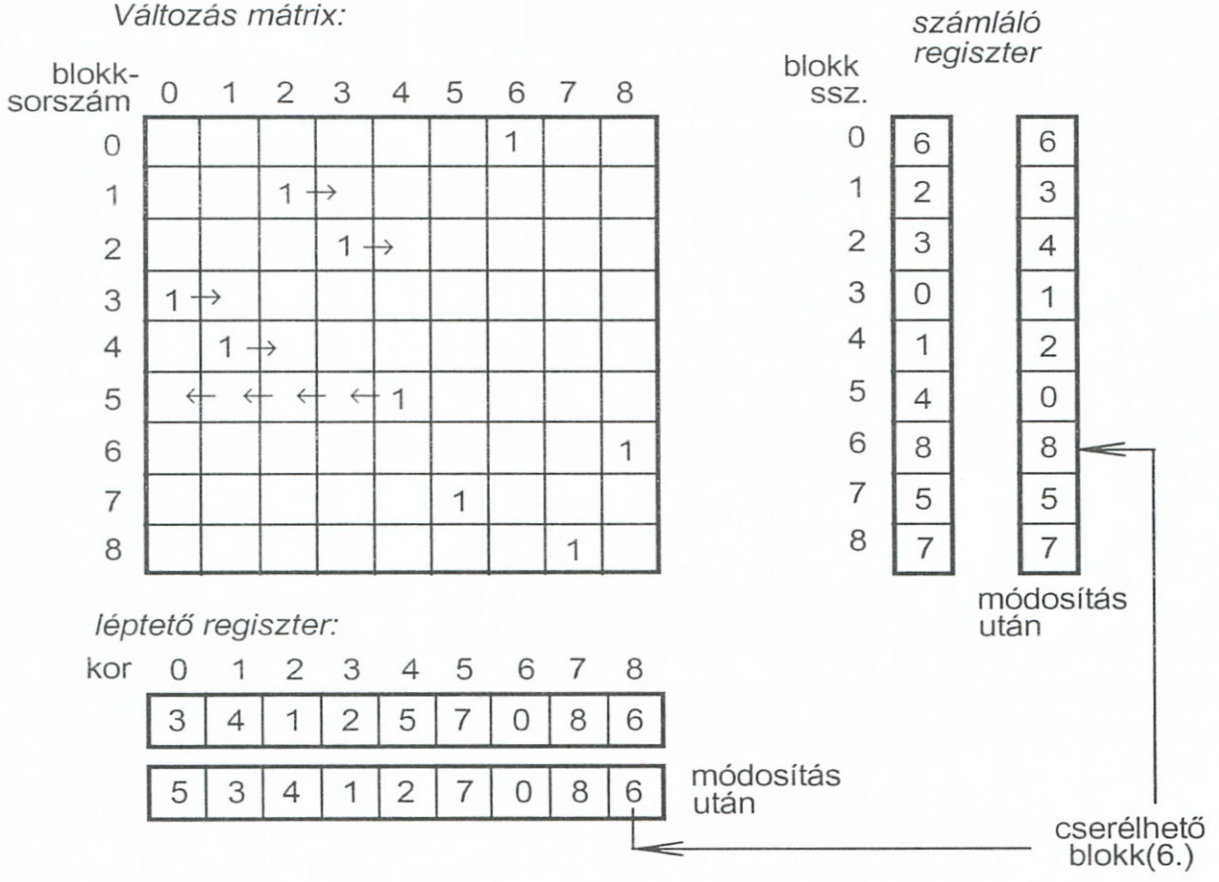
Egyes processzorok(pl.: az i486-os, a PowerPC 601-es processzorok) esetében, szoftver úton választható a kétféle(átírásos, vagy visszaírásos) aktualizálási módszer egyike.

c.)Helyettesítési eljárások

A cache-tárak hatékonyságát az biztosítja, ha a lehető legkevesebbszer kell a tartalmát cserélni. Ahhoz, hogy ez elérhető legyen, a közvetlen leképzésű cache-tár kivételével, olyan cserélési, helyettesítési stratégiát kell alkalmazni, amely erre lehetőséget nyújt.

A cache-táraknál az **LRU**(least recently used) stratégiát, a **legkevésbé mostanában használt** blokkok cseréjét alkalmazzák a leggyakrabban.

A blokkok használatának és cseréjének követése a változás mátrix alapján lehetséges. A mátrix sorai blokkonként jelzik a blokk korát, azaz, hogy milyen rég nem használták(4-8.ábra).



4-8.ábra: *LRU helyettesítési módszer*

Az ábra alapján látható, hogy a mátrixnak annyi sora és oszlopa van, ahány blokk a cache-ben elhelyezhető. A blokk sorában annyiadik helyen áll az 1-es, ahányadik a használat gyakoriságában a blokk. Ha a blokk igénybevételre kerül, akkor a sorának első helyére kerül az 1-es és az összes többi, megelőzött blokkhoz tartozó 1-est egy pozicióval hátrébb írjuk.

A fenti eljárás megvalósítására az alábbi módszerek alkalmazhatók:

- **számláló regiszter** használata, amely blokkonként a blokkok 'korát', azaz az 1-esének mátrixbeli pozicióját tárolja; és a legmagasabb 'korú' blokkot lehet cserélni;
- **léptető**(shift) **regiszter** alkalmazásával, amelyben az adott blokk sorszáma annyiadik pozición található, amennyi a 'kora'; a legutolsó helyen lévő blokk cserélhető;
- használati gyakoriság szerinti **megelőzési**(háromszög) **mátrix** alapján; a mátrix (i,j)-dik eleme = 1, ha az i-dik blokk gyakrabban használt, mint a j-dik, azaz a változás mátrixban előrébb áll, mint a másik; ha az i-dik blokkot használja a processzor, akkor az i-dik sor minden elemét 1-esre és az i-dik oszlop minden elemét pedig 0-ásra kell beállítani; azt a blokkot kell cserélni, amelynek sorában minden elem 0, és oszlopában pedig minden elem 1-es értékű.

d.)Adategyezőség biztosítása

A cache-tárak használata esetén, fontos megoldandó probléma a cache-tárak és a főtár tartalmának mindenkori egyezőségét biztosítani. Ennek szükségessége több okra vezethető vissza. *Egyrészt* olyan adatbeírási módszert használ a processzor, amely a főtár tartalmát csak időközönként aktualizálja(write-back eljárás), *másrészt* a cache-tár használata közben, a főtár tartalmát egy másik egység módosíthatja(pl. lapváltás, DMA segítségével lebonyolított I/O művelet, stb.). Az így keletkező különbözőségeket a lehető legrövidebb időn belül meg kell szüntetni.

Az adategyezőség feladatának megoldásában meghatározó:

- az alkalmazott visszaírási technika,
- a cache-tár strukturális elhelyezkedése(virtuális cím fordítása előtt, vagy után; I/O adatátvitelek cache-táron keresztül, vagy közvetlenül történnek-e).

1.Visszaírási módszerek

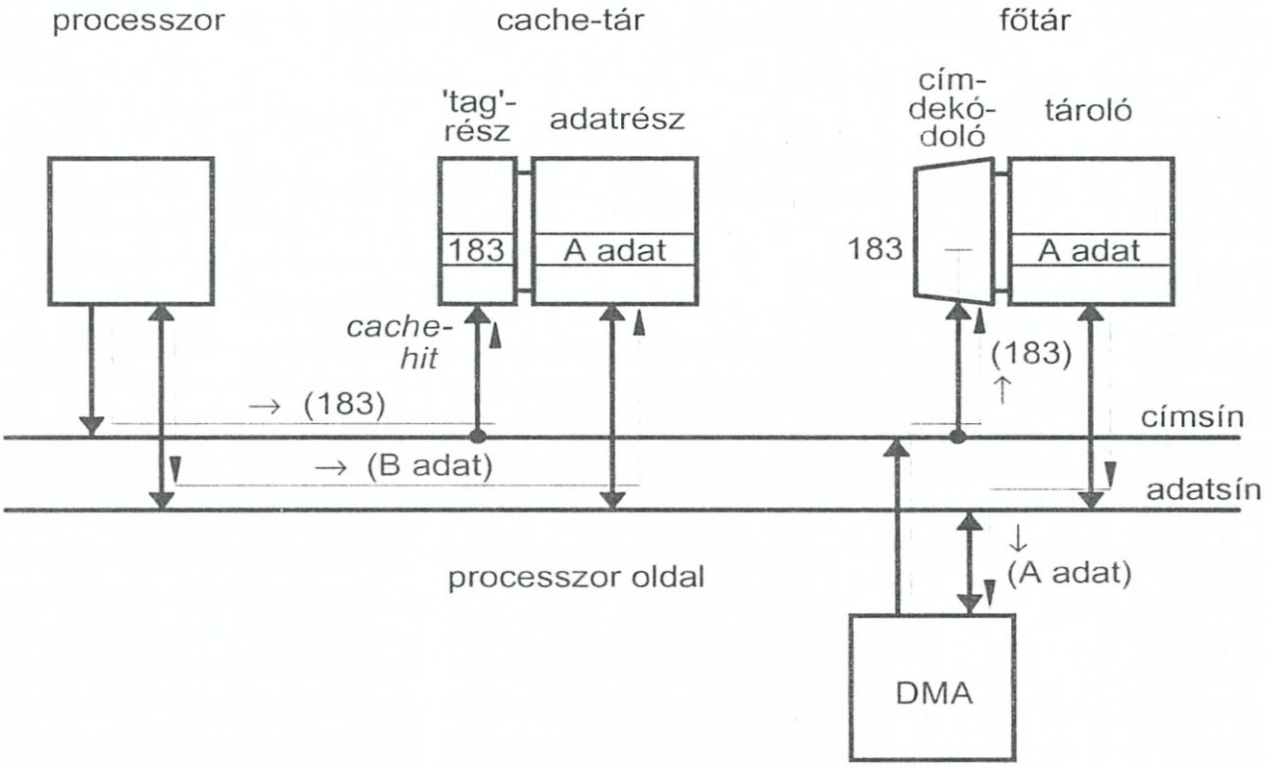
A visszaírási módszerek közül a *közvetlen átírás*(write through) alkalmazása a legproblémamentesebb, mert a cache-tár felülírásával egyidőben, a processzor a főtár tartalmát is aktualizálja.

A *pufferelt közvetlen átírás*(buffered write-through) azonban már gondot okozhat, mivel a pufferben tárolt adatok késleltetve aktualizálják a főtárat és így egy közbenső főtár-olvasási művelet hamis adatot szolgáltathat a pro-

cesszornak. A puffer kis méretével és ellenőrző logika beépítésével megszüntethető ez a probléma.

A *visszaírási módszer*(write-back) használata biztosan problémákat okoz, ha a főtárhoz más eszközök(DMA, másik processzor) is hozzáférhetnek és ez az általános. Ennek strukturális formáját mutatja be a következő, 4-9.ábra.

Látható, hogy miközben a processzor a cache-tár adataival foglalkozik, addig a DMA-n keresztül a főtár és valamelyik I/O eszköz között adatcsere történhet, amely felülírhat, vagy olvashat olyan tárolóhelyeket, amelyeknek a tartalma a cache-ben is megtalálható.



4-9.ábra: *Cache-tár elhelyezkedése a főtár és a processzor között*

2.Cache-tár strukturális elhelyezkedése

A cache-tár adatáramlási útvonalakon való elhelyezkedése két szempontból érdekes különösen.

Címfordítás szempontjából, amely a virtuális memóriakezeléshez (4.4.pont) kapcsolódik szorosan. (1) Ha a tárolókezelő egység(MMU) a processzor és a cache-tár között van, akkor a cache-tár már fizikai címeket kap meg és így a cache-tár címei és a főtár tárolóhelyei között egyértelmű megfeleltetés hozható létre. Azaz a cache-tár adott címmel rendelkező adata (elvben) ugyanaz, mint a főtároló ugyanilyen című tárolóhelyének a tartalma. Tehát címfordításból származó adategyezőségi probléma nincsen, azonban a címfordítás miatt a cache használata lelassulhat. (2) Ha azonban a memóriakezelő egység a cache-tár és a főtár között helyezkedik el, akkor a működése ugyan gyors, de mivel a cache-tár virtuális címekkel dolgozik, elképzelhető, hogy két kü-

lönböző feldolgozáshoz tartozó virtuális címhez is ugyanazt a fizikai címet rendelte hozzá az MMU. Ez nyilvánvalóan ütközést okoz és az adategyezőség nem biztos, hogy fenntartható. Ezeket a virtuális címeket *szinonímák*nak (synonyms) nevezik. Az ilyen címütközések kiszűrésére alkalmazzák az *inverz címfordító egységet*(ITB=Inverse Translation Buffers). Cache-hiba esetén, a keresett adat virtuális címét a TLB alakítja át fizikai címmé, amelyet azonnal visszafordít az ITB, annak megállapítására, hogy létezik-e másik virtuális cím is a cache-ben, amelyhez ugyanez a fizikai cím tartozik. Ha ilyen nincs, akkor a főtárból kikeresett adat bekerül a cache-be, egyébként a virtuális címet a már létező címre módosítja az ITB.

I/O átvitelek szempontjából kérdéses azok lebonyolításának megoldási módja. (1) Ha az I/O adatátvitel közvetlenül a főtárba irányul(4-9.ábra), akkor az adategyezőség nem biztosítható egyszerűen. Ilyen esetekben alkalmazzák azt a figyelő rendszert(*snoop logic*, bus watcher), amely minden címet ellenőriz, amelyhez tartozó adat a sínen átvitelre kerül. Ha az I/O művelet a főtárban átír egy memóriahelyet, a figyelő logika ellenőrzi, hogy a cache-ben van-e ugyanolyan című adat; ha igen, akkor érvényteleníti a cache-tartalmát(V-bit). Ritkábban a cache-tár tartalmát is aktualizálja az I/O művelet. Főtárból olvasáskor ugyancsak ellenőrzi azt, hogy a cache-tárban megtalálható-e ugyanazon című adat és ha igen akkor a cache-ből továbbítja azt a kimenetre. (2) Ha az I/O adatátvitel a cache-táron keresztül bonyolódik le(I/O through cache), akkor ugyan adategyezőségi probléma nem lép fel, viszont a cache használatának gyakorisága jelentősen megnövekszik. Ezért, a cache-tárhoz többszörös hozzáférési lehetőséget kell teremteni, a használat iránti igényeket rangsorolni kell, amely feladatok további vezérlő logikát kívánnak.

3.MESI protokoll

Multiprocesszoros esetekben az adategyezőség biztosítása még nehezebbé válik, különösen, ha több cache-tár is van a rendszerben. Ezekben a helyzetekben különösen fontos az egységes megoldási módok alkalmazása. Ezt szolgálja az ú.n. MESI protokoll. A MESI rövidítés a **'modified, exclusive, shared, invalid'** szavak kezdőbetűiből lett kialakítva, amely szavak a cache-tár blokkjainak aktuális állapotára vonatkoznak.

A cache-tár blokkjainak lehetséges állapotai az alábbiak:

- módosított(**modified**), amely esetben a cache-tár blokkja a főtárhoz viszonyítva módosítva lett és ez a blokk az egyetlen érvényes adatblokk;
- kizárólagos(**exclusive**), amely esetben a cache-tár blokkja a főtárral egyező, érvényes adatokat tartalmaz és más cache-tárban nem fordul elő az adott adatblokk;
- megosztott(**shared**), amely esetben a cache-tár blokkja a főtárral egyező, érvényes adatokat tartalmaz és legalább még egy cache-tárban megtalálható ugyanaz az adatblokk;
- érvénytelen(**invalid**), amely esetben a blokk érvénytelen adatokat tartalmaz.

4.4.VIRTUÁLIS TÁRKEZELÉS(CÍMZÉSI ELJÁRÁSOK II.)

A központi egység elemeinek tárgyalásakor a tárolóhelyek kikeresését biztosító címzési eljárások, módszerek alapjairól már szó esett. A 3.2.2.b.pontban tárgyalt különböző címképzési eljárásoknak a kiterjesztése a most bemutatandó virtuális címzés, amely nem csak a központi tár, hanem a háttértárolók területét is elérhetővé teszi, ugyanazon módszerek alkalmazásával.

4.4.1.A tárolóhasználat problémái

A programvégrehajtás alapproblémája, hogy a mindenkori, végrehajtás alatt álló programrésznek és a feldolgozandó adatoknak legalább a központi memóriában kell lenniük. Ugyanakkor a főtár mérete nem teszi lehetővé, hogy a teljes program és az összes adat egyszerre a tárban legyen. Habár a tárolók mérete egyre növekszik, de a feldolgozó programok mérete és a feldolgozandó adatok mennyisége is folyamatosan növekszik.

A központi tár véges kapacitása miatt, az aktuálisan nem használt programokat, adatokat valamilyen háttértárolón kell tárolni és szükség esetén betölteni a központi memóriába. A háttértároló használatának azonban nem kizárólag csak a kapacitáshiány az oka; közrejátszik ebben gazdaságossági és cserélhetőségi ok is. Nagy mennyiségű adatot célszerű alacsony fajlagos költségű([Ft/bit]), egyszerűen kezelhető és esetleg hordozható tárolón elhelyezni. Ma még a mágneses tárolóeszközök, pl.a mágnesszalag is, emiatt használatosak másodlagos tárolókként.

A leggyakrabban használt háttértárolókat(mágneslemez, mágnesszalag, optikai lemez) a 2.1.2.c.pontban már bemutattuk. A virtuális tárolás szempontjából, ezek közül a mágneslemez(részlegesen az optikai lemez), mint közvetlen elérésű tároló a lényeges; ezért a továbbiakban háttértároló esetében mindig a mágneslemez tárolókra(de nem a hajlékonylemezekre - floppy-kra) gondolunk.

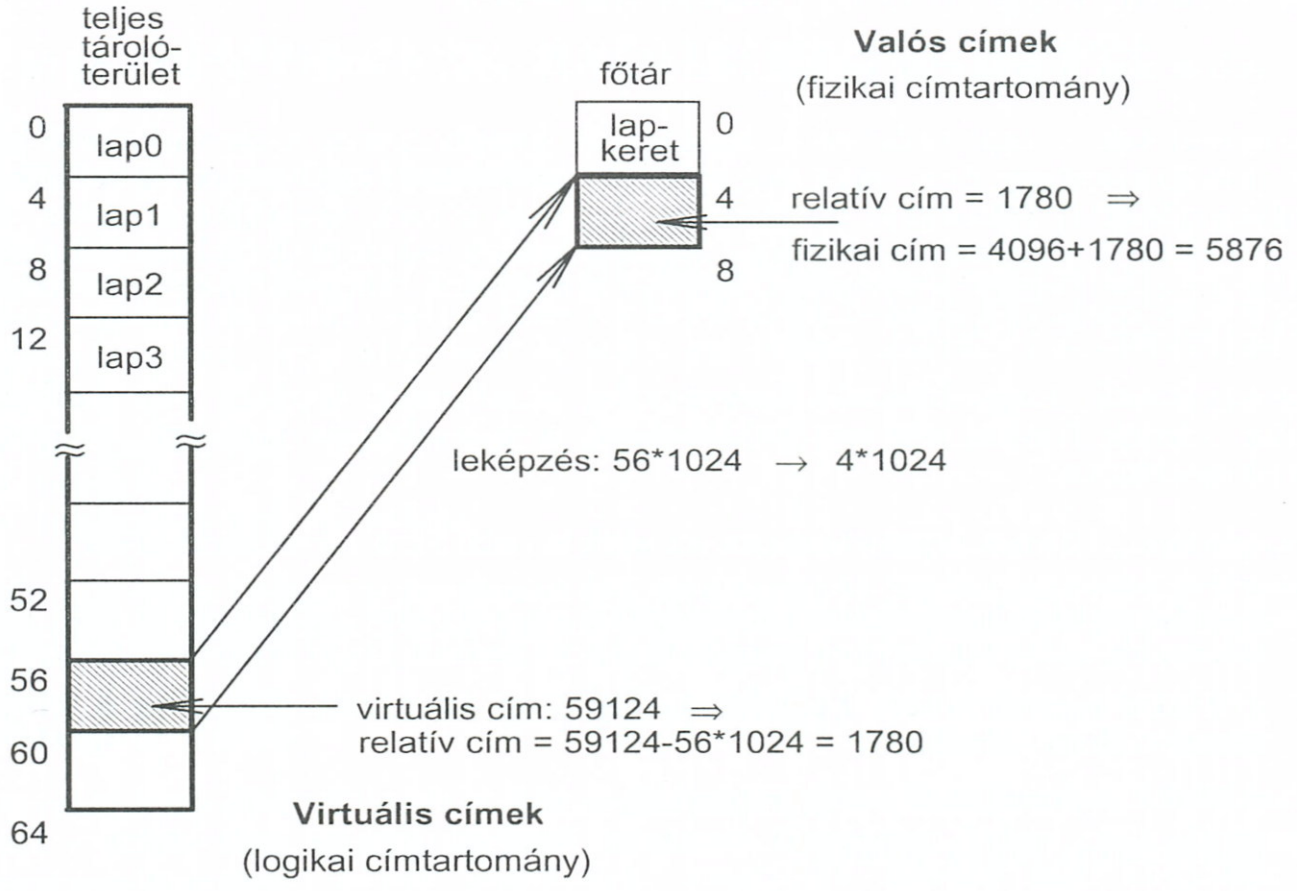
A háttértárolón lévő programrészek központi tárba való betöltésére a programozók kedvelt módszere az ú.n. **overlay technika**, amelynél az újonnan betöltött programrész egy másik, már nem használt rész helyére kerül. Ennél a megoldásnál, tehát az overlay technikánál a problémát az jelenti, hogy a programozónak kell gondoskodnia az adott rész megfelelő időpontbeli betöltéséről és elhelyezéséről. Habár ez utóbbi feladatot a fordító-szerkesztő programok már megoldják, az átlapolásos program futtatása igen körülményes. A háttértárolóról való betöltés automatizálására dolgozták ki az 'automatikus overlay' technikát, a **lapozásos tárolókezelési módszer**t. A lapozási módszer a nagygépeken régóta alkalmazott eljárás, amelynek kezelő rendszere az operációs rendszernek részét képezte, majd később hardver megoldásokkal ez majdnem teljesen átkerült a processzor hatáskörébe. A lapozási eljárás ma már a mikroprocesszorok esetében is általánosan használt módszer és vezérlése teljes körű hardver támogatással rendelkezik(pl.: i386/486-os, MC68030/68040-es processzoroknál), a processzorba beépített módon.

Az automatikus lapozás alkalmazásával lehetővé vált az is, hogy a teljes rendelkezésre álló tárolóterületet, vagy legalábbis annak nagy részét lefedő területet, látszólag közvetlenül címezhessük, azaz egy egységnek, látszólagos központi tárnak tekintsük. Ezt a címzési lehetőséget, módszert nevezik **virtuális címzésnek**, tárkezelésnek. Az így rendelkezésre álló címtartományt, a lehetséges, használható címek összességét, **virtuális címtartománynak** nevezik.

A virtuális címtartomány bármely részében(fizikailag nagy valószinűséggel valamelyik másodlagos tárolón) elhelyezkedő blokkot a feldolgozáshoz a központi tár területére kell átvinni. Az átvitt blokk mérete lehet változó, ekkor **szegmens**nek és lehet rögzített, ekkor **lap**nak nevezzük.

A végrehajtáshoz a processzornak ismernie kell az eredeti virtuális cím valódi helyét a központi tárban, a **valós címtartomány**ban. A 4-10.ábra segítségével felírható a címfordítás, **címleképzés**(memory mapping) alapösszefüggése:

valós cím = átvitt blokk fizikai kezdőcíme + relatív cím



4-10.ábra: *Virtuális cím átszámítása valós címmé*

A felhasználó, a programozó számára a *virtuális cím*, mint **logikai cím** (amellyel programjában dolgozik, mintha létező, közvetlenül címezhető táro-

lóhely lenne); a *valós cím*, mint tényleges, a központi tár **fizikai címe** jelenik meg.

A címkialakítás lényegére mutat példát a 4-10.ábra. Az ábrán a központi tár 8Kbyte kapacitású, míg a kialakítható virtuális címtartomány 64Kbyte nagyságú. Az átvihető tárolóterület nagysága, a lapméret 4Kbyte.

Ha a program futásakor az 59124-es sorszámú rekesz tartalmára van szükség, akkor a tárkezelő rendszer megállapítja annak pontos háttértárolóbeli helyét, kiolvassa az 56K-s(56*1024=57344) címtől kezdődő lapot és átviszi a központi memóriába. A központi tárban megvizsgálja azt, hogy a két **lapkeret (frame)**, azaz az 1-1 lapnyi adat befogadására alkalmas tárolóterület közül melyik szabad, vagy melyik szabadítható fel. Példánkban a 2.lapkeret helyére tölti be a lapot a memóriakezelő rendszer. Ennek a lapkeretnek a kezdőcíme: 4096.

Az említett tárolóhely(59124-es) relatív címe a logikai lapkezdettől számítva:

$$\text{relatív cím} = 59124 - 56*1024 = 1780$$

evvel a **valós,** központi tárbeli **fizikai cím** értéke:

$$\text{valós cím} = \text{lapkeret kezdőcíme} + \text{relatív cím} = 4096 + 1780 = 5876$$

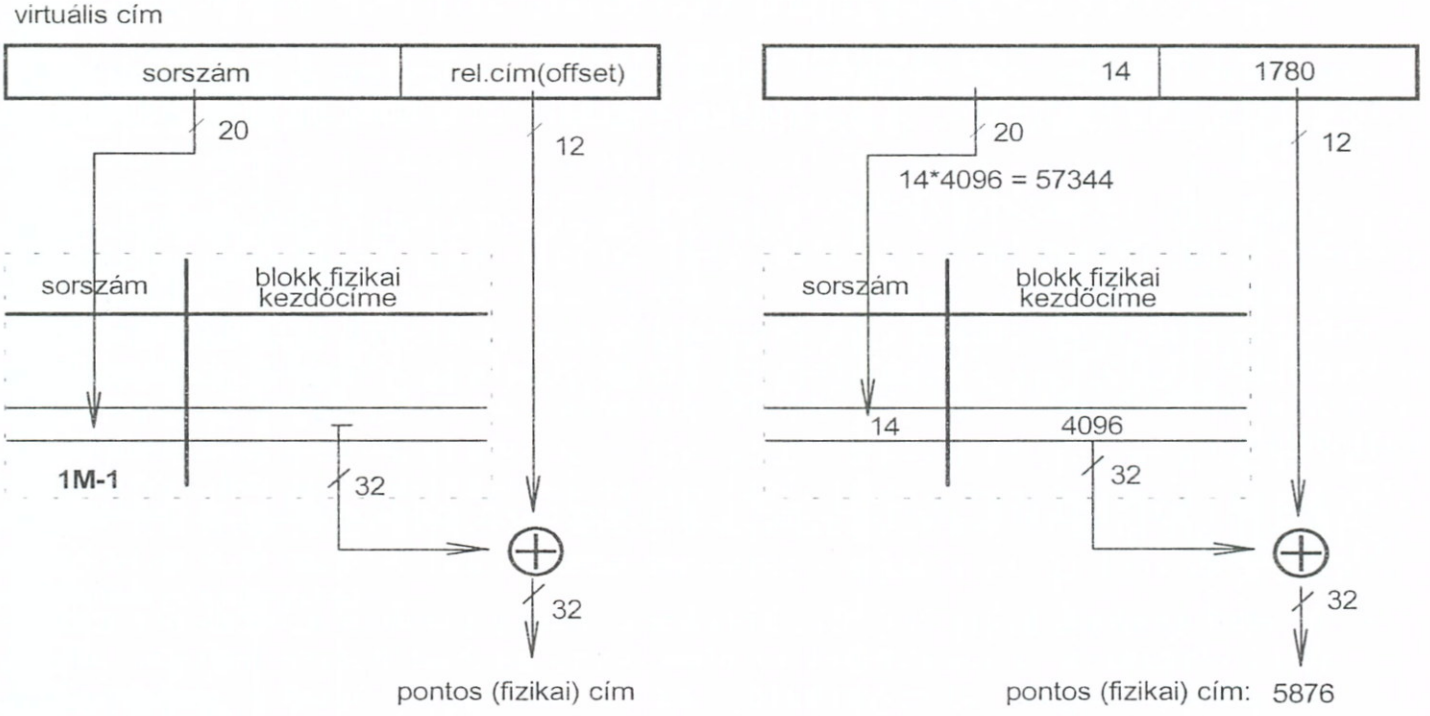
A szegmensek, a lapkezdetek, a lapkeretek kezdőcímeit, a tárolókezelő rendszer táblázatokban őrzi más kiegészítő adatokkal együtt. A **tárolókezelő rendszer**(MMU) **feladata:** a virtuális, logikai címek átalakítása valós, fizikai címekké a szegmens- és/vagy laptáblázatok adatainak felhasználásával. Ebben az értelemben a virtuális címzés táblázatok alapján megvalósított indirekt címzés (4-11.ábra).

Megjegyzendő, hogy hasonló konverziós, címfordító lépésre van szükség a logikai címek fizikai címmé alakításánál, a másodlagos tárak(pl. mágneslemez) kapcsán is.

A szegmens-, illetve laptáblázatok az alábbi adatokat tartalmazhatják:

- a logikai blokk sorszáma, kezdőcíme; a sorszám többnyire a cím legértékesebb helyiértékeinek meghatározott számú jegyéből álló számérték;
- a blokk memóriabeli kezdőcíme(lapkeret sorszám, cím), ha a blokk a memóriában található;
- a blokk mérete(csak szegmensnél);
- hozzáférési jogok jelzői;
- állapotjelzők;
- a blokk másodlagos tárbeli fizikai címe(pl. sáv, szektor).

A táblázatok egy-egy szegmenshez, vagy laphoz tartozó adatsorát szegmens-, vagy lap**deszkriptornak**(descriptor) nevezzük.



4-11.ábra: *Fizikai cím kiszámítása táblázat alapján*

4.4.2.Szegmens, lap fogalma

Az előző fejezetrészben a virtuális tárkezelés alapfogalmaival megismerkedve, említésre került már a tárolt adatok kezelésének két alapvető formája. A gyakorlatban megvalósított formák az alábbi esetek valamelyikébe tartoznak:

- szegmentálás,
- lapozás,
- lapozásos szegmentálás.

A két alapvető forma, a szegmentálás és a lapozás leglényegesebb jellemzőit, a következő részekben külön-külön valamivel részletesebben összefoglaljuk.

a.)Szegmentálás

A virtuális tárolóterületről a központi memóriába az adatokat blokkos(több tíztől - több ezer adatig terjedő adatsor) formában lehet csak átvinni. A **szegmens**(segment) olyan adatblokk, amelynek mérete nem rögzített, választható. A kialakított szegmensek átlapolódóan is megadhatók; ez egyben azt jelenti, hogy két szegmensen belül is megcímezhető ugyanaz az adat.

A 4-10.ábra alapján a cím kiszámításának módja:

logikai cím = szegmens logikai sorszáma + relatív cím

fizikai cím = szegmens fizikai kezdőcíme(báziscím)+ relatív cím

A szegmens fizikai(azaz központi táron belüli) kezdőcíme a szegmenstáblázatból lesz meghatározva, a szegmens logikai sorszáma alapján keresve a táblázatban.

A szegmentált tárkezelésnél, az átlapolódó és nem egyforma hosszú szegmensek miatt, idővel igen sok üres, nem használt hely keletkezik a központi memóriában. Egy új szegmens elhelyezése ezért gondot okozhat a memóriakezelő rendszernek. Az elaprózódott üres helyek felszámolására és összefüggő foglalt és szabad terület kialakításához, időnként szükséges a memória átrendezése. Ezt a lépést nevezik 'szemétgyüjtés'-nek(**garbage collection**).

A szegmens betöltésére, elhelyezésére a következő módszereket, stratégiákat alkalmazzák:

- az **első szabad hely**(first fit) keresése, amely esetben a memória kezdetétől vizsgálva, a szegmenst az első olyan szabad helyre helyezi, ahová méreténél fogva elhelyezhető; ez a stratégia kisebb szegmensek esetében egy viszonylag jól összefüggő, foglalt tárolóterületet hoz létre;
- a **következő szabad hely**(next fit) keresése esetében, az üres hely felkutatása nem a tároló elejétől kezdődik mindig, hanem az előző szegmens elhelyezésétől kezdődően; ez az eljárás a teljes memória egyenletes kihasználását célozza meg;
- a **legjobb hely**(best fit) kiválasztása; a legjobb hely ebben az esetben az az üres, nem foglalt tárolóterület, amelyre a szegmenst betöltve, a lehető legkevesebb szabad hely marad szabadon; ennél a módszernél, nagyobb, összefüggő foglalt területek alakulnak ki;
- a **legrosszabb hely**(worst fit) kiválasztása az előzőnek a fordítottja, itt az a cél, hogy a lehető legtöbb szabad hely maradjon a betöltött szegmens mellett; ennél a stratégiánál a nagyobb, összefüggő szabad területek kialakítása a cél.

A szegmentált tárkezelés előnye az átlapolhatóság, azaz az osztott felhasználás lehetősége, valamint a tárolt adatok jellege szerinti elkülönült adatkezelés lehetősége(mint pl. programszegmens, adatszegmens kialakítása az i286, i386/486-os processzoroknál). Hátránya az igen nagy méretű szegmensek kialakulási lehetősége, amelynek a cseréje már nehézzé válhat.

b.)Lapozás

A **lap**ok(page) olyan adatblokkok, amelyek mérete azonos és rögzített. A lapok mérete 512 byte és 8 Kbyte között mozog. A lapok nem átlapolhatók és meghatározott, a lappal megegyező méretű helyekre kerülhetnek a memóriában. Ezeket a helyeket **lapkeret**eknek(frame) nevezzük.

A 4-10.ábra alapján a cím kiszámításának módja:

logikai cím = lap logikai sorszáma + relatív cím

fizikai cím = lapkeret fizikai kezdőcíme(báziscím)+ relatív cím

A fizikai cím kialakítása, a címfordítás történhet a lapkeret sorszámának a megadása alapján is, ekkor nem összeadással áll elő a fizikai cím, hanem az egyes részek egymás mellé helyezésével (konkatenálásával). A mikroprocesszorok többsége(így az i386/486-os processzor is) ezt a módszert használja, helytakarékosság miatt.

A lap fizikai(azaz központi táron belüli) kezdőcíme a laptáblázatból lesz meghatározva, a lap logikai sorszáma alapján keresve a táblázatban.

Lapváltásnál nem okoz gondot az elaprózódó tárolóterület, mert a lapok csak a lapkeretek helyére kerülhetnek és így legfeljebb a nem használt lapkeretek üresek. Ha nincs üres lapkeret, akkor valamilyen stratégia alapján, amely eljárás hasonló a cache-táraknál már megismert eljárással, ki kell választani a kiürítendő lapkeretet.

Lapbetöltés

Új lap betöltésére, mindig a **felmerülő igény**(demand fetching) esetében kerül sor, mert nehéz megitélni az esetleg következőnek igényelt lap sorszámát. A lap mérete is nagyobb annál, mint amit éredemes esetleg feleslegesen is betölteni.

Az igény felmerülése(azaz olyan adatra van szükség, ami nem található a memóriában lévő lapokon) a lapváltási eljárást indítja el(page fault trap), amely a kívánt adatot tartalmazó lapot betölti a memóriába.

A lapváltási igény felmerülésekor egy utasítás végrehajtásának folyamatát kell felfüggeszteni és később visszatérni rá. A felfüggesztett utasítás végrehajtására két módszer közül lehet választani:

- az **utasítás folytatása** a felfüggesztési fázistól(ez az operandus előkészítési fázis többnyire); ez esetben a felfüggesztési időpontra vonatkozó állapotjellemzőket el kell menteni, majd a folytatáskor visszatölteni; mivel itt elemi lépésekről van szó, a mikroprogramvezérelt processzoroknál egy mikroprogram végrehajtásának felfüggesztéséről, ez meglehetősen bonyolult megoldást igényel;
- az **utasítás újrakezdése** a lapváltás után; ez egyszerűbb megoldás, mert a felfüggesztett utasítás mikroprogramját előről lehet indítani, kevesebb állapotjelzőt kell tárolni, ugyanakkor a felfüggesztés pillanatában vissza kell állítani az utasítás megkezdése előtti állapotot; többnyire az utasítás újrakezdését alkalmazzák megoldásként.

Helyettesítési eljárások

A felszabadítandó lapkeret kiválasztása történhet:

- a lapok használatára alapozva(usage-based), amelynél a gyakran használt lapokat tartjuk a memóriában;

- a lapok nem-használatára alapozva(not-usage-based), amelynél a legkevésbé használt lapokat kísérelјük meg kicserélni.

A cserélendő lapok körének kiválasztásánál is kétféle módszer között választhatunk:

- a cserélhető lapok körébe az összes, memóriában lévő lapot bevonjuk (global paging),
- a cserélhető lapok körét csak az adott feladathoz tartozó lapok alkotják (local paging).

A laphasználat gyakoriságának vizsgálatakor fontos szerepet tölt be az **aktuálisan használt lapok köre**, az ú.n. '**working set**'. A working set-be tartozó lapok meghatározásakor azt vizsgáljuk, hogy egy adott időtartamon belül, mely lapok lettek leggyakrabban használva. Ezek száma egy-egy feladatnál egy konstans értékhez tart és ha ezeket a memóriában tartja a tárolókezelő rendszer, akkor igen ritkán kell új lapot behozni. Természetesen ennek feltétele, hogy *a memória használható területe nagyobb legyen a 'working set' által igényelt tárolóterületnél.*

A lapváltásokhoz használt leggyakoribb **helyettesítési algoritmusok** az alábbiak:

- *véletlenszerű választás*, amelynél a cserélendő lapot a lehetséges lapok közül véletlenszerűen választja ki a tárolókezelő rendszer; ez a módszer, amely nem veszi figyelembe az adatok processzorközeliségét (locality), nem igazán működik jól és csak a gyakori lapváltást igénylő esetekben ajánlható;
- *legrégebben bent lévő lap cseréje*(FIFO=first-in-first-out), amelynél függetlenül az igénybevételtől, a legrégebben betöltött lapot cseréli az eljárás; emiatt nagyon könnyen előfordulhat, hogy az éppen használt lapot viszi ki a rendszer, amit esetleg a következő lépésben vissza is kell hoznia; a lapok használatának gyakoriságát az eljárás nem veszi figyelembe és sok esetben rosszabbul működik, mint a véletlenszerű lapváltási módszer;
- *legrégebbi nem használt lap cseréje*(clock replacement; FINUFO= first-in-not-used-first-out); ennél a módszernél a legrégebben bent lévő lapokat veszi sorra a rendszer, de csak abban az esetben cseréli ki a vizsgált lapot, ha azt nem használta a program; az eljárás során, a használatot jelző bitet vizsgálja és nullázza, ha az nem nulla és az első 0 értékű jelzőbittel rendelkező lapot cseréli ki; viszonylag egyszerű, de elegendően hatékony módszer;
- *adott idő alatt nem használt lap cseréje*(working set replacement); a módszer használatakor azokat a lapokat cseréli a rendszer, amelyek egy adott időszak alatt nem lettek használva; a 'working set' kialakítása kötődhet időtartamhoz, vagy a használat gyakoriságához(page-fault frequency replacement); bonyolultabb, jó hatékonyságú eljárás;
- *legkevésbé használt lapok cseréje*(LRU=least recently used) módszer a leggyakrabban használt eljárás, amelynek lényegét a cache-tárak esetében(4.3.2.c.pontban) már ismertettük.

A lapozásos memóriakezelésnek előnye, hogy mindig csak a szükséges lapok vannak a memóriában; hátrányaként említhető, hogy a szegmensekkel ellentétben nem lehet az azonos jellegű, tartalmú adatterületeket együtt kezelni, összefogni.

c.)Tárkezelés gyorsítása

A szegmens-, illetve lapváltások lelassítják a program futását, ezért célszerű megfelelő előkészítéssel ezek számát a lehető legalacsonyabbra szorítani. (Az igen gyakran lapváltást kikényszerítő programokat **'trashing'** programoknak nevezik.)

A szegmens- és lapváltások száma csökkenthető megfelelő, strukturált, objektum-orientált programozástechnikával, amelynek eredményeként jól elhatárolható programegységek alakulnak ki és az ugrások száma csökkenthető. Csökkenti a szegmens- és lapváltások számát, ha a lokális változók számát növeljük, mert a rájuk történő hivatkozások nagy valószínűséggel lapon belül tarthatók.

Másik lehetőség a működés gyorsítására, a szegmens- és laptáblázatok egy részének cache-tárban való elhelyezése. Ilyen lapváltást segítő cache-tár a TLB (Translation Look-aside Buffers), amely a leggyakrabban használt lapok adatait tárolja.

4.4.3.Virtuális címek leképzése

Mint, ahogy arról már korábban szó volt, a lapkezdetek, a lapkeretek kezdőcímeit a tárolókezelő rendszer táblázatokban őrzi, más kiegészítő adatokkal együtt. A **tárolókezelő rendszer**(MMU) **feladata:** a táblázatok tartalmának kialakítása, karbantartása, illetve a virtuális, logikai címek átalakítása valós, fizikai címekké a szegmens- és/vagy laptáblázatok adatainak felhasználásával. Ebben az értelemben a virtuális címzés táblázatok alapján megvalósított, egy-, vagy többlépcsős indirekt címzés.

A tárolókezelő rendszer munkáját segíti a leggyakrabban használt lapok adatait(deszkriptorait) tartalmazó cache-tár, a **lapcímfordító cache-tár** (TLB=Translation Lookaside Buffer). Ha lapváltáskor nincs találati hiba (cache-miss), azaz a lapdeszkriptor a TLB-ben van, akkor a virtuális lapcímet helyettesíti a fizikai lapcímmel és átadja a védelmi jelzőket a processzornak. Ha az adott lapdeszkriptor nincs a cache-ben, akkor a tárkezelő rendszer a laptáblázatok segítségével állítja elő(table-walk) a fizikai címet és a szükséges adatokat betölti a TLB-be, felszabadítva annak valamelyik sorát.

A TLB-cache teljesen asszociatív, vagy 2-4-utas csoport asszociatív cachetárként működik. Mérete 64-1024 sor között mozog és a találati arány 99% felett van. A visszakeresésnél használt címrész('tag') nem mindig a cím legértékesebb bitjeit tartalmazza, mivel az nem biztosítana egyenletes kihasználást a cache-ben. Ezért, a sorok szétszórása céljából, a címből ú.n.

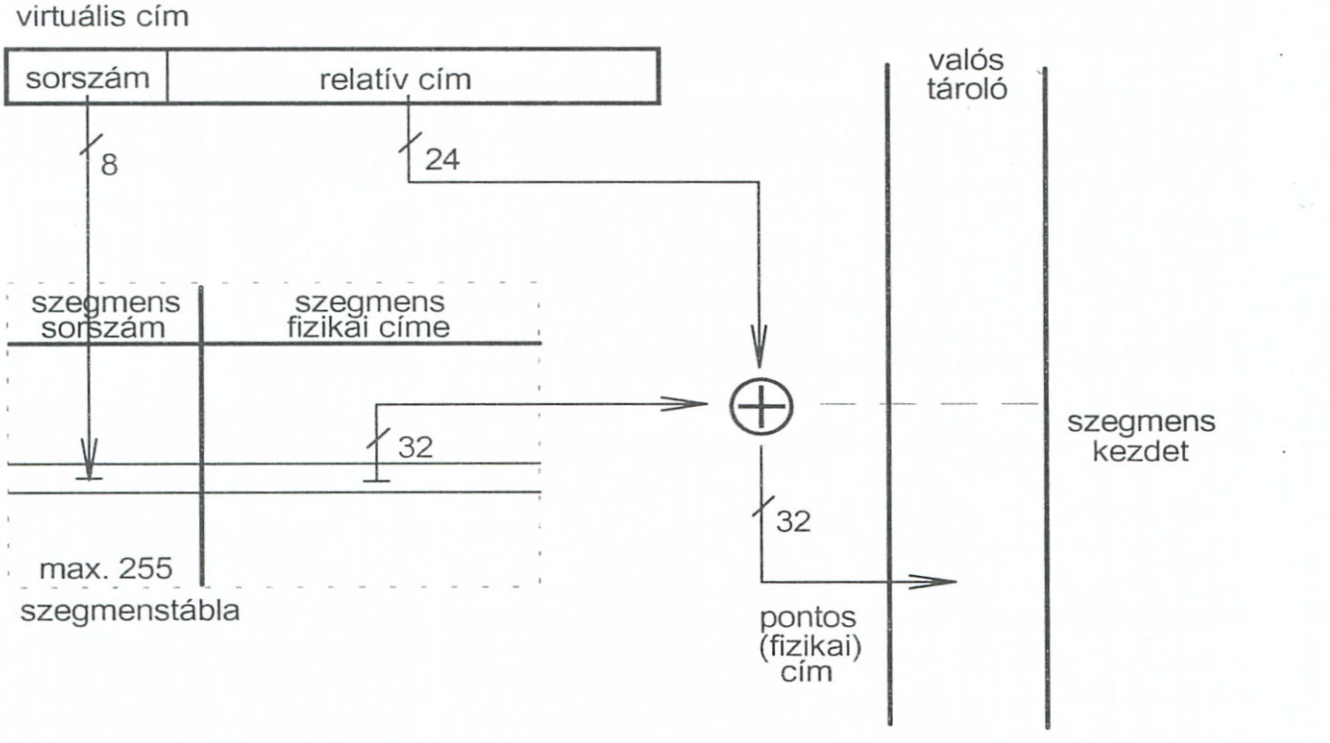
'hashing' eljárással(legtöbbször XOR művelet felhasználásával) képeznek új bitsorsozatot és ez képezi a 'tag'-et.

A következőkben a leggyakoribb címkiszámítási módokat mutatjuk be ábrák segítségével és rövid magyarázatokkal. *A bemutatott **címformátumok, mezőméretek** nem kapcsolódnak közvetlenül konkrét processzorok által használt megoldásokhoz, azok elsősorban a címkiszámítási mód elvi bemutatását szolgálják.* Ahol kapcsolat van egy konkrét mikroprocesszorral, ott ezt megemlítjük. A 6.fejezetben néhány ismert mikroprocesszort részletesebben is tárgyalunk és akkor az általuk használt címzési eljárásokra is utalunk.

Az ábrákon látható táblázatok valójában még más adatokat is tárolnak a szegmens-, vagy lapkezdet címeken kívül, ahogy ezt, a 4.1.pontban már említettük.

a.)Egylépcsős címképzés

Az egylépcsős címleképzés alkalmazásakor, a szegmens-, vagy laptáblázat tartalmazza a központi memóriabeli szegmens-, vagy lapkezdet fizikai címét. A táblázatban a keresett helyet a virtuális cím egy része, mint index jelöli ki, vagy egy külön regiszterben, tárolóban található a sorszám, az index értéke.



4-12.ábra: *Egylépcsős szegmenscím-kiszámítás*

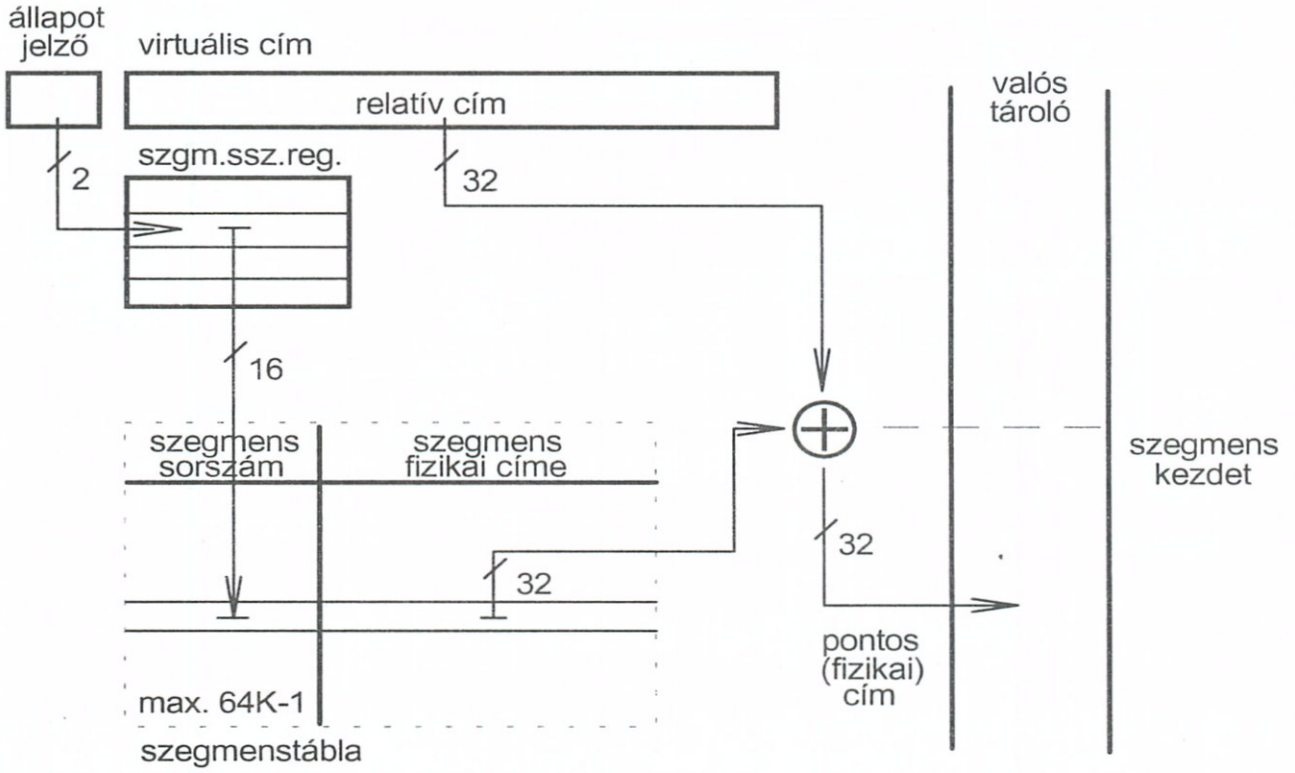
Szegmenscímek

A 4-12.ábrán a szegmenscímzés *azon változata* látható, amelynél a cím felső 8 bitje(**beágyazott sorszám**) jelöli ki a táblázat megfelelő sorát, amelyben a

szegmens kezdetének fizikai címe(32 bites) található. A pontos, fizikai cím kiszámításához a kezdőcím és az utasításban található relatív cím(eltérés, displacement, offset) összegét kell képezni.

A szegmenssorszám 8 bitje alapján a táblázatnak 256 bejegyzése lehet és egy-egy szegmens max. mérete, a relatív cím mérete alapján, 2^{24}=16MB lehet.

A szegmensek címzésének egy *másik változata*, amelynél a táblázat sorindexe gyanánt működő érték **nem a címben magában található**, hanem egy kijelölt regiszterben, a szegmensregiszterben(4-13.ábra). A felhasználandó szegmensregiszter kijelölése a processzor aktuális működési módjától, állapotától függ. Ezt jelzi a rajzon az állapotjelző.



4-13.ábra: *Egylépcsős szegmenscím-kiszámítás szegmensregiszter felhasználásával*

A pontos fizikai címet a táblázatból vett kezdőcím(32 bites) és a címben található 32 bites relatív cím összege adja.

Ha a táblázat sorindexét adó szegmensregiszter 16 bites, akkor összesen 2^{16}=64K bejegyzés képzelhető el és a szegmensek maximális mérete a 32 bites virtuális cím(relatív cím) alapján, 2^{32}=4GB lehet.

A szegmenstábla, a nagy mérete miatt, a főtárban helyezkedik el.

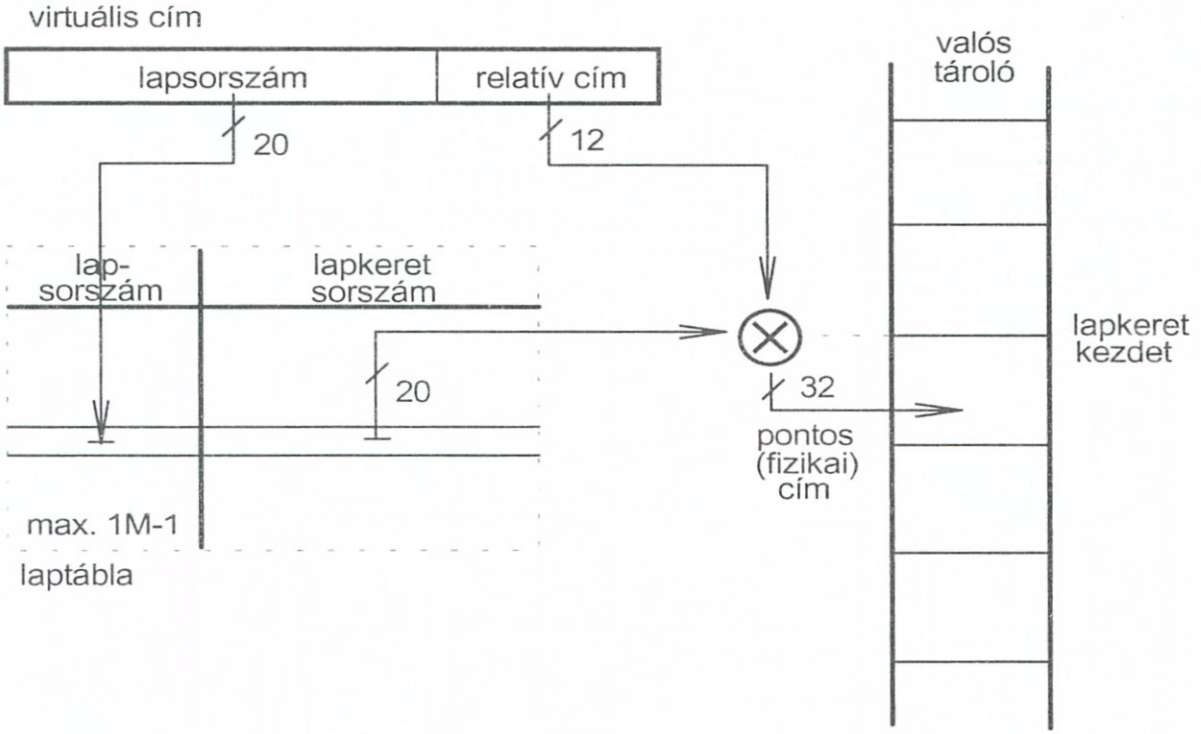
Ezt a címkiszámítási módot használják az i386/486-os, PowerPC 601-es processzorok is. Az aktuálisan használt szegmensekhez tartozó szegmensleírók

(deszkriptorok) a szegmensregiszter melleti cache-tárban, vagy a címfordító (TLB) cache-tárban helyezkednek el.

Lapcímek

A 4-14.ábrán látható a lapcímzés egy példája, amelynél a virtuális cím felső 20 bitje, mint logikai lapsorszám, jelöli ki a laptáblázat megfelelő sorát, amelyben a lapkeret sorszámát(20 bites) tároljuk.

A pontos fizikai cím a táblázatbeli lapkeret sorszám és az utasítás alsó 12 bitje által leírt relatív cím egyesítése(konkatenálása) adja.



4-14.ábra: *Egylépcsős lapcím-kiszámítás*

A megadott méretekkel a táblázatnak 2^{20}=1MB bejegyzése lehet és a lap mérete 2^{12}=4KB.

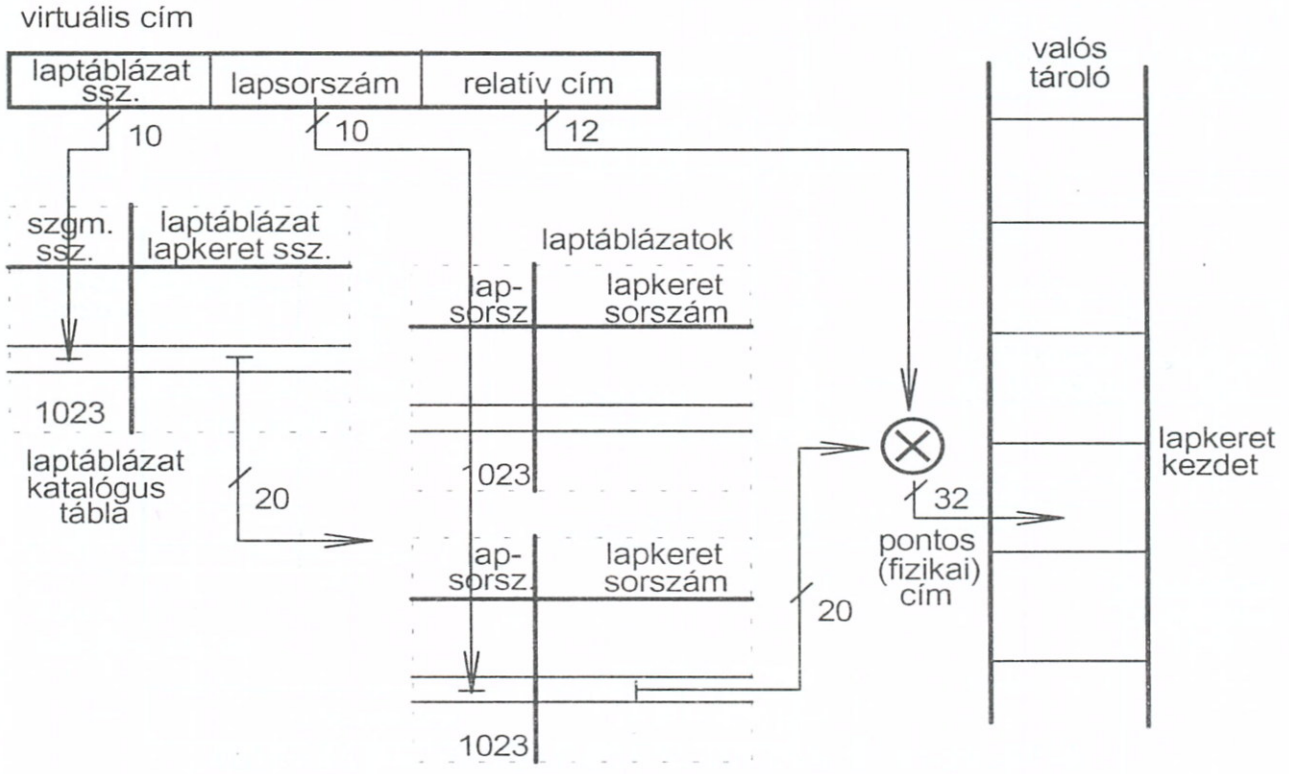
Mivel a laptáblázat mérete igen nagy, nagyobb mint 1MB, ezért azt a háttértárolókon kell elhelyezni. A leggyakrabban használt lapok adatait a címfordító cache-tárak(TLB-Translation Look-aside Buffers) fogadják be.

b.)Többlépcsős címképzés

A többlépcsős címkiszámítási eljárások vegyítik a szegmenscímzést a lapcímzéssel. A korszerű mikroprocesszorok mind alkalmasak már a lapozási technika használatára.

1.Többlépcsős lapcímzés

A 4-15.ábrán látható **kétlépcsős változat**nál a virtuális cím három részből áll, amelyek közül az első, a legfelső 10 bit a laptáblakatalógusban jelöli ki a használandó laptáblázat kezdőcímét, a második 10 bites rész, mint logikai lapsorszám, a laptáblázaton belül jelöli ki a lapkeret sorszámát és végül a legalsó 12 bit, mint relatív cím járul hozzá a pontos cím értékéhez.



4-15.ábra: *Kétlépcsős lapcím-kiszámítás*

A pontos cím a laptáblázatból kiválasztott lapkeret sorszám(20 bit) és a virtuális címből származó relatív cím(12 bit) egyesítéséből áll össze.

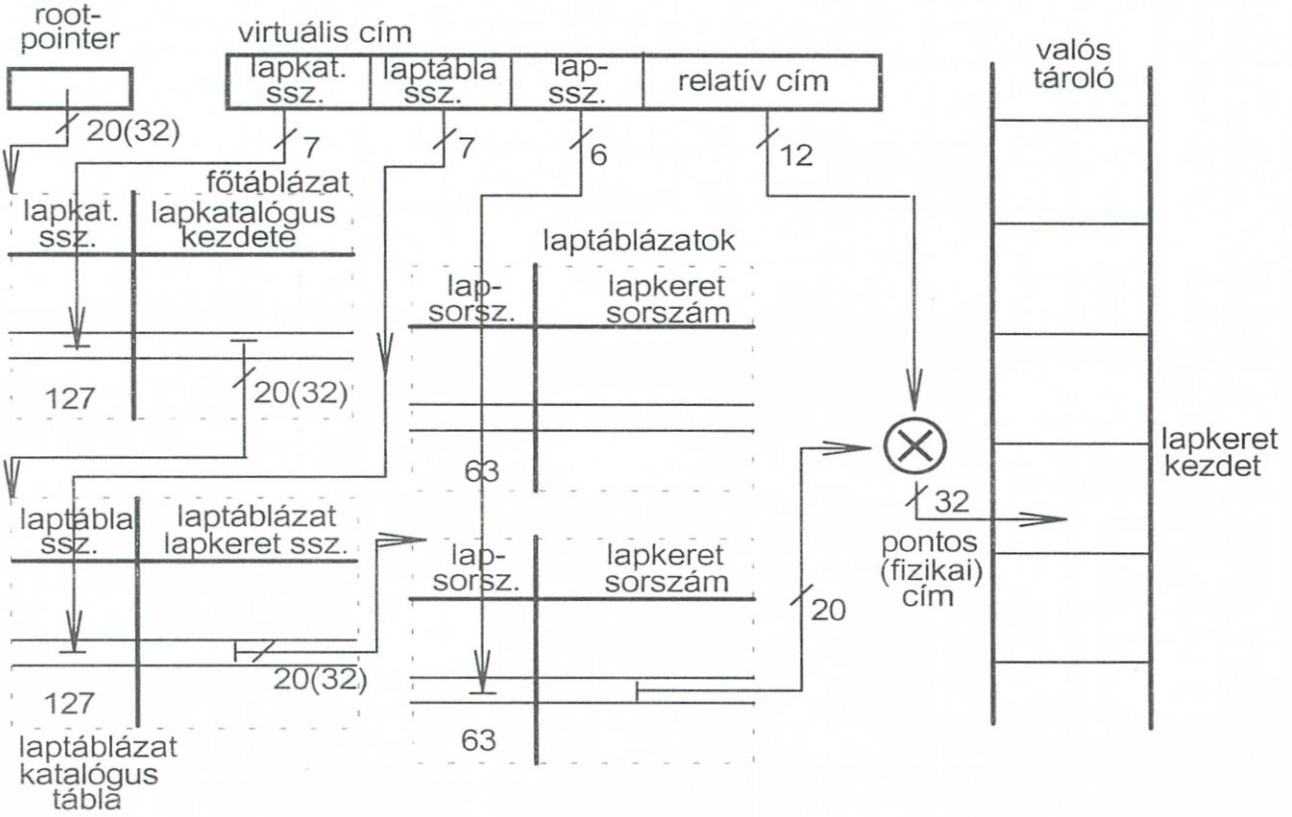
A megadott méretek mellett a laptábla katalógusban 2^{10}=1024 bejegyzés lehet, tehát ennyi laptáblázatot lehet kialakítani és egy-egy laptáblázatban ugyancsak 1024 lapnak a sorszámát lehet kijelölni. A lapok mérete pedig, a 12 bites relatív cím miatt, 2^{12}=4KB nagyságú.

A laptáblázatok mérete ugyancsak legalább 4KB, azaz 1 lap méretű.

Kétlépcsős címkiszámítási eljárást használ az Intel i860-as, a Motorola MC88000 processzor.

A 4-16.ábrán egy **háromlépcsős címkiszámítási eljárás**t mutatunk be. A virtuális cím 4 részből áll: a laptáblakatalógus kezdetét kijelölő laptáblakatalógus-sorszámból(7 bit), a laptáblázat kezdőcímét kijelölő laptábla-sor-

számból(7 bit), a lapot kijelölő logikai lapsorszámból(6 bit) és végül a relatív címből(12 bit).



4-16.ábra: *Háromlépcsős lapcím-kiszámítás*

A pontos, fizikai cím kiszámítása a laptábla kijelölése után, a megjelölt lapkeret sorszám és a relatív cím egyesítésével jön létre.

A megadott méretek mellett, a főtáblának és a lapkatalógus tábláknak egyenként 2^7=128 bejegyzése lehet és azok kiválasztott sorai megadják a lapkatalógus táblának, illetve a laptáblának a kezdőcímét(lapkeretsorszámát). A laptáblázatoknak 64 bejegyzése lehet és a lapkeretek sorszámát(20 bit) adják meg. Az egyes táblázatok mérete 1KB alatt van, de igen sok van belőlük, ezért csak a főtábla található a központi memóriában. Ennek helyét(root-pointer) és hosszát külön regiszterek tárolják.

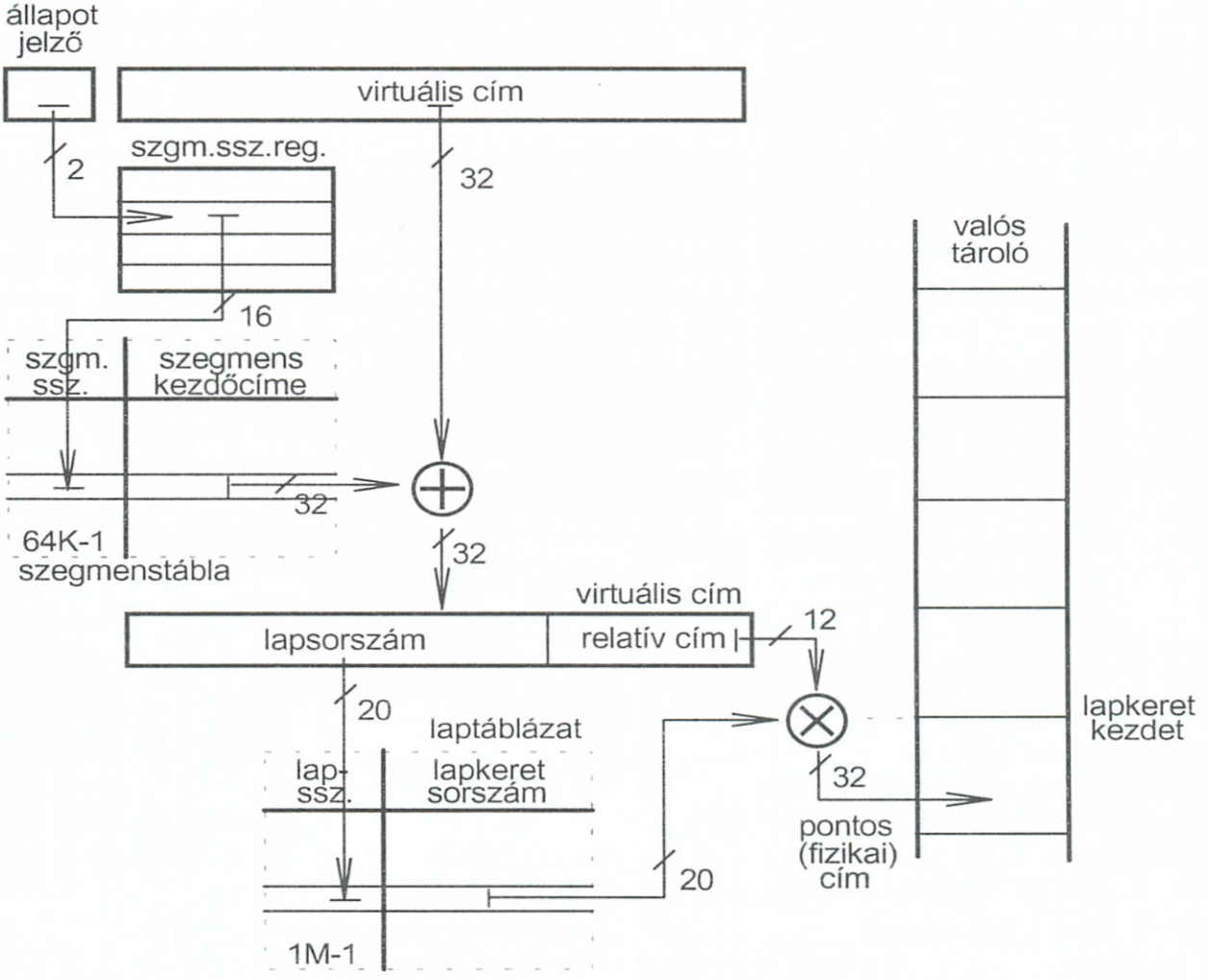
A háromszintes rendszer használata igen időigényes, ezért ehhez is alkalmazzák a TLB cache-t a leggyakrabban használt lapok adatainak tárolására.

3-lépcsős címkiszámítási módot használ a Motorola MC68040-es, a Zilog Z80000-es, a Sun Microsystems SPARC processzora; 5-lépcsős megoldást alkalmaz az MC68030-as processzor.

2.Többlépcsős szegmentált lapcímzés

A szegmentált lapcímzéseknél együtt alkalmazzák a szegmenscímzést és az azon belüli lapcímzést.

Kétlépcsős címkiszámítási megoldást mutat be a 4-17.ábra, amelynél a szegmens kijelölése nem a virtuális cím alapján, hanem egy erre a célra szolgáló szegmensregiszter tartalma alapján történik.



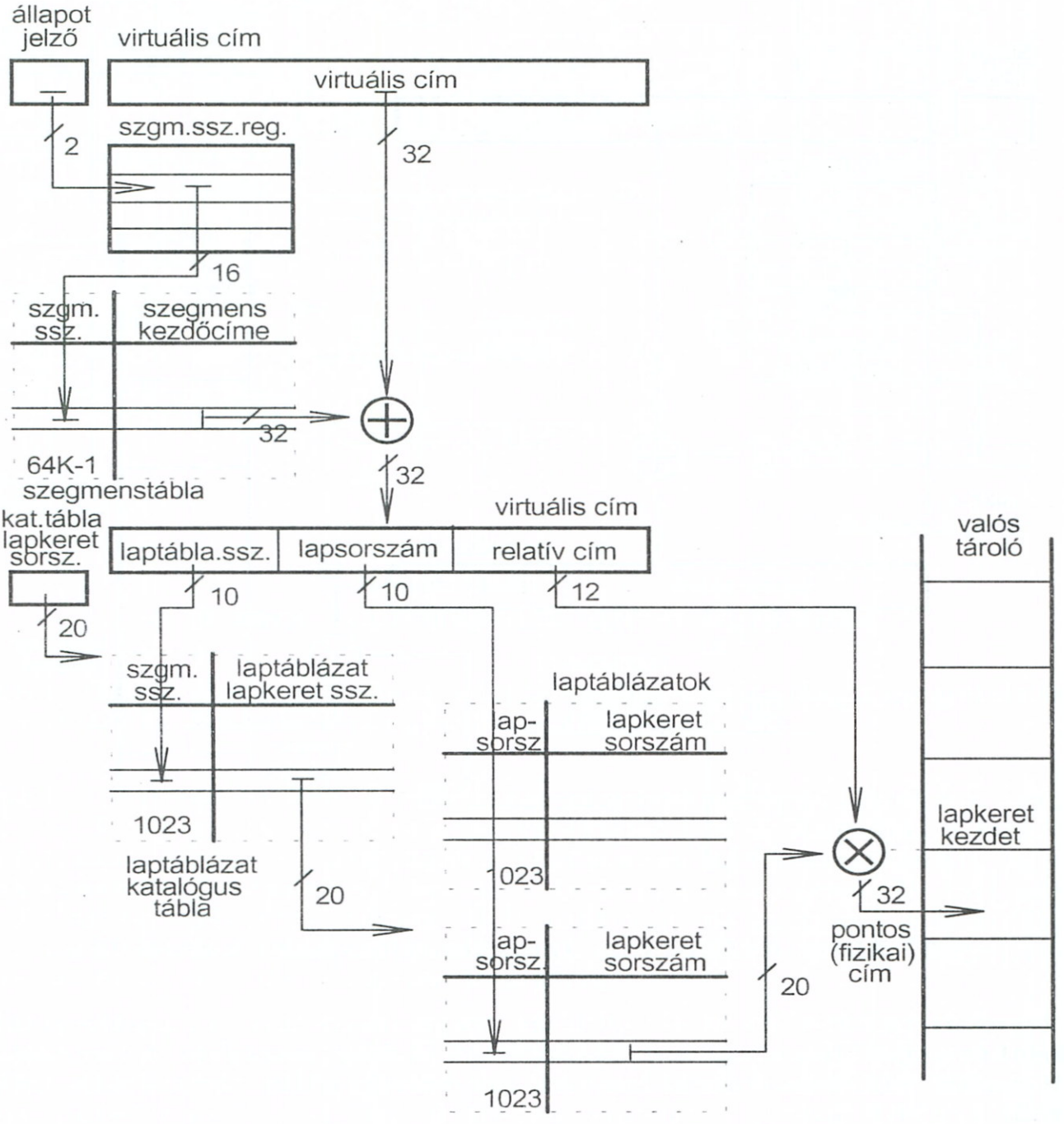
4-17.ábra: *Kétlépcsős, szegmentált lapcím-kiszámítás*

A szegmensregiszterben tárolt 16 bites szegmenssorszám a szegmenstáblázat indexeként kijelöli a szegmens kezdőcímét(32 bit), amelyhez a virtuális cím, mint relatív cím lesz hozzáadva. Az így keletkező újabb virtuális cím két részből tevődik össze, a lapsorszámból(20 bit) és a lapon belüli relatív címből(12 bit).

A keresett pontos címet a lapsorszám által kijelölt lapkeret sorszám és a relatív cím egyesítése(konkatenálása) adja. A kívánt szegmensregiszter kiválasztását egy állapotjelző jelzi.

Az ábrán jelölt méretek mellett, a szegmenstáblázatban 2^{16}=64K, a laptáblázatokban 2^{20}=1M bejegyzés tehető. Ennek következtében egy-egy táblázat helyigénye kb. 4MB, ami nem igazán használható. Ezért a háromlépcsős megoldás az, amit alkalmazni célszerű.

A következő, 4-18.ábra a **háromlépcsős címkiszámítási** megoldást mutatja be.



4-18.ábra: *Háromlépcsős, szegmentált lapcím-kiszámítás*

Ennél a megoldásnál a szegmensregiszterben tárolt 16 bites szegmenssorszám a szegmenstáblázat indexeként kijelöli a szegmens kezdőcímét(32 bit), amelyhez a virtuális cím, mint relatív cím lesz hozzáadva. Az így keletkező

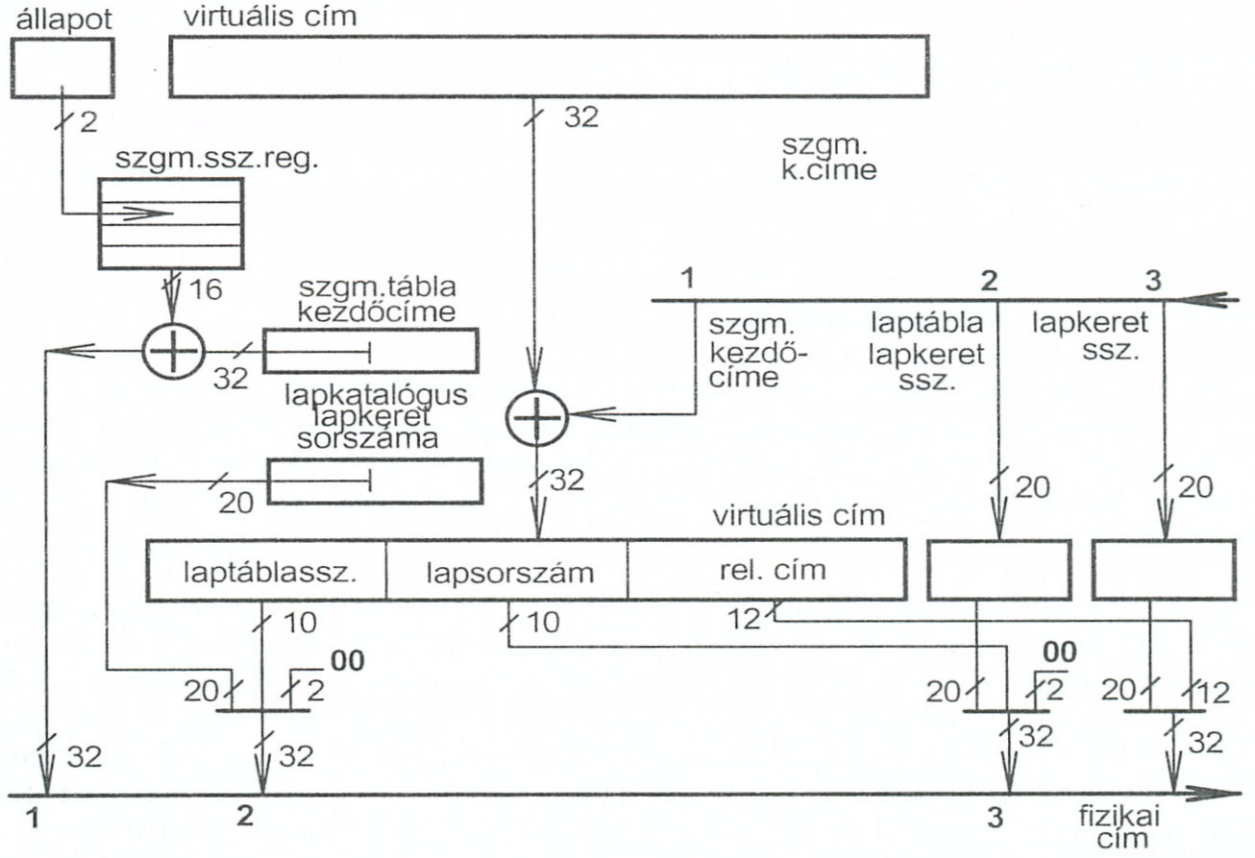
újabb virtuális cím három részből tevődik össze, a laptáblázat sorszámból(10 bit), a lapsorszámból(10 bit) és a lapon belüli relatív címből(12 bit).

A keresett tároló pontos címe a két lépcsőben kiválasztott lapkeret sorszámából és a relatív címből tevődik össze a két rész egyesítésével.

Az ábra megadott méretei mellett, a szegmenstáblázatban 2^{16}=64K, a laptábla katalógusban és laptáblázatokban 2^{10}=1024 bejegyzés tehető. Ennek következtében a szegmenstáblázat nagysága min. 256 Kbyte, egy-egy laptáblázat helyigénye pedig kb. 4KB, ami egy-egy lap méretének felel meg. A lapkatalógus helyét, lapkeret sorszámát, egy erre a célra szolgáló regiszter tartalma adja meg.

A szegmenstábla és a lapkatalógus tábla a memóriában, az egyes laptáblázatok a virtuális tárolóban találhatók meg.

Ezt a háromlépcsős címkiszámítási módszert alkalmazzák az i386/486-os processzorok is, avval a megszorítással, hogy a szegmenssorszám(szelektor) felhasználható hossza tulajdonképpen csak 13 bit(lásd 6.fejezet). A fizikai címkiszámítás folyamatának vázlatát mutatja be a 4-19.ábra.



4-19.ábra: *Háromlépcsős címkiszámítás folyamata*

Az ábra alapján, a végrehajtandó lépések a következők lehetnek: először az utasítás prefix által meghatározott szegmensregiszter tartalma, mint sorindex, a szegmenstáblázat kezdőcímének értékével együtt(1), a memóriában ki-

jelöli a szegmenstáblázat kívánt sorát, illetve a szegmens kezdőcímét(1). Ezt a kezdőcímet a virtuális címmel összeadva kapjuk az új virtuális címet, amelynek egyes részei kijelölik az egyes táblázatokban a szükséges lapkeret sorszámokat. A cím legfelső 10 bitje és a lapkatalógus lapkeretének sorszámát tároló regiszter tartalma, kiegészítve az alsó 2 bit 00 értékével(2), kijelöli a tárolóban elhelyezkedő lapkatalóguson belül az aktuális laptáblázat lapkeretének sorszámát(2). Ehhez a lapkeretsorszámhoz hozzátéve a virtuális cím második 10 bitjét, valamint kiegészítve 0-kal(3) megkapjuk a tárolóban a laptáblázaton belül a keresett lap lapkeretének sorszámát(3). Ezt az értéket végül egyesítve a virtuális cím alsó 12 bitjével, kapjuk a pontos, fizikai címet(4).

4.5.CÍMZÉS ÁLTALÁNOSÍTÁSA(CÍMZÉSI ELJÁRÁSOK III.)

A következő részben a címzésekkel összefüggésben néhány hatékonyságnövelő megoldást mutatunk be. Mivel a processzor és a memória(valamint más eszközök) közötti sebességkülönbség meglehetősen nagy, ezért a teljesítmény vagy a folyamatok párhuzamosításával, vagy az erőforrások többszörözésével növelhető.

4.5.1.Tárolóhoz fordulás gyorsítása

A tárolóhoz fordulás gyorsítására két, a címzéssel kapcsolatos megoldást mutatunk be igen röviden.

a.)Memóriatömbök használata

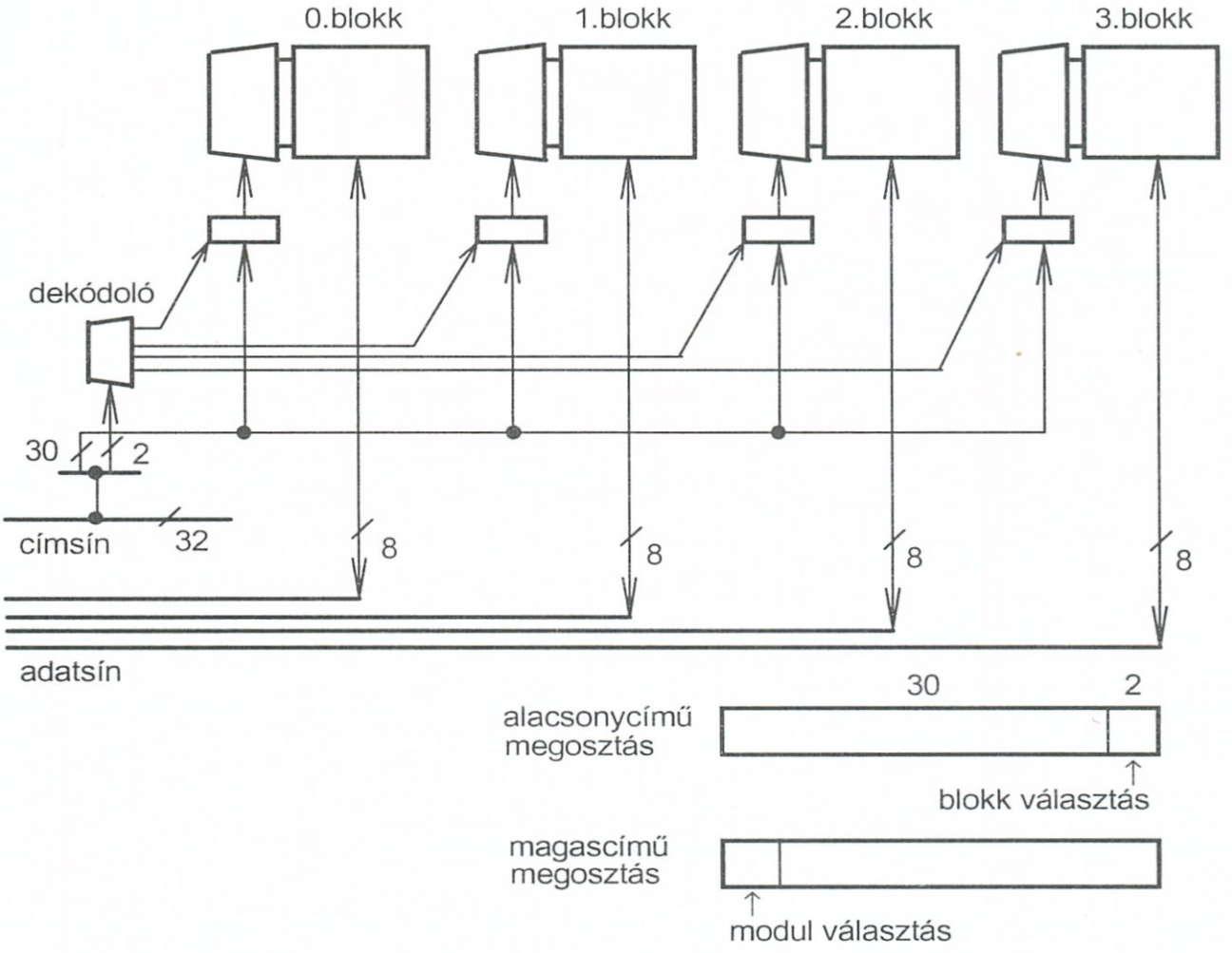
A folyamatok párhuzamosításán alapul a memóriatömbök használatának átlapolása, a **'memory interleaving'** módszere, ahogy erre már a 3.5.2.pontban is hivatkoztunk. A memóriát, címzés szempontjából blokkokra bontják, mindegyikhez külön hozzáférési lehetőséggel. A felbontás 2, 4, 8 tömbre történhet(4-20.ábra).

Például 32 bites szavak esetében, a 4 tömbre bontás azt eredményezi, hogy az első tömbben a 0,4,8,12,... című, a második tömbben az 1,5,9,13,... című, a harmadik tömbben a 2,6,10,14,... című és végül a negyedik tömbben a 3,7,11,15,... című tárolóhelyek érhetők el. Így egyetlen cím kiküldésével, amelynek alsó 2 bitje 0 értékű(ez biztosítja, hogy 4-gyel osztható címet címezzünk meg, illetve ezt a 2 bitet arra használjuk fel, hogy a memóriatömböt kiválasszuk), egyszerre 4 byte-nyi adatot tudunk kiolvasni. Különböző címeket is megadhatunk az egyes tömbök számára és akkor tetszőleges 4 adatot olvashatunk ki a memóriából.

A memória blokkokra bontása kétféle címzéssel érhető el:

- *alacsony című megosztással*(low order interleaving); ez az előbb ismertetett megoldás, amely a memóriaolvasás gyorsítására gyakrabban alkalmazott módszer;

- *magas című megosztással*(high order interleaving); amely a modularizált, egymástól független memóriablokkok használatát teszi lehetővé.



4-20.ábra: *Átlapolt memóriatömb kezelés*

b.)Átlapolt címzés

A memóriaelérés gyorsításának egy másik lehetőségét adja az **átlapolt címzés** (interleaved addressing) alkalmazása.

A memória elemi eszközeinek tárgyalásakor meghatározásra került a tárakra jellemző elérési idő és ciklusidő fogalma. Az elérési idő az az időtartam, amely a címzés megindítása és az adat megjelenése között eltelik, míg a ciklusidő a tárnak az a legrövidebb időtartama, amelynek két tárhozfordulás között el kell telnie.

A statikus RAM(SRAM) tárolók esetében az elérési- és a ciklusidő gyakorlatilag megegyezik, míg a dinamikus RAM(DRAM) tárolóknál, amelyekből a központi tár is felépül, a ciklusidő közel kétszerese az elérési időnek. Ez az időtöbblet a DRAM-ok feléledési(recovery) idejéből adódik.

Az átlapolt címzési technika a tárciklusnak ezt a feléledési időtartamát használja fel arra, hogy egy újabb címzési folyamatot elindítson a processzor. Ez a címzés azonban nem vonatkozhat ugyanarra a tárolótömbre, hiszen az nem tud címet fogadni. Ehhez a tárolót tömbökre(bank-okra) kell osztani, általában 2 tömb elegendő és felváltva kell olvasni, vagy írni az egyik, vagy a másik tömbbe.

A memóriaciklusok fenti elvű átlapolását valósítják meg chip-en belül, az újabban terjedő EDO(Extended Data Output) DRAM-ok is.

4.5.2.Eszközök címzése

A processzor a számítógép különböző erőforrásaihoz, perifériáihoz úgy tud hozzáférni, hogy meghatározza az eszközt és az adatátvitel irányát.

Az igénybevenni kívánt eszköz kiválasztásának egyik lehetséges (hardver) módja az eszközhöz tartozó eszközválasztó jel(S_i - select) kijelölése és az írás/olvasás irányának(R/W* - read/write) a meghatározása.

A programozó oldaláról tekintve, az egyes írható/olvasható eszközök úgy jelennek meg, mint különböző lehetséges tárolási helyek, amelyek kényelmes kezelési lehetőségét azok címezhetősége adná. Azaz, ha a különböző eszközöket ugyanúgy kezelheti, mint egy közönséges tárolót.

A különböző eszközöknek, elsősorban is a memória és a perifériák fentiek szerinti elérésére, az alábbi módszerek használhatók:

- Közvetlen, elkülönült elérési mód(**isolated addressing**) alkalmazása esetén, a memória, illetve a perifériák elérésére használt utasítások műveleti kódja közvetlenül meghatározza az írás/olvasás helyét az M/IO* (memory/input-output) jel beállításával. A választás után, az egyes eszközök ugyanúgy címezhetők. Ez tulajdonképpen a címtartomány megduplázását eredményezi. Mivel a perifériák, input/output eszközök száma lényegesen kisebb, mint a memória címezhető tartománya, ezért a címvonalak(címbitek) egy részére ilyenkor nincs szükség, elegendő például csak az alsó 16 bit erre a célra, amely 64K-nyi használható címet jelent.

- Beágyazott, a főtárral azonos elérési mód(**memory mapped addressing**), amely esetében a perifériák ugyanúgy címezhetők, mint a memória. A perifériák, mint a memóriaterület egyik részén elhelyezkedő tárolóhelyek használhatók. Ennek következtében bármelyik memóriára hivatkozó utasítás, egyaránt elérheti a memóriát és a perifériákat. Legtöbb mikroprocesszor esetében a memórialeképzésű perifériakezelés a használható eljárás. (Ezt a perifériakezelési eljárást a DEC PDP-11-es gépcsaládnál vezették be először.)

4.5.3.Program- és adattárolás

Mivel a programutasítások és az adatok felhasználási módja különbözik, az egyik esetben csak olvasás történik, míg a másik esetben az írás és olvasás egyaránt feladat, meggyorsítaná a feldolgozást, ha a **programtárolás és az adattárolás** szétválna.

Azoknál a gépeknél, amelyeknél a külön tárolás megoldott, azaz önálló memóriablokkok vannak az utasítások és az adatok tárolására, a processzornak a memóriához forduláskor tudnia kell, hogy programutasítást, vagy adatot kell elérnie. Ennek legegyszerűbb megoldási módja, hogy a használható címtartomány felezésével, a cím legfelső bitjét a választás(P/D* - program/data) vezérlésére használják fel. Ez a megoldás megakadályozza a program esetleges felülírását is.

4.5.4.Működési mód megválasztása

A processzorok több működési mód(pl.: rendszer/felhasználói, védett/nem védett, stb. mód) valamelyikében dolgozhatnak és a működési módnak megfelelően használhatják az egyes eszközöket, így a memóriát is.

A működésmóddal meghatározott hozzáférési jogok befolyásolják, többnyire korlátozzák a tárolt adatokkal való műveletvégzés(csak olvasható, írható/olvasható, csak végrehajtható, stb.) lehetőségét.

A különböző működésmódokkal kapcsolatos memóriahasználat, memóriaterület kiválasztás szabályozható külön vezérlő jellel(pl.: S/U* - supervisor/user) is, de a legcélszerűbb a memória címzéssel szabályozott kiválasztása. Ez gyakorlatilag a memória felosztását eredményezi rendszer(supervisor) és felhasználói(user) területre, amelyek külön-külön használhatók.

Az i386/486-os processzoroknál a lapleírókban(deszkriptorokban) lévő S/U* bit határozza meg a lap által kijelölt memóriaterület használati lehetőségét, a rendszer és a felhasználó között, amely az előző elvnek egy leegyszerűsített, lapszintű változata.

4.5.5.Adatformától függő címzés

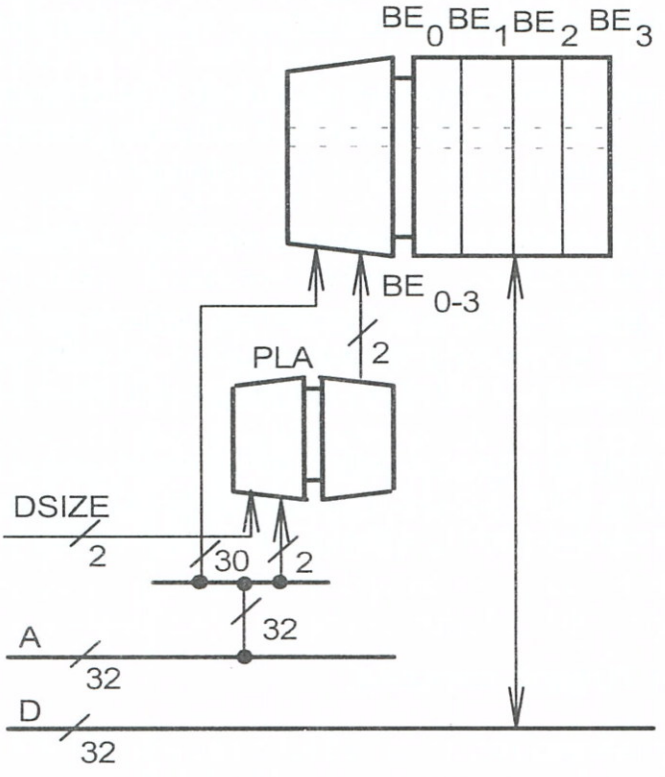
A memóriahasználat során, a processzor által igényelt adatméret igen változó lehet. Többnyire 1, 2, 3 vagy 4 byte egyidejű átvitelére van szükség. A működés gyorsítása csak úgy érhető el, ha az adatsínen mindig a lehetséges maximális adatméretnek megfelelő adatszó kerül átvitelre. Ez a mikroprocesszorok 16, 32 bites adatsínje miatt célszerüen 2, vagy 4 byte. Az átvitel gyorsítását segíti elő az is, ha a memóriából nem átlapolódó címkijelöléssel olvassuk az adatokat, azaz csak olyan 2 vagy 4 byte-os szavakat olvasunk ki, amelyek kezdete 2-vel, vagy 4-gyel osztható címeken helyezkedik el.

A 32 bites adatsínnel működő processzorok esetében nem mindegy, hogy a 4 byte-os adatterületek melyik byte-jától kezdődően kell kiolvasni az adatot. Az előzőekben leírtak szerint a legjobb, ha mindig csak 4 byte-onként, 4-gyel osztható címektől kezdődően olvassuk az adatokat. Ugyanakkor szükség van arra is, hogy ennél kisebb adatot és tetszőleges helyről is kiolvashassunk.

A problémák megoldásához a processzorok két jelet használnak a címkiválasztás segítésére:

- adatméret(**DSIZE**=Data Size) jel, amelynek értéke megszabja, hogy a processzor 1, 2, 3 vagy 4 byte-ot olvasson ki egy 4 byte-os szó területéről, a memóriacím alsó 2 bitjének értékétől függő címtől kezdődően;
- byte engedélyező(**BE_{0-3}** =byte Enable) jelek, amelyek értéke meghatározza azt, hogy egy 4 byte-os memóriaterület mely byte-jai legyenek kiolvashatók.

A tárolókezelés vázolt módját a 4-21.ábra mutatja be. A byte engedélyező jelek lehetséges értékeit a 4-1.táblázat tartalmazza.



4-21.ábra: *Adatmérettől függő címzés logikai vázlata*

Az egyes processzor típusok(RISC/CISC) a gyorsítás érdekében, az adatok memóriabeli kezelését, elhelyezési lehetőségét is szabályozza. Az alkalmazott elrendezési módok az alábbiak lehetnek:

- **Rendezett adatelhelyezés**(data alignment); a memória címzése szempontjából a 4-gyel osztható cím a kiinduló pont és ezen a 4 byte-os szón belül lehet kiválasztani a kívánt adatterületet. Ezt a tárolókezelési módot alkalmazzák a RISC processzorok az addatforgalom gyorsítása végett.

  A RISC processzorok esetében, a 4 byte-os szón belül csak meghatározott elhelyezkedésü adatok érhetők el és az adathossz 2 hatványa(1, 2, 4) lehet csak. A használatos kombinációkat a 4-1.táblázat tartalmazza.

- **Rendezetlen adatelhelyezkedés**(data misalignment); amelynél a memória bármelyik byte-ja közvetlenül címezhető és a használt adathossz 1, 2, 3, 4 byte egyaránt lehet.

  A CISC processzorok esetében ez a módszer használatos; a tárolóhoz fordulása ezekben az esetekben két részből áll, az első részben történik a méret beállítása, majd az adatok átvitele. A kapcsolódó, használható kombinációkat a 4-1.táblázat tartalmazza.

4-1.táblázat: *RISC és CISC processzorok adatmérettől függő címzése*

Adat-méret	Proc. típus	DSIZE	A_{1-0}	BE_0	BE_1	BE_2	BE_3	Átvitt byte szám
byte	RISC	0 0	0 0	1	0	0	0	1
	RISC	0 0	0 1	0	1	0	0	1
	RISC	0 0	1 0	0	0	1	0	1
	RISC	0 0	1 1	0	0	0	1	1
félszó	RISC	0 1	0 0	1	1	0	0	2
	RISC	0 1	1 0	0	0	1	1	2
	CISC	0 1	0 1	0	1	1	0	2
byte	CISC	0 1	1 1	0	0	0	1	1
3 byte	CISC	1 1	0 0	1	1	1	0	3
szó	RISC	1 0	0 0	1	1	1	1	4
3 byte	CISC	1 0	0 1	0	1	1	1	3
félszó	CISC	1 0	1 0	0	0	1	1	2
byte	CISC	1 0	1 1	0	0	0	1	1

4.6.VÉDELMI LEHETŐSÉGEK

A tárolókezelő rendszer feladatai között, ahogy azt a 4.1.pontban már érintettük, a címkiszámítás mellett, a tárolt programok, adatok védelme is fontos szerepet tölt be. Ez más megközelítésben, a következő területekkel való foglalkozást jelenti:

- a memóriaterület védelme; ez a címzések helyességének ellenőrzését jelenti, amelyet a MMU-nak mindig el kell végeznie, megakadályozandó a nem létező és hibásan megadott címek használatát;

- a rendszerprogramok védelme a felhasználó beavatkozásaitól;
- a felhasználók(pontosabban feladataik) egymástól való védelme;
- a tárolt adatokhoz történő hozzáférési lehetőségek ellenőrzése.

Az első kérdéssel részletesebben nem kell foglalkozni, mert a megoldás értelemszerűen adott.

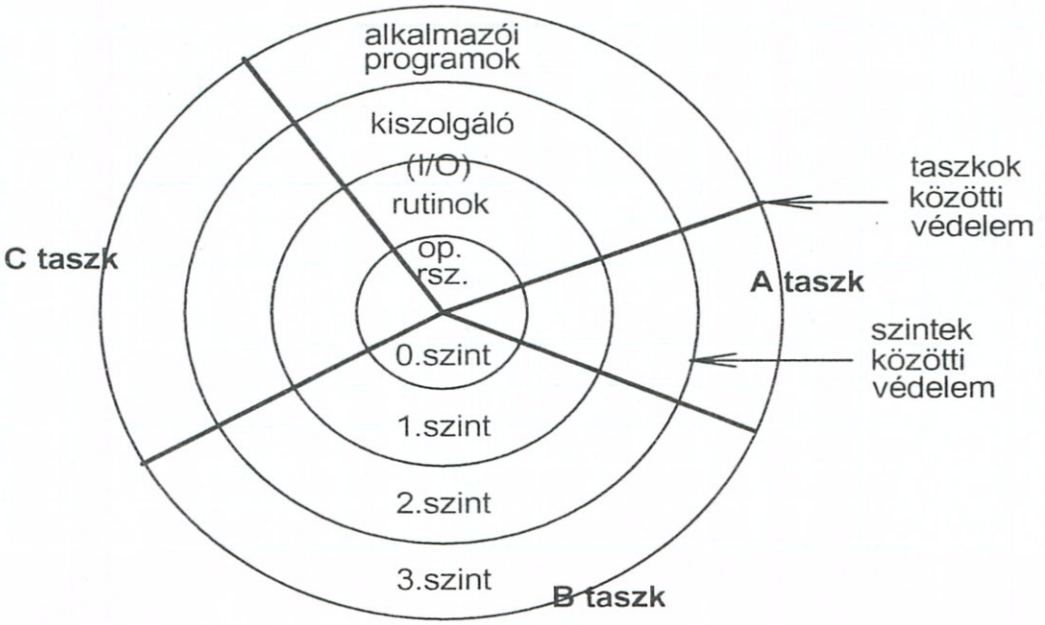
a.)Programok és felhasználói feladatok védelme

A védelmi módszerek két típusa alkalmazott:

- hierarchikus rendszer(ring protect system),
- nem hierarchikus rendszer(capability based protect system).

Hierarchikus védelmi rendszer

A hierarchikus védelmi rendszert alkalmazzák az Intel processzorai, az i286-os, i386/486-os processzorok is, amelyeknél 4-szintű hierarchiát alkalmaznak(4-22.ábra).



4-22.ábra: *Hierarchikus védelmi szerkezet*

Az ábrán a koncentrikus körök a programok (és adatok) egyes védelmi szintjeit jelzik, míg a körcikkek a felhasználói területeket(task-okat) jelzik. A legbelső kör a legmagasabb privilégiumot jelzi és evvel csak az operációs rendszer rendelkezhet. Kifelé haladva csökkennek a lehetőségek, a 2. és a 3.sáv a különböző, rendszerkiszolgáló programok, mint például I/O rutinok, védelmi szintjeinek felel meg. A legkülső sáv a felhasználói programok területe, a legalacsonyabb védelmi szintű terület.

A hierarchikus rendszerben a **programok** más rutinokat csak a saját védelmi szintjükön, vagy magasabb szinten hívhatnak. A magasabb védelmi szintet csak speciális ellenőrzött módon, ú.n. kapukon keresztül lehet elérni. Ugyancsak kapukon keresztül lehet elérni egy másik feladat(task) valamilyen rutinját. Ezeknek a megoldásoknak az oka, egyrészt a megbízhatóság fenntartása (ezért nem lehet alacsonyabb védelmi szintű programot elindítani), másrészt a magasabb védelmi szintű rendszerprogramok védelme a felülírástól, átírástól.

A feldolgozó programok **adatokat** csak saját szintjükön, illetve alacsonyabb védelmi szinten érhetnek el. Ennek a szabálynak is a megbízható működés elősegítése a célja.

A **szintek közötti védelmet** az egyes szintek saját veremtárolója is biztosítja.

A **feladatok**(task-ok) **közötti védelmet** az egyes feladatok saját deszkriptor táblája(LDT=Local Descriptor Table) biztosítja.

A hierarchikus rendszer előnye a hardver általi megvalósítása és ebből eredő gyorsasága.

Nem-hierarchikus védelmi rendszer

A nem-hierarchikus rendszerekben, minden feladathoz(task-hoz) egy műveleti tábla van hozzárendelve, amely meghatározza azokat a 'task' által elvégezhető műveleteket, amelyek más feladatokat érintenek. Ahhoz, hogy egy olyan műveletet végezhessen egy feladat, amely egy másik feladatot is érint, rendelkeznie kell ennek lehetőségével a műveleti táblájában.

Ezt a fajta védelmet, amely igen bonyolult rendszert alkot, hardver úton még nem, csak operációs rendszer segítségével hozták eddig létre.

b.)Adatok védelme

A feladatok által használt adatok védelme érdekében, a szegmensekhez, lapokhoz való **hozzáférési jogok**at szabályozza a védelmi rendszer. A leggyakoribb hozzáférési jogok:

- **olvasási jog**(read access); a tárolóhoz(laphoz, szegmenshez) forduló feldolgozás tetszőleges adatot kiolvashat az adott területről;
- **írási jog**(write access); a feldolgozás átírhatja a tárolóterület adatait, beírhat új adatokat, törölheti az ott lévő adatokat;
- **végrehajtási jog**(execute access); a feldolgozás a tárolóterületen található kódot elindíthatja; végrehajtási jog csak programot tartalmazó laphoz, vagy szegmenshez rendelhető hozzá, adatszegmenshez nem.

Az elérési jogokat a szegmensekhez, lapokhoz tartozó leírók(deszkriptorok) tartalmazzák, ezért a folyamatoknak előbb a deszkriptortáblákban kell ellenőrízniük a jogosultságukat az adott tárolórészhez.

A szegmentált tárolókezeléshez kapcsolódó védelem valamivel jobb, mint a lapokhoz tartozó, mert az előbbivel például egy teljes programot védhetünk, míg lapozásos módszernél minden laphoz külön hozzá kell rendelni a védelmet és ez kevésbé biztonságos.

További gond, hogy a védelemnek ez a formája a tárolóterülethez van hozzárendelve és nem a felhasználóhoz. Ennek kiküszöbölésére alkalmaznak kétszintű védelmi rendszert, amelyben a felhasználóhoz kapcsolódó jogokat a szegmens- és lapelérési jogoktól külön tárolják.

4.7.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Mit értünk tárhierarchia alatt? Milyen elemei vannak a processzor köré kialakított tárhierarchiának?
2. Mi a tárkezelő rendszer feladata? Milyen jellegű védelmet kell biztosítania a tárkezelő rendszernek?
3. Mire szolgálnak a regiszterek? Mit értünk regisztertár alatt?
4. Milyen követelményeket támaszthatunk a regisztertárakkal szemben? Milyen elérési lehetőséggel kell rendelkeznie a regisztertáraknak?
5. Milyen tárkezelési módszereket használnak a regisztertárak esetében?
6. Mi a 'regiszter-bank' technika jellemzője?
7. Mi a 'regiszter-ablak' technika jellemzője? Milyen programozástechnikai eredetű feladatot lehet az ablaktechnika segítségével egyszerűbbé és gyorsabbá tenni?
8. Mire szolgálnak a cache-tárak? Milyen elv szerint történik az adatok visszakeresése a cache-tárból?
9. Milyen formájú az adatátvitel a cache-tár és a főtár között? Soroljon fel néhány, a cache-tárakat leíró jellemzőt.
10. Mit értünk cache-találat(cache-hit) alatt? Mit jelent a cache-miss fogalom?
11. Milyen típusait ismeri a cache-táraknak?
12. Milyen részekből állnak a cache-tárak? Mit tartalmaz a cache-tár 'tag'-része? A cache-tárak megbizható működéséhez milyen jelzőbitek megléte szükséges legalább?
13. Milyen jellemzőkkel rendelkezik a teljesen asszociatív cache-tár? Vázolja fel a teljesen asszociatív cache-tár felépítését és a cím alapján történő keresés kapcsolatait!
14. Milyen előnyökkel és hátrányokkal rendelkezik a teljesen asszociatív cache-tár használata? Hány összehasonlító áramkör használata szükséges egyidejüleg az asszociatív cache-tár esetében?
15. Milyen jellemzőkkel rendelkezik a közvetlen leképzésű cache-tár? Miért előnyös és hátrányos a közvetlen leképzésű cache-tár használata? Hány összehasonlító áramkör használata szükséges egyidejüleg a közvetlen leképzésű cache-tár esetében?
16. Vázolja fel a közvetlen leképzésű cache-tár felépítését és a cím alapján történő keresés kapcsolatait! A cache-be töltött blokk hány helyre kerülhet?

17. Milyen jellemzőkkel rendelkezik a csoport asszociatív cache-tár? Vázolja fel a csoport asszociatív cache-tár felépítését és a cím alapján történő keresés kapcsolatait!
18. Miért előnyös, vagy hátrányos a csoport asszociatív cache-tár használata? A cache-be töltött blokk hány helyre kerülhet?
19. Hány összehasonlító áramkör használata szükséges egyidejüleg az csoport asszociatívcache-tár esetében?
20. Miért nem okoz gondot az utasításokat tartalmazó cache-tárak tartalmának karbantartása? Miért nem szükséges az utasításokat tartalmazó cache-tárak aktualizálása?
21. A cache-tár feltöltésénél milyen módszereket alkalmaznak?
22. Mit értünk a cache-tárak aktualizálása alatt és milyen módszereket alkalmaznak? Ismertesse az egyes aktualizálási módszereket, azok előnyeinek és hátrányainak a bemutatásával!
23. Miért szükséges a megfelelően kiválasztott helyettesítési stratégia a cahce-tárak adatkarbantartásánál? Melyik módszer a leggyakrabban alkalmazott helyettesítési stratégia a cache-tárak esetében?
24. Mi jellemzi az LRU(least recently used) helyettesítési stratégiát? Melyik blokkot cseréli ki az LRU helyettesítési stratégia? Milyen eszközökkel valósítható meg az LRU helyettesítési stratégia?
25. Mit értünk a cache-tárak és a főtár adategyezősége alatt? Milyen szabályrendszer alapján történik az adategyezőség biztosítása? Mit jelent a MESI rövidítés?
26. Miért szükséges a rendelkezésre álló tárolóterület egységes kezelése? Milyen tárolóeszközök vonhatók be a virtuális tárkezelés hatáskörébe?
27. Mit értünk virtuális tárkezelésen? Mit értünk virtuális címtartomány alatt? Milyen címzési módnak tekinthető a virtuális címzés?
28. A processzor mely része vesz részt a virtuális címek fizikai címekké alakításában? Mi a feladata a memóriakezelő egységnek (MMU) a virtuális címek kapcsán?
29. Mit tartalmaznak a szegmens- és laptáblázatok? Vázolja fel a címfordítás lényegét tükröző ábrát.
30. Milyen címekkel dolgozik a cache-tár, ha a memóriakezelő egység a processzor és a cache-tár között, illetve ha a memóriakezelő egység a cache-tár és a főtár között van?
31. Mit értünk szegmens alatt? Milyen a szegmensek mérete? Milyen a szegmentált tárkezelés helykihasználása?
32. Milyen szegmensbetöltési módszereket alkalmaznak? Ismertesse ezek jellemzőit!
33. Mi az előnye, illetve a hátránya a szegmentált tárkezelésnek?
34. Mit értünk lap alatt? Mekkora a lapok szokásos mérete? Mit értünk lapkeret alatt?
35. Mi a lapozásos tárkezelés lényege? Milyen lapbetöltési módszert használnak a lapozásos tárkezelésnél?
36. A lapozási igény által megszakított utasításfeldolgozás milyen módszerekkel folytatható? Melyik módszert alkalmazzák leginkább?

37. Milyen lehetőségek vannak a cserélhető lapok körének kiválasztására?
38. Mit értünk 'working set' alatt? Milyen lapváltási technikákat alkalmaznak? Ismertesse azok lényegét!
39. Mit jelent a virtuális címek leképzése fizikai címekre?
40. Vázolja fel az egylépcsős szegmenscímzés lényegét! Vázolja fel a szegmensregiszterek felhasználásával végrehajtott egylépcsős szegmenscímzés lényegét!
41. Vázolja fel az egylépcsős lapcímzés lényegét! Vázolja fel a többlépcsős lapcímképzés folyamatát tükröző ábrát!
42. Milyen címképzési mód a szegmentált lapcímzés?
43. Ismertesse az i386/496-os processzorok címkiszámítási folyamatának vázlatát rajz segítségével!
44. Miből adódik a memóriatömbök(memory interleaving) kialakításának gyorsító hatása? Milyen adatelérési formát eredményez a 'memory interleaving'?
45. Az adatkiolvasás gyorsítása melyik címzési móddal(alacsony, vagy magas című megosztással) érhető el? A magas című megosztással(high order interleaving) milyen memóriakezelés érhető el?
46. Milyen módon gyorsítható az adatok elérhetősége tárolóhoz fordulás során? Mit jelent az átlapolt címzési eljárás (interleaved addressing)? A memória milyen strukturális változtatása szükséges az átlapolt címzési technika alkalmazásához?
47. Milyen módszereket alkalmaznak az egyes eszközök kiválasztására, kijelölésére? Milyen eszközkiválasztási eljárás az 'isolated addressing' mód? Mit értünk a beágyazott (memory mapped) eszközelérési mód alatt?
48. Milyen módon lehet szétválasztani a program- és adattárolási területet a címzés segítségével? Milyen előnye van a program- és az adattárolási terület szétválasztásának?
49. Milyen módon lehet címzés segítségével szétválasztani a felügyelő program memóriahasználatát a felhasználói memóriahasználattól? Milyen előnye van a felhasználói memóriaterület és a felügyelő programrendszer által használt memóriaterület szétválasztásának?
50. Miért előnyös az adatok 4-gyel osztható címektől kezdődő kiolvasása 32 bites adatsín használata esetén?
51. Mit értünk rendezett adatelhelyezés(data alignment) alatt? Milyen címeken kezdődő adatok érhetők el rendezett adatelhelyezkedés(data alignment) esetén?
52. Mit értünk rendezetlen adatelhelyezkedés(data misalignment) alatt? Milyen címeken kezdődő adatok érhetők el a rendezetlen adatelhelyezkedés(data misalignment) esetén?
53. A RISC processzorok milyen adatelhelyezkedés használatát feltételezik?
54. Milyen védelmi funkciókat tölt be a memóriakezelő rendszer? Milyen védelmi rendszereket használnak a tárolt adatok védelmére?
55. A hierarchikus rendszerben mely programok rendelkeznek a maximális prioritással?

56. A hierarchikus rendszerben a programok milyen (al)programokat hívhatnak meg és milyen módon? A hierarchikus védelmi rendszerben a feldolgozó programok milyen adatokhoz férhetnek hozzá?
57. Milyen módon működik a nem-hierarchikus védelmi rendszer?
58. Milyen módon biztosítják az adatokhoz való hozzáférés ellenőrizhetőségét?
59. Mit szabályoznak a hozzáférési jogok?
60. Mit jelent a kétszintű védelmi rendszer?

5. FEJEZET KAPCSOLATOK KEZELÉSE

Az eddigi fejezetekben a processzor és a memória közötti kapcsolatokról volt szó elsősorban. Ugyanakkor az adatok átvitele a processzor és a külvilág között, valamint más eszközök felé is szükséges. A számítógépen belül az adatátvitel legfontosabb útvonalai, kapcsolatai:

- processzor - memória,
- processzor - I/O eszközök, perifériák,
- memória - I/O eszközök,

A processzor és a memória közötti adatátvitel témáját az előző, 3. és 4.fejezetben részleteiben is bemutattuk. A processzor, valamint a memória és az I/O eszközök közötti adatátvitel a témája az 5.fejezetnek.

Az adatátvitel lebonyolításában az eddig megismerteken túlmenően, két eszközcsoport játszik szerepet:

- a számítógép sínrendszere(buszrendszere) és
- a perifékákhoz tartozó I/O regiszterek, 'port'-ok.

Az I/O portok, többnyire, mint memóriabeli tárolóhelyek kezelhetők és a perifériális eszközök bemeneteiként, kimeneteiként működnek.

5.1.SÍNRENDSZEREK

A mikroszámítógép **sínrendszere(buszrendszere)** egy több tucat vezetékből álló vezetékrendszer, amelyen az adatok, vezérlő jelek, eszközcímek meghatározott, specifikációban rögzített módon vihetők át. A vezetékek és a jelek összerendelése, azaz, hogy melyik vezetéken milyen jelet továbbít a rendszer, szintén rögzített.

5.1.1.Sínrendszer lényege

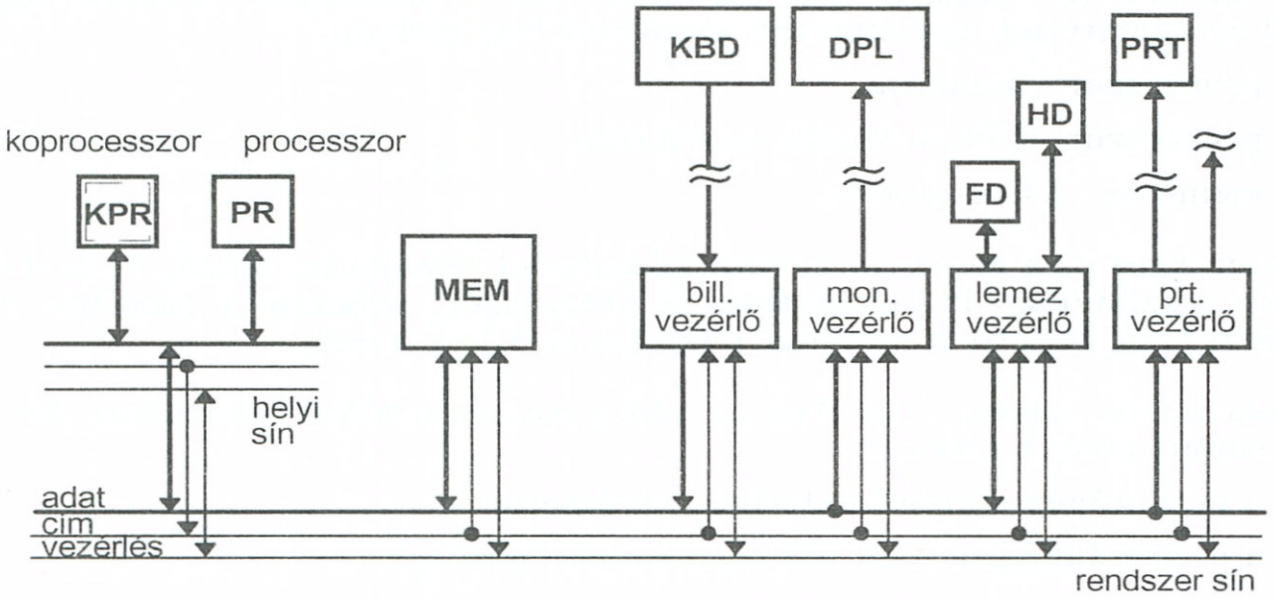
A sínrendszer olyan, mint az országút, összeköti a gép különböző részeit még pedig szabályozott, egységes módon. A gép különböző egységei egymás számára, csak ezen keresztül továbbíthatnak adatokat, vezérlő jeleket. Mivel minden eszköz ugyanarra a sínrendszerre kapcsolódik, az átvitel létrehozásakor

- meg kell oldani az adatátvitelben résztvevő eszközök kijelölését; ez a kijelölés történhet egy cím megadásával, hasonlóan a tároló egyes he-

lyeinek kijelöléséhez, vagy az utasítás alapján létrehozott hardver kapcsolattal,

- meg kell határozni az adatátvitel irányát, és
- meg kell oldani a kapcsolatban résztvevő eszközök működésének összehangolását, szinkronizálását.

A mikroszámítógépek sínrendszerre alapuló szokásos felépítését vázolja fel az 5-1.ábra.



5-1.ábra: *Mikroszámítógépek szokásos felépítése*

A sínrendszer használatának előnye, hogy a szabványosított jelhasználat és vezetékkiosztás miatt, könnyen cserélhetők a csatlakoztatott eszközök, illetve azok vezérlő kártyái és így gyártótól, géptől függetlenné válik azok használata. Az így felépített gépek karbantartása is könnyebbé válik, mivel a hibás részegységek egyszerűbben behatárolhatók. A sínrendszer előírásait teljesítő kártyák pedig bármelyik gépben használhatók.

5.1.2.Sínrendszer felépítése

a.)Struktúra

A mikroszámítógépen belül a sínrendszer nem csak a processzor és más, a processzoron kívüli eszközök összekötésére szolgál, hanem magán a processzoron belül is kialakítottak belső sínrendszert. Ennek megfelelően megkülönböztethetünk:

- belső sínrendszert,
- külső sínrendszert.

A **belső sínrendszer** kialakítását az elérni kívánt teljesítmény szabja meg. Nagyobb teljesítményű processzorok esetében, az átvitelek gyorsítása érdekében, 3-sínes rendszer kialakítása a célszerű, amelynél a címsín mellett, külön adatsín van írásra és olvasásra. Ezzel a közel egyidejű írás és olvasás megoldható. (A belső sínrendszer esetében a vezérlésre szolgáló vezetékek nem szerepelnek a sínrendszer részeként, mivel a vezérlés módjából adódóan az nem különül el önálló rendszerként.) Egyszerűbb megoldást ad a 2-sínes(adat- és címsín) rendszer, amely általánosan elterjedt megoldás a processzorok körében. A közös adat- és címsín használata csak a nagyon egyszerű, célfeladatokra használt processzoroknál alkalmazott.

A sínek számának csökkenésével a processzor alkalmazási hajlékonysága növekszik, de teljesítménye csökken. Tipikus példák az egyes változatokra:

- 3-sínes rendszert alkalmaz: Am2900, Am29300, RISC processzorok,
- 2-sínes rendszert alkalmaz: LSI-11, MC68020, i386/486,
- 1-sínes rendszert alkalmaznak egyes egyszerűbb mikrovezérlők; a mikroprocesszorok szempontjából nincs jelentősége már.

A **külső sínrendszer** az összekapcsolt területek alapján lehet:

- *helyi sín*(local bus), amely a processzorhoz közvetlenül kapcsolódó részt jelenti, tehát a processzor hajtja meg; erre kapcsolódnak azok az eszközök(pl. társprocesszor), amelyek esetében a gyorsaság lényeges;
- *rendszersín*(system bus), amely a processzort köti össze - egy sínmeghajtó közbeiktatásával - a gép egyéb részeivel, elsősorban az I/O eszközökkel;
- *memóriasín*(memory bus), amely nem minden esetben képez önálló részt, de nagyobb rendszernél célszerű leválasztani a rendszersínről a memória területét.

A sínrendszer az egységek összekötését biztosítja, amely tartalmilag három részre osztható:

- *címsín*, amely az eszközök címzését szolgálja, azok címét továbbítja rajta a processzor; szélessége 32(esetleg 16-20-24) bitnek megfelelően ugyanennyi vezeték;
- *adatsín*, amelyen keresztül a továbbítandó adatot küldi, vagy fogadja a processzor; az adatsín szélessége többnyire 32-64 bit, illetve ugyanennyi vezeték;
- *vezérlősín*, amelynek vezetékeit a processzor a vezérlőjelek kiküldésére, vagy azok fogadására használja fel; a vezérlőjelek száma változó, általában 10-15 körül van minimálisan.

A **vezérlőjelek** körét érdemes egy kicsit bővebben is megismerni. Legegyszerűbben a következő csoportosítás adható funkcionális szerepük alapján:

- **adatátvitelt vezérlő jelek**, amelyek között megtalálhatók(a jelölések változhatnak az egyes processzoroknál):

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

- **megszakítást vezérlő jelek**, amelyek között a megszakításkérő és a megszakítást visszaigazoló jeleket tarthatjuk számon(a megszakításokról az 5.2.pontban lesz szó bővebben);
- **sínvezérlő jelek**, amelyek a sín kérését, foglalását és a foglalás visszaigazolását szolgáló jeleket foglalják magukban;
- **egyéb jelek**, amelyek között említhetjük az órajeleket, amelyek a gép ütemezését, szinkronizálását szolgálják, valamint megemlíthetjük még a tápfeszültség és a test(föld)vezetékeket is

A sínrendszerhez szorosan hozzátartozik a helyi sínt a rendszersíntől elválasztó **sínvezérlő**(sínmeghajtó) **egység**(bus interface), amely

- *egyrészt*, a megfelelő nagyságú jeleket biztosítja a hosszabb rendszersín számára,
- *másrészt*, elválasztja az egyes sínrészeket egymástól, amelyek bizonyos mértékig egymástól függetlenül működhetnek; a helyi sín számára lehetséges a gyorsabb működés;
- *harmadrészt*, a sínt vezérlő jeleket szolgáltatja, szabályozza a sínfoglalásokat.

b.)Sín használói(master, slave)

A sínt egyidőben csak egy eszközpár használhatja. A sín használatát valamelyik eszköz kezdeményezi, amelyet **aktív eszköz**nek(master) neveznek, szemben a kapcsolatban résztvevő másik, **passzív eszköz**zel(slave), amely csak fogadja és végrehajtja az aktív eszköztől származó vezérléseket. A mikroszámítógépeknél a sín irányítását megszerző eszköz a processzor, vagy valamelyik közvetlen tárolóhozzáférést(DMA) alkalmazó I/O eszköz lehet.

A sín használatát eredményező folyamatot a passzív eszköz jelzése alapján (megszakítási kérelem) az aktív eszköz, például a processzor kezdeményezi valamilyen feladat elvégzésére. A beérkező igény kétféle módon értékelhető ki:

1. Az egyik esetben minden passzív eszköz önálló megszakítási lehetőséggel rendelkezik, amely alapján a processzor azonnal el tudja dönteni, hogy

melyik eszköz küldte a jelzést és el tudja indítani a sín lefoglalása iránti kérelmét. Ha egy eszköz több megszakítási lehetőséggel rendelkezik, akkor a processzornak előbb szoftver úton, program segítségével ellenőrizni kell a lehetséges okokat, majd ezt követően kezdeményezheti az átviteli kapcsolat létrejöttét, a sín foglalását. Ezt a módszert **nem vektoros sínfoglalás**nak nevezik.

2. A másik esetben a passzív eszköz saját, megszakítást értékelő logikával rendelkezik, amelynek eredményeként, jelzésére a processzor egy elfogadási jelzést küld ki, valamint a 'slave' azonosítóját. A processzor második elfogadási jelének megjelenése után, a passzív eszköz a megszakítás okának megfelelő vektort, címet helyez az adatvonalra, amelyet felhasználva, a processzor elindítja az átviteli kapcsolat kialakítását, a sín foglalását. Ezt az eljárást **vektoros sínfoglalás**nak nevezik.

Sok esetben az adott eszköz egyik esetben aktív szerepet és más helyzetben passzív szerepet is betölthet. A tároló mindig csak passzív szereplő lehet az átvitelek lebonyolításában. A lehetséges szerepeket, párosításokat az alábbi kis táblázat mutatja be:

Aktív szereplő (master)	Passzív szereplő (slave)	Kapcsolat
processzor	tároló	utasítás-, és adatelőkészítés
processzor	I/O eszköz	adatbevitel, és -kivitel
processzor	koprocesszor	lebegőpontos utasítás végrehajtása
I/O eszköz	tároló	közvetlen tárolóhozzáférés(DMA)
koprocesszor	tároló	adatelőkészítés

c.)Sínprotokoll

A sínrendszer általános kapcsolatteremtő funkciója következtében, meg kell felelnie valamilyen elfogadott előírás rendszernek, amely megszabja fizikai, mechanikai és elektromos jellemzőit, a rákapcsolódó eszközökre előírt követelményeket. Az általános használat érdekében szabványosított megoldások alakultak ki.

A sínrendszer működésére vonatkozó, az előbbiekben körülírt tartalmú szabályrendszert **sínprotokoll**nak(bus protocol) nevezik.

Tehát a sínprotokoll megszabja a működési szabályokat, a mechanikus és elektromos jellemzőket és azt, hogy milyen eszközök és hogyan kapcsolódhatnak a sínre.

Az idők során igen sokféle sínrendszert alakítottak ki, amelyek közül természetesen csak néhány az, amelynek a használata széles körben elterjedt. Az ismertebb sínrendszerek néhánya: IBM PC, PC/AT-ISA sín, EISA sín(pl.: i80386/486 mellett), Nubus(Apple Macintosh II.), Unibus (DEC PDP-11), VME bus(MC680x0).

5.1.3.Sínrendszer működése

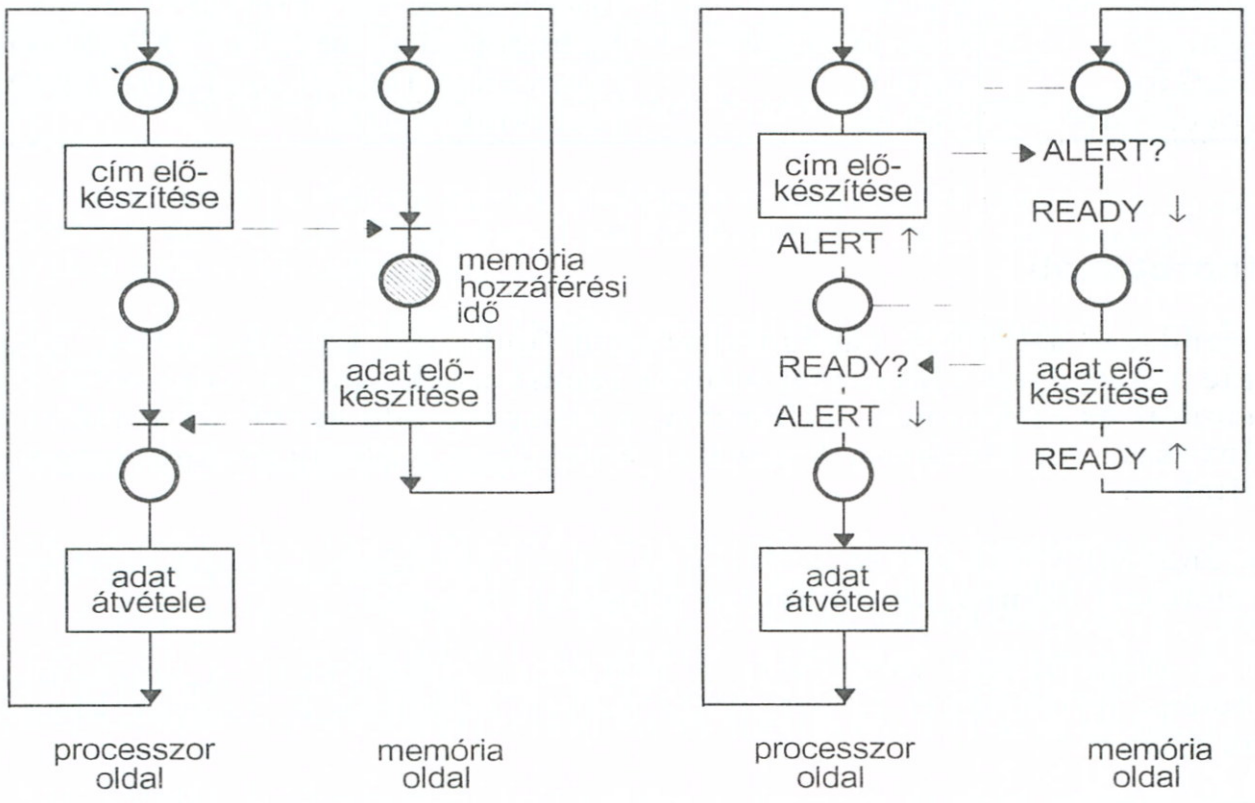
A sínprotokoll kapcsán már említésre került, hogy a sín működése szabályozott módon zajlik és fontos feladata sínrendszernek, hogy az adatátvitelhez az összekapcsolt eszközök működését összehangolja.

a.)Sínvezérlés módja

A sínrendszerek a vezérlés szempontjából két fő csoportra oszthatók:

- **szinkron** ütemezésű sínek, amelyek saját órajellel(ütemezéssel) rendelkeznek és ezek ütemei szabják meg a sínen zajló folyamatok, műveletek időbeli lefutását;
- **aszinkron** ütemezésű sínek, amelyek saját órajellel(ütemezéssel) nem rendelkeznek és a folyamatok és műveletek lefutását, az egymást követő elemi lépések befejezése szabályozza.

Minden síntevékenységhez meghatározott időtartam, ütemszám szükséges. Azt az időtartamot(ütemszámot), amely egy adatáviteli folyamat lefutásához - egy következő tevékenység megindítási lehetőségéig - kell, **sínciklusnak** (bus cycle) nevezzük.



5-2.ábra: *Szinkronizálás megoldásának alapelve*

Mind a szinkron, mind az aszinkron működési mód mellett az összekapcsolt eszközöknek összehangolt módon kell működniük. Az összekapcsolt eszközök szinkronizálási(együttfutási) problémáit az 5-2.ábra segítségével, a **pro-**

cesszor és a tároló közötti adatátvitel esetében mutatjuk be. (A párhuzamos folyamatok működését az ú.n. Petri-gráfok elemeinek felhasználásával, egyszerűsített formában mutatjuk be.)

Az átvitel megadásához meg kell határozni a kiválasztott eszközt, pl.: az S_i (select i-th device) jellel, az átvitel irányát, pl.: az **R/W***(read/write) jellel; emellett a kijelölt eszközt fel kell szólítani az átvitel fogadására(ezt az **ALERT**, illetve a cím sínre helyezését jelző **AS**[AddressStrobe] jel teszi meg), amelynek elfogadását, az eszköz készenlétét a **READY** jellel jelezzük.

Az ábra csomópontjait(köreit) összekötő éleken keresztül húzott rövid egyenesek egy-egy jelátmenet helyét jelzik, illetve az odafutó élek az átmenet bekövetkezéséhez szükséges vezérlőjelekre utalnak. Az egymás mellé helyezett két részábra egyike a szinkronizálás alapproblémájának szöveges megjelenítése, a másik ennek egy kicsit pontosabb, műveletek és vezérlőjelek megjelölésével rendelkező változata.

Az ábrából látható, hogy a memóriaoldal mindaddig várakozik, amíg az átvitelre felszólító jelet(**ALERT**) nem észleli. A processzor oldala viszont ezek után arra vár, hogy a művelet befejeztéről a kész(**READY**) jelzést megkapja. Látható az is az ábrából, hogy az egyes oldalak a vezérlőjel észlelése után, a saját oldaluk által szabályozott jelet ellenkező állapotba helyezik és evvel felfüggesztik a másik folyamat továbbhaladását mindaddig, amíg az előírt feladatot, tevékenységet el nem végezték.

A **kérés-visszaigazolás** előbbiekben ismertetett változata, az eszközök kapcsolatainak kezelésében igen általánosan elterjedt megoldás. Ezt nevezik **hand-shaking**, 'kézfogás'-os technikának.

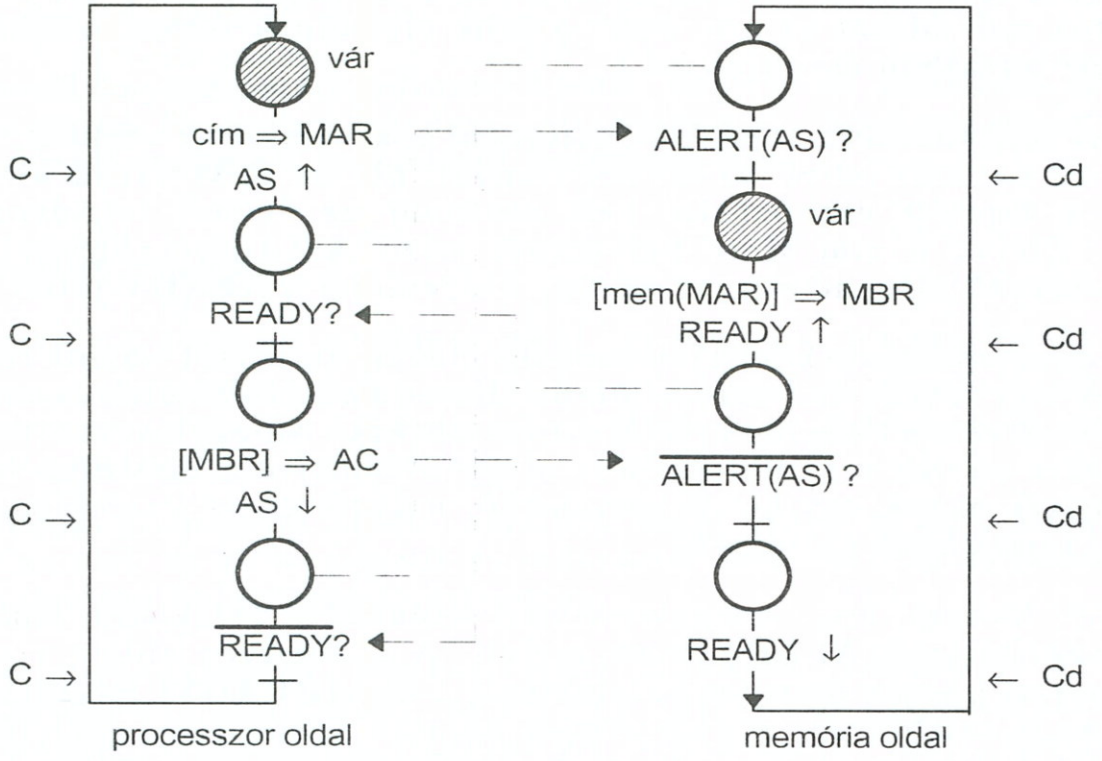
Az 5-2.ábra szerinti megoldásnál azonban problémát okozhat, hogy a processzor-oldal és a memória-oldal folyamata a kérés-visszaigazolás (ALERT-READY jelpár) után 'megszaladhat', azaz egymással nem összehangoltan, szinkronizálatlanul futhat tovább. Ennek megoldására az aszinkron vezérlésnél egy második hand-shaking kapcsolatot vezetnek be, míg a szinkron megoldásnál a 'kész' jelzés időbeli, megfelelő eltolását alkalmazzák.

b.)Aszinkron vezérlés

Aszinkron vezérlésnél a két összekapcsolt eszköz működtetése nem azonos ütemezéssel történik, de ugyanakkor az összekapcsolt eszközök(pl. processzor és memória) saját ütemezéssel rendelkezhetnek. A párhuzamos folyamatok és műveletek lefutását, összehangolását, az egymást követő elemi lépések befejezése szabályozza.

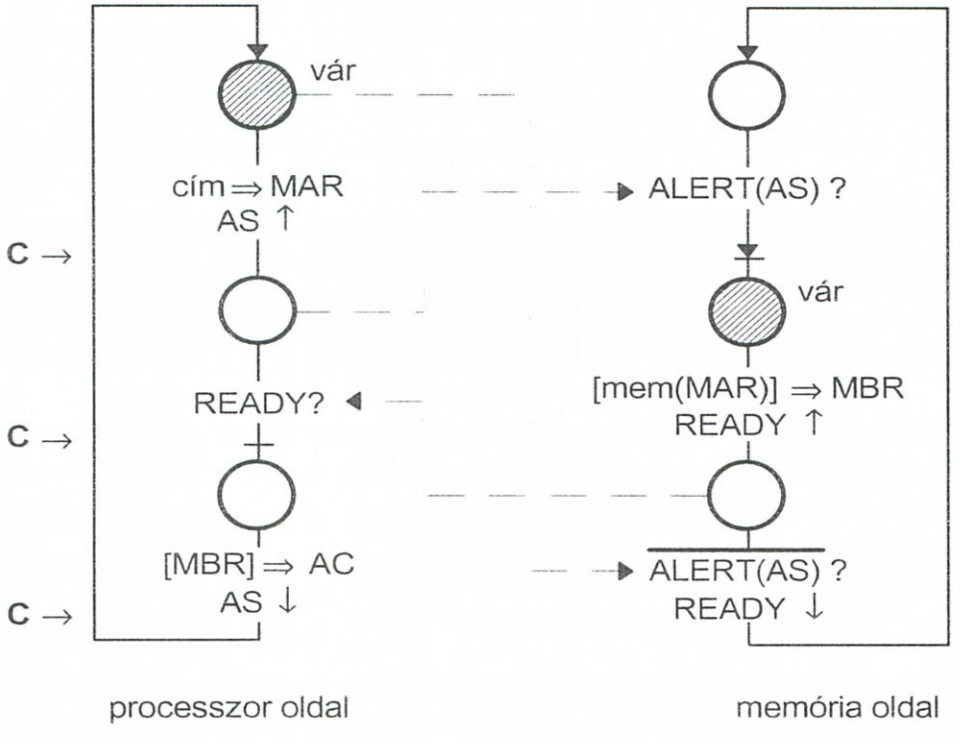
Saját ütemezéssel működő passzív eszköz aszinkron vezérlésének folyamatát az 5-3.ábra mutatja be, a már megismert módon.

A párhuzamos folyamatok összehangolatlanságát a második kérés-visszaigazolás bevezetése szünteti meg, a saját ütemezésen túl. Ezt a kettős kérés-visszaigazolást nevezik **full hand-shaking**-nek.



5-3.ábra: *Aszinkron sínhasználat mindkét oldali, saját ütemezéssel*

Saját ütemezés nélküli megoldást mutat be az 5-4.ábra, amelyet pl. a Motorola MC68020-as processzor mellett használnak.



5-4.ábra: *Aszinkron sínhasználat egyoldali ütemezéssel*

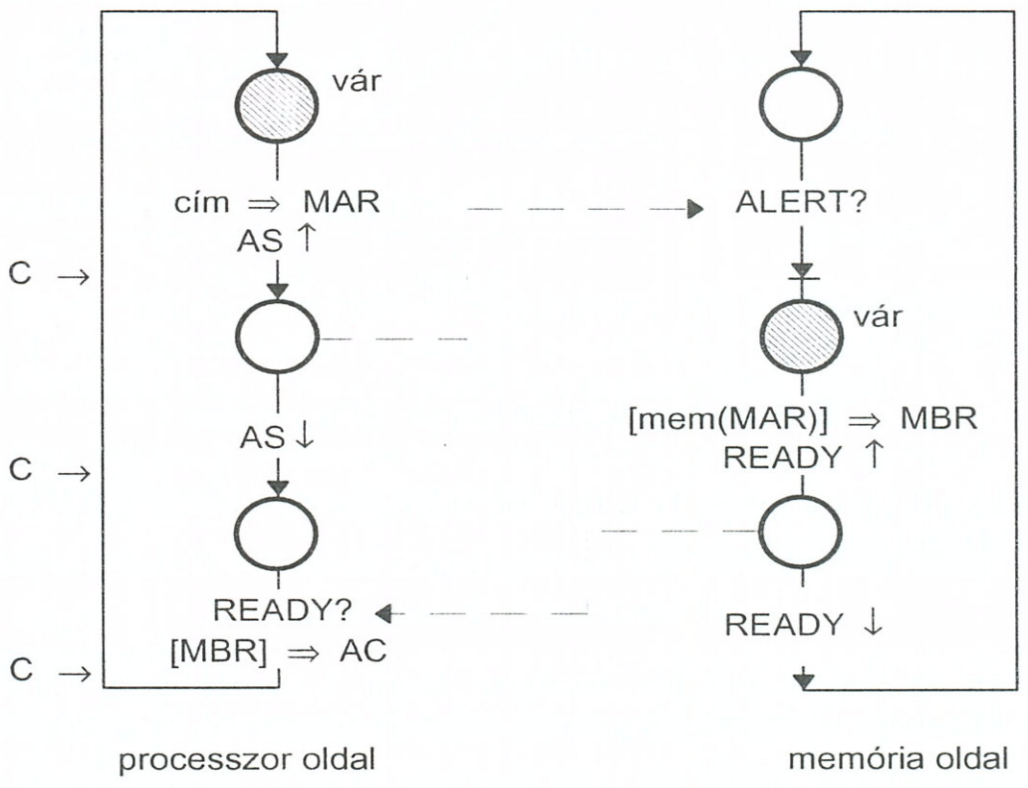
Az aszinkron átviteli kapcsolat létrehozása bonyolultabb megoldást igényel, de különböző típusú eszközök összekapcsolása esetén jobb megoldást ad, mint a szinkron vezérlési változat.

c.)Szinkron vezérlés

Szinkron vezérlésnél a sínre kapcsolt, az átvitelben résztvevő eszközök azonos ütemezés alapján dolgoznak. Az ütemezés történhet a vezérlőjelek azonos(pl. felfutó) élével, de lehetséges a jelek ellenkező(felfutó-lefutó) élével is vezérelni az egyes folyamatokat.

A szinkronizált vezérlés folyamatát mutatja be az 5-5.ábra. A memóriaoldal ellentétes éllel való vezérlésével a tároló oldali ütemek száma csökkenthető. Ilyen megoldás található az i386/486-os processzorok mellett. A sín ütemezése a processzor órajelével(pl.: PCI sínrendszer), vagy annak valamilyen leosztott értékével(pl.: ISA sínrendszer) történik.

A szinkron adatátviteli kapcsolat egyszerűbb működést eredményez, de tervezése sokkal hosszadalmasabb, mivel az egyes fázisok időtartamát szigorúan össze kell hangolni. Használata a hasonló típusú eszközök alkalmazásánál előnyös. A számítógép működésében általában szinkronizált folyamatokat építenek ki.



5-5.ábra: *Szinkron sínhasználat*

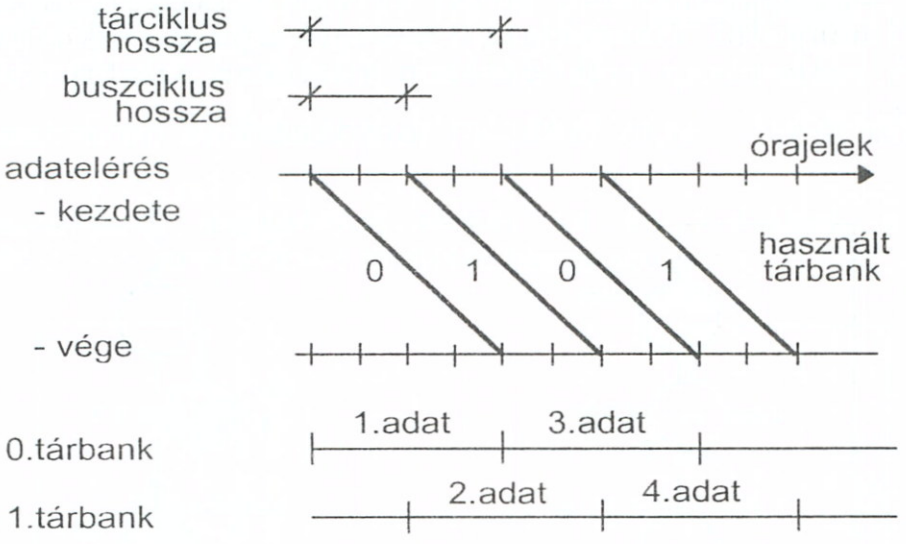
5.1.4.Tárkezelés gyorsítása

A tárolókezelés gyorsításának a már eddig megismert módjain(átlapolt-memória[memory interleaving], átlapolt-címzés[interleaved addressing] használatán) kívül, további lehetőségeket nyújt a sínrendszer működésében rejlő tartalékok kihasználása, mint a sínciklusok átlapolása, vagy a blokkos adatátvitel gyorsítása.

a.)Átlapolódó sínciklus

A sínciklusok átlapolásának lehetőségét az adja, hogy a tárolóciklus időtartama hosszabb, mint a sínciklusé. A sínciklus hossza legalább 2 órajel ütem, míg a memóriaciklus ennél hosszabb.

Ha a memória elérési ideje nagyobb, mint a sínciklus időtartama, akkor átlapolással gyorsítható az átvitel; azonban ennek feltétele, hogy a memória, címzés szempontjából 'bank'-okra bontható legyen.



5-6.ábra: *Sínciklusok átlapolása*

Az átlapolódó ciklusok időbeli lefutását mutatja be az 5-6.ábra. A bemutatott példánál a feltételezés az, hogy

1 tárciklus = 2 sínciklus

és ennek alapján teljes átfedés hozható létre 2 tárolóbank alkalmazásával. Minden két ütemnyi időtartam egy-egy adat kiolvasását és átvitelét jelenti.

Ha a tárolóciklus és a sínciklus aránya más, akkor az átlapolódás mértéke, a sín kihasználtsága és a szükséges tárolóbankok száma változik.

Átlapolt sínciklust alkalmaznak az Intel processzorok; az i286-os, i386/486-os processzoroknál 2 sínciklust, az i860-as RISC processzornál 3 sínciklust lapolnak át.

b.)Blokk-sínciklus(burst cycle)

A blokk-sínciklus arra ad lehetőséget, hogy több adat blokkos átvitelénél(pl. cache-tárak feltöltésénél, visszaírásánál), ha azok a tároló egymást követő helyein találhatók, az ismétlődő, egyszerű tárhozfordulás helyett, egy gyorsabb megoldást használjunk. A blokk-sínciklus esetében nem egyenként történik az adatok címzése, hanem automatikusan a tároló következő helyét veszi a vezérlő. Így ez a megoldás lerövidíti a teljes átviteli időt.

Az ilyen típusú átvitel jelzésére, pl. a BI(burst inhibit) jelet lehet használni.

Ha a 'burst' üzemmód engedélyezett(BI=0), akkor például egy 4 szavas(16 byte-os) blokknál

az 1.szó normál üzemmódú átvitele: min.2 ütem

a 2.-4.szó rövidített elérési idővel: 1-1 ütem.

a teljes blokk-sínciklus= 1normál +3 rövid elérési idő=**5 ütem**.

Ha a blokk üzemmód nem engedélyezett(BI=1), akkor

1.-4.szó normál üzemmóddal: 4x2 ütem = **8 ütem**.

Ilyen burst üzemmódra lehetőség van az i486-os, az MC68030-as, MC68040-es processzoroknál.

5.1.5.Sínfoglalás(bus arbitration)

Az adatátvitelek lebonyolításához egyidőben több aktív eszköz(master) is igényelheti a sín használatát. Ilyenkor valamilyen eljárással el kell dönteni, hogy melyik eszköz kapja meg először a sínhasználat jogát.

A sínhasználat jogának eldöntésére szolgáló folyamatot nevezik sínfoglalásnak, **sín arbitrációnak**(bus arbitration).

A sín használata iránti igények kiszolgálása két módon történhet:

- soros, vagy
- párhuzamos módon.

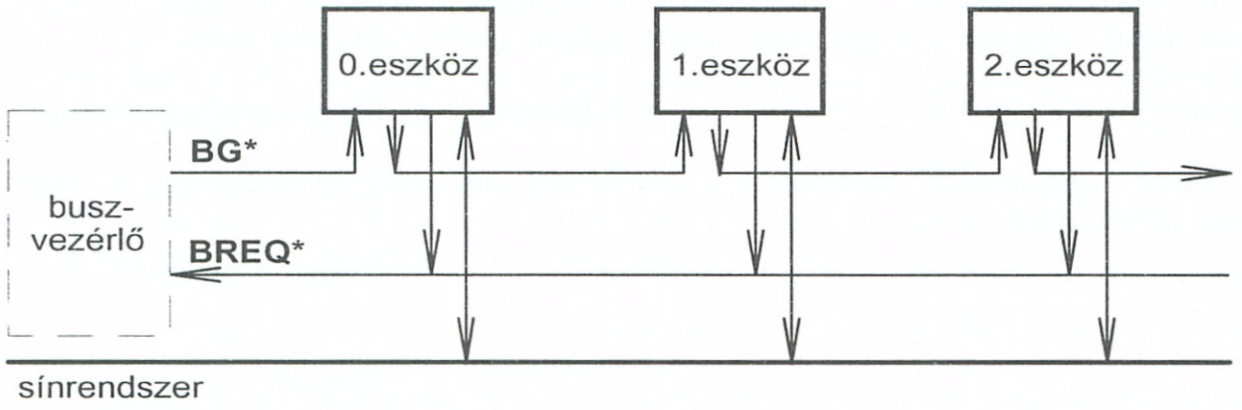
Mindkét esetben a jogosultság megállapítása történhet:

- *centralizált módon*, amely esetben egy központi prioritásvezérlő logika szabja meg a hozzáférés sorrendjét, vagy
- *decentralizált módon*, amely esetben a prioritizáló logika elosztott formában valósul meg, az egyes eszközök vezérlői által.

Soros kiszolgálás(daisy chain)

A soros kiszolgálás alkalmazásakor az eszközök sorba vannak kötve és a lánc mentén az elhelyezkedésük szabja meg, hogy mikor kaphatják meg a sín használatát. Amelyik eszköz a legközelebb van a vezérlőhöz, annak a prioritása a legmagasabb.

A sínfoglalás folyamatát az 5-7.ábra segítségével mutatjuk be. A kérelmező eszközök két jelet használnak, a sínkérést jelző BREQ* (bus request) és a sínengedélyező BG*(bus grant) jelet.



5-7.ábra: *Soros sínfoglalás elve*

A sínt kérő eszközök a BREQ* jel segítségével jelzik igényüket a sín használatára. A beérkező jeleket a vezérlő logika fogadja és a BG* vonalra kiadja az engedélyező jelet, amelyre sorosan fel vannak fűzve az eszközök. Az az eszköz, amelyik a sorban legelöl igényelte a sínt, nem engedi tovább a BG* jelet és így a maga részére lefoglalja a sínt.

A prioritási rendszer *lehet több szintű is*, azaz több BREQ* és BG* vonal van és a vezérlő a beérkező legmagasabb prioritású kérelemnek megfelelő prioritású BG* vonalon ad ki jelet, amelyet a kapcsolódó, adott prioritású sínkérelmező eszközök közül a sorrendben első eszköz nem enged tovább és lefoglalja a sínt.

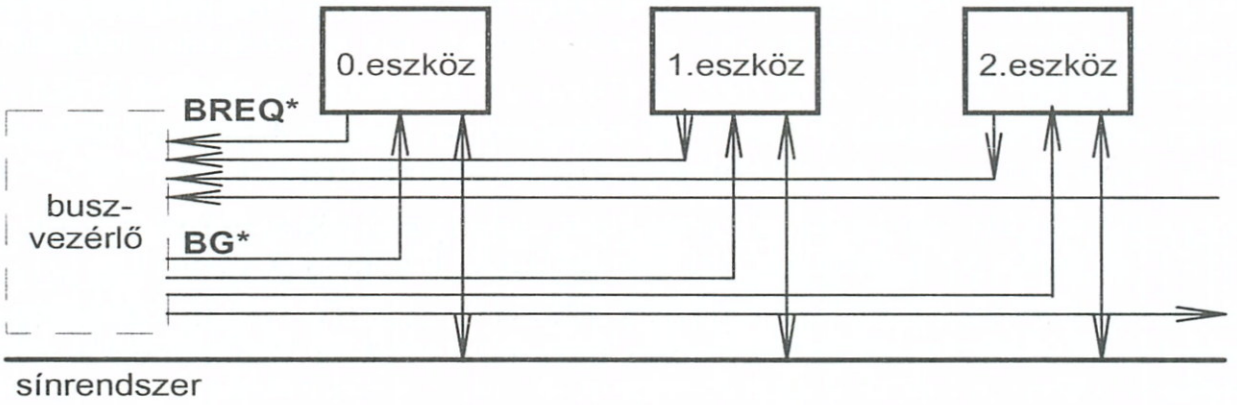
A sínkérelmek kezelésének gyorsítására be lehet vezetni egy harmadik vezérlő jelet is, a visszaigazoló BAKN(bus acknowledge) jelet, amelyen a sínfoglalás megtörténtét jelzi a vezérlő és evvel lehetővé teszi az újabb sínkérelmek fogadását.

A soros rendszerben a processzor szokásosan a legalacsonyabb prioritású azért, hogy minden eszköz megkapja az eszközt, ha igényli.

A soros kiszolgálás egyik változata a **lekérdezéses(polling) rendszer**, amelynél a vezérlő logika sorra lekérdezi az eszközöket a sínfoglalás iránti igényükről. Itt a lekérdezés sorrendje adja a soros láncot.

Párhuzamos kiszolgálás

Párhuzamos sínkiszolgálás alkalmazásakor minden eszköz önálló sínkérő (BREQ*) és sínengedélyező(BG*) vezetékkel rendelkezik. A beérkező igényeket a vezérlő logika sorolja, dekódolja és a legmagasabb prioritású eszköz számára engedélyezi a sín használatát. Ezt mutatja be az 5-8.ábra vázlata.



5-8.ábra: *Párhuzamos sínfoglalás elve*

Prioritás meghatározása

Az eszközök prioritásának szabályozására mind a soros, mind a párhuzamos vezérlésnél többféle módszert alkalmaznak.

A **soros kiszolgálás**nál alkalmazott módszerek:

- *soros sínhasználat engedélyezés*(daisy chained bus grant), amelynél a sínkérelem beérkezése után az engedélyező jel halad végig a láncon és a sorrendben legelől álló kérelmező letiltja a jel továbbhaladását és igénybe veszi a sínt; ez a módszer a legáltalánosabban használt;
- *soros sínkérelem engedélyezés*(daisy chained bus request), amely esetnél a BREQ* vonal halad végig az eszközökön; a sínt kérő eszközök kiadják a sínkérelmező jelet az alacsonyabb prioritás irányába, amelyek elérik a jelenlegi sínhasználó eszközt, ha az alacsonyabb prioritású; ez az eszköz ekkor egy közös engedélyező vonalra kiadja a sínengedélyező (BG*) jelet, amelyet minden eszköz érzékel, de csak az az eszköz veheti igénybe a sínt, amelynél magasabb prioritású egység nem kérte a sínt (tehát, amelynek a BREQ* bemenetén nincs jel);
- *soros engedélyezés*(daisy chained enable), amely megoldásnál közös BREQ* és BG* vezeték van és egy engedélyező vonal halad végig az eszközökön; az az eszköz, amelyik igénybe akarja venni a sínt, letiltja kimenetén a további, alacsonyabb prioritású eszközök számára a sín kérésének lehetőségét.

A **párhuzamos kiszolgálás**nál alkalmazott eljárások:

- *egyszerű körbejáró*(simple rotating), amelynél minden sínengedélyezés után a prioritás sorrendje változik, minden eszköz eggyel alacsonyabb prioritást kap és az eddigi legalacsonyabb prioritású eszköz kerül a legmagasabb szintre;
- *elfogadástól függő körbejáró*(acceptance dependent rotating), amelynél a sínkérelem kiszolgálása után a prioritás sorrendje változik, minden eszköz eggyel magasabb prioritást kap és az eddigi legmagasabb prioritású eszköz a legalacsonyabb szintre kerül; ezt a módszert használják a legáltalánosabban;
- *véletlenszerű*(random), amelynél a prioritási sorrend minden kiszolgálás után újra kiosztásra kerül, még pedig véletlenszerűen;
- *egyenlő*(equal) prioritás alkalmazása, amelynél több igény esetén, a vezérlő egyenlő valószínűséggel szolgálja ki bármelyik igényt;
- *legkevésbé használt eszköz*(LRU=least recently used) kiszolgálása, amelynél a sínt legrégebben nem használó eszköz kapja meg a sínt először.

5.2.MEGSZAKÍTÁSI RENDSZER

A számítógépek munkájának összehangolása, mint többszereplős, bonyolult berendezésnek, igen fontos probléma. Ennek megoldásában segít a gépek **megszakítási rendszere**(interrupt system).

5.2.1.Megszakítás fogalma, keletkezése

A számítógépi folyamatok közben igen gyakran következnek be olyan események, amelyek a feldolgozás szempontjából váratlannak tekintendők. A kérdés az, hogy hogyan lehet ezeket a váratlan eseményeket úgy kezelni, hogy az a feldolgozás egészét a lehető legkevésbé zavarja, vagy akadályozza.

A folyamatok közben keletkező eseményeket a következőképpen csoportosíthatjuk:

- vannak a program futása szempontjából, jól meghatározható helyen, időpontban várható események, ezeket **szinkron eseményeknek** nevezik, mivel minden futáskor ugyanott következnek be; pl. valamilyen művelet végrehajtása közben(túlcsordulás, nullával való osztás, lapváltás iránti igény);
- vannak a program futása során várható, de időpontjuk szempontjából ismeretlen, váratlan események; ezeket **aszinkron, várható eseményeknek** nevezik; ilyenek pl. az adatok beolvasása/kiíratása, DMA vezérelte adatátvitel;
- vannak olyan események, amelyek váratlanok és időpontjuk ebből származóan nem meghatározható, ismeretlen; ezek az események az **aszink-**

ron, váratlan események; ilyen esemény valamely hardver hiba, áramkimaradás bekövetkezése.

Ezeknek az eseményeknek a kiváltója

- lehet maga a **program**, például valamilyen futás közbeni hiba(nullával osztás, túlcsordulás, stb.) esetében, amelynek kezelésére egy kiszolgáló eljárást(trap, exception kezelés) indít el a processzor; ezek **szinkron** események, mert a program azonos helyén következnek be minden futtatáskor;
- lehet a **hardver**, például valamelyik periféria, amely az adatátvitel lebonyolításának idejére, a program futásának ideiglenes felfüggesztését kezdeményezi; ezek **aszinkron** események.

Az ilyen események feldolgozására, kezelésére szolgálnak a megszakítások, a számítógép **megszakítási rendszere**. A **megszakítási kérelem** egy jelzés a processzor számára valamely esemény bekövetkeztéről, amely valamilyen kiszolgáló folyamatot indíthat el egy későbbi időpontban. Ez a későbbi időpont a megszakítás időpontja. A **megszakítás** a futó folyamat felfüggesztése a megszakítási kérelem hatására, annak kiértékelésére, kiszolgálására. A megszakítási kérelem kiszolgálására egy hardver-szoftver együttes szolgál, amely együttesen végzi el a kérelem kiértékelését és ennek eredményeképpen a szükséges tevékenységeket. Ez, a futó folyamatba a megszakítási kérelem hatására beiktatott tevékenységsor a **megszakítási kérelem kiszolgálása**.

A megszakítási események vizsgálatakor, különbséget kell tenni a többnyire külső eredetű megszakítások(**interrupt**) és az utasítások szabályszerű végrehajtását megállító kivételek(**exception**) között.

Az utasításvégrehajtástól független, külső események által okozott, a program végrehajtását akadályozó eseményeket **megszakítások**nak(interrupt-oknak) nevezik. Ezekben az esetekben, a végrehajtás alatt álló utasítást a processzor szabályszerűen befejezi, majd a megszakítás kiszolgálása után, a soronkövetkező utasítás feldolgozásával folytatódik a program végrehajtása. Ezzel szemben az utasítások szabályszerű végrehajtását akadályozó megszakítási eseményeket(mint például túlcsordulás, lapváltási igény, stb.) **kivételeknek**(exception-öknek) nevezik. A kivételek esetében, a kiváltó esemény kiszolgálása után, a processzor ismételten megkisérli a megszakított utasítás végrehajtását.

Egyes eszközök esetében a megszakítás lehetősége engedélyezhető (enable), vagy tiltható(disable). Az engedélyezés, vagy a tiltás egy regiszter bitjeinek a beállításával történik és ezt nevezik **maszkolásnak**. A megszakítási kérelmek között vannak maszkolható és nem maszkolható kérelmek. Ez azt jelenti, hogy a nem maszkolható kérelmek kiszolgálása nem tiltható le, azok mindig érvényre jutnak.

A megszakítási kérelmeknek két forrása lehet: szoftver és hardver. A *szoftver megszakítási kérelmek*, tehát amelyek programból lettek kezdeményezve, nem maszkolhatóak. A *hardver megszakítási kérelmek* többsége maszkolható, de van nem maszkolható(NMI=Non Maskable Interrupt) kérelem is. Ilyen

nem maszkolható hardver megszakítási kérelem valamilyen súlyos hardver hiba esetén következik be.

5.2.2.Megszakítások, kivételek kiszolgálása

A megszakítások kiszolgálásakor több olyan kérdés van, amelyet a rendszernek meg kell oldania. Ilyen probléma

- a megszakítási kérelem keletkezési helyének a megállapítása, azaz, hogy *melyik eszköz kezdeményezte* a megszakítást;
- a feladattól, a futó programtól függő módon, a megszakítási lehetőségek szabályozása, egyes eszközök ideiglenes kizárása a megszakítást kérő egységek köréből, azaz a *megszakítások maszkolása*;
- több, egyidőben bekövetkező megszakítási kérelem kiszolgálási sorrendjének a meghatározása, azaz a *prioritások szabályozása*;
- *többszörös megszakításkiszolgálás* megoldása, azaz a kiszolgáló folyamat közben bekövetkező újabb megszakítási kérelmek kezelésének megoldása.

A következő pontokban ezekre a kérdésekre, problémákra adunk választ, ismertetve az alkalmazott módszereket, lehetőségeket.

a.)A megszakítási kérelem keletkezési helye

A megszakítási kérelem keletkezési helyének megállapítására két fő rendszer alkalmazható:

- a szoftver és
- az alapvetően hardver úton történő megállapítás lehetősége.

Szoftver módszer

A kérelem helyének megállapítására csak egyszerűbb esetekben alkalmazzák a **szoftver módszer**t. Ebben az esetben egy program, többnyire az operációs rendszer részét képező rutin, bizonyos időközönként sorra megvizsgálja a megszakítási kérelem szempontjából szóbajöhető eszközök állapotjelzőjét és amelyiknél a megszakítás iránti igényét az eszköz jelezte, ott elindítja az aktuális eszközhöz tartozó kiszolgáló programot. Ezt az eljárást **lekérdezéses megszakításkezelésnek**(polling interrupt) nevezik.

Hardver módszerek

A hardver módszerek esetében egy megszakításvezérlő szabályozza program segítségével, vagy anélkül a beérkező megszakítási kérelmek kiszolgálását. A megszakítási kérelem(INT jel) vezérlő általi elfogadását egy visszaigazolás (IACK jel) követi. A mikroszámítógépek megszakítási rendszerei, megszakítási vezérlői egy, vagy több megszakítási vezetékkel rendelkeznek. A kiszolgálás módját az aktuális rendszer nagy mértékben meghatározza.

Egy megszakítási vonal esetén, a keletkezés helyének meghatározása történhet *szoftver úton*, lekérdezéses módszerrel(polling), amely esetben a beérkező INT jel egy kiszolgáló rutin indít el a processzor segítségével, amely sorra végigvizsgálja az eszközöket ugyanúgy, mint a már ismertetett szoftver módszer esetében. A kérelem helyének *hardver úton* történő meghatározása a visszaigazoló IACK vezeték segítségével, sorosan(daisy chain) történik, oly módon, hogy a visszaigazoló jel a kiszolgálást kérő eszköztől már nem halad tovább, hasonlóan a sínkérelmek kiszolgálásához. Ez pedig elindítja a kiszolgáló rutint.

Több megszakítási vonal esetén, amikor minden eszköz saját megszakítást kérő vezetékkel rendelkezik, a kérelem helye egyértelműen megállapítható és a hozzá tartozó kiszolgáló rutin elindítható.

A **vektoros módszer alkalmazása** a megszakítási rendszerek legáltalánosabban használt formája. Ennél a változatnál, a megszakítást kérő eszköz valamilyen módon a kiszolgáló rutin kezdőcímét határozza meg a megszakítási vezérlő és a processzor számára. A következő formákat alkalmazzák:

- a megszakítást kérő eszköz az őt kiszolgáló rutint elindító, ú.n. hívó utasítást(pl.: CALL INT_RUT) teljes egészében átadja processzornak végrehajtásra és evvel a kiszolgáló rutinnak az elindítására;
- a megszakítást kérő eszköz annak a tárolóhelynek a címét(pl.: ADR) adja át a processzornak, amelyben a kiszolgáló rutint hívó utasítás (CALL INT_RUT) található, azaz [ADR] = CALL INT_RUT; ennek a címnek(ADR) a tartalmát a processzor, mint utasítást feldolgozza és evvel elindítja a kiszolgáló rutint;
- a megszakítást kérő eszköz az őt kiszolgáló rutinnak a kezdőcímét (INT_RUT) adja át a processzornak, amely azt mint a soronkövetkező utasítás címét betölti a PC-be és onnét folytatja az utasítások feldolgozását;
- a megszakítást kérő eszköz egy sorszámot ad át a processzornak, amely sorszám a megszakításokat kiszolgáló rutinok kezdőcímeit tartalmazó táblázatban, a **megszakítási vektortáblában,** kijelöli az adott eszközt kiszolgáló rutin kezdőcímét; tehát a sorszám, mint index működik a táblázathoz; ez a módszer, a vektoros megszakításkiszolgálás(vector interrupt) a leggyakrabban alkalmazott eljárása;
- a megszakítást kérő eszköz az előző módszernek megfelelően egy sorszámot ad át a processzornak, amely a vektortáblázatban kijelöli a kiszolgáló rutin kezdőcímét, a különbség csak annyi, hogy ebben az esetben a vektortáblát a processzor tárolja belső táblázatában; ezt a módszert nevezik autovektoros(autovector interrupt) eljárásnak.

Nagyon gyakran a hardver megszakítási vonalak(INT jelek) fixen összekapcsoltak a vektortáblázat meghatározott bejegyzéseivel. Így a hardveres és szoftveres kezdeményezésű megszakítási kérelmek azonos módon kezelhetők. Ezt a rendszert alkalmazzák az IBM-PC kompatibilis gépek is.

Mint említettük, a mikroprocesszorok esetében, a megszakítási vektortábla használata a leggyakoribb forma. Az Intel processzorok esetében, a megszakítási vektorsorszám vagy a vektortábla egy 4 byte-os sorát jelöli ki, amely

egy szegmenscímet és azon belüli eltolást (relatív címet) tartalmaz, vagy védett üzemmódban, egy táblázatnak(IDT=Interrupt Descriptor Table) egy sorára mutat. A táblázatban található deszkriptor adja meg a kiszolgáló rutinnak a helyét a tárban.

b.)A megszakítások kiszolgálásának lépései

A megszakítási kérelem helyének és a kiszolgáló rutin kezdőcímének megállapítása után, a processzor elindíthatja magát a kiszolgáló eljárást. A megszakítások kiszolgálásának folyamata az alábbi lépésekből tehető össze:

hardver által:

- az eszközvezérlő beállítja az INT vonalat, jelezve evvel a megszakítási kérelmét,
- a processzor visszaigazolja(IACK) a megszakítási kérelem elfogadását,
- az eszközvezérlő az adatvonalra helyezi a megszakítási vektor sorszámát,
- a processzor tárolja a megszakítási vektor sorszámát,
- a processzor elmenti az utasításszámláló(PC)- és az állapot- (PSW) regiszterek tartalmát a veremtárolóba,
- a vektortáblából a sorszám alapján kikeresi a kiszolgáló rutin kezdőcímét és azt betölti az utasításszámláló regiszterbe; majd elkezdi a kiszolgáló rutin végrehajtását;

szoftver által:

- a megszakított feldolgozás részeredményeit elmenti a regiszterekből a verembe, vagy a tárolóba,
- ha a megszakítást kiszolgáló rutin több eszközhöz is tartozhat, akkor a megszakítást kérő eszköz azonosítása egy külön tárolóhely felhasználásával, amely az eszközazonosítót tárolja,
- a kiszolgáláshoz szükséges egyéb paraméterek összegyűjtése,
- a megszakítást okozó esemény kezelése, megszüntetése,
- a megszakítás kiszolgálása befejeztének jelzése,
- a felfüggesztett feldolgozás adatainak a visszatöltése a regiszterekbe,
- a kiszolgáló rutin befejezése, visszatérés a feldolgozó programhoz;

hardver által:

- az elmentett PSW és PC visszatöltése, a feldolgozás folytatása.

5.2.3.Megszakítások, kivételek sorolása, prioritás

a.)Kiszolgálás sorrendje

Az egyszerre bekövetkező megszakítási kérelmek feldolgozása, azok sorolása, többféle megoldással történhet.

A legegyszerűbb lehetőség a **megszakítások egyenkénti kiszolgálása**, azok beérkezési sorrendjében. A megszakítást kezdeményező eszközök köre szabályozható a megszakítási kérelmek maszkolásával. Többszörös megszakításra nem lévén lehetőség, azért, hogy annak bekövetkezését megakadályozza, a kiszolgáló program az első lépésében letiltja az újabb megszakítási kérelmek kiszolgálását. Ennek a módszernek nagy hátránya, hogy a további megszakítások letiltásával, a közben beérkező, esetleg el nem halasztható megszakítási kérelmek elveszhetnek és így adatok is elveszhetnek.

A megszakítási kérelmeket általában **prioritási elv felhasználásával** szolgálják ki.

- A kiszolgálás sorrendjének meghatározása történhet **szofver úton**, a kiszolgáló program segítségével, amikor is a programbani vizsgálati sorrend meghatározza a kiszolgálás sorrendjét és így az eszközök prioritását.

- A szoftver módszernél jobb a **megszakítási vezérlő alkalmazása**, amely hardver és szoftver együttes alkalmazását jelenti. A megszakítási kérelmek kiértékelése, sorrendjüknek a megállapítása történhet megszakítási vezérlővel(centralizált módon) és történhet decentralizált módon, az egyes eszközök felhasználásával. Ez utóbbi esetben, a visszaigazolás sorosan halad végig az eszközökön(daisy chain), evvel meghatározva azok prioritását is egyúttal. A kiválasztott eszköz a processzornak visszaadja a kiszolgáló rutinhoz tartozó vektorsorszámot, amely ennek alapján el tudja indítani azt.

b.)Többszörös megszakítás

A megszakítási rutin feldolgozása szempontjából a megszakítási rendszer lehet egy-, vagy többszintű.

Az **egyszintű megszakítási rendszer**ben nincs lehetőség a kiszolgáló rutin felfüggesztésére egy újabb megszakítási kérelem által.

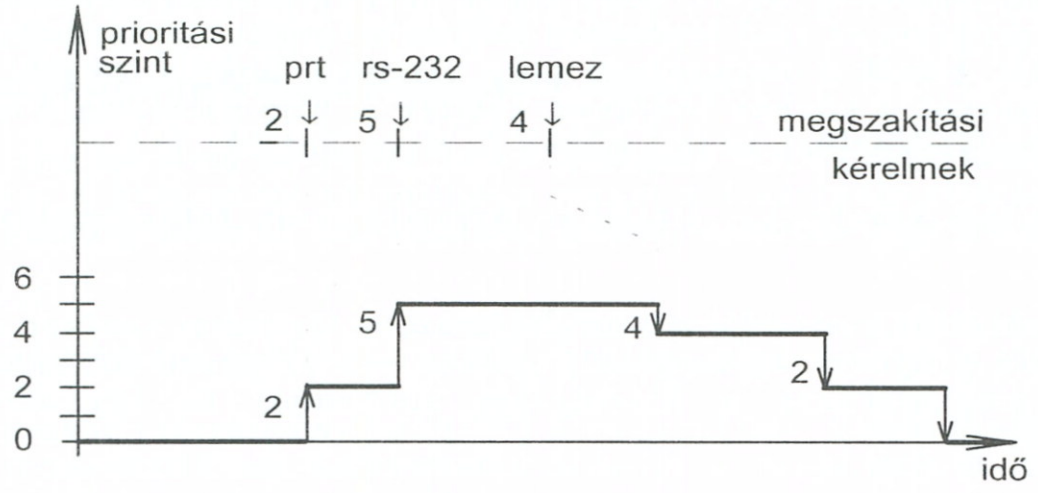
A **többszintű rendszerek**ben a megszakítást kiszolgáló rutin is megszakítható, de csak bizonyos szabályok betartásával. Ezek a módszerek az alábbiak lehetnek:

- a kiszolgáló rutin a vele egyező, vagy nála alacsonyabb prioritású megszakítási kérelmeket letiltja, így azok semmilyen körülmények között nem szakíthatják meg a kiszolgálási folyamatot;

- a kiszolgáló rutin a folyamat kezdetekor ideiglenesen alacsonyabb prioritási szintre sorolja magát, így a vele egyező, vagy nála magasabb prioritású kérelmek megszakíthatják a kiszolgálás folyamatát;

- a kiszolgáló rutin ideiglenesen új prioritásokat rendel az egyes eszközökhöz és így a kiszolgálás alatt más prioritási rend érvényesül, mint egyébként.

A többszintű megszakítás első változat szerinti kezelésére mutat példát az 5-9.ábra.



5-9.ábra: *Többszörös megszakítás kezelése*

A 2-tes prioritási szintű nyomtatási kérelmet megszakíthatja a magasabb prioritású soros vonali kérelem(a beérkező adatok különben elvesznének), amelynek befejeztekor, előbb a közben beérkezett 4-es szintű megszakítási kérelmet szolgálja ki a processzor, majd ezt követően tér vissza az eredeti nyomtatási megszakítás feldolgozásához.

5.2.4.Megvalósítási példák

A megszakításkezelés megoldásának két változatát ismertetjük röviden az alábbiakban.

Az **Intel i8086/8088, i286/386/486**-os processzorok alaphelyzetben kétszintű megszakítási rendszerrel rendelkeznek: egy maszkolható (INT) és egy nem maszkolható(NMI) megszakítási vonallal. Mivel ez a két megszakítási lehetőség a gyakorlati felhasználásban nem elegendő, ezért külön megszakításvezérlőt(i8259A) használnak, amely 8(illetve két darab vezérlő 'kaszkád'-ba kötése esetén 15) megszakítási kérelmet tud feldolgozni. A megszakítási bemenetek sorrendje egyúttal prioritási sorrendet is jelent.

Ha a megszakítási vonalak száma kevés, akkor több eszköz is kapcsolható egy-egy vonalra, de ekkor a kiszolgáló rutinnak kell meghatároznia a megszakítási kérelem keletkezési helyét.

A processzorok valós(real) üzemmódban, a 00000h-ás címtől kezdődően elhelyezkedő vektortábla, míg védett(protected) üzemmódban a megszakítási rutinok deszkriptorait tartalmazó megszakítási deszkriptor tábla(IDT) alapján dolgozzák fel a megszakítási kérelmeket.

A **Motorola MC68000, MC68020/68030/68040** processzorok belső megszakításvezérlővel működnek, amely segítségével 8-szintű prioritási rendszert lehet kezelni. A processzorok 3 megszakítási bemenettel rendelkeznek, amelyek együttes kombinációja, mint bináris szám adja meg a prioritás szintjét. A legmagasabb szint a 7-es, azaz amikor minden bemeneten 1-es szint van; ez felel meg a nem maszkolható megszakítási kérelemnek. Az MC68020-as processzortól kezdve, a megszakítási lehetőségek között van az ú.n. 'autovector interrupt' használata, amely esetben a kapcsolódó bemenet aktivizálása esetén a processzor és nem a kérő eszköz szolgáltatja a végrehajtó rutin vektorát.

Az aktuális prioritási szintet a processzor állapotregiszterének részét képező megszakítási maszk adja meg; ha a beérkező megszakítási kérelem nagyobb ennél, akkor a megszakítást kiszolgáló rutin végrehajtásra kerül. A megszakítási rutin címét a vektortábla kezdőcímének('vector base register' tartalmának) és a megszakítást kérő eszköztől beérkező vektorsorszámnak az összege jelöli ki a táblázatban.

5.3.ADATBEVITEL/KIVITEL

A processzornak a leggyakrabban a memóriával kell kapcsolatot teremtenie, de ugyanilyen fontos a gép számára a perifériákkal lebonyolított adatforgalom is. A legnagyobb gondot ezekben az esetekben az eszközök(processzor, periféria) igen eltérő sebessége okozza, de problémát jelent az eszközök változatossága is és az ebből eredő egyéni kezelési, megoldási módok. Az eszközök közötti eltéréseket csökkenti a közös sín használata, illetve az eszközt a sínhez illesztő vezérlő egység. A bemeneti/kimeneti eszközök ilyen módon történő sínre kapcsolódását mutatja be, a fejezet elején, az 5-1.ábra.

5.3.1.Eszközök elérése, átviteli módok

a.)Eszközök kezelése, I/O portok

A perifériális eszközök kapcsolatát a processzorral, az eszközvezérlőkben található regiszterek segítségével oldják meg. Minden adatforgalom, parancskiküldés, illetve állapotlekérdezés ezeken keresztül valósul meg.

Azokat a regisztereket, amelyeken keresztül a periféria és a processzor között a kapcsolat létrehozható, **I/O port**-oknak nevezik. Az I/O portokat a következő típusú regiszterek alkotják többnyire:

- **parancs(command)regiszter**, amely írható regiszter és egyes bitjei az eszközvezérlőnek szóló előírásokat jelentenek;
- **állapot(status)regiszter**, amely olvasható regiszter és az eszközvezérlő segítségével a periféria állapotáról ad információt a processzor számára;

az állapotregiszter minden bitje más és más állapotinformációt szolgáltat;

- **adatkimenet(data output) regiszter**, amely a kiírandó adatot fogadja be átmenetileg;
- **adatbemenet(data input) regiszter**, amely a beolvasott adatot tárolja ideiglenesen.

Általában szokásos ugyanazon regisztereket közös parancs- és állapotregiszterként, illetve adatbemenet- és kimenet regiszterként használni. Erre lehetőséget ad az, hogy a parancsot írni, az állapotot olvasni kell csak; ugyancsak az adatírás és olvasás is megoldható ugyanazon regiszteren keresztül. Bonyolultabb eszközvezérlők esetében(pl. videokártyák), az I/O portok száma, a megoldandó feladatok nagy száma miatt, lényegesen nagyobb számú is lehet.

Az adatátvitel folyamata, lépései, például egy karakter kinyomtatásakor, az alábbiak lehetnek:

- a processzor ellenőrzi a nyomtató állapotregiszterét, illetve annak foglaltságot jelző bitjét; ha ez foglaltságot jelöl, akkor várakozik a felszabadulásra;
- a processzor a kiíratandó karaktert az adatkimenet regiszterbe tölti;
- a processzor a parancsregiszterbe betölti a karakterkiírás parancsát;
- a nyomtatóvezérlő értelmezi a parancsot, foglalt jelzést állít be az állapotregiszterben;
- a vezérlő az adatkimenet regiszterben lévő karaktert továbbítja a nyomtatáshoz;
- a kiírás befejezte után a foglaltság jelzés megszüntetése.

A processzor számára az I/O portok elérésére két lehetőség van:

- *közvetlen I/O*(input/output) *utasításokkal*, amelyek segítségével az eszközvezérlő regiszterébe közvetlenül írhatunk, vagy onnét olvashatunk; vagy
- *tárolóhoz rendelt módon*(memory mapped addressing), azaz valamilyen tárolócímen keresztül, ugyanúgy, mint ahogy bármelyik közönséges tárolóhelyet is el lehet érni. A memóriacímeknek ez a halmaza lehet része a tényleges memóriacímeknek, de lehet a perifériáknak önálló címtartománya is, mint például az Intel processzorok esetében.

b.)Átviteli módok

Az adatátvitel módja a processzor és a különböző I/O eszközök között kétféle lehet:

- **párhuzamos**, azaz az adatszó minden bitje egyszerre kerül átvitelre, vagy
- **soros**, amelynél az adatbitek időben egymás után lesznek átvíve.

A perifériális eszköz kezelése is többféle lehet a teljesítményigényeknek, gépstruktúrának megfelelően. Ennek illeszkedően a következő módszereket alkalmazzák az I/O eszközök kezelésére.

- *feltétel nélküli, közvetlen(direkt) átvitel*, amelyet egyszerűbb esetekben alkalmaznak, mint például kijelzők beállítása, egyszerű érzékelők leolvasása, stb.; tehát ahol mindenféle feltételtől függetlenül kell az adatátvitelt végrehajtani és ahol ellenőrzésre sem előtte, sem utána nincs szükség;
- *feltételes átvitel*, amelynél valamilyen feltétel teljesülésétől tehető függővé az átvitel lebonyolítása; a feltétel lehet valamilyen jelzőbit(flag) előírt értéke, pl. az eszköz foglaltságának vizsgálata; a kijelölt feltétel teljesülésének ellenőrzésére a kérés-visszaigazolás(hand shaking) módszere, vagy megszakítási kérelem fogadása használható;
- *közvetlen tárolóhozzférés*(DMA=Direct Memory Access) alkalmazása, amelynél az adatátvitel, a memória és az I/O eszköz között, a processzortól függetlenül, a DMA vezérlő hatálya alatt történik;
- *önálló, programozott vezérlővel rendelkező adatátviteli csatorna* használata, amelyet a nagygépeknél alkalmaznak; ez esetben a csatorna a saját vezérlőjének az irányítása alatt, a processzortól függetlenül végzi az adatátvitelt a periféria és a memória között;
- *I/O processzor és csatorna alkalmazása*, amely az előző változat továbbfejlesztése egy saját processzor alkalmazásával.

Ezek közül az átviteli módszerek közül a mikroszámítógépek által a leggyakrabban használtak:

- programozott(feltételes, feltétel nélküli) I/O átvitel,
- megszakításos I/O átvitel,
- közvetlen tárolóhozzáféréses(DMA) I/O átvitel.

5.3.2.Párhuzamos adatátvitel

A mikroszámítógépes rendszerekben a legnagyobb számban, vagy megszakításos, vagy közvetlen tárolóhozzáféréses párhuzamos adatátvitel történik, kiegészítve a programozott I/O átvitellel.

a.)Programozott I/O

A programozott adatátvitelre az I/O eszközök közvetlen irányítása a jellemző, a programból periféria utasítások segítségével lehet elérni a beviteli/kimeneti eszközöket. Az I/O utasításokban a perifériát meg kell címezni; ez a cím vagy az I/O port sorszáma, vagy a memória egy tárolóhelyének(memory mapped addressing) a címe. A perifériautasítások a processzor egy regisztere és a periféria (regisztere) között valósítanak meg átvitelt.

Az alkalmazható programozott átviteli módszerek:

Feltétel nélküli(direkt) **programozott adatátvitel**, amelyet egyszerűbb esetekben lehet alkalmazni, mint például kijelzők(LED-ek) beállítása, érzékelők lekérdezése.

Feltételes, lekérdezéses adatátvitel(polled I/O), amelyben a periféria állapotjelzőinek ellenőrzése alapján és után történhet az átvitel; amennyiben a jelzőbit nem megfelelő az átvitelhez, akkor a processzornak várakoznia kell a megfelelő állapot bekövetkeztéig(váró ciklus). Az átvitel karakterenként történik; blokkos adatátvitelkor, amelyhez külön utasítás tartozik, a két oldalt(processzor - periféria) össze kell hangolni, szinkronizálni kell; erre szolgál a kérés-visszaigazolás(hand shaking) szinkronizálási módszere.

A programozott adatátvitel nem túl előnyös, mert lassú átvitelt eredményez és a *processzor idejét teljesen leköti*. Az átvitelt mindig a processzor kezdeményezi. Az előbbiek miatt nem túl gyakran alkalmazzák.

b.)Megszakításos I/O

A processzor idejét célszerű felszabadítani az átviteli feladatok alól, ezért olyan módszert kell alkalmazni, amely ezt lehetővé teszi. Ez a módszer a megszakításos adatátvitel, amelyben a periféria kezdeményezi az átvitelt. Az ilyen átvitelek lebonyolításához, az I/O portoknak regiszterekkel kell rendelkeznie.

Az átvitel kezdetén, a megfelelő utasítás kiadásával, a processzor jelzi az I/O eszköz számára az átvitelre vonatkozó indítási igényét. Az I/O eszköz ezután az átvitel kezdetére alkalmas időpontot a processzor felé küldött megszakítási kérelmével jelzi. A megszakítás kiszolgálása eredményezi az adatátvitelt.

A megszakításos adatátvitel előnye a programozott I/O átvitellel szemben, hogy a *processzor lényegesen kevesebb időt tölt* az adatátvitel irányításával, csak a megszakítás kiszolgálásának időtartama alatt foglalt. Ugyanakkor továbbra is a processzoron keresztül zajlik az átvitel és a nagyobb sebességű eszközök(pl.: merevlemez) esetében, ez az eljárás már nem kellően megbízható.

c.)Közvetlen tárolóhozfordulás(DMA)

A közvetlen tárolóhozfordulásos adatátvitelt a nagyobb sebességű eszközök használata és az adatblokkos átvitel esetén célszerű alkalmazni.

A közvetlen tárolóhozfordulásos átvitelnél, a processzor által elindított DMA vezérlő önállóan irányítja az adatátvitelt a tároló és a kijelölt I/O eszköz között a processzor kihagyásával. A processzor és a DMA vezérlő között a kapcsolat a megszakítási vonalak segítségével jön létre.

A DMA segítségével bonyolított adatátvitel a következő lépésekkel írható le:

processzor oldal:

- a processzor megvizsgálja az I/O eszköz állapotjelzőjét, hogy fogadni tudja-e az átviteli kérelmet; ha nem tudja fogadni, akkor egy hibavizsgáló rutin elindításával a processzor megpróbálja megállapítani a hiba okát;
- a processzor kiadja az átviteli utasítást és a szükséges paramétereket(az átviendő blokk kezdőcímét, hosszát, az átvitel irányát) az előírt módon előkészíti a DMA vezérlő számára;
- elindítja a DMA vezérlőt, amely átveszi az irányítást;

DMA oldal:

- sín előkészítése az adatátvitelre;
  - DMA sínkérelem jelzése(bus request),
  - sínciklus befejezése,
  - processzor engedélyezi a sín használatát(bus grant),
- cím sínre tétele;
- adat sínre tétele;
- számláló csökkentése; ha nem nulla, akkor további adatok címzése, átvitele;
- számláló = 0, akkor a sínkérelem megszüntetése;
- megszakítási kérelem a processzor felé;

processzor oldal:

- a processzor ellenőrzi a DMA vezérlőt az átvitel végrehajtásáról; a sín engedélyezést megszünteti.

A DMA vezérlő a sínrendszert kétféle módon tudja igénybe venni:

- ha az adatátvitel blokkos formájú, akkor célszerű a sínt az átvitel teljes időtartamára lekötni;
- ha az átviendő adatok nem blokkos formájúak, akkor csak egy-egy adat átvitelére kell igénybe venni a sínt; ezt az eljárást nevezik **ciklusloppásnak**(cycle stealing), amely tulajdonképpen a sín időosztásos használata a processzorral közösen.

A DMA vezérlők használatakor, annak több paraméterét választhatóan be lehet állítani. Ilyen lehetőségek:

- az átviteli kapcsolat választása; a tároló-I/O eszköz kapcsolat mellett, létrehozható memória-memória, vagy I/O-I/O eszköz közötti kapcsolat is;
- a sínhasználat módjának(ciklusloppásos, blokk-sínciklusos) választása;
- a DMA vezérlő csatornáihoz prioritási értékeket rendelhetünk hozzá.

A közvetlen tárolóhozfordulásos átvitel előnye, hogy a szükséges megszakítások száma csökken, így *a processzor ideje felszabadul.*

5.3.3.Soros adatátvitel

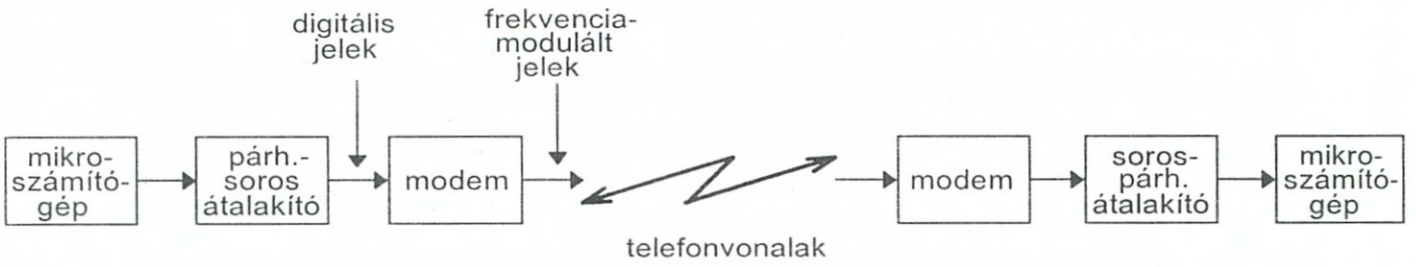
A számítógépen belül és kisebb mértékben külső átviteli célra használt párhuzamos adatátvitel mellett, amely nagyobb távolságú megbízható átvitelt nem tesz lehetővé, szükség van távoli I/O eszközök, esetleg egy másik számítógép elérésére is. Az ilyen átvitelek megvalósításának viszonylag egyszerű és megbízható formája az adatok **soros átvitele**.

a.)Soros átvitel lényege

A soros adatátvitel az adatok olyan továbbítása, amelyben az adatokat bitenként, a kiegészítő ellenőrző jelekkel együtt, időben egymás után továbbítja a számítógép.

Mivel a számítógépen belül az adattovábbítás párhuzamos formában történik, ezért a soros adatátvitelhez előbb szükség van egy párhuzamos-soros, illetve ilyen jelek fogadásakor, egy visszaalakító soros-párhuzamos átalakításra. (A párhuzamos-soros átalakításra szolgáló berendezés az **UART**-Universal Asynchronous Receiver/Transmitter.)

A nagyobb távolságra történő adattovábbításhoz, a telefonvonalakat lehet igénybe venni, amelyek használatához a jeleket rá kell 'ültetni' egy hangfrekvenciás hordozójelre. Az erre a célra szolgáló eszköz a **modem** (5-10.ábra).



5-10.ábra: *Soros átvitel modem segítségével*

A modemek(**mo**dulator/**dem**odulator) által használt leggyakoribb modulációs eljárások a frekvencia-, az amplitudó- és a fázismodulációs eljárások.

A két, modemen keresztül összekötött, berendezés egymással **fél-duplex**, illetve **duplex** üzemmódban tud kapcsolatot tartani. A fél-duplex üzemmódnál az adattovábbítás mindkét irányban lehetséges, de egyidőben csak az egyik irányban; a teljes duplex üzemmódban egyidőben mindkét irányban lehet adatokat továbbítani. Az összeköttetéshez 2-, vagy 4-vezetékes rendszer szükséges. A fél-duplex üzemmódhoz 2-, a duplexhez 4-vezetékes kapcsolat kell.

Az átvitel formája, elektromos jellemzői szabályozottak. Az átvitel módját meghatározó szabályrendszert nevezik **protokoll**nak. A soros adatátvitel lehet aszinkron és szinkron ütemezésű.

b.)Aszinkron átvitel

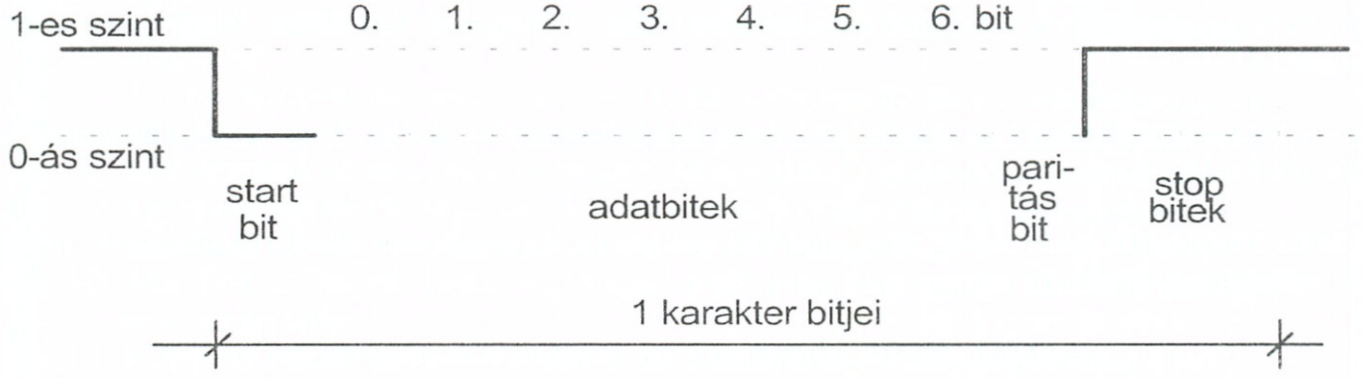
Az aszinkron ütemezésű adatátvitelnél, a karakterek ütemezés nélkül követik egymást. Minden karakter

1	start bitből,
7	adat bitből,
1	paritás bitből,
1-2	stop bitből, összesen 10-11 bitből tevődik össze (5-11.ábra).

A start/stop bitek miatt a jelsorozat eléggé redundáns, tehát információtartalom szempontjából felesleges jeleket is tartalmaz, ugyanakkor a vevő oldalon nincs szinkronizálva a vétel és emiatt nagyobb sebességű (>9600 bit/sec) átvitel nem biztonságos.

Az aszinkron soros átvitelhez használt átviteli sebességek:

110, 300, 1200, 2400, 9600, 19200 [bit/s]

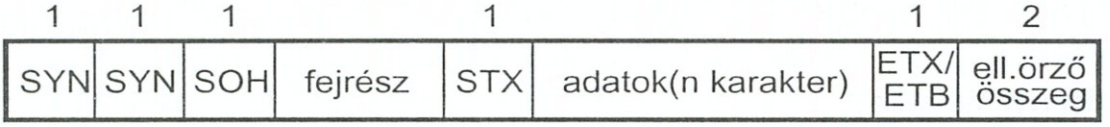


5-11.ábra: *Start-stop bitekkel kiegészített jelsorozat*

c.)Szinkron átvitel

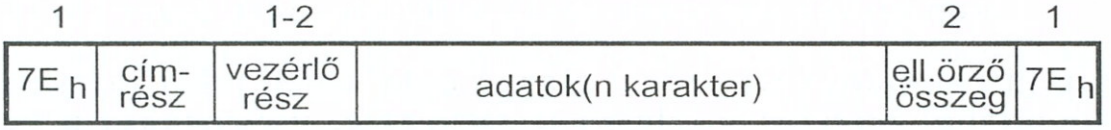
Szinkron adatátvitelnél, az egymást követő jelek ütemezetten, szinkronizáltan követik egymást. Az adatok átvitele blokkos formában történik, amelyet kiegészítenek még szinkronizáló bitekkel is. Ezt a formát nevezik 'frame'-nek, keretnek. Az 5-12.ábrán a keret két formája látható, a karakteres és a bináris adattovábbítás blokkja.

a.) Karakter orientált protokol



SYN = szinkronmező
SOH = fejrész kezdete(start of header)
STX = szöveg kezdete(start of text)
ETX = szöveg vége(end of text)
ETB = átviteli blokk vége(end of transmission block)

b.) Bitorientált protokol



5-12.ábra: *Szinkron átviteli protokollok blokkszerkezete*

Az átvitel egyik formája a **karakter orientált protokoll**(COP= character oriented protocol), amelynek használata esetén a grafikus adatok továbbítása körülményessé válik. Ezért azok átviteléhez a **bit orientált protokollt** (BOP=bit oriented protocol) használják. Ilyen, az ISO által is elfogadott bit orientált protokoll az ábrán is látható HDLC(high-level data link protocol) protokoll.

A szinkron átvitelnél a redundancia alacsonyabb, de a kapcsolódó hardver bonyolultabb.

Az alkalmazható sebesség magasabb, mint az aszinkron átviteleknél. A használt sebesség értékek:

4800, 9600 [bps], vagy magasabb érték.

5.3.4.Szabványosított interface

A külső párhuzamos és soros adatátvitelek lebonyolításához tartozó kimeneteket/bemeneteket(interface-eket) szabványosították.

A **párhuzamos csatlakozási felület** szokásosan alkalmazott szabványa a CENTRONICS szabvány, amely egy-, vagy kétirányú adatátvitelt is lehetővé tesz. Az egyirányú vezérlést a nyomtatók vezérléséhez alkalmazzák elsősorban, míg a szkennerekhez, vagy a számítógépek közötti nagyobb sebességű adatátvitelhez kétirányú kapcsolat kialakítására alkalmas kártyákat használnak.

A mikroszámítógépek környezetében használt szabványosított **soros csatlakozási felület**ek az Electronic Industries Association(USA) által kidolgozott:

RS 232C(20 vezetékes), és az

RS 449(30 vezetékes) szabvány.

Ez utóbbihoz kapcsolódik még két másik szabvány, amely az elektromos jellemzőket specifikálja, az RS 422A és az RS 423A előírás. Az első nagyobb sebességet és nagyobb kábelhosszt enged meg(2 Mbit/s, kb.65m), míg a második előírás csak 140 Kbit/s sebesség alkalmazását teszi lehetővé.

A sebességet [Baud]-ban is meg szokták adni(Baud rate), amelynek értéke a bináris jelek továbbításakor megegyezik a [bit/s]-ban mért értékkel.

5.4.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Milyen eszközei(erőforrásai) vannak a gépen belüli kapcsolatteremtésnek?
2. Milyen eszközök között kell biztosítani a hatékony adatforgalmat?
3. Mi a sínrendszer? Mi a feladata a sínrendszernek?
4. Milyen funkcionális részekből áll a sínrendszer?
5. Strukturálisan milyen részekre bontható a sínrendszer?
6. Milyen széles általában a címsín? Milyen széles általában az adatsín?
7. A sín használói milyen eszközök lehetnek? Mi a különbség az aktív(master) és a passzív(slave) sínhasználó eszközök között?
8. Mit értünk sínprotokol alatt? Mit értünk sínciklus alatt?
9. Milyen sínvezérlési módokat ismer? Vázolja fel a szinkronizálás elvét és magyarázza el a folyamatát!
10. Mit fed a 'hand-shaking' fogalom? Hogyan működik a 'kézfogásos' technika?
11. Milyen kapcsolatot jelent a kettős kérés-visszaigazolás (full handshaking) alkalmazása?
12. Mit értünk a sínrendszer aszinkron vezérlése alatt? Vázolja fel az aszinkron vezérlés folyamatát tükröző ábrát és magyarázza el a lejátszódó folyamatot!
13. Mit értünk szinkron vezérlés alatt? Vázolja fel a szinkron vezérlés folyamatát tükröző ábrát és magyarázza el a lejátszódó folyamatot!
14. Milyen módszereket alkalmaznak a tárkezelés gyorsítására a sínrendszer igénybe vételekor? Átlapolt sínciklusok segítségével hogyan lehet felgyorsítani a tárhozfordulások ütemét? Mit értünk blokk-sínciklus alatt?
15. Mit értünk sínfoglalás alatt? Milyen eszközök kezdeményezhetik a sín iránti igényüket? Milyen módszerekkel lehet kérni a sín foglalását?

16. Milyen sínfoglalási módszer a soros eljárás? Milyen prioritási elv érvényesül a soros sínfoglalási módszernél? Melyik eszköznek a legmagasabb a prioritása a soros sínfoglalásnál? Mit fed a 'daisy-chain' prioritási rend?
17. Miért a processzor a legalacsonyabb prioritású eszköz a soros foglalási rendszerben?
18. Milyen sínfoglalási módszer a párhuzamos eljárás?
19. Mi a szerepe, feladata a megszakítási rendszernek?
20. Milyen események befolyásolhatják a számítógépi folyamatok zavartalan lefutását? Mi lehet a megszakítási események kiváltó oka? Mihez viszonyítva vizsgáljuk a megszakítási események szinkronitását?
21. Mi a megszakítás és megszakítási kérelem? Milyen megszakítási kérelmek vannak és hogyan csoportosíthatók?
22. Mit értünk a megszakítások maszkolása alatt? Mely megszakítási kérelmek nem maszkolhatók?
23. Mit értünk a megszakítási kérelem kiszolgálása alatt? Sorolja fel, hogy milyen problémákat kell megoldani egy megszakítás kiszolgálásához?
24. Milyen módszerekkel állapítható meg a megszakítás keletkezési helye? Milyen szoftver és hardver módszereket ismer?
25. Milyen módszer a vektoros megszakításkezelő eljárás és milyen változatait ismeri? Mi található a megszakítási vektortáblákban?
26. Ismertesse a megszakítási kérelem kiszolgálásának fő lépéseit! A kiszolgálás mely lépéseit hajtja végre a processzor és mely lépéseit hajtja végre a kérelmet kiszolgáló programrész?
27. Milyen lehetőségeket ismer az egyidőben beérkező megszakítási kérelmek sorolására? Miért nem előnyös a megszakítási kérelmek egyenkénti(egymás utáni) kiszolgálása?
28. Milyen módon állapítható meg a megszakítási kérelem prioritása szoftver úton? Milyen hardver sorolási módszereket ismer? Mi a 'daisy-chain' sorolási módszer?
29. Mit értünk többszörös megszakítás alatt? Milyen rendszer az egyszintű megszakítási rendszer?
30. Milyen szabályokat alkalmaznak a többszintű megszakítási rendszerben a kérelmek kiszolgálására?
31. Mit értünk I/O portok alatt? Milyen regiszterek alkotják az I/O portokat? Mire szolgál a perifériák parancsregisztere, illetve állapotregisztere?
32. Milyen utasításokkal lehet elérni az I/O portokat? Mit jelent a tárolóleképzésű(memory mapped) perifériacímzés?
33. Milyen módszereket alkalmaznak a perifériákkal való kapcsolattartás lebonyolítására?
34. Mit értünk az I/O eszközök felé irányuló feltételes, illetve feltétel nélküli, közvetlen adatátvitel alatt?
35. Milyen I/O adatátviteli mód a csatornák segítségével lebonyolított átvitel?
36. Mit értünk közvetlen memóriahozzáférés(DMA) alatt? Milyen szerepe van a DMA átvitel lebonyolításában a processzornak, illetve a DMA vezérlőnek?

37. Milyen párhuzamos adatátviteli módokat használnak? Mi ezek lényege? Mi az alkalmazásuk előnye-hátránya?
38. Miért rosszabb a megszakításos I/O adatátvitel, mint a közvetlen tárolóhoz fordulásos(DMA) módszer?
39. Mi a csatorna-elv lényege? Miért előnyösebb a csatornák használata, mint a DMA adatátvitelé? Milyen gépek esetében használják a csatornákat az I/O adatátvitelre?
40. Megoldható-e nagyobb távolságra lévő perifériák használata párhuzamos adatátvitel esetén?
41. Miért van szükség a soros I/O adatátvitelre? Mire szolgál a soros/párhuzamos átalakító, illetve a modem a soros átvitel lebonyolításában?
42. Hogyan történik az adatok soros továbbítása? Milyen soros adatátviteli módokat különböztetünk meg?
43. Milyen adatátviteli mód az aszinkron soros adatátvitel? Rajzolja fel egy karakter átvitelének jelalakját. Miért redundáns az aszinkron soros átvitel jelsorozata?
44. Milyen adatátviteli mód a szinkron átvitel? A szinkron, vagy az aszinkron adatátvitel biztosít magasabb adatátviteli sebességet?
45. Milyen szabványosított csatlakozási felületeket(interface-t) alkalmaznak a párhuzamos és soros adatátvitelhez?

6. FEJEZET

TIPIKUS PROCESSZOROK

6.1.ÁLTALÁNOS JELLEMZŐK

A processzorok a mikroszámítógépek leglényegesebb részét alkotják, amelyek leggyakrabban használt típusainak jellemző tulajdonságait érdemes jobban is megismerni.

Ebben a fejezetben a két fő processzor típus,

- az összetett utasításkészletű(CISC) és
- a redukált utasításkészletű(RISC) processzorok

legismertebb, vagy legjellemzőbb tagjait mutatjuk be részletesebben. Az összetett utasításkészletű processzorok közül az Intel és a Motorola processzorcsalád legfontosabb tagjait, valamint a RISC processzorok két jellegzetes képviselőjét ismertetjük vázlatosan. Bővebb és pontosabb információkat a gyártó cégek ismertetői nyújtanak.

6.1.1.Összetett utasításkészletű(CISC) processzorok

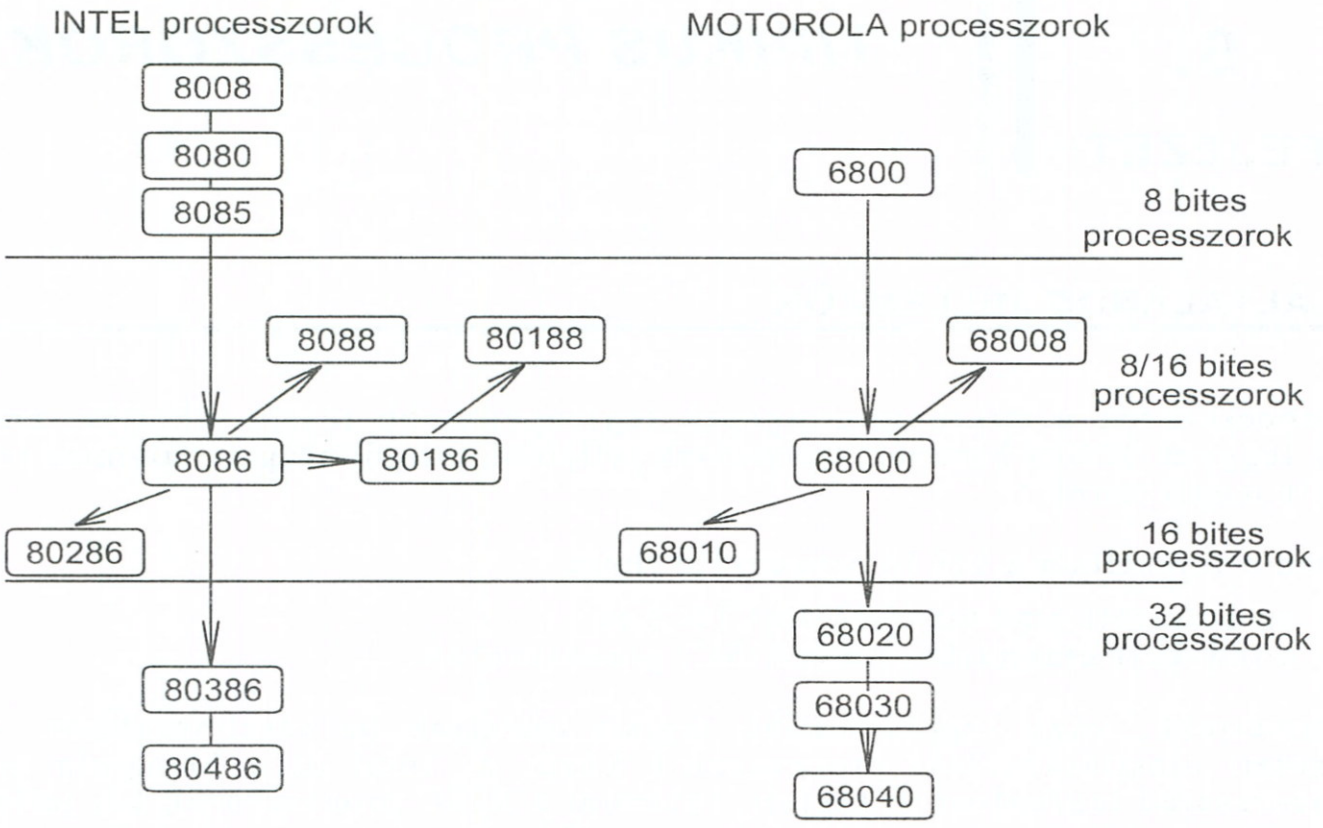
Az ismertetendő két processzorcsalád mind 'tudásában', mind fejlődési lépcsőiben igen sok hasonlóságot mutat. Ismertségüket leginkább a felhasználóik által(IBM az egyik, Apple, Atari, Sun a másik oldalon) szerezték meg és az egyes géptípusok elterjedtségének megfelelően váltak széles körben ismertté.

Az alábbiakban összefoglaljuk a két család tagjaira vonatkozó leglényegesebb jellemzőket. A két 'családfá'-t a 6-1.ábrán mutatjuk be.

a.)INTEL processzorcsalád

A legismertebb processzorcsaládot az Intel cég termékei alkotják, annál is inkább, mert a különböző IBM kompatibilis gépekben többnyire ezeket használják. Az Intel által gyártott i4004-es processzor volt az első egytokos központi egység 1971-ben. Az egyes processzorok legfontosabb adatait a 6-1.táblázat foglalja össze.

Az **i4004**-es processzor 4 bites, igen egyszerű központi egység, amelynek címezhető memóriatartománya összesen 1KB volt. Az ezt követő **i8008**-as már 8 bites processzor, amelynek címezhető tárolóterülete 16KB-nyi volt.



6-1.ábra: *Intel és Motorola processzorok családfája*

Az első igazi, általános célú 8 bites processzor az **i8080** processzor volt, amelynek némiképp módosított, más tokozású változata az **i8085**-ös egység.

6-1.táblázat: *Intel processzorok főbb adatai*

processzor típusa	címsín szélessége	adatsín szélessége	címezhető memória	kibocsátási év
i4004	10	4	1K	1971
i8008	16	8	16K	1972
i8080	16	8	64K	1974
i8085	16	8	64K	1974
i8086/88	20	16/8	1M	1978/79
i80186/188	20	16/8	1M	1982
i80286	24	16	16M	1983
i80386	32	32	64T	1985
i80386SX	32	16	64T	1988
i80486	32	32	64T	1989

Az **i8086/88**-as processzorok 16, illetve 8 bites processzorok, amelyek címsíne 20 bites, így a címezhető memória 1MB nagyságú. Az igen nagy számban gyártott mikroprocesszorok közül az i8088-as változat az IBM-PC-k központi egysége volt. A processzorok utasításkészlete 133 db 1-6 byte hosszúságú utasításból áll. Közvetlenül 64KB(ez a max. szegmensméret) cí-

mezhető. Az I/O eszközök címtartománya 64KB, amelyből 256 címezhető közvetlenül.

Az **i80186/188**-as processzorok nem terjedtek el széles körben, bár tudásuk nagyobb, mint az i8086/88-as processzoroké. Ezek a processzorok beépített órajelgenerátorral, megszakításvezérlővel és DMA-val rendelkeznek. Utasításkészletük megegyezik az I8086/88-as processzorokéval, kibővítve további 10 utasítással.

Az **i80286**-os 16 bites processzor(24 bites címsin) egy új szakaszt nyitott a processzorgyártásban, mivel memóriakezelése(virtuális tárkezelés), a védett üzemmód alkalmazása, lehetővé tette a multiprogramozott(multitaszkos) feldolgozást. Utasításkészlete 159 utasításból áll. A processzorból hiányzik az i80186/188-ba beépített órajelgenerátor, megszakításvezérlő, DMA. Ugyanakkor a virtuális tárkezelés kapcsán a szegmensleírók tárolására kisméretű cache-tárat építettek a processzorba. Az i80286-os processzor alkotja az IBM-PC/AT gépek központi egységét.

Az **i80386**-os valódi 32 bites processzor, amely szintén széles körben alkalmazott. Memóriakezelése lehetővé teszi a lapozásos technika használatát a szegmentálás mellett. Az i80386-osnak egy egyszerűbb változata az **i80386SX** típus, amely 16 bites adatsínnel és az i80286-ossal megegyező tokozással rendelkezik és ez utóbbi helyettesíthető vele.

Az **i80486**-os processzor az i386-os továbbfejlesztése, magában foglal egy 8KB-os általános célú cache-tárat, lebegőpontos koprocesszort és tárolókezelő egységet(MMU) is.

A processzorgyártás fejlődésére jellemző, hogy az i8086-os processzor még csak kb. 30000, az i80486-os processzor már 1.16 millió tranzisztort tartalmaz.

b.)MOTOROLA processzorcsalád

A másik széleskörben használt és emiatt fontos processzorcsalád a Motorola cég MC680x0 sorozata, amelynek tagjai az Apple, az Atari, az Amiga, a Sun Microsystems gépek központi egységeiként működtek és működnek. A család legfontosabb tagjainak lényeges adatait a 6-2.táblázat foglalja össze.

A Motorola cég az Intel8080-as processzorral körülbelül egyidőben hozta ki első processzorát, az **MC6800**-ast. Az MC6800-as, amelynek a 6809-es egy bővített változata, 8 bites processzor, amelyet ipari készülékekben használtak elsősorban.

A jelenleg forgalomban lévő processzorcsalád első tagja egy teljesen új fejlesztésű processzor volt, az MC68000-es típusú. Az **MC68000**-es processzor belső regiszterei 32 bitesek, adatsíne viszont csak 16 bites volt. Ez a processzor volt a központi egysége az Apple Macintosh, az Atari, a Commodore Amiga mikroszámítógépeknek. A címezhető memóriatartománya 16 MB nagyságú. Az MC68000-es processzor egyszerűbb változata az **MC68008**-as típus, amely 8 bites adatsínnel rendelkezik.

6-2.táblázat: *Motorola processzorok főbb adatai*

processzor típusa	címsín szélessége	adatsín szélessége	címezhető memória	kibocsátási év
MC68000	24	16	16M	1979
MC68008	22	8	4M	1982
MC68010	24	16	16M	1983
MC68012	31	16	2G	1983
MC68020	32	32	4G	1984
MC68030	32	32	4G	1987
MC68040	32	32	4G	1989

A következő piaci forgalomba került processzor az **MC68010**-es, amely már virtuális tárolókezelést is alkalmazott. Adatsíne 16 bit, címsíne 24 bit széles, így 16 MB memória kezelésére alkalmas. Az **MC68012**-es csak annyiban különbözik az MC68010-estől, hogy több címvonala van és így 2GB tárolót lehet használni mellette.

A Motorola első 32 bites processzora az **MC68020**-as, illetve annak tárolókezelővel(MMU) bővített változata, az **MC68030**-as típus. Mindegyik 32 bites cím- és adatsínnel rendelkezik és a címezhető memóriatartomány 4GB nagyságú. A két processzor igen nagy számban került felhasználásra, elsősorban nagy teljesítményű munkaállomások processzoraiként.

Az **MC68040**-es típus, amely az MC68030-asnak gyorsabb változata, abban is különbözik elődjétől, hogy beépített koprocesszort és cache-tárat is tartalmaz.

6.1.2.Redukált utasításkészletű(RISC) processzorok

A csökkentett utasításkészletű(RISC) processzorok közül két jellegzetes típust tekintünk át röviden. Mind a kettő kutatóintézetek eredményeihez kapcsolódik igen szorosan.

A Berkeley University által kifejlesztett RISC processzorok utódjának lehet tekinteni a Sun Microsystems **SPARC** mikroprocesszorát. A SPARC processzorok úgynevezett 'nyilt architektúrá'-t képviselnek, amelyhez a Sun cég meghatározta azt, hogy assembly szinten milyennek lássa a programozó a processzort és nem definiálta konkrétan annak megvalósítási formáját. Azt lehet mondani, hogy a céljuk az volt, hogy a processzort szoftver oldalról könnyen lehessen kezelni.

Evvel ellentétes megközelítésű, a Stanford University kutatásaira alapuló MIPS Computer Systems **R2000, R3000, R4x00**-as processzorainak kialakítása. A processzorok tervezői elsősorban a hardver gyorsaságát kívánták biztosítani és a különleges, ritkább feladatok elvégzését a programozóra bízták.

6.2.INTEL PROCESSZOROK

Az Intel processzorok közül a következő részekben csak a legfontosabbaknak tekintettekkel foglalkozunk, így a 16 bites i8086-os, i80286-os és a 32 bites i80386-os, i80486-os processzorokkal.

6.2.1.i8086/88, i80286-os processzorok

Az i8086/88-as processzorok felépítésükben egyformák, a különbség csak annyi, hogy az i8088-as típus 8 bites adatsínnel rendelkezik, az i8086-os pedig 16 bites processzor, amelyek címsíne 20 bites, így a címezhető tartomány 1MB nagyságú.

Az i80286-os processzor, amely az IBM-PC/AT gépek processzora, felépítését tekintve, hasonló az i8086-oshoz, avval a többlettel, hogy a kompatibilitás megőrzése végett, kétféle üzemmódot alakítottak ki rajta. Az i80286-os processzor

valós(real) üzemmód választása esetén úgy működik, mint egy i8086-os processzor, azaz a címezhető memóriatartománya csak 1MB;

védett(protected) üzemmód választásakor a címezhető fizikai tárolóterület a 24 vezetékes címsín miatt 16MB, míg a virtuális memóriaterület 1GB nagyságú. *A védett üzemmód és a virtuális memóriakezelés csak az i80286-os(illetve az azt követő processzorok) jellemzője.*

A továbbiakban a processzorokat együtt vizsgáljuk, elsősorban az i80286-os processzor jellemzőire koncentrálva és külön megemlítve a lényeges eltéréseket.

a.)Felépítés

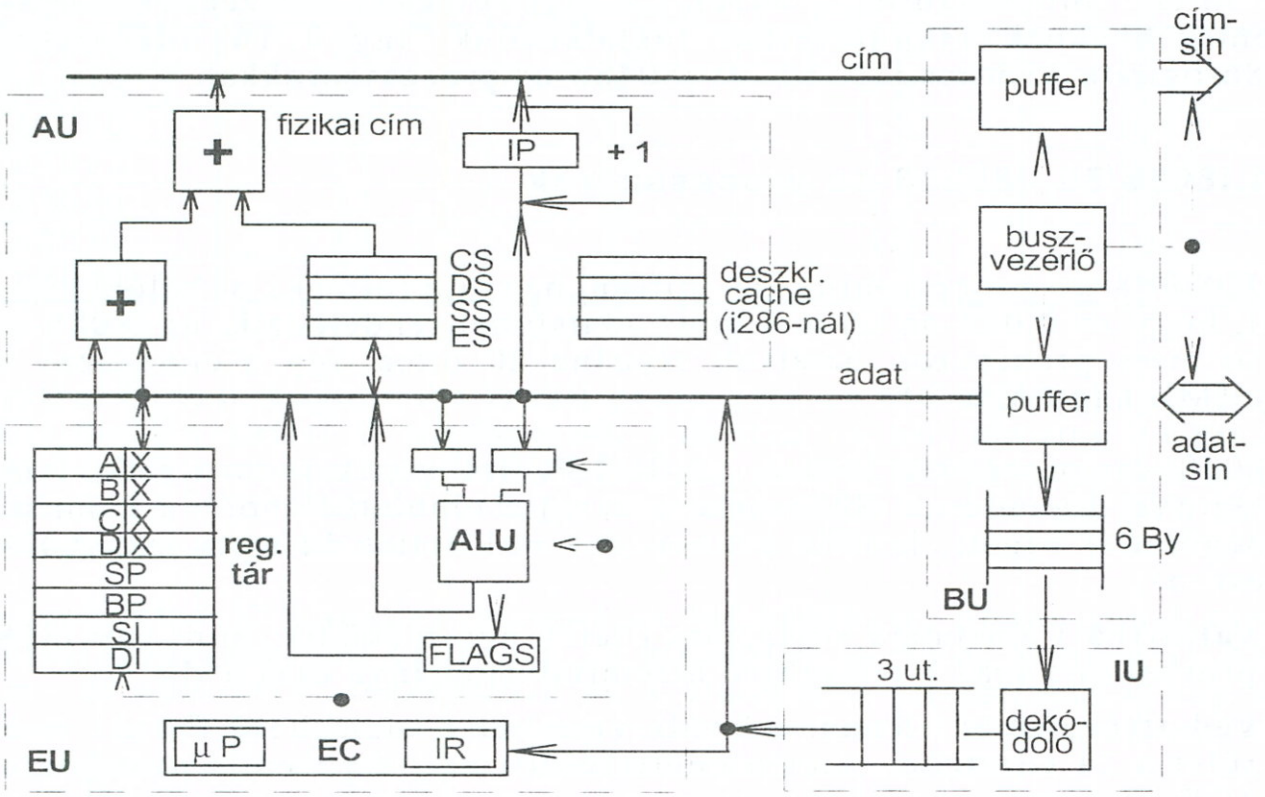
A processzorok belső felépítésének rajzát, fő egységeit a 6-2.ábra mutatja be.

A processzor négy funkcionális részből áll. A **sínvezérlő egység** (BU=bus unit) feladata a processzor és a memória, vagy az I/O eszközök közötti kapcsolat kialakítása. Előkészíti a tárból az utasításokat, tárolja az adatokat; szabad sínciklus alatt előkészíti és tárolja a következő utasítást egy 6 byte-os(i8088-as processzornál 4 byte-os) tárban.

Az **utasításfeldolgozó egység**(IU=instruction unit) az előkészített utasítást dekódolja a végrehajtáshoz. Az utasításfeldolgozó egység egyszerre 3 utasítást képes dekódolni és sorbaállítani a végrehajtáshoz.

A **végrehajtó egység**(EU=execution unit) a dekódolt utasításokban előírtakat hajtja végre a műveleti vezérlő(EC=execution control) irányítása alatt. A műveleti vezérlő az utasításregiszterben(IR) tárolt műveleti kód alapján elindítja a ROM tárban tárolt végrehajtó mikroprogramot. A végrehajtó egység

foglalja magában az aritmetikai egységet(ALU) és egy 8 regiszterből álló általános célú regisztertárat.



6-2.ábra: *i8086/88-as és i80286-os processzorok felépítése*

Az utasításban lévő operanduscímeket a **címkiszámító egység**(AU=address unit) állítja elő és adja át a sínvezérlő egységnek íráshoz, olvasáshoz. A címkiszámító egység dolgozza fel a virtuális címeket is és határozza meg a pontos, fizikai címeket. A címkiszámító egység valós üzemmódban 20 bites, védett üzemmódban 24 bites címeket állít elő. A virtuális tárkezeléshez, a szegmensszelektorokat tároló szegmensregiszterek mellett, a szegmensleírók (deszkriptorok) tárolására egy kisméretű(4x48 bit) cache-tár található.

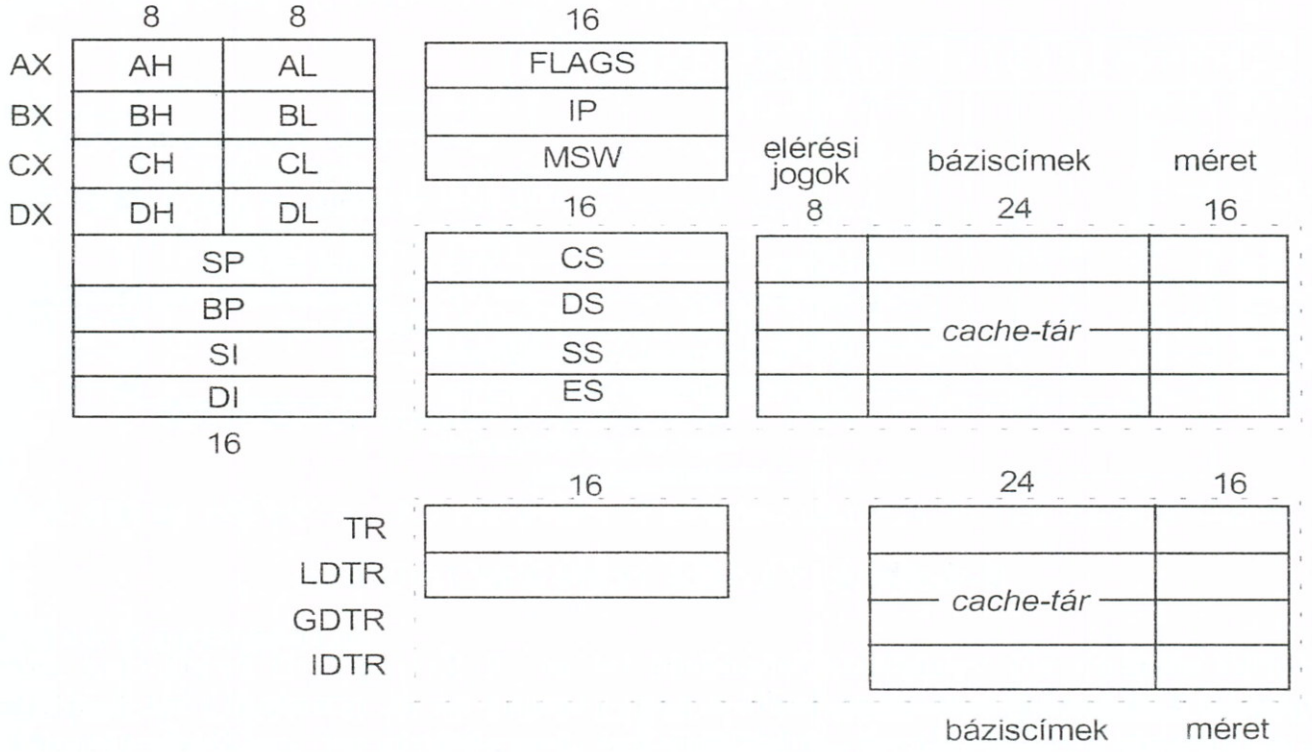
Általános regiszterek

A processzor végrehajtó egységében és a címkiszámító egységében több speciális funkciójú, vagy általános célú regiszter található (6-3.ábra).

Az általános célú regiszterek: **AX**(accumulator), **BX**(base), **CX**(count), **DX** (data), amelyek nevük szerint ugyan egyedi funkciójúak, de általános módon használhatók. Ezek 8 bites részei is önállóan címezhetők, megőrizve a kompatibilitást az i8086-os processzorral. További 4 regiszter a címkiszámításban játszik szerepet, ezek az **SI**(source index), **DI**(destination index), **BP** (base pointer), **SP**(stack pointer) regiszterek.

A címkiszámító egységben vannak a szegmens alapcímeket, vagy szegmens szelektorokat tároló regiszterek: **CS**(code segment), **DS**(data segment), **SS**

(stack segment), **ES**(extra segment). Védett üzemmód alkalmazásakor ezek mindegyikéhez tartozik egy, a felhasználó által nem elérhető, a szegmensleírót(deszkriptort) tároló cache-tár.



6-3.ábra: *Az i80286-os Intel processzorok regiszterei*

A címkiszámító egység regisztereihez tartozik még az **IP** utasításszámláló (instruction pointer) regiszter is.

További regiszterek még a védett üzemmód mellett használt deszkriptortáblák szelektorait tároló **TR**(task segment register) és **LDTR**(local descriptor table register) regiszterek. Ezek mellett magát a deszkriptort egy cache-tár tárolja.

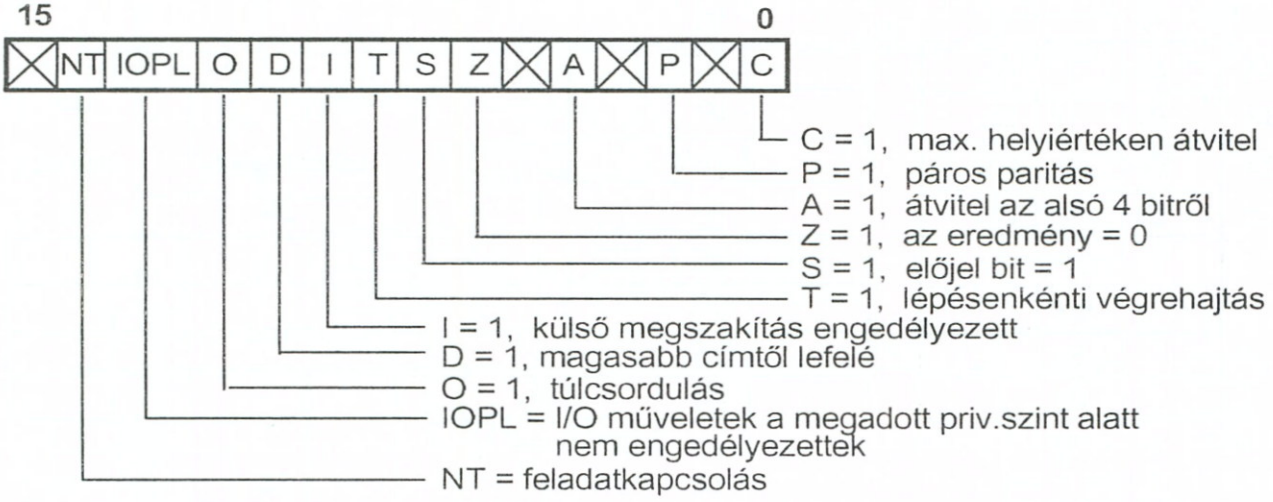
Állapotjelző, vezérlő regiszterek

A végrehajtó egységhez kapcsolódik a jelző- és vezérlőbiteket tartalmazó FLAGS regiszter, valamint az állapotjelző biteket tároló MSW(machine status word) regiszter.

Az utasítások végrehajtásának eredményét tükrözik vissza, a végrehajtás menetét szabályozzák a **FLAGS regiszter** egyes bitjei. A FLAGS regiszter tartalmát a 6-4.ábra mutatja be.

A regiszter néhány bitje az elvégzett művelet eredményével kapcsolatosan nyújt információt és ezek lekérdezésére, ellenőrzésére utasítások állnak rendelkezésre. A **C**(carry) bit akkor kap 1-es értéket, ha a legfelső helyiértéken átvitel keletkezik; a **P**(parity) bit az eredményben lévő 1-esek párosságát jelzi; az **A**(auxiliary) bit az alsó 4 bitről a felső 4 bitre történt átvitelt jelzi,

amelynek a BCD kódú számolásnál van jelentősége; a **Z**(zero) bit az eredmény nulla értékét jelzi; az **S**(sign) bit az eredmény negatív voltát jelzi; az **O**(overflow) bit az eredmény túlcsordulását jelzi.

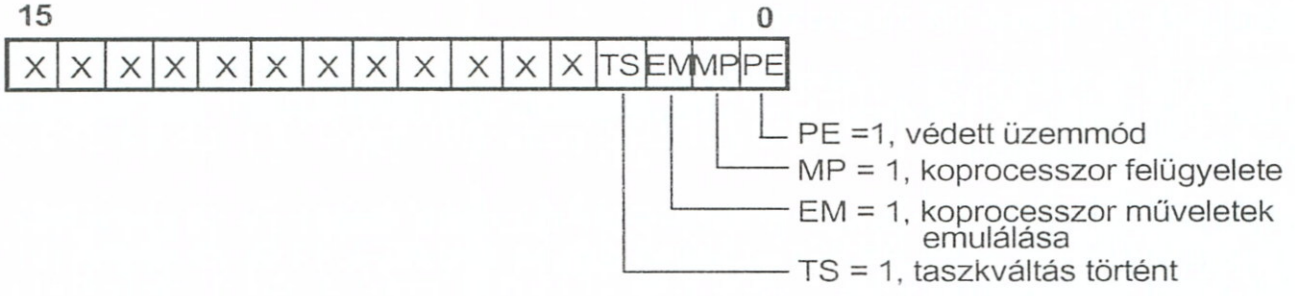


6-4.ábra: *FLAGS regiszter tartalma*

További néhány bit vezérlési feladatot lát el. A **T**(trap) bit beállítása esetén, az utasítás végrehajtása után leáll a processzor; az **I**(interrupt) bit beállításával a megszakításkérelmek feldolgozása engedélyezett; a **D**(direction) bit karaktersorozaton végzett műveletekkor az irány kijelölésére szolgál.

A fennmaradó két mező **csak az i286-osnál** használt, az **IOPL**(I/O privilage level) bitek az I/O műveletek engedélyezett védettségi szintjét jelzik; az **NT**(nested task) bit pedig a beágyazott feladatot, feladatvégrehajtást jelzi.

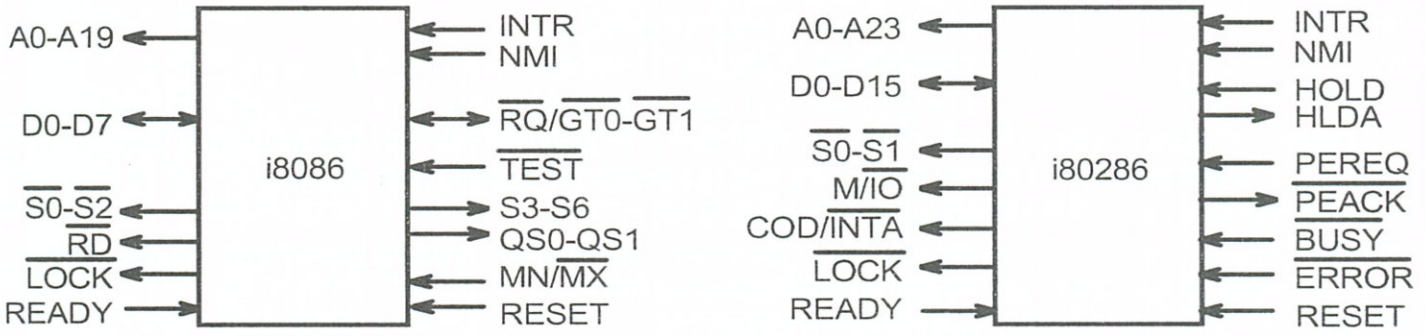
A processzor állapotát tükrözi vissza még a belső használatú **MSW(machine status word) regiszter**(6-5.ábra), amelynek bitjei a védett működési módot(**PE**=protected mode enabled), a koprocesszorral összefüggésben a taszkváltást(**TS**=task switched), valamint a koprocesszor(ha van) állapotát jelzik.



6-5.ábra: *MSW regiszter tartalma*

A TS=1 jelzi azt, egy lebegőpontos utasítás végrehajtás előtt, hogy időközben taszkváltás történt és a koprocesszor korábbi állapota még nem

lett elmentve esetleg. Az MP(monitor processor extension) bit beállítása a koprocesszor jelenlétét jelzi és várakozást jelent a lebegőpontos művelet befejezésére; az EM(emulate processor extension) 1-es értéke jelzi azt, hogy a lebegőpontos műveletet nem a koprocesszorral, hanem az operációs rendszer megfelelő rutinjával kell végrehajtani.



6-6.ábra: *i8088-os és i80286-os processzorok külső kapcsolatai*

b.)Külső kapcsolatok

A processzor kapcsolatait a külvilággal, a kivezetésein keresztül tartja. A következő, 6-6.ábra az i8088-as és az i80286-os processzorok legfontosabb logikai kapcsolódásait mutatja be.

Az egyes kivezetések funkciója az alábbi(a kivezetés megnevezése melletti csillag[*] a logikai alacsony szintet jelzi):

i8088 processzor

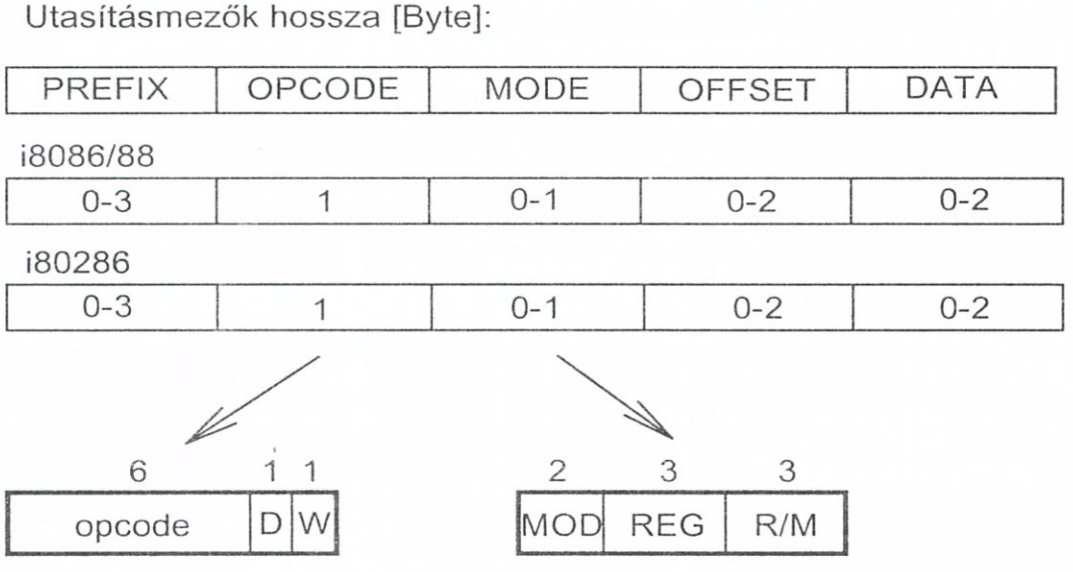
A0-A19	címvezetékek(20 db),
D0-D7	adatvezetékek(8 db),
S0*-S2*	sínciklus típus kijelölése,
RD*	olvasási ciklus jelölése,
LOCK*	sínfoglalás jelölése,
READY*	memória jelzése az adat rendelkezésre állásáról, vagy várakoztatásról,
INTR	megszakításkérés,
NMI	nem maszkolható megszakításkérés,
RQ*/GT0*-GT1*	sínfoglalás(arbitráció) kérés/engedélyezés, elsősorban a koprocesszor esetében,
TEST*	a koprocesszor ellenőrzésére, befejezte-e a lebegőpontos műveletet,
S3-S6	a processzor belső állapotáról ad információt,
QS0-QS1	a processzor belső állapotáról ad információt,
MN/MX*	a processzor minimum, illetve maximum módba kapcsolására,
RESET	a processzor alaphelyzetbe hozására szolgáló bemenet.

i80286-os processzor

A0-A23	címvezetékek(24 db),
BHE*	(bus high enable) byte címzés használatának lehetővé tétele, ha BHE*=1, mivel egyébként 16 bites szavakat olvas, illetve ír a processzor,
D0-D15	adatvezetékek(16 db),
S0*-S1*	sínciklus típus kijelölése,
M/IO*	sínciklus típus kijelölése(memória, vagy I/O),
COD/INTA*	sínciklus típus kijelölése; a négy kivezetés (S0*-S1*, M/IO*, COD/INTA*) együttesen határozza meg a kiválasztott sínciklus típusát,
LOCK*	sínfoglalás jelölése,
READY*	memória jelzése az adat rendelkezésre állásáról, vagy várakoztatásról,
INTR	megszakításkérés,
NMI	nem maszkolható megszakításkérés,
HOLD	sínfoglalás(arbitráció)kérése,
HLDA	sínfoglalás visszaigazolása,
PEREQ	koprocesszor adatelőkészítés kérése,
PEACK*	visszajelzés az adat sínre helyezéséről,
BUSY*	a koprocesszor foglaltságnak jelzésére,
ERROR*	a koprocesszor hibajelzése,
RESET	a processzor alaphelyzetbe hozására szolgáló bemenet.

c.)Utasításformák

Az i8088/86-os, i286-os Intel processzorok által használt utasításszerkezetek változó hosszúságú(1-9 byte) utasításokat eredményeznek, amelyek 1-, vagy 2-címes utasítások. Az utasítások általános szerkezetét a 6-7.ábra mutatja be.



6-7.ábra: *i8086/88 és i80286-os processzorok utasításszerkezete*

Az utasítás műveleti jelrész(opkód) byte-ja a műveleti előíráson kívül, gyakran két biten az operandus helyét és a hosszát is jelzi, mégpedig

ha	d=0	⇒	az eredmény helye a (MOD&R/M) mezők által meghatározott tárolóhely,
	d=1	⇒	az eredmény helye a regiszter;
ha	w=0	⇒	az operandus 1 byte-os,
	w=1	⇒	az operandus 2 byte-os.

A MODE byte a címzéseknél ismertetendő módon határozza meg az operandus helyét, majd ezt követően helyezkedik el 1-2 byte-on a szegmensen belüli relatív cím(displacement, offset) és a közvetlen adatkonstans 1-2 byte-on.

Az utasítást megelőzhetik 1-3 byte-on az ú.n. prefixek, amelyek módosítják az utasítás tartalmát, még pedig többnyire saját szegmensen kívüli címzés megadásakor használják.

d.)Címzési módok

A használható címzési módok mindegyike tulajdonképpen indirekt, vagy indirekt relatív címzési mód, mivel a szegmens alapcíme mindig szerepel a cím kiszámításának folyamatában.

A cím meghatározásában a MODE byte tartalma vesz részt. Annak egyik része (REG) az utasításban előírt művelethez szükséges egyik operandus regiszterét; másik két része(MOD, R/M) pedig a másik operandus helyét határozza meg. Az igénybevett regiszter méretét(8, illetve 16 bites) a műveleti jelrész melletti 'w' bit jelöli ki. A REG bitek értékének megfelelő regiszter hozzárendeléseket a 6-3.táblázat, a MOD és R/M bitek címzési módot meghatározó összerendeléseit a 6-4.táblázat tartalmazza.

6-3.táblázat: *Címzésnél használt regiszterek*

REG	16 bites (w=1)	8 bites (w=0)
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

A lehetséges címzési módok közül csak egy ad lehetőséget a közvetlen címzésre(direct addressing). A címzések többségében egy, vagy több regiszter(BX, BP, SI, DI) tartalmának és az utasításban megadott eltolásnak(displacement=dpl) az összege adja meg a memória azon helyét, melynek

tartalmát a műveletvégzéshez fell kell használni. A MOD mező [MOD]=11 tartalma mellett mindkét operandus valamelyik regiszterben található.

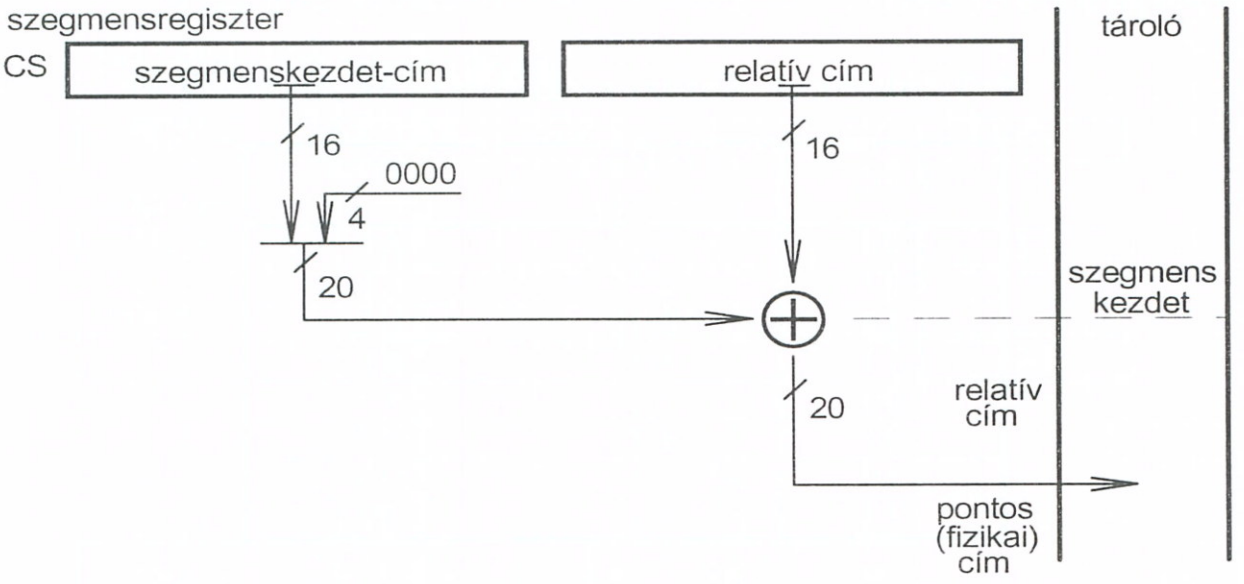
6-4.táblázat: *Címzési módok*

R/M	MOD 00	01	10	11
000	m[BX+SI]	m[BX+SI+d8]	m[BX+SI+d16]	AX vagy AL
001	m[BX+DI]	m[BX+DI+d8]	m[BX+DI+d16]	CX vagy CL
010	m[BP+SI]	m[BP+SI+d8]	m[BP+SI+d16]	DX vagy DL
011	m[BP+DI]	m[BP+SI+d8]	m[BP+SI+d16]	BX vagy BL
100	m[SI]	m[SI+d8]	m[SI+d16]	SP vagy AH
101	m[DI]	m[DI+d8]	m[DI+d16]	BP vagy CH
110	direkt címzés	m[BP+d8]	m[BP+d16]	SI vagy DH
111	m[BX]	m[BX+d8]	m[BX+d16]	DI vagy BH

A táblázatban az m[....] jelölés a zárójelben leírt kifejezés értéke, mint **pointer** által kijelölt memóriahely tartalmát jelenti. A [MOD]=01 oszlop címzési módjaiban 1 byte-os eltolással(d8), a [MOD]=10 oszlop címzési módjaiban 2 byte-os eltolással(d16) kell számolni.

e.)Virtuális címzés

Az i286-os processzor által nyújtott új lehetőség a virtuális címzés használata, amelynek révén a közvetlenül címezhető memória tartomány mérete: 1GB nagyságú lehet. Az i8088/86-os processzorok csak valós működési módban tudnak működni, így rájuk csak az ott leírtak érvényesek.



6-8.ábra: *i80286-os processzor címzési módja valós üzemmódban*

Valós működési mód

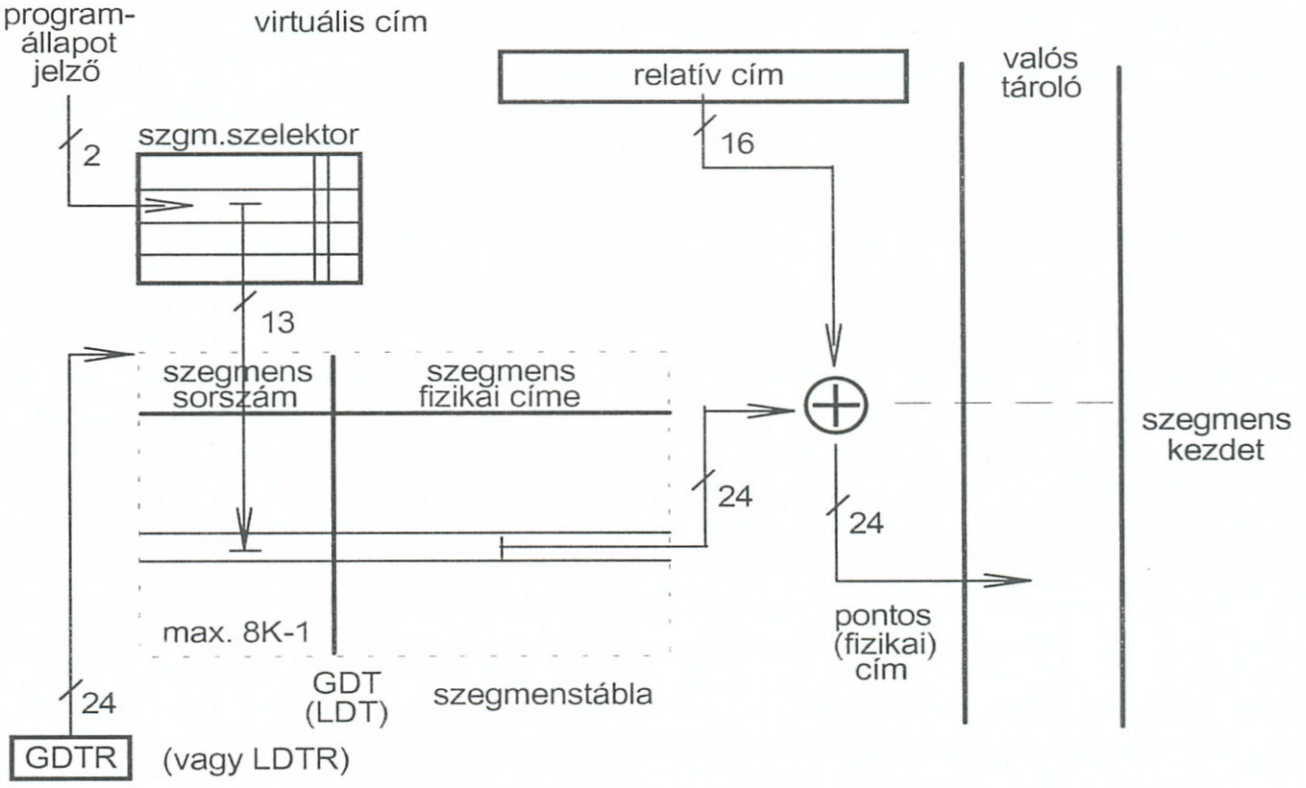
Valós(real) működési mód mellett a processzor úgy viselkedik, mint ha egy valódi i8086-os processzor lenne. Ennek megfelelően a maximális fizikai tárméret 2^{20}=1MB lehet és a legnagyobb szegmens mérete 2^{16}=64KB.

A valós működési mód melletti címkiszámítási módot mutatja be a 6-8.ábra rajza. A pontos cím értékét a szegmensregiszterben tárolt érték 4 nullával alulról kiegészitett értékének és az utasításban megadott relatív címnek az összege adja.

Védett, virtuális működési mód

Mint említettük, ez a működési forma az i8086-os processzornál nem létezik, csak az i286-os processzortól kezdve van ennek alkalmazására lehetőség. Az i286-ossal elérhető virtuális tárméret: $2^{14}.2^{16}=2^{30}$=1GB; ugyanakkor a fizikailag kezelhető méret(a 24 db címvezeték miatt), 2^{24}=16MB nagyságú. A virtuális cím méretét a szegmensszelektor felső 14 bitje és a 16 bites relatív cím alkotja.

Az i286-os processzor esetében csak szegmentált tárkezelésre van lehetőség és egy-egy szegmens maximális mérete a relatív cím által meghatározott, azaz 2^{16}=64KB méretű. A címkiszámítás módját a 6-9.ábra vázlata mutatja be.



6-9.ábra: *i80286-os processzor címzési módja védett üzemmódban*

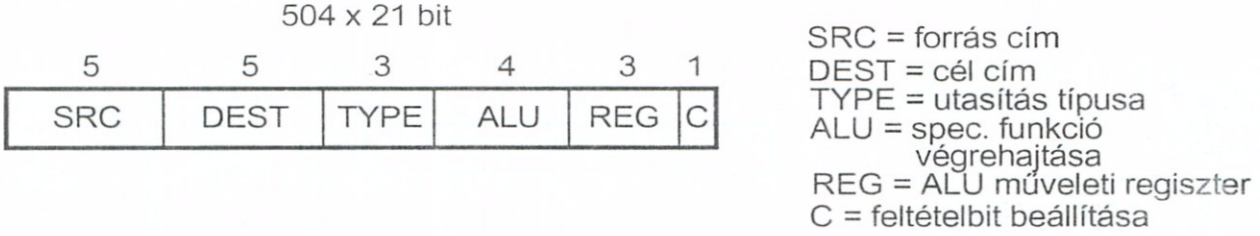
A szegmens kezdőcímét egy deszkriptor, a szegmensleíró tartalmazza, amely a globális(GDT), vagy a lokális(LDT) deszkriptortáblában található meg. A használandó deszkriptortáblát a szegmensszelektor 2.bitje jelöli ki. Kikeresés után, ez a deszkriptor ideiglenes jelleggel, átkerül a szegmensszelektorok mellett található cache-tárba.

A deszkriptorokról, a GDT-ről, illetve az LDT-ről az i386/486-os processzorok kapcsán teszünk említést.

f.)Utasítás végrehajtása, műveleti vezérlés

A gépi utasítások végrehajtásában egy 4-fokozatú átlapolódás (pipelining) érvényesül. Ezek a fokozatok: az utasítás előkészítése, az utasítás dekódolása, a címkiszámítás és a művelet végrehajtása, amelyek végrehajtására önálló egységek szolgálnak a processzoron belül.

Az utasítások **végrehajtásának vezérlésére** mikroprogramozott műveleti vezérlő szolgál, amely az összes Intel processzornál alapvetően azonos formájú. A mikroutasítások vertikális szerkezetűek, egy-egy mikroutasítás egy-egy elemi művelet végrehajtását eredményezi. A tipikus utasításszerkezet a 6-10.ábrán látható.



6-10.ábra: *Intel mikroutasítások felépítése*

Az utasítás első két mezője az adatmozgatás forrás- és a célregiszterének meghatározására szolgál, a további mezők, amelyek az egyes utasításokban az utasítás típusának megfelelően kis mértékben változhatnak, az utasítás típusát, az aritmetikai egység műveletét és az ehhez felhasznált operandus regiszterét jelölik ki, továbbá az utolsó bit a feltétel bit beállítását írja elő, ha szükséges. A TYPE mezőben megadható utasítás típusok:

- ALU művelet,
- tároló művelet,
- rövid, vagy hosszú ugrás,
- mikroeljárás hívás, stb.

A mikroutasítások 21 bit hosszúságúak és kb. 500 ilyen utasítást tárol a végrehajtás vezérlőben(EC) lévő ROM tároló.

6.2.2.i80386/486-os processzorok

Az i80386-os és az i80486-os processzorok 32 bites eszközök és belső felépítésük azonosnak modható, avval a kiegészítéssel, hogy az i486-os processzor beépített koprocesszorral és egy 8 KB-os belső cache-tárral rendelkezik. A 32 bites címsínből adódóan a címezhető tárolótartomány 4 GB nagyságú.

Az i80386/486-os processzorok, a kompatibilitás megőrzése végett, többféle üzemmódban képesek dolgozni.

Valós(real) üzemmód választása esetén a processzorok úgy működnek, mint egy i8086-os processzor, azaz védett üzemmód nélkül és a címezhető memóriatartománya is csak 1MB. Ez, a védelem hiánya miatt is, nem teszi lehetővé a multiprogramozott üzemmódot.

Védett(protected) üzemmód választásakor, további három lehetőség áll a felhasználó rendelkezésére.

- ***16 bites üzemmód*** választásakor a processzorok úgy működnek, mint az i286-os processzor, azaz csak a 16 bites utasítások használhatók, a címezhető tárolóterület 16MB, és csak a szegmentált tárkezelés alkalmazható.
- ***16 bites, virtuális i8086-os üzemmód*** választásakor, a védett üzemmódból eredően, az i8086-oson futó program úgy kerül végrehajtásra az i386/486-os processzoron, mintha annak egy taszkja lenne. Azaz kihasználva a processzor összes erőforrását, szimulálja az i8086-os környezetet, ellentétben a valós üzemmóddal, amikor csak az eredeti i8086-os számára lehetséges erőforrás mennyiséget(pl. tárolóméretet) tudja igénybe venni processzor.
- ***32 bites üzemmód***, amely a processzorok saját üzemmódja, az összes erőforrás igénybevételével, multitaszkos feldolgozási lehetőséggel.

A továbbiakban a processzorok között nem teszünk különbséget, külön megemlítve az esetleges eltéréseket.

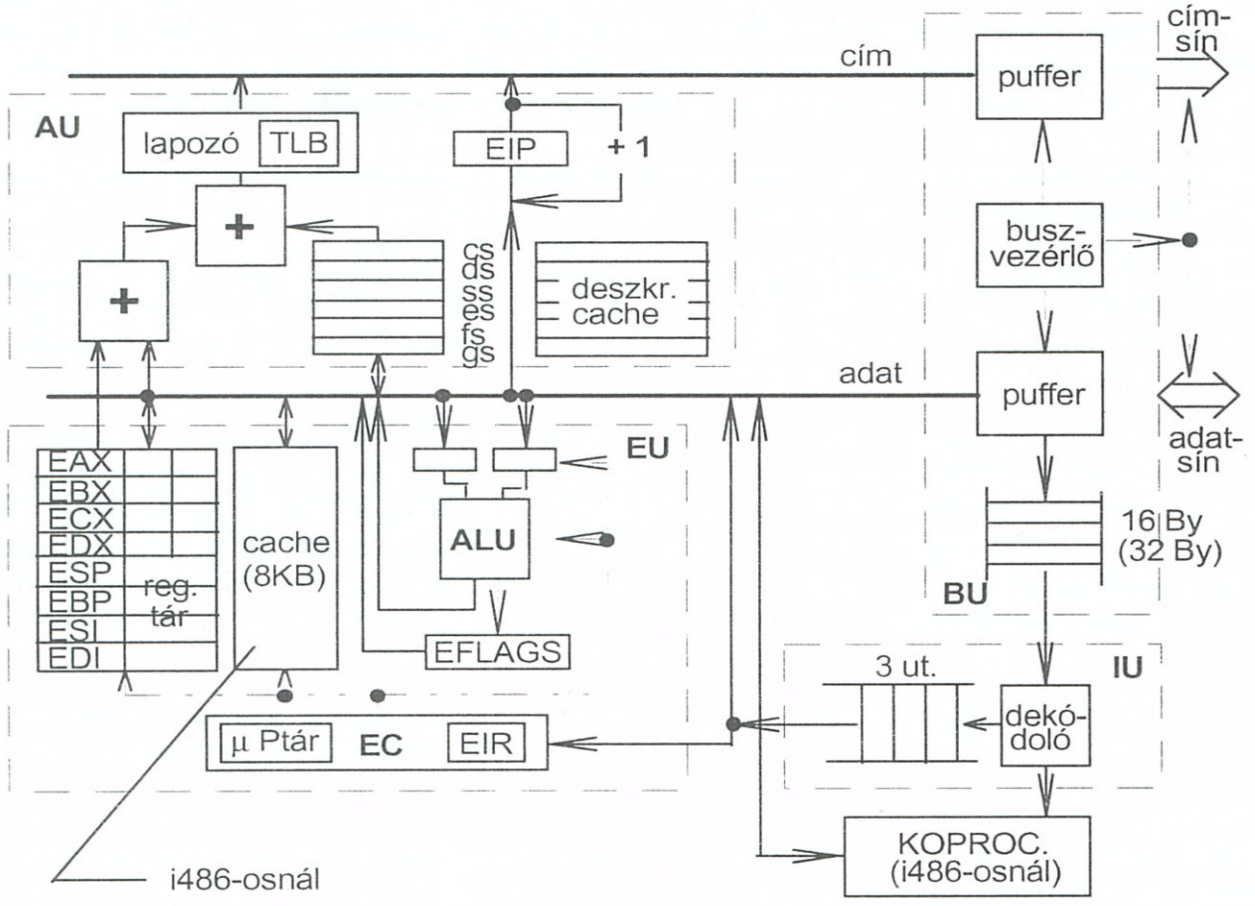
a.)Felépítés

Az i386/486-os processzorok belső felépítése nagy mértékben hasonló az i286-oséhoz, azokkal a többletekkel, amit a memóriakezelés(lapozás), a belső cache(i486-osnál), a beépített koprocesszor(i486-osnál) jelent. A processzor rajzát, fő egységeit a 6-11.ábra mutatja be.

A processzor, hasonlóan a korábbi változatokhoz, négy funkcionális részből áll. A **sínvezérlő egység**(BIU=bus interface unit) feladata a processzor és a memória, vagy az I/O eszközök közötti kapcsolat kialakítása. Előkészíti a tárból az utasításokat, tárolja az adatokat; szabad sínciklus alatt előkészíti és tárolja a következő utasítást egy 16 byte-os tárban.

Az **utasításfeldolgozó egység**(IU=instruction unit) az előkészített utasítást dekódolja a végrehajtáshoz. Az utasításfeldolgozó egység egyszerre 3 utasítást képes dekódolni és sorbaállítani a végrehajtáshoz.

A **végrehajtó egység**(EU=execution unit) a dekódolt utasításokban előírtakat hajtja végre a műveleti vezérlő(EC=execution control) irányítása alatt. A műveleti vezérlő az utasításregiszterben(EIR) tárolja az utasítást és a műveleti kód alapján elindítja a ROM tárban tárolt végrehajtó mikroprogramot. A végrehajtó egység foglalja magában az aritmetikai egységet(ALU) és egy 8 db 32 bites regiszterből álló általános célú regisztertárat. Az i486-os processzor esetében ez még kiegészül egy 8 KB-os, általános célú cache-tárral.



6-11.ábra: *i80386/486-os processzorok felépítése*

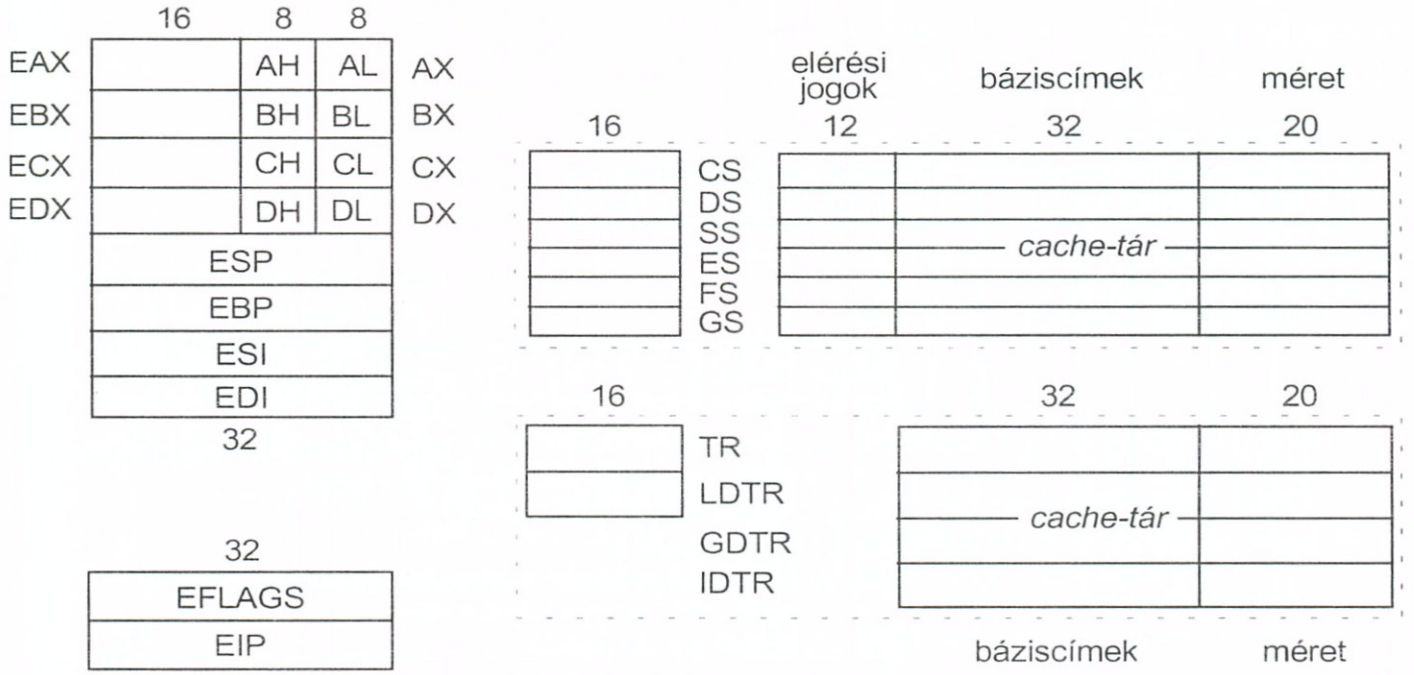
A végrehajtó egységhez kapcsolódik logikailag az i486-os processzor beépített lebegőpontos processzora is.

Az utasításban lévő operanduscímeket a **címkiszámító egység**(AU=address unit) állítja elő és adja át a sínvezérlő egységnek íráshoz, olvasáshoz. A címkiszámító egység a címösszeadókon kívül, még tartalmaz egy lapozó, lapfordító egységet is, a leggyakrabban használt lapok adatainak tárolására szolgáló cache-tárral(TLB=Translation Lookaside Buffers) együtt. A címkiszámító egység dolgozza fel a virtuális címeket és határozza meg a pontos, fizikai címeket. A virtuális tárkezeléshez, a szegmensszelektorokat tároló

szegmensregiszterek(6 db) mellett, a szegmensleírók (deszkriptorok) tárolására egy kisméretű(6x64 bit) cache-tár található.

Általános regiszterek

A processzor végrehajtó és címkiszámító egységében több, speciális funkciójú, vagy általános célú regiszter található(6-12.ábra). A regiszterek nevében a 32 bites változatot az E betű jelzi.



6-12.ábra: *i80386/486-os processzorok általános regiszterei*

Az általános célú regiszterek: **EAX**(accumulator), **EBX**(base), **ECX**(count), **EDX**(data), amelyek nevük szerint ugyan egyedi funkciójúak, de az i286-oshoz hasonlóan, általános módon használhatóak. Ezek 8, illetve 16 bites részei önállóan is címezhetők, megőrizve a kompatibilitást az i8086-os, i286-os processzorokkal. További 4 regiszter a címkiszámításban játszik szerepet, ezek az **ESI**(source index), **EDI**(destination index), EBP(base pointer), **ESP** (stack pointer) regiszterek.

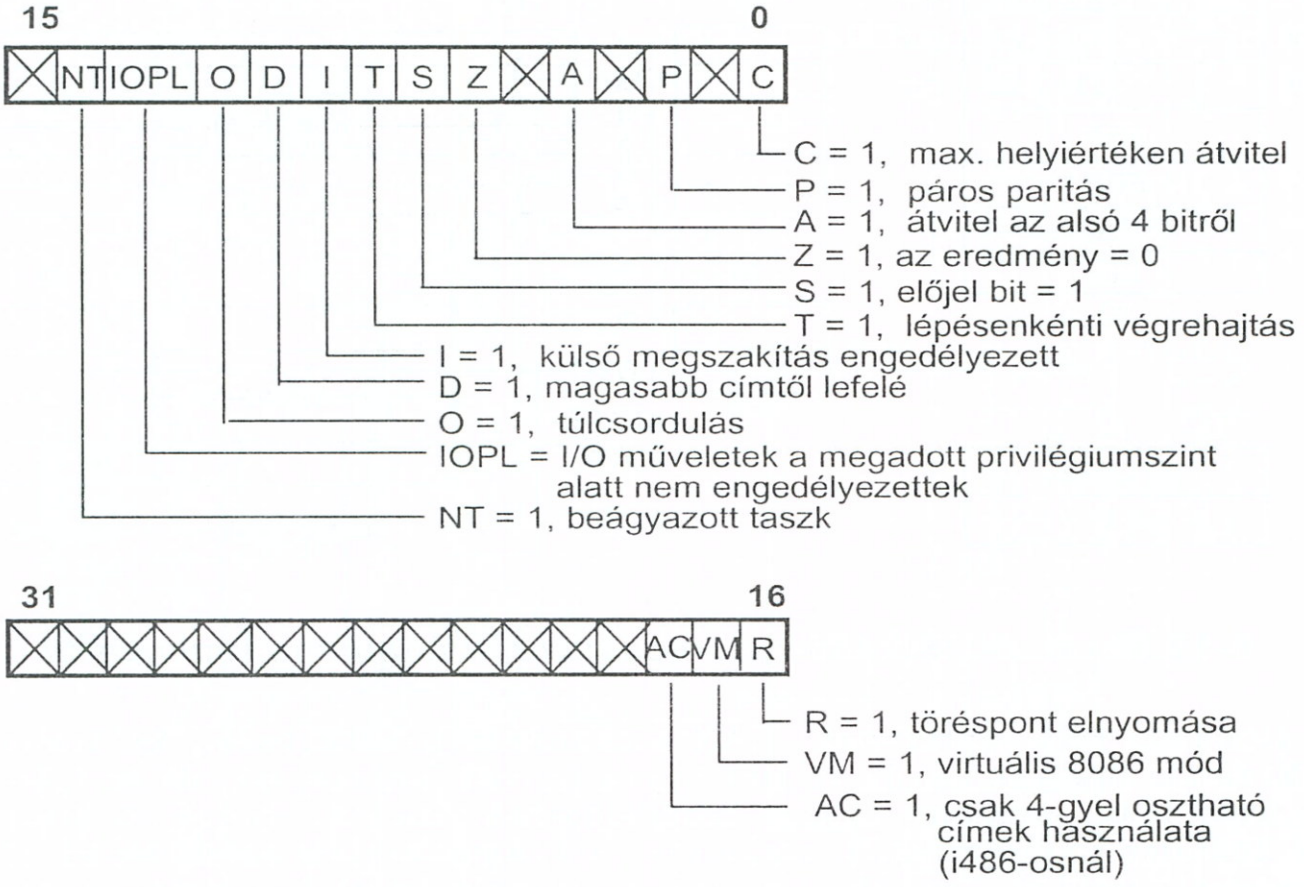
A címkiszámító egységben vannak a szegmens alapcímeket, vagy szegmens szelektorokat tároló regiszterek: **CS**(code segment), **DS**(data segment), **SS** (stack segment), **ES**(extra segment), valamint további két szegmensregiszter (**FS**, **GS**). Védett üzemmód alkalmazásakor ezek mindegyikéhez tartozik egy felhasználó által nem elérhető, a szegmensleírót(deszkriptort) tároló cache-tár.

A címkiszámító egység regisztereihez tartozik még az **EIP** utasításszámláló (instruction pointer) regiszter is.

Állapotjelző, vezérlő regiszterek

A végrehajtó egységhez kapcsolódik a jelző- és vezérlőbiteket tartalmazó **EFLAGS** regiszter(6-13.ábra), valamint az állapotjelző és vezérlő biteket tároló **CR0-CR3** vezérlő regiszterek(6-14.ábra).

A CR0 regiszter tartalmazza az általános vezérlő biteket, a CR1 regiszter nincs használatban, a CR2 regiszter az utolsó lapváltást előidéző pontos címet tárolja, a CR3 regiszter pedig a lapkatalógus kezdőcímét(lapkeret sorszámát) tárolja.

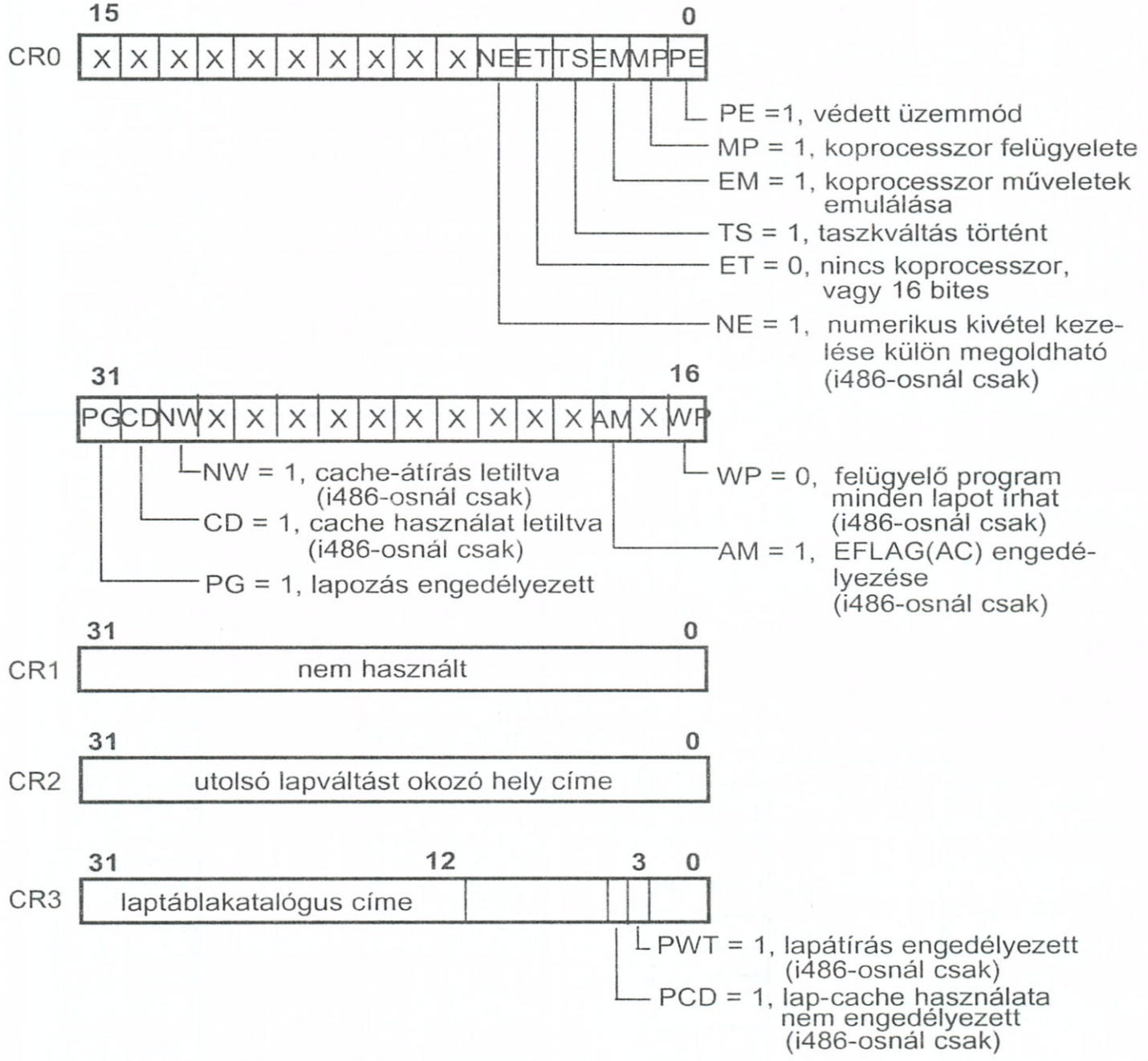


6-13.ábra: *EFLAGS regiszter tartalma*

A hibakeresést segítik a töréspontok tárolására szolgáló és a folyamatot szabályozó **DR0-DR7** regiszterek; a processzor belső teszteléséhez (cache-tár, TLB cache) a **TR3-TR5**(csak az i486-osnál) és a **TR6-TR7** regiszterek használatosak.

További regiszterek még, a védett üzemmód mellett használt taszk állapotszegmens(TSS) és lokális deszkriptortábla szelektorait tároló **TR**(task segment register) és **LDTR**(local descriptor table register) regiszterek. Ezek mellett, magát a deszkriptort egy cache-tár tárolja. Ugyancsak a rendszerre-

giszterek közé tartozik a globális deszkriptortábla, valamint a megszakítási deszkriptortábla címét, méretét tároló regiszterek(**GDTR**, **IDTR**).



6-14.ábra: *CR0-CR3 regiszterek tartalma*

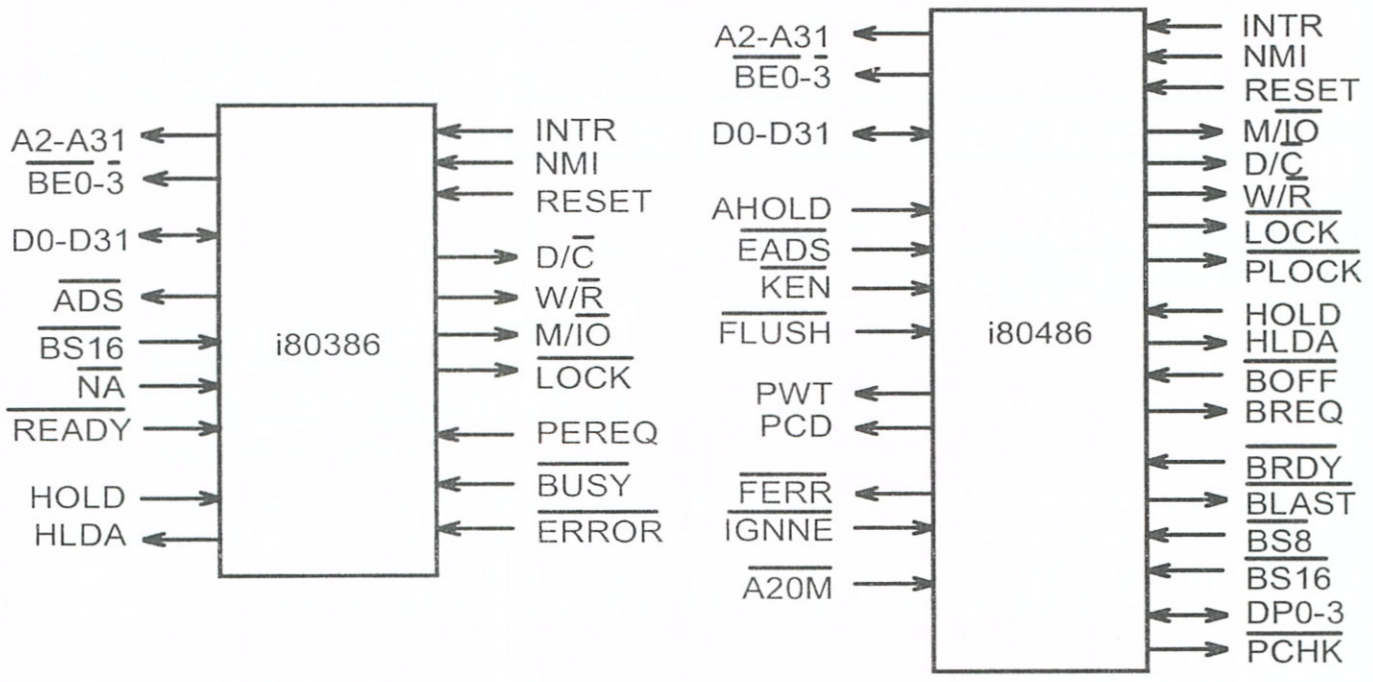
Az i286-os, i386/486-os processzorokban lévő regiszterek adatait foglalja össze a 6-5.táblázat.

6-5.táblázat: *i286-os, i386/486-os processzorok regisztereinek adatai*

A regiszter típusa	i286	i386	i486
általános regiszterek	8x16	8x32	
szegmens regiszterek	4x16	6x16	
szegmensleíró cache	4x48	6x64	
általános célú cache			8kB
rendszer regiszterek(TR, LDTR)	2x16	2x16	
rendszerregiszter cache	2x40	2x64	
rendszer regiszterek(GDTR, IDTR)	2x40	2x48	
utasításszámláló(IP)	1x16	1x32	
vezérlő regiszter(MSW,CR0)	1x16	1x32	
vezérlő regiszterek(CR1-CR3)		3x32	
nyomkövető regiszterek (DR0-DR7)		8x32	
tesztregiszterek(TR3-5)			3x32
tesztregiszterek(TR6-7)		2x32	

b.)Külső kapcsolatok

Az i386/486-os processzorok legfontosabb kapcsolatai a 6-15.ábra alapján:



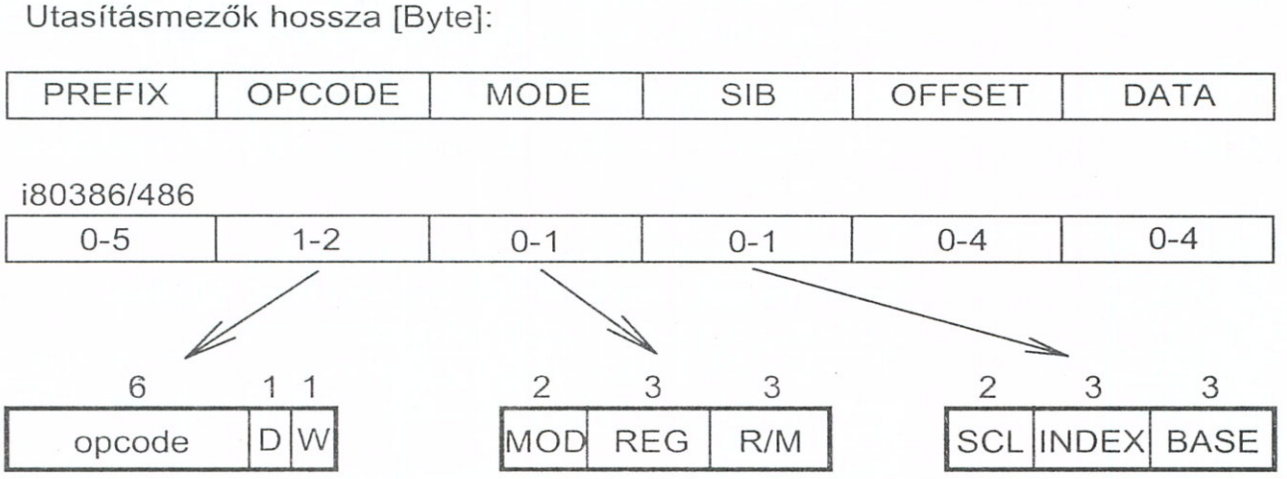
6-15.ábra: *i80386/486-os processzorok külső kapcsolatai*

A2-A31	címvonalak(30 db); mivel a processzor 32 bites szavakat kezel, ezért a használható címek 4-nek többszörösei; ezért az alsó 2 bit mindig 0 és A0-A1 nem szükséges,
BE0*-BE3*	(byte enabled) meghatározza azt, hogy melyik byte kerüljön átvitelre;
D0-D31	adatvonalak(32 db);
DP0-DP3	(i486-osnál csak): adat paritásbitek; minden bytehoz egy tartozik;
PCHK*	(i486-osnál csak): olvasáskor az engedélyezett byteok paritás bitje;
W/R*	sínciklus típus kijelölése; írás/olvasás megkülönböztetése;
D/C*	sínciklus típus kijelölése; adat, vagy vezérlési ciklus (megszakításvisszaigazolás, utasításelőkészítés, halt) megkülönböztetése;
M/IO*	sínciklus típus kijelölése; memória, vagy I/O ciklus;
LOCK*	sín foglaltságának jelzése;
PLOCK*	(i486-osnál csak): az aktuális átvitel befejezése több sínciklust igényel;
READY*(RDY*)	jelzés a sínciklus befejeztéről(i486-osnál);
ADS*	jelzi, hogy érvényes sínciklus van folyamatban;
NA*	(i386-osnál csak): a következő cím előkészítése megkezdődhet a sínciklus befejezte előtt,
BS16*	16 bites sín csatlakozik a processzorhoz, ezért egy 32 bites szót két 16 bites egységként visz át a sín;
BS8*	(i486-osnál csak): 8 bites eszköz csatlakozik a processzorhoz;
INTR	megszakításkérés,
NMI	nem maszkolható megszakításkérés;
BREQ	(i486-osnál csak): belső sínkérelem jelzése;
HOLD	sínfoglalás(arbitráció) kérése,
HLDA	sínfoglalás kérés visszaigazolása,
BOFF*	(i486-osnál csak): a sín lekapcsolása a processzorról;
BRDY*	(i486-osnál csak): blokk sínciklus(burst cycle) végének jelzésére;
BLAST*	(i486-osnál csak): ha a BRDY* megjelenik, akkor a blokk sínciklus befejeződött;
PEREQ	(i386-osnál csak): koprocesszor adatelőkészítés kérése;
BUSY*	(i386-osnál csak): a koprocesszor foglaltságnak jelzésére;
ERROR*	(i386-osnál csak): a koprocesszor hibajelzése;
RESET	a processzor alaphelyzetbe hozására szolgáló bemenet;
AHOLD	(i486-osnál csak): külső eszköz kéri a címsínt a memóriába íráshoz;
EADS*	(i486-osnál csak): a címsínen érvényes cím van, ezt használja a processzor a belső cache tartalmának érvénytelenítéséhez;
KEN*	(i486-osnál csak): cache-be töltés engedélyezése;

FLUSH*	(i486-osnál csak): a belső cache alaphelyzetbe hozása;
PWT	(i486-osnál csak): lap átírás(page write-through) jelzése, tartalma megegyezik a laptáblázatbeli értékkel;
PCD	(i486-osnál csak): lap cache-be írásának tiltása(page cache disable), tartalma megegyezik a laptáblázatbeli értékkel;
FERR*	(i486-osnál csak): lebegőpontos hiba jelzése;
IGNNE*	(i486-osnál csak): lebegőpontos hiba figyelmen kívül hagyása és a végrehajtás folytatása más utasítással;
A20M*	(i486-osnál csak): az A20-as címvonal maszkolása a tárolóhoz fordulás előtt, az i8086-os processzornak megfelelő módon.

c.)Utasításformák

Az i386/486-os processzorok által használt változó hosszúságú(1-17 byte) utasításszerkezetek hasonlóak az i286-oséhoz és 1, vagy 2 címes utasítások. Az utasítások általános szerkezetét a 6-16.ábra mutatja be.



6-16.ábra: *i386/486-os processzorok utasításszerkezete*

Az utasítás műveleti jelrész(opcode) byte-ja a műveleti előíráson kívül, gyakran két biten az operandus helyét és a hosszát is jelzi, még pedig

ha	d=0	⇒	az eredmény helye a (MOD & R/M) mezők, vagy a (MODE & SIB) byte-ok által meghatározott,
	d=1	⇒	az eredmény helye a regiszter,
ha	w=0	⇒	az operandus 1 byte-os,
	w=1	⇒	az operandus 2 byte-os 16 bites adatkezelésnél és 4 byte-os 32 bites adatkezelésnél.

A MODE byte a címzéseknél ismertetendő módon határozza meg az operandus helyét, ezt követően helyezkedik el a SIB(scale-index-base) byte, amely az indexelést szabályozza, majd 4, 2, 1, vagy 0 byte-on a szegmensen belüli relatív cím(displacement, offset) és a közvetlen adatkonstans 4, 2, 1, vagy 0 byte-on.

Az utasítást megelőzhetik 1-5 byte-on az ú.n. prefixek, amelyek módosítják az utasítás tartalmát, még pedig többnyire saját szegmensen kívüli címzés megadásakor használják.

Az i386/486-os processzorok utasításkészletét úgy alakították ki a korábbi processzorokhoz képest, hogy minden 16 bites utasításnak megalkották a 32 bites megfelelőjét és minden memória utasítás számára lehetővé tették az összes 32 bites címzési mód használatát.

d.)Címzési módok

A cím meghatározásában a MODE és a SIB byte tartalma vesz részt. A MODE byte egyik része(REG) az utasításban előírt művelethez szükséges egyik operandus regiszterét; másik két része(MOD, R/M mezők) pedig a másik operandus helyét határozza meg. Az igénybevett regiszter méretét(8, illetve 16, vagy 32 bites) a műveleti jelrész melletti 'w' bit jelöli ki. A REG bitek értékének megfelelő regiszter-hozzárendeléseket a 6-6.táblázat, a MOD és R/M bitek címzési módot meghatározó összerendeléseit a 6-7.táblázat tartalmazza.

6-6.táblázat: *Adatmérettől függő regiszterkijelölés*

REG	32 bites adatok (w=1)	16 bites adatok (w=1)	8 bites (w=0)
000	EAX	AX	AL
001	ECX	CX	CL
010	EDX	DX	DL
011	EBX	BX	BL
100	ESP	SP	AH
101	EBP	BP	CH
110	ESI	SI	DH
111	EDI	DI	BH

A lehetséges címzési módok közül egy ad lehetőséget a közvetlen címzésre (direct addressing). A címzések többségében egy, vagy több regiszter(EBX, EBP, ESI, EDI) tartalmának és az utasításban megadott eltolásnak (displacement=d) az összege adja meg a memória azon helyét, melynek tartalmát a műveletvégzéshez fell kell használni. A MOD mező [MOD]=11 tartalma mellett mindkét operandus valamelyik regiszterben található.

A táblázatban az m[....] jelölés a zárójelben leírt kifejezés értéke, mint **pointer** által kijelölt memóriahely tartalmát jelenti. A [MOD]=01 oszlop

címzési módjaiban 1 byte-os eltolással(d8), a [MOD]=10 oszlop címzési módjaiban 4 byte-os eltolással(d32) kell számolni.

6-7.táblázat: *i80386/486-os processzorok címzési módjai*

R/M	MOD			
	00	01	10	11
000	m[EAX]	m[EAX+d8]	m[EAX+d32]	EAX vagy AL
001	m[ECX]	m[ECX+d8]	m[ECX+d32]	ECX vagy CL
010	m[EDX]	m[EDX+d8]	m[EDX+d32]	EDX vagy DL
011	m[EBX]	m[EBX+d8]	m[EBX+d32]	EBX vagy BL
100	SIB	SIB+d8	SIB+d32	ESP vagy AH
101	d32	m[EBP+d8]	m[EBP+d32]	EBP vagy CH
110	m[ESI]	m[ESI+d8]	m[ESI+d32]	ESI vagy DH
111	m[EDI]	m[EDI+d8]	m[EDI+d32]	EDI vagy BH

Az [R/M]=100 esetekben, a SIB által meghatározott skálázott, indexelt címzési módokat kell használni. Az indexeléshez bármelyik általános célú regiszter felhasználható, amelynek értékét a skála értékkel(1,2,4,8) meg kell szorozni. A tényleges cím értéke a MOD és a BASE mezők, az INDEX mező és az eltolás(offset) értékétől függ. A következő, 6-8.táblázat tartalmazza ezeket az összefüggéseket. A táblázatban 'ix' jelöli a választott, skálázott indexregisztert a 6-9.táblázat szerint

6-8.táblázat: *i80386/486-os processzorok indexelt címzési módjai*

BASE	MOD		
	00	01	10
000	m[EAX+ix]	m[EAX+ix+d8]	m[EAX+ix+d32]
001	m[ECX+ix]	m[ECX+ix+d8]	m[ECX+ix+d32]
010	m[EDX+ix]	m[EDX+ix+d8]	m[EDX+ix+d32]
011	m[EBX+ix]	m[EBX+ix+d8]	m[EBX+ix+d32]
100	m[ESP+ix]	m[ESP+ix+d8]	m[ESP+ix+d32]
101	m[d32+ix]	m[EBP+ix+d8]	m[EBP+ix+d32]
110	m[ESI+ix]	m[ESI+ix+d8]	m[ESI+ix+d32]
111	m[EDI+ix]	m[EDI+ix+d8]	m[EDI+ix+d32]

6-9.táblázat: *Indexelt címzési módhoz használható regiszterek*

INDEX	ix
000	EAX
001	ECX
010	EDX
011	EBX
100	--- (scale=1)
101	EBP
110	ESI
111	EDI

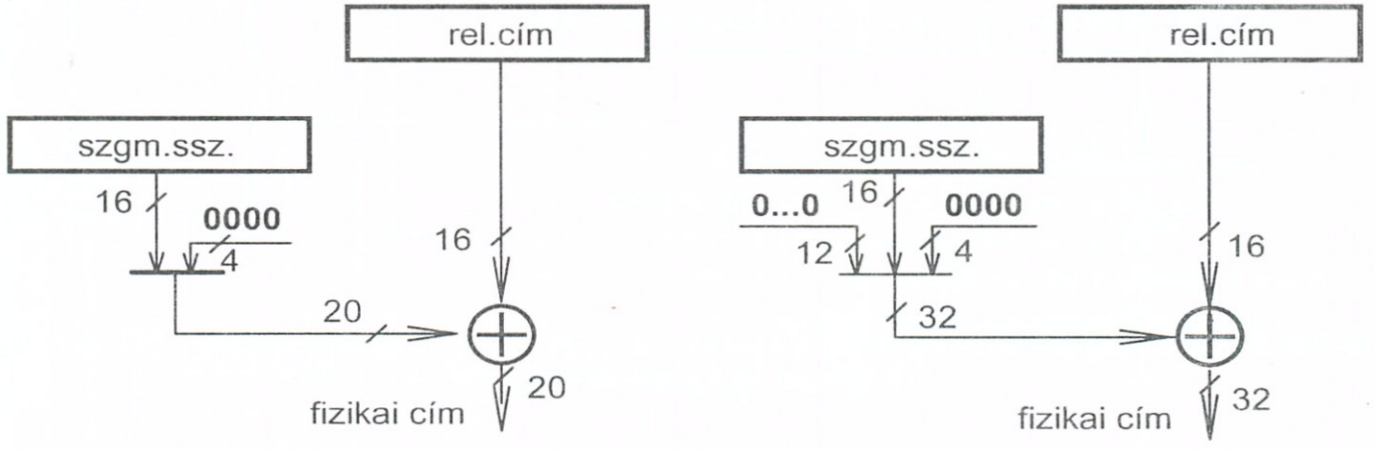
e.)Virtuális címzés

Az i386/486-os processzorok által közvetlenül címezhető memóriatartomány mérete 4GB nagyságú, a virtuális címtartomány mérete pedig 2^{14}. $2^{32}=2^{46}=64$TB lehet.

Valós működési mód

Valós működési mód mellett a processzor úgy viselkedik, mintha egy valódi i8086-os processzor lenne. Ennek megfelelően a maximális fizikai tárméret $2^{20}=1$MB lehet és a legnagyobb szegmens mérete $2^{16}=64$KB.

A valós működési mód melletti címkiszámítási módot mutatja be a 6-8. és a 6-17.ábra rajza. A pontos cím értékét a szegmensregiszterben tárolt érték 4 nullával alulról kiegészitett értékének és az utasításban megadott relatív címnek az összege adja.



6-17.ábra: *i386/486-os processzor címszámítása valós és védett, virtuális 8086-os üzemmódban*

Védett, virtuális működési mód

Ez a működési forma az i8086-os processzornál nem létezik, csak az i286-os processzortól kezdve van ennek alkalmazására lehetőség. Az i386/486-ossal elérhető virtuális tárméret: 64TB; ugyanakkor a fizikailag kezelhető méret 4GB nagyságú.

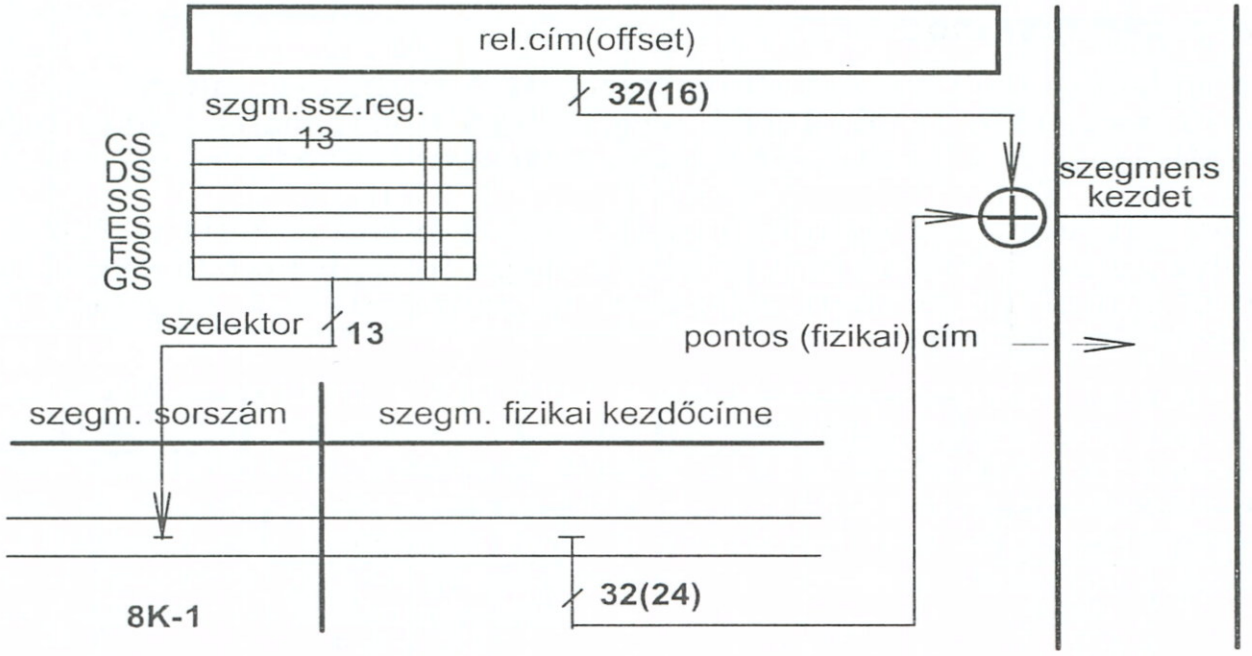
Az i386/486-os processzorok esetében szegmentált és szegmentált-lapozásos tárkezelésre van lehetőség.

Szegmentálás

A szegmentált tárkezelésnél egy-egy szegmens maximális mérete a relatív cím által meghatározott nagyságú, azaz $2^{32}=4$GB méretű lehet. A címkiszámítás módját a 6-18.ábra vázlata mutatja be.

A szegmens kezdőcímét egy szegmens deszkriptor tartalmazza, amely vagy a globális(GDT), vagy a lokális(LDT) deszkriptortáblában található meg. Ki-

keresés után ez a deszkriptor ideiglenes jelleggel, átkerül a szegmens szelektorok mellett található cache-tárba. A szelektor és a szegmensleíró(szegmens deszkriptor) formája a 6-19.ábrán látható.

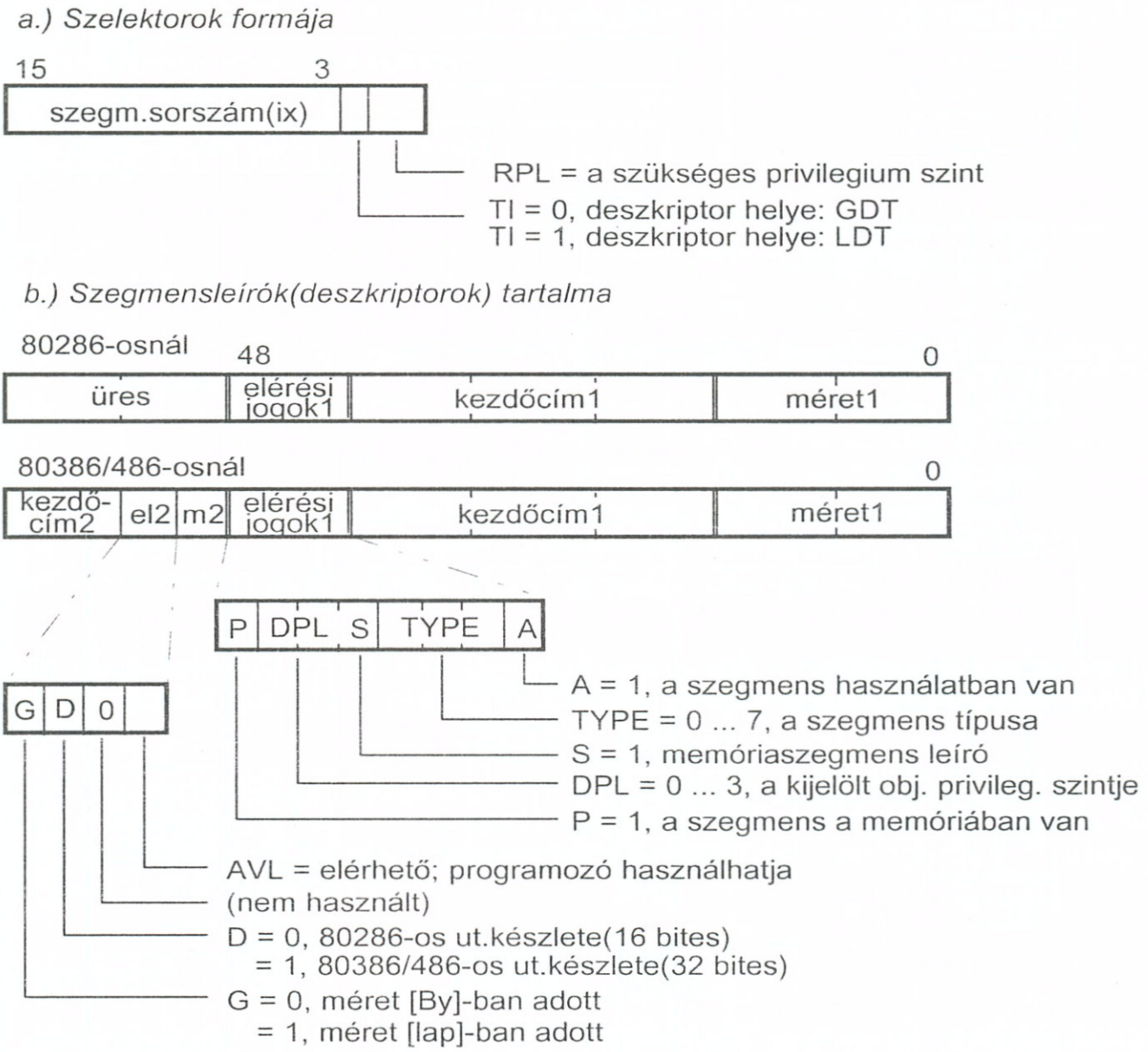


6-18.ábra: *i386/486-os processzorok cím számítása védett, 32 és 16 bites üzemmódban*

A szegmensszelektor három mezőt tartalmaz:

- **INDEX**(13 bit), amely, mint index, a deszkriptor táblában meghatározza a keresett szegmensleíró helyét;
- **TI**(table indicator, 1 bit) bit megadja azt, hogy a szegmensleíró melyik rendszertáblában található meg; ha TI=0, akkor a globális deszkriptortáblában, ha TI=0, akkor a lokális deszkriptortáblában van a szegmensleíró;
- az **RPL**(requestor privilege level, 2 bit) mező megszabja azt a legmagasabb privilégium szintet, amilyen privilégium szintű szegmenst elérhet a szelektor.

A szegmensleíró felső 2 byte-ja 16 bites(i286-os) feldolgozásnál üres, míg 32 bites(i386/486-os) üzemmódban 8 byte hosszúságú. A szegmens kezdőcíme 2 mezőben összesen 32 bit hosszúságú, a szegmens mérete 2 mezőben összesen 20 bit hosszúságú. További bitjei az elérési jogokat részletező mezőket tartalmazzák.

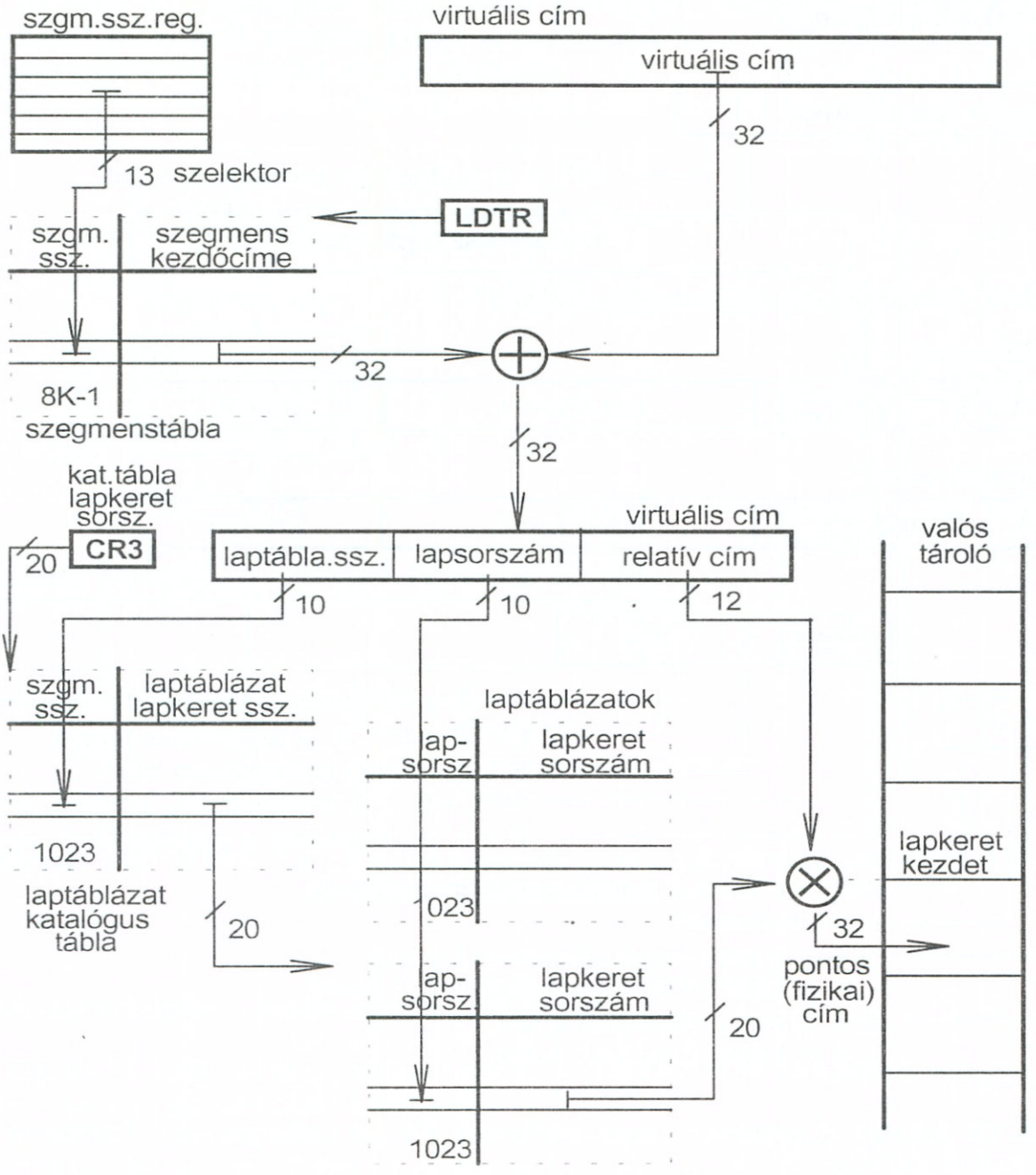


6-19.ábra: *Szegmensszelektorok és deszkriptorok formája*

Lapozás

A virtuális tárkezelés másik formája a lapozás, amelynél a memóriát, a tárolt adatokat azonos méretű egységekre, lapokra bontva kezeli a pro-cesszor. Az alkalmazott lapméret 4 KB. A lapokat a szegmens szelekto-rok által kijelölt szegmensen belül helyezi el a processzor. Ilyen érte-lemben, tulajdonképpen szegmentált és lapozásos memóriakezelésről beszélhetünk.

A lapozásos tárkezelés az i386/486-os processzoroknál kétszintű, (a szeg-menstáblázatot is figyelembe véve háromszintű) laptáblázatokon keresztül megvalósított virtuális tárkezelés, amelynek vázlatát a 6-20.ábra mutatja be.

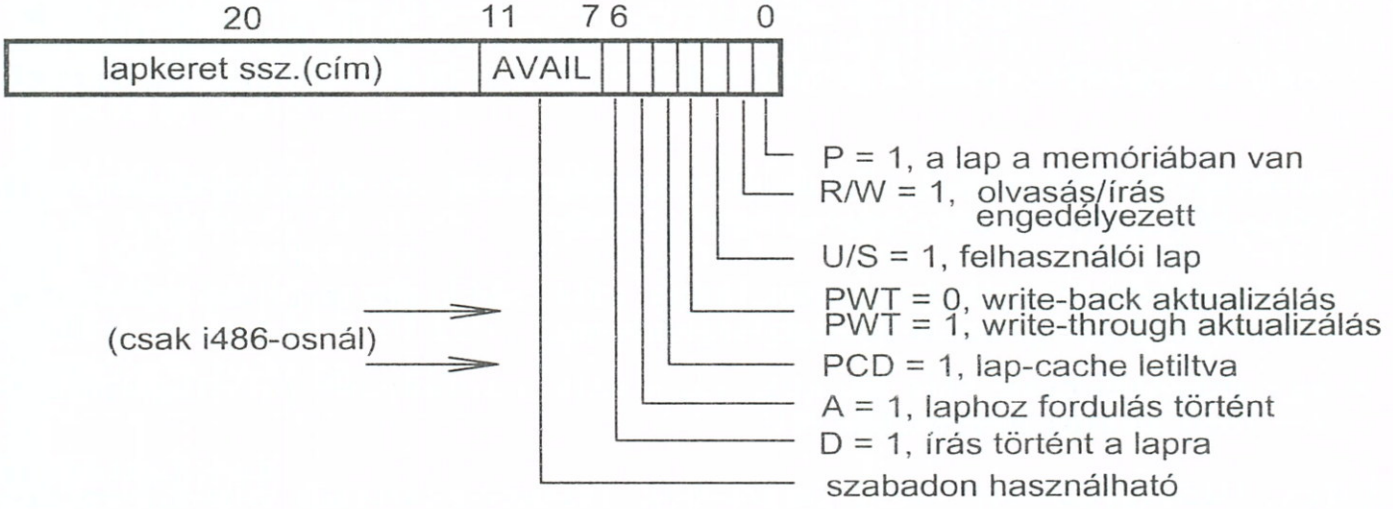


6-20.ábra: *i386/486-os processzorok lapozásos tárolókezelése*

A szegmenstáblázat alapján meghatározott szegmenskezdőcím és az utasításból kiszámítható tényleges cím(effective address) összege képezi azt az új virtuális címet, amelyet a laptáblázatokon keresztül fordít le a processzor a végleges, fizikai címre. A szegmenstáblázat kezdőcímét az LDTR, vagy a GDTR regiszter jelöli ki.

Az első szinten az ú.n. lapkatalógus található, amely az egyes laptáblázatokhoz tartozó leírókat(deszkriptorokat) tartalmazza. A második szinthez tartozó laptáblázatokban találhatók az egyes lapokhoz tartozó lapleírók(lapdeszkriptorok). A **lapdeszkriptorok**ban található az aktuális lap tárolóbeli elhelyezkedését megadó lapkeret sorszám, amely a lapkeret kezdőcímének a felső 20 bitjét adja. Ez az alsó helyiértékeken a relatív címmel egészül ki és

így adja ki a 32 bites pontos címet. A lapdeszkriptor tartalmát a 6-21.ábra mutatja be.



6-21.ábra: *Lapdeszkriptorok formája*

Az aktuális feladathoz tartozó lapkatalógus kezdőcímét(memóriabeli lapkeretének sorszámát) a CR3 regiszter tárolja, amelyet taszkváltáskor elment a rendszer.

A lapozás gyorsítását szolgálja a TLB(Translation Lookaside Buffer) cache, amely a 32 leggyakrabban használt lap adatait(deszkriptorát) fogadja be. A lapozó cache 4-es csoport asszociatív(4-way set associative cache) tár, amelynek a mérete 32x4 byte.

f.)Védelem

A processzorok védett üzemmódja különböző eszközök felhasználásával, többszintű és hatékony védelmet tud nyújtani a multitaszkos feldolgozásokhoz. A védelem eszköze *egyrészt* az elérhető erőforrásokhoz tartozó, az elérési jogosultságot szabályozó védelmi szintek(0-3 között) használata, *másrészt* az erőforrások, mint rendszerobjektumok(pl.: szegmensek, megszakítás-kapuk, táblázatok, stb.) csoportosítása, amelyek igénybevétele csak az ú.n. rendszertáblákon(GDT, LDT) keresztül történhet. Ezek a rendszertáblák az erőforrások legfontosabb adatait, többek között az objektum privilégiumszintjét is tartalmazó deszkriptorokat tárolják.

Deszkriptorral rendelkező rendszerobjektumként a következőket használjuk:

- program- és adatszegmensek,
- lapok,
- taszkok,
- táblázatok(LDT=local descriptor table, TSS=task state segment),
- kapuk(gates; taszk-kapu, megszakítás-kapu, call-kapu, trap-kapu)

Privilégiumok

Az i286/386/486-os processzorok 4-szintű védelmi rendszerrel rendelkeznek. A legmagasabb privilégiumszinttel(PL=0) az operációs rendszer, alacsonyabb(PL=1, PL=2) privilégiummal a különböző kiszolgáló rutinok, I/O rutinok, míg a legalacsonyabb szinttel(PL=3) a felhasználói programok rendelkeznek.

A különböző privilégiumszinteken lévő objektumok között a kapcsolat csak a következő szabályok betartásával jöhet létre:

- egy program csak vele azonos, vagy nála magasabb privilégiumszinttel rendelkező **programot** hívhat, indíthat el;
- egy program csak vele azonos, vagy nála alacsonyabb privilégiumszinten lévő **adatot** használhat fel.

Minden, feldolgozás alatt lévő feladathoz, taszkhoz tartozik valamilyen privilégium szint. Ezt nevezik **aktuális privilégium szint**nek (**CPL**=current privilege level). A CPL értékét a programszegmens szelektorának RPL értéke adja.

A taszkok a táblázatokban elhelyezett deszkriptorokat, illetve rajtuk keresztül a kívánt objektumot, a szelektorok felhasználásával érik el. A szelektorok alsó két bitje meghatározza az azokhoz rendelt privilégium szintet (**RPL**=requestor privilege level). A taszk privilégiumszintje(CPL) és az objektum szelektorának privilégiumszintje(RPL) együttesen határozzák meg a választott objektum elérhetőségét; a CPL és az RPL közül mindig a kisebb (privilégiumú) adja a feladat **tényleges privilégium szint**jét(**EPL**=effective privilege level).

A taszkok az I/O műveleteket csak meghatározott esetekben végezhetik el közvetlenül. Az I/O eszközök elérhetőségét az EFLAGS regiszter **IOPL** (input-output privilege level) mezője szabályozza. Ha CPL>IOPL, azaz az aktuális privilégiumszint alacsonyabb, mint az I/O eszközöké, akkor a taszk közvetlenül nem végezhet I/O műveletet. Az i286-os processzornál ez esetben egy 13-as kivétel(szoftver eredetű megszakítás) keletkezik; az i386/486-os processzoroknál a feladathoz tartozó taszk-szegmens(TSS) input-output bittérképe(I/O permission bitmap) dönti el, hogy 13-as megszakítást kell-e előállítani, vagy a művelet végrehajtható.

A különböző privilégiumszintű programok közötti kapcsolat(interlevel communication) az úgynevezett **'kapu'**-kon(gate) keresztül jöhet csak létre. Ezek deszkriptorai szabályozzák a kívánt privilégium szintet. Ilyen kapcsolat lehet például:

- más privilégium szintű rutin végrehajtása(call) és visszatérés az eredeti feladathoz,
- más feladat végrehajtása(call) és visszatérés az eredetihez,
- áttérés más feladat végrehajtására(jump).

A kapcsolatok létrehozására szolgáló kapuk típusai:

- call kapu, paraméterek átadásával,
- megszakítás kapu, megszakítások kezelésére,
- trap kapu, szoftver eredetű megszakítások kezelésére,
- taszk kapu, amelyen keresztül egy másik feladat feldolgozására lehet áttérni, majd visszalépni.

Taszkváltáskor(pl. taszk kapu igénybevételekor) a regiszterek aktuális tartalmát az operációs rendszer elmenti a taszk állapotszegmensbe (TSS), betölti az új taszk szelektorát a TR regiszterbe, majd ennek alapján betölti a memóriába az új taszk TSS -t és ebből az új regiszter tartalmakat, majd elindítja a taszkot. Call kapu igénybevételekor csak paraméterátadás történik a meghívott rutin számára.

Deszkriptorok

A védelem kialakításához, a multitaszkos feldolgozáshoz az alábbi rendszertáblákat használja a processzor:

GDT(global descriptor table), **globális deszkriptortábla**, amely minden feladat által elérhető táblázat, bármilyen objektumhoz tartozó deszkriptort tartalmazhat, kivéve a kiszolgáló rutinok(megszakítások, trap-ek) deszkriptorait. A globális deszkriptortábla első helye nem használt, az erre mutató deszkriptorokat null-szelektornak nevezik. A globális tábla kezdetének címét és a táblázat méretét a GDTR regiszter tárolja.

LDT(local descriptor table), **lokális deszkriptortábla**, amely egy-egy feladathoz tartozó deszkriptorokat(program- és adatszegmens, call kapu, taszk kapu) tartalmazza. Az operációs rendszer minden feladathoz létrehoz egy LDT-t. Azok a deszkriptorok, amelyek sem a GDT-ben, sem a feladathoz tartozó LDT-ben nem találhatók, azokhoz a taszk nem férhet hozzá. A lokális táblázat kezdőcímét és méretét az LDTR-ben tárolt szelektor alapján keresi ki a rendszer a globális deszkriptortáblából és tárolja az LDTR melletti nem elérhető regiszterben(cache-tárban).

IDT(interrupt descriptor table), **megszakítási deszkriptortábla**, amely védett üzemmódban a megszakítások, kivételek kezelését biztosítja. Az IDT-ben csak megszakítás kapu, trap kapu, taszk kapu deszkriptora lehet. A táblázat kezdőcímét és méretét az IDTR regiszterben tárolja a rendszer.

A multitaszkos feldolgozást segíti a taszkváltásokat biztosító **TSS**(task state segment) **állapotszegmens**, amely a feladathoz kapcsolódó legfontosabb adatokat tartalmazza. Ezek a következők:

- regiszterek tartalma,
- hívó(régi) taszk szelektora,
- a taszkhoz tartozó lokális táblázat szelektora(LDTR tartalma),
- a lapkatalógus címe(CR3 regiszter tartalma),

- I/O bittérkép TSS-en belüli kezdetének mutatója,
- I/O bittérkép(minden egyes I/O eszközhöz egy-egy bit tartozik; 1=tiltott eszköz),
- trap kapcsoló(T=trap on task switch), ha T=1, akkor a taszk indítása után 'debug' kivétel keletkezik.

Az i286-os processzoroknál az I/O bittérkép és annak mutatója hiányzik.

A rendszertáblázatok lehetséges tartalmát, deszkriptorait foglalja össze a 6-10.táblázat.

6-10.táblázat: *Rendszertáblázatok tartalma*

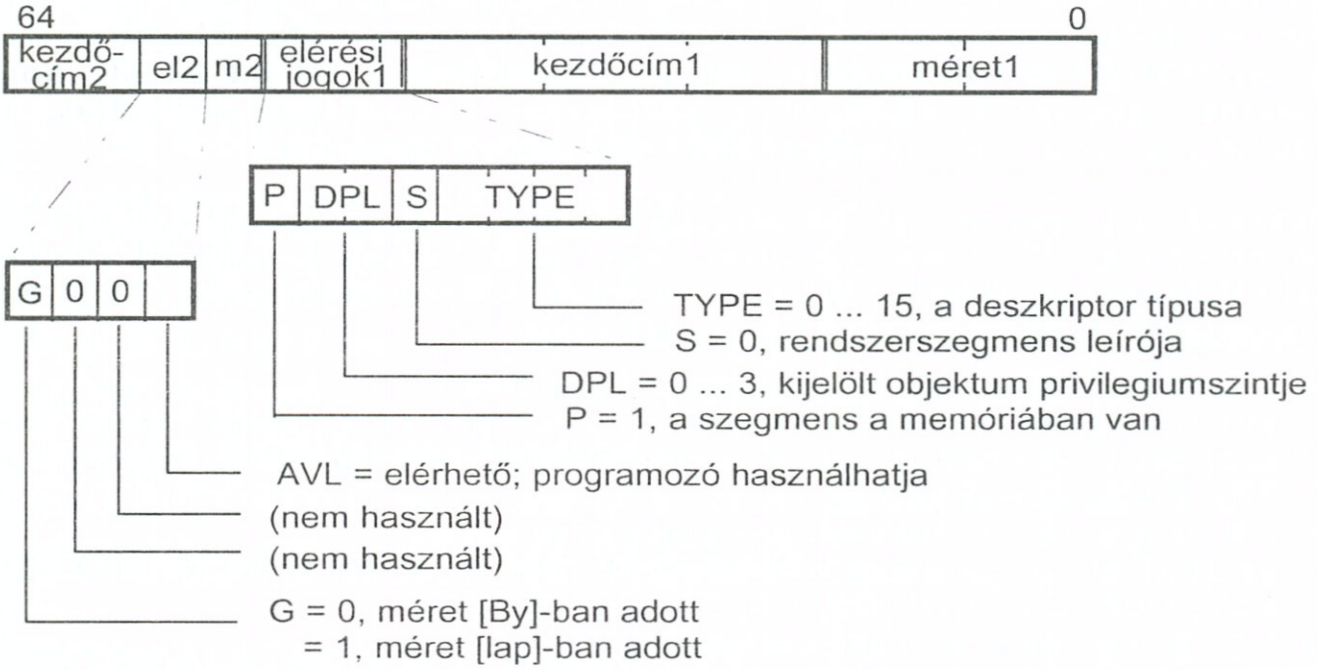
táblázat	GDT	LDT	IDT
kezdőcím helye	GDTR	LDTR	IDTR
szelektorral kijelölt deszkriptor	kódszegmens, adatszegmens, LDT, TSS	kódszegmens, adatszegmens	
kapu deszkriptor	call kapu, taszk kapu	call kapu, taszk kapu	megszakítás, trap kapu, taszk kapu

A táblázatokban használt deszkriptorok közül a program- és adatszegmens deszkriptorát, valamint a lapdeszkriptort már tárgyaltuk. Így azokat most nem mutatjuk be ismételten. A 6-22.ábrán az LDT, a TSS szegmensek, valamint a 'call', a megszakítás és a taszk kapuk deszkriptorait vázoltuk fel. A deszkriptorok elérési jogok nevű részében szereplő TYPE mező típustól függő tartalmát a 6-11.táblázat adja meg. A TYPE mező jobbról 2.bitje, önálló jelöléssel (B=busy) is rendelkezik.

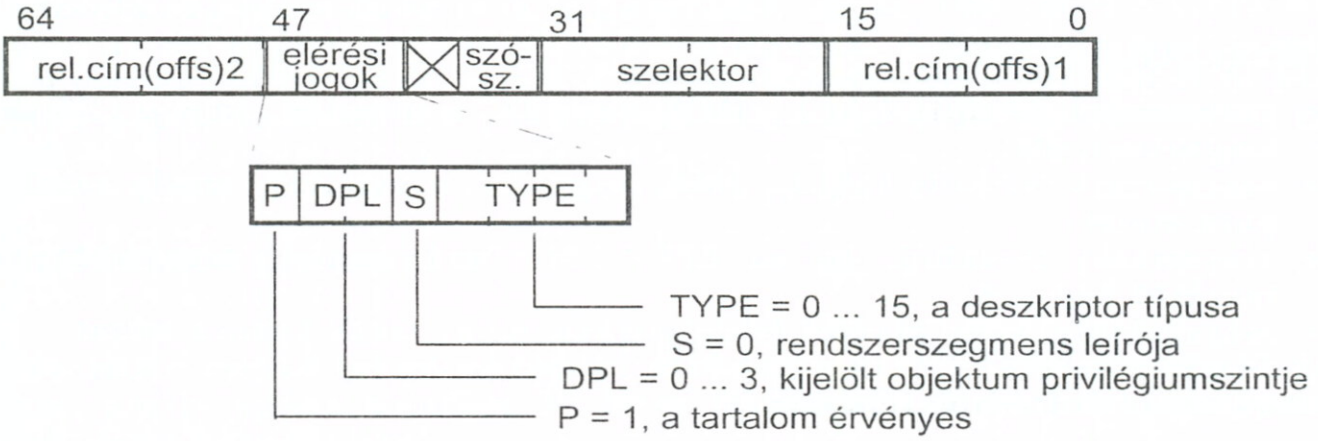
6-11.táblázat: *Deszkriptorok típusai*

TYPE	deszkriptor típusa	TYPE	deszkriptor típusa
0000	(nem használt)	1000	(nem használt)
0001	i286-os TSS	1001	i386/486-os TSS
0010	LDT	1010	(nem használt)
0011	aktív i286-os TSS	1011	aktív i386/486-os TSS
0100	i286-os call kapu	1100	i386/486-os call kapu
0101	taszk kapu	1101	(nem használt)
0110	i286-os megszakítás kapu	1110	i386/486-os megszakítás kapu
0111	i286-os trap kapu	1111	i386/486-os trap kapu

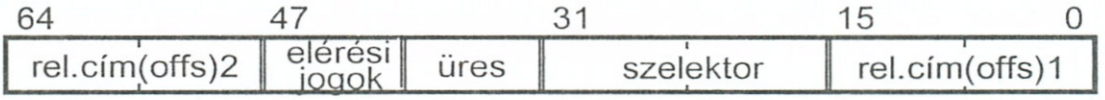
a.) LDT, TSS szegmensleíró



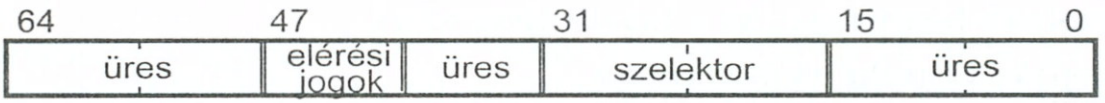
b.) CALL-kapu



c.) Megszakítás- és trap-kapu



d.) Taszk-kapu



6-22.ábra: *LDT, TSS szegmensek és kapuk deszkriptorai*

g.)Multitaszkos feldolgozás

A processzorok által biztosított védelem az alapja a multitaszkos (multiprogramozott) feldolgozásnak. Multitaszkos feldolgozáskor több feladat látszólag egyidejű(valójában időosztásos) végrehajtása történik. Ez azt jelenti, hogy a feladatok egymásnak adják át a processzor használatát(taszkváltás, task switching) az operációs rendszer által megszabott módon.

A taszkváltást szegmensek közötti ugrás(jmp), vagy eljáráshívás(call) eredményezi, ha az TSS deszkriptorra, vagy taszk kapura hivatkozik. Egy-egy taszkváltáskor, az aktuális állapotot a taszk állapotszegmensbe (TSS) menti a processzor. A TSS az alábbiakat tartalmazza:

- a hívó taszk(régi taszk) szelektora(back link),
- minden egyes privilégiumszinthez tartozóan, az aktuális verem (stack) szegmens szelektora és a veremmutató(stack pointer) értéke, azaz SSn:ESPn, ahol n=0,1,2;
- CR3 regiszter: a lapkatalógus tábla kezdőcíme;
- az általános regiszterek tartalma(EIP, EFLAGS, EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, ES, CS, SS, DS, FS, GS);
- LDTR regiszter: az LDT szelektora;
- T(trap on task switch) jelzőbit;
- I/O bittérkép mutatója(i386/486-osnál);
- I/O bittérkép(i386/486-osnál).

A taszkváltás oka lehet:

- szegmensek közötti CALL, JMP, vagy INTn utasítás, amely TSS deszkriptorra mutat,
- szegmensek közötti CALL, JMP, vagy INTn utasítás, amely taszk-kapura mutat,
- visszatérés taszkból, ha az EFLAGS(NT)=1 értékű, azaz beágyazott taszkról van szó,
- megszakítás, vagy kivétel feldolgozása, ha az taszk kapura mutat.

A taszkváltás fontosabb lépései:

- a privilégium ellenőrzése, ha a taszkváltás oka nem megszakítás, kivétel kiszolgálása, vagy más taszkból való visszatérés;
- az aktuális feladathoz tartozó TSS jelenlétének ellenőrzése, a taszk-állapot kimentése;
- új TSS jelenlétének ellenőrzése, az új taszkhoz tartozó regiszter tartalmak betöltése; ha új lapkatalógus(CR3) kerül betöltésre, akkor a TLB kiürítése;
- az állapotjelzők beállítása; ha a taszkváltás oka:
  - JMP utasítás végrehajtása, akkor a
    régi TSS deszkriptorában: B(busy)=0
    új TSS deszkriptorában: B=1;
  - megszakítások feldolgozása, akkor a
    régi TSS deszkriptorában: B=1,
    új TSS deszkriptorában: B=1,

új EFLAGS(I)=0;

- CALL utasítás végrehajtása, akkor a
  régi TSS deszkriptorában: B=1,
  új TSS deszkriptorában:B=1,
  új EFLAGS(NT)=1,
  új TSS(back link) = régi TSS szelektora;
- visszatérés taszkból(IRET), akkor a
  régi TSS deszkriptorában: B=0.

- a taszkváltás jelzése: CR0(TS)=1;
- az új CPL beállítása az új TSS kódszegmensének szelektora alapján;
- LDTR feltöltése;
- szegmens deszkriptorok betöltése;
- ha az új TSS(T)=1, akkor 'debug' kivétel(1-es kivétel) generálása;
- egyébként az új taszk indítása CS:EIP -től.

6.3.MOTOROLA PROCESSZOROK

A Motorola cég processzorainak legfontosabb tagjai, amelyeket röviden áttekintünk, az MC68000, MC68020/30/40-es típusok. A Motorola processzorok kisebb mértékben ismertek, mivel a legnagyobb számban gyártott IBM kompatibilis gépek az Intel cég processzoraival, vagy azok hasonmásaival működnek.

6.3.1.MC68000 processzor

A processzor család tulajdonképpeni első tagja az MC68000-es típusú processzor, amely ugyan belsőleg 32 bites felépítésű, de 16 bites adatbusszal csatlakozik a külvilághoz.

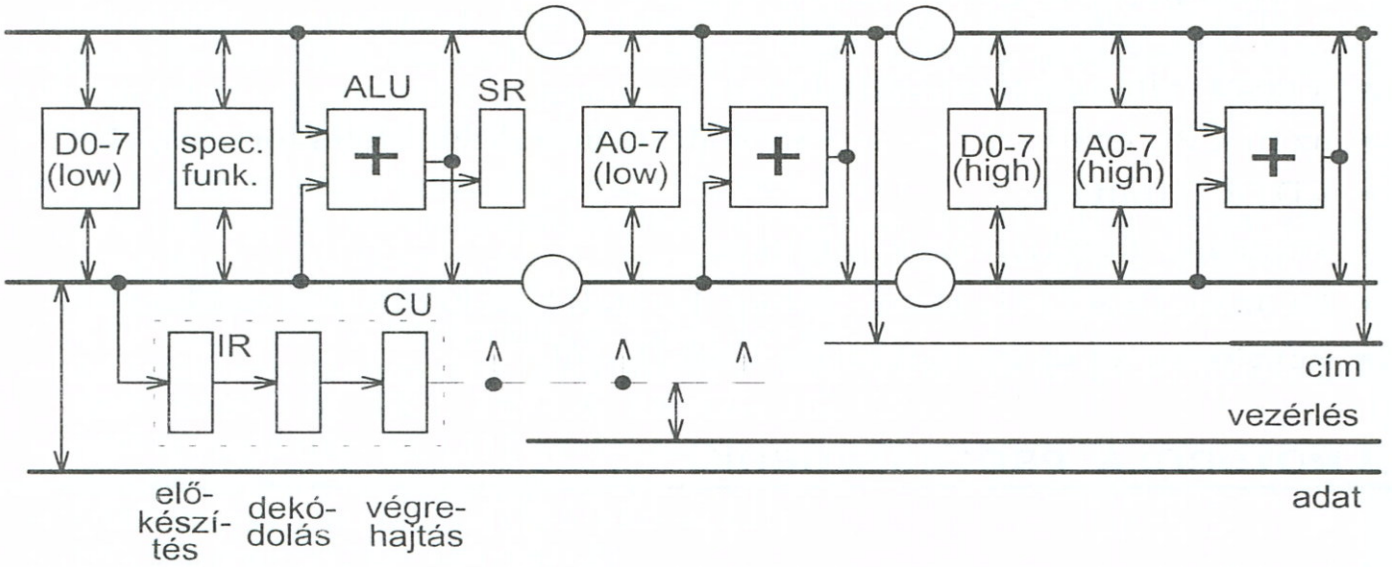
a.)Felépítés

A processzor felépítése, a 6-23.ábra alapján megállapíthatóan, 3 önállóan is működőképes részre bontható. Mindegyik rész egy-egy címsínnel és adatsínnel rendelkezik, amely részek egy-egy kapcsoló egységgel kapcsolhatók össze.

A processzor *egyik részét* az adatregiszterek alsó 16 bitje, az aritmetikai-logikai egység(ALU), az állapot regiszter(SR) és egy speciális funkciójú egység alkotja. Az aritmetikai egység a bemeneteinek egyik részét az adatsínről, másik részét az címsínről kapja, az eredményt bármelyik sínre teheti. A speciális funkciójú egység az ALU kiterjesztésének tekinthető és olyan műveleteket végez el, mint különböző bitmanipulációs műveletek, léptetések.

A *középső rész* foglalja magában a címregiszterek alsó 16 bitjét és a címkiszámításhoz egy összeadót. A kapcsolatok itt is biztosítottak mindkét sín felé.

A *harmadik rész* tartalmazza az adatregiszterek és a címregiszterek felső 16 bitjét és egy összeadót a címkiszámításhoz.



6-23.ábra: *MC68000-es processzor felépítése*

A sínrészek elválaszthatósága lehetőséget teremt az egyes részek független műveletvégzésére. A műveletek vezérlésére szolgáló vezérlő egység (CU=control unit) teljesen mikroprogramozott és helymegtakarítás végett kétszintű tárolást alkalmaznak.

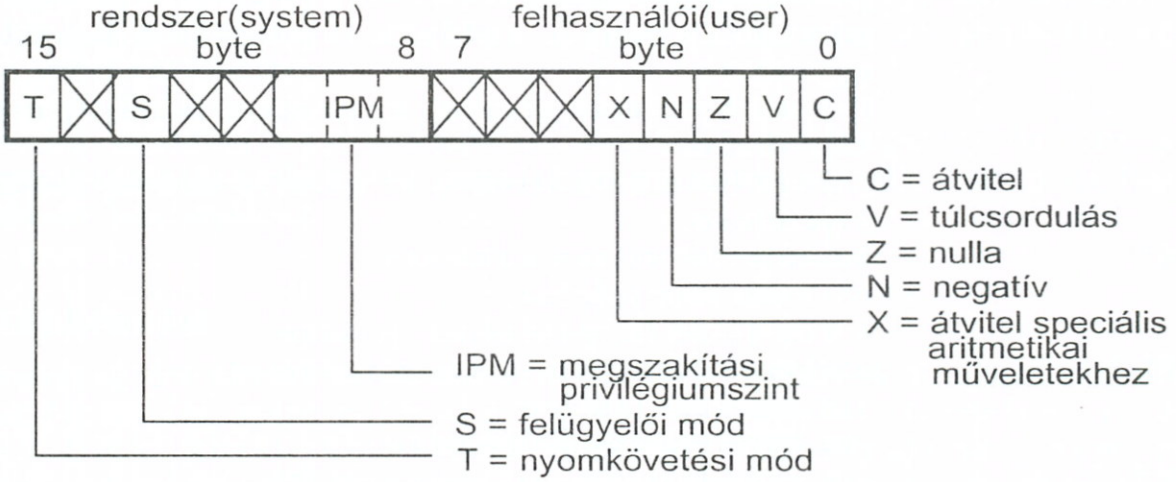
Regiszterek

A processzorban 8 db 32 bites *adatregiszter*(**D0-D7**) van, amelynek mindegyike 16, illetve 8 bites adatot is befogadhat feldolgozásra. Ekkor a regiszter alsó bitjei kerülnek felhasználásra. A 7 db *címregiszter*(**A0-A6**) 32 bites címeket fogad be. Az **A7** regiszter, mint felhasználói veremmutató(user stack pointer), az **A7'** regiszter pedig, mint a felügyelő program veremmutatója (supervisor stack pointer) használatos.

A címregiszterek bármelyike veremmutatóként, vagy báziscímregiszterként is használható. Emellett mind a 17 regiszter használható, mint indexregiszter.

További regiszter még a 32 bites *utasításszámláló regiszter*(**PC**) és a 16 bites **SR** *állapotregiszter*(6-24.ábra).

A regiszter alsó fele az állapotjelzőket tartalmazó rész, amely a *felhasználói byte*(user byte), míg a felső fele azokat a vezérlő biteket foglalja magában, amelyek a *rendszer irányítását* befolyásolják(system byte); ezeket csak az operációs rendszer tudja beállítani.



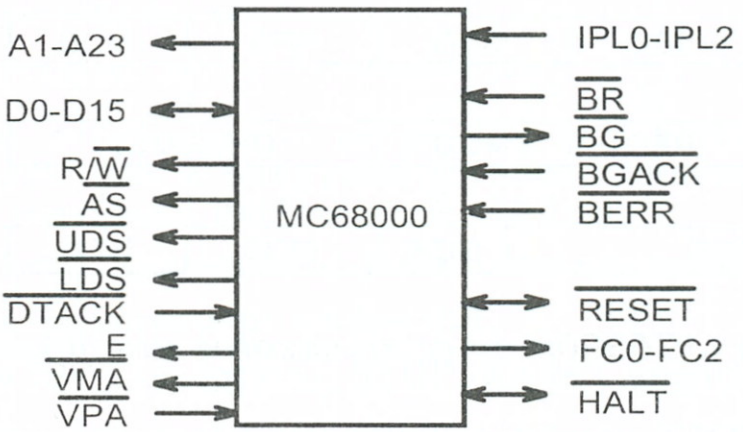
6-24.ábra: *MC68000 processzor állapotregisztere*

A **C**(carry), **V**(overflow), **Z**(zero), **N**(negative) jelzőbitek tartalma már ismert: az átvitel, a túlcsordulás, a nulla eredmény és a negatív eredmény jelzésére szolgálnak. Az **X**(extend) bit néhány aritmetikai, léptető és forgató utasításnál az átvitel bit értékét kapja meg.

Az **I0-I2**(IPM) bitek a megszakítást és a megszakítási prioritást jelzik a processzor részére az 5.2.4.pontban már megismert módon. Az **S**(supervisor/user) bit a processzor működési állapotát(felügyelői/felhasználói), a **T**(trace enable) bit a nyomkövetési állapotot jelzi.

b.)Külső kapcsolatok

Az MC68000-es processzor külső kapcsolatait a 6-25.ábra mutatja be.

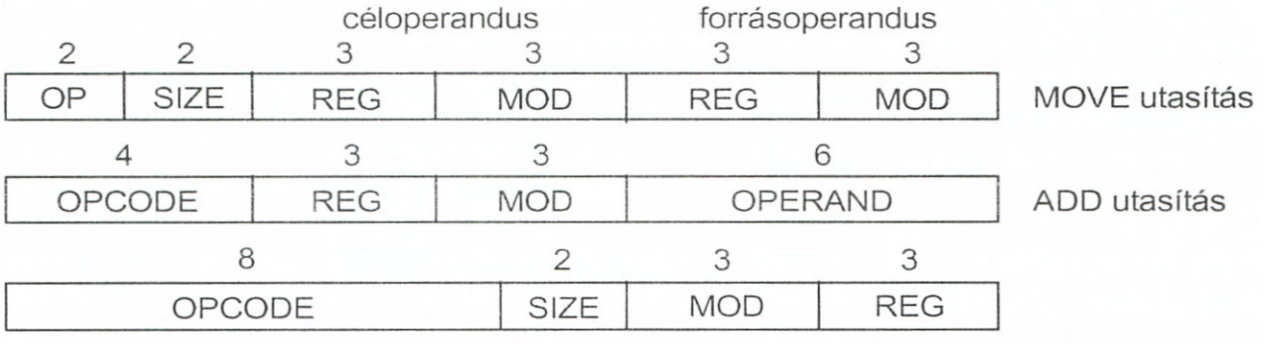


6-25.ábra: *MC68000-es processzor kapcsolatai*

A1-A23	címsín(23 vonal), amely szócímzést tesz lehetővé, így a címezhető tárolótartomány: 2^{23}=8Mszó=16MB;
D0-D15	adatsín(16 vonal), amelyen 1-1 szó(16 bit) vihető át egyszerre;
UDS*	(upper byte data strobe) az adatsín felső byte-jának kijelölése;
LDS*	(lower byte data strobe) az adatsín alsó byte-jának a kijelölése;
DTACK*	adatátvitel visszaigazolása;
R/W*	olvasás/írás jelzése;
AS*	a cím sínre helyezésének jelzése;
IPL0-IPL2	megszakítási prioritásszint jelzése;
BR*	sín kérelem jelzése;
BG*	sín kérelem engedélyezése;
BGACK*	sín engedélyezés visszaigazolása;
FC0-FC2	(function code) a processzor állapotának(pl.: felhasználói adat, program; felügyelői adat, program feldolgozása; megszakítás visszaigazolása) visszajelzésére szolgáló bitek;
RESET*	a processzor alaphelyzetbe hozására; ehhez azonban a HALT* vonalnak is aktívnak kell lenni;
HALT*	a processzor leállítására, lehetőség van a ciklusonkénti utasításvégrehajtásra;
BERR*	(bus error) külső, sínhiba jelzésére;
E, VMA*, VPA*	korábbi processzorváltozatokhoz használt kiegészítő eszközökkel való kompatibilitáshoz szükségesek.

c.)Utasításszerkezet

Az MC68000-es processzornál használatos utasítások hossza 1-5 szó(2-10 byte) között mozog. Az utasítások egy-, vagy kétcímes formátumúak. Jellegzetes változataik a 6-26.ábrán láthatók.



6-26.ábra: *MC68000-es processzorok utasításszerkezete*

Az első utasítás(MOVE utasítás) címmezője két részre osztható, a forrás- és a célhely címére. Mindegyik esetben a '**mode**'(MOD) rész a címzés módját határozza meg, a '**register**'(REG) rész pedig az aktuális regisztert, amely a címkiszámításban résztvesz. Ha ez nem használt, akkor a 'register' mező a

'mode' mező kiegészítéseként alkalmazható. Az utasítás méret(**size**) mezője az átvitelben résztvevő adat méretét(byte, szó, hosszú szó) adja meg. Az utasítás most leírt részét(operation word) követheti 2-8 byte-on az operandusok címe, ha annak meghatározásához a regiszterek nem elegendőek, azaz ha a címzési mód ezt előírja.

A második utasítás egy összeadási utasítás; a harmadik utasítás példa az egycímes utasításszerkezetre, amelyben az egyes mezők funkciója hasonló az előzőekben leírtakéhoz.

Utasításkészlet

A processzor 56 alaputasítással rendelkezik, amelyek a szokásos utasítástípusokat ölelik fel, mint például a logikai, aritmetikai, átviteli és vezérlő utasítások. Ezek közül kiemelhető néhány bitmanipulációs utasítás, több regiszter tartalmának a mozgatása a tároló és a processzor között, szorzási/osztási utasítás, regiszter tartalmának növelésére, csökkentésére szolgáló utasítás, stb. Vannak olyan utasításai, amelyek ú.n. 'gyors' utasítások, amelyek nem igénylik a cím egy-, vagy kétszavas kiterjesztését.

Címzési módok

Az MC68000 processzor esetében mindazok a címzési módok (közvetlen, közvetett, indexelt) használhatóak, amelyekről a 3.1.2.b.pontban már szó esett. Emellett az indirekt címzések között lehetőség van alkalmazni a cím utólagos növelésének(indirect with postincrement), illetve az előzetes csökkentésének (indirect with predecrement) módszerét. Ez a módosítás az utasítás végrehajtása után, illetve előtte következik be. Ezekkel a címzési módokkal könnyen megvalósítható az adatsorok kezelésére az ú.n. FIFO(first-in-first-out) tároló.

d.)Utasításvégrehajtás, műveleti vezérlés

A processzor két üzemmódban dolgozhat:

- felhasználói(user mode) és
- felügyelői(supervisor mode) üzemmódban.

Ennek megfelelően a felhasználói programok nem hajthatnak végre egyes utasításokat, amelyek csak a felügyelői üzemmód esetében hajthatók végre. Ilyen *privilégizált utasítások* például azok, amelyek

- az állapotregiszter 'system byte'-jának tartalmát módosítják,
- a felhasználói veremmutatót módosítják,
- az RTE(return from exception), STOP, RESET utasítások.

Felügyelői módba kapcsol át a processzor külső, vagy belső megszakítások, kivételek feldolgozásakor. Felhasználói módba az RTE utasítás hatására, vagy az állapotregiszter **S** bitjének törlésével vált át a processzor.

Az *utasításvégrehajtás* 'pipelined' formában történik és emiatt mindig három utasítás van sorbaállítva a feldolgozáshoz. Az adatcsatornás feldolgozás három átlapolt fázisa: az utasítás előkészítése, dekódolása és végrehajtása.

A *műveleti vezérlés* megoldására a processzor mikroprogramozott megoldást használ, horizontális vezérlési formával. A mikroprogram tárolásához kétszintű tárolási módot használ a processzor(lásd: 3.3.2.b.pontban), a vezérlő biteket tartalmazó részt a nanotárolóban elhelyezve. A működés gyorsítása végett, a nanoutasítások ugyanazon címen találhatók a tárban, mint a mikroutasítások a mikrotárolóban. A hatékony tárkihasználás végett, a vezérlőbitsorozatok többszörözött tárolásának elkerülése érdekében, speciális hardver megoldással érik el ugyanazon bitsorozat felhasználását különböző helyeken lévő mikroutasításokhoz.

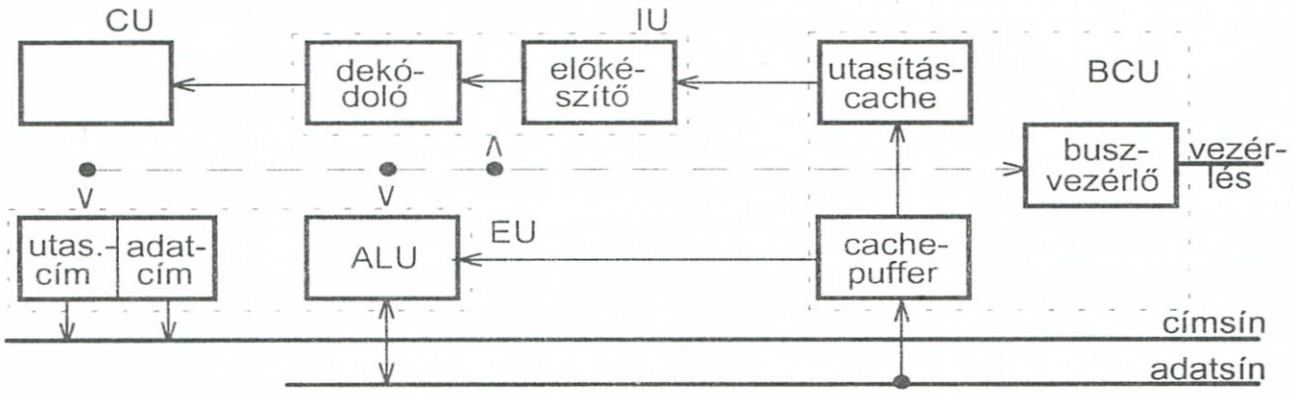
A mikroutasítások 17 bit hosszúságúak, a nanoutasítások 68 bit szélesek. Összesen 544 mikro- és 336 nanoutasítást tartalmaz a mikrovezérlő egység.

6.3.2.MC68020, MC68030/68040 processzorok

A címben felsorolt processzorok mindegyike 32 bites processzor és struktúrájukban nagyjából azonosak. Mindegyikre igaz a 2-3 fokozatú pipelining, a virtuális memóriakezelés, cache-tároló alkalmazása. Lehetőség van koprocesszor igénybevételére is.

a.)Felépítés

Az MC68020-as processzor felépítése, amely a 6-27.ábrán látható, nagymértékben hasonlít a valamivel bővebb MC68030/40-es processzorokéhoz. Az MC68020-as típus a korábbi MC68000-eshez képest annyiban módosult, hogy a külső adatsín is 32 bites, belső utasításcache-tárral rendelkezik és virtuális tárkezelést is lehetővé tesz. Az MC68030-as változat az utasításcache-tár mellett, adatcache-tárral és tárolókezelő egységgel(MMU) is rendelkezik. Az MC68040-es típus a 68030-asnak egy gyorsabb változata.



6-27.ábra: *MC68020-as processzorok felépítése*

A processzor az alábbi fő egységekből áll:

A **sínvezérlő egység**(bus control unit) kapcsolja össze a processzort a külső egységekkel. A sínvezérlő egységhez tartozik szorosan egy 128 szavas(64x32 bites) közvetlen leképzésű utasítás cache, amely a leggyakrabban használt utasításokat tárolja. A sínvezérlőhöz tartozik még a dinamikus sínszélesség-vezérlő is.

Az utasítások feldolgozását egy 4-fokozatú **adatcsatornás egység**(instruction pipeline) végzi, amelynek első lépcsője az utasítástároló regiszter(instruction holding register), amelyet az utasításcache-ből tölt fel a processzor, ha az utasítás ott megtalálható. Amennyiben ez nem áll, akkor a sínvezérlőn keresztül a külső tárolóból kéri be azt. Az utasítás végighalad az utasítás adatcsatornán és a teljesen dekódolt forma kerül a mikrovezérlő egységhez. A vezérlő egység számára a közbülső fázisban tartózkodó kiterjesztések(címzés, közvetlen operandus) is elérhetők, így a végrehajtáskor ezeket rögtön fel tudja használni.

A **végrehajtó egység**(execution unit) három önálló aritmetikai egységgel rendelkezik, amelyek egymással párhuzamosan tudnak dolgozni a címkiszámításon, az adatfeldolgozáson. Az aritmetikai egységek(ALU) egyike alkalmas a logikai műveletek elvégzésére is, valamint a gyors léptetésre, forgatásra is az ú.n. **barrel léptető**(shifter) segítségével.

A **műveleti vezérlő** a már korábban megismert formájú, mikroprogramozott, kétszintű műveleti vezérlő

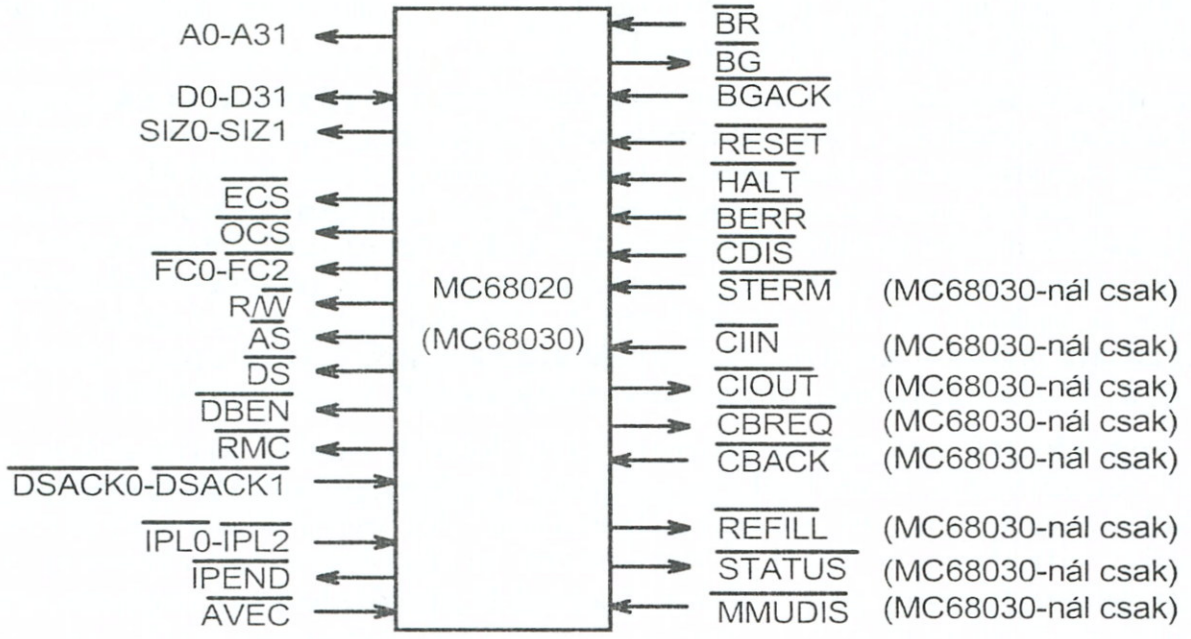
A processzor részét képezik még az adat- és cím**regiszterek**, valamint az utasításszámláló(PC) és állapotregiszter(SR) amelyek ugyanolyanok, mint az MC68000-es processzornál. További regiszter még a *vektor bázisregiszter*(**VBR**=vector base register), amely a kivételek vektortáblájának kezdőcímét tárolja; valamint az egyik átviteli utasítás(MOVES) működését befolyásoló **SFC**(source function code) és **DFC**(destination function code) regiszterek. A felügyelői veremmutató helyett megszakítási veremmutató(ISP) és rendszer veremmutató(MSP=master stack pointer) regiszterek(**A7'**, **A7''**) találhatók a processzorban. A cache-tár kezeléséhez, vezérléséhez használatos a **CAAR**(cache address register) és a **CACR**(cache control register) regiszter.

A processzor legfontosabb jelemzői az alábbiak szerint foglalhatók össze:

- 32 bites címsín(a címezhető tárolótartomány 4GB);
- 32 bites adatsín;
- 3 db, egymással párhuzamosan működni képes 32 bites aritmetikai egység;
- dinamikus sínszélesség-vezérlő, amely 8, 16, 32 bites sínszélesség használatát teszi lehetővé;
- kibővített utasításkészlet;
- kibővített címzési lehetőségek;
- 64x32 bites utasításcache;
- virtuális tárolókezelés;
- kibővített kivételkezelés;
- koprocesszor támogatása.

b.)Külső kapcsolatok

Az MC68020-as, MC68030/40-es processzorok külső kapcsolatait mutatja be a 6-28.ábra.



6-28.ábra: *MC68020-as és MC68030/40-es processzorok kapcsolatai*

A0-A31	címsín(32 vonal),
D0-D31	adatsín(32 vonal),
SIZ0-SIZ1	az átvitt adat mérete(8, 16, 32 bit),
ECS*	külső sinciklus kezdete,
OCS*	adatolvasás, -írás sinciklus kezdete,
FC0*-FC2*	sinciklus típusát meghatározó bitek,
R/W*	olvasási/írási ciklus,
AS*	cím sínre tétele befejeződött,
DS*	adat sínre tétele befejeződött,
DBEN*	adatpuffer engedélyezése,
RMC*	sínfoglalás jelzése,
DSACK0*-1*	olvasás, írás befejezése,
IPL0*-IPL2*	megszakítási prioritási szint beállítása,
IPEND*	megszakítási kérelem elfogadásának jelzése,
AVEC*	megszakítási vektor automatikus módon,
BR*	sínfoglalás kérése,
BG*	sínfoglalás engedélyezése,
BGACK*	sínfoglalás visszaigazolása,
RESET*	processzor alaphelyzetbe hozása(MC68030/40-esnél csak),
HALT*	processzor leállítása(MC68030/40-esnél csak),
BERR*	sínhiba jelzése(MC68030/40-esnél csak),
CDIS*	belső cache letiltása(MC68030/40-esnél csak),
STERM*	szinkron sinciklus vége(MC68030/40-esnél csak),
CIIN*	cache vezérlő jel(MC68030/40-esnél csak),
CIOUT*	cache vezérlő jel(MC68030/40-esnél csak),

CBREQ*	cache vezérlő jel(MC68030/40-esnél csak),
CBACK*	cache vezérlő jel(MC68030/40-esnél csak),
REFILL*	MC68020-as emulálása(MC68030/40-esnél csak),
STATUS*	MC68020-as emulálása(MC68030/40-esnél csak),
MMUDIS*	MC68020-as emulálása(MC68030/40-esnél csak).

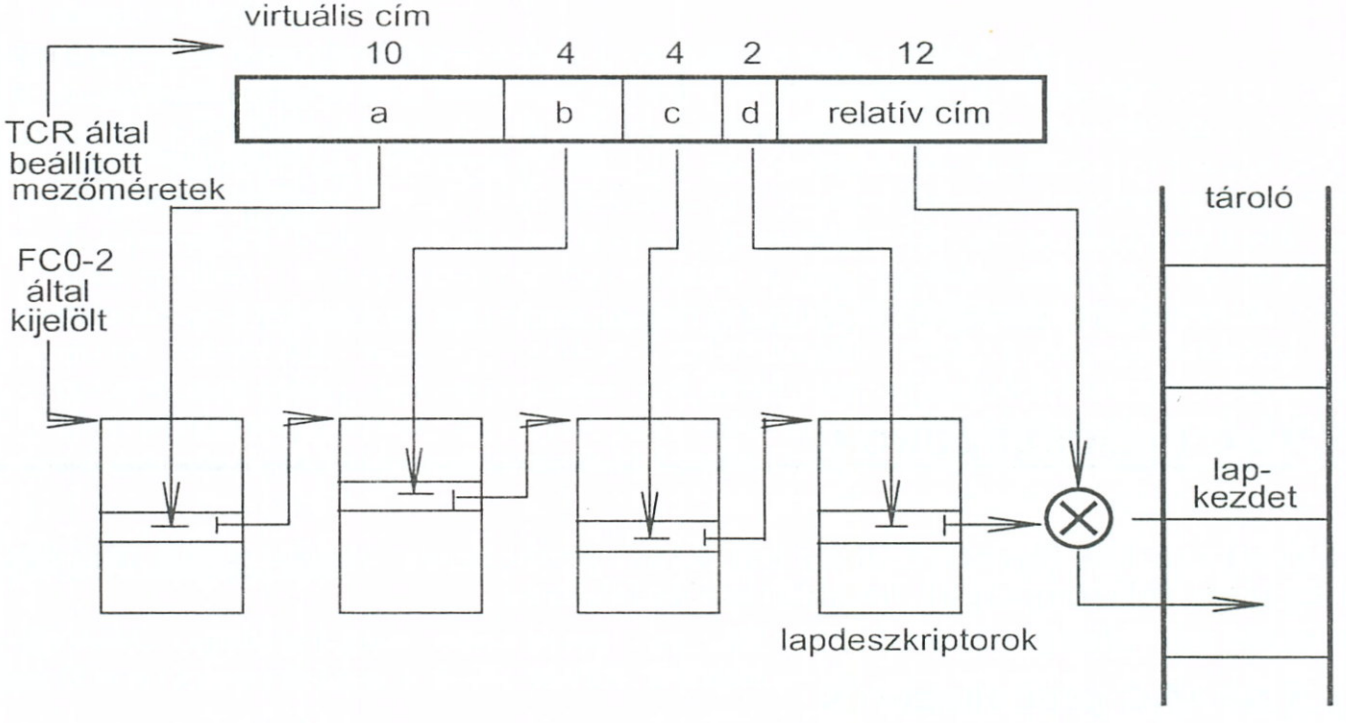
c.)Utasításszerkezet

Az MC68020-as, MC68030/40-es processzorok utasításformái ugyanazok, mint a 68000-es processzoré, csak az utasításkészlet bővült, valamint a címzési formák száma növekedett. Így memória indirekt utasítás, skálázott indexelt utasítás is alkalmazható.

A különböző címzési lehetőségeket a 3.1.2.b.pontban foglaltuk össze a Motorola processzorokra vonatkozóan is.

d.)Virtuális címzés

A virtuális címzés támogatása az MC68020-as processzornál külső, az MC68030/40-es processzoroknál már beépítetett tárolókezelő egységgel (MMU) történik.



6-29.ábra: *MC680x0 processzorok virtuális cimzési módja*

A Motorola processzorok nem használnak társzegmentálást, csak a lapozási technikát alkalmazzák(6-29.ábra). Így, a virtuális cím közvetlenül a címkiszámítás végén kapott 32 bites tényleges cím(effective address), amit egy 0-4 szintű lapozási eljárás során alakít át a processzor a végleges, fizikai

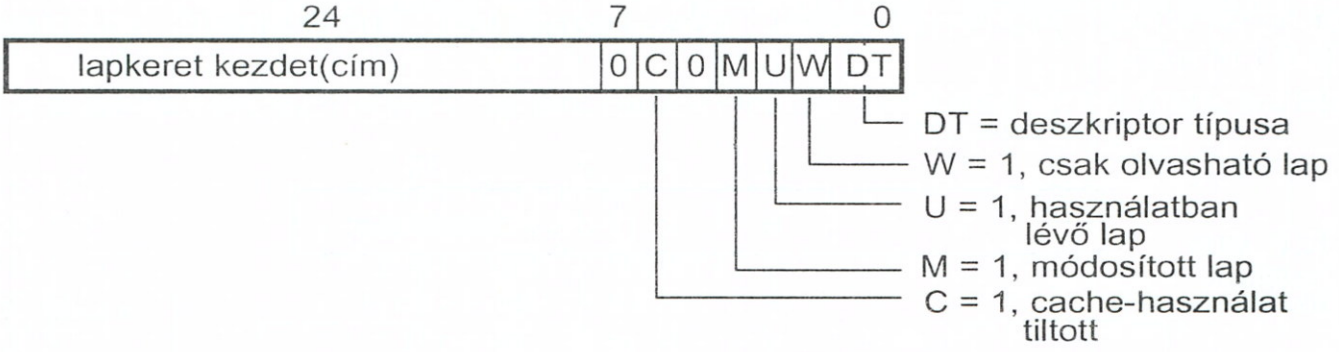
címmé. A többszintű lapozás előnye a multitaszkos feldolgozásnál jelentkezik, abban, hogy ugyanazt a lapot meg lehet osztani több taszk között is.

A szintek száma programból választható és a virtuális cím felosztása az egyes szintek között a **TCR**(Translation Control Register) **regiszter** tartalma alapján történik, amelyet az operációs rendszer tud beállítani. Lehetőség van a virtuális cím megadott számú felső bitjének figyelmen kívül hagyására is.

A lapok mérete az operációs rendszer által, 256 byte és 32 KB között szabályozható. A laptáblázatok legvégén lévő táblázatban, egy lapdeszkriptor (6-30.ábra) határozza meg a virtuális címnek megfelelő lapot.

A lapdeszkriptor a felső 24 bitjén tartalmazza a lapkeret kezdőcímét, amelyhez hozzá kell adni az utasításból származó eltolást. Mivel a lapméret 256 byte-tól 32 KB-ig változhat, ezért a lapkeret címének mérete is változik, a nagyobb lapoknál rövidebb lesz. Ilyenkor a 24 bites címrész alsó bitjei nem használatosak.

A lapozást egy kisméretű, 22 lap adatait befogadó cache tár(look-up table) segíti.



6-30.ábra: *MC680x0 processzor lapdeszkriptora*

6.4.RISC PROCESSZOROK

A következő két fejezetrészben két RISC processzort ismertetünk röviden, a jellegzetes tulajdonságok bemutatása érdekében.

6.4.1.SPARC processzorok

A SPARC(Scalable Processor Architecture) architektúrát 1987-ben ismertette a Sun Microsystems Inc., amely tulajdonképpen nem egy processzor, hanem egy definició, amelyben a tervezők csak azt határozták meg, hogy hogyan nézzen ki a processzor assembly szintről nézve. A konkrét megvalósításra nem adtak előírást. Tehát fizikai szinten, többféle megvalósítási formája is

létezhet ugyanannak a SPARC architektúrának. A processzor mellett, egy tárolókezelő egység(MMU) struktúráját is meghatározták.

A SPARC architektúra alapjait, ősét a *Berkeley University* által kifejlesztett RISC I, RISC II, SOAR processzorok alkották.

a.)Felépítés

Az architektúra definiálása nem kizárólag csak a processzorra vonatkozik, hanem amellett egy lebegőpontos egységet(FPU=floating point unit) és egy koprocesszort(CP=coprocessor) is meghatároztak, amelyek egymáshoz kapcsolhatók.

A SPARC struktúra legfontosabb jellemzői a következők:

- 32 bites adat- és címsín,
- lapozásos tárkezelés, 4GB-nyi, byte-onként címezhető tártartománnyal,
- 32 bites szóhosszúság,
- a memória szervezése címzés szempontjából ú.n. 'big endian' típusú, ami azt jelenti, hogy a 32 bites szó legmagasabb helyiértékű byte-jához van hozzárendelve a 0-dik sorszám, majd az 1-es, 2-tes, 3-as, ugyanúgy, mint ahogy a Motorola MC680x0-ás típusú processzorok kezelik a memóriát;
- a memória írása, olvasása 8, 16, 32, 64 bites egységekben történhet csak,
- multiprocesszoros környezet kialakításának lehetősége,
- nagyfokú adatcsatornás(pipelining) feldolgozás; a csatorna egyes fokozatait az előkészítés, a dekódolás, a végrehajtás és a visszaírás alkotják;
- nagy méretű regisztertár(104-520 regiszterből), amelyet ablaktechnika (register windowing) alkalmazásával használnak(a regisztertárakról a 4.2.pontban volt szó részletesen); összesen 32 ablak használható egymás mellett; a regisztertár beteltekor a legrégebbi ablakhoz tartozó tartalmakat kimenti a rendszer a főtárba; a **WIM**(window invalid mask) mező bitenként egy-egy ablak foglaltságát jelzi; a regisztertárral való takarékoskodás, optimalizálás céljából, az eljáráshívási lánc legvégén álló eljárásnál(tehát, amelyik már nem hív újabb eljárást), csak a kimenő adatokat tároló regisztereket használják.

b.)Utasításkészlet

A SPARC architektúra utasításai fix, 32 bit hosszúságúak. Az összesen négyféle utasításforma a 6-31.ábrán látható.

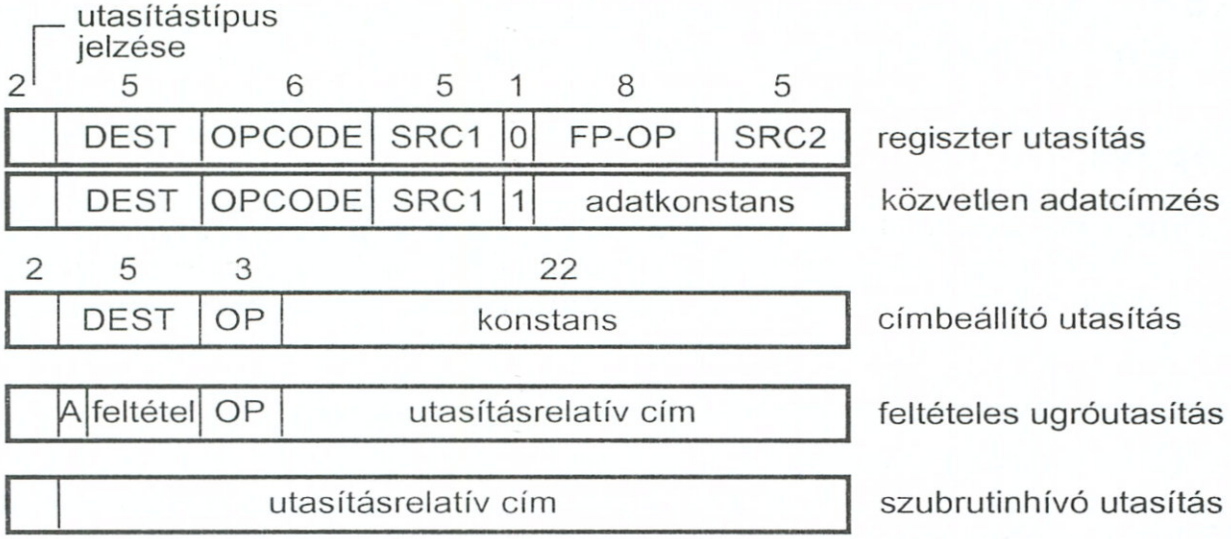
A LOAD/STORE utasítások az első két utasításváltozatot használják. Ezekben az esetekben a memóriacímet az utasítás alsó 19 bitje határozza meg oly módon, hogy:

memória cím = [SRC1] + [SRC2],

vagy

memória cím = [SRC1] + előjeles adatkonstans.

Közvetlen memóriacímzés nincsen, az csak regiszter-indirekt címzési móddal oldható meg. Az aritmetikai utasítások 3-címesek.



6-31.ábra: *SPARC processzorok utasításszerkezete*

A SPARC főprocesszorok nem rendelkeznek szorzási, osztási utasítással, ezt a feladatot, vagy a lebegőpontos processzor(FPU), vagy szoftver segítségével oldja meg a rendszer.

A lebegőpontos műveletek külön koprocesszorral végezhetők el. A lebegőpontos aritmetika szabályai követik az IEEE lebegőpontos szabvány előírásait. Az adatok a memória közvetítésével kerülnek a processzorból a koprocesszorba, vagy vissza.

A processzor csak utasításrelatív(PC-hez viszonyított) ugró utasítással rendelkezik, ±8MB távolság megadási lehetőségével.

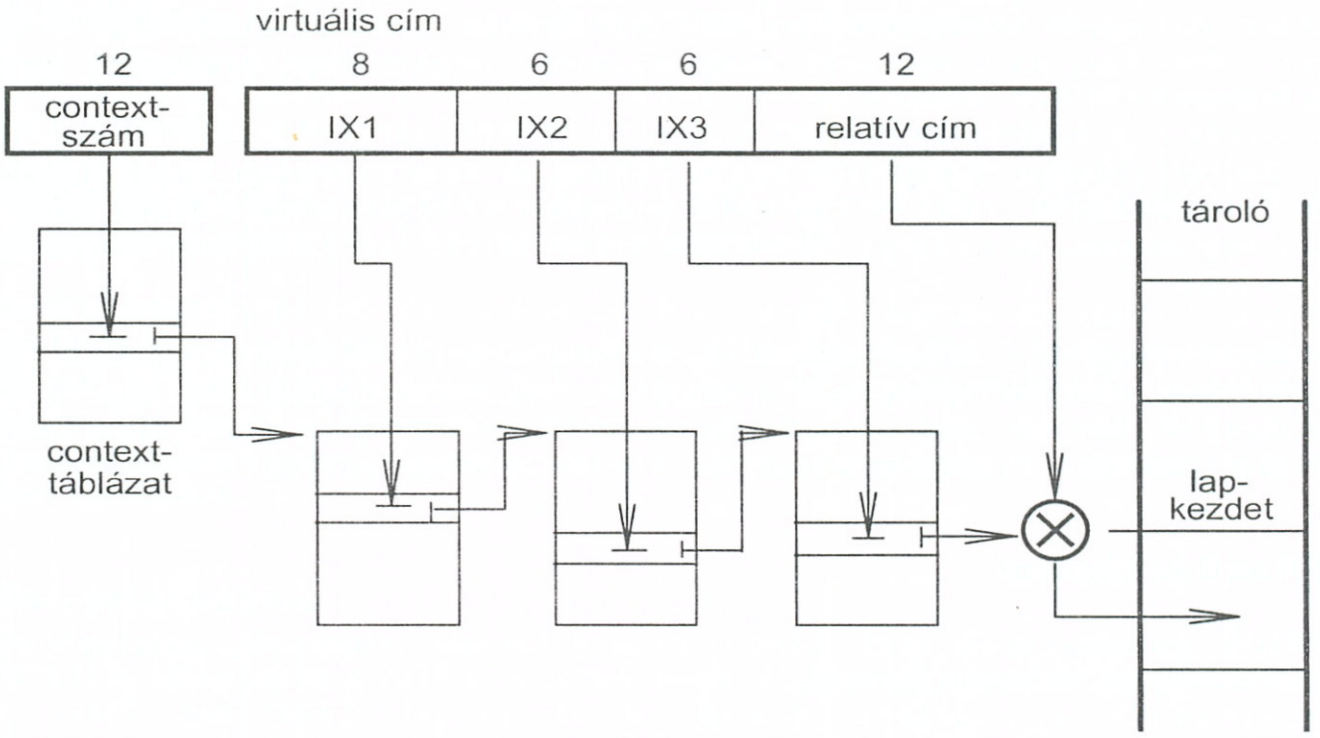
c.)Memóriakezelés

A SPARC processzorok 3-szintű lapozásos tárolókezelést alkalmaznak. A processzoron futó folyamatok mindegyike egy, 1-4096 közé eső számmal (context) rendelkezik, amely a context-tábla alapján kijelöli a folyamathoz tartozó laptáblázatokat. A tárolócímzés vázlata a 6-32.ábrán látható.

A memóriakezelést gyorsítja egy kereső táblázat(look-up table), valamint az a tény, hogy a virtuális cím egyszerre kerül a tárolókezelő egységbe(MMU) és a cache-tárba.

d.)Megszakítások

A megszakítások és a kivételek kezeléséhez szükséges, hogy a következő ablak mindig szabad legyen, mivel a megszakítás kiszolgálásához a jelenlegi állapotot el kell menteni.



6-32.ábra: *SPARC processzorok virtuális címzési módja*

6.4.2.MIPS processzorok

A MIPS(Microprocessor without Interlocked Pipeline Stages) Computer Systems R2000-es, R3000-es, R4x00-es processzorainak eredete visszanyúlik a *Stanford University*-n végzett fejlesztésekhez.

A MIPS processzorok 32 db 32 bites regiszterrel rendelkeznek, 32 bites szóhosszúsággal dolgoznak. A címezhető tárolótartomány 4 GB nagyságú, amelyből 2 MB az operációs rendszernek van fenntartva. Virtuális memóriakezelés biztosítja a tároló jó kihasználását. Míg a SPARC processzor társzervezése 'big endian' típusú, addig a MIPS processzorok mind 'big endian', mind 'little endian' típusú megoldást megengednek, méghozzá programból választhatóan.

a.)Felépítés

A processzorban nincs nagy méretű regisztertár, így annak helyén el lehetett helyezni egy tárolókezelő egységet, cache-tárat a vezérlőjével együtt.

A processzorban 32 db 32 bites általános célú regiszter van, amelyből az első, az R0-s regiszter tartalma hardverileg 0 értékűre van beállítva. Az egyes regiszterek tartalma, az általános célú felhasználhatóság ellenére, általában mindig azonos jellegű.

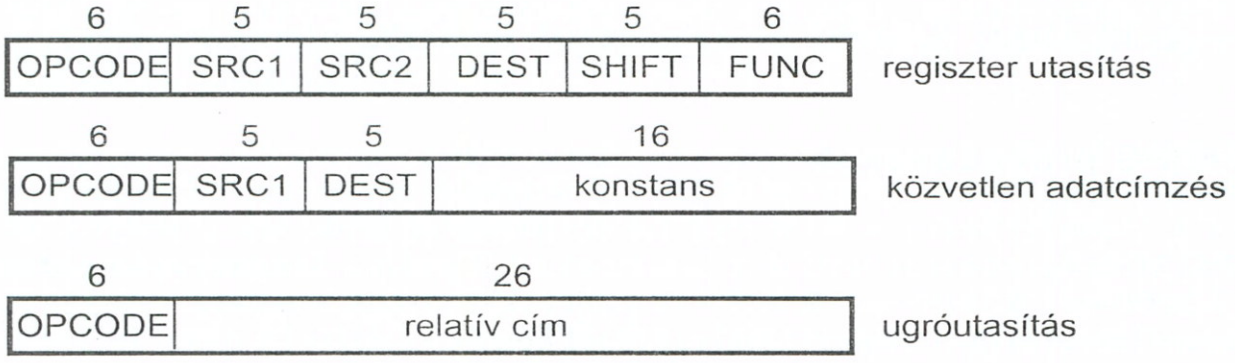
Az utasítások feldolgozására egy 5-fokozatú adatcsatornát használ a processzor. Az egyes fokozatok: utasítás előkészítés, utasítás dekódolás és operandus olvasása, ALU művelet, adatvisszaírás megkezdése, eredmény tárolása a helyén.

Az utasításfeldolgozó pipeline egyes fokozatai nem szorosan kapcsoltak(not fully interlocked); azaz, ha egy töltő utasítást követő utasítás előbb akar olvasni egy adatot, mint hogy az be lett volna töltve, az eredmény meghatározatlan lesz.

Az adatok megfelelő elhelyezéséhez, a tárolóhoz fordulások számának csökkentéséhez, a fordítóprogram elemzi az eljáráshívásokat és egy eljárás hívása előtt, csak azokat a regiszter tartalmakat menti el, amelyekre később szükség lehet és a regiszter tartalmát az eljárás felhasználná.

b.)Utasításkészlet

A MIPS processzorok csak három utasításformát ismernek, amelyeknek a szerkezete a 6-33.ábrán látható.



6-33.ábra: *MIPS processzorok utasításszerkezete*

Az első utasítás egy 3-címes szerkezet, a második pedig egy 2-címes, adatkonstanssal kiegészített utasítás. A konstans egy 16 bites előjeles szám lehet. A harmadik utasítás ugró utasítás, amely egy 256 MB nagyságú lapon belül enged meg ugrást.

A memória utasítások(LOAD/STORE) tulajdonképpen egyetlen címzési módja a relatív címzési mód. Egyes értékek speciális volta mellett lehet másfajta címzést is megvalósítani. Így a következők lehetségesek:

regiszter relatív	[REG] + konstans,
regiszter indirekt	[REG] + 0,
közvetlen	[R0] + konstans.

A 32 bites cím előállítása két lépésben oldható meg egy speciális utasítás segítségével.

A processzornak nincs utasítása dupla szó betöltésére, ugyanakkor nem csak szóhatártól(4-gyel osztható címtől) tud betölteni szavakat.

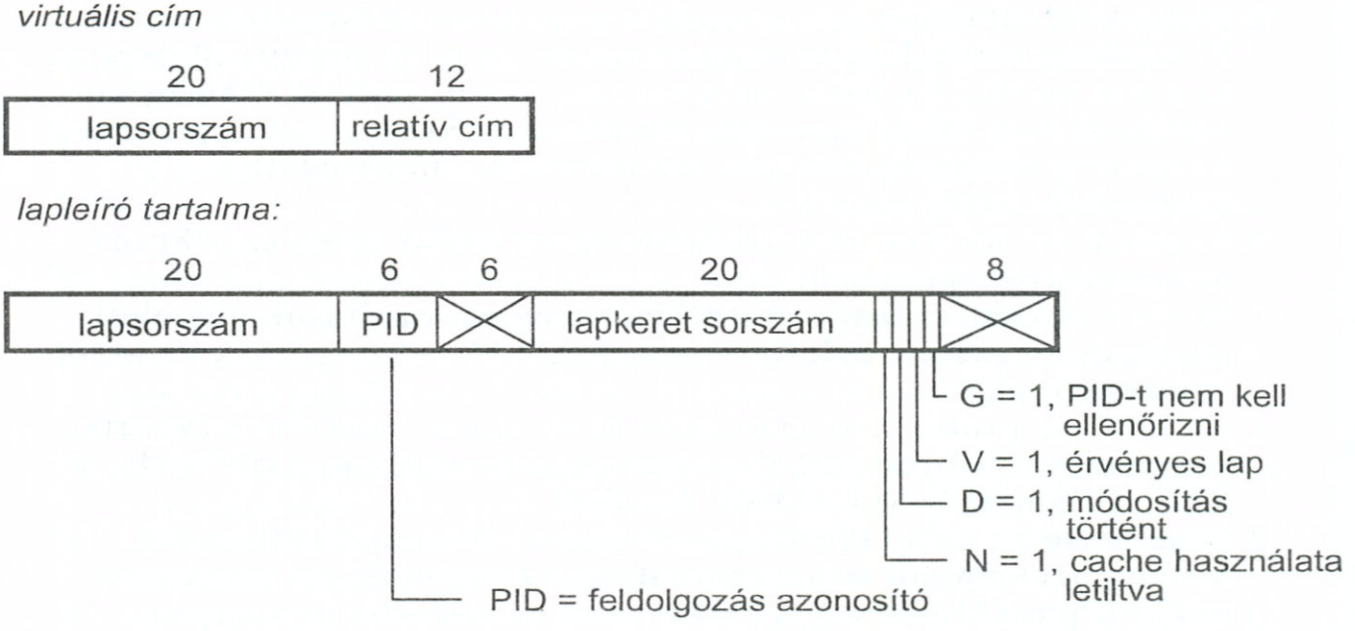
Az aritmetikai utasítások között, a RISC processzorok esetében szokatlan módon, szorzási és osztási utasítás is van. A logikai utasítások a szokásosak.

A lebegőpontos műveleteket külön koprocesszorral lehet megoldani. A koprocesszor regisztereibe közvetlenül átírható adat.

A vezérlő utasítások között, a feltétel bit vizsgálatára nincs utasítás, aminek a feladatát két lépésben(összehasonlítás, ellenőrzés) oldja meg a processzor. A feltétel nélküli ugró utasításnál, csak 256 MB-os lapon belüli ugrás lehetséges. A feltételes elágazáshoz összesen 6 utasítás ($=0$, $\neq 0$, <0, >0, ≤ 0,≥ 0) áll rendelkezésre. Az eljárások hívása közvetlen, vagy regiszter-indirekt módon történhet.

c.)Memóriakezelés

A virtuális tárkezelés a MIPS processzoroknál kissé szokatlan módon megoldott, mert nincsenek laptáblázatok, hanem ehelyett, a deszkriptorok tárolására egy 64 szavas TLB(Translation Lookaside Buffer) tárat használ a rendszer. A TLB-ben egy-egy lapnak a deszkriptora található, amely tartalmazza a virtuális lap fizikai lapkeretcímét,amelyet a virtuális lap címének a helyére helyettesít be(6-34.ábra).



6-34.ábra: *MIPS processzorok virtuális címzése*

Ha a lapleíró nincs a TLB-ben, akkor megszakítás(trap) következik be, amelyet az operációs rendszer dolgoz fel és végzi el a címkonverziót. A TLB tartalmának a cseréje véletlenszerű kiválasztással történik.

Multiprogramozott környezetben, a lapok programhoz rendelt felhasználását a lapleíróban található **PID**(Process ID) feladatazonosító ellenőrzésével lehet biztosítani.

6.5.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Mi jellemzi az i8086/8088-as processzorokat? Hány bites címsíne van az i8086/8088-as processzoroknak? Miben különbözik egymástól a két processzortípus?
2. Melyek az i80186/i80188-as processzorok legfontosabb jellemzői?
3. Melyek az i80286-os processzorok legfontosabb jellemzői? Miben tér el az i80286-os processzor az i8086-os processzortól?
4. Milyen üzemmódjai vannak az i286-os processzornak? Mire szolgál a valós üzemmód?
5. Vázolja fel az i8086/i80286-os processzorok felépítésének rajzát! Milyen fő egységei vannak az i80286-os proceszszornak? Mi az egységek feladata?
6. Milyen regiszterei vannak az i286-os processzornak és melyik mire szolgál?
7. Melyek a regiszterek közül az ú.n. rendszerregiszterek és mire szolgálnak? Mely regiszterek használtak a címkiszámításban? Mire szolgálnak a szegmensregiszterek?
8. Melyek az állapotjelző és vezérlő regiszterek? Ismertesse a FLAGS regiszter legfontosabb vezérlő és jelző bitjeit! Melyik az a két mező a FLAGS regiszterben, amely csak az i286-osban található meg. Milyen mezői vannak az MSW állapotregiszternek? Melyik regiszterben van a védett üzemmódot jelző mező?
9. Rajzolja fel milyen részekből állnak az i8086-os és az i286-os processzorok utasításai.
10. Milyen információkat tartalmazhat még az utasítások műveleti jelrészét magában foglaló utasításrész? Mire szolgálnak az utasítás prefixek?
11. Az i286-os processzor utasításai hány címes utasításszerkezetet képviselnek? Mire szolgál a címmeghatározásban a MODE byte REG, valamint R/M és MOD mezője?
12. Ismertesse az i386/486-os processzorok általános jellemzőit! Milyen üzemmódokat használhatnak az i386/486-os processzorok? Miben különbözik az i386-os és az i486-os processzor egymástól?
13. Vázolja fel az i386/486-os processzorok felépítésének rajzát! Milyen részei vannak az i386/486-os processzornak? Hány bites címsínnel, illetve adatsínnel rendelkeznek a processzorok? Mi az egyes egységek feladata?
14. Hogyan működik az i386/486-os processzor valós, illetve védett üzemmódban? Milyen védett üzemmódjai vannak az i386/486-os processzoroknak?

15. Milyen regiszterei vannak az i386/486-os processzoroknak? Hány szegmensregiszterrel rendelkezik az i386/486-os processzor?
16. Mutassa be az EFLAGS regiszter jelzőbitjeit és azok funkcióit! Mire használtak a vezérlő regiszterek(CR0, CR1, CR2, CR3)?
17. Melyik vezérlő regiszter tárolja a lapkatalógus kezdőcímét? Mire szolgál a TLB?
18. Milyen regiszterek segítik az i386/486-os processzorok nyomkövetési, hibakeresési feladatait? Mire szolgálnak a tesztregiszterek?
19. Ismertesse az i386/486-os processzorok utasításszerkezetét. Mire szolgál az utasítás MODE és SIB byte-ja? Mire szolgálnak az utasítás-prefixek? Milyen utasításcsoportokat ismer?
20. Milyen címzési módokat ismer az i386/486-os processzorok esetében? Mire szolgál a címmeghatározásban a MODE byte REG és R/M és MOD mezője?
21. Mire szolgál a címmeghatározásban a SIB byte INDEX és BASE mezője?
22. Milyen címzési mód használható az i386/486-os proceszszorok valós üzemmódjában? Valós üzemmódban, mekkora lehet a szegmens maximális mérete és a fizikai címtartomány?
23. Milyen címleképzési eljárásokat használnak a processzorok védett üzemmódban? Mekkora a fizikai címtartomány az i386/486-os processzorok védett üzemmódjában?
24. Vázolja fel az i386/486-os processzorok címkiszámítási módját szegmentált címzés esetében. Milyen részekből tevődik össze a szegmensszelektor? Mire szolgál a szelektor TI és RPL mezője?
25. Hol találhatók a szegmensdeszkriptorok és milyen részei vannak?
26. Milyen tárkezelési forma a lapozásos technika? Mekkora az i386/486-os processzorok által használt lapméret?
27. Vázolja fel az i386/486-os processzorok szegmentált lapozásos címkiszámítási eljárásának módját!
28. Miért használnak kétszintű(szegmentálással együtt háromszintű) címkiszámítást az i386/486-os processzorok?
29. Mit tartalmaznak a lapdeszkriptorok? Hol található a lapkatalógus-tábla kezdőcíme? Milyen méretű a lapcímfordítást segítő TLB?
30. Valós üzemmódban milyen védelmi lehetőséggel rendelkeznek a processzorok? Mit szabályoznak a védelmi előírások?
31. Mit nevezünk rendszerobjektumoknak? Hogyan lehet elérni a rendszerobjektumokat? Mit nevezünk kapuknak?
32. Az egyes privilégiumszintekhez milyen erőforrások(programok) vannak hozzárendelve? Mely programok rendelkeznek a legmagasabb privilégiumszinttel?
33. Egy adott privilégiumszinten lévő program milyen privilégiumszinten lévő más programokat, illetve adatokat tud elérni?
34. Mit értünk aktuális(CPL) és tényleges(EPL) privilégiumszint alatt? Hogyan szabályozza a CPL és az RPL értéke a tényleges privilégiumszintet?

35. Hogyan szabályozott az I/O eszközök elérhetősége az i386/486-os processzoroknál? Hogyan szabályozza az I/O eszközök elérhetőségét az IOPL értéke?
36. Milyen deszkriptortáblákat használnak az i386/486-os processzorok? Mire szolgál a globális(GDT) és a lokális(LDT) deszkriptortábla? Mire szolgál a megszakítási deszkriptortábla(IDT)?
37. Mit tartalmaz a taszk állapotszegmens táblázata? Mit értünk taszkváltás alatt? Mi lehet oka a taszkváltásnak? Ismertesse a taszkváltás fontosabb lépéseit!
38. Ismertesse a Motorola MC68000-es processzorok felépítését; főbb részeit! Melyek a processzorok regiszterei? Ismertesse az SR állapotregiszter funkcióit!
39. Ismertesse az MC68000-as processzor utasításformáit, címzési módjait, az utasításfeldolgozás formáját!
40. Mutassa be az MC processzorok felépítését, főbb egységeit!
41. Milyen az MC68020/68030/68040-es processzorok utasításszerkezete, címzési formája.
42. Ismertesse a SPARC processzorok legfontosabb jellemzőit, utasításformáit, címzési formáit!
43. Ismertesse a MIPS processzorok felépítését, utasításformáit, memóriakezelését!

7. FEJEZET TÁROLÓESZKÖZÖK

A tárolóeszközök tárgyalásakor elsősorban az általános jellemzőket ismertetjük, kiegészítve azokkal az információkkal, amelyek a legelterjedtebb **IBM PC-XT/AT kompatibilis gépek**kel kapcsolatosak. Így a géppel való kapcsolattartást a **BIOS**(Basic Input/Output system) és a **DOS**(Disk Operating System) szintjén vizsgáljuk, ha ez szükséges.

A BIOS egyes részprogramjait(funkcióit) szoftver megszakítás útján lehet igénybe venni. A rutin működéséhez szükséges paramétereket, adatokat a processzor megadott regisztereibe kell előkészíteni és az esetleges eredményt is onnét lehet megkapni. Az elvégzendő feladat jellegéből adódóan, a BIOS rutinjait többnyire assembly-szintű programokból lehet használni.

Az operációs rendszer(DOS) funkcióit billentyűzetről begépelt, vagy programból kiadott DOS parancsokkal lehet igénybe venni.

7.1.ÁLTALÁNOS JELLEMZŐK

A mikroszámítógép melletti, központi egységen kívüli tárolóeszközök a nagyobb mennyiségű adat, hosszabb program aktuálisan nem szükséges részeinek tárolására szolgálnak, illetve a gépek közötti adatcsere eszközeként, adathordozójaként(floppy) használatosak. Ez utóbbi formájában, mint bemeneti/kimeneti adathordozó alkalmazott.

Három eszköztípust, a hajlékonylemezt(floppy-t), a merevlemezt(winchester-t) és az optikai lemezt vizsgáljuk meg részletesebben.

7.1.1.Hajlékonylemezek(floppy-k)

A hajlékonylemezes tárolók a legszélesebb körben alkalmazott adathordozói a mikroszámítógépek környezetének. Többféle méretben és jellemzővel gyártják.

Az adatok tárolására, a vékony műanyag hordozóra felvitt mágnesezhető réteg szolgál, amelyet papir(5.25"-es lemeznél), vagy merev műanyag(3.5"-es lemeznél) tokban helyeznek el. Az adatok a lemez felületén koncentrikus körök, **sávok**(track) mentén helyezkednek el. A meghajtóba helyezett lemezt, a berendezés 300-360 f/p fordulatszámmal forgatja és az olvasófej a lemez felületéhez hozzáérve írja/olvassa a tárolt információt. Az író/olvasófej helyzete miatt, a lemez csak íráskor és olvasáskor forog, mert különben az olvasófej rövid időn belül tönkremenne. A lemez tartalma a felülírástól(átírástól) optikailag, mechanikusan védhető. Az 5.25"-es lemez papírtokján,

oldalt lévő bevágást fényt át nem eresztő fóliával leragasztva, megakadályozható a lemezre írás. A kis méretű(3.5"-es) lemeznél, a műanyag tokon lévő kis retesz eltolásával, az ellenőrző fény útját szabaddá téve, tiltható le a lemezre írás.

A különböző típusú lemezek kapacitása a fizikai mérettől, a sávsűrűségtől és a jelrögzítési jelsűrűségtől függ.

A floppy-k legfontosabb adatait a következő táblázatban foglaltuk össze.

7-1.táblázat: *Floppy lemezek legfontosabb adatai*

kapacitás[byte]	360K	1.2M	720K	1.44M	2.88M
átmérő[inch]	5.25	5.25	3.5	3.5	3.5
fejszám	2	2	2	2	2
sávszám	40	80	80	80	80
szektorszám	9	15	9	18	36
sávsűrűség[tpi]	48	96	135	135	135
jelrögzítés	MFM	MFM	MFM	MFM	MFM
ford.szám[1/p]	300	360	360	360	360
átv.seb.[Kb/s]	250	500	250	500	1000

Az adatokat a BIOS segítségével tudjuk beállítani; a lemez BIOS által elvileg kezelhető, maximális kapacitása: 64MB.

7.1.2.Merevlemez(winchester)

A merevlemezek, amelyek kapacitása egyre nagyobb lesz, a mikroszámítógépek belső, háttértárolóiként szolgálnak.

A 6-15 db mágnesezhető réteggel bevont könnyűfém lemezt, amely zárt védőburkolatban van elhelyezve, a meghajtó 3600 f/p sebességgel forgatja. Az állandó forgás miatt az olvasófejek nem érnek hozzá a lemez felületéhez, hanem ú.n. 'repülő fej'-eket használnak, amelyek kialakítása olyan, hogy a lemez forgása következtében kialakuló légpárna hatására, a lemez felületétől néhány mikron távolságra kerülnek.

A merevlemezes egységeknek többféle változatát használják, így:

- beépített(80MB-1.8.GB),
- cserélhető(80-540 MB),
- hordozható(20-80MB) lemezek használatosak.

A lemezek mérete egyre csökken, a szokásos méretek: az 5.25"-es, 3.5"-es és 2"-es átmérő. Ugyanakkor a lemezek kapacitása viszont nő, így tájékoztató jelleggel azt lehet mondani, hogy:

- 5.25"-es lemezek kapacitása 20-80MB, elérési ideje 25-50ms, átviteli sebessége 625 Kb/s nagyságú;

- 3.5"-es lemezek kapacitása 50-540MB, elérési ideje 10-20ms, átviteli sebessége 900 Kb/s nagyságú.

A nagyobb teljesítményű gépekben egyre gyakoribb az 1GB, vagy afeletti kapacitású(8-10 ms elérési idejű) merevlemezek használata is.

A BIOS által elvileg kezelhető, maximális lemezkapacitás 8.38GB nagyságú.

7.1.3.Optikai lemezek

Az optikai jelrögzítés valamilyen formáját alkalmazó eszközök az utóbbi néhány évben terjedtek el, már ma is igen széles körben.

Az adatrögzítés, a korong felületén egy spirális pályán, lézersugárral kialakított digitális jelsorozat által történik. A használt lemeztípusokat három csoportra lehet osztani: a **csak olvasható** CD-ROM(Compact Disc Read Only Memory), az **egyszer írható** WORM(Write Once Read Meny) és az **írható/olvasható** lemezek csoportjára.

A lemezek átmérője 80-120 mm, kapacitásuk 210-650 MB és az adatátviteli sebességük 150, 300, 600 Kb/s nagyságú. A lemezen elhelyezhető adatmennyiség nagy mértékben függ az alkalmazott adattömörítési eljárástól; a HDCD(High Densitiy CD), SDCD(Super Density CD) videolemezek 3.7-10 GB-nyi információt is tárolhatnak.

7.2.AZ INFORMÁCIÓTÁROLÁS FORMÁI

A tárolóeszközök kapcsán az információtárolás formáit három szinten mutatjuk be:

- *fizikai szinten*, azaz az eszközön magán milyen fizikai formában történik az adatok elhelyezése;
- *BIOS szinten*, amely a felhasználó számára az adattárolás módjának lehetséges legalsó befolyásolási, meghatározási szintje;
- *DOS szinten*, amely a felhasználó számára a tárolóeszköz szokásos alkalmazási szintje.

7.2.1.Fizikai szintű tárolás

a.)Mágneses adatrögzítés

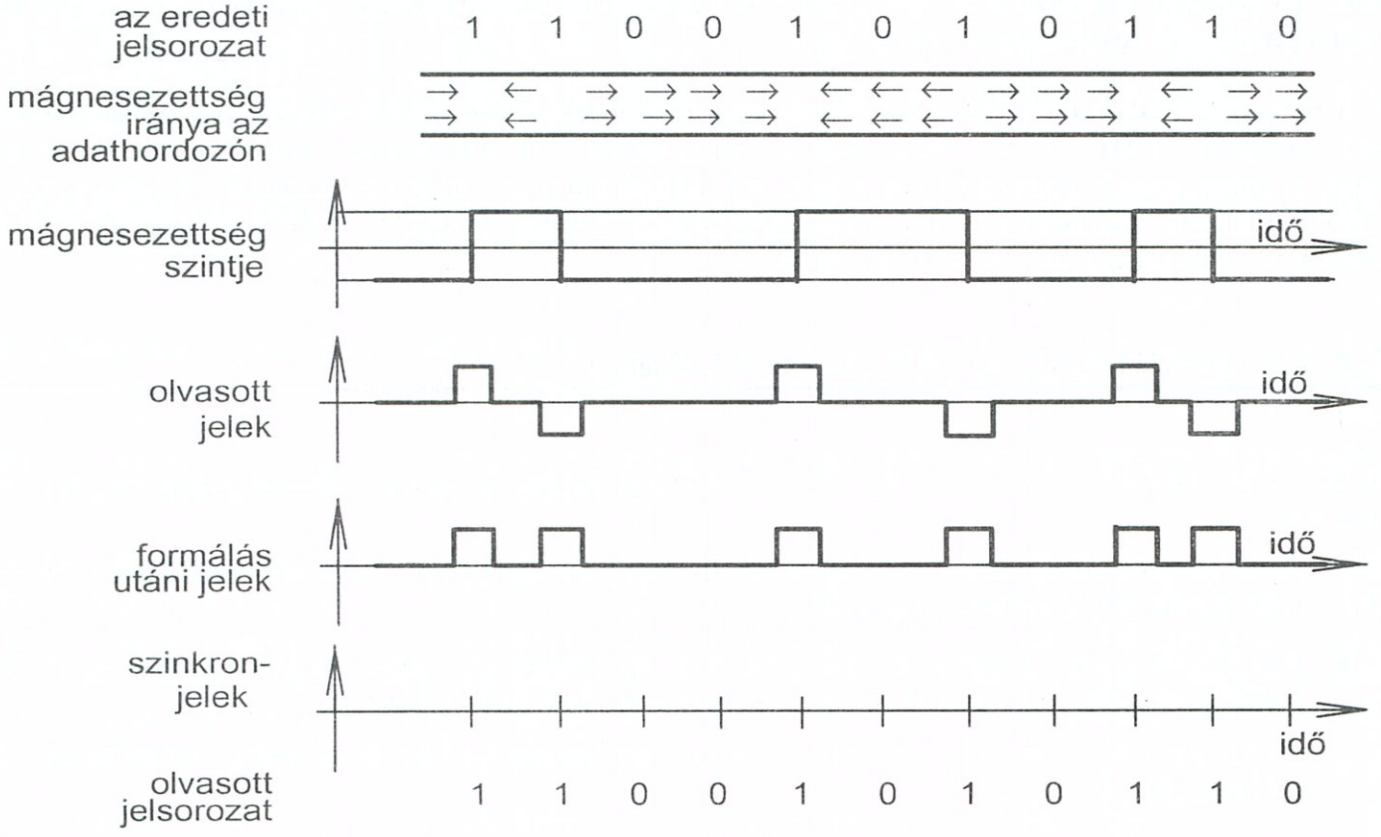
A mágneses elvű adatrögzítés a leggyakrabban alkalmazott jelrögzítési forma a számítógépek mellett. Az adatrögzítés formájának kialakításakor azt kell biztosítani, hogy az adatok elhelyezése a lehető legtömörebb legyen, ugyan-

akkor az adatok visszanyerése, visszaolvasása is kellő megbízhatóságú legyen.

Jelrögzítés formája

A számítógépek mellett alkalmazott mágneses adatrögzítés módja hasonló a közönséges magnetofonnál használt jelrögzítési technikához, avval a lényegi különbséggel, hogy a számítógépi adatrögzítés mindig digitális megoldású.

Az információ hordozója egy nem mágnesezhető alapanyagon lévő vékony mágnesezhető réteg, illetve annak mágnesezettségi iránya és szintje. A digitális jelrögzítés miatt csak két mágnesezettségi szintet használnak, ellentétben a közönséges magnónál alkalmazott analóg jelrögzítéssel. A másodlagos táraknál az adathordozó mozog és így a változó mágneses tér az olvasófejekben elektromos jeleket hoz létre, amelyeket felerősítve és formálva használnak fel azután.



7-1.ábra: *Mágneses jelrögzítés NRZI módszerrel*

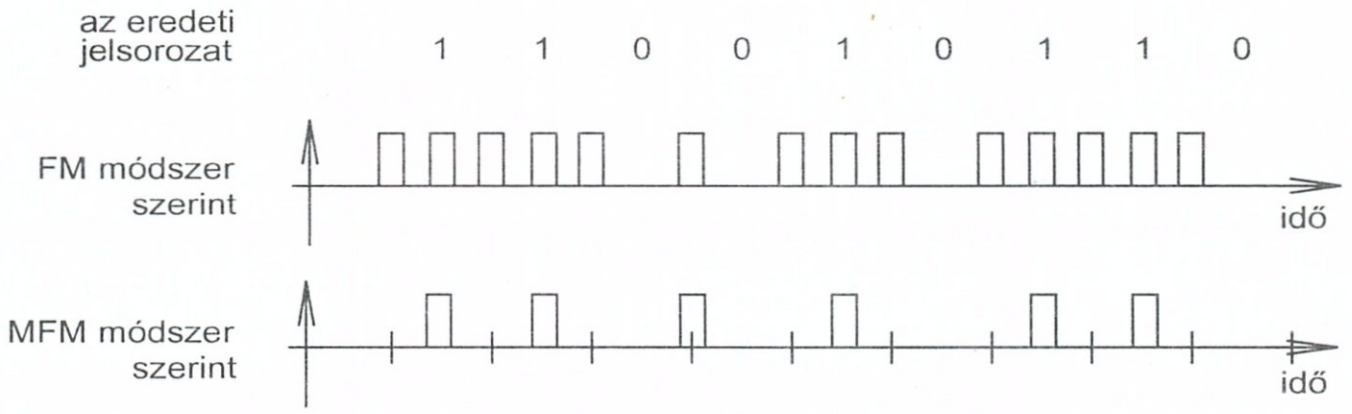
Az alkalmazott többféle módszer közül a leggyakrabban használt **NRZI**(no return to zero impulse) jelrögzítési módon keresztül mutatjuk be az adatok tárolási módját(7-1.ábra). A mágnesezettség iránya kétféle lehet:

- *párhuzamos* a felülettel(longitudinal recording), amint ezt az ábrán is bemutatjuk; illetve

- *merőleges* a felületre(perpendicular recording), amely a korszerűbb változat.

Az NRZI jelrögzítési módszer két mágnesezettségi szintet(-B, +B) használ az adatok rögzítéséhez. A két mágnesezettségi jelszint között átmenet akkor következik be, ha a rögzítendő jelzorozatban 1-es szerepel. (Az információt kifejező két állapot az átmenet léte, illetve nem léte.). A 7-1.ábrán láthatók a különböző jelalakok, többek között a jelformálás után a kimeneten kapott jelsorozat is. A jelsorozat helyes értelmezéséhez azonban ismerni kell azokat az időpontokat, amikor a jelsorozatot ellenőrizni, 'mintavételezni' kell, hogy megállapítható legyen a jelsorozat 1-es, vagy 0 volta. Ezért az olvasó részére **szinkron jeleket kell biztosítani**, amelyek megszabják a mintavételezés időpontját. Ezt a szinkronjel sorozatot az NRZI rendszernél magából a jelsorozatból állítják elő.

A lemezegységeknél használt kódolási módok közül az egyszeres sűrűségű lemezeknél alkalmazott FM(frequency modulated), a dupla sűrűségű lemezeknél alkalmazott módosított frekvenciamodulált(MFM=modified frequency modulated) jelrögzítési technikát mutatja be a 7-2.ábra.



7-2.ábra: *FM, MFM jelrögzítési módok*

Az FM módszernél minden egyes bitmező egy szinkron impulzussal kezdődik és a bitmező közepén akkor van jel, ha 1-es bitértéket kell rögzíteni. Az MFM jelrögzítési módnál külön szinkronjel csak akkor kerül rögzítésre, ha a 0 jelet ismét egy 0 érték követi.

Adattárolás módja

A lemezek felületét használat előtt formázni kell, azért, hogy kialakuljon az a szabványos jelrögzítési forma(struktúra, szinkronizációs jelek), amelyet a gépek operációs rendszerei, olvasó berendezései kezelni tudnak.

A lemez felülete **sávokra**(track) van osztva, amelyen belül kisebb egységek **szektorok**(sector) kerülnek kialakításra. Egy-egy szektor az, amit a kezelő szoftver, az operációs rendszer, illetve annak részeként működő BIOS, egy egységként tud írni, olvasni. Ha több lemezfelület áll rendelkezésre a tároláshoz(hajlékonylemeznél kettő, a merevlemeznél 8-10, vagy több), akkor a

fizikailag egymás felett elhelyezkedő sávokat együtt **cilindernek**(cylinder) nevezik.

Igen gyakran, az adattárolásra szolgáló sávok mellett, a gyorsabb adatelérés érdekében, külön sávokat használnak az indextáblázatok kialakításához(index track), továbbá a nagyobb megbízhatóság érdekében tartalék sávokat (alternate track) képeznek a meghibásodott és nem használható szektorok, sávok tartalmának átmentéséhez.

A szinkronizáció mellett, a sávok kezdetét is jelezni kell az olvasó rendszer számára. Ennek a jelölésére mindig valamilyen fizikai jel szolgál, pl. a hajlékonylemezeknél a lemezen lévő lyuk, vagy merevlemezeknél a lemezen lévő bevágás, hasíték, amelyet többnyire fotoelektronikus módszerrel érzékel a berendezés. A sávon belüli szektorkezdetek jelzésére két módszer használatos:

- **szoft-szektoros**(soft sectored) megoldás, amelynél a sáv részekre osztása megfelelő szabványos jelsorozatok alkalmazásával történik, azaz minden szektort valamilyen egységes jelsorozat vezet be;

- **hard-szektoros**(hard sectored) megoldás(a mikroszámítógépek környezetében ritkábban használt), amelynél a szektorok kezdetét is valamilyen fizikai jel érzékelése alapján határozza meg a berendezés. A hajlékonylemezeknél ezt a módszert nem használják.

A szektorok logikai felépítése két részre tagolja azt: a fej- és az adatrészre. A **fejrész** az azonosításhoz szükséges információkat, valamint a szinkronizáláshoz szükséges jeleket tárolja. Az **adatrész** magát a tárolandó adatsort, valamint a megbízhatóság növelése érdekében az ú.n. ellenőrző összeget foglalja magában. Az ellenőrző összeg a teljes szektorra vonatkozik. A szektor egyes részeit, valamint a szektorokat egymástól, üres részek(gap) választják el.

Megbízhatóság növelése

A lemezeken rögzített adatok biztonságos visszanyerése érdekében többféle módszert alkalmaznak a különböző berendezések. A lemezek hibáinak okai:

- fizikai hiba(hard error), amely lehetetlenné teszi az adott helyen(bad spot) az adatrögzítést, ezért az adatokat a tartaléksávra kell áttenni;
- látszólagos hiba(soft error), amely többszöri olvasási kisérlet hatására megszűnik;
- szennyeződés, amely gyengíti az érzékelt jel nagyságát és emiatt bizonytalanná válik a működés;
- keresési hiba(seek error), amely az olvasófej rossz pozicionálásából adódik.

A jelrögzítés hibáinak felfedésére, vagy kijavítására különböző szabályok szerint képzett ellenőrző összegeket használnak. Az egyik leggyakrabban alkalmazott, hibafelfedésre lehetőséget adó eljárás a polinomiális, vagy ciklikus kód(**CRC**=polynomial or cyclic redundancy code). Hibajavításra is al-

kalmas ellenőrző összegképző eljárás az **ECC**(error correcting and checking) módszer.

A CRC összeg képzéséhez, a jelsorozat 1-eseinek megfelelő helyiértékek alapján kiszámított polinomértéket egy előre meghatározott, generált polinomértékkel elosztva, a kapott maradék értékét az ellenőrzőtt jelsorozathoz hozzácsatolják. Az ebből a kibővített jelsorozatból képzett ellenőrző összeg, már osztható lesz a generált értékkel. Ennek alapján a jelsorozat olvasásakor előállítják az ellenőrző összeget és elosztják a generált értékkel; ha a maradék 0, akkor az olvasás hibátlan volt.

b.)Optikai adatrögzítés

Jelrögzítés formája

Az optikai lemezek jelrögzítési technikája egészen más, mint a mágneses adatrögzítés módszere. A lemezek előállítási módja függ attól is, hogy milyen típusú, azaz csak olvasható, vagy írható/olvasható lemezről van szó.

A **csak olvasható lemezek** készítésének első fázisa a mester-lemez elkészítése. A rögzítendő jelsorozatot nagy teljesítményű lézersugárral égetik bele a lemez felületébe. A 0.5 µm átmérőjű beégetett lyukak(pit) és közbülső szakaszok(land) váltakozása adja a tárolandó jelsorozatot, még pedig az NRZI módszerhez hasonlóan, a lyuk és nem-lyuk átmenetek adják az 1-esnek megfelelő biteket.

A mester-lemezről készül a negatív nyomóforma, amellyel a pozitív (végső) lemezt préselik műanyagból. A lemez felületét vékony fényvisszaverő fém (alumínium)réteggel, majd egy műanyag, védőréteggel vonják be.

A lemezeken az adatok tárolása spirális pálya mentén történik, hasonlóan a hagyományos hanglemezekhez. A sávok távolsága(a menetemelkedés) 1.6 µm, az adattárolási jelsűrűség állandó. Ez utóbbi miatt, az olvasófej helyzetétől függően, a lemez fordulatszáma 250-500 f/p között változik.

Az információ tárolásához kialakított lyukak(pit-ek) szélessége 0.5 µm, hossza 0.833-3.056 µm(3÷11 hosszegység) közötti.

Olvasáskor, a cimkézéssel ellentétes oldalról, egy kisteljesítményű(kb. 1 mW-os) lézersugárral világítják meg a lemezt. A sugár átmérője a lemez felületén(a belépési ponton) 0.8 mm, amelyet az alkalmazott lemezanyag (polikarbonát) fénytörését kihasználva, a tükröző felületig kb. 1.7 µm átmérőjűvé fókuszálnak.

Az olvasófej pályán tartását egy részben mechanikus, részben elektromágneses rendszer biztosítja, amelynek vezérléséhez a visszavert fénysugár erősségének változását használják fel.

Az **írható/olvasható lemezek** ettől eltérő módon készülnek. A jelrögzítés magneto-optikai elvek alapján történik. A lemez felületét különleges fémből

(gadolinium, terbium) készült vékony réteggel vonják be, amelynek mágneses tulajdonságai különlegesek.

Ezek a fémek alacsony hőmérsékleten nem mágnesezhetők, viszont felmelegítve(magas hőmérsékleten) mágnesezhetővé válnak. A jelek felviteléhez lézersugárral felmelegítik a fémréteget, majd felmágnesezik a kívánt irányban.

Az így kialakított felületet, a CD-ROM-hoz hasonlóan, olvasáskor egy gyenge lézerrel tapogatják le. A mágnesettségtől függően, a visszavert lézerfény polarizációs iránya változik, amelynek alapján a tárolt információ visszanyerhető.

Adattárolás módja

Az optikai lemezek olvasása nagy hibaarányt eredményez, ezért minden rögzítendő byte-ot egy hibajavító kód alkalmazásával, kiegészítenek 14 bitre, majd ezeket csoportosítják.

Egy-egy adat**csoport**(group) 24 kibővített adatbyte-ot, valamint egyéb kiegészítő információkat foglal magában. Az adatcsoport 588 bit hosszúságú és az alábbi felépítésű:

- szinkronmező, 24 bit(801002h jelsorozat),
- másodlagos vezérlési mező, 14 bit,
- adatmező, 24x14 bit=336 bit,
- ellenőrző mező, 8x14 bit=112 bit,
- kitöltő(margin) bitek, az adatcsoport végén, valamint minden kibővített byte között 3-3 bit, összesen: 102 bit.

98 ilyen adatcsoportból épül fel az adat**keret**(frame) oly módon, hogy minden csoport között, az előbbiekben már említett módon, 3 elválasztó bit található.

Egy-egy ilyen keret alkot egy, a felhasználó által kezelhető kb. 2KB(2352 byte)-os felhasználói **adatblokk**ot. Az adatblokkok tartalmi kialakítására vonatkozóan, az évek során több szabvány is kidolgozásra lerült(így pl. a 'Yellow Book'-ban, a 'Green Book'-ban, vagy az 'Orange Book'-ban rögzítettek, az adattárolással kapcsolatosan).

Megbízhatóság növelése

Az optikai lemezek felülete, az olvasás technikája igen sok hibát eredményezhet. A hibák mérséklésére egyrészt speciális olvasóberendezést alkalmaznak, másrészt, az adatrögzítéskor különleges hibajavító kódot(cross-interleaved Reed-Solomon code) használnak. Az adatok visszanyerésének biztonságát növelik avval is, hogy az adatcsoportokon belül az egyes adatbyte-ok nem sorrendben követik egymást, hanem megkeverve, egy meghatározott algoritmus szerint.

7.2.2.BIOS szintű információkezelés

A lemezeken lévő adatok sáv- és szektorszintű kezelését a mikroszámítógépek beviteli/kiviteli műveleteit megvalósító programrendszer, a BIOS(Basic Input/Output System) oldja meg. A BIOS egyes részprogramjait(funkcióit) szoftver megszakítás útján lehet igénybe venni. A rutin működéséhez szükséges paramétereket, adatokat a processzor megadott regisztereibe kell előkészíteni és az esetleges eredményt is onnét lehet megkapni.

a.)Hajlékonylemez

A hajlékonylemezen a sávok(cilinderek) és a szektorok azonosítása sorszámok segítségével történik. A sávok számozása 0-39(360KB-os lemez esetében), vagy 0-79 közötti értéket vehet fel; a sávon belüli szektorok számozása 1-9(360 KB-os lemeznél), illetve 1-15 közötti számokkal történik. A szektorok mérete 512 byte.

Külön kiemelendő, hogy a szektorok számozása BIOS szinten 1-essel kezdődik és nem 0-val, mint az operációs rendszer szintjén.

A szektorok fizikai szintű(ahogy a lemez felületén egymás után következik) és BIOS szintű logikai sorszámozása megegyezik a hajlékonylemez alacsony fordulatszáma miatt. Ugyanis nagyobb fordulatszám mellett(merevlemeznél), több szektornyit is elfordulhat a lemez a következő olvasás időpontjáig és emiatt a logikai sorszámozás, célszerűen, más lesz, mint a fizikai sorrend.

A sáv- és szektorműveletek a hexadecimális **13-as(13h) megszakítási vektoron** keresztül érhetők el. A választott funkció sorszámát az AH regiszterben kell megadni. A használható funkciók az alábbiak:

00h	reset; a meghajtót alapállapotba hozza, az olvasófejet a 0.sávra viszi;
01h	állapotjelzők olvasása; az utoljára végrehajtott lemezművelet utáni állapot kiolvasása;
02h	olvasás; bármelyik megadott szektor beolvasása;
03h	írás; szektor(ok) lemezre írása;
04h	verifikálás; a megadott szektor ellenőrző összegének (CRC) az újbóli előállítása ellenőrzés végett;
05h	sáv formázása a megadott paraméterek szerint;
08h	lemezmeghajtó paramétereinek a lekérdezése;
15h	lemezmeghajtó típusának megállapítása;
16h	lemezcsere megtörténtének megállapítása;
17h	lemez formátumának a megállapítása;
18h	lemez formátumának beállítása formázáshoz.

b.)Merevlemez

A merevlemeznél a sávok(cilinderek) és szektorok sorszámozása hasonló a hajlékony lemezéhez; a cilinderek száma a lemez kapacitásától nagymértékben függ, de 0-1023 között mozog az átlagos kapacitás mellett. A szektorok számozása a merevlemeznél is 1-essel kezdődik, 1-64 közötti értékeket felvéve.

A merevlemez nagyobb fordulatszáma miatt, a fizikai sorrend és a BIOS szintű logikai sorrend elválik egymástól, ahogy erről a hajlékonylemez kapcsán már említést tettünk. Ez az eltolódás(interleave) 1:6, 1:3, 1:2 arányú lehet. Az alábbi táblázat mutatja be a fizikai és a logikai sorszámozás értékeit az egyes arányok mellett.

fizikai sorszámok		1	2	3	4	5	6	7	8	9	10	11
logikai sorszámok	1:6	1						2				
	1:3	1			2			3			4	
	1:2	1		2		3		4		5		6

A korszerű, nagy kapacitású és gyors működésű lemezeknél, az alkalmazott 'interleave' érték már **1:1 nagyságú**, azaz a fizikai és a logikai sorszámozás megegyezik.

A sáv-és szektorműveletek a merevlemez esetében is a **13h-as szoftver megszakítási vektor** felhasználásával végezhetők el. Ha merevlemez is van a gépben, akkor a hajlékonylemez funkcióit a 40h-es megszakítási vektoron keresztül lehet elérni. Ez esetben a 13h-asra irányuló, hajlékonylemezre vonatkozó kéréseket a rendszer automatikusan a 40h-es vektorra irányítja.

A merevlemez esetében használható funkciók az alábbiak. A funkció sorszámát az AH regiszterben kell elhelyezni.

00h	reset; a meghajtó alaphelyzetbe hozására;
01h	állapotjelzők olvasása;
02h	olvasás; tetszőleges, megadott szektor(ok) olvasása;
03h	írás; tetszőleges, megadott szektor(ok)ra írás;
04h	verifikálás; az ECC ellenőrző összeg helyességének vizsgálata;
05h	formázás; tetszőleges sáv, megadott paraméterekkel való formázása;
08h	a lemezmeghajtó paramétereinek lekérdezése;
09h	külső meghajtó rendszerbe illesztése;
0Ah	bővített olvasás; szektor(ok) olvasása a szektorvégi ellenőrző összeg(ECC) értékével együtt;
0Bh	bővített írás; szektor(ok) írása a lemezre, a szektorvégi ellenőrző összeggel együtt;

0Ch	író/olvasófej beállítása;
0Dh	reset; a meghajtó alaphelyzetbe állítására;
10h	a meghajtó üzemkész állapotának a lekérdezésére;
11h	az író/olvasófejek 0.sávra állítása, valamely hiba bekövetkezése után;
14h	a meghajtó vezérlőjének ellenőrzése, tesztelése;
15h	a meghajtó típusának a lekérdezése;
19h	az olvasófej nyugalmi helyzetbe állítása.

7.2.3.DOS szintű információkezelés

A lemeztartalom felhasználói programokból történő használata egységes formátumot kíván minden esetben. A DOS rendszer ennek biztosítására, a lemez felületén létrehoz olyan táblázatokat, nyilvántartásokat, amelyek a lemeztartalom automatikus kezelését teszik lehetővé.

A lemezegységek DOS szintű kezelése nagyjából hasonló a hajlékonylemezes és a merevlemezes egységeknél, csak bizonyos részek a hajlékonylemezek esetében hiányoznak. (Megjegyzendő, hogy a DOS a szektorok sorszámozását 0-val kezdi.)

A lemezek DOS által létrehozott és kezelt részei az alábbiak:

- particiós tábla(merevlemeznél),
- betöltő szektor(boot sector),
- állomány elhelyezési tábla(FAT=file allocation table),
- állomány elhelyezési tábla másolata(FAT 2),
- főkönyvtár(root directory),
- további állományok(rendszer és felhasználói).

a.)Betöltő szektor(boot sector)

A betöltő szektor az ú.n. rendszerlemezek 0.logikai sorszámú szektora, amely az operációs rendszer betöltését hajtja végre a gép indítása után. A gép indításakor a ROM tárolóban elhelyezett és automatikusan elinduló ellenőrző program, a gép egységeinek ellenőrzése után a merevlemez, vagy ha az nincs, akkor valamelyik hajlékonylemezes meghajtóban lévő lemez 0.szektorát tölti be a memóriába és átugratja a végrehajtást a szektor első byte-jára, azaz az ott elhelyezett utasításra. A szektorban tárolt betöltő program ezután betölti az operációs rendszer szükséges részeit a memóriába.

Azokat a lemezeket, amelyeken ez a betöltő szektor megtalálható **rendszerlemezek**nek hivják. Az adatlemezek ezt a betöltő programot nem tartalmazzák.

A betöltő szektor részei és helyfoglalásuk [byte]-ban:

ugró utasítás	3
gyártó neve, verziószám	8
szektoronkénti byte-szám	2
klaszterenkénti byte-szám	1
foglalt szektorok száma	2
FAT másolatok száma	1
főkönyvtár bejegyzéseinek száma	2
szektorok száma összesen	2
adathordozó azonosító kódja	1
FAT mérete[szektor]	2
sávonkénti szektorszám	2
író/olvasófejek száma	2
rejtett szektorok száma	2
szektorok száma összesen	4
lemez fizikai azonosító kódja	1
(üres)	1
ellenőrző kód	1
lemez azonosító száma	4
kötetazonosító	11
FAT típus azonosítója	6
foglalt	2
betöltő program	

A betöltő program a főkönyvtár első két állományát, az IO.SYS, MSDOS.SYS rendszerállományokat tölti be.

b.)Állományelhelyzési tábla(FAT)

A lemezek állományelhelyezési táblázata szolgál arra, hogy az operációs rendszer nyilvántartsa és nyomonkövesse azt, hogy a lemez mely szektorai foglaltak, vagy szabadok.

A foglaltság nyilvántartása nem szektoronként történik, hanem nagyobb egységenként, klaszterenként(cluster). Egy-egy klaszter kettő valamely hatványának megfelelő darabszámú szektort foglal magában. (Pl.: a merevlemezeknél ez 4-32 szektor/klaszter, a kialakított partició nagyságától függően.)

Minél nagyobb a klaszterekbe fogott szektorok száma, annál gyorsabb a lemezen a keresés, ugyanakkor a helykihasználás mértéke ennek következtében igen leromolhat.

A FAT-ben egy-egy klaszterhez tartozó adatokat 12, vagy 16 bit(12 bit, ha a klaszterek száma < 4087-nél) hosszúságú területeken, mezőkön tárolja a rendszer. A FAT mezőinek tartalma:

0.mező:	a mező 1.byte-ja a lemez azonosító kódját, a maradék rész 1-eseket tartalmaz;
1.mező:	csupa 1-est tartalmaz;

többi mező: ha **0000**, akkor még szabad klaszter,

ha **nnnn**, akkor a klaszterhez kapcsolódó következő klaszter sorszáma = nnnn,

ha **FFF0-FFF6**, akkor a klaszter foglalt,

ha **FFF7**, akkor a klaszter fizikailag sérült,

ha **FFF8-FFFF**, akkor ez az állomány utolsó klasztere.

c.)Főkönyvtár(root directory)

A főkönyvtár, vagy gyökérkönyvtár a FAT másolatok után helyezkedik el a lemezen és minden egyes állományhoz(file-hoz) egy 32 byte-os bejegyzést tartalmaz.

Ezek a bejegyzések az alábbi részeket tartalmazzák(a méretek [byte]-ban adottak):

az állomány neve	8
az állománynév kiterjesztése	3
az állomány attributum byte-ja	1
foglalt	10
utolsó módosítás időpontja	2
utolsó módosítás dátuma	2
az állomány első klasztere	2
az állomány hossza	4

7.3.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Milyen hajlékonylemez típusokat használnak a mikroszámítógépek környezetében? Ismertesse a hajlékonylemezek típusait és azok legfontosabb adatait!
2. Az 1.2MB kapacitású floppy hány sávval rendelkezik? Mekkora az 1.2MB-os floppy-meghajtó adatátviteli sebessége?
3. Milyen jellemzőkkel rendelkezik az 1.44MB kapacitású floppy?
4. Mekkora lehet a BIOS alapján a floppy-k maximális kapacitása?
5. Milyen merevlemezeket használnak a mikroszámítógépekbe beépítve? Milyen felépítésűek a merevlemezes egységek?
6. Milyen adatátviteli sebességgel rendelkeznek a 3.5" átmérőjű merevlemezek?
7. Ismertesse az optikai lemezek legfontosabb jellemzőit! Milyen kapacitásúak az adattárolásra használt CD-ROM-ok?
8. A mágneses adattárolók esetében, milyen jelrögzítési módszereket használnak az adatok tárolásának fizikai szintjén? Hogyan oldják meg az olvasás szinkronizálását?

9. Milyen jelrögzítési forma az NRZI forma? Rajzolja fel a kapcsolódó idődiagramot! Ismertesse az FM és MFM kódolási mód lényegét!
10. Milyen módon történik az adatok elhelyezése a mágneslemezek felületén? A lemez formázásakor milyen adatokat helyeznek el a lemezen? Mi jelzi a sávok kezdetét?
11. Ismertesse a különböző optikai lemeztípusok legfontosabb jellemzőit! Ismertesse a CD-ROM-oknál alkalmazott optikai jelrögzítési formát!
12. Ismertesse az optikai lemezeknél használt adattárolási formát, az adatok elhelyezésének, a megbízhatóság növelésének módszerét!
13. Milyen a sávok(cilinderek) BIOS szintű sorszámozása? Milyen a szektorok BIOS szintű sorszámozása?
14. Miért nem egyezik meg a merevlemezeknél a fizikai és a logikai (BIOS) szintű szektorsorszámozás? Mit értünk az 'interleave' fogalma alatt?
15. A hajlékonylemez egységeknél milyen műveletek végezhetők el BIOS szinten?
16. Soroljon fel néhány BIOS szintű lemezműveletet a merevlemezek esetében!
17. Milyen a szektorok DOS szintű sorszámozása? Milyen adatokat helyez el az operációs rendszer a lemez felületén?
18. Ismertesse a betöltő szektor(boot sector) tartalmát! Mit tartalmaznak az állományelhelyezési táblázatok(FATs)?
19. Ismertesse a főkatalógus felépítését, tartalmát!
20. Soroljon fel néhány parancsot, amellyel a merevlemez meghajtó vezérelhető.

8. FEJEZET MONITOROK

A különböző monitor típusok ismertetésénél, a témakör igen kiterjedt volta miatt, csak a leglényegesebb jellemzők ismertetésére szorítkozunk. Az általános ismereteken túl, a forgalomban és használatban lévő leggyakoribb monitor vezérlőkártyatípusok alaptulajdonságait foglaljuk össze a 8.2.pontban. Mivel a korszerűbb kártyatípusok(pl. VGA, SVGA, de már az EGA is) igen bonyolult vezérlési mechanizmust valósítanak meg, csak a legegyszerűbbek esetében(MDA, HGC, CGA) részletezzük kicsit bővebben a képernyőkezelés lehetőségeit, annak érdekében, hogy az olvasó ismeretet szerezhessen a megoldások elvéről.

8.1.ÁLTALÁNOS JELLEMZŐK, FUNKCIÓK

A mikroszámítógépekhez használt monitorok fizikai működése nagy mértékben hasonló a megszokott TV készülékek működéséhez. A képernyő tartalmát egy elektronsugár rajzolja fel a fénykibocsátó réteggel bevont felületre, még pedig a képernyő bal felső sarkától kezdődően, jobbra és lefelé haladva, párhuzamos sávokra bontott részekben. Egy teljes képernyő tartalom kirajzolása 1/50 sec-ig tart.

A képernyő végigpásztázásában kétféle eljárást alkalmaznak:

- **folytonos**, egymást követő soronkénti pásztázás(non-interlacing), amely kevésbé finom felbontást, de gyakoribb képfelfrissítést eredményez;
- **váltott soros**, minden második soronkénti pásztázás (interlacing), amely finomabb felbontást, de alacsonyabb frissítési frekvenciát eredményez. A váltott soros pásztázásnál félképenként 1/50 sec szükséges, így a teljes kép kirajzolása 1/25 sec alatt történik.

A képernyő elektronsugár által végigpásztázott egy-egy sávját nevezik **rasztersor**nak.

A képernyőtartalom vibrálását, villódzását elkerülendő, a világító festékanyag **utánvilágítási ideje** hosszabb, mint a közönséges TV készülékeké.

A képernyővel kapcsolatos jellemzők többsége állítható, választható. Ilyen jellemzők például:

- a **kurzor alakja**, tulajdonsága; a karakteres kurzor két formája használatos általában: az aláhúzás(a karaktermátrix 9., vagy 10.sorában), vagy a blokk forma, amely egy karakternyi hely inverzben való megjelenítése; többnyire beállítható a kurzor villogása is; grafikus képernyőn különféle grafikákat(nyíl, kéz, stb.) használnak kurzorként.

- a **karakterek tulajdonságai**(attributumai), mint például a karakter aláhúzása, villogtatása, vagy inverz formában történő kiírása; lehetőség van a karakter kiemelt(fényesebb) formájú kiíratására is.

- a **szín megválasztása**, amely három színnek(háttér, előtér és a keret színének) a megadását jelenti.

- a **képernyő görgetése**(scrolling), **lapozása**(paging); a képernyő tartalmának soronkénti lefelé, vagy felfelé történő mozgatása a görgetés; ez történhet karaktersoronként(character scroll), vagy folyamatosan(soft scroll). A képernyő tartalmának képernyőnkénti váltása a lapozás, amely lehet teljes képernyős(full page = 24 sor), vagy félképernyős(half page).

- a **képernyő megosztása**(split), amelynek hatására két részre osztható a képernyő és a két részben más és más rész jeleníthető meg például egy szövegállományból.

A képernyőn megjelenő karakterek formáját a **karakter generátor**ok szabják meg. A karakter generátor egy olyan vezérlő, amely minden karakter raszterpontonkénti leírását tartalmazó ROM tároló segítségével, a kiíratandó karaktert megjeleníti a képernyőn. A karakter-ROM raszterpont-mátrix formájában (1-0-ások sorozatával) tárolja azt, hogy mely pontot kell kivilágítani és melyeket nem. A kiíratandó karakter ASCII kódja egy mutató a ROM tároló azon helyére, ahol a karakterhez tartozó pont-mátrix tárolása elkezdődik. Az itt lévő jelsorozatot egy léptető regiszterrel soros jellé alakítja az egység és ezt a jelsorozatot küldi a monitor vezérlésére.

A ROM tároló cseréjével egyszerű módon megoldható a képernyőn megjelenő karakterek kiírási fomája. Ha grafikus képernyőforma használatos(tehát pontonként programozható a kiíratás), akkor akármilyen betűforma kialakítható és alkalmazható.

A képernyő működésével kapcsolatos alapjellemzők(sorok, oszlopok száma, felbontás finomsága, a képernyőtartalom tárolásának helye a memóriában, stb.) értékét a vezérlőkártyákon beállítható **video üzemmódok** határozzák meg. A video üzemmódokat sorszámokkal jelölik, amelyek száma ma már, a fejlődés révén, több tucatnyira növekedett. A mai gépekben lévő monitorvezérlő kártyák(többnyire VGA, SVGA típusúak) ismerik a korábbi video üzemmódokat is és így a régebbi monitorokhoz készült programok továbbra is használhatók maradnak.

8.2.KÁRTYATÍPUSOK

A következőkben a különböző monitor típusokhoz, pontosabban vezérlőkártyákhoz tartozó legjellemzőbb adatokat ismertetjük. A ma eladott mikroszámítógépek monitor vezérlőkártyái, mint ahogy ezt már korábban említettük, VGA, SVGA típusúak, amelyek azonban biztosítják mindazon üzemmódokat, amelyek a korábbi monitortípusokat(Hercules, CGA, EGA, stb.) használó programok futtatását is lehetővé teszik.

8.2.1.Alaptípusok

A következő 8-1.táblázatban az egyes monitor vezérlőkártyatípusok legjellemzőbb tulajdonságait foglaltuk össze, az összehasonlíthatóság érdekében. A használható video üzemmódok teljes listája a kártyák dokumentációjában található meg.

8-1.táblázat: *Vezérlőkártyák jellemző tulajdonságai*

monitor típus	formátum	jellemző video üzemmódok	jellemző felbontás	színek száma	min.RAM [KB]
MDA	kar.	7	720x350	2	256
HGC	kar./graf.	7	720x350	2	256
HGC+	kar./graf.	7+	720x400	2	256
CGA	kar./graf.	0,1,2,3,4,5,6	640x200	16/4	256
EGA	kar./graf.	0*,1*,2*,3*,D,E,F,10	640x350	16	256
MCGA	kar./graf.	11,13	640x480	256	256
VGA	kar./graf.	0+,1+,2+,3+,7+,12	640x480	256	256
SVGA	kar./graf.	0+,1+,2+,3+,7+,50-62	800x600	256	512,1024

A táblázatban említett monitortípusok közül az MDA, CGA, EGA, MCGA típusok jellemzőit az IBM definiálta, a többi vezérlőkártyatípust más gyártók definiálták és vált általánosan használttá.

A gyakorlati felhasználások során sokszor alkalmaznak egyszerre két monitort, egyet a szöveges, egyet pedig a grafikus információk megjelenítésére. Sajnos a videomemória tárolási helye miatt, csak egy monokróm(MDA, HGC) és egy színes(CGA, EGA, VGA, SVGA) monitor használható együtt, kivéve az EGA típust, amely esetben mindkettő EGA monitor lehet.

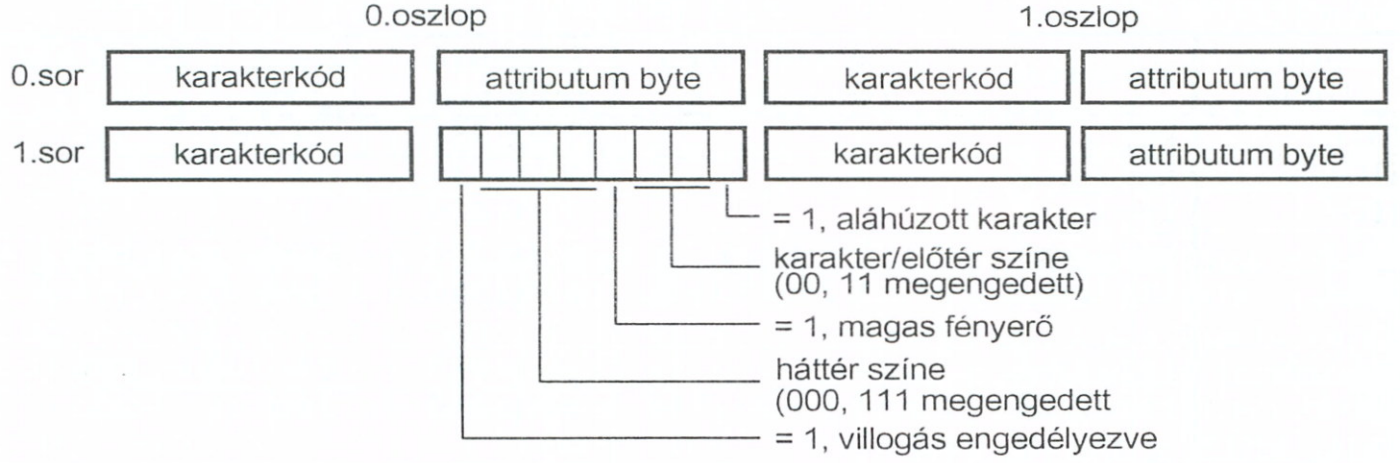
8.2.2.MDA kártya

Az MDA(Monochrom Display Adapter) kártya monokróm, szöveges monitorok támogatására szolgál.

Az alkalmazott **video üzemmód** és jellemzői:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
7	szöveges	80x25	9x14		2

A képernyőtartalom tárolása a videomemória területén, karakter pozíciónként 2 byte-ot igényel, amelyből az első a karakter ASCII kódját tartalmazza, a második byte az ú.n. **'attributum' byte**, amely a karakter megjelenítésére vonatkozó információkat tárolja. A fentieknek megfelelő **tárolási módot** és az attributum byte tartalmát a 8-1.ábra mutatja be.



8-1.ábra: *MDA kártya képtartalom tárolási módja*

Az attributum byte-ban megadott 000 háttérszín a fekete szín és ehhez természetszerűleg csak 11 értékű karakterszín társulhat és fordítva is igaz.

Egy tetszőleges **karakter relatív címe** a videomemória kezdetéhez viszonyítva:

$$kar.címe = 160*sor + 2*oszlop$$

Egy teljes képernyő tartalmának tárolásához 80x25x2=4000 [byte] **memória szükséges**.

A **videomemória kezdőcíme**: B0000h

8.2.3.HGC kártya

A HGC(Hercules Graphic Card) vezérlőkártya mind szöveges, mind grafikus megjelenítéshez használható, de csak fekete-fehér üzemmódban.

Az alkalmazott **video üzemmódok** és jellemzőik:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
7	szöveges	80x25	9x14		2
7	grafikus			720x350	2
7+	szöveges	80x25	9x16		2
7+	grafikus			720x400	2

A képernyő **tárolási formája:**

Szöveges formátum mellett megegyezik az MDA tárolási módjával, azaz karakterenként 2 byte-ot használ fel, az első byte-on a karakter ASCII kódját, a második byte-on az attributum byte-ját tárolva.

Az attributum byte tartalma:

7.bit	villogtatás engedélyezése,
6-4.bit	háttér színe, láthatósága, csak a 000 és 111 kódok használhatók,
3.bit	karakter kiemelt fényerővel,
2-1.bit	karakter színe, láthatósága, csak a 00 és 11 kódok megengedettek,
0.bit	aláhúzás.

Grafikus formátum alkalmazásakor egy-egy képpont tárolásához 1 bitet használnak fel. A képsorokat(720/8=90 byte) váltakozva a tárolóhely négy külön területén(blokkjában) tárolják. A tárolás módja a 4 szegmensben:

0.,4.,8., ...	sorok 90-90 byte-ja,	B0000h címtől,
1.,5.,9., ...	sorok 90-90 byte-ja,	B2000h címtől,
2.,6.,10.,..	sorok 90-90 byte-ja,	B4000h címtől,
3.,7.,11.,..	sorok 90-90 byte-ja,	B6000h címtől.

Az egyes tárolási módok mellett a **karakterek, illetve képpontok relatív címe** a videomemória kezdetéhez viszonyítva, az alábbiak szerint adható meg.

Szöveges üzemmód esetén:

kar.címe = 160*sor + 2*oszlop

Grafikus üzemmód esetén, az (x,y) képpont(pixel) helye:

pxl byte címe = 8192*(y mod 4) + 90*int(y/4) + int(x/8)

bitcím = 7 - (x mod 8)

Egy teljes képernyő tartalmának tárolásához szöveges üzemmódban 80x25x2=4000 [byte] **memória szükséges**. Grafikus üzemmódban (720/8)*350=31500 [byte] tárolóhely szükséges.

A **videomemória kezdőcíme**: B0000h. A lefoglalt terület: B0000-B7FFF-ig tart.

8.2.4.CGA kártya

A CGA(Color Graphic Adapter) vezérlőkártya mind szöveges, mind grafikus megjelenítéshez használható, fekete-fehér és színes üzemmódban.

Az alkalmazott **video üzemmódok** és jellemzőik:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
0,1	szöveges	40x25	8x8	320x200	16
2,3	szöveges	80x25	8x8	640x200	16
4,5	grafikus	40x25	8x8	320x200	4
6	grafikus	80x25	8x8	640x200	2

A képernyő **tárolási formája:**

Szöveges formátum mellett megegyezik az MDA tárolási módjával, azaz karakterenként 2 byte-ot használ fel, az első byte-on a karakter ASCII kódját, a második byte-on az attributum byte-ját tárolva.

Az attributum byte tartalma fekete-fehér és színes üzemmód mellett:

fekete-fehér üzemmódban:

7.bit	villogtatás engedélyezése, vagy a háttérszín erősségének beállítása(a választást az üzemmód regiszter 5.bitje szabja meg),
6-4.bit	háttér színe, láthatósága,
3.bit	karakter kiemelt fényerővel,
2-1.bit	karakter színe, láthatósága,
0.bit	aláhúzás(nem minden esetben).

színes üzemmódban:

7.bit	villogtatás engedélyezése, vagy a háttérszín erősségének beállítása(a választást az üzemmód regiszter 5.bitje szabja meg),
6-4.bit	háttér színe,
3.bit	karakter kiemelt fényerővel,
2-0.bit	karakter színe.

A színes üzemmódban a karakterek színkódjait és a szín intenzitásának (erősségének) értékét az alábbi 8-2.táblázat tartalmazza:

8-2.táblázat: *Színek kódtáblázata*

intenzitás	színkód	szín	intenzitás	színkód	szín
0	000	fekete	1	000	sötétszürke
0	001	kék	1	001	világoskék
0	010	zöld	1	010	világoszöld
0	011	türkiz	1	011	világostürkiz
0	100	vörös	1	100	piros
0	101	lila	1	101	világoslila
0	110	barna	1	110	sárga
0	111	világosszürke	1	111	fehér

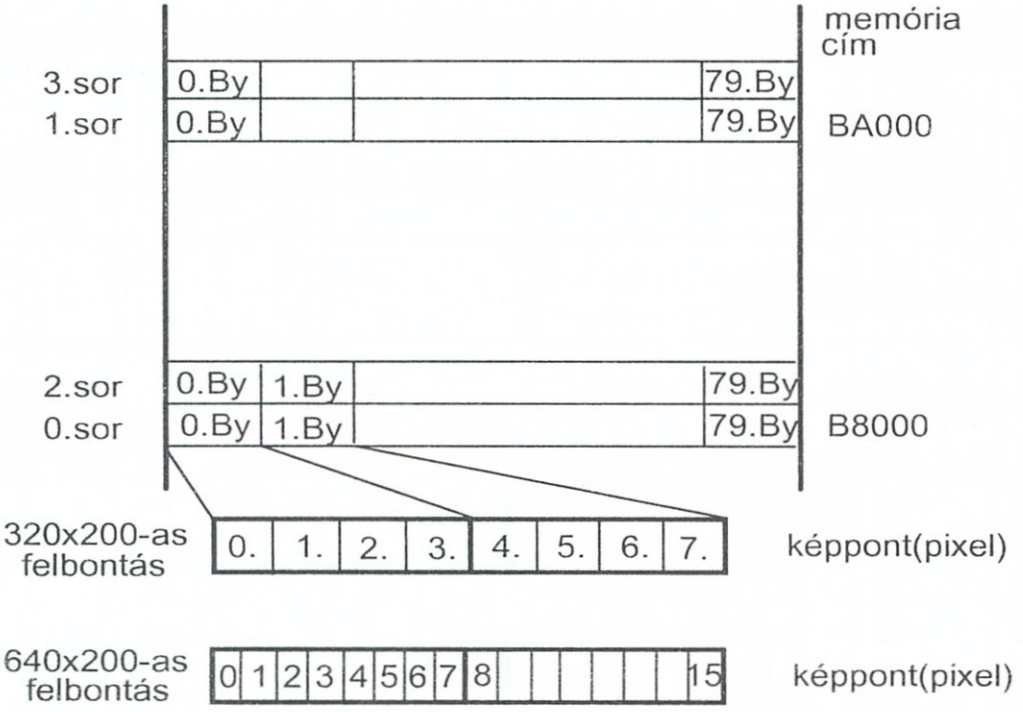
Grafikus formátum alkalmazásakor egy-egy képpont tárolásához 1 bitet (640x200 felbontás), vagy 2 bitet(320x200 felbontás) használnak fel. A képsorokat váltakozva a tárolóhely két külön területén(blokkjában) tárolják. A tárolás módját, 320x200-as felbontást feltételezve, a 8-2.ábra mutatja be.

A nagyobb felbontású, 640x200 pontos megjelenítés mellett a színek meghatározására nincs mód; a képpont bitjének 1-es értéke a fehér, 0-ás értéke a fekete színt képviseli.

A kisebb finomságú felbontás(320x200 pontos) mellett, a háttér színe 16 színből választható ki, az előtér színe két színpaletta 3-3 színéből választott érték. A képpont 2 bites leírója a paletta valamelyik színét, vagy a háttér színét jelöli ki. A háttér színének, valamint a színpalettának a beállítása külön regiszterben történik. A színpaletták színeit az alábbi 8-3.táblázat tartalmazza:

8-3.táblázat: *Színpaletták színkódjai*

kód	0.paletta	1.paletta
00	háttérszín	háttérszín
01	piros	türkiz
10	zöld	lila
11	sárga	fehér



8-2.ábra: *CGA kártya grafikus tárolási formája*

Az egyes tárolási módok mellett a **karakterek, illetve képpontok relatív címe** a videomemória kezdetéhez viszonyítva, az alábbiak szerint adható meg.

Szöveges üzemmód esetén:

kar.címe = 160*sor + 2*oszlop

Grafikus üzemmód esetén, az (x,y) képpont(pixel) helye:

320x200 pontos felbontás:

pxl byte címe = 8192*(y mod 2) + 80*int(y/2) + int(x/4)

bitcím = 6 - 2*(x mod 4)

640x200 pontos felbontás:

pxl byte címe = 8192*(y mod 2) + 80*int(y/2) + int(x/8)

bitcím = 7 - (x mod 8)

Egy teljes képernyő tartalmának tárolásához szöveges üzemmódban 40x25x2=2000 [byte], illetve 80x25x2=4000 [byte] **memória szükséges**. Grafikus üzemmódban (640/8)*200=16000 [byte] tárolóhely szükséges.

A **videomemória kezdőcíme**: B8000h és BA000h a két blokkra. A lefoglalt terület: B8000-BBFFF-ig tart.

8.2.5.EGA kártya

A EGA(Enhanced Graphic Adapter) vezérlőkártya mind szöveges, mind grafikus megjelenítéshez használható, fekete-fehér és színes üzemmódban, az egyszerűbb vezérlőkártyák üzemmódjait is lehetővé téve.

Az alkalmazott **video üzemmódok** és jellemzőik:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
0*,1*	szöveges	40x25	8x14	320x350	16
2*,3*	szöveges	80x25	8x14	640x350	16
4,5	grafikus	40x25	8x8	320x200	4
6	grafikus	80x25	8x8	640x200	2
7	szöveges	80x25	8x14	720x350	2
D	grafikus	40x25	8x8	320x200	16
E	grafikus	80x25	8x8	640x200	16
F	grafikus	80x25	8x14	640x350	2
10	grafikus	80x25	8x14	640x350	16

A képernyő **tárolási formája:**

Szöveges formátum mellett megegyezik az MDA, HGC, CGA karakteres tárolási módjával, azaz karakterenként 2 byte-ot használ fel, az első byte-on a karakter ASCII kódját, a második byte-on az attributum byteját tárolva.

Az *attributum byte* tartalma fekete-fehér és színes üzemmód mellett:

fekete-fehér üzemmódban:

7.bit	villogtatás engedélyezése, vagy a háttérszín erősségének beállítása(a választást az üzemmód regiszter 5.bitje szabja meg),
6-4.bit	háttér színe, láthatósága,
3.bit	karakter kiemelt fényerővel, vagy a karaktergenerátor választása,
2-0.bit	karakter színe, láthatósága.

színes üzemmódban:

7.bit	villogtatás engedélyezése, vagy a háttérszín erősségének beállítása(a választást az üzemmód regiszter 5.bitje szabja meg),
6-4.bit	háttér színe,
3.bit	karakter kiemelt fényerővel, vagy a karaktergenerátor választása,
2-0.bit	karakter színe.

Grafikus formátum alkalmazásakor a képernyő tartalmának tárolása már lényegesen összetettebb, mint a CGA kártyánál.

- A kártya CGA 4, 5, 6-os üzemmódjainál a tárolás módja ugyanaz, mint a CGA kártya esetében.
- Az EGA kártya F üzemmódjánál minden képponthoz 2 bit tartozik, amely segítségével egyrészt a karakter láthatósága, másrészt a villogást, vagy a kiemelt fényerőt lehet jelezni. Azaz minden képponthoz két bitsík tartozik.

  Ennek megfelelően a tárolási helyigény:

  640x350 = 224000 [bit] = 28000 [byte]

  Így két bitsík tárolásához kb. 64 KB **memória szükséges**.
- Az EGA 10-es üzemmódjánál 4, vagy 16 szín tárolásához 2, illetve 4 bitsík kell,

  azaz így összesen kb. 128 KB **memória szükséges**.

  A szín beállítása, a színkód által, a paletta regiszteren keresztül történik.

A **videomemória kezdőcíme**: a 0*, 1*, 2*, 3*, 4, 5, 6-os üzemmódoknál a B8000h, a 7-es HGC üzemmódnál a B0000h cím, míg a D, E, F, 10 -es EGA üzemmódoknál az A0000h cím.

8.2.6.MCGA kártya

Az MCGA(Multi Color Graphic Adapter) kártya, amely az IBM PS/2 mikroszámítógépek egyes típusaihoz készült, nagy mértékben hasonlít a CGA kártya felépítéséhez.

Az alkalmazott **video üzemmódok** és jellemzőik:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
0,1	szöveges	40x25	8x8	320x200	16
2,3	szöveges	80x25	8x8	640x200	16
4,5	grafikus	40x25	8x8	320x200	4
6	grafikus	80x25	8x8	640x200	2
11	grafikus	80x30	8x16	640x480	2
13	grafikus	40x25	8x8	320x200	256

A képernyő **tárolási formája:**

Szöveges formátum nagyjából megegyezik a CGA szöveges tárolási módjaival, avval a különbséggel, hogy

- a 4 bites színkód, egy színkódtár címzésével jelöli ki a színárnyalatot,
- az egyes attributum bitek maszkolhatók,
- 8 karaktergenerátor érhető el, egyenként 512 karakterforma meghatározásával.

Grafikus formátumok a CGA(4, 5, 6-os) üzemmódoknál megegyeznek a CGA kártyánál leírtakkal.

- A 11-es video üzemmódnál, a 640x480-as felbontású fekete-fehér megjelenítésnél, a képernyő tartalmának a tárolása képpontonként 1 bitet igényel. A képpontok színe az ú.n. színkód regiszter, vagy a színkiválasztó regiszter által meghatározott.
- A 13-as üzemmód 320x200 pontos, 4-színű felbontásánál a képpont tárolt bitjeihez a színkiválasztó regiszter 2 bitje tevődik hozzá, amelyek együttesen jelölik ki a színkód regisztert.
- A 13-as üzemmód 320x200 pontos 256-színű felbontásánál minden egyes képpont színét 1 byte tárolja.

8.2.7.VGA, SVGA kártya

A VGA(Video Graphic Array) és az SVGA(Super VGA) kártyák előírásait nem az IBM definiálta, de jellemzői miatt, igen széles körben elterjedtek.

Az alkalmazott VGA **video üzemmódok** és jellemzőik:

video üzemmód	formátum	jellemző karakteres felbontás	karakter méret	jellemző grafikus felbontás	színek száma
0+,1+	szöveges	40x25	9x16	360x400	16
2+,3+	szöveges	80x25	9x16	720x400	16
7+	szöveges	80x25	9x16	720x400	2
11	grafikus	80x30	8x16	640x480	2
12	grafikus	80x30	8x16	640x480	16
13	grafikus	40x25	8x8	320x200	256

A vezérlőkártyák az említett üzemmódokon kívül, többnyire még további lehetőségeket biztosítanak a felhasználónak.

A képernyő **megjelenítési formái:**

A **szöveges** megjelenítés a CGA és az EGA megoldások továbbfejlesztésével alakult ki. Az ú.n. paletta és színkód regiszterek felhasználásával, az alkalmazott színkészletek(4-16-féle) könnyen válthatók. Emellett egyidőben 8-féle karakterkészlet is használható.

Grafikus üzemmódok esetében, a VGA kártya többféle új lehetőséget biztosít a megjelenítésben. A 256-féle színnel történő megjelenítéshez, hasonlóan az EGA kártyához, minden képpont színének tárolásához 1 byte-ot használ fel a rendszer.

Az **SVGA vezérlőkártya** a VGA továbbfejlesztett változata, amelynél a felbontás mértéke és az alkalmazható színek száma növekedett.

Szöveges megjelenítéseknél 25, 30, 43, 60 sor és soronként 80, vagy 132 karakter jeleníthető meg.

Grafikus megjelenítési formáknál 800x600, 1024x768, 1280x1024 pontos felbontások használhatók, 16, 256, 32768 szín(árnyalat) felhasználásával.

8.3.BIOS SZINTŰ KÉPERNYŐKEZELÉS

A képernyővel kapcsolatos, az előzőekben említett, vagy azokat kiegészítő további tevékenységek a BIOS **10h-es szoftver megszakítási vektorán** keresztül végeztethetők el. A választott funkció sorszámát az AH regiszterben kell elhelyezni.

Felsorolásszerűen, a következő funkciók vehetők igénybe:

00h	video üzemmód beállítása; a különböző karakteres, illetve grafikus képernyőkezelési formák választása;
01h	kurzor alakjának meghatározása;
02h	kurzor helyének beállítása;
03h	kurzor helyének (és formájának) kiolvasása;
04h	fényceruza helyének kiolvasása;
05h	aktuális képernyőoldal kiválasztása, ha a vezérlőkártya lehetőséget ad több képernyőoldal kezelésére;
06h	aktuális képernyőoldal tartalmának görgetése(scroll) felfelé;
07h	aktuális képernyőoldal tartalmának görgetése(scroll) lefelé;
08h	a kurzor aktuális helyén lévő karakternek és színének kiolvasása;
09h	karakter és színének beírása a kurzor aktuális helyére;
0Ah	karakterek beírása a kurzor aktuális poziciójától;
0Bh	00h alfunkció: keret- és háttérszín kiválasztása; 01h alfunkció: színpaletta választása 320x200-as grafikus üzemmódnál;

0Ch	képpont színének beírása grafikus üzemmódban;
0Dh	képpont színének kiolvasása grafikus üzemmódban;
0Eh	karakter(ek) beírása a kurzor aktuális poziciójától grafikus üzemmódban;
0Fh	video üzemmód kiolvasása;
10h	EGA, VGA kártyák paletta regisztereinek beállítása/kiolvasása;
11h	EGA, VGA kártyák karaktergenerátorainak kezelése;
12h	EGA, VGA kártyák konfigurációs értékeinek kezelése;
13h	karakterlánc kiírása a kurzor aktuális poziciójától;
1Ah	két monitor használatához kapcsolódó beállítások;
1Bh	VGA kártyák állapotjellemzőinek lekérdezése;
1Ch	VGA kártyák állapotjellemzőinek elmentése, visszatöltése.

8.4.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Ismertesse a számítógépi monitorok jellemző működési elvét! Miért lényeges a képernyő utánvilágítási ideje?
2. A képernyő milyen jellemzőit lehet módosítani szoftver úton?
3. Mire szolgál a karaktergenerátor és mi a működési elve?
4. Sorolja fel az alaptípusnak minősülő vezérlő kártyákat!
5. Milyen maximális felbontási lehetőséggel rendelkezik a Hercules (HGC) kártya?
6. Milyen maximális felbontási lehetőséggel rendelkezik az CGA kártya?
7. Milyen maximális felbontási lehetőséggel rendelkezik az EGA kártya?
8. Milyen maximális felbontási lehetőséggel rendelkezik az VGA kártya?
9. Milyen maximális felbontási lehetőséggel rendelkezik az SVGA kártya?
10. Milyen párosításokkal alakítható ki két-monitoros rendszer?
11. Milyen párosításban használható két grafikus monitor?
12. Ismertesse az MDA monokróm monitor felbontási lehetőségeit, tárolási módját! Ismertesse az 'attributum' byte tartalmát!
13. Milyen jellemző felbontási értékekkel rendelkezik a CGA kártya karakteres és grafikus üzemmódban?
14. Milyen a CGA kártyák képernyőtartalom tárolása karakteres és grafikus üzemmódban? Milyen információkat tárol az 'attributum' byte karakteres és grafikus üzemmódban?
15. Milyen felbontási lehetőségekkel rendelkezik a HGC kártya karakteres és grafikus üzemmódban?
16. Milyen a HGC kártyák képernyőtartalom tárolási módja karakteres és grafikus üzemmódban? Milyen adatokat tárol az 'attributum' byte karakteres és grafikus üzemmódban?
17. Milyen memóriacímen tárolja a képernyőtartalmat a HGC kártya?
18. Milyen jellemző felbontási lehetőségekkel rendelkezik az EGA vezérlő kártya karakteres és grafikus üzemmódban?

19. Hogyan érhetők el a vezérlő kártya regiszterei és funkciói?
20. Milyen jellemző megjelenítási módokkal rendelkezik az MCGA kártya? Mely gépekhez fejlesztették ki az MCGA kártyákat? Milyen memória címtartományban helyezkedik el a video RAM?
21. Milyen jellemző felbontási móddal rendelkezik a VGA kártya? Milyen memória címtartományban helyezkedik el a video RAM?
22. MIlyen jellemző felbontási módokkal rendelkezik az SVGA kártya? Milyen címtartományban helyezkedik el a video RAM?
23. Milyen címtartományban található a videomemória általában?
24. Melyik BIOS megszakításon keresztül lehet elérni a monitorvezérlőt és milyen főbb funkciókat lehet igénybe venni?

9. FEJEZET

EGYÉB BEVITELI, KIVITELI ESZKÖZÖK

9.1.BILLENTYŰZET, EGÉR

A mikroszámítógépek beviteli eszközei közül a billentyűzet és az egér az, amelyeket a leggyakrabban használnak a felhasználók. Nagyobb mennyiségű adat bevitelére a billentyűzet nem alkalmas, ezért ilyen célra más eszközöket (pl. hajlékonylemezt) kell használni.

9.1.1.Billentyűzet

A billentyűzet a mikroszámítógép közvetlen irányítására, kisebb mennyiségű adat bevitelére használható. Az alkalmazott változatai a 84, illetve a 101/102 gombos billentyűzet. A 101/102 gombos változatnál külön számbillentyűzet van a numerikus adatok könnyebb bevitelére.

A billentyűk funkcionálisan több csoportba sorolhatók:

- a billentyűzet legnagyobb részét, az írógép billentyűzethez hasonló felépítésű, **alapkarakterek**(betűk, számjegyek, speciális karakterek) **billentyűi** alkotják;
- a numerikus adatok bevitelét könnyíti meg a **számbillentyűzet**, amely a billentyűzet jobboldalát foglalja el;
- a (101/102 gombos) billentyűzet felső részét foglalják el a **funkcióbillentyűk**, amelyek hatása az egyes felhasználói, alkalmazói programoknál más és más;
- az alapkarakterek és a számbillentyűk között helyezkednek el a **képernyőkezelő billentyűk**, amelyek a kurzor mozgatására, a képernyő tartalom mozgatására szolgálnak.

Egy másik csoportosítási lehetőség:

- karakterbillentyűk,
- funkcióbillentyűk,
- váltóbillentyűk(shift, ctrl, alt), amelyek egy másik billentyűvel együtt lenyomva módosítják az eredeti billentyű hatását,
- kapcsolóbillentyűk(CapsLock, NumLock, ScrollLock), amelyek tartósan módosítják egyes billentyűcsoportok hatását.

A billentyűzet működése

A billentyűzet saját vezérlővel rendelkezik, amely nem csak adatokat tud küldeni az alaplapon lévő billentyűzet illesztő felé, hanem parancsokat is tud fogadni.

A billentyűzet megszakítási rutin valamelyik billentyű lenyomásakor annak billentyűkódját(**scan code**) is átküldi a billentyűzet pufferbe. A lenyomott billentyű azonosítására egy sorszám szolgál és nem a karakter ASCII kódja, mivel ugyanahhoz a billentyűhöz több karakter is tartozik.

Minden billentyűlenyomáskor a vezérlő 2 byte-ot küld át a pufferbe. Azoknál a billentyűkombinációknál, amelyekhez ASCII kód rendelhető, az átküldött byte-ok közül az egyik az ASCII kódot, a másik a billentyűkódot(scan code) tartalmazza. Azoknál a kombinációknál, amelyek valamely funkcióbillentyűhöz tartoznak, az átküldött byte-ok közül az első **0** értéket, a második a karakterkódot foglalja magában.

Tetszőleges ASCII kódérték(a 0 kivételével) bevihető, az ALT billentyű lenyomásával és vele egyidőben, a számbillentyűzeten a decimális kódérték bebillentyűzésével. Mivel a 0 kódérték a funkcióbillentyűk jelzésére van fenntartva, ezért azt nem lehet bebillentyűzni.

A billentyűzet BIOS szintű kezelése

A billentyűzetvezérlő az átküldött ASCII kódokat és billentyűkódokat egy 32 byte-os, ciklikus kezelésű pufferben helyezi el. A puffer kezdetét és végét jelző mutatók folyamatosan körbejárnak, azaz ha elérik a puffer végét, akkor az elejére ugranak.

A pufferből a BIOS **16h-os megszakítás vektorának** segítségével lehet kiolvasni az adatokat.

Az igénybe vehető funkciók:

00h	karakter- és billentyűkód beolvasása a pufferből; ha a puffer üres, akkor addig várakozik, amíg a billentyűzetről karakter kerül a pufferbe;
01h	puffer ellenőrzése: van-e karakter a pufferben? ha van, akkor annak kódjával tér vissza; ha üres a puffer, akkor nem várakozik;
02h	billentyűzet állapotbyte-jának a lekérdezése;
03h	billentyűzet ismétlési jellemzőinek beállítása;
05h	karakter- és billentyűkód elhelyezése a pufferben;
10h	a 00h-ás funkció megvalósítása 101/102 gombos billentyűzet esetében;
11h	a 01h-es funkció megvalósítása 101/102 gombos billentyűzet esetében;
12h	a váltó, módosító billentyűk állapotának lekérdezése 101/102 gombos billentyűzet esetében.

9.1.2.Egér

Az egér(mouse) az utóbbi idők grafikus képernyőinek kedvelt 'mutató' eszköze, amivel a képernyő különböző helyein lévő objektumokra lehet rámutatni és a pozició alapján, az egér billentyűivel vezérelni lehet az objektumhoz kapcsolódó rutint.

Amíg az egérhez hasonló funkciójú botkormányhoz(joystick) és fényceruzához (light-pen) beépített BIOS rutinok tartoznak, addig az egér kezelő rutinjait önálló és külön betöltendő meghajtó(driver) tartalmazza. Ezt a meghajtót az egér használata előtt betöltve, az állandó jelleggel a memóriában marad a gép kikapcsolásáig.

Működés szempontjából, az egérnek alapvetően háromféle típusát használják:

- **mechanikus vezérlésű** egérnél két, egymásra merőleges görgőrendszer mozgását érzékeli a berendezés és ennek jeleit továbbítja a gép soros portja felé;
- **optikai vezérlésű** egér esetében, egy hálózatos rajzolatú egér alátéten mozgatva az egeret, egy LED fényének a visszaverődését érzékeli a berendezés és továbbitja a processzor felé;
- **opto-mechanikai vezérlésű** egér működése hasonló a mechanikus megoldásúéhoz, avval a különbséggel, hogy az egér elmozdulásával együtt elforduló hasított korongon fényt átbocsátva, az így kapott fényimpulzusokat érzékeli a berendezés.

A szellemesen 'hanyatt-egér'-nek nevezett 'track-ball' esetében a felhasználó magát az elmozdulást közvetítő golyót mozgatja az ujjaival.

Az egér és a processzor közötti kapcsolat a soros porton keresztül jön létre. Bizonyos időközönként, a soros bemeneten keresztül 3, vagy 5 byte-ot(protokoll) küld az egér érzékelője.

A három byte tartalma:

1.byte	az egér billentyűinek helyzete, állapota;
2.byte	az utolsó 100ms alatti **x**-irányú elmozdulás előjeles értéke;
3.byte	az utolsó 100ms alatti **y**-irányú elmozdulás előjeles értéke.

Az 5 byte-os protokoll esetében, 2-2 byte tartalmazza az x, illetve az y irányú elmozdulás előjeles értékét.

Az egér kezelése, megszakításkérés alapján, a **33h-as** BIOS **megszakítási rutinon keresztül** történik. Az egyes funkciók megvalósítására a külön betöltött meghajtóprogram(driver) szolgál.

9.2.NYOMTATÓK

9.2.1.Általános jellemzők

A nyomtató a mikroszámítógépek azon kimeneti eszköze, amely az ember számára közvetlenül olvasható, értelmezhető eredményt szolgáltat. A mikroszámítógépek mellett kisebb teljesítményű nyomtatókat használnak, amelyeknél fontos követelmény a jó minőségű nyomtatási kép.

A leggyakoribb nyomtatók az alábbi csoportokba sorolhatók:

- **karakterenkénti nyomtatók**(teljes karaktert nyomtatók, mátrixnyomtatók), amelyek egy-egy karaktert kiírva a nyomtatófejet egy pozicióval jobbra, vagy balra viszik;
- **sornyomtatók**, amelyek egyszerre egy teljes sor összes karakterét kiírják.

Emellett beszélhetünk karakter- és grafikus nyomtatókról. Ez utóbbiak esetében pontonként történhet a nyomtatás.

9.2.2.Nyomtatótípusok

a.)Karakternyomtatók

A *mikroszámítógépek* mellett a leggyakoribb karakternyomtatók a ma már ritkán használt margarétakerekes(daisy-wheel printer) nyomtatók voltak. A margarétakerekes nyomtatók esetében, a karakterek képei egy forgó, hasított tárcsa(olyan, mint a margaréta virág) szélén helyezkednek el. A forgó tárcsa mögött elektromágnessel működtetett kalapács helyezkedik el, amely a szükséges időpontban a tárcsa valamelyik szirmát az előtte lévő papírhoz nyomja. A betűt tartalmazó tárcsa és a papír között található a festékszalag.

A margarétakerekes nyomtatók igen jó minőségű kiírást eredményeznek, különösen karbonszalag alkalmazásával. A nyomtatás sebessége: 30-50 kar/sec.

A *nagygépek* mellett forgóláncos, vagy forgódobos sornyomtatókat használnak. A forgóláncos nyomtatónál egy lánc felületén ismétlődik többször is a teljes karakterkészlet, amelyet a kellő időpillanatban egy kalapács nyom a papírhoz festékszalagon keresztül. A forgódobos nyomtatók elve hasonló, csak annál egy dob felületén helyezkednek el a karakterek.

b.)Mátrixnyomtatók

A mátrixnyomtatók a legáltalánosabban használt nyomtatók, viszonylagos olcsóságuk miatt, különösen kisebb teljesítményigény mellett.

A nyomtatófejben, egy oszlopban 9, vagy 24 nyomtatótű található, amelyek egyenként vezérelhetők. A karakterek alakját egy 5x7-es, 9x11-es, vagy 18x23-as pontmátrix pontjaival rajzolja ki a nyomtató, egyidőben mindig csak egy oszlop pontjait nyomtatva. Az egy sorba írható pontok(dot) száma adja a felbontás és így a nyomtatás finomságát. Az alkalmazott felbontási pontszám 60-144 dpi[dot/inch] között mozog, a leggyakoribb érték 72 dpi.

Az egyes nyomtatóknál használt NLQ(near letter quality), vagy LQ(letter quality) nyomtatási formánál a jobb minőség, a folytonos vonal látszatának elérése érdekében, többszörös nyomtatást valósítanak meg 1/2-1 pontnyi eltolással az egyes nyomtatások között.

A nagyobb tűszámú nyomtató használata természetszerűleg lassúbb nyomtatást eredményez a nyomtatási kép javulása mellett.

A nyomtatók sebessége 100-200 kar/s nagyságrendű, amelynek értéke NLQ, LQ minőség mellett a harmadára, negyedére eshet vissza.

c.)Tintasugaras nyomatók

A tintasugaras(ink-jet), buborék(bubble-jet) sugaras nyomtatók igen jó minőségű nyomtatási képet eredményeznek, szinte zajtalanul és alacsonyabb áron, mint a lézernyomtatók.

A nyomtatási kép több tucat fúvókán keresztül kilövelt, porlasztott apró tintacsepp hatására alakul ki a papiron. A nyomtatóval jó minőségű grafikus nyomtatás is elérhető, akár színesben is.

A nyomtatási finomság szokásos értéke 300x300 dpi; a sebesség pedig igen széles skálán, 20-200 kar/s között mozog.

d.)Lézernyomtatók

A mikroszámítógépek környezetében a legjobb minőségű nyomtatási képet a lézernyomtatók adják, amelyek ára ma már a hasonló teljesítményű tintasugaras nyomtatók árának durván kétszerese.

A lézernyomtatók **működési elve** hasonlít az elektrosztatikus elvű másolók működéséhez. A nyomtatóban egy fényérzékeny, töltéstároló képességű bevonattal rendelkező forgó henger van, amelyet kb. 1000 V-ra feltöltenek. A forgó henger felületét a felbontási finomságnak megfelelő sűrűséggel, az alkotója mentén végigpásztázza egy lézersugár, amit a nyomtatandó kép jelével modulálnak. Ahol fénysugár éri a hengert, ott a fénysugár erősségének megfelelő mértékben a henger felülete elveszti a töltését. A forgó henger felülete elhalad a finom festékport tartalmazó kazetta előtt, amelyből a töltéssel arányos mennyiségben festékpor tapad a henger felületére, amelyet átad, rányom a hozzányomott papír felületére. A papír fűtött hengerek között halad át, amelyek a festékanyagot 'ráégetik' a papírra. A forgó henger felületét a nyomtatás után letisztítják és így alkalmas lesz a következő lap nyomtatására.

A nyomtatás előtt a teljes lap tartalmát pontokra bontva elő kell állítani, hogy a lézersugarat megfelelően vezérelni lehessen. A **nyomtatási kép előállítása** többféle módon történhet:

- A nyomtató **saját processzorral rendelkezik**, amely a számítógéptől kapott kiíratandó adatokat átalakítja a szükséges ú.n. bittérképes(bitmap) formába, azaz elkészíti a kiíratandó lap teljes képét pontokra bontva. Ennek a módszernek előnye, hogy a számítógép és a nyomtató között viszonylag alacsony(2000-8000 bit/s) adatátviteli sebesség szükséges, így a soros(RS-232-es) csatlakozón keresztül átküldhetők az adatok és a nyomtatás nagy mértékben függetleníthető a számítógéptől. Ugyanakkor a nyomtató saját vezérlője gyakorlatilag egy kisméretű számítógép, saját processzorral és memóriával. Egy-egy lap bittérképes tárolása kb. 1MB memóriát igényel.
- A nyomtatási képet **a számítógépben állítják össze** és az elkészült képet továbbítják a nyomtatóhoz. Ennél a megoldásnál ugyan olcsóbbá válik a nyomtató, de ugyanakkor leköti a számítógépet a kép előállításával, másrészt a szükséges adatátviteli sebesség lényegesen magasabb(1-1.5Mb/s), mint az előző változatnál.
- A korszerűnek tekinthető megoldásnál, a processzor a nyomtatási képet egy közvetítő nyelv, a **PostScript segítségével írja le** és ezt az előírást továbbítja a nyomtató felé, amely ennek értelmezésére képes és ennek alapján kialakítja a végleges nyomtatási képet.

A lézernyomtatók teljesítménye változó, a mikroszámítógép mellett használt nyomtatók 3-8 lap/perc sebességgel működnek.

9.2.3.BIOS szintű nyomtatókezelés

A számítógép és a nyomtató közötti kapcsolat létrehozása leggyakrabban a párhuzamos csatlakozón(porton) keresztül történik. A párhuzamos csatlakozóknál használt illesztőfelület, a CENTRONICS szabvány. Ez a csatlakozás egy-, vagy kétirányú lehet; az utóbbi esetben ezen keresztül megvalósítható a számítógép-számítógép kapcsolat is.

BIOS szinten a párhuzamos csatlakozóra kötött nyomtatót a **17h-es megszakításon** keresztül lehet elérni. A megszakításhoz kapcsolódó lehetséges funkciók az alábbiak:

00h	egy karakter kiküldése a nyomtatóra; a karakter a nyomtató pufferébe kerül, ahonnan a karakterek kiírása a puffer megteltére, vagy egy kocsivissza-soremelés karakter hatására történik;
01h	a nyomtató inicializálása;
02h	a nyomtató állapotbyte-jának beolvasása.

9.3.ELLENŐRZŐ KÉRDÉSEK, FELADATOK

1. Milyen funkcionális felosztása van a billentyűknek?
2. Milyen adatot szolgáltat a billentyűzet egy-egy billentyű lenyomásakor? Mit tartalmaznak a billentyűzetpufferbe átküldött byte-ok?
3. Hogyan jelzi a billentyűzet valamelyik funkcióbillentyű lenyomását?
4. Hány byte-os a billentyűzet puffer?
5. Milyen elvek alapján működnek az egerek? Melyik vezérlési módszert alkalmazzák a leggyakrabban?
6. Hány byte-nyi adatsort küld át a processzornak az egér? Mit tartalmaznak az egér által átküldött byte-ok?
7. Milyen karakter-nyomtató berendezéseket ismer? Hogyan működik a margarétakerekes nyomtató?
8. Hogyan működnek a nagygépes nyomtatók(forgóláncos, forgódobos)?
9. Hogyan működnek a mátrixnyomtatók? Hány tű található a nyomtatófejekben a mátrixnyomtatók esetében?
10. Hogyan jön létre a 'levél minőségű'(LQ, NLQ) nyomtatás? Mekkora a mátrixnyomtatók sebessége?
11. Milyen elven működnek a tintasugaras nyomtatók? Milyen felbontási lehetőségekkel rendelkeznek a tintasugaras nyomtatók? Mekkora a tintasugaras nyomtatók sebessége?
12. Milyen elven működik a lézernyomtató? Milyen módszerekkel állítják elő a lézersugár vezérléséhez a nyomtatandó képet? Mekkora a lézernyomtatók nyomtatási sebessége?
13. Mi az előnye és a hátránya a saját processzorral rendelkező lézernyomtatónak?
14. Mi a hátránya annak, ha a kinyomtatandó képet a számítógépben állítják elő és aztán továbbítják a nyomtató felé?
15. Milyen előnnyel rendelkezik a PostScript leíró nyelv használata?
16. Melyik csatolón keresztül kötik össze többnyire a nyomtatót és a számítógépet?

10. FEJEZET

FELHASZNÁLT IRODALOM

[01] *Abonyi Zs.*: PC hardver kézikönyv, ComputerBooks, Budapest, 1992, 1996(2.kiadás)

[02] *Boér L.-Dóra Gy.-Fenyő L.-Seres A.*: Az IBM PC-k belső felépítése, LSI, Budapest, 1989

[03] *Cserny L.*: Számítástechnika I. - Számítógép ismeretek, Tankönyvkiadó, Budapest, 1979(YMÉMF jegyzet)

[04] *Cserny L.*: RISC processzorok, LSI, Budapest, 1995

[05] *Cserny L.*: Számítógépek architektúrája, ME DFK, Dunaújváros, 1996(ME DFK jegyzet)

[06] *Erényi I.-Vajda F.*: Mikroprocesszoros rendszerek fejlesztése, Műszaki Könyvkiadó, Budapest, 1983

[07] *Gorsline, G.W.*: Computer Organization: Hardware/Software, Prentice-Hall, Englewood Cliffs, 1986(2nd edition)

[08] *Inotai L.-Lázár L.*: IBM PC XT/AT rendszerprogramozás, I-II-III., Novotrade, Budapest, 1991

[09] *INTEL*: Microprocessors: Volume I.: Intel386, 80286 & 8086 Microprocessors, 1994

[10] *INTEL*: Microprocessors: Volume II.: Intel486 Microprocessors, 1994

[11] *INTEL*: Microprocessors: Volume III.: Pentium processors, 1994

[12] *Kaszanyiczki L.*: Az egér programozása, LSI, Budapest, 1990

[13] *Kovács M.*: 32-bites mikroprocesszorok: 80386/80486, I-II., LSI, Budapest, 1991, 1994

[14] *Kovács M.*: Egyszerűen a mikroszámítógépről, LSI, Budapest, 1985

[15] *Liebig, H.-Fliek, T.*: Rechnerorganisation; Prinzipien, Strukturen, Algorithmen, Springer Verlag, Berlin, 1993(2.Auflage)

[16] *Milutinovic, V.M. ed.*: High-level Language Computer Architecture, Computer Science Press, 1989

[17] *Mimar, T.*: Programming and Designing with the 68000 Family; Including the 68000, the 68010/12, the 68020 and the 68030, Prentice-Hall, Englewood Cliffs, 1991

[18] *Németh G.-Horváth L.*: Számítógéparchitektúrák, Akadémiai Kiadó, Budapest, 1993(2.kiadás)

[19] Oxford Számítástechnikai értelmező szótár, Novotrade, Budapest, 1989

[20] *Protopapas, D.A.*: Microcomputer Hardware Design, Prentice-Hall, Englewood Cliffs, 1988

[21] *Ryan, B.*: M1 challanges Pentium, byte, 83-87, January, 1994

[22] R*yan, B.-Thompson, T.*: RISC Grows Up, byte, 91-98, January, 1994

[23] *Segee, B.-Field, J.*: Microprogramming and Computer Architecture, Wiley, New York, 1991

[24] *Tabak, D.*: Which System is RISC?, IEEE Computer, 19(10), 85-86, December 1986

[25] *Tanenbaum, A.S.*: Structured Computer Organization, Prentice-Hall, Englewood Cliffs, 1990(3rd edition)

[26] *Triebel, W.A.-Singh, A.*: The 68000 and 68020 microprocessors: Hardware, Software and Interfacing Techniques, Prentice-Hall, Englewood Cliffs, 1991

[27] *Wilkinson, B.*: Computer architecture: Design and Performance, Prentice-Hall, London, 1991

[28] *Wulf, W-A.*: Compilers and Computer Architecture, IEEE Com-puter, 14(7), 41-47, July 1981

TÁRGYMUTATÓ

2

A

Á

B

C

D

E

É

F

G

H

M

N

O

Ó

Ö

P

Q

R

S

T

U

V

W

ISBN 963-577-188-6

INFORMATIKA

Hetente

Magyarország vezető informatikai lapja. Számítástechnika, telekommunikáció, irodatechnika együtt: informatika – hetente 32 oldalon, 10 ezer példányban.

Havonta

Magyarország vezető számítástechnikai lapja. Hardvertesztek, hálózatok, szoftverbemutatók, alkalmazási tippek, vírusinformációk – havonta 100 oldalon, 10 ezer példányban.

CT PRESS
COMPUTERTECHNIK PRESS

H-1138 Budapest
Váci út 202.
Tel.: (36-1) 120-8007
Tel./fax: (36-1) 120-1636