Orgoványi József–Pszota József

Digitális technika

3. kiadás

Tankönyvmester Kiadó,
Budapest

Sorozatszerkesztő: Futterer László
Lektor: Szűcs Gyula

Felelős szerkesztő: Molnár Ervin
Borítóterv: Szlovencsák Ádám

3. kiadás

Felelős kiadó: a Tankönyvmester Kiadó ügyvezetője

ISBN 978 963 275 007 1

A tankönyv megrendelhető:
Tankönyvmester Kiadó
1141 Budapest,
Fogarasi út 111.
Tel.: 220-22-37
Fax: 221-05-73
www.tankonyvmester.hu
e-mail: info@tankonyvmester.hu

A könyv formátuma: B/5
Terjedelme: 10,6 (A/5) ív
Azonossági szám: TM–12003
Készült az MSZ 5601:1983 és 5602:1983 szerint
Szedés, nyomdai előkészítés: HEXACO GNH Kft.
Nyomta és kötötte: Regiszter Kiadó és Nyomda Kft., Budapest

TARTALOMJEGYZÉK

Előszó

A **Digitális technika** c. könyv, amit kezében tart a kedves Olvasó, a Tankönyvmester Kiadó villamos ipari és rokon szakmák számára készített tankönyvcsaládjának egyik középszintű tankönyve, ami szorosan a **Digitális elektronika** c. tankönyvre épül és annak folytatása.

A tankönyv a korábbi tanulmányok során megismert informatikai alapfogalmakra és alapáramkörökre építve a digitális rendszerek felépítését, tervezési elveit és működését tárgyalja. A digitális rendszerek leginkább elterjedt formája a digitális számítógép, ezért a tankönyv ennek fő elemeit (a vezérlőegységet, a sínrendszert, a memóriát, az aritmetikai és logikai egységet stb.) ismerteti, de kitér a PLA elemek alkalmazási lehetőségeire is. A tanultak összefoglalása céljából a könyv az I8086 mikroprocesszor-család és az MCS51 mikrovezérlő-család rövid leírásával zárul.

A könyv minden olyan szakterület (digitális irányitási rendszerek tervezése, üzemeltetése, számítógép karbantartás stb.) számára nélkülözhetetlen, amelynél a digitális rendszerek megismerése, a rendszerszemlélet kialakítása a cél.

A tankönyv az OKJ-ben előírt követelményeknek megfelelő tartalommal készült és az egyes témákat az előírásoknak megfelelő mélységben dolgozza fel.

Mindazoknak, akik további elméleti ismereteket szeretnének szerezni az egyes szorosan, ill. tágabban kapcsolódó témákban, ajánljuk a tankönyvcsalád következő könyveit:

TM-11001 Hámori Zoltán: **Az elektrotechnika alapjai,**
TM-11002 Kerékgyártó László: **Elektrotechnika,**
TM-11003 Zombori Béla: **Az elektronika alapjai,**
TM-11004 Zombori Béla: **Elektronika,**
TM-11008 Gyetván Károly: **A villamos mérések alapjai,**
TM-11209 Gyetván Károly–Futterer László: **Villamos mérési feladatok,**
TM-12008 Szerz. kollektíva: **Alkalmazott számítástechnika.**

A felsorolt könyvek az adott témákat teljeskörűen, egymásra építve dolgozzák fel, így szélesítik a tudást és kiegészítő információhoz juttatják az érdeklődőt.

Eredményes tanulást és szakmai sikereket kíván minden kedves Olvasójának a

Tankönyvmester Kiadó

BEVEZETÉS

Az elektronika a villamos ipari és rokon szakmák egyik alapozó tantárgya, amit fontossága és összetettsége miatt sok helyen két „független" témakörre, az analóg elektronikára és a digitális elektronikára bontva oktatnak. A digitális elektronika alapjait a Tankönyvmester Kiadó: **Digitális elektronika** c. tankönyve tartalmazza. Ebből a tanulók megismerkedhetnek az informatikai alapokkal (számrendszerekkel, kódolással, logikai algebrával), valamint a kombinációs és sorrendi hálózatok felépítésével, működésével és tervezési módszereivel, ill. az integrált áramkörös megvalósítási lehetőségekkel.

Ez a **Digitális technika** c. tankönyv a Digitális elektronika c. tankönyvre épül, és amíg az a digitális áramkörökkel, ez a digitális rendszerekkel foglalkozik.

A digitális rendszerek leginkább ismert és alkalmazott változata a digitális számítógép. A Digitális technika c. tankönyv bemutatja, hogyan lehet a megismert alapáramkörök (kapuk, multiplexerek, demultiplexerek, kódolók, dekódolók, számlálók, flip-flopok, egyszerű aritmetikai elemek stb.) felhasználásával és újszerű szervezési módszereket alkalmazva komplex, intelligens rendszereket létrehozni.

Ezeknek a rendszereknek a megvalósítási lehetőségei a komplexitás növelésével hatványozottan növekszenek. Egy konkrét módszert nem tudunk adni, helyette a könyvben csak a lehetőségeket (és azok előnyeit, ill. hátrányait) mutatjuk be.

A konkrét megvalósítást a könyv két utolsó fejezete tartalmazza, amelyek a 8086 mikroprocesszorral és az MCS 51 mikrokontroller családdal foglalkoznak.

A mikroprocesszorok fejlődését és további, széleskörben használt mikroprocesszorokat mutatja be a Tankönyvmester Kiadó: Alkalmazott számítástechnika c. tankönyve. Ott található meg az MCS 48 mikrokontroller-család ismertetése is.

1. SZÁMÍTÓGÉPES ADATÁBRÁZOLÁS

A digitális számítógépek működése a kettes számrendszeren alapul. Ennek egyik legfontosabb oka, hogy a kettes számrendszer könnyen megvalósítható elektronikus eszközökkel.

A kettes számrendszer két számjegyét (0, 1) két feszültség- vagy áramállapottal valósíthatjuk meg.

Pl. tranzisztor esetében:

- 0 az eszköz lezárt állapota, $I_C = 0$, $U_{CE} = \text{max.}$,
- 1 az eszköz nyitott állapota, $I_C = \text{max.}$, $U_{CE} = U_{CES} = 0$.

A számítógépes ábrázolás egysége a bit, amely egyetlen állapot (0 vagy 1) leírására alkalmas. Összetett, többértékű adatok bitsorozatokból kialakított egységekben ábrázolhatók. Ilyen egység a bájt (1 bájt = 8 bit) és a szó, ami a bájt egész számú többszöröse.

A tárolt adatok tartalmukat tekintve négy csoportba sorolhatók:

- numerikus adatok,
- karakteres adatok,
- logikai adatok,
- címek.

1.1. A numerikus adatok ábrázolása

1.1.1. Az előjel nélküli egész számok ábrázolása

Az előjel nélküli egész számokat a számítógép a kettes számrendszerbeli megfelelőjével ábrázolja (amit az egyszerűbb írásmód érdekében sokszor a kettes számrendszerre visszavezethető oktális vagy hexadecimális formában adunk meg).

A decimális számok kettes, nyolcas vagy tizenhatos számrendszerbe alakítására, ill. a bináris, oktális vagy hexadecimális számrendszerbeli számok decimális számmá alakítására alkalmas módszer a Tankönyvmester Kiadó: Digitális elektronika c. tankönyvében található.

BCD számábrázolás

Mivel tízes számrendszerben szeretünk számolni, ezért a számítógépeket is felkészítették arra, hogy tízes számrendszerbeli számokkal tudjanak dolgozni. Ezért hozták létre a Binary Coded Decimal – binárisan kódolt decimális – kódot, amit röviden BCD kódnak neveznek. A BCD kódban egy decimális számjegyet egy négybites bitsorozattal ábrázolunk, pl. $52_{10} = 0101\ 0010_{BCD}$.

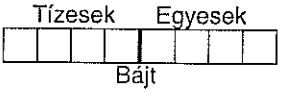
Ezek kétféleképp tárolhatók:

- pakolt BCD számábrázolással,
- pakolatlan (zónázott) BCD számábrázolással.

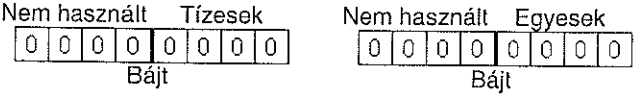
A kétféle tárolás formátuma eltérő, erre a programozónak ügyelnie kell.

A pakolt forma egy bájton két számjegyet tárol, így helytakarékos. A számítógépek BCD-aritmetikája ilyen pakolt formájú számokkal végez műveletet. A pakolatlan forma esetén a bájt felső négy bitje üres, ezért ez a forma nagy adattömeg tárolására nem alkalmas. Ennek a számábrázolási formának az adatok be- és kivitele esetén van jelentősége. A bájt felső négy bites üres helyére megfelelő kódot (ún. zónát) helyezve a decimális szám karakteres (nyomtatható) formája áll elő. Adatbeolvasáskor hasonló módon kapjuk az egyes számjegyeket, így csak a zónát kell törölni a pakolt BCD szám előállításához.

Pakolt BCD számok:



Pakolatlan BCD számok:



Az ilyen kódban ábrázolt számokkal végzett műveletek jóval bonyolultabbak, mint a bináris műveletek, de az eredmény könnyebben alakítható át tízes számrendszerbe. A problémát az okozza, hogy négy biten összesen 2^4, azaz 16 különböző számjegy ábrázolható. A tízes számrendszerben viszont csak tíz különböző számjegy van. Így a kódkészletben lesz hat olyan kódszó, amely nem értelmezhető BCD kódban (1010, 1011, 1100, 1101, 1110, 1111). Ezeket **tiltott kódszavaknak** nevezzük. Ha a BCD kódban végzett művelet eredménye ilyen kódszót tartalmaz, az nem értelmezhető, helytelen. Ilyenkor a helyes eredmény előállítása érdekében korrekciót kell alkalmazni. A korrekció a tiltott kód átlépését jelenti. Ez összeadásnál 6 hozzáadását jelenti minden tiltott kódot tartalmazó BCD helyiértéken. Kivonásnál a korrekció 6 kivonásával végezhető el.

Mintafeladat

Végezzük el a $14_{10}+28_{10}$ műveletet BCD kódban!

$14_{10} = 0001\ 0100_{BCD}$

$28_{10} = 0010\ 1000_{BCD}$

$0001\ 0100_{BCD}$

$+ \quad 0010\ 1000_{BCD}$

$0011\ \boxed{1100_{BCD}}$ tiltott kód

$+ \quad 0110_{BCD}$ korrekció

$0100\ 0010_{BCD} = 42_{10}$

1.1.2. Az előjeles egész számok ábrázolása

A pozitív és negatív számok megkülönböztetésére ún. **előjelbitet** használhatunk a szám legnagyobb helyiértékű bitje helyén. A 0 előjelbit pozitív számot, 1 előjelbit negatív számot jelöl. Ezzel az ábrázoláshoz szükséges bitek száma eggyel nő. A pozitív számok ábrázolása 0 előjelbittel kódolatlan kettes számrendszerben történik. A negatív számok ábrázolására többféle módszer lehetséges, mint pl. az előjeles abszolút érték, az 1-es komplemens kód vagy a 2-es komplemens kód.

Előjeles abszolút érték

A negatív számot 1 előjelbittel, és a szám abszolút értékét bitsorozattal megadva ábrázoljuk (**1.1.** ábra). Ez a számábrázolás egyszerű, de közvetlen műveletvégzésre nem alkalmas.

1	Abszolút érték

1.1. ábra. Előjeles abszolút értékes számábrázolás

1-es komplemens (1-es kiegészítő, 1C) kód

Előjeles számokat alkalmazva nincs szükség a kivonás műveletére, hiszen A–B =A+(–B). A komplemens kódú számábrázolás révén a kivonás és összeadás egy műveletté vonható össze, mivel kivonás helyett a komplemens kódú (negatív) szám hozzáadását végezhetjük. Természetes követelmény, hogy az előjeles számok összege a kód szabályai szerinti ábrázolásban előjelhelyes eredményt adjon. Ez lehetővé teszi a számítógépek műveletvégző egységének egyszerűbb felépítését.

A **komplemens** magyarul kiegészítőt jelent, vagyis azt a számot, ami a kérdéses számot egészíti ki az n biten ábrázolható legnagyobb számra. Ennek megfelelően egy adott, n biten ábrázolt pozitív számmal megegyező abszolút értékű 1-es komplemens kódú (negatív) számot megkapjuk, ha az n biten ábrázolható legnagyobb számból a pozitív számot kivonjuk. Az n biten ábrázolható legnagyobb szám n db

1-esből áll. Vegyük észre, hogy bármely bináris számot az 111...111 számból kivonva az eredmény a kivonandó szám bitenkénti negáltja lesz. Ezért az egyes komplemens képzést a gyakorlatban a pozitív szám összes bitjének negálásával végezzük.

Mintafeladat

Határozzuk meg 01101101_2 szám (+109) 1-es komplemensét!

A biteket egyenként negálva az 10010010_{1C} komplemens kódú számot kapjuk.

Mintafeladat

Végezzük el 8 biten, 1C kódban a 37–65 = 37+(–65) = –28 műveletet!

A műveleti szabályok 1-es komplemens kódban:

- a teljes számmal (az előjelet is beleértve) elvégezzük a műveletet,
- ha az n. helyiértékről az (n+1).-re átvitel keletkezik, azt a legkisebb helyiértéken hozzáadjuk az eredményhez,
- ha az eredmény első bitje 0, akkor az eredmény pozitív és kódolatlan bináris számként állt elő,
- ha az eredmény első bitje 1, akkor az eredmény negatív és 1-es komplemens kódban állt elő.

+37	$0\ 0100101_2$	pozitív szám nem komplementáljuk $0\ 0100101_2$
+65	$0\ 1000001_2$	–65-höz komplementálni kell, $1\ 0111110_{1c}$

$$\begin{array}{rr} & 0\ 0100101_2 \\ + & 1\ 0111110_{1c} \\ \hline & 1\ 1100011_{1c} \end{array}$$

Az eredmény negatív és komplemens kódú.

Az eredmény abszolút értékét megkapjuk, ha az eredményt (vissza)komplementáljuk: $0\ 0011100_2$ (+28).

Mint látható, az előjeles számok összeadása 1-es komplemens kódban előjelhelyes eredményt ad. Leegyszerűsíti a számolást, mivel nincs szükség kivonási műveletre. Így a műveletvégzés egyszerűbb felépítésű aritmetikai egységgel valósítható meg. Hibája, hogy a nulla ábrázolása nem egyértelmű.

Írjuk fel a 0 környezetébe eső első néhány szám 1-es komplemens kódját!

3	00000011
2	00000010
1	00000001
0	00000000
–1	11111110
–2	11111101
–3	11111100

Látható, hogy a bináris számsorból az 11111111 hiányzik. Ezt felfoghatjuk 00000000 1-es komplemensének, azaz –0-nak. A kétféle nulla a számolásban nehézségeket okoz. Célszerű a nullát a két kód közül csak az egyikkel azonosítani. Ezt nevezzük gépi nullának.

2-es komplemens (2C) kód

Egy adott, *n* biten ábrázolt pozitív számmal megegyező abszolút értékű kettes komplemens kódú (negatív) számon azt a számot értjük, amelyik a kérdéses számot az *n* biten ábrázolható legnagyobb számnál eggyel nagyobb számra egészíti ki. Azaz a szám kettes komplemensét megkapunk, ha az *n* biten ábrázolható legnagyobb számból a pozitív számot kivonjuk és a különbséghez egyet hozzáadunk. A kettes komplemenskéképzést a **gyakorlatban** egyszerűbben végezzük: jobbról balra haladva az első egyesig a szám összes bitjét változatlanul leírjuk (még az első egyest is), majd tovább haladva az összes bitet negáljuk.

Mintafeladat

Határozzuk meg 01101101_2 szám (+109) kettes komplemensét!

Az előző módszert alkalmazva (jobbról balra haladva az első egyes utáni összes bitet negáljuk) az10010011_{2C} komplemens kódú számot kapjuk.

Mintafeladat

Végezzük el 8 biten, 2C kódban a 37_{10}–65_{10} műveletet!

A műveleti szabályok 2-es komplemens kódban:

- a teljes számmal (az előjelet is beleértve) elvégezzük a műveletet;
- ha átvitel keletkezik az (n + 1). bitre és az megegyezik az n. bittel, akkor az átvitelt elhanyagoljuk, ha a két bit nem egyezik meg az eredmény csak n + 1 biten ábrázolható;
- ha az eredmény első bitje 0, akkor az eredmény pozitív és kódolatlan bináris kódban állt elő;
- ha az eredmény első bitje 1, akkor az eredmény negatív és 2-es komplemens kódban állt elő.

+37	$0\ 0100101_2$	pozitív szám, nem komplementáljuk	$0\ 0100101_{2c}$
+65	$0\ 1000001_2$	–65-nél, komplementálni kell	$1\ 0111111_{2c}$
		$0\ 0100101_{2c}$	
		$+\ 1\ 0111111_{2c}$	
		$1\ 1100100_{2c}$	

Az eredmény negatív és komplemens kódú.

Az eredmény abszolút értékét megkapjuk, ha visszakomplementáljuk: $0\,0011100_2$ ($+28_{10}$).

Kettes komplemens kódban az átvitel elhanyagolása miatt jóval egyszerűbb a műveletvégzés, mint egyes komplemensben, ezért a digitális számítógépekben negatív számok ábrázolására ezt a kódot alkalmazzák.

1.1.3. Nem egész számok ábrázolása, a bináris pont helye

Bináris törteknél fontos kérdés, hogy a bináris pontot a rendelkezésre álló bitsorozatban hogyan helyezzük el, hiszen ez alapvetően meghatározza az ábrázolható értéktartományt és az ábrázolási pontosságot. A bináris pontot rögzíthetjük valamely helyiérték mellett, ezt fixpontos számábrázolásnak nevezzük.

A bitsorozat egy részét felhasználva megadhatjuk a bináris pont helyét (a bináris pont helye nem rögzített), ilyenkor lebegőpontos számábrázolásról beszélünk.

Fixpontos számábrázolás

A fixpontos számábrázolás hátránya, hogy az ábrázolható értéktartomány és a pontosság előre meghatározott (**1.2.** ábra). Nincs lehetőség kisebb pontossággal számolva, a törtrész bitjeinek csökkentésével az ábrázolási tartományt növelni, ill. nagyon kis számok ábrázolása esetén az egészeket leíró bitek számának csökkentésével és a törtrészbitek számának növelésével a pontosságot növelni.

Előjel	Egészek	.	Törtek

1.2. ábra. Fixpontos számábrázolás

Műveletek fixpontos számokkal

Összeadás, kivonás

Előjeles számok összeadása, kivonása kettes komplemens kódban történik. Ha az eredmény az adott hosszúságú bitsorozaton nem ábrázolható, **átvitel** (carry) keletkezik. Ezt a legnagyobb helyiértékű bit elé írva eggyel hosszabb bitsorozat adja a helyes eredményt. Az átvitelt a processzorok egy belső állapotbittel tárolni tudják. Ha előjeles számokkal végzünk műveletet, ennél sokkal bonyolultabb eset is előfordulhat. Ugyanis az előjeles számok legnagyobb helyiértékű bitje előjelet határoz meg.

Ha eggyel kevesebb biten nem ábrázolható az eredmény, akkor az „rácsorog" az előjelbitre, és ha azt megváltoztatja, helytelen eredményt kapunk. Ezt a jelenséget **túlcsordulásnak** (overflow) nevezzük. A processzorok figyelik a túlcsordulást, és ha ez előáll, megszakítják a program működését.

Szorzás

A szorzás műveletét többféle algoritmus szerint végezhetjük el, az operandusok abszolút értékét használva (és az eredményt a operandusok előjele alapján utólag előjelezve). Az egyik legelterjedtebb a szorzást sorozatos léptetésekre és összeadásokra lebontó algoritmus (a tízes számrendszerben is így végezzük papíron a szorzást). Két n bites számot összeszorozva az eredmény $2n$ biten ábrázolható.

Mintafeladat

Végezzük el az 1101·0101 négybites szorzást binárisan (feltételezve, hogy mindkét szám pozitív)!

```
  1101·0101
  0000
   1101
    0000
+    1101
  01000001
```

Az eredmény nyolc biten ábrázolható.

Osztás

Az osztást szintén a számok abszolút értékével végezzük és az eredményt utólag előjelezzük. A műveletet sorozatos léptetésekre, kivonásokra és összehasonlításokra vezethetjük vissza. Egy $2n$ bites számot n bitessel osztva (ha nincs túlcsordulás) n bites hányados egész részét, és n bites maradékot kapunk.

Mintafeladat

Végezzük el a 00001001/1101 osztást binárisan (8 bit/4bit)!

A=1001	B=1101	$A<B$, Q1=0
00010010	léptetés	$A>B$, Q2=1
−1101	kivonás (Q=1 után)	
00000101	különbség	
00001010	léptetés	$A<B$, Q3=0
00010100	léptetés	$A>B$, Q4=1
−1101	kivonás (Q=1 után)	
00000111	különbség	
00001110	léptetés	$A>B$, Q5=1
−1101	kivonás (Q=1 után)	
00000001		

Hányados:0.1011, a maradék $1 \cdot 2^{-4}$

Lebegőpontos számábrázolás

Alapja a számok normalizálása. Minden kettes számrendszerbeli szám felírható a következő alakban: $N = \pm M \cdot 2^{\pm K}$, ahol M az előjeles mantissza és K az előjeles karakterisztika.

A lebegőpontos formájú számnál tehát ábrázolni kell a mantisszát és előjelét, valamint a karakterisztikát és előjelét. Az ábrázolás módjában az egyes gépi megvalósítások eltérnek.

A mantisszát normalizálhatjuk:

- egészre (binárisan M = 1.xxxx alakúra), ilyenkor M decimális értékét tekintve $1 \le M < 2$,
- törtre (binárisan M = 0.1xxx alakúra), ilyenkor M decimális értékét tekintve $0{,}5 \le M < 1$.

Látható, hogy akár egészre, akár törtre normalizálunk, az első értékes számjegy (1 értékű bit) fix helyen van. Ha ezt kihagyjuk az ábrázolásból (nem felejtjük el, csak nem ábrázoljuk), az ábrázolási tartomány kétszeresére nő. Ezt a rejtett bitet **implicit bitnek** nevezzük. Amikor számolni akarunk a számmal, az implicit bitet ismét „hozzáragasztjuk" az ábrázolt számhoz, és így végezzük el a műveletet. Az eredményt ismét normalizáljuk, és a tárolást az implicit bit elhagyásával végezzük. A módszer hátránya, hogy a nullát nem tudjuk pontosan ábrázolni, ezért gyakorlatban a nullát a nulla értékű karakterisztikával azonosítják, ezt gépi nullának nevezzük.

A szokásos lebegőpontos számábrázolási módok az Intel és az IBM cégé.

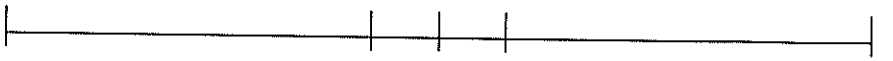
Az Intel cég formátumai lebegőpontos számábrázolásra. Az Intel cég két különböző, az ún. rövid valós és az ún. hosszú valós formátumot használja a lebegőpontos számok ábrázolására (IEEE Standard).

Rövid valós (short real) formátum. Négybájtos (32 bites), implicit bites, törtre normalizált, 128-cal eltolt karakterisztikájú ábrázolásmód (**1.3.** ábra).

Mantissza előjele: 1 bit	+128-cal eltolt karakterisztika 8 bit	Implicit bites, törtre normalizált mantissza 23 bit

1.3. ábra. Short real számábrázolás

Ábrázolási tartománya nullára szimmetrikus.



$S_{min} = 2^{-128}$, ill. $S_{max} = 2^{127}$.

Pontosság: 24 bináris jegyre (kb. 7 decimális jegyre) pontos.

Mintafeladat

Hozzuk a -37.25_{10} számot real short formátumúra!

$37.25_{10} = 100101.01_2$

- A bináris pont léptetésével a mantisszát törtre normalizáljuk: $M = 0.10010101$ és $K = 6$ (eltolás 6 bittel balra),
- Az ábrázolt mantisszát az implicit bit elhagyásával kapjuk: $M' = .0010101$.
- Az ábrázolt karakterisztika (+128-cal eltolva): K'=10000110 (134).
- Előjelbit = 1 (negatív szám).

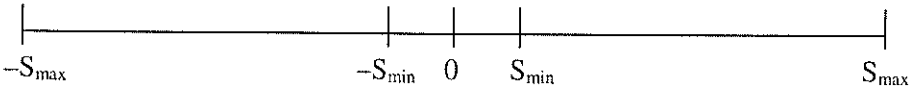
A szám real short alakja: 1 10000110 00101010000000000000000

Hosszú valós (long real) formátum. A hosszú valós formátum nyolcbájtos (64 bites), implicit bites, törtre normalizált, 1024-gyel eltolt karakterisztikájú ábrázolásmód (**1.4.** ábra).

Mantissza előjele: 1 bit	+1024-gyel eltolt karakterisztika 11 bit	Implicit bites , törtre normalizált mantissza 56 bit

1.4. ábra. Long real számábrázolás

Ábrázolási tartománya nullára szimmetrikus.



$S_{min}=2^{-1025}$, ill. $S_{max}=2^{1024}$.

Pontosság: 56 bináris jegyre (kb. 17 decimális jegyre) pontos.

Az IBM cég formátumai lebegőpontos számábrázolásra. Az IBM cég nagy számítógépeiben a lebegőpontos számok ábrázolását tizenhatos alapú számrendszerben végzik.

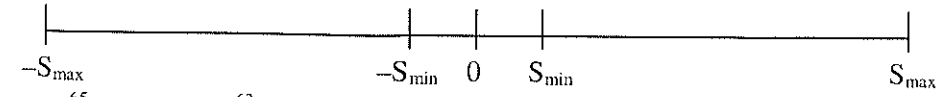
Egy hexadecimális számjegyet négy biten ábrázolnak. Kétféle pontosságú ábrázolást használnak: a négybájtos real és a nyolcbájtos double formátumot.

IBM real formátum. Négybájtos (32 bites), törtre normalizált, 64-gyel eltolt karakterisztikájú ábrázolásmód (**1.5.** ábra).

Mantissza előjele: 1 bit	+64-gyel eltolt karakterisztika 7 bit	Törtre normalizált mantissza 24 bit (6 félbájt)
		16^{-1} 16^{-2} 16^{-3} 16^{-4} 16^{-5} 16^{-6}

1.5. ábra. IBM real számábrázolás

Ábrázolási tartománya nullára szimmetrikus.



$S_{min}=16^{-65}$, ill. $S_{max}=16^{63}$.

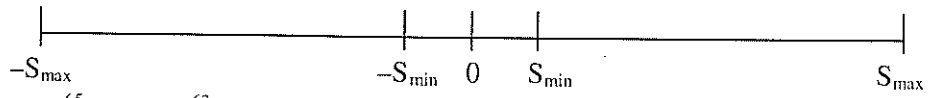
Pontosság: hat hexadecimális jegyre (kb. 7 decimális jegyre) pontos.

IBM double formátum. Nyolcbájtos (64 bites), törtre normalizált, 64-gyel eltolt karakterisztikájú ábrázolásmód (**1.6.** ábra).

Mantissza előjele: 1 bit	+64-gyel eltolt karakterisztika 7 bit	törtre normalizált mantissza 56 bit (14 félbájt)
		16^{-1} 16^{-2} ..... 16^{-6} 16^{-14}

1.6. ábra. IBM double számábrázolás

Ábrázolási tartománya nullára szimmetrikus.



$S_{min}=16^{-65}$, $S_{max}=16^{63}$.

Pontosság: tizennégy hexadecimális jegyre (kb. 17 decimális jegyre) pontos.

Műveletek lebegőpontos számokkal

Összeadás, kivonás

Összeadni és kivonni csak azonos karakterisztikájú számokat lehet. Az eredményt a közös karakterisztikára hozás után a mantisszák összegéből és a közös karakterisztikából kapjuk. A műveletvégzés után az eredményt normálalakra kell hozni.

Szorzás

A mantisszák abszolút értékét a már megismert algoritmussal össze kell szorozni, a karakterisztikákat össze kell adni, majd az eredményt előjelezni és normalizálni kell.

Osztás

Az osztandó mantisszájának abszolút értékét a már megismert algoritmussal el kell osztani az osztó mantisszájának abszolút értékével, és az osztandó karakterisztikájából ki kell vonni az osztóét, majd az eredményt előjelezni és normalizálni kell.

Mintafeladat

Adott két real short alakban ábrázolt szám:

A = 40700000h,

B = C1900000h.

Végezzük el a $S = A + B$ műveletet és az eredményt adjuk meg real short alakban, hexadecimálisan!

- A hexadecimálisan megadott két számot írjuk fel binárisan:

  A = 0 10000000 11100000000000000000000,

  B = 1 10000011 00100000000000000000000

  (mindkét szám real short formátumú).
- Meghatározzuk a két szám normálalakját, hogy a műveletet el tudjuk végezni:

  M'_A =111 M_A= 0.1111 (az implicit bitet visszaírjuk!),

  K'_A =10000000 K_A= 0,

  M'_B =001 M_B= 0.1001 (az implicit bitet visszaírjuk!),

  K'_A =10000011 K_B= 11.
- A két szám 0 karakterisztikával ábrázolva:

  A = +0.1111 A = 0 000.1111 (előjelbittel 8 biten),

  B = –100.1 B = 1 011.1000 (előjelbittel 8 biten),

  B szám negatív, ezért az összeadás előtt az abszolút értékét komplementáltuk!
- Összeadás előjelbittel:

$$\begin{array}{rr} & 0\ 000.1111 \\ + & \underline{1\ 011.1000} \\ S = & 1\ 100.0111 \end{array}$$

- Az eredmény negatív 2-es komplemens kódú, az ábrázoláshoz vissza kell komplementálni:

  S = –11.1001
- Az eredményt normalizáljuk:

  M = 0.111001 M' = 11001 (ábrázolt mantissza),

  K = 10 K' = 10000010 (ábrázolt karakterisztika)

- Az eredmény real short alakban:

  S = 1 10000010 11001000000000000000000,

  Hexadecimálisan

  S = C1640000h

1.2. Karakterek ábrázolása

A számítógépen a karaktereket is csak bináris jelsorozattal ábrázolhatjuk. A leggyakrabban használt kódrendszer az ún. **ASCII kód** (American Standard Code for Information Interchange). Ez egy hétbites kód, amely alkalmazásával 2^7, azaz 128 különböző karakter ábrázolására van lehetőség. A kódolási eljárás során az ábrázolni kívánt jelek mindegyikéhez hozzárendelünk egy egybájtos számot, ez az illető karakter ASCII kódja. A hozzárendeléseket az **ASCII kódtábla** tartalmazza. Az ASCII kódokat a számítógép egy bájton tárolja, így fennmarad egy bit, amelyet paritásbitként használhatunk adatátviteli hibák felismerésére, vagy a kódot nyolcbitesre bővítve lehetőség van 256 különböző karakter ábrázolására. Ezeket a nyolcbites kódot nevezzük ASCII-8 kódnak. (Az ASCII-8 kódok táblázata a Tankönyvmester Kiadó: Digitális elektronika c. tankönyvében található.) Ezek alkalmazásával hozták létre az ún. **nemzeti kódtáblákat**, amelyek az angol ábécé betűin kívül az adott nemzet ábécéjének más betűit is tartalmazzák. Pl. *a magyar nemzeti kódtábla a 852-es sorszámú, amely torzítatlanul tartalmazza a magyar ékezetes betűket.*

1.3. Logikai adatok ábrázolása

A logikai adatok értékkészlete Igen vagy Nem lehet, ami jól illeszkedik a bináris számábrázoláshoz. Ennek megfelelően egy logikai adatot egy biten ábrázolhatunk (pl. 0 megfelel a Nem, Hamis, False állításnak, 1 megfelel az Igen, Igaz, True állításnak). Ez rendkívül tömör adattárolást tesz lehetővé. Minden számítógép rendelkezik olyan utasításokkal, amelyek adatbájtok tetszőleges bitjével logikai műveleteket (negálás, ÉS, VAGY, kizáró-VAGY kapcsolat) végez, ill. az adott bit állapotát teszteli (az eredményt a Flag regiszter megfelelő bitjének beállításával adva meg). A bitműveletek általában elég időigényesek, ezért az egyszerűség kedvéért sok esetben egy logikai adatot egy bájton tárolnak (pl. a bájt 0 értéke az Igaz, tetszőleges nullától különböző értéke a Hamis állítást jelenti).

1.4. Címek ábrázolása

A digitális számítógépek fontos adatformája a cím, ami megadja az egyes adatok, utasítások vagy I/O eszközök helyét. A címek egyszerű bináris számként ábrázolhatók, mivel jellegüknél fogva csak pozitív egész számok lehetnek. A címek ábrázolásánál az egyetlen problémát a címek hossza jelenti, mivel a korszerű számítógépek címtartománya rendkívül nagy lehet (így a szokásos bájtos vagy szavas adatforma nem elegendő egy cím tárolására). Ennek a problémának a megoldására különböző módszereket dolgoztak ki, amelyeket a tankönyv következő fejezeteiben tárgyalunk.

Ellenőrző kérdések, feladatok

1. Ismertessük a kettes számrendszer előnyeit a számítógépes adatábrázolásnál!
2. Ismertessük a fixpontos számábrázolás jellemzőit!
3. Ismertessük a lebegőpontos számábrázolás jellemzőit!
4. Alakítsuk át a –34,75 számot fixpontos kettes számrendszerbeli számmá!
5. Ismertessük a kettes komplemens kódú műveletvégzés szabályait összeadás és kivonás esetén!
6. Ismertessük a lebegőpontos osztás és szorzás műveleti szabályait!
7. Alakítsuk át a –34,75 számot real short alakú számmá!
8. Végezzük el a következő real short alakban megadott számokkal a kijelölt műveletet! N = 83450000h, M = 81320000h.
   - $S = N + M$?
   - $D = N - M$?
9. Ismertessük az IBM számítógépeken alkalmazott lebegőpontos számábrázolási módokat!
10. Ismertessük a digitális számítógépek karakterábrázolásának jellegzetességeit!
11. Hogyan ábrázoljuk a címeket?

2. MIKROSZÁMÍTÓGÉPEK

A mikroszámítógép viszonylag kis teljesítményű, kis műveleti sebességű mikroprocesszoros rendszer, egybeépítve a programfuttatáshoz szükséges operatív tárolóval, és a kezeléséhez nélkülözhetetlen be- és kimeneti perifériákkal.

A rendszer alapelveinek kidolgozásában nagy jelentősége volt a magyar származású Neumann Jánosnak. Ma már léteznek nem Neumann-elven működő számítógépek is, amelyeket a számítógép teljesítményének növelése érdekében hoztak létre. A Neumann-architektúra azonban még mindig jelentős a kisszámítógépek világában.

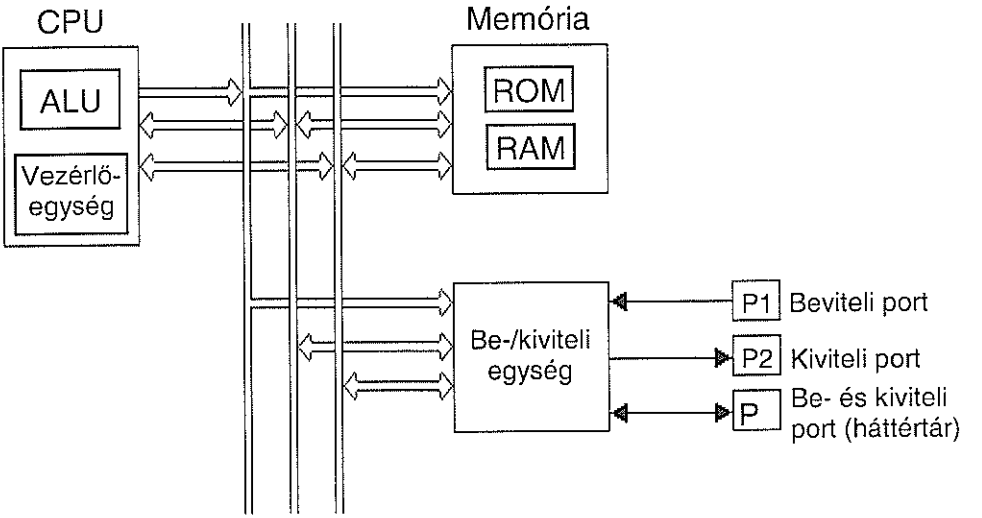
A Neumanni elvek alapján felépített számítógép fő jellemzői:

- teljesen elektronikus felépítés (gyors),
- kettes számrendszer használata (egyszerűen megvalósítható),
- belső elektronikus tár, amely adatokat és utasításokat tárol,
- a tárolt program alapján a gép önállóan futtatja a programot.

2.1. A mikroszámítógépek rendszertechnikai felépítése

A számítógép nagyon sok változáson ment át napjainkig, azonban szinte a kezdetektől jellemző rá az újrakonfigurálhatóság, az utólagos bővíthetőség. Ez csak olyan formában képzelhető el, hogy a processzor és a külső egységek egyfajta szabványos csatlakozófelülettel, ún. sínrendszerrel vannak összekötve (**2.1.** ábra). A mikrogépek legfőbb rendszertechnikai jellemzője tehát a sínrendszer, amely a gép funkcionális egységeit kapcsolja össze. A sín egy fizikai és logikai kialakítás szempontjából szabványos vezetékcsoport, amelyen keresztül az egyes egységek meghatározott szabályok (protokoll) szerint kommunikálnak egymással. A sínrendszert a kommunikációban végzett feladatától függően több sínre oszthatjuk (címsín, adatsín, vezérlősín).

Gyakorlati jelentősége ma már csak a két-, ill. háromsínes rendszereknek van. A sínrendszer felépítéséről, feladatáról részletesebben az 3. fejezetben lesz szó.



2.1. ábra Mikroszámítógép rendszertechnikai felépítése

2.2. A mikroszámítógépek funkcionális felépítése

A mikrogép **2.1.** ábrán látható legfontosabb funkcionális egységei a következők:

- vezérlőegység (Controll Unit, CU),
- aritmetikai és logikai egység (Aritmetic Logic Unit, ALU),
- operatív tároló (memória),
- be- és kiviteli egység (I/O-vezérlő).

A CU és ALU többnyire egy közös tokban, a CPU-ban (Central Process Unit) helyezkedik el, fizikailag ez a mikroprocesszort jelenti.

Ebben a fejezetben a funkcionális egységek mikroszámítógépes rendszerbeli feladatát (funkciójukat), ill. a rendszer globális működését mutatjuk be, míg részletes felépítésüket és működésüket a további fejezetek tárgyalják.

2.2.1. CPU: Central Process Unit

A rendszer központi egysége. Egyenként előveszi a program utasításait az operatív tárból, értelmezi azokat, és ennek alapján vezérlőjeleket állít elő a belső egységek (ALU, regiszterek) és külső egységek (memória, perifériák) számára.

Működését és felépítését részletesebben l. a 6. fejezetben.

2.2.2. Memória (operatív tároló)

A mikroprocesszor belsejében lévő regiszterek nem elegendőek egy program és a szükséges adatok tárolására. A működő program utasításait és adatait ezért a memóriában kell tárolni, ahonnan a mikroprocesszor tölti be a megfelelő memóriarekesz tartalmát.

> A memóriák félvezetős tárak, nagy integráltságú LSI áramkörök.

Alapvetően kétféle típusú félvezetős tárat alkalmazunk.

- ROM (Read Only Memory) csak olvasható memória, a rendszer működtetéséhez nélkülözhetetlen programokat, rutinokat tárolja (pl. ROM BIOS). Tartalma állandó.
- RAM (Random Access Memory) írható-olvasható memória, a felhasználói programokat és adatokat tárolja a programfuttatás során. Tartalma a tápfeszültség kikapcsolásakor elvész.

Ezt a témakört a könyv 4. fejezete tárgyalja.

2.2.3. Be- és kiviteli egység (I/O vezérlő)

A perifériák kapcsolatát biztosítja a rendszerrel. Fogadja a perifériák kéréseit, a kérésekkel a processzorhoz fordul, majd a processzor válaszjelei alapján előállítja a perifériák vezérlőjeleit.

Perifériák

Az operatív tárral kiegészített CPU már működőképes, de a felhasználó nem tudja használni, mert nem tud a programfutás során adatokat bevinni, ill. a program eredményeit nem látja.

> A perifériák azok az eszközök, amelyek az ember-gép kapcsolatot megvalósítják.

A beviteli perifériák az utasítások, programok, adatok bevitelét teszik lehetővé. A kiviteli perifériák a programfeldolgozás eredményeit teszik érzékelhetővé (láthatóvá, hallhatóvá). A beviteli-kiviteli perifériák általában adat- és programtárolási feladatokat látnak el.

Beviteli perifériák:

- billentyűzet,
- egér,
- scanner,
- CD ROM.

Kiviteli perifériák:

- monitor,
- nyomtató,
- hangkártya.

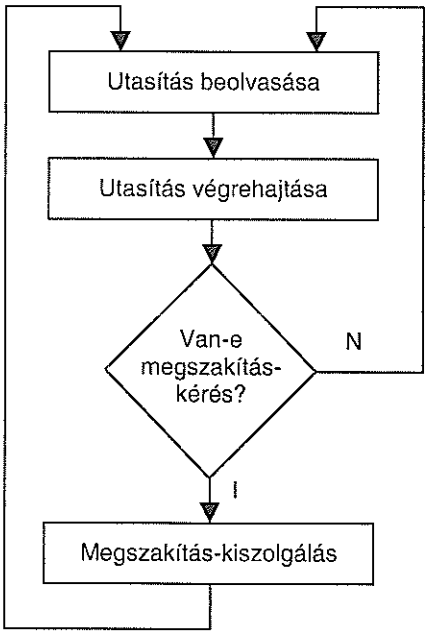
Be- és kiviteli perifériák:

- winchester,
- floppy,
- streamer.

2.3. A mikroszámítógép működése

A számítógép működtetését a mikroprocesszor a tárolt program alapján végzi. A program utasítások és adatok sorozata.

Ezek a memóriában bináris (gépi kódú) formában helyezkednek el. A processzor az utasításokat egyenként beolvassa az operatív tárból, értelmezi azokat, majd órajellel szinkronizált vezérlőjel-sorozattal biztosítja az utasítás végrehajtását.



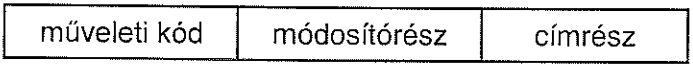
2.2. ábra Mikroszámítógép működésének folyamatábrája

A mikrogép működésének folyamatábrája a **2.2.** ábrán látható. A mikroszámítógép működtetése során előfordulhatnak olyan események, amelyek a futó programnál fontosabb feladat végrehajtását teszik szükségessé. Ezek lehetnek belső, processzoron belüli események (pl. nullával való osztás), ill. külső események (pl. a periféria kiszolgálást kér). Ilyenkor az aktuális utasítás befejezése után a programot meg kell szakítani, és a megszakítási ok azonosítását követően a megszakítást egy rutin lefuttatásával ki kell szolgálni, majd a megszakított programot kell folytatni. A megszakítás-kiszolgálással részletesebben az 3. fejezetben foglalkozunk.

2.3.1. A gépi kódú utasítás általános felépítése

A gépi kódú utasítás (**2.3.** ábra) három fő részből áll:

- műveleti kód, ami meghatározza, hogy a mikrogép milyen műveletet végezzen,
- módosítórész, ami a műveleti kód, ill. a címek pontos értelmezéséhez ad módosító előírást,
- címrész, ami a műveletvégzés operandusait vagy azok memóriabeli címét tartalmazza.



2.3. ábra. Gépi kódú utasítás felépítése

Utasításszerkezet

Általános esetben egy utasítás végrehajtásakor a processzornak négy címre van szüksége:

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

Attól függően, hogy az utasítás címrésze hány címet tartalmaz, beszélhetünk négy-, három-, kettő-, egy- és nullacímes utasításszerkezetről.

Négycímes utasítás.

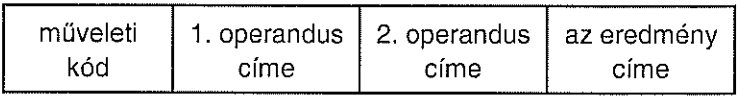
A fejlettebb processzorokban már nem alkalmazzák, mert az ilyen szerkezetű utasítás helyfoglalása nagy (**2.4.** ábra).

műveleti kód	1. operandus címe	2. operandus címe	az eredmény címe	a következő utasítás címe

2.4. ábra. Négycímes utasítás szerkezete

Háromcímes utasítás.

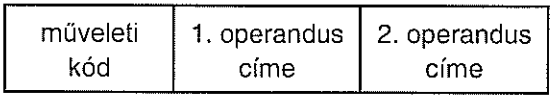
Az utasításszámláló regiszter (Program Counter, PC) bevezetésével a következő utasítás címét nem kell az utasításban elhelyezni, mert azt mindig a PC tartalmazza (**2.5.** ábra). **Az utasítás végrehajtásakor a processzor automatikusan beállítja a PC-t a következő utasítás címére.** Ennek feltétele, hogy az utasítások a végrehajtás sorrendjében helyezkedjenek el az operatív tárban. Ugyanakkor a programban a soros utasításfeldolgozás megbontásához vezérlésátadó (ugró) utasítások bevezetése szükséges.



2.5. ábra. Háromcímes utasítás szerkezete

Kétcímes utasítás.

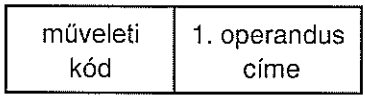
A processzor a műveletvégzés eredményét automatikusan visszaírhatja az egyik operandus helyére (**2.6.** ábra). Így az eredmény címe elhagyható.



2.6. ábra. Kétcímes utasítás szerkezete

Egycímes utasítás.

Az utasításvégrehajtás során a második operandus helye lehet rögzített (alapértelmezett) (**2.7.** ábra). Ehhez ki kell jelölnünk egy regisztert, amely a második operandust tartalmazza. Erre a célra az ún. akkumulátorregisztert használják. Ilyenkor a műveletvégzés előtt az akkumulátort egy külön utasítással fel kell tölteni a második operandussal.



2.7. ábra. Egycímes utasítás szerkezete

Nullacímes utasítás.

Csak műveleti részből áll, nem tartalmaz címrészt (**2.8.** ábra). **Egyetlen operandusú műveleteknél alkalmazzák.**

A műveletvégzés operandusa alapértelmezett helyen található. A regisztertartalmakkal végzett műveleteknél (pl. akkumulátor tartalmának bitenkénti negálása), ill.

veremtár műveleteknél használatos. A veremtár címzése saját, automatikusan kezelt címekkel történik, így nincs szükség az utasításban cím megadására.

2.8. ábra. Nullacímes utasítás szerkezete

2.3.2. Címzésmódok

Az utasításban megadott címből az ún. címmódosítással állítják elő a pontos címet.

A címmódosításra több okból lehet szükség:

- az utasítás címrésze nem elég hosszú a teljes operatív tár eléréséhez,
- a memóriában egymás után elhelyezkedő, azonos jellegű adatokon kell ugyanazt a műveletet végrehajtani,
- a program áthelyezhetőségét kell biztosítani.

Az utasítás általános felépítésénél láttuk, hogy az utasítás módosítórészében adható meg az operandusok megtalálásának módja. A módosító rész ezt megadó bitjeit címzésmód biteknek nevezzük. Az operandusok címzésére sokféle módszert használunk annak érdekében, hogy a különböző feladatokat minél hatékonyabban lehessen megoldani. A címzési módokat az egycímes utasításszerkezeten tárgyaljuk, de a többcímes formákra is kiterjeszthető. Az operandusok címének meghatározására a következő címzésmódokat használjuk.

Közvetlen adatcímzés (immeditate vagy literális címzés)

Az utasítás címrészében maga az operandus van. Gyors címzésmód, az operandus meghatározásához nincs szükség újabb sínciklusra (nem kell a memóriából kiolvasni), hátránya viszont, hogy így általában csak egyszerű (a rendelkezésre álló hely szűkössége miatt egy- vagy kétbájtos) operandusok adhatók meg.

Regisztercímzés

A címrészben azt a regisztert jelöljük ki, ahol az operandus megtalálható,

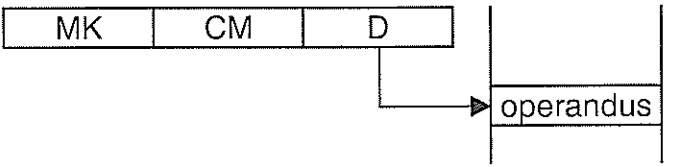
Memória címzési módok

Az utasítás címrésze alapján az operandus memóriabeli címe meghatározható. A címrész helyén valójában csak az egyik címkomponens van, amit displacementnek (D), eltolásnak nevezünk. Ezt és a címzésmódtól függően esetleg más komponen-

seket is felhasználva történik meg az operandus operatív tárbeli tényleges címének meghatározása.

Direkt címzésmód

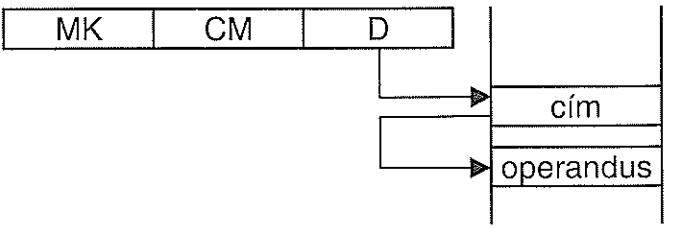
Az eltolás (D) az operandus tényleges címét tartalmazza. Ez a cím csak pozitív érték lehet, hiszen tényleges tárbeli címet jelöl. Operandus tárbeli címének meghatározása látható a **2.9.** ábrán direkt címzésmód esetén. *N* bites eltolás esetén a címezhető memóriatartomány $0 - (2^N - 1)$.



2.9. ábra. Direkt címzés

Indirekt címzésmód (cím címének megadása)

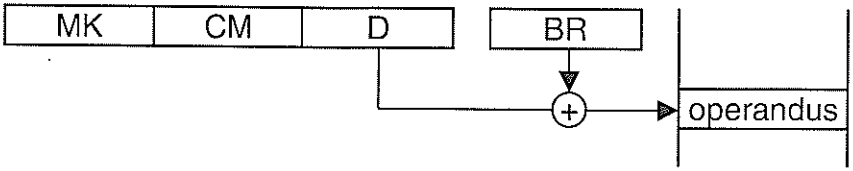
Ha D nem elég hosszú a tényleges cím megadásához (vagy ha pl. a tényleges címet a program állította elő valamilyen művelettel), akkor a címet a tárban helyezzük el, és az eltolás a cím helyét jelöli ki a tárban (ezért szintén csak pozitív érték lehet). Ilyenkor a tár adatszélessége, azaz az egy címen lévő adatbitek száma határozza meg a címezhető memóriatartományt. Mivel egy tárolóbeli helyen több bájton helyezhető el a tényleges cím, az indirekt címzési mód jóval nagyobb tártartomány címzését teszi lehetővé, mint a direkt címzés. Az operandus tárbeli címének meghatározása látható a **2.10.** ábrán indirekt címzésmód esetén. Maga az operandus csak két tárhozzáfordulással érhető el, mert az első lépésben csak az operandus címét tudjuk meg.



2.10. ábra. Indirekt címzés

Bázisrelatív címzés

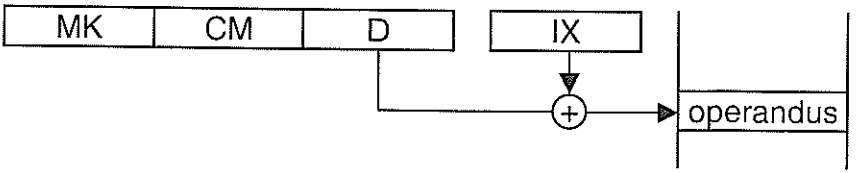
Az operandus tárbeli címét egy címzésre használt regiszter, az ún. bázisregiszter és az utasításban levő eltolás összege adja. Azért is nevezzük a címzésmódot bázisrelatívnak, mert a bázisregiszter által mutatott címhez képesti eltolással jelöljük ki az operandus tárbeli helyét. Az eltolást ebben az esetben előjeles, többnyire kettes komplemens kódban ábrázoljuk. Operandus tárbeli címének meghatározása látható a **2.11.** ábrán bázisrelatív címzésmód esetén. A címezhető memóriatartományt a bázisregiszter mérete határozza meg. A bázisregiszter tartalmának módosításával a memória más-más szelete érhető el.



2.11. ábra. Bázisrelatív címzés

Indexelt címzés

Az operandus tárbeli címét egy címzésre használt regiszter, az ún. indexregiszter és az utasításban szereplő eltolás összege adja (hasonlóan a bázisrelatív címzéshez). Az operandus tárbeli címének meghatározása látható a **2.12.** ábrán. Homogén adatszerkezeteken végzett ciklikusan ismétlődő műveleteknél nagyon hasznos, ha a processzor külön utasítás nélkül el tudja érni a következő adatelemet. Ez csak úgy lehetséges, ha a cím automatikusan növelődik (inkrementálódik), ill. csökken (dekrementálódik). Az indexregiszter ilyen tulajdonságokkal rendelkezik (szemben a bázisregiszterrel, aminek tartalma a műveletek végzése során változatlan). A gyakorlatban az indexregisztert a ciklus előtt feltöltjük az adattömb kezdőcímével, majd a ciklusutasítás automatikusan csökkenti, ill. növeli a regiszter tartalmát.

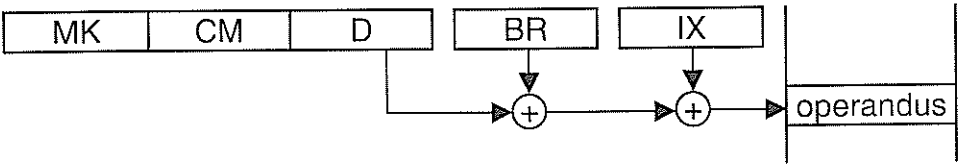


2.12. ábra. Indexelt címzés

Bázisrelatív-indexelt címzés

Az operandus tárbeli címének meghatározása három címkomponens alapján lehetséges. Az operandus címének meghatározása a **2.13.** ábrán látható. **Az effektív címet a bázisregiszter tartalmának, az indexregiszter tartalmának és az utasításban szereplő eltolásnak az összege eredményezi.**

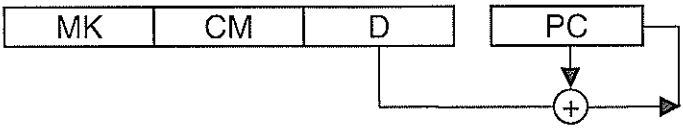
A memóriában áthelyezhető tömbök címzésére használható.



2.13. ábra. Bázisrelatív-indexelt címzés

Önrelatív címzés (PC-relatív címzés)

Általában utasításcímzéshez használatos. A következő utasítás tárbeli címét a PC tartalma és az utasításban szereplő eltolás (D) összege adja (**2.14.** ábra). Az eltolás kettes komplemens kódú (előjeles) szám lehet, és az eredmény a PC új értéke lesz.

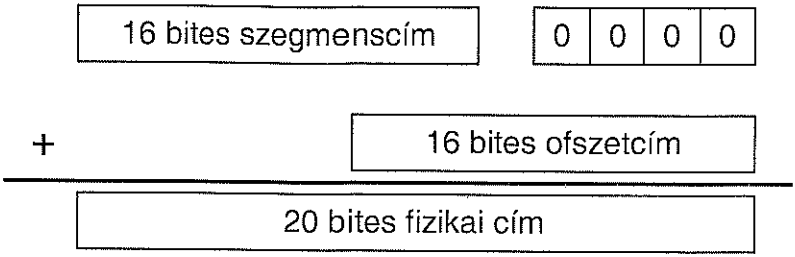


2.14. ábra. PC relatív címzés

Szegmentált címzés

Abban az esetben alkalmazzuk, ha a processzor címtere nagy és sem indirekt címzéssel, sem bázisrelatív címzéssel nem tudjuk (vagy a feleslegesen hosszú címek elkerülése érdekében nem akarjuk) elérni a teljes címtartományt. Ebben az esetben a címet egy címszámító egység (ún. címaritmetika) állítja elő két regiszter tartalmának felhasználásával. Az egyik regiszter (szegmensregiszter) tartalmát néhány bittel balra léptetve (szorzás) egy ún. szegmenscímet kapunk, majd ehhez adjuk hozzá az eltolást. Ez a cím az eltolás mértékével több bitből áll, így nagyobb tartomány címzésére alkalmas. A szorzás miatt viszont így nem érjük el az összes memóriarekeszt. A címzés során a szegmensregiszter a címezni kívánt tartomány, szegmens kezdetét jelöli ki. A szegmensen belüli eltolás egy másik regiszterben megadott értékkel adható meg (ofszet). A szegmentált memóriacímzés esetén az ef-

fektív (fizikai cím) meghatározása látható a **2.15.** ábrán. Ilyen címzést végez pl. az Intel 8086 processzor.

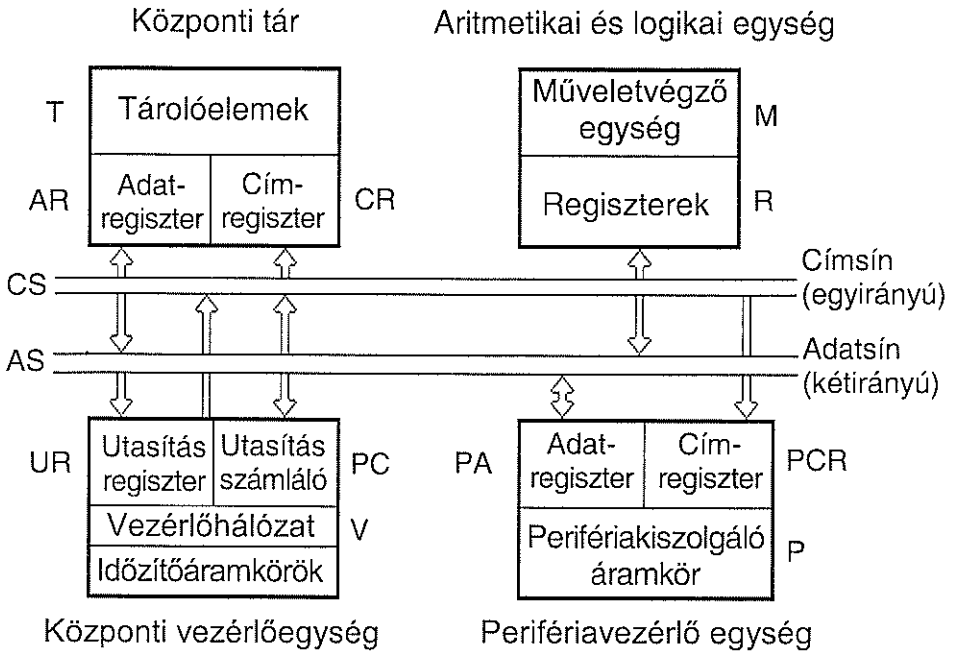


2.15. ábra. Szegmentált címképzés

Mint a példából látható, a szegmenscím összegzés előtti balra léptetésére (szorzására) csak azért van szükség, hogy a felesleges nullákat ne kelljen tárolni. A szegmentált címzés lényegében a tároló szegmensekre (azonos méretű, a példában 16 bájtos részekre) osztását jelenti, ahol a szegmensen belüli címeket az ofszettel adjuk meg.

2.4. Utasítás-végrehajtás

Az utasítás működésének leírásához egy nagyon egyszerű szimbolikus nyelvet fogunk használni. A nyelvben összesen egy műveletet alkalmazunk, a mozgatás műveletet, ezt a „←" jellel jelöljük. Tulajdonképpen minden utasítás felfogható adatmozgatások sorozatának. Ha az adatmozgatás a memória rekeszének tartalmát hozza be a processzorba, akkor betöltés történik. Ha az adatot az ALU-ba mozgatjuk műveletvégzés történik. Az utasítás végrehajtásához szükséges adatfolyamot a vezérlőjelek irányítják, amelyet a vezérlőegység állít elő. A nyelvben a vezérlőjeleket konkrétan nem írjuk le, mert ez jelentősen bonyolítaná az utasításleírást, a vezérlés tényét csupán jelöljük. A leíráshoz használnunk kell még egy mikrogépmodellt, amely rendelkezik a végrehajtáshoz szükséges egységekkel. Az egységeket rövidítésekkel jelöljük, ezek lesznek a nyelvünk változói, amelyek az adatmozgatás során változtatják a tartalmukat. A modell bonyolultsága szintén meghatározza a leírás összetettségét. Nem célszerű túl egyszerű modellt használni, mert így az utasítás leírása túl bonyolult lesz. Ezért a modellben háromsínes architektúrát használunk (a vezérlősínt nem ábrázoljuk, mert a vezérlőjeleket nem írjuk le). A leíráshoz használt mikrogépmodell a **2.16.** ábrán látható. Továbbá az egyszerűség kedvéért feltételezzük, hogy minden utasításunk egycímes utasításszerkezetű.



2.16. ábra. Mikroszámítógép felépítése

2.4.1. Az utasítás-végrehajtás vezérlése

A vezérlőegység az utasítás műveleti kódja alapján határozza meg, hogy a processzornak milyen sorrendben és milyen vezérlőjelsorozatot kell előállítania. Az utasítások a processzor számára több lépésben végrehajtható vezérlési feladatot jelentenek. Minden utasítás felbontható egy ún. gépi ciklus sorozatra. Minden processzor néhány tipikus gépi ciklusból állítja össze az összes utasítását. A gépi ciklusok sorrendjét az utasítás jellege határozza meg.

A tipikus gépi ciklusok a következők.

- **Utasításlehívás (fetch).** Minden utasítás első gépi ciklusa, az utasítás beolvasását végzi az operatív tárból, majd a dekódolt utasítástól függően ezt még követheti több más gépi ciklus.
- **Memóriaolvasás.** A memória címzését és adat szállítását végzi a megcímzett memóriarekeszből a processzor egy belső regiszterébe.
- **Memóriaírás.** A memória címzését és adatszállítását végzi a processzor egy belső regiszteréből a megcímzett memóriarekeszbe.
- **Port-olvasás.** Periféria címzését és adat szállítását végzi a megcímzett periféria adatregiszteréből a processzor egy belső regiszterébe.

- **Port-írás.** Periféria címzését és adat szállítását végzi a processzor egy belső regiszteréből a megcímzett periféria adatregiszterébe.
- **Várakozás (wait).** Szünet beiktatása: ha a processzornak valamilyen folyamattal szinkronizáltan kell működnie, és a folyamat még nem zajlott le, pl. lassú periféria kezelésénél.
- **Belső műveletek.** Ezek a folyamatok a processzor belsejében játszódnak le, ilyen folyamat pl. egy belső ALU művelet. Időbeli lefolyásuk a felhasználó által közvetlenül nem követhető, mert a processzor ilyenkor nem használja a rendszersíneket.

A végrehajtás menetét a processzor belső felépítése határozza meg. Katalógusból szerezhetünk róla információt.

2.4.2. Az utasítás-végrehajtás időzítése

Az utasítás végrehajtásánál a processzor által kiadott vezérlőjelek megfelelő sorrendje teszi lehetővé az utasítás végrehajtását. A megfelelő sorrend azonban még nem elegendő. Figyelembe kell venni, hogy az egyes jelek a sínmeghajtókon véges sebességgel terjednek, ill. a rendszert alkotó áramkörök működési sebessége is véges. Az egyes vezérlőjeleket ezért csak megfelelő késleltetéssel, ütemezéssel adhatja ki a processzor.

A mikroprocesszor által szolgáltatott jelek csak meghatározott sorrendben és időkülönbséggel juthatnak a sínekre.

A jelek megfelelő ütemezését a mikroprocesszor az órajel segítségével végzi. Az ütemezés garantálja, hogy a rendszersínre egyszerre csak egy, az éppen aktuális információ jusson. **Az órajel periódusideje az ütem. A gépi ciklus végrehajtási ideje ennek egész számú többszöröse.**

Az órajellel ütemezett vezérlőjelek, adatjelek és címjelek mérésével jól követhetők a processzor gépi ciklusai, ill. a gépi ciklusokban elvégzett tevékenységek (kivéve a belső műveleteket).

A gépi ciklusok vizsgálatához szükséges tipikus vezérlőjelek a következők.

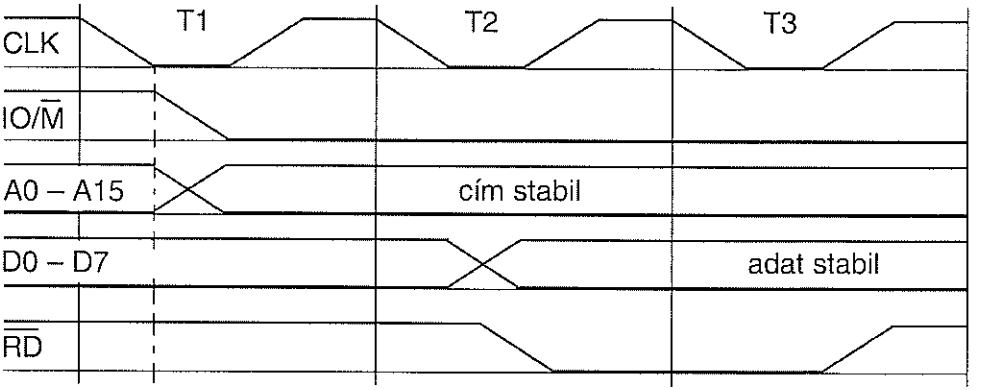
- CLK: órajel,
- IO/$\overline{M}$: portcím - memóriacím „szétválasztó" vezérlőjel,
- A0-A15: címjelek,
- D0-D7: adatjelek,
- $\overline{RD}$: olvasás engedélyezés vezérlőjel,
- $\overline{WR}$: írás engedélyezés vezérlőjel.

A következőkben néhány jellegzetes gépi ciklus leírását és idődiagramját adjuk meg.

Memóriaolvasás gépi ciklusának időzítése

A ciklus során a következő három esemény követi egymást (**2.17.** ábra).

- Az első órajelciklus lefutó élének hatására a processzor kiadja a címsínre az olvasni kívánt memóriarekesz a címét (A0-A15). A következő eseményig ez a cím már nem változhat, a címsín jeleinek feszültségszintjei állandósulnak. A cím stabillá válását követően a mikroprocesszor adat fogadására kész állapotba kerül. Ekkor a mikroprocesszor adatcsatlakozási pontjai bemenetként működnek. Ezzel egyidőben az IO/$\overline{M}$ jel alacsony szintje engedélyezi a memória használatát.
- A második órajelciklus lefutó élét követően a processzor az $\overline{RD}$ vezérlőjel aktiválásával (alacsony szint) engedélyezi a memória adatsínre kapcsolódását. Ekkor a memória adatcsatlakozási pontjai kimenetenként működnek. A megcímzett memóriarekesz tartalma (D0-D7) az adatsínre kerül.
- Az adatvezetékek jelszintjeinek stabillá válása után, a harmadik órajelciklus lefutó élét követően, az adatsín jelei a processzor egy belső regiszterébe íródnak. Ezután a $\overline{RD}$ vezérlőjel magas szintű lesz, ezzel megszűnik a memória adatsínre csatlakozása.



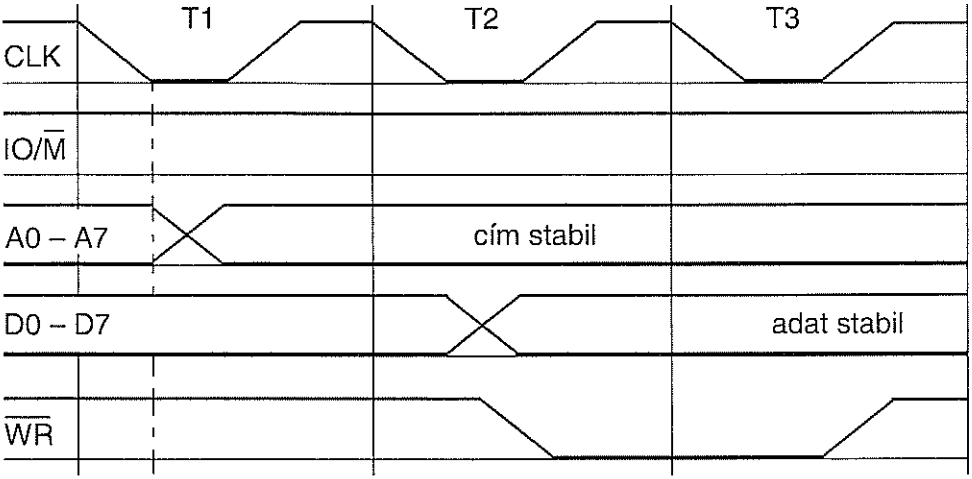
2.17. ábra. A memóriaolvasás időzítése

Port- (periféria-) írás gépi ciklusának időzítése

A ciklus során a következő három esemény követi egymást (**2.18.** ábra).

- Az első órajelciklus lefutó élének hatására a processzor kiadja a címsínre (ill. mivel általában csak kevés perifériát használunk, annak csak az alsó nyolc vonalára) az írni kívánt periféria címét (A0-A7). A következő eseményig ez a cím már nem változhat, a címsín jeleinek feszültségszintjei állandósulnak. A cím stabillá válását követően a mikroprocesszor adat küldésére kész állapotba kerül. Ekkor a mikroprocesszor adatcsatlakozási pontjai kimenetként működnek Ezzel egyidőben az IO/$\overline{M}$ jel magas szintje engedélyezi a perifériavezérlő használatát.

- A második órajelciklus lefutó élét követően a processzor a $\overline{WR}$ vezérlőjel aktiválásával (alacsony szint) engedélyezi a perifériavezérlő adatsínre kapcsolódását. A perifériavezérlő adatcsatlakozási pontjai bemenetként működnek. A processzor belső regiszterének (akkumulátor) tartalma az adatsínre (D0-D7) kerül.
- Az adatvezetékek jelszintjeinek stabillá válása után (a harmadik órajelciklus lefutó élét követően) az adatsín jelei a perifériavezérlő adatregiszterébe íródnak. Ezután a $\overline{WR}$ vezérlőjel magas szintű lesz, ezzel megszűnik a perifériavezérlő adatsínre csatlakozása.



2.18. ábra. Perifériaírás időzítése

2.4.3. Az alaputasítások végrehajtása

A következőkben a processzorok utasításkészletéből néhány alaputasítást nézünk végig, az általunk definiált modellben és nyelven. Elvileg több jó megoldás is elképzelhető. A végrehajtás egy konkrét, megvalósított rendszerben nyilván eltérhet ettől. Egyelőre csak az utasításvégrehajtás elveit szeretnénk ismertetni, konkrét processzoros rendszerrel a könyv 6. fejezete foglalkozik.

> Azokat az utasításokat, amelyek végrehajtásához a fetchen kívül legalább még egy memóriaciklusra van szükség, memóriareferens-utasításoknak nevezzük.

A memória címzésére az alaputasítások tárgyalásakor direkt címzésmódot használunk, ahol ettől eltérünk, azt a leírás előtt jelezzük.

Regiszter töltése (LOAD)

A memória megcímzett rekeszének tartalmát egy regiszterbe töltjük.

CS ← PC az utasítás címe a címsínre kerül (**fetch kezdete**),
CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) az operandus címe a címsínre kerül,
CR ← CS a cím beíródik a tár címregiszterébe,
AR ← M(CR) a megcímzett rekesz tartalma az adatregiszterbe íródik,
AS ← AR az operandus az adatsínre kerül,
R ← AS az adatsínről az operandus a regiszterbe íródik.

Regiszter tartalmának mentése (STORE)

A regiszter tartalmát adott (megcímzett) memóriarekeszbe írjuk.

CS ← PC az utasítás címe a címsínre kerül (**fetch kezdete**),
CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) a memóriarekesz címe a címsínre kerül,
CR ← CS a rekesz címe a címregiszterbe íródik,
AS ← R a regiszter tartalma az adatsínre kerül,
AR ← AS az adatsínről az adat a memória adatregiszterébe íródik,
M(CR) ← AR az operandus az adatregiszterből a megcímzett rekeszbe íródik.

Összeadás (ADD)

Az első operandus legyen az utasításban megadott címen (D), a 2. operandus pedig az R2 regiszterben. Az eredmény az R2 regiszterben tárolódjon.

CS ← PC az utasítás címe a címsínre kerül (**fetch kezdete**),

CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) az utasítás displacementje a címsínre kerül,
CR ← CS a cím beíródik a memória címregiszterébe,
AR ← M(CR) a memória megcímzett rekesze az adatregiszterbe kerül,
AS ← AR az adatregiszter tartalma a címsínre kerül,
R1 ← AS a kiolvasott adat R1-be íródik,
R2 ← R1+R2 az ALU elvégzi az összeadást és az eredményt visszaírja R2-be.

Ugrás (JUMP)

CS ← PC az utasítás címe a címsínre kerül (**fetch kezdete**),
CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) az ugrási cím kikerül a címsínre,
PC ← CS a cím a címsínen keresztül beíródik a PC-be.

Adat beolvasása perifériáról regiszterbe (IN)

CS ← PC az utasítás címe a címsínre kerül, (**fetch kezdete**),
CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma, a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) a periféria címe a címsínre kerül,

PCR ← CS a cím a perifériavezérlő címregiszterébe íródik,
PAR ← P(PCR) a megcímzett port tartalma a perifériavezérlő adatregiszterébe íródik,
AS ← PAR az adatregiszter tartalma az adatsínre kerül,
R ← AS az adatsínről az adat a processzor regiszterébe kerül.

Adat írása regiszterből perifériára (OUT)

CS ← PC az utasítás címe a címsínre kerül (**fetch kezdete**),
CR ← CS a cím beíródik a tár címregiszterébe,
PC ← PC+1 következő utasításcím előkészítése,
AR ← M(CR) a memória CR-rel megcímzett rekeszének tartalma a memória adatregiszterébe íródik,
AS ← AR az utasítás kódja az adatsínre kerül,
UR ← AS az utasítás kódja beíródik a processzor utasításregiszterébe,
V← UR(MK) a vezérlőegység dekódolja (értelmezi) a műveletet, ettől függ az utasítás további végrehajtása, további gépi ciklusai (**fetch vége**),
CS ← UR(CÍM) a periféri címe a címsínre kerül,
PCR ← CS a cím a perifériavezérlő címregiszterébe íródik,
AS ← R az R regiszter tartalma az adatsínre kerül,
PAR ← AS az adatsín tartalma a perifériavezérlő adatregiszterébe íródik,
P(PCR)←PAR a perifériavezérlő írja a megcímzett perifériát.

Ellenőrző kérdések, feladatok

1. Soroljuk fel a mikroszámítógép rendszertechnikai elemeit, ismertessük feladatukat!
2. Ismertessük a mikroszámtógép utasításszerkezetét!
3. Soroljuk fel a mikroszámítógép fontosabb címzésmódjait!
4. Ismertessük a mikroszámítógép legfontosabb gépi ciklusait!
5. Ismertessük a memóriaírás vezérlésének időzítését!
6. Ismertessük a bázisrelatív címzésmódú ADD A utasítás végrehajtását szimbolikus nyelven, háromsínes modell esetén!
7. Ismertessük az indirekt címzésmódú JMP utasítás végrehajtását szimbolikus nyelven, háromsínes modell esetén!

3. SÍNRENDSZEREK

A sín olyan szabványosított jelvezeték-csoport, amelyen keresztül a mikroszámítógépes rendszer részegységei közötti kommunikáció megvalósul.

A kommunikáció során adatok, címek, ill. a vezérléshez szükséges információk átvitelét kell biztosítani. A központi egység logikailag a **címsínen,** az **adatsínen** és a **vezérlősínen** keresztül kommunikál a rendszer egységeivel. A három sín fizikailag is jól elkülöníthető, együttesen alkotják a mikroszámítógép sínrendszerét. A sínrendszer használatának előnye, hogy a szabványosított jelhasználat és vezetékkiosztás miatt könnyen cserélhetők a csatlakoztatott eszközök és azok vezérlőkártyái, így gyártó- és gépfüggetlenné válik ezek használata. A sínrendszer előírásait teljesítő kártyák bármely gépen használhatók. A sínek legfontosabb jellemzője, hogy hány vezetéket foglalnak magukba, ezt a sín szélességének nevezzük. A sín működésére jellemző, hogy egyidejűleg mindig csak egy eszköz használhatja.

3.1. A sínrendszer részei

Címsín (Address Bus)

Az a sín (jelvezetékcsoport), amelyen keresztül a mikroprocesszor megadja azt a címet, amellyel az adatátvitel végpontja vagy forrása kiválasztható. A címsínre jelet (címet) csak a processzor vagy a DMA vezérlő adhat, a többi rendszerelem csak olvashatja. A címsín vezetékeinek száma határozza meg a megkülönböztethető címek számát (I/O vagy memória). **n címvezetékkel 2^n cím különböztethető meg.** A napjainkban alkalmazott processzorok 32–64 címvezetéket használnak.

Adatsín (Data Bus)

Ezen keresztül jut az adat **a megcímzett helyről a CPU-ba, vagy a CPU-ból a megcímzett helyre.** Az adatsínre a CPU, memória vagy I/O egység juttathat adatot.

Az adatsínen **kétirányú az átvitel**. Az adatsín vezetékszáma határozza meg, hogy a μP hány bitet tud egyszerre mozgatni (8, 16, 32 vagy 64 bit).

Vezérlősín (Control Bus)

Az a jelvezetékcsoport, amelyen a rendszer vezérlőjelei terjednek. Ezek lehetnek
- adatátvitelt vezérlő jelek,
- megszakítást vezérlő jelek,
- sínhasználatot vezérlő jelek,
- szinkronizációs jelek.

3.2. A sínrendszer megvalósítása

A gyakorlatban meg kell különböztetnünk a processzoron belüli kommunikációra létrehozott belső sínrendszert és a külső egységek csatlakoztatására az alaplapon létrehozott külső sínrendszert.

Belső sínrendszer

A belső sínrendszer a processzoron belüli egységeket kapcsolja össze. Nagysebességű átvitelt valósít meg, órajele megegyezik a processzoréval. Kialakítását az elérni kívánt teljesítmény szabja meg. Általában a processzoron belüli vezérlőjelek szállítására nincs külön vezérlősín. A közös adatsínt és címsínt tartalmazó egysínes architektúráknak ma már nincs gyakorlati jelentősége. A külön címsínt és adatsínt tartalmazó struktúrán kívül alkalmaznak még ún. háromsínes rendszert, ahol külön adatsínt használnak az írásra és olvasásra. Így megoldható a közel egyidejű írás és olvasás, ezzel a növelhető a processzor sebessége.

Külső sínrendszer

A külső sínrendszer több részre van osztva. Az egyes rendszerelemek eltérő módon csatlakoznak a rendszerhez.

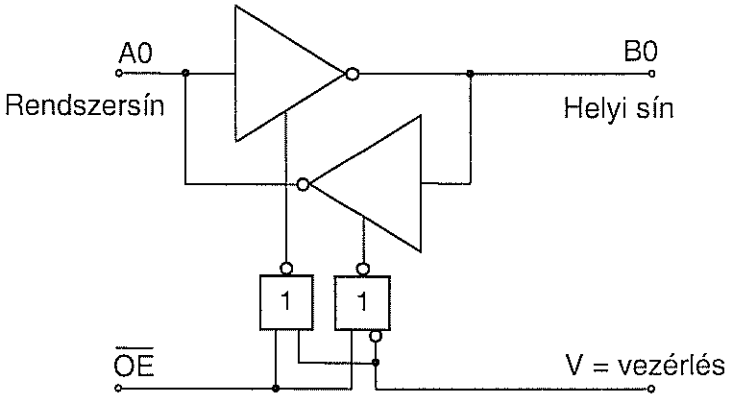
Helyi sín (local bus). A mikroprocesszorral közvetlen kapcsolatban álló egységeket csatlakoztatja a rendszerhez. Ehhez kapcsolódik pl. a memória, egyes meghatározott típusú grafikus kártyák stb. Az átvitel a processzor órajelével szinkronban történik, tehát nagysebességű.

Rendszersín (system bus). Meghajtókkal elválasztott sínrendszer. Az elválasztás oka, hogy a mikroprocesszor nem képes tetszőleges számú egységet közvetlenül vezérelni, ezért a rendszersínt sínmeghajtó áramkörök kötik össze a mikroprocesszorral (bus interface). Lassúbb átvitelt tesz lehetővé, alapvetően I/O eszközök csatlakoztatására való.

3.3. Sínmeghajtó egység

A sínmeghajtó egység fontos része a sínrendszernek, feladata a megfelelő nagyságú jelek létrehozása a rendszersín számára, az egyes sínrészek és egységek leválasztása, valamint a sínfoglalások szabályozása.

Az egységek sínre csatlakoztatását háromállapotú (tristate) sínmeghajtó áramkörök végzik (pl. 8286, 8287). Ezeken a vezérléstől függően kétirányú adatforgalom lehetséges. Kétirányú sínmeghajtó áramkör rajza látható a **3.1.** ábrán.



3.1. ábra. Kétirányú sínmeghajtó áramkör

A sínmeghajtó áramkör létesít kapcsolatot az A rendszersín és a B helyi sín azonos helyiértékű vezetékei között. A kapcsolat létrehozását és az adatátvitel irányát vezérlőjelek határozzák meg.

- V az adatátvitel irányát adja meg

  0: a jeleket a rendszersínről a helyi sínre visszük át (A0→B0),

  1: a jeleket a helyi sínről a rendszersínre visszük át (B0→A0.

- OE (Output Enable), kimenet-engedélyezés

  0: nagyimpedanciás (Hi Z) állapot, nincs kapcsolat a sínvezetékek között,

  1: aktív állapot, a két sínvezeték között a V vezérlőjel által kijelölt irányú meghajtó kapcsolatot létesít, a másik nagyimpedanciás állapotban marad.

3.4. Sínhasználat

3.4.1. A sínhasználat fázisai

A sínt egyidőben csak egy eszközpár használhatja. Ezek közül az egyik lehet a **kezdeményező (master) eszköz**, a másik eszköz **passzív (slave)**, csak végrehajtja a mastertől kapott utasításokat. Mikroszámítógépes rendszerek esetén aktív eszköz csak a processzor vagy a közvetlen memória-hozzáférést alkalmazó I/O eszköz lehet. A sínt használó eszközpáron kívül a többi egység nagyimpedanciásan leválik a sínekről.

A sínhasználat a következő négy eseményből áll:

- sínhasználatkérés (Bus Request),
- sínengedélyezés (Bus Grant),
- sínhasználat (Bustransaction),
- sínfelszabadítás (Release).

Ezt a négy eseményt nevezzük együttesen sínciklusnak.

A master feladatai:

- elindítja és vezérli az átvitelt,
- címet küld.

A slave feladatai:

- válaszol az átviteli igényre,
- a sínre teszi vagy fogadja az adatokat.

3.4.2. Sín-arbitráció

Egyidejűleg több master is igényelheti a sín használatát. Ilyenkor valamilyen módszerrel el kell dönteni, hogy a masterek milyen sorrendben kapják meg a sín használatának a jogát. Ezt az eljárást nevezzük sín-arbitrációnak. **Azt a hardver egységet, amely a sínfoglalási kérelmeket fogadja és elbírálja, sín-arbiternek nevezzük.** Ez lehet önálló hardveregység, vagy lehet a processzor része. A sínhasználati jogot az arbiter időosztásos módszerrel vagy dinamikusan oszthatja szét.

Az **időosztásos (time sharing) módszer** lényege, hogy minden master egy meghatározott időszeletre kapja meg a sínhasználat jogát. Ez a módszer nem alkalmazható megfelelő hatékonysággal, ha a masterek sínhasználati igénye megközelítőleg nem egyforma, mert a kis sínfoglalási igényű master feleslegesen várakoztatja a nagyobb igényű mastereket.

A **dinamikus sínhasználat** során a masterek között prioritásokat (fontossági sorrendet) állapítunk meg. Ha azonos időpontban több master jelzi sínhasználati igényét, a sínhasználati jogot a legnagyobb prioritású kapja meg, az összes többi kérelmet egy várakozási sorba helyezik. Ennek a módszernek az a hátránya, hogy egy olyan rendszernél, ahol a sínfoglalások gyakoriak, előfordulhat, hogy a kisebb prioritású master soha sem kap sínhasználati jogot. Ennek feloldására alkalmazhatjuk azt a módszert, hogy a nagyobb prioritású master sínfoglalási igénye a várakozó sorban nem előzheti meg a kisebb prioritásúakat, csak a sor végére kerülhet. Ez az ún. **körben forgó sínfoglalási algoritmus**. A rendszer hatékonyságának növelése érdekében sokféle módszer terjedt el a sínfoglalás szabályozására.

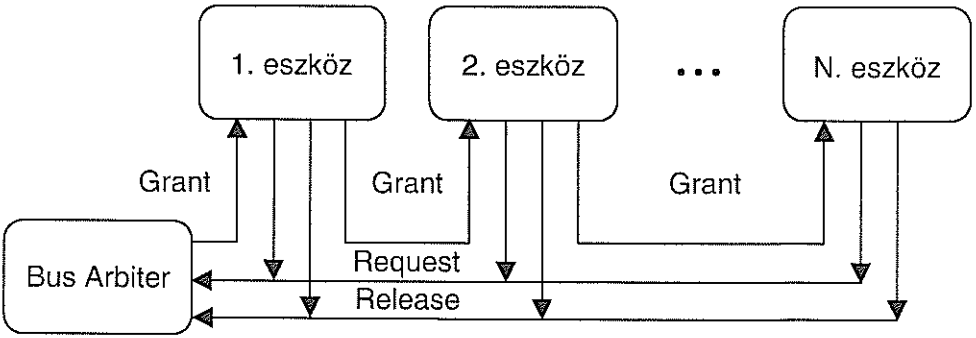
3.4.3. Sínfoglalás

A sínhasználati igényt a masterek a **Request** jellel jelzik az arbiternek. Az arbiter az alkalmazott algoritmusnak megfelelően kiválaszt egy mastert és az igény elfogadását a **Grant** jellel jelzi vissza. Ez hardveresen soros, ill. párhuzamos sínfoglalással történhet.

Soros sínfoglalás

A masternek közös Request (kérés), és sorosan felfűzött (daisy-chain) Grant (engedélyezés) vezérlővonala van. A master nem tudja, hogy a közös Request vonalon melyik eszköz kérte a sín használatának a jogát, de ha ez lehetséges, a Grant vonalon keresztül engedélyezi azt. Az adott eszköz csak akkor engedi tovább a Grant jelet, ha saját maga nem igényli a sínt. Így a felfűzés sorrendje határozza meg az eszközök prioritását.

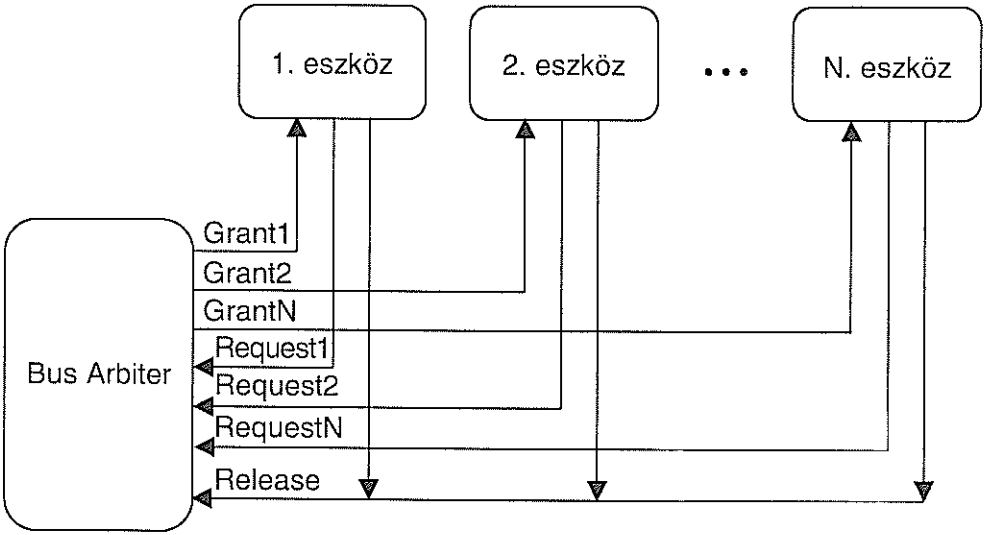
A soros sínfoglalás hardveres felépítése a **3.2.** ábrán látható. A módszer előnye az egyszerű felépítés, **hátránya, hogy a kisebb prioritású** (a láncban hátrább álló) **eszközök sínhozzáférési lehetőségei nagyon korlátozottak.**



3.2. ábra. Soros sínfoglalás

Párhuzamos sínfoglalás

Minden sínhasználó eszköz saját Request (kérés) és Grant (engedélyezés) vezérlővonallal rendelkezik. A prioritási sorrendet az arbiter prioritási algoritmusa határozza meg. A párhuzamos sínfoglalás hardverfelépítése látható a **3.3.** ábrán. **Hátránya, hogy a megvalósítás jóval bonyolultabb**, viszont a sínfoglalások többféle algoritmus szerint végezhetők.



3.3. ábra. Párhuzamos sínfoglalás

3.4.4. Sínvezérlés

Az átvitelben résztvevő eszközöknek meghatározott szabályok szerint, összehangoltan kell működniük. Mechanikai és villamos követelményeket kell megfogalmazni, amelyek a rendszerhez csatlakozás feltételeit írják le. Ezeknek a szabályoknak az összességét nevezzük **sínprotokollnak**. Ezzel kapcsolatban a gyakorlatban sokféle sínszabvány létezik. Az ismertebb sínrendszerek a következők: ISA, EISA, MC, VLB, PCI, AGP, USB.

Az adatátvitel vezérlése szempontjából a sínrendszerek alapvetően aszinkron- és szinkronütemezésű sínekre oszthatók.

Szinkron sínvezérlés

A sínen kommunikáló eszközök azonos órajellel ütemezve dolgoznak. Az adás és vétel mindig azonos sebességű, nem kell kapcsolatfelvétel és visszaigazolás. Így nagyobb átviteli sebesség érhető el, viszont hátrányként említhető, hogy azonos órajelet kell biztosítani az összes egység számára.

Aszinkron sínvezérlés

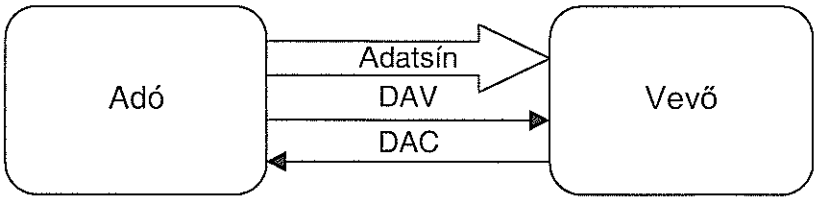
A sínen kommunikáló eszközök bármely időpontban igényelhetik a síneket, és átvitelt folytathatnak. **A pontos, zavartalan átvitelhez kapcsolatfelvétel és vételi visszaigazolás szükséges (ún. handshake-eljárás).** Nem kell közös órajelet vezetni a sínre kapcsolt összes eszközhöz, ezért eltérő sebességű eszközök kiszolgálását is lehetővé teszi, viszont a kapcsolatfelvételhez a handshake- („kézfogás") eljárás beépítése szükséges.

Handshake eljárás

A biztonságos adatátvitelhez a vevőt informálni kell az adatcsomag elküldéséről és a vevőnek vissza kell jelezni az adó felé, hogy az elküldött adatcsomag hibátlanul megérkezett. Ez a célja az ún. handshake-, „kézfogás" eljárásnak (**3.4.** ábra).

Az eljárás fontosabb lépései:

- az adó a **DAV** (DATA VALID, ADAT ÉRVÉNYES) vezérlőjellel tájékoztatja a vevőt, hogy adatot helyezett a sínre;
- a vevő érzékeli a DAV jel aktív szintjét, olvassa a sín adatait, ellenőrzi az átvitel hibátlanságát, majd a hibátlan adatátvitelt a **DAC** (DATA ACCEPTED, ADAT ELFOGADVA) vezérlőjellel jelzi az adónak;
- a folyamat kapcsolatbontásig tart.



3.4. ábra. Handshake-eljárás

3.5. I/O alrendszer és a mikroszámítógép kapcsolata

Az I/O egységek és a processzor kapcsolatát a perifériavezérlő áramkör valósítja meg a rendszersínen keresztül. Ezen keresztül válnak a perifériák címezhetővé, azaz a processzor által elérhetővé, ill. ez szabályozza vezérlőjelekkel a perifériák sínhasználatát.

3.5.1. Az I/O eszközök címzése

A perifériális eszközöket a memóriarekeszekhez hasonlóan bináris sorszámokkal címezi a processzor. Ezeket **I/O címeknek (portcímeknek)** nevezzük. A processzor a perifériákat közvetlen vagy közvetett portcímzéssel címezheti.

Közvetlen portcímzés

A processzor az IO/$\overline{M}$ vezérlőjellel rendelkezik, amellyel szétválaszthatók a memóriacímek és portcímek.

Ha IO/$\overline{M}$ = 0, akkor a memóriacímzés,

ha IO/$\overline{M}$ = 1, akkor a portcímzés engedélyezett.

A processzor utasításkészletében ilyenkor külön utasítás szükséges a perifériák kezeléséhez (pl. IN, OUT). Ezt a módszert alkalmazzák pl. az Intel processzorok.

Közvetett (memóriába ágyazott) portcímzés

A portcímeket nem különböztetjük meg a memóriacímektől. A processzor a portokat és a memóriát ugyanazzal az utasítással kezeli, mintha az I/O eszköz regisztere az operatív tár része lenne. Ilyenkor az operatív tár egy tartományát ki kell jelölni az I/O eszközök számára. Ha a címzés erre a tartományra esik, automatikusan I/O művelet megy végbe. Ezt a módszert alkalmazzák pl. a RISC processzorok (bővebben l. a 6. fejezetben).

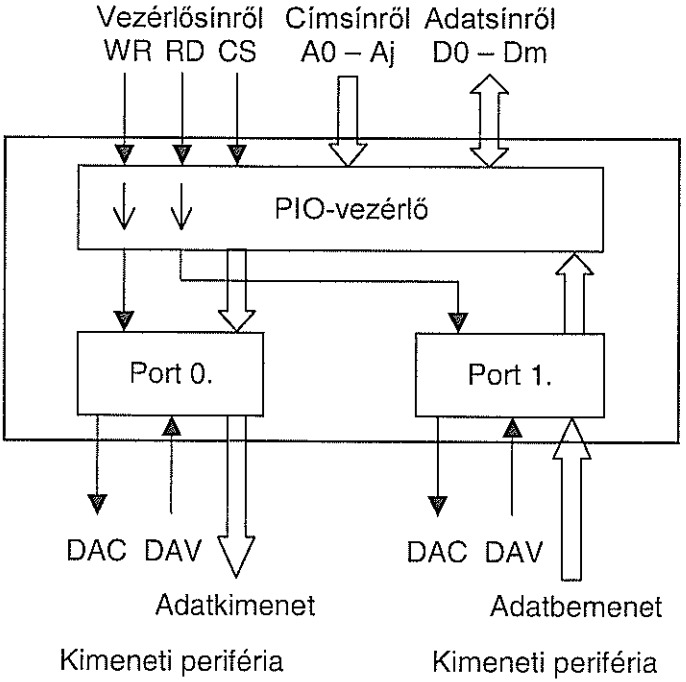
3.5.2. Az I/O átvitel típusai

A gyakorlatban igen sokféle módszer terjedt el az I/O eszközök adatátviteli feladatainak lebonyolítására. Az átvitel fizikailag történhet bitenként, ilyenkor **soros adatátvitelről** beszélünk, ill. egyszerre több vezetéken több bit továbbításával, ezt nevezzük **párhuzamos adatátvitelnek**. Továbbá az átvitel történhet szinkron módon, órajellel ütemezve, ill. aszinkron módon pl. handshake-eljárással. A soros adatátvitel az átvitel irányától függően lehet egyirányú, ún. **szimplex**, váltakozva

kétirányú, ún. **félduplex** (egyidőben csak egyirányú), vagy egyidőben kétirányú, ún. **duplex átvitel**. Az átvitel lebonyolítására céláramköröket fejlesztettek ki, amelyeket közös néven interfész- (illesztő-) áramköröknek nevezzük Ezek az adatátvitel különféle módjaihoz nyújtanak támogatást, ill. a perifériák működési paramétereit (sebesség, adatszélesség stb.) illesztik a processzorhoz. A leggyakrabban alkalmazott átviteli módokat szabványok rögzítik.

Párhuzamos adatátvitel

A kétirányú párhuzamos interfészt az IEEE 1284-1944 szabványban rögzítették. A párhuzamos adatátvitel lebonyolítására ún. PIO (Paralel Input Output) vezérlőáramköröket alkalmaznak, amelyek esetleg több port illesztését is elvégzik. PIO interfész logikai felépítése látható a **3.5.** ábrán.



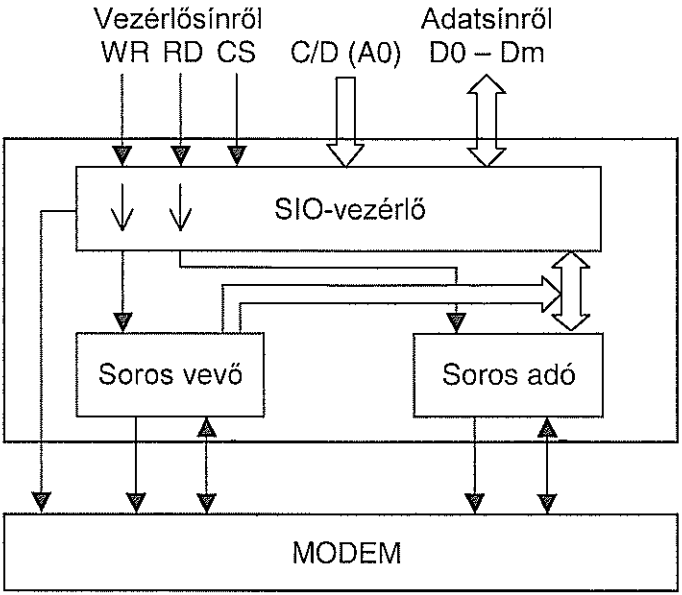
3.5. ábra. PIO interfész

PIO interfész vezérlőjelei

- CS (Chip Select): a megfelelő PIO engedélyezése,
- RD (Read): olvasásengedélyezés, az adat a porttól az adatsín felé halad,
- WR (Write): írásengedélyezés, az adat az adatsíntől a port felé halad,
- DAV (Data Valid): *adat érvényes* vezérlőjel,
- DAC (Data Accepted): *adat elfogadva* vezérlőjel.

Soros adatátvitel

Mikroszámítógépes környezetben a soros, aszinkron interfészt az RS 232C szabvány írja le. A soros vonali illesztéseket soros interfész-céláramkörök végzik. Ezeket SIO (Serial Input Output) vezérlőáramköröknek nevezzük. Egyes típusok modemes kapcsolatok lebonyolítására is képesek. SIO interfész logikai felépítése a **3.6.** ábrán látható. A SIO az adatátviteli jellemzők beállítására saját parancsregisztereket tartalmaz, ezek tartalmát az adatátvitel megkezdése előtt a processzor állítja be.



3.6. ábra. SIO interfész

SIO interfész vezérlőjelei

- CS (Chip Select): a megfelelő SIO engedélyezése,
- RD (Read): olvasásengedélyezés, az adat a porttól az adatsín felé halad,
- WR (Write): írásengedélyezés, az adat az adatsíntől a port felé halad,
- C/D (Command/Data): adat, ill. parancsregiszter kiválasztása.

3.5.3. Az I/O adatátviteli eljárásai

Az I/O eszköz és egy másik egység közötti adatátvitel háromféle módon történhet meg:

- programozott I/O átvitellel,
- megszakításos I/O átvitellel,
- közvetlen memóriahozzáféréssel (DMA).

Programozott I/O adatátvitel (polling, lekérdezés)

Az I/O eszköz és a processzor közötti adatátvitel programutasítás hatására jön létre (szoftveres megoldás). A program periodikusan, ún. **állapothurok eljárást** alkalmazva lekérdezi a perifériák állapotregiszterét. Az állapotregiszter egy adott bitjét tesztelve kiderül, hogy az adott periféria kért-e adatátvitelt. Ha igen, akkor a processzor egy rutinhívással kiszolgálja az I/O eszközt és folytatja a programot. Ennél a módszernél teljeskörűen a processzor ellenőrzi és vezérli az átvitelt. Az állapotregiszterek folyamatos tesztelése terheli a processzort, lassítja a programvégrehajtást.

Megszakításos I/O adatátvitel

A megszakítás a futó program (egy fontosabb feladat elvégzése érdekében történő) felfüggesztését, majd a megszakítást kérő tevékenység elvégzése után a megszakított program folytatását jelenti. **A megszakítási folyamatok felügyelete a megszakítási rendszer feladata.** A megszakítási rendszer a perifériák kiszolgálásának hardver/szoftver módja. A megszakítás úgy tekinthető, mint egy aszinkron eljáráshívás.

A megszakítás létrehozható:

- szoftveresen,
- I/O egység kérésére egy megszakításkérő vonalon keresztül (hardveres megszakítás),
- a mikroprocesszor hiba észlelése esetén maga is kiválthat ún. belső megszakítást (pl. 0-val való osztás esetén).

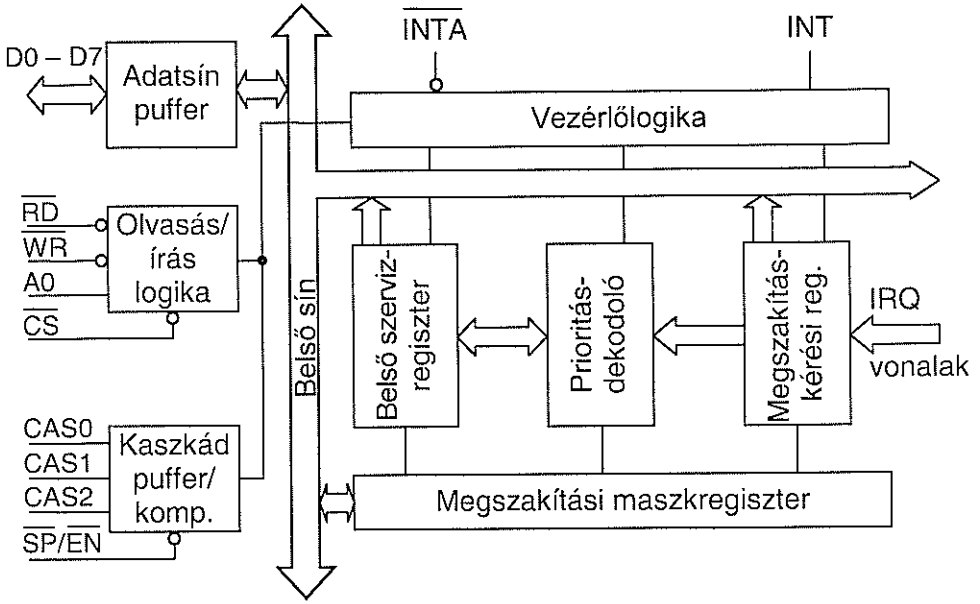
A megszakítás kiszolgálás lépései:

- futó program megállítása,
- programállapot (PC, Állapotregiszter) elmentése a verembe,
- megszakítás forrásának felismerése,
- a megfelelő kiszolgálórutin elindítása,
- visszatérés a megszakított programhoz (PC és Állapotregiszter visszaállítása).

A kiszolgálórutinok a működtető szoftver (operációs rendszer) részei. A megszakítás-kiszolgálás bonyolódik, ha a rendszerben egyszerre több eszköz kérhet megszakítást. A versenyhelyzet kivédése érdekében **minden megszakításhoz egy prioritási (fontossági) szint tartozik**. Minél nagyobb egy megszakítás prioritása, annál előbb kerül kiszolgálásra. Általában létezik egy legnagyobb prioritású megszakítás, az NMI (Non-Maskable Interrupt). Ezen kívül a hardver- és szoftvermegszakítások mindegyike maszkolható, azaz programból tiltható. Mikroprocesszoros rendszerben a megszakítások kezelése ún. megszakításvezérlő áramkörön keresztül történik (pl. Intel 8259).

Intel 8259 megszakításvezérlő

Nyolc független csatornán (IRQ0-IRQ7, Interrupt Request) érkező megszakításkérést tud kezelni. A legnagyobb prioritású kérés az IRQ0, a legkisebb prioritású az IRQ7. Az áramkör egy **maszkregisztert** is tartalmaz, amelyen keresztül a megszakítási vonalak tilthatók vagy engedélyezhetők (**3.7.** ábra).



3.7. ábra. 8259 belső felépítése

A megszakítás kiszolgálása

A vezérlő a megszakításkéréseket sorrendbe rakja, meghatározza melyik eszköznek van a legnagyobb prioritása. Először ezt fogja kiszolgálni. Ezután a mikroprocesszor INT vonalát 1-es szintűre állítja.

Amennyiben a mikroprocesszor I-flagje (megszakítás jelzőbit) 1-es értékű , a processzor fogadja a megszakításokat és az INTA (Interrupt Acknowledge) vonalon keresztül engedélyezi a megszakítást. A 8259 ezután az adatvonal alsó 8-bitjén (D0-D7) beadja a megszakítási rutin sorszámát (az ún. megszakítási vektort). A mikroprocesszor ez alapján meghatározza a megszakítási rutin címét, elmenti a PC-t és az állapotregisztert, majd futtatja a megszakítás-kiszolgáló rutint. A rutin futtatása után visszatér a megszakított programhoz.

A 8259 vezérlők kaszkádba köthetők, így a vonalak száma növelhető.

DMA adatátvitel (Direct Memory Access, közvetlen memóriahozzáférés)

Nagysebességű átvitelt tesz lehetővé a memória és az I/O készülék között a mikroprocesszor igénybevétele nélkül. A DMA vezérlő tulajdonképpen egy processzor szintű, intelligens átvitelvezérlő áramkör. A DMA adatátvitel alkalmazásának előnye, hogy a közvetlen adatátvitel miatt **az adatátvitel gyorsabb**, ill. a DMA ciklus alatt a μP belső műveleteket végezhet, ezért **a program végrehajtása is gyorsabb**. A DMA vezérlő és a mikroprocesszor egyszerre nem irányíthatják az átvitelt, mert közös sínrendszeren osztozkodnak, ezért a DMA átvitel alatt a processzor nagyimpedanciásan leválik a külső sínrendszerről. Azt, hogy a sínhasználat milyen ütemezéssel és milyen szabályok szerint történik, a DMA adatátviteli eljárás típusa határozza meg.

DMA adatátviteli eljárások

- **CPU leállás (CPU halt).** A DMA vezérlő kérésére a μP a teljes átvitel időtartamára lekapcsolódik a sínekről (nagyon lelassul a μP).
- **Időszeletelés.** A memóriaciklus két időtartamra oszlik, az egyikben a μP, a másikban a DMA vezérlő használhatja a síneket.
- **Ciklusloppás.** Csak akkor van időszeletelés, ha a DMA vezérlő jelzi sínhasználati igényét, ha nem, akkor a μP tovább működik.

A DMA vezérlő a processzorhoz hasonlóan az átviteli művelet vezérléséhez szükséges operandusait belső regiszterekben tárolja. A DMA átvitel előtt a regiszterek tartalmát a CPU állítja be. Ez után veszi át a DMA vezérlő a sínek, az I/O eszközök és a memória vezérlését.

DMA vezérlő regiszterei

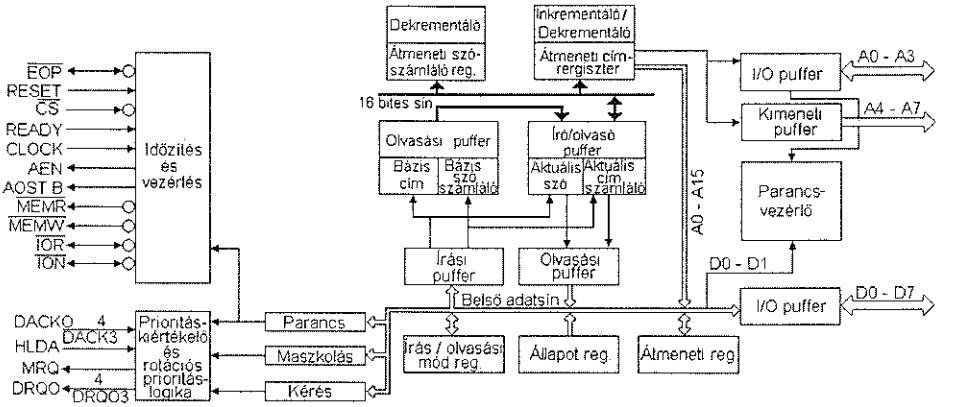
- **Címregiszter.** Mindig az átvitelben szereplő memóriarekesz címét tartalmazza, értéke az átvitel során automatikusan nő.
- **Számlálóregiszter.** Az átvitel elején az átviendő szavak számát tartalmazza, értéke az átvitel során automatikusan csökken.
- **Állapotregiszter (parancsregiszter).** Tartalma az átvitel módját és irányát határozza meg.

Intel 8237 DMA vezérlő (célprocesszor)

A perifériák DREQ0 – DREQ3 (DMA REQuest) csatornán keresztül kérhetnek közvetlen memória-hozzáférést. Az átvihető legnagyobb adatblokk mérete 64 KByte (**3.8.** ábra).

A DMA vezérlő működése

A 8237 vár, amíg valamely DREQn vonalán magas szint jelenik meg, amivel az I/O-eszköz DMA kiszolgálást kér. Ekkor 8237 1-es állapotba helyezi a HRQ (Hold Request) vonalát, ezzel jelzi a processzornak, hogy át kívánja venni a sínek vezérlését. A processzor HLDA (Hold Acknowledge) vonala jelzi a 8237-nek, hogy a μP HiZ állapotba helyezte adat- és vezérlővonalait, azaz a síneket felszabadította a DMA vezérlő részére. Ezután a 8237 a DACKn (DMA Acknowledge, DMA nyugtázás) vonalon jelzi az I/O eszköz számára, hogy megkezdődhet a DMA átvitel. Az I/O eszköz az átviteli blokk első adatát az adatsínre helyezi. A DMA vezérlő a címvonalakra teszi az aktuális memóriarekesz címét, majd kiadja a MEMW (memóriaírás) vezérlőjelet. Az I/O eszközre való közvetlen memória-hozzáféréses írás hasonló módon megy végbe.



3.8. ábra. A 8237 belső felépítése

Ellenőrző kérdések, feladatok

1. Ismertessük a sínrendszer fogalmát, használatának előnyeit!
2. Hasonlítsuk össze a külső és belső sínrendszer felépítését, jellemzőit!
3. Ismertessük a kétirányú sínmeghajtó áramkör felépítését, működését!
4. Ismertessük a sínhasználati igények elbírálásának módszereit!
5. Hasonlítsuk össze a szinkron és aszinkron sínvezérlés jellemzőit!
6. Ismertessük a fontosabb I/O adatátviteli eljárásokat!
7. Ismertessük a megszakítás-kiszolgálás lépéseit!
8. Ismertessük a 8259 megszakításvezérlő felépítését, működését!
9. Soroljuk fel a DMA adatátvitel előnyeit, lépéseit!
10. Ismertessük a 8237 DMA vezérlő működését!

4. MEMÓRIÁK, MEMÓRIASZERVEZÉS

A memóriák, ill. szélesebb körben értelmezve a tárak olyan eszközök, amelyek az információ hosszabb-rövidebb ideig tartó megőrzésére alkalmasak. Kialakulásuk, fejlődésük során különféle elnevezések voltak használatosak, de összefoglaló néven leginkább tároknak nevezzük őket.

A tárakon belül megkülönböztetünk egy olyan csoportot, amely félvezető alkatrészekből épül fel, és viszonylag nagy mennyiségű információ tárolására alkalmas. Ezt nevezzük memóriának. A szakirodalomban és a köznapi nyelvben azonban ez a két csoportnév gyakran keveredik (pl. számítógép operatív tára). Ebben a fejezetben a tárolók közül elsősorban a memóriákról lesz szó. Ezeknél az áramköröknél az információtárolás alapegysége a bit, és egy tárolóelem egy bitnyi információ tárolására alkalmas.

A tárolókkal kapcsolatban sok olyan kifejezés, fogalom használatos, amely feltétlenül szükséges működésük, használatuk megértéséhez, mint pl. a következők.

Memóriakapacitás

Az adott egységben tárolható adatmennyiséget határozza meg. Általában bájtban adjuk meg (1 bájt = 8 bit). A váltószám a bináris számrendszer miatt itt 2^{10}=1024. Így a leggyakrabban alkalmazott jelölések:

1 Kbájt = 1024 bájt

1 Mbájt = 1024×1024 bájt

1 Gbájt = 1024×1024×1024 bájt

Megadható a kapacitás a bitek számával is, de ez nagy méretű tárak esetében nem használatos.

Megadható a kapacitás a rekeszek számával is. Egy rekeszben meghatározott bitszámú információ tárolható. Ennek hosszúsága az a rendszerre jellemző érték, amit az egyszerre (párhuzamosan) tud kezelni. Az egy rekeszben tárolható információ mennyiségét 1 szónak is nevezzük.

Elérési (hozzáférési) idő

Az az időtartam, amely az adat kiolvasását, ill. beírását elindító parancs kiadásától, az adat rendelkezésre állásáig, ill. beírásáig eltelik. Ez jelenleg $n \times 10$ ns értéket jelent. (Ez a gyorsaságra jellemző érték.)

Memóriaszervezés

Azt mutatja meg, hogy a tárolt információnak mekkora egysége érhető el egy olvasási/írási ciklussal. Ezek alapján megkülönböztetünk:

- bites szervezésű (minden bitre külön-külön írható-olvasható),
- bájtos szervezésű (bájtonként írható-olvasható) és
- szavas szervezésű (szavanként írható-olvasható) memóriát.

Elérési mód

Azt mutatja meg, hogy az adathoz hogyan tudunk hozzáférni, milyen módszerekkel, milyen úton tudjuk elérni a szükséges információt.

4.1. A tárolók típusai, csoportosításuk

A tárak fejlődése során különböző fizikai elveken működő eszközöket fejlesztettek ki a mindenkori technológiai módszerek segítségével, így sokfajta csoportosítási mód van használatban.

Csoportosítás a fizikai működési elv szerint

Az egyes tárolótípusoknak más-más a működési elvük. Ezek közül a legfontosabbak a következők.

Mágneses elven működő tárak

Az információ (bitek) rögzítése a hordozóanyag mágnesezhetőségén alapul. Ilyenek a ferritgyűrűs tárak, a mágnesszalagos tárak, a **winchesterek** és a **floppy lemezek.** Nagy előnyük, hogy az információt sokáig megőrzik, viszont a mágneses hatásoktól óvni kell őket. A mágneses elven működő tárakat általában nagy kapacitású háttértárakként használjuk. Elérési idejük nagy.

Félvezetőkből felépülő tárak

Az információ rögzítése ún. elemi tárolócellákban (flipflopokban) történik. Egy cella egy bitnyi információ rögzítésére alkalmas. Gyártástechnológiájuk alapján további két fajtájuk létezik: a bipoláris és a MOS technológiával készített. Tulajdonképpen az ezekből felépülő, nagyobb kapacitású tárakat nevezzük memóriáknak. Előnyük a nagy sebesség és a bővíthetőség.

Optikai elven működő tárak

Az információ eltárolása és kiolvasása lézersugárral történik. Íráskor az ép felületen kis méretű lyukakat alakítanak ki, spirális pálya mentén. Olvasáskor a visszaverődő fény a kialakított bemélyedések (pit) miatt más-más fázisban (időkésleltetéssel) verődik vissza, és ebből lehet következtetni a tárolt információra.

Csoportosítás az adathozzáférés módja szerint

Soros (szekvenciális elérésű)

Az adatok tárolása sorosan – fizikailag egymás után – történik az adathordozó mentén, pl. egy mágnesszalagon. Így a kiolvasás is csak sorosan történhet, vagyis a kiadott cím alapján elindul és attól függően, hogy éppen hol állt, hosszabb vagy rövidebb ideig tart a keresés. Ezeknél a táraknál a hozzáférési idő nagyon változó. Ilyenek pl. a **mágnesszalagos tárak** és a **lyukszalagok** is.

Tetszőleges (véletlen) elérésű

Legfontosabb jellemzője az, hogy bármelyik adathoz a kiadott cím alapján, azonos idő alatt lehet hozzáférni. Előnye a gyorsaság. Ilyeneknek tekinthetők a **mágneslemezes**, a **félvezetős** és az **optikai** tárak is.

Asszociatív elérésű

Tartalom szerinti címzésűnek is nevezzük, mert az adatok címzése az adat egy részének megadásával történik.

Csoportosítás az információ beírhatósága szerint

A beírhatóság szerint két fő csoportot különböztetünk meg.

ROM (Read Only Memory)

Csak kiolvasható memóriának nevezzük. Tartalma egyszer rögzítésre kerül, és többé nem, vagy csak speciális eszközökkel módosítható.

RAM (Random Access Memory)

Tetszőleges elérésű és változtatható tartalmú memória. Tartalma akárhányszor átírható, változtatható.

Mindkét tárolófajtának léteznek további típusai is, amelyeket később tárgyalunk.

Csoportosítás a tárolás időbeli módja szerint

Ez alapján két típust különböztetünk meg.

Statikus memória

A benne tárolt információt a tápfeszültség megszüntetéséig megőrzi. Viszonylag gyors működésű, SRAM-nak is nevezzük.

Dinamikus memória

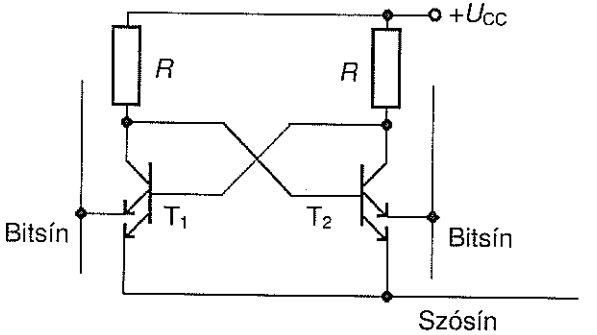
A benne tárolt információt időnként frissíteni kell, egyébként a tartalma véglegesen elveszne. Ezek lassúbb működésűek, DRAM-oknak is nevezzük.

4.2. A memóriacellák felépítése és működése

A félvezetős tárak alkotóelemei a memóriacellák. Ezek 1 bit információ tárolására alkalmasak, tárolásmódjuk lehet statikus vagy dinamikus.

4.2.1. Statikus memóriacellák

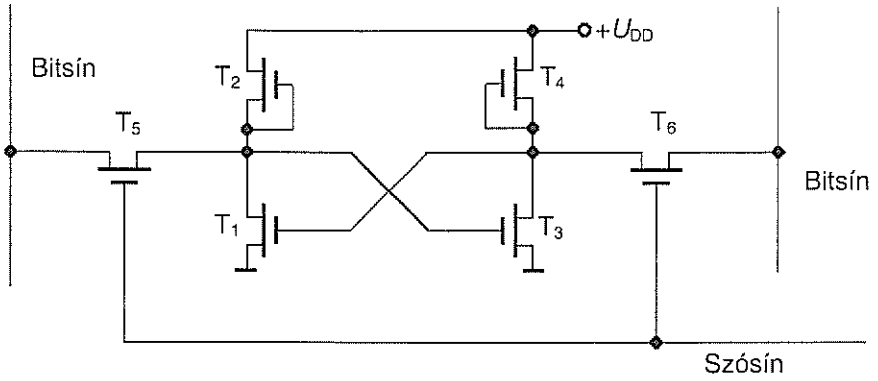
Az információt a tápfeszültség kikapcsolásáig megőrzik. A bipoláris technológiával előállított memóriacella egy változata a **4.1.** ábrán látható.



4.1. ábra. Bipoláris technológiával előállított memóriacella

Ez az áramkör nem más, mint az elektronikában megismert bistabil multivibrátor multiemitteres tranzisztorokkal megvalósítva. Az ilyen tranzisztorokra azért van szükség, mert ezzel oldható meg az információ be- és kiviteléhez szükséges kapuzás. A gyakorlatban a bipoláris memóriacella ma már kevésbé használatos, mint a MOS változata.

A MOS technológiával előállított áramkör egyszerűsített vázlata a **4.2.** ábrán látható. A gyakorlatban leginkább ez a hat MOS tranzisztorból álló változat terjedt el.



4.2. ábra. MOS technológiával előállított memóriacella

Ez az áramkör szintén egy bistabil multivibrátor. A T_1, T_3 alkotja a multivibrátort, a T_2, T_4 aktív munkaellenállások, a T_5 és T_6 által történik a kapuzás.

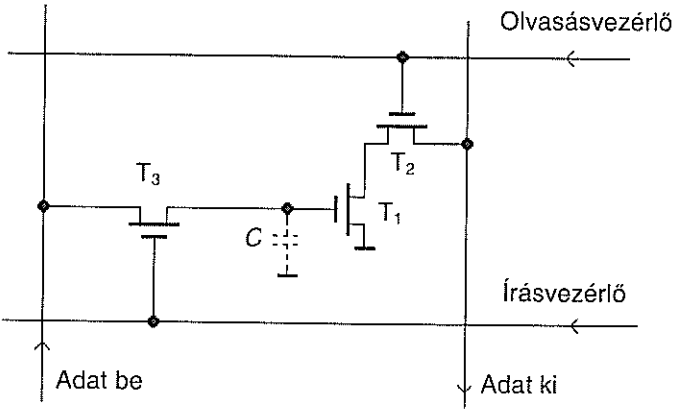
Olvasáskor a szósínre adott jel kinyitja a T_5, T_6 tranzisztorokat, és a T_1, T_3 által tárolt adatok a bitsínre kerülnek. Íráskor a szósínre adott jel hatására szintén kinyit a T_5 és T_6 tranzisztor, és a bitsínen érkező adat kerül beírásra. A pontos működéshez ezeket a tárolócellákat ki kell egészíteni ún. író/olvasó áramkörrel is.

4.2.2. Dinamikus memóriacellák

A bennük tárolt információt csak rövid ideig őrzik meg. MOS tranzisztorokból épülnek fel és a MOS tranzisztorok saját kapacitását használják fel tárolásra. Ennél a memóriacellánál feltétlenül szükséges a C töltésének szinten tartása (frissítése). Felépítése a **4.3.** ábrán látható.

A T_1 tranzisztor a töltéstároló alkatrész. Beíráskor az Írásvezérlő vonalra adott jellel kinyitjuk a T_3-at, és az Adat be vonalról beíródik az információ a T_1-be.

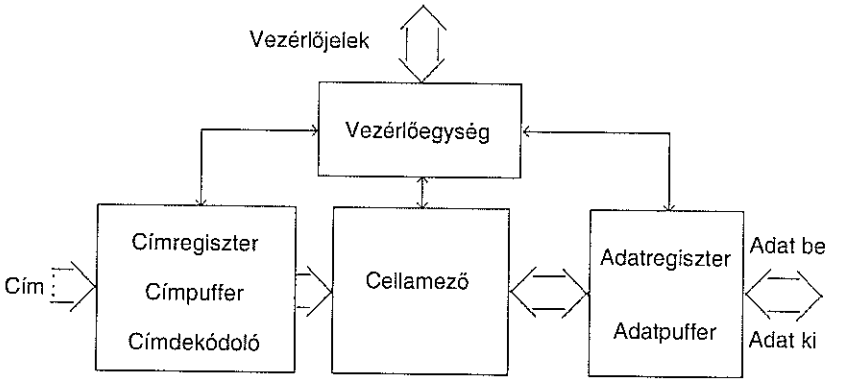
Kiolvasáskor az Olvasásvezérlőre adott jellel kinyitjuk a T_2-t, és az Adat ki vezetéken megjelenik az információ. A cellák tartalmának szintentartását a T_3-on keresztül frissítő áramkör végzi.



4.3. ábra. A dinamikus memóriacella felépítése

4.3. A memóriamodulok felépítése, működése

A memóriaegységek legfontosabb része a cellamező. Ez az előzőekben tárgyalt elemi cellákból épül fel. Itt történik az információ tárolása. Ezt az egységet különféle áramkörökkel kell kiegészíteni, amelyek segítségével történik a címzés, a kiolvasás és a beírás (**4.4.** ábra).



4.4. ábra. Egy memóriamodul elvi felépítése

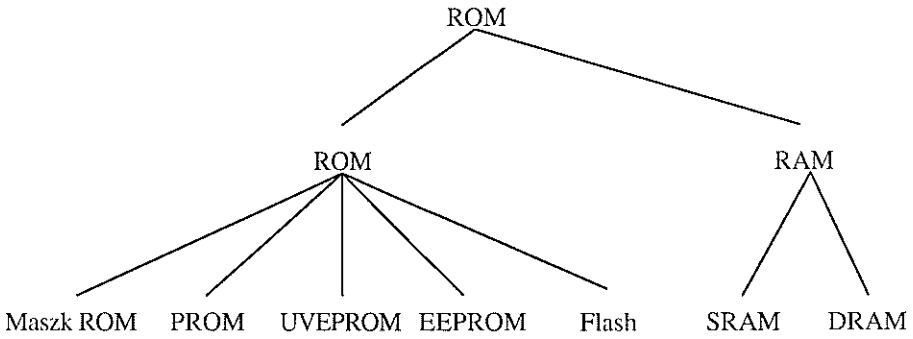
A kiadott cím a címsínről a címregiszterbe kerül. A címdekódoló ez alapján kijelöli a konkrét memóriarekeszt. (A címdekódoló használatával lehetőség van a címvezetékek számának csökkentésére.) A kijelölt rész tartalma bekerül az adatregiszterbe. Ez RAM esetén kétirányú (Be/Ki), ROM esetén pedig egyirányú. A vezérlő-

áramkör a különböző helyekről jövő vezérlőjeleket fogadja és ezek alapján kapuzási, sorba állítási, engedélyezési és egyéb feladatokat lát el.

Ezek közül a legfontosabbak:

- engedélyezés (ME),
- chip-kiválasztás (CS),
- írásengedélyezés (WR),
- olvasásengedélyezés (R vagy RD),
- külső frissítés (RFRSH).

A félvezető memóriamodulok leggyakoribb fajtái a **4.5.** ábrán láthatók.



4.5. ábra. A félvezető memóriák fajtái

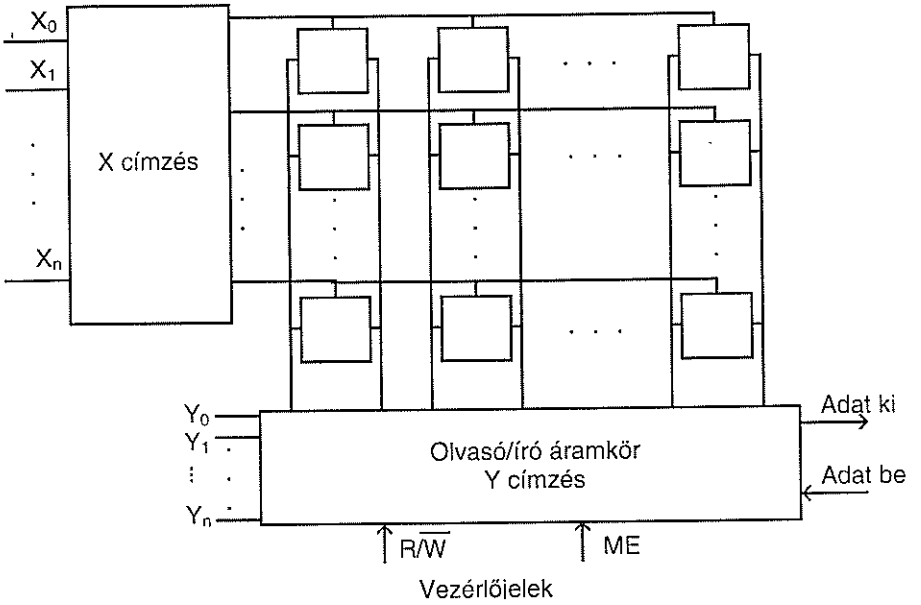
4.3.1. A memóriacellák szervezése

A cellamező kiépítésére többfajta módszer is használatos. Ezek közül a leggyakrabban a bites szervezésű és a szavas szervezésű memóriákat alkalmazzák.

Bites szervezésű memóriák

Ennél a módszernél a tárolócellák egyenként (bitenként) jelölhetők ki. Ez úgy lehetséges, hogy a cellák egy kétdimenziós kiépítés szerint helyezkednek el. Így egy cella kijelölése a megfelelő X és Y vezetékek aktivizálásával érhető el.

Beíráskor az adatbemenetre adott jel erre a cellára lesz hatásos, kiolvasáskor pedig az ebben a cellában lévő információ kerül az adatkimenetre. Vázlatos felépítése a **4.6.** ábrán látható.

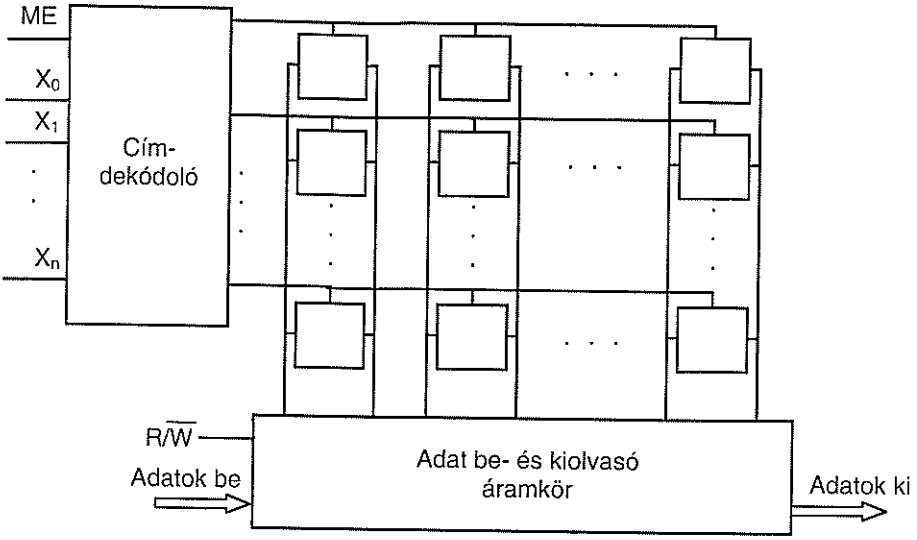


4.6. ábra. Bites szervezésű memória felépítése

Ezzel a módszerrel szavak tárolása és kiolvasása is lehetséges úgy, hogy annyi ilyen síkbeli egységet helyezünk a térben egymás mögé, ahány bites egy-egy szó. Így egy-egy memóriasíkon minden szónak ugyanolyan helyiértékű bitje található. Az adat beírása és kiolvasása az oszlopkijelölést végző áramkörhöz kapcsolódó demultiplexer (beírás) és multiplexer (kiolvasás) áramkörökkel megvalósítható. Ezt nevezzük a módosított bit szervezésű memóriának.

Szavas szervezésű memóriák

A tárolócellák ennél a módszernél is kétdimenziós kiépítés szerint helyezkednek el. A címzés során egy teljes mátrixsor kijelölése történik meg. Egy-egy sor alkot egy szót, ennek bitjeit egyszerre olvassuk ki (ROM, RAM), ill. egyszerre írjuk be (RAM). Vázlatos felépítése a **4.7.** ábrán látható.



4.7. ábra. Szavas szervezésű memória felépítése

A memória működése az ME címdekódoló engedélyezése, ill. az $R/\overline{W}$ adatbeírás/kiolvasás vezérlőjelekkel aktivizálódik.

4.3.2. ROM áramkörök

Az információt vagy az áramkör gyártásakor viszik be a ROM-ba, vagy egyes típusoknál (megfelelő eszközökkel és módszerekkel) a felhasználó írhatja be (programozhatja, ill. újra programozhatja a ROM-ot). A ROM előnye, hogy a tápfeszültség kikapcsolásával az információ nem vész el. Felépítése az előző részben tárgyaltak szerinti, és kisebb a felületigénye, mint a RAM áramköröknek, ezért egy tokban nagyobb kapacitású tároló hozható létre.

A ROM áramkörök leggyakoribb típusai a következők.

Maszk-programozott ROM. Ezeket egyszer lehet programozni, és ez a gyártás során történik. Ennek megfelelően csak speciális célokra használható (pl. kalkulátorokban, mikroszámítógépek rendszerprogramjának tárolására stb).

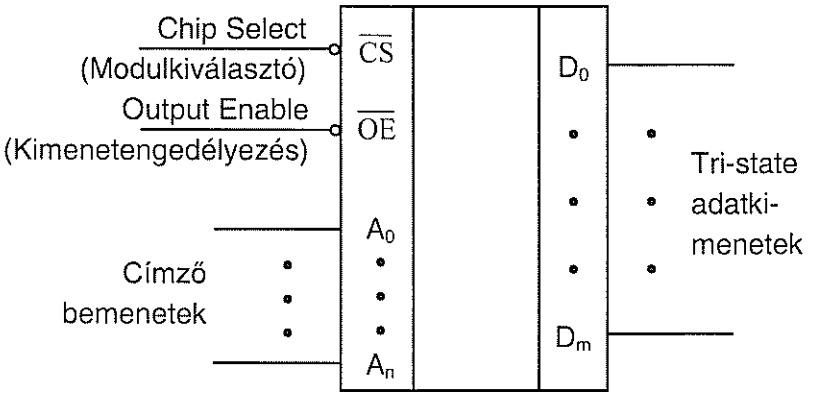
PROM (Programmable ROM). Speciális készülékekkel (ún. beégető áramkör) a felhasználó is tudja programozni, de ez a folyamat csak egyszer végezhető el.

UVEPROM (Ultraviolet Erasable PROM). Tartalmuk ultraibolya fénnyel törölhető, így újra írhatók. Ezeket MOS technológiával gyártják.

EEPROM (Electrically Erasable PROM). Elektromosan törölhető PROM. Ez már nagyon hasonlít a RAM áramkörökre, de a beírás csak korlátozott számban ismételhető meg. Az EEPROM-ok írása lassú, körülményes és speciális áramköröket igényel.

Flash-ROM. Elektromosan törölhető és programozható áramkörök, tartalmukat a tápfeszültség kikapcsolása után is megtartják. Inkább hasonlítanak a RAM-okra, mint az EEPROM-ok. A beírás időtartama jóval nagyobb, mint az olvasásé. A cellák írása előtt az áramkört törölni kell.

Egy tipikus ROM modul lábkiosztása a **4.8.** ábrán látható.



4.8. ábra. Egy tipikus ROM modul lábkiosztása

Az egyes kivezetések feladata a következő.

A_0–A_n: meghatározott számú címző bemenet;

D_0–D_m: meghatározott számú, tri-state felépítésű adatkimenet, így szükség esetén lekapcsolható az adatbuszról;

CS: 0-ra aktív tokengedélyező jel, több tok esetén a kívánt kiválasztásra használható;

OE: 0-ra aktív kimenet-engedélyező jel.

4.3.3. RAM áramkörök

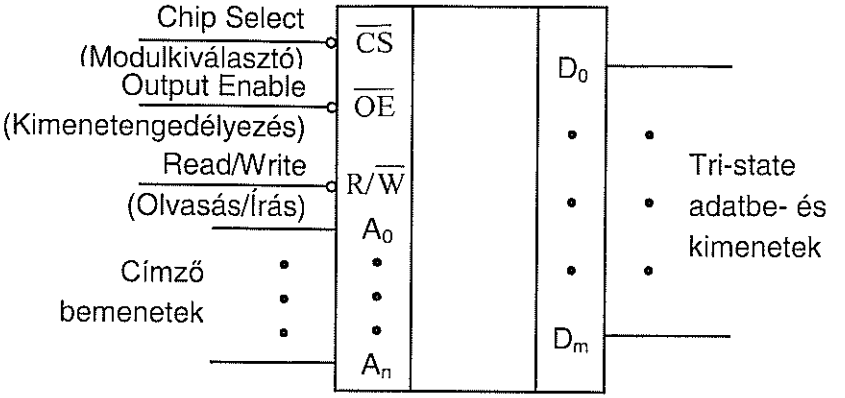
A RAM áramkörök véletlen hozzáférésű, írható-olvasható memóriák. Az adatok kiolvasásával a tárolt információ nem semmisül meg, de a cellák a tápfeszültség kikapcsolása után elvesztik tartalmukat. Az adatok beírása és kiolvasása akárhányszor megismételhető.

Statikus és dinamikus RAM áramköröket különböztetünk meg.

Statikus RAM (SRAM) áramkörök

A tároló cellái flipflopok (bistabil multivibrátorok). Viszonylag sok alkatrészből állnak, ezért nagy a helyigényük. Gyors működésűek.

Egy tipikus SRAM modul lábkiosztása a **4.9.** ábrán látható.



4.9. ábra. Egy tipikus SRAM modul lábkiosztása

Az egyes kivezetések feladata a következő.

$\overline{CS}$: a tok engedélyező jele, 0-ra aktív;

$\overline{OE}$: a kimenet engedélyező jele, 0-ra aktív;

$R/\overline{W}$: olvasás vagy írás engedélyező jele;

A_0–A_n: címző bemenetek;

D_0–D_m: adat ki- és bemenetek.

Az SRAM-k MOS vagy bipoláris technológiával gyárthatók. A bipoláris memóriák működése gyorsabb, a MOS technológiával készítetteknek pedig kisebb a helyigényük (így nagyobb a tok tárolókapacitása).

Kialakítása lehet bites vagy szavas szervezésű.

Jellemzői

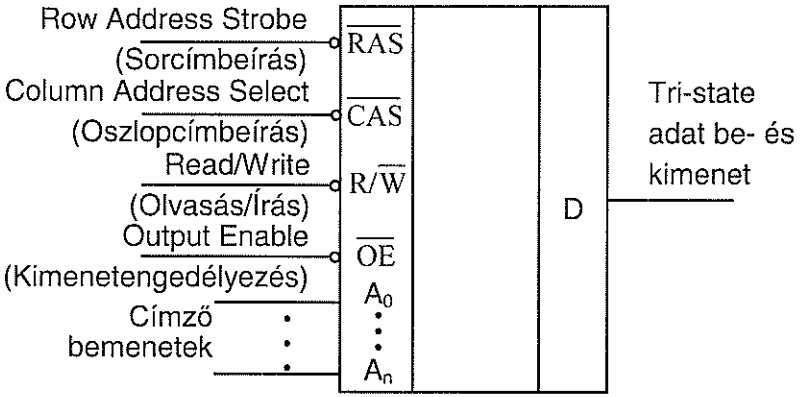
- kapacitása: 4 K×1 bit, 16 K×1 bit, 8 K×8 bit;
- technológia: TTL, NMOS, CMOS;
- elérési idő: 10–100 ns;
- fogyasztás: n×100 mW.

Dinamikus RAM (DRAM) áramkörök

Tároló cellái kondenzátorként működő MOS tranzisztorok, az információ a tárolt töltés formájában van jelen.

Ez a kondenzátor viszonylag rövid idő alatt kisül, ezért a töltés utánpótlására van szükség. Ezt a **frissítő áramkörök** valósítják meg. Az ilyen memóriák felépítése sokkal egyszerűbb, mint az SRAM-oké, ezért kisebb a helyigényük is. Elsősorban nagy kapacitású memóriák kiépítésére használják.

Egy DRAM modul lábkiosztása a **4.10.** ábrán látható.



4.10. ábra. Egy tipikus DRAM lábkiosztása

Az egyes kivezetések feladata a következő.

$\overline{RAS}$: egy órajelgenerátort vezérlő jel;

$\overline{CAS}$: a cellák kijelölését és kiolvasását ütemező 2. órajelgenerátor vezérlőjele;

$R/\overline{W}$: olvasás vagy írás engedélyezőjele;

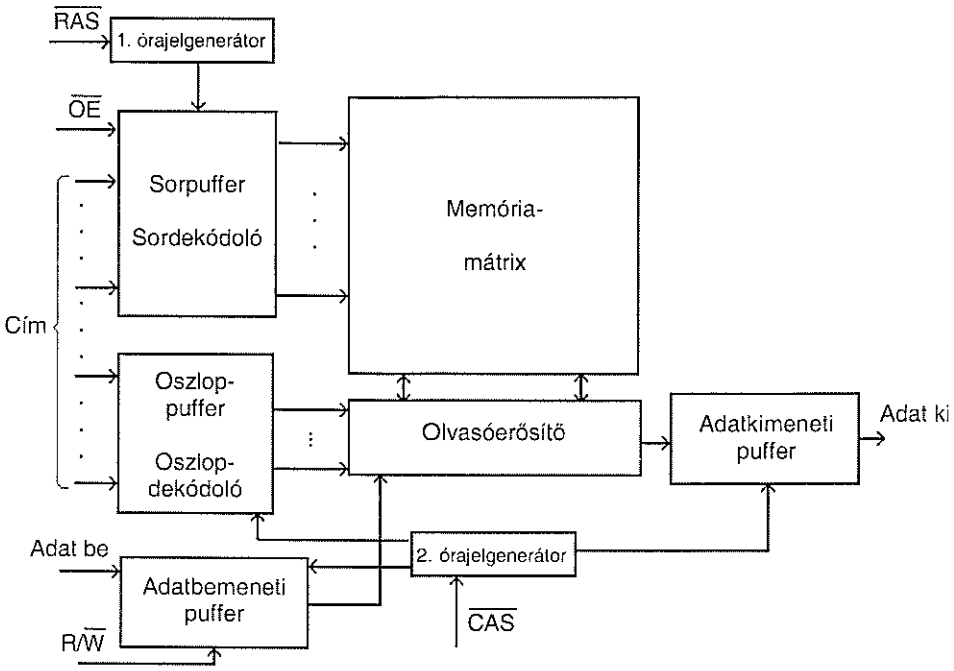
$\overline{OE}$: kimenet engedélyezése;

A_0–A_n: címbemenetek;

D: adatbemenet, ill. kimenet.

Frissítés: a CAS és a RAS órajelekkel válik lehetővé a megcímzett sor felfrissítése. Ez kb. 2 ms-onként aktuális, a frissítés ideje alatt nincs hozzáférési lehetőség.

Egy DRAM részletes felépítése a **4.11.** ábrán látható.



4.11. ábra. Egy DRAM felépítése

Kialakítása többnyire bites szervezésű.

Néhány példa a legfontosabb paramétereire:

- kapacitás: 16 K×1 bit, 1 M×1 bit, 32 K×8 bit;
- technológia: NMOS, CMOS;
- elérési idő: 70–150 ns;
- fogyasztás: n×100 nW.

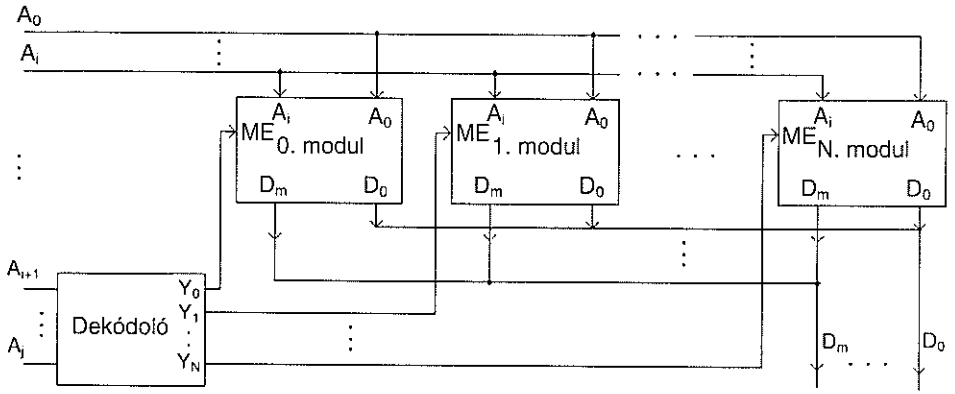
A jelenleg gyártott DRAM-ok már belső frissítőáramkörökkel rendelkeznek, ezért kifelé látszólag úgy működnek, mint az SRAM-ok.

4.4. Memóriamodulok összekapcsolása

Az adott célnak megfelelő kapacitású és szóhosszúságú memóriát az előzőekben tárgyalt memóriamodulok összekapcsolásával lehet kialakítani. Az összekapcsolás során a kapacitást vagy a szóhosszt bővíthetjük.

4.4.1. A kapacitás bővítése

A kapacitás bővítésének oka az, hogy egy modul tárolóképessége az adott feladat ellátására kevés. A kapacitás növelésének érdekében több modult kell összekapcsolni úgy, hogy adott az alkalmazandó modulok szóhossza (pl. 8 bit) és címvezetékeinek száma (pl. 10 bit). Adott a rendszer címvezetékeinek a száma is (pl. 13 bit). Ez szabja meg a kapacitás bővítésének felső határát. Ezek ismeretében meghatározható, hogy a szükséges memóriakapacitás kiépítéséhez hány címvezeték és hány modul szükséges. Ezek ismeretében a modulok adatvezetékeit helyiértékhelyesen párhuzamosítjuk, ezek lesznek az adatvezetékek. Majd az egyes modulokhoz bekötjük – szintén helyiérték-helyesen párhuzamosítva – a rendszer címvezetékeinek egy részét (A_0–A_i-ig). A rendszer fennmaradó címvezetékeit egy dekódoló címző bemeneteire vezetjük, majd annak kimeneteit kötjük rá az egyes modulok tokengedélyező vezetékeire. A bővítés vázlatos rajza a **4.12.** ábrán látható.



4.12. ábra. A memória kapacitásának bővítése

Az egyes jelek és vonalak feladata a következő.

A_0–A_i: az egyes modulok címvezetékeinek száma (pl. 10, ha a modul kapacitása 2^{10} = 1 K);

A_0–A_j: a rendszer szükséges címvezetékeinek száma (pl. 13, ha 2^{13}=8 K memóriát szeretnénk kiépíteni);

D_0–D_n: az adatvezetékek száma (pl. 8);

Y_0–Y_N: a dekódoló kivezetései (pl. 8, mert ennyi modulra van szükség);

ME: a tokok engedélyező bemenete.

A kapcsolás működése: A példa adatai alapján a rendszertől jövő címvezetékek (A_0–A_{12}) alsó 10 bitjét (A_0–A_9), mindig mindegyik modul megkapja. Az A_{10}, A_{11}, A_{12} címvezetékeken lévő információ hatására a dekódoló a 8 modul közül mindig csak egyet engedélyez. Ugyanekkor a többi le van tiltva, és a kiadott cím nem érvényes rá.

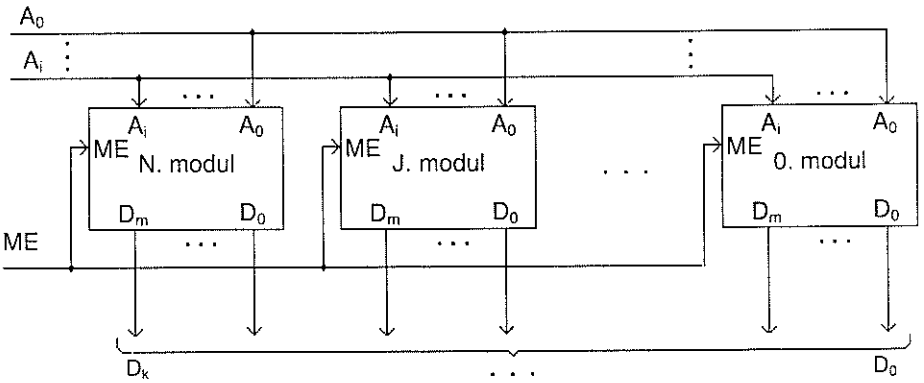
4.4.2. A szóhossz bővítése

Erre a módszerre akkor van szükség, ha a memóriában tárolható szavak száma megfelelő, de a hosszuk a szükségesnél rövidebb. Pl. ha a szóhosszt szeretnénk a kétszeresére növelni, akkor minden modult meg kell duplázni. Adott egy modul címvezetékeinek és adatvezetékeinek a száma, és adott a rendszer címvezetékeinek és adatvezetékeinek a száma is.

Pl. a rendszer és a modul címvezetékeinek a száma egyaránt 16, az adatvezetékek száma pedig a moduloknál 8, a rendszernél pedig 32, akkor látható, hogy egy modul 64 Kszót képes tárolni és egy szó 8 bites, a rendszer szintén 64 Kszót képes kezelni, de egy szó 32 bites. Tehát négy modult kell összekapcsolni, hogy a megfelelő szóhossz elérjük.

A bővítéshez a rendszer címvezetékeit összekötjük mindegyik modul címvezetékeivel (A_0–A_i párhuzamosítása), a modulok adatvezetékeinek összessége (D_0–D_k) adja a rendszer adatvezetékeinek számát.

A bővítés vázlatos rajza a **4.13.** ábrán látható.



4.13. ábra. A memória szóhosszának bővítése

Az egyes jelek és vonalak feladata a következő.

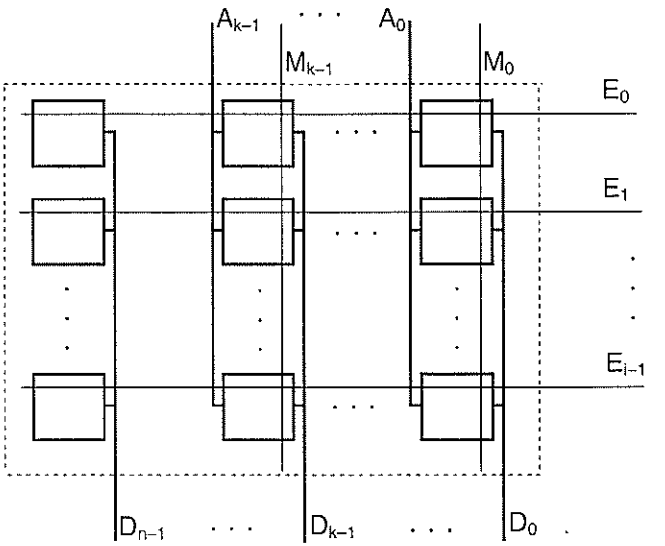
A_0–A_i: az egyes modulok és a rendszer címvezetékei;
D_0–D_m: az egyes modulok adatvezetékei;
D_0–D_k : a rendszer adatvezetékei;
ME: a tok engedélyező bemenete.

A kapcsolás működése: A rendszer által kiadott cím minden bitjét mindegyik modul megkapja (A_0–A_i). Mindegyik tok egyszerre kapja meg az engedélyező jelet (ME) is. Az egyes modulok a rendszer által meghatározott szóhossz (D_0–D_k) egy-egy adott hosszúságú darabját (D_0–D_m) tárolják.

4.5. Asszociatív memóriák

Ezeket tartalom szerint elérhető memóriáknak vagy CAM-nak (Content Addressable Memory) is nevezzük. Az eddig tárgyalt memóriákhoz képest összehasonlító áramköröket (komparátorokat) is tartalmaznak. Ezeket összeépíthetik a tárolócellákkal is. Az ilyen fajta memóriáknak az az előnyük, hogy a szükséges adatokat gyorsan el tudjuk érni. Ez úgy lehetséges, hogy a cím feladatát tulajdonképpen a tárolt adat egy része (pl. bájtos szervezés esetén annak alsó 4 bitje) veszi át. Ezek után keressük azt a bájtot, amelynek az alsó 4 bitje adott, ha ez megvan akkor azt kiolvassuk.

Az ilyen memóriák bonyolult áramköri felépítésűek, sok összehasonlító áramkört tartalmaznak, ezért csak viszonylag kis kapacitású egységeket készítenek belőlük. Pl. ilyen a cache, amelyről a következő fejezetben lesz szó részletesen. Az asszociatív memória vázlatos felépítése a **4.14.** ábrán látható.



4.14. ábra. Egy asszociatív tár felépítése

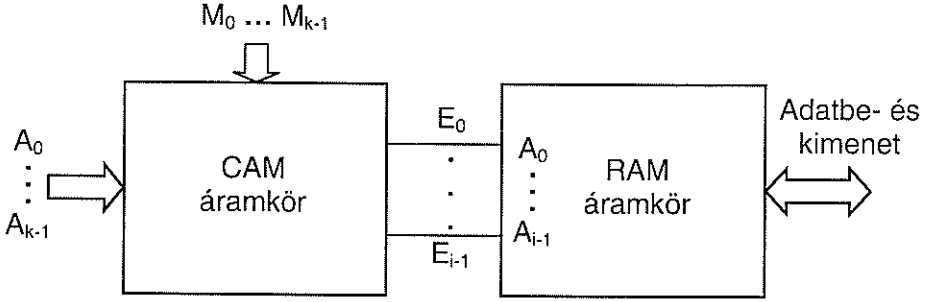
Az egyes vonalak feladata a következő.

D_0–D_{n-1}: adat be- és kimenetek;
A_0–A_{k-1}: az adat alsó négy helyiértéken lévő bitje, amely itt a címzésre való;
E_0–E_{i-1}: az egyezés kimenetek;
M_0–M_{k-1}: a maszkoló bemenetek.

Az asszociatív tár működése: A rendszer az A_0–A_{k-1} vezetékeken kiadja a címet, ami valójában a keresett adat egy része. Az összehasonlító áramkörök megnézik, hogy melyik sorban van egyezés, és ahol ez fennáll, annak a sornak az összes bitje (D_0–D_{n-1}) kiolvasásra, ill. beírásra kerül. Ennek a sornak az egyezés kimenetén (E_i) logikai-1 szint jelenik meg, amely még egyéb felhasználási lehetőséget is biztosít számunkra. A maszkoló bemenetekre adott jelekkel (0, 1) lehetőség van arra, hogy csak meghatározott helyiértékű biteken történjen meg az összehasonlítás. Ezzel a felhasználási lehetőségek köre tovább bővül.

Ezek a CAM áramkörök többféle célra is felhasználhatók. Egy gyakori felhasználási lehetőség az, amikor a CAM áramkörrel azt döntjük el, hogy egy valamilyen szempontból kigyűjtött adatcsoportban megtalálható-e a címzett adat.

Ennek elvi felépítése a **4.15.** ábrán látható.



4.15. ábra. Asszociatív tár alkalmazása

Az egyes vonalak feladata a következő.
A_0–A_{k-1}: a CAM áramkör címző bemenetei;
M_0–M_{k-1}: a CAM áramkör maszkoló bemenetei;
E_0–E_{i-1}: a CAM áramkör egyezés kimenetei;
A_0–A_{i-1}: a RAM áramkör címző bemenetei.

Az áramkör működési elve: A CAM áramkörben a RAM-ban bent lévő adatok címeinek egy részlete található. A CAM áramkör a rendszer által kiadott címrész alapján gyorsan eldönti, hogy bent van-e a kért adat a memóriában (valamelyik E vezetéken logikai-1 szint jelenik meg). Ha igen, akkor az annak megfelelő sor tartalmát kiolvassa, ill. abba a sorba beír.