

ESZR

Herboly Miklós

Bevezetés a nagyobb ESZR modellek architektúrájába



Bevezetés a nagyobb ESZR modellek architektúrájába

Introduction to the architecture of larger UCS models

Miklós Herboly

**Introduction to the
architecture of
larger UCS models**

Central Statistical Office
International Computer Education and Information Centre
Budapest, 1980

Herboly Miklós

**Bevezetés
a nagyobb ESZR modellek
architektúrájába**

Központi Statisztikai Hivatal
Nemzetközi Számítástechnikai Oktató és Tájékoztató Központ
Budapest, 1980

ESZR KÖNYVEK

Lektorálta:

TOMKA ERZSÉBET
KISS ISTVÁN

Supervised by

ERZSÉBET TOMKA
ISTVÁN KISS

Második kiadás

Second edition

© Herboly Miklós, 1977

ISBN 963 7552 30 8

Kiadja: KSH Nemzetközi Számítástechnikai Oktató és Tájékoztató Központ

Felelős kiadó: Faragó Sándor igazgató

Készült a SZÁMOK Reprográfiában

A nyomdai megrendelés törzsszáma: 80/162

Budapest, 1980

Felelős szerkesztő: Jakab Ágnes

Megjelent 18 (A/5) ív terjedelemben

Printed in Hungary

Tartalom

ELŐSZÓ - - - - -	9
1. A SZÁMÍTÁSTECHNIKA TÖRTÉNETE - - - - -	11
1.1 A számítástechnika kialakulása - - - - -	11
1.2 Analóg és digitális számítógépek - - - - -	15
1.3 Az ESZR számítógépcsalád - - - - -	16
2. ARITMETIKAI ÉS LOGIKAI MŰVELETEK - - - - -	23
2.1 A kettes, tizes és a tizenhatos számrendszer - - - - -	23
2.2 Számkonverziók - - - - -	27
2.3 Műveletek. Műveletvégzés a számítógépben - - - - -	27
2.3.1 Összeadás kettes és tizenhatos számrendszerben - - - - -	32
2.3.2 Kivonás kettes és tizenhatos számrendszerben - - - - -	33
2.3.3 Komplementálás. Negatív számok ábrázolása - - - - -	34
2.3.4 BCD összeadás - - - - -	38
2.3.5 Logikai műveletek - - - - -	39
3. ADATKÖZLÉS A SZÁMÍTÓGÉPPEL - - - - -	45
3.1 Információk és adatok - - - - -	45
3.2 Adathordozók és kódrendszerek - - - - -	47
3.2.1 Lyukkártyás adatrögzítés - - - - -	47
3.2.2 Lyukszalagos adatrögzítés - - - - -	49
3.2.3 Mágnesszalagos adatrögzítés - - - - -	50
3.2.4 Adatrögzítés mágneslemezen - - - - -	52

4. ESZR PERIFÉRIÁLIS EGYSÉGEK - - - - -	57
4.1 A perifériák és a számítógép - - - - -	57
4.2 Perifériatípusok - - - - -	58
4.2.1 Lyukkártyaolvasó (ESZ-6012) - - - - -	59
4.2.2 Lyukkártyalyukasztó (ESZ-7010) - - - - -	61
4.2.3 Lyukszalagolvasó (ESZ-6022) - - - - -	63
4.2.4 Lyukszalaglyukasztó (ESZ-7022) - - - - -	65
4.2.5 Sornyomtató (ESZ-7032) - - - - -	66
4.2.6 Konzolirőgép (ESZ-7070) - - - - -	70
4.2.7 Mágnesszalagegység (ESZ-5511) - - - - -	71
4.2.8 Mágneslemezegység (ESZ-5052) - - - - -	81
5. A CSATORNARENDSZER - - - - -	97
5.1 Input-output műveletek - - - - -	97
5.2 A csatornák fajtái - - - - -	98
5.2.1 A csatornarendszer üzemmódjai - - - - -	99
5.3 Az input-output rendszer címzése - - - - -	100
5.4 Input-output műveletek - - - - -	105
5.4.1 Input-output utasítások - - - - -	106
5.4.2 Csatornaparancsok, csatornaprogram - - - - -	107
5.4.3 Vezérlőjelek - - - - -	109
5.5 Az input-output rendszer állapotának nyilvántartása - - - - -	110
6. AZ OPERATÍV TÁR - - - - -	113
6.1 Az operatív tár szerepe a számítógépben - - - - -	113
6.1.1 A tárolás elve - - - - -	114
6.1.2 Az operatív tár működése - - - - -	114
6.2 Standard tárrekeszek - - - - -	120
6.3 A főmemória információvédelme - - - - -	122
6.4 Az operatív tár fő funkcionális részei - - - - -	125
7. A KÖZPONTI EGYSÉG (PROCESSZOR)- - - - -	127
7.1 A processzor feladata - - - - -	127
7.1.1 A program állapotának nyilvántartása - - - - -	127
7.2 A programmegszakítási rendszer - - - - -	129

7.3	Gépi utasításformátumok - - - - -	133
7.3.1	Operandusformátumok - - - - -	139
7.4	A processzor felépítése - - - - -	146
8.	A FOLYAMATÁBRA-KÉSZÍTÉS ALAPJAI - - - - -	149
8.1	A folyamatábra mint eszköz - - - - -	149
8.2	A folyamatábrák fajtái - - - - -	151
8.2.1	A feladat felbontása - - - - -	152
8.2.2	Ciklusszervezés - - - - -	156
9.	BEVEZETÉS AZ ESZR ALAPSOFTWARE-BE - - - - -	163
9.1	Software - - - - -	163
9.1.1	Az alapsoftware kialakulása - - - - -	163
9.1.2	Operációs rendszerek - - - - -	166
9.2	ESZR operációs rendszerek - - - - -	171
9.2.1	Az operációs rendszer komponensei - - - - -	171
9.2.2	A program futtatása DOS alatt - - - - -	175
10.	FILE-SZERVEZÉSI ELJÁRÁSOK - - - - -	179
10.1	A szervezés kiinduló adatai - - - - -	179
10.1.1	A file egyes jellemzői - - - - -	179
10.1.2	A feldolgozás egyes jellemzői - - - - -	180
10.1.3	File-szervezési módszerek - - - - -	180
10.1.4	Az operációs rendszer funkciói - - - - -	182
10.2	A szekvenciális file felépítése - - - - -	182
10.3	Az indexelt szekvenciális file felépítése - - - - -	184
10.3.1	Indexek - - - - -	184
10.3.2	Tulcsordulási területek - - - - -	187
10.3.3	File-bővítés - - - - -	189
10.3.4	A file lemezterület-igényének becslése - - - - -	193
10.4	A direkt file felépítése - - - - -	194
10.4.1	Címzés - - - - -	194
10.4.2	Címkonverziós eljárások - - - - -	197
10.4.3	A file létrehozása - - - - -	200

Contents

1. THE HISTORY OF THE COMPUTING TECHNIQUE	11
2. ARITHMETIC AND LOGIC OPERATIONS	23
3. DATA COMMUNICATION WITH THE COMPUTER	45
4. PERIPHERALS OF THE UCS MACHINES	57
5. CHANNEL SYSTEM	97
6. WORKING MEMORY	113
7. CENTRAL PROCESSING UNIT (PROCESSOR)	127
8. FUNDAMENTALS OF FLOWCHARTING	149
9. INTRODUCTION TO THE BASIC SOFTWARE OF THE UCS	163
10. FILE ORGANIZATION METHODS	179

Előszó

Ez a könyv elsősorban azok számára készült, akik valamilyen szervezett tanfolyam keretén belül kívánják elsajátítani a nagyobb ESZR modellek programozásához szükséges ismereteket. Ahhoz, hogy valaki jól képzett programozó legyen, elengedhetetlen a használt számítógéprendszer főbb hardware-funkcióinak áttekintése.

Az ismeretközlés során semmilyen számítástechnikai előképzettséget nem feltételeztünk; sokkal inkább azon igyekeztünk, hogy a kezdő programozók a programozási ismeretek befogadására minél előbb alkalmassá váljanak. Mindemellett reméljük, hogy gyakorlott programozók is eredményesen használhatják a jegyzetet, amelynek felépítése maximálisan támogatja az önálló felkészülést is.

Ezuton is kifejezem hálás köszönetemet a software és hardware lektor-nak, akik alkotó észrevételeikkel nagyban hozzájárultak ahhoz, hogy ez a jegyzet most a felhasználók kezébe kerülhet.

Budapest, 1977

A szerző



1. A számítástechnika története

1.1 A SZÁMÍTÁSTECHNIKA KIALAKULÁSA

Az ember - természeténél fogva - már az ősidők óta állandóan azon fáradozott, hogy munkáját - a természet átalakítását - mind könnyebbé és egyszerűbbé tegye. Ezért fejlesztette ki az idők során különböző munkaeszközöket, amelyek kezdetben fizikai munkáját, majd később fizikai és szellemi munkáját könnyítették meg. Végül soron az ember munkaeszközei közé sorolhatók a korszerű elektronikus számítógépek is, amelyek a szellemi munka sokszor ismétlődő, bonyolult, un. rutin feladatait képesek elvégezni. A számítógépek tehát nem fordulnak szembe az emberrel, aki megalkotta őket, hanem a gondolkodó ember irányítása alatt őt kiszolgálják. A szellemi munka ilyen értelmű "gépesítése" világméretekben csak a XX. század derekán kezdődött meg, bár a korábbi idők nagy feltalálói és matematikusai igen jelentős mértékben hozzájárultak ahhoz, hogy a számítástechnika mai fejlettségi színvonalát elérhesse.

Történelmi tény, hogy Kínában már az időszámításunk előtti évezredekben ismerték az egyik legegyszerűbb számológyszerkezetet, az abacust. A szerkezet egy keretbe foglalt rudsorból áll, ezeken a rudakon golyók (gyöngyök) mozgathatók, tologathatók. Segítségével igen gyorsan elvégezhetők egyszerű összeadási-kivonási műveletek, s ez teszi indokolttá többek között, hogy egyes keleti országok üzleteiben ma is abacusszal számolják ki a fizetendő számla összegét.

A további fejlődés szempontjából nagy jelentőségű a francia Pascal felfedezése, az átvitelképzés. Az átvitel keletkezését az alábbi egyszerű példán szemléltetjük:

$$\begin{array}{r} 5 \\ + 9 \\ \hline 14 \end{array}$$

↑
átvitel

Helyértékes számrendszerekben tehát (ilyen a tízes számrendszer is), ha az egy helyértéken képződő összeg eléri vagy meghaladja a számrendszer alapszámát, átvitel keletkezik az eggyel nagyobb helyértékre. Ennek ismeretében Pascal meg is szerkesztette a saját kis számolóberendezését, amely fogaske-
rekekkel működött, és figyelembe tudta venni az esetleges átvitelt.

1671-ben Leibniz német tudós rájött, hogy a szorzás visszavezethető ismételt összeadásra. Azt javasolta továbbá, hogy a számítástechnikában a kettes alapú (bináris) számrendszert használják a műveletek elvégzésekor. E felfedezés annyira előremutató volt, hogy mindmáig valamennyi elektronikus számítógép végső soron bináris számrendszerben dolgozik.

1786-ban Müller bevezette a regiszter fogalmát, és összeadó berendezést tervezett regiszterek felhasználásával. A regiszter az információ (például szám, adat) átmeneti tárolására szolgál, vagyis addig tartózkodik benne az információ, amíg éppen elvégezzük vele az előírt műveletet. A regiszterek a mai számítógépeknek is fontos alkotórészei, a korszerű számítógéptechnikában is sokoldalúan felhasználhatók.

1833-ban Babbage angol matematikus elektromechanikus számológépet tervezett, amely egyszerre 1000 db 50 helyértékes számot tudott tárolni. Az összeadást 1 másodperc alatt, a szorzást pedig 1 perc alatt végezte el. A gép egy példánya a londoni Műszaki Tudományos Múzeumban (Museum of Science and Industry) látható, bár végleges formájában sohasem készült el, mert alkotója folyamatosan átervezte.

1847-ben George Boole megalkotta a róla elnevezett eseményalgebrát, az ún. Boole-algebrát. Itt nem számokkal, hanem logikai változókkal, eseményekkel végezzük a műveleteket. A Boole-algebra kulcsfontosságú a mai számítástechnika szempontjából is, hiszen a számítógépek logikai döntései a Boole-algebra alapján állnak.

1890-ben Herman Hollerith bevezette a máig is használatos lyukkártyás adatfeldolgozást. Találmányát az USA 1890. évi népszámlálásánál használták fel első ízben. A lyukkártyás adatfeldolgozás forradalmi lépésnek tekinthető a számítástechnika történetében. Jellemző, hogy hagyományos módszerrel dolgozva a megelőző népszámlálás adatainak értékelése még nem fejeződött be, amikor már esedékessé vált az újabb népszámlálás. A Hollerith-féle adatfeldolgozó gépek ugyanezt az értékelést néhány hét alatt elvégezték, és az eredményeket papírra nyomtatva szolgáltatták.

1936 körül Karl Zuse megalkotta a mai számítógépek őseit, az első programvezérlésű számológépet. Az információt elektromechanikus jelfogók (relék) zárt vagy nyitott állapota tartalmazta. A gép bináris számrendszerben számolt, 30-50 művelet/perc gyorsasággal. Megvalósításához Zuse mintegy 2600 relét használt fel. Tőle függetlenül 1938-ban Kozma László műegyetemi tanár is megalkotta hasonló elven működő relés számítógépét.

1941-46 között az USA-ban már működött az első elektroncsövekkel működő számítógép, az ENIAC (Electronic Numerical Integrator And Calculator). Egy perc alatt 5000 összeadást tudott végezni, tehát lényegesen gyorsabban működött Zuse relés gépénél. A szükséges műveletek elvégzésére a gépet kívülről, az ún. dugaszoló táblán összeállított huzalozással lehetett utasítani; ez a felhasználást nagyon nehézkesé tette, és sok hibalehetőséget rejtett magában. Az elektroncsövek működtetése ezenkívül igen nagy villamos teljesítményt igényelt.

Ugyanebben az időben, éppen az említett nehézségek áthidalására dolgozta ki a magyar származású Neumann János matematikus az ún. tárolt program koncepciót. A Neumann-elv értelmében a számítógépnek címezhető belső tárolója van, amelyben mind az adatok, mind az adatfeldolgozást vezérlő utasítások (program) helyet foglalnak. A megfelelő adatot vagy utasítást a helyét meghatározó cím alapján választhatjuk ki. A Neumann-elv szerint működő számítógépek sokkal gyorsabbak, megbízhatóbbak a külső programozású (dugaszoló táblás) gépeknél; a világ mai számítógépparkja ugyyszólván kizárólag a tárolt program koncepció alapján működik. Az elektroncsövekből felépített, tárolt programmal vezérelt elektronikus számítógépeket szokás I. generációs számítógépeknek is nevezni, utalva ezzel a legkorábbi elektronikus alkatelem,

az elektroncső felhasználására. 1951-ben jelent meg az első, a kereskedelemben is kapható elektronikus számítógép, a UNIVAC, majd 1953-ban hozták nyilvánosságra az első szovjet gyártmányú elektronikus számítógépet.

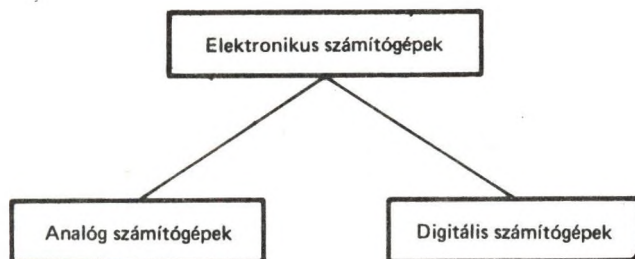
Az ötvenes években az elektroncsöveket fokozatosan felváltották a tranzistorok, amelyek villamos teljesítményigénye sokkal kisebb a csövéknél, továbbá lényegesen kevesebb helyet foglalnak el kis méreteik miatt. Megbízhatóságuk nagyobb, élettartamuk hosszabb. A tranzistorok alkalmazása a számítógépgyártásban 1965-66-ig szinte teljesen kiszorította az elektroncsöveket; tranzistorokból épül fel a világon az egyik legelterjedtebb számítógépcsalád, az IBM/360 is. A tranzistorokból felépített elektronikus számítógépeket szokás II. generációs gépeknek is nevezni, utalva arra, hogy a gép fő alkateleme - a tranzistor - az elektroncső utáni fejlettségi fokot képviseli az elektronikában.

1971-től megjelentek az első integrált áramkörös elektronikus számítógépek (például az IBM/370). Az integrált áramkörök tranzistoros áramkörök, de az áramkörök technológiája olyan kialakítást tesz lehetővé, hogy egy közös tranzistor helyén több száz miniatűr tranzisztort helyezhetnek el. Ezzel a technikával gyártott gépek helyfoglalása ismét tovább csökkent, villamos tulajdonságaik pedig nagymértékben javultak. Az integrált áramkörös elektronikus számítógépeket szokás III. generációs gépeknek is nevezni, utalva arra, hogy a gép alkatelemei a közös tranzistoroknál korszerűbb integrált áramkörök.

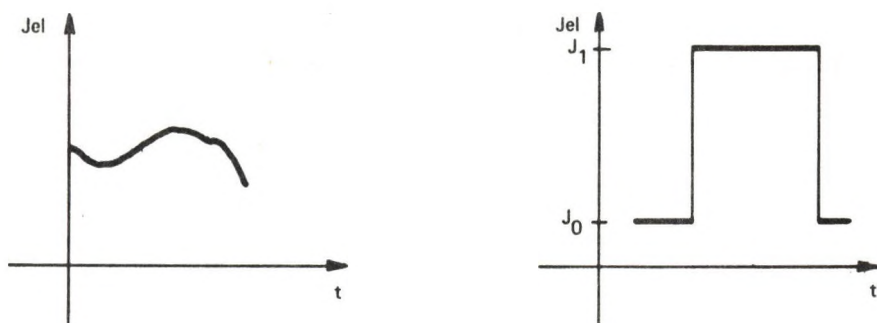
Szokás még un. IV. generációs gépekről is beszélni, amelyek műszaki tulajdonságai egy speciális utasítássorozattal (programmal) megváltoztathatók. A fejlődés várható iránya - úgy tűnik - a molekuláris tárolók irányába mutat, amelyekben még kisebb helyen még több információ rögzíthető és dolgozható fel. A világ számítógépparkját azonban ma még zömmel II. és III. generációs számítógépek alkotják, tehát leggyakoribb a tranzistor és az integrált áramkör alkalmazása.

1.2 ANALÓG ÉS DIGITÁLIS SZÁMÍTÓGÉPEK

A ma forgalomban levő elektronikus számítógépek működési elvük alapján két csoportra oszthatók:



Az információ rögzítését - analóg, illetve digitális uton - az 1. ábrán mutatjuk be.



1. ábra. Analóg és digitális jel

Látható, hogy az analóg jel folytonos, tehát minden időpillanatban meghatározott információtartalommal rendelkezik (például a jel nagysága). Ezzel szemben a digitális jel csak bizonyos diszkrét értékeket vehet fel, amelyeket számjegyes formában fejezünk ki (digit = számjegy). A digitális információ-ábrázolás pontosságát a rendelkezésre álló helyértékek száma határozza meg (például három tizedesjegynél $1/1000$ stb.). A ma használt számítógépek nagyobb része digitális elven működik, tehát az információt számjegyekké ala-

kitva, számjegyes formában tárolja. Mindamellettt egyes tudományos-műszaki számítások kényelmesebben végezhetők el analóg számítógépeken. Mivel e jegyzet ESZR számítógépek ismertetésével foglalkozik, analóg számítógépekről a továbbiakban egyáltalán nem teszünk említést (valamennyi ESZR számítógép digitális elven működik).

1.3 AZ ESZR SZÁMÍTÓGÉPCSALÁD

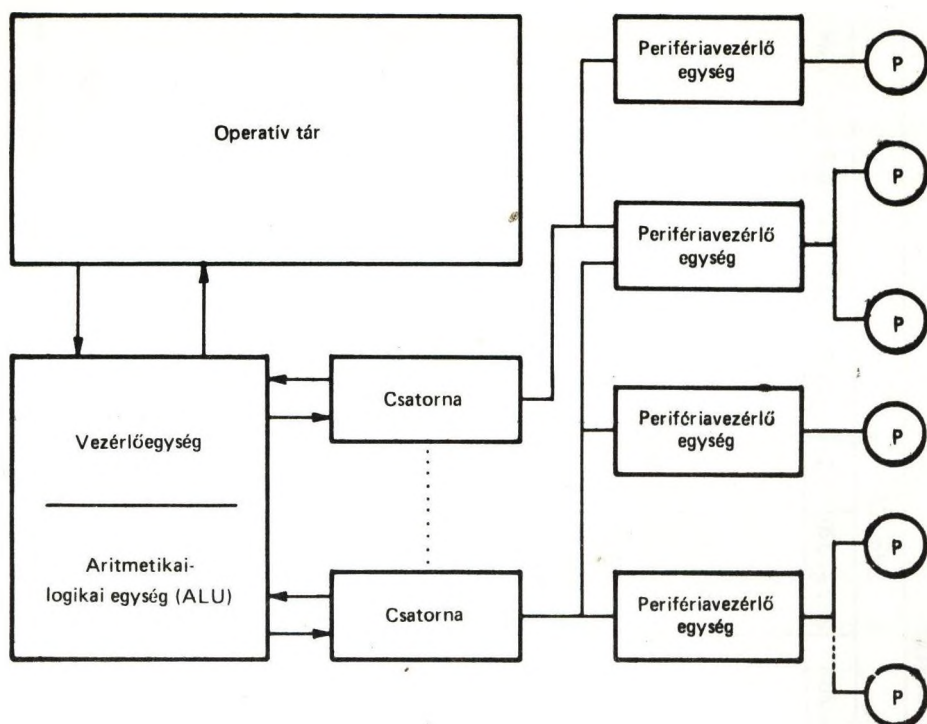
A szocialista országokban jelenleg a heterogén számítógéppark egységesítése folyik, melynek során egymással kompatibilis tagokból álló számítógépcsaládot hoznak létre. A gyártás és tervezés országok közötti munkamegosztás keretében folyik. A gépcsalád egyik fő tulajdonsága az, hogy bármelyik gépére írt program változtatás nélkül futtatható a gépcsaládhoz tartozó másik számítógépen, feltéve, hogy a program által igényelt erőforrások a másik gépen is megvannak (tárkapacitás, utasításkészlet, perifériakészlet stb.). A számítógép erőforrásain értjük a hardware-konfiguráció valamennyi elemét, a központi egység foglaltsági idejét, a használt adathalmazokat, sőt magukat a programokat is. A gépek fenti tulajdonságát tekintjük kompatibilitásnak, összeegyeztethetőségnek. Az ESZR programban (Egységes Számítógép Rendszer) hazánk a legkisebb modell, az R 10 gyártásával veszi ki részét, amely az illesztési problémák kiküszöbölése után kompatibilis a sorozat többi tagjával.

Az ESZR gépcsalád főbb tulajdonságait az 1. táblázatban foglaltuk össze. Az ESZR-hez tartozó valamennyi számítógép integrált áramkörös kivitelű, tehát az elektronika és a technológia legújabb eredményeinek felhasználásával készül. A 2. ábrán az ESZR számítógépek elvi műszaki felépítését szemlélítjük.

A vezérlőegységet és az aritmetikai-logikai egységet együttesen központi egységnek, processzornak is szokás nevezni. A központi egységet tekinthetjük a tulajdonképpeni "számítógépnek". A vezérlőegység irányítja a teljes számítógéprendszer működését, az aritmetikai-logikai egység pedig különböző műveleteket végez a feldolgozandó adatokkal.

Az ESZR számítógépei

Paraméter	Modell						
	ESZ-1010	ESZ-1020A	ESZ-1020	ESZ-1030	ESZ-1040	ESZ-1050	ESZ-1060
Alapműveletek végrehajtási ideje (μs)							
- Rövid műveletek	Speciális utasításkészlet	15	20-30	5-8	0,9-1,8	1	0,5
- Lebegőpontos összeadás-kivonás	A műveletek programozottak	-	50-70	10-16	2,6-3,5	1,4-2,4	0,5
- Szorzás	A felszavas összeadás ideje	80-120	220-350	35	5,5-13,1	2-2,4	1
- Duplaszavak szorzása	2,6-3,6 μs	-	1200	60	12	12	1,5
Az utasításrendszer különleges-ségei		Speciális utasításkészlet	Teljes programkompatibilitás				
A vezérlés elve	Mikroprogramozott						
Az operatív tár kapacitása (Kbyte)	8 (bővíthető)	16-64	64-256	128-512	256-1024	128-1024	256-2048
Az operatív tár ciklusideje (μs)	1	1,5	2,0	1,25	1,35	1	0,6
A multiplex csatorna átviteli sebessége (Kbyte/s)	160	35-220	10-16	40	50-200	100-450	100-450
Szelektor csatornák maximális száma	1	2	2	3	6	6	6
Szelektor csatornák átviteli sebessége (Kbyte/s)	240	120	max. 300	600	1200	1300	1300
Integrált áramkörök típusa	TTL ECL						



2. ábra. ESZR számítógép blokkvázlata

Az operatív tár (központi tár, munkatár) a feldolgozást vezérlő programok és az eredmények, adatok tárolására szolgál (a Neumann-féle tárolt program koncepció értelmében). Az információ egységenként tárolódik. A legkisebb információegység a bit (binary digit). Egy bit információtartalmu például egy kettes számrendszerbeli számjegy, egy logikai változó értéke stb. Mivel a bit igen kis egység, a számítógéptechnikában nagyobb információegységeket használnak. Ezek az ESZR gépeknél az alábbiak:

- 1 byte = 8 bit + 1 ellenőrző bit,
- Félszó = 2 byte,
- Szó = 4 byte,
- Duplaszó = 8 byte.

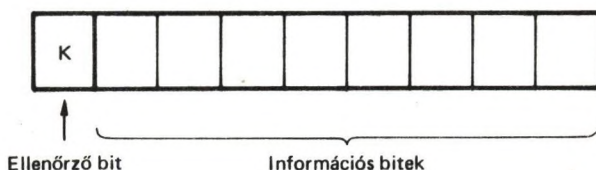
A legáltalánosabban használt információegység a byte, a nagyobb egységek mind többszöröse a byte-nak. Az operatív tár is byte-onként tárolja az információt. Az egyes byte-ok 0-tól kezdődően sorszámokkal vannak ellátva; ezt a

sorszámot az illető byte címének nevezzük. Ha tehát az operatív tárban valamely byte-ra akarunk hivatkozni (például meg akarjuk változtatni a tartalmát), akkor meg kell címeznünk a kívánt byte-ot. Ezzel biztosítjuk azt, hogy a műveletvégzésben csak a megcímzett byte vesz részt. Ha a műveletet gépi szóval végezzük, akkor egyszerre négy byte-ot címzünk meg (1 szó = 4 byte) stb. Az operatív tár kapacitása - vagyis annak mérőszáma, hogy az illető tár hány byte-ot tartalmaz - igen fontos tulajdonsága a számítógépnek. Mivel a tárkapacitás néhány ezertől néhány millió byte-ig terjedhet, bevezettek egy nagyobb egységet, a Kilobyte-ot (Kbyte).

$$1 \text{ Kbyte} = 1024 \text{ byte}$$

A tárolókapacitást általában Kbyte-ban (ritkábban Kszóban) fejezzük ki. Annak oka, hogy 1 Kbyte definíciószerűen 1024 byte, nem pedig 1000 byte, a kettes számrendszerhez való alkalmazkodásban keresendő. Jó közelítéssel azt mondhatjuk, hogy egy tár Kbyte-ban kifejezett kapacitása akkora, ahány 1000 byte-os az illető tár (a becslésben kb. 2 %-os hibával számoltunk).

Megjegyezzük, hogy a byte információtartalmát illetően mindig a nyolc un. információs bitre hivatkozunk, az ellenőrző bitet itt nem vesszük figyelembe:



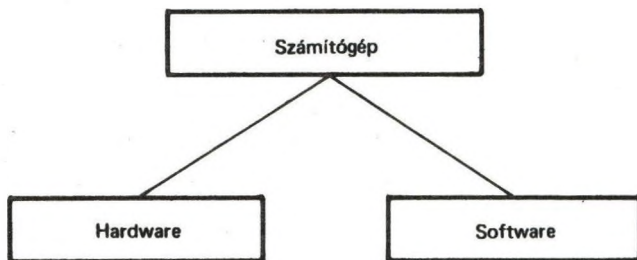
A csatornák feladata az, hogy a központi egység és a perifériák közötti beviteli-kihozatali (input-output) adatforgalmat irányítsák. A csatorna tehát egy kisebb bonyolultsági fokú speciális számítógépnek tekinthető, amely az adatátvitelt önállóan vezérli és ellenőrzi.

A perifériavezérlő egységek irányítják magukat a perifériákat, a felhasználó számára is részben hozzáférhető input-output eszközöket. Egyes perifé-

riák feladata olykor nem kifejezetten az információcsere, hanem az információ bizonyos tárolása; ezeket az operatív tár külső kibővítésének tekinthetjük. Vég-
ső soron azonban ezek is adatforgalmat bonyolítanak le a központi egységgel.

A korszerű számítógépek hatékony működtetéséhez már nem elegendő ma-
gának a gépnek valamennyi jó műszaki tulajdonsága, adottsága sem. Szükség
van olyan speciális üzemeltető programrendszerre (operációs rendszerre),
amely irányítani képes a számítógépet mint igen bonyolult automatát. Ennél-
fogva a számítógép a felhasználó oldaláról nézve két fő részből áll:

- a gép mint automata összes elektronikus áramkörével és műszaki beren-
dezésével, adottságával (hardware);
- a gépet üzemeltető, irányító programrendszer (software).



A számítógép hardware-funkciói és felépítése a gyártás során alakulnak ki, és a felhasználó általában nem változtathatja meg azokat. A software-ellátottság ezzel szemben a felhasználó által is bővíthető, változtatható; a software fejlesztése és karbantartása többnyire a rendszerprogramozók feladata.

A 2. ábrából látható, hogy valamely számítógép központi egysége egynél több csatornát is ki tud szolgálni. Ugyanigy egy csatornára általában több perifériavezérlő egység, egy perifériavezérlő egységre pedig egy vagy több periféria kapcsolható. A számítógéprendszer egy adott hardware-kiépítését a rendszer (hardware) konfigurációjának nevezzük. A konfigurációban tehát számszerűen is kifejezésre juttatjuk azt, hogy az adott rendszerben hány csatorna, mennyi és milyen perifériavezérlő egység, illetve periféria üzemel. A konfiguráció változtatásának alsó és felső határa van: minimális és maximális konfi-

gurációról beszélhetünk. A minimális konfigurációba azok a hardware-elemek tartoznak, amelyek nélkül a számítógéprendszer nem üzemeltethető (például feltétlenül szükség van operatív tárra, központi egységre, legalább egy input és output perifériára stb.). A maximális konfiguráció ezzel szemben azt a hardware-kiépítettséget határozza meg, amelynél tovább nem bővíthető a rendszer hardware-elemekkel (például maximálisan hány csatorna üzemeltethető, egy-egy csatornára legfeljebb hány és milyen perifériavezérlő egység, illetve periféria kapcsolható stb.). Megjegyezzük, hogy a gyakorlatban üzemeltetett számítógéprendszerek általában nem érik el a maximális konfigurációt, tehát a maximális konfiguráció inkább csak hardware-bővítési lehetőséget jelent.

2. Aritmetikai és logikai műveletek

2.1 A KETTES, A TÍZES ÉS A TIZENHATOS SZÁMRENDSZER

Az előző fejezetben említettük, hogy valamennyi digitális számítógép végső soron kettes alapú (bináris) számrendszerben dolgozik. Így például az operatív tár egyes byte-jaiban elhelyezett információ is bináris számok alakjában jelenik meg, sőt a gépek még a decimális számokat is bináris formában ábrázolják. Mindez azzal áll összefüggésben, hogy a számítógépek alapáramkörei két lehetséges állapotban lehetnek, és e két állapot egymástól élesen megkülönböztethető. Mivel a bináris számrendszerben is kétféle számjegy - 1 és 0 - fordulhat elő, kézenfekvő volt az alapáramkörök két diszkrét állapotának a bináris 1-et és 0-t megfeleltetni. E körülmények indokoltá teszik, hogy a számítógépek működésének tanulmányozása előtt alaposabban is megismerkedjünk a kettes és a tizenhatos számrendszerrel.

Tekintsük a következő, tízes számrendszerbeli számot:

4 6 0 3 , 7 9

Minthogy helyértékes számrendszerben ábrázoltuk a számot, mindennapi tapasztalatunkkal egyezően például a 6-os számjegy értékét

alaki értéke ("6") és

helyértéke (100)

határozza meg. Az adott példában - mivel a 6-os számjegy a százask helyén áll, $6 \cdot 100 = 600$ lesz a tényleges értéke.

Ugyanígy állunk a tizedesvesszőtől jobbra levő törtrész értékével is: a 7-es itt $7/10$ -et, a 9-es pedig $9/100$ -ot jelent. Ennek alapján az idézett számot mintegy "összeállíthatjuk" az egyes helyértékeken levő számjegyek tényleges értékeiből:

$$\begin{array}{rcl}
4 \cdot 1000 & = & 4\,000 \\
6 \cdot 100 & = & 600 \\
0 \cdot 10 & = & 00 \\
3 \cdot 1 & = & 3 \\
7 \cdot 0,1 & = & 0,7 \\
9 \cdot 0,01 & = & 0,09 \\
& + & \\
\hline
\text{Összesen:} & & 4\,603,79
\end{array}$$

Vegyük észre, hogy valamennyi helyérték kifejezhető a számrendszer alapszámának - 10-nek - valamelyik hatványával. Így például

$$\begin{array}{rcl}
1\,000 & = & 10^3 \\
100 & = & 10^2 \\
10 & = & 10^1 \\
1 & = & 10^0 \\
0,1 & = & 10^{-1} \\
0,01 & = & 10^{-2}
\end{array}$$

Tíz hatványai tehát balról jobbra helyértékenként eggyel csökkennek, és a tizedesvesszőtől jobbra negatív értéket vesznek fel a kitevők. Egy szám negatív kitevőjű hatványa pedig nem más, mint a szám azonos abszolút értékű pozitív kitevőjű hatványának reciproka, tehát például

$$10^{-2} = \frac{1}{10^2}$$

Ezen ismeretek birtokában most már akármilyen nagy számot fel tudunk írni 10-es számrendszerben, ha felhasználjuk a rendelkezésre álló 10 db számjegyet (0, 1, 2, ..., 9), és meghatározzuk azok tényleges értékét az elfoglalt helyérték alapján. Írjuk fel az előbb vizsgált számot 10 hatványai segítségével:

$$4603,79 = \underline{4} \cdot 10^3 + \underline{6} \cdot 10^2 + \underline{0} \cdot 10^1 + \underline{3} \cdot 10^0 + \underline{7} \cdot 10^{-1} + \underline{9} \cdot 10^{-2}$$

Valamely számnak ilyen formájú felírását polinom alakban való ábrázolásnak nevezzük, hiszen az adott számot a fenti meghatározott szerkezetű többtagú kifejezés (polinom) alakjában fejeztük ki. Bebizonyítható, hogy minden szám felírható polinom alakban, és a felírás egyértelmű. A szokásos számforma visszaállítása ebből a polinom alakból úgy történik, hogy a 10 hatványai előtt álló szorzótényezőket (együtthatókat, a tulajdonképpeni számjegyeinket) sorban egymás után írjuk, ügyelve arra, hogy a 0-dik hatvány együtthatója után kitegyűk a tizedesvesszőt. Így visszakapjuk az eredeti számot:

$$4\ 6\ 0\ 3\ ,\ 7\ 9$$

Ha egész számról van szó, akkor a tizedesvesszőt mindig odaképzeltethetjük a szám végére, s így könnyebben megállapíthatjuk az egyes helyértékekhez tartozó hatványokat.

A tízes számrendszerhez teljesen hasonlóan írjuk fel a számokat kettes és tizenhatos számrendszerben. A különbség csupán az, hogy a kettes számrendszer alapszáma 2, a tizenhatosé 16 (innen az elnevezés).

Általános formában a következőképpen írhatjuk fel a számokat kettes, illetve tizenhatos számrendszerben:

a) Kettes számrendszerben (binárisan):

$$\dots + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \dots$$

b) Tizenhatos számrendszerben (hexadecimálisan):

$$\dots + b_3 \cdot 16^3 + b_2 \cdot 16^2 + b_1 \cdot 16^1 + b_0 \cdot 16^0 + b_{-1} \cdot 16^{-1} + \dots$$

Az a_3, a_2, a_1, a_0 stb., valamint a b_3, b_2, b_1, b_0 stb. együtthatók az alkalmazott számrendszer számjegyeit jelentik. Itt érzékeltetjük azt, hogy a szám egészrésze és törtrésze egyaránt tetszőleges hosszúságú lehet (a pontok a lehetséges további hatványokat, helyértékeket szimbolizálják). Ha tehát egy számban rögzített az egészrészt a törtrésztől elválasztó vessző helye, akkor a szám jobbról és balról egyaránt tetszőleges számú értéktelen zérussal egészíthető ki anél-

kül, hogy értéke megváltoznék. Ezt a lehetőséget a konvertálásoknál (lásd 2.2 pont) ki is használjuk.

Valamely számrendszerben a számrendszer alapszámát radixnak nevezzük. A radix sohasem lehet 0 vagy 1, mert ezek valamennyi hatványa 0, illetve 1, tehát nem alkalmasak helyértékes számábrázolásra. Az eddig megismert számrendszerek alapszámai:

	Tizes számrendszer	Kettes számrendszer	Tizenhatos számrendszer
Radix	10	2	16

A tizes számrendszer használatából tudjuk, hogy a 9-nél nagyobb számok már nem fejezhetők ki egyetlen helyértéken: a 10 - a számrendszer alapszáma - is két helyértéket vesz igénybe, két számjegy, az 1 és 0 segítségével ábrázolható. Ugyanez a helyzet a kettes és a tizenhatos számrendszerben: az előforduló legnagyobb alaki értékű számjegy az 1, illetve a 15. Megfogalmazhatjuk tehát az általános érvényű szabályt:

Egy számrendszerben az előforduló legnagyobb alaki értékű számjegy = radix - 1.

Mivel a tizenhatos számrendszerben az előforduló számjegyek 15-ig haladnak, és a tizes számrendszer szokásos számjegyeivel csak 9-ig jelölhetők, bevezették az ábécé első hat betűjét a további számjegyek jelölésére. A 2. táblázatban feltüntettük a tizes, kettes és tizenhatos számrendszer számjegyeinek ábrázolását egészen 16-ig.

A táblázatból látható, hogy a tizes számrendszerbeli 10-től kezdődően 15-ig az ábécé első hat nagybetűjével jelöljük a tizenhatos számrendszer ide eső számjegyeit; a betűs jelölés azért indokolt, mert így 1 helyértéken kifejezhetjük az egyes hexadecimális számjegyeket, még azokat is, amelyek 10-nél nagyobbak, vagyis amelyek a tizes számrendszerben 2 számjeggyel lennének csak kifejezhetők.

2. táblázat

Tizes számrendszer	Kettes számrendszer	Tizenhatos számrendszer
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

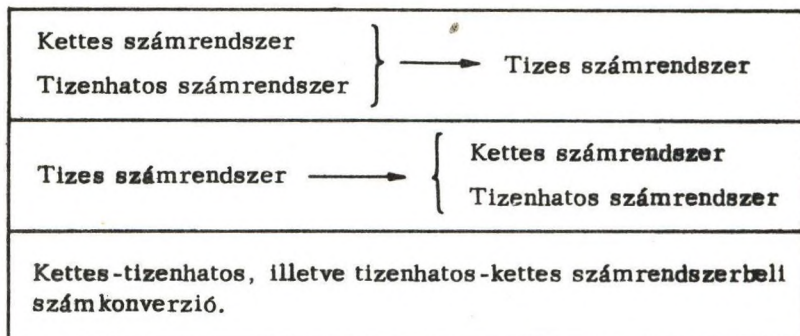
A kettes számrendszer használata azért célszerű, mert a számítógép csak "1" és "0" állapotot tud megvalósítani alapáramköreivel. A tizenhatos számrendszer alkalmazása a felhasználónak jelent könnyítést, mivel nem kell olyan sok 1-es és 0-t írnia a programozás során. A két számrendszer egymás közötti kapcsolatát, valamint a tizes számrendszerrel való összefüggését a most következő részben ismertetjük.

2.2 SZÁMKONVERZIÓK

Számkonverziókn értjük valamely számnak egy adott számrendszerből egy másik számrendszerbe való átírását. Ügyeljünk arra, hogy a konverzió során az adott szám értéke nem változik, csupán megjelenési formája, "külsőalakja"

alkalmazkodik az új számrendszerhez. Például tízes számrendszerből kettesbe konvertáláskor a 2, 3, 4, 5 stb. számjegyek eltűnnek, hiszen a bináris számrendszer csak 0 és 1 számjegyeket használ.

A számkonverziókat az alábbi felosztásban ismertetjük:



A számkonverziók fontosságát alátámasztja az is, hogy a számítógép kettes számrendszert használ, a felhasználó azonban szívesebben dolgozik tizenhatos vagy tízes számrendszerben.

a) Kettes-tízes és tizenhatos-tízes számrendszerbeli számkonverzió

Mindkét konverzió esetében azt az eljárást követjük, hogy megvizsgáljuk, vajon 2 vagy 16 megfelelő hatványa előfordul-e, és ha igen, hányszor fordul elő az adott számban. E hatványokat tízes számrendszerbe átírva kapjuk a szám decimális számrendszerbeli alakját.

Példák

$$\begin{array}{cccccccccc}
 1 & & 1 & & 0 & & 1 & & 0 & & 1 & , & 0 & & 1 & & 1 \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow^{(2)} \\
 1 \cdot 2^5 & + & 1 \cdot 2^4 & + & 0 \cdot 2^3 & + & 1 \cdot 2^2 & + & 0 \cdot 2^1 & + & 1 \cdot 2^0 & + & 0 \cdot 2^{-1} & + & 1 \cdot 2^{-2} & + & 1 \cdot 2^{-3} \\
 32 & + & 16 & + & 0 & + & 4 & + & 0 & + & 1 & + & 0 & + & \frac{1}{4} & + & \frac{1}{8} = 53\frac{3}{8}
 \end{array}$$

$$\begin{array}{ccccccc}
 1 & & A & & 3 & & E & , & 7 & & F \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow^{(16)} \\
 1 \cdot 16^3 & + & 10 \cdot 16^2 & + & 3 \cdot 16^1 & + & 14 \cdot 16^0 & + & 7 \cdot 16^{-1} & + & 15 \cdot 16^{-2} \\
 4096 & + & 2560 & + & 48 & + & 14 & + & \frac{7}{16} & + & \frac{15}{256} = 6718\frac{127}{256}
 \end{array}$$

A konvertálandó számok indexei (2, illetve 16) arra utalnak, hogy az adott szám melyik számrendszerben értelmezendő. Ezt az indexet megkülönböztetés céljából ajánlatos odairni minden számhoz; így tehetjük egyértelművé a szám-ábrázolást. A mindennapi életben a tízes számrendszerbeli számokhoz nem írjuk oda a "10" indexet, de több számrendszer-egyidejű használata, előfordulása esetén célszerű jelölni az alapszámot. Például a "10" szám éppugy jelenthet 2-t, 10-et vagy 16-ot attól függően, melyik számrendszerben értelmezzük!

b) Tízes-kettes és tízes-tizenhatos számrendszerbeli számkonverzió

Mindkét konverzió esetében megvizsgáljuk, hogy 2-nek, illetve 16-nak mely hatványai vesznek részt a konvertált számban, és azok hányszor fordulnak elő benne. A vizsgálatot az egészrész 0-dik hatványától kezdjük, és külön algoritmus szerint végezzük az egészrészre és a törtrészre vonatkozólag. Az eljárást egy példán mutatjuk be.

Példa: Irjuk át a 73,63 decimális számot kettes számrendszerbe!

Megoldás:

Az egészrész átalakítása

73 ₍₁₀₎ : 2		
36	1	K i o l v a s á s
18	0	
9	0	
4	1	
2	0	
1	0	
0	1	

A törtrész átalakítása

0,63 ₍₁₀₎ · 2		
1	26	i r á n y a
0	52	
1	04	
0	08	
0	16	
0	32	
0	64	

A teljes megoldás: 1001001,1010000...₍₂₎

Az egészrész konvertálását úgy hajtottuk végre, hogy a decimális egészrész addig osztottuk 2-vel, amíg az osztás eredménye 0-t nem adott. A decimális törtrészt pedig addig szoroztuk 2-vel, amíg a kívánt pontosságot el nem értük. Az osztásnál mindig a maradékot írjuk a függőleges vonal jobb oldalára, szorzásnál pedig a szorzat első számjegyét. A kettes számrendszerbeli törtrész nem biztos, hogy pontosan egyezik a decimális törtrésszel, azaz előfordul, hogy a törtrész konvertálása végtelen bináris törtrészt eredményez. Ilyenkor a szükséges pontosság alapján döntünk arról, meddig folytassuk a konvertálást. Az egész-, illetve törtrész átalakítása után a konvertált egész- és törtrész összeillesztésével nyerjük a teljes konvertált számot. Megjegyezzük, hogy az egészrész mindig pontosan konvertálható.

Példa: Írjuk át a 2611,63 decimális számot tizenhatos számrendszerbeli

Megoldás:

Az egészrész átalakítása

$$2611_{(10)} : 16$$

163	3	K
10	3	i
0	A	o
		l
		v
		a
		s
		á
		s

A törtrész átalakítása

$$0,63_{(10)} \cdot 16$$

A	08
1	28
4	48

A teljes megoldás: $A33, A14 \dots_{(16)}$

c) Kettes-tizenhatos és tizenhatos-kettes számrendszerbeli számkonverzió

Az átalakítás alapjául az szolgál, hogy 2-nek 16 egész kitevőjű hatványa: $16 = 2^4$. Ennélfogva minden egyes hexadecimális számjegy megfelel négy bináris számjegynek, azaz hexadecimális számjegyenként 4-gyel nő 2 hatványa

a megfelelő bináris számban. Ez nagyon egyszerűvé teszi az átalakítást, hiszen 4 bináris számjegyet összevonhatunk 1 hexadecimális számjeggyé és megfordítva. Lássunk néhány példát az elmondottakra!

Példa: Alakítsuk át tizenhatos számrendszerbeli számmá az alábbi bináris számot!

$$\begin{array}{ccccccc} \underbrace{00011001} & \underbrace{00001001} & \underbrace{00111101} & \underbrace{0100} & & & \\ 1 & 9 & 0 & 9 & 3 & D & 4 \end{array} \begin{array}{l} (2) \\ (16) \end{array}$$

Megoldás: A fenti felosztás értelmében a tizedesvesszőtől kezdve az egész- és törtrészt 4-4 bináris számjegyet tartalmazó csoportokra osztjuk. A szám "végein", ha a legszélső négyes csoportok nem lennének teljesek, megfelelő számú zérust írunk a számhoz. Ezt jogunk van megtenni, hiszen egy szám értéke nem változik, ha egészrésze elé és törtrésze után tetszőleges számú 0-t írunk. Az így kapott - 4 bináris számjegyből álló - számokat értékük szerint átírjuk hexadecimális számjegyekké. A négyes csoportok értelmezése a következő:

$$2^3 \ 2^2 \ 2^1 \ 2^0$$

$$b \ b \ b \ b ,$$

ahol b egy bináris számjegyet jelöl (1 vagy 0). A példa megoldása tehát a jelölt csoportosítás mellett a következő:

$$1909,3D4_{(16)}$$

Az átalakítás szabályai fordított irányban is érvényesek: minden egyes hexadecimális számjegy felbontható 4 bináris számjegyre. A hexadecimális számjegyeket sorban átírva binárisakká a kívánt kettes számrendszerbeli számhoz jutunk.

Példa: Írjuk át bináris számmá az alábbi hexadecimális számot!

$$\begin{array}{ccccccc} 4 & 0 & F & A & , & C & 7 \\ \underbrace{0100} & \underbrace{0000} & \underbrace{1111} & \underbrace{1010} & , & \underbrace{1100} & \underbrace{0111} \end{array} \begin{array}{l} (16) \\ (2) \end{array}$$

Végül "elfelejtjük", hogy a bináris szám tulajdonképpen négyes csoportokból jött létre, s így nyerjük a konvertálás végeredményét, amelyben már nem vesszük figyelembe a végek értéktelen zérusait.

$$100000011111010, 11000111_{(2)}$$

2.3 MŰVELETEK. MŰVELETVÉGZÉS A SZÁMÍTÓGÉPBN

2.3.1 ÖSSZEADÁS KETTES ÉS TIZENHATOS SZÁMRENDSZERBN

Bináris és hexadecimális számokat elvileg ugyanugy adunk össze, mint decimális számokat. Mindkét esetben figyeljük, hogy elértük vagy meghaladtuk-e az egy helyértéken képződő részösszeg értékeivel az adott számrendszer alapszámát (2-t, illetve 16-ot). Ha igen, akkor átvitel keletkezik az eggyel nagyobb helyértékre, és ezt az átvitelt ott összeadandóként kezeljük. Vigyázzunk a számok megjelenítésére; a részösszeg megállapításánál gondoljunk a polinom alakra, arra, hogy a bináris és hexadecimális számok saját számrendszerükben más alakot öltenek, mint decimális megfelelőik a tízes számrendszerben.

A feladatok megoldását könnyen ellenőrizhetjük, ha az összeadandókat és a kapott eredményt decimális számmá konvertáljuk, és az átalakított számokkal is elvégezzük a műveletet tízes számrendszerben. Így egyszersmind a konvertálást is gyakorolhatjuk.

Példák bináris összeadásra:

$$\begin{array}{r}
 0 \\
 + 0 \\
 \hline
 0
 \end{array}
 \quad
 \begin{array}{r}
 0 \\
 + 1 \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 + 0 \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 + 1 \\
 \hline
 10
 \end{array}$$

$\uparrow$
 átvitel

$$\begin{array}{r}
 1001 \\
 + 1111 \\
 \hline
 0110
 \end{array}
 \quad
 \begin{array}{r}
 11001 \\
 + 00111 \\
 \hline
 11110
 \end{array}$$

$$\begin{array}{r}
 + 11 \\
 \hline
 11000
 \end{array}
 \quad
 \begin{array}{r}
 + 1 \\
 \hline
 100000
 \end{array}$$

A fenti példákban az összeadást két lépésben végeztük el: először meghatároztuk az egyes helyértékeken képződő részösszegeket és a keletkezett átviteleket, majd az átvitelekkel korrigáltuk a részösszegeket. Kellő gyakorlattal rendelkezők az átviteleket egy lépésben kezelhetik a részösszeg megállapításával, azonban így nagyobb a tévedés valószínűsége. Háromnál több összeadandó esetén - mivel az esetleg halmozódó átvitelek észben tartása fáradságos - javasolt módszer, hogy az összeadást több lépésben, mindig két-két taggal végezzük. Figyeljük meg, hogy bináris összeadásnál sokkal gyakoribb az átvitelkezelés, mint decimálisnál.

Példák hexadecimális összeadásra:

$$\begin{array}{cccccccc}
 8 & 8 & 8 & 8 & 8 & 8 & 8 & F \\
 + 1 & + 2 & + 3 & + 7 & + 8 & + 9 & + A & + F \\
 \hline
 9 & A & B & F & 10 & 11 & 12 & 1E
 \end{array}$$

átvitelek

$$\begin{array}{ccccccc}
 & 1 & F & 2 & D & A & \\
 + & 2 & 8 & 9 & D & 1 & \\
 \hline
 & 3 & 7 & B & A & B & \\
 + & 1 & 1 & & & & \\
 \hline
 & 4 & 7 & C & A & B &
 \end{array}$$

Az átvitelek kezelésére ugyanaz érvényes, mint a bináris összeadásnál, azzal a különbséggel, hogy itt viszonylag ritkább az átvitel keletkezése, mert nagyobb valószínűséggel nem éri el valamely részösszeg a számrendszer alapszámát, a 16-ot.

2.3.2 KIVONÁS KETTES ÉS TIZENHATOS SZÁMRENDSZERBEN

A bináris és hexadecimális számok kivonása elvileg ugyanugy megy végbe, mint a decimális számok kivonása. Ha valamely helyértéken a kivonandóban nagyobb számjegy áll, mint a kisebbítendőben, a kivonást nem tudjuk elvégezni. Ezért az eggyel nagyobb helyértékről áthozatot képeztünk (az átvitel ellentéte), s így már el tudjuk végezni a kivonást az adott helyértéken. Az áthozat értékét az eggyel nagyobb helyértéken hozzáadjuk a kivonandóhoz, ezzel korrigáljuk az áthozat okozta eltérést.

Lássunk egy példát először a tízes számrendszerben!

$$\begin{array}{r} 4 \quad 3 \\ - 1 \quad 8 \\ \hline \end{array}$$

$$\begin{array}{r} \quad \quad 1 \\ \quad \quad \curvearrowright \\ 4 \quad 13 \\ - 1 \quad 8 \\ \hline 5 \end{array}$$

$$\begin{array}{r} 4 \quad 13 \\ - 2 \quad 8 \\ \hline 2 \quad 5 \end{array}$$

Mivel 3-ból 8-at nem tudunk kivonni, a tízesek helyéről áthozatot képeztünk (ennek értéke 10), s most már 13-ból ki tudjuk vonni a 8-at. Az áthozat visszajuttatásáról gondoskodnunk kell, tehát a kivonandóban a tízesek helyén álló számjegyhez (1-hez) hozzáadjuk az áthozat értékét, s azután folytatjuk a kivonást. Az áthozatképzés tetszőlegesen sok helyértéken át "terjedhet", amíg csak áthozat van, annak visszajuttatásáról gondoskodnunk kell.

Példák bináris kivonásra:

$$\begin{array}{r} 0 \\ - 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ - 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ - 1 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 10 \\ - 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \\ - 1 \ 1 \ 1 \ 1 \\ \hline 0 \ 0 \ 1 \ 1 \ 0 \end{array}$$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \ 1 \\ - 1 \ 0 \ 1 \ 1 \ 0 \\ \hline 0 \ 1 \ 0 \ 1 \ 0 \ 1 \end{array}$$

Példák hexadecimális kivonásra:

$$\begin{array}{r} 8 \\ - 4 \\ \hline 4 \end{array}$$

$$\begin{array}{r} E \\ - 9 \\ \hline 5 \end{array}$$

$$\begin{array}{r} F \\ - B \\ \hline 4 \end{array}$$

$$\begin{array}{r} 10 \\ - 3 \\ \hline D \end{array}$$

$$\begin{array}{r} A \ 5 \ 0 \ 7 \ E \ 2 \ F \\ - 3 \ 8 \ F \ F \ B \ 7 \ E \\ \hline 6 \ C \ 0 \ 8 \ 2 \ B \ 1 \end{array}$$

2.3.3 KOMPLEMENTÁLÁS. NEGATÍV SZÁMOK ÁBRÁZOLÁSA

Komplementáláson azt értjük, hogy egy számot egy másik számmá egészítünk ki. Például ha 56-hoz hozzáadunk 44-et, az eredmény 100 lesz, és azt mondhatjuk, hogy 56-ot 100-hoz komplementáltuk, vagyis 56-nak a 100-as komplemente 44. A kettes számrendszer sajátosságaiból adódóan kétféle komplementet különböztetünk meg: egyes és kettes komplementet.

a) Egyes komplement

Valamely bináris szám egyes komplementét úgy kapjuk meg, hogy minden egyes számjegyet ellentétesre fordítjuk (invertálás). Tehát ahol 0 volt, oda 1 kerül, ahol 1 volt, oda 0. Például 13 egyes komplemente a következőképpen származtatható:

$$\begin{array}{rcl} 13 = & 1 & 1 & 0 & 1 &_{(2)} \\ \text{invertálás} & 0 & 0 & 1 & 0 &_{(2)} \end{array} \quad (=2_{10})$$

Látható, hogy egy bináris szám egyes komplemente az adott bináris számot az elfoglalt helyértékeken ábrázolható maximális értékű bináris számmá egészíti ki (példánkban $13 + 2 = 15$). Tekintsük még 5-nek az egyes komplementét:

$$\begin{array}{rcl} 5 = & 1 & 0 & 1 &_{(2)} \\ \text{invertálás} & 0 & 1 & 0 &_{(2)} \end{array} \quad (=2_{10})$$

$5 + 2 = 7$, a három bináris helyértéken ábrázolható maximális értékű bináris szám (ui. $7_{10} = 111_2$).

b) Kettes komplement

Valamely bináris szám kettes komplementét úgy kapjuk meg, hogy az egyes komplementéhez 1-et hozzáadunk. Például 13 kettes komplemente a következőképpen származtatható:

$$\begin{array}{rcl} 13 = & 1 & 1 & 0 & 1 &_{(2)} \\ \text{invertálás} & 0 & 0 & 1 & 0 &_{(2)} & - \text{ egyes komplement} \\ 1 \text{ hozzáadása} & + & & & 1 & \\ & & 0 & 0 & 1 & 1 &_{(2)} & - \text{ kettes komplement} \end{array}$$

Tehát 13-nak a kettes komplemente 3. Látható, hogy egy bináris szám kettes komplemente az illető bináris számot eggyel nagyobb számmá egészíti ki, mint az egyes komplement. Tehát a kettes komplement 2. egész számú hatványához komplementál:

$$\begin{array}{r}
 13 \\
 + \quad 3 \\
 \hline
 16
 \end{array}
 \qquad
 \begin{array}{r}
 1 \ 1 \ 0 \ 1_{(2)} \\
 + \quad 0 \ 0 \ 1 \ 1_{(2)} \\
 \hline
 1 \ 0 \ 0 \ 0 \ 0_{(2)}
 \end{array}$$

A kettes komplementens fenti tulajdonsága igen jól kihasználható a kivonásnál, ugyanis a kivonást összeadásra vezethetjük vissza, ha a kivonandót kettes komplementensben ábrázoljuk.

Tekintsük az alábbi kivonást:

$$\begin{array}{rcl}
 13 & - & 7 \\
 1 \ 1 \ 0 \ 1_{(2)} & - & 13 \\
 0 \ 1 \ 1 \ 1_{(2)} & - & 7 \\
 1 \ 0 \ 0 \ 0_{(2)} & - & 8 \text{ (7 egyes komplementense)} \\
 1 \ 0 \ 0 \ 1_{(2)} & - & 9 \text{ (7 kettes komplementense)}
 \end{array}$$

Adjuk hozzá 13-hoz 7-nek a kettes komplementensét:

$$\begin{array}{rcl}
 1 \ 1 \ 0 \ 1_{(2)} & - & 13 \\
 + \quad 1 \ 0 \ 0 \ 1_{(2)} & - & + \quad 9 \\
 \hline
 1 \ 0 \ 1 \ 1 \ 0_{(2)} & & 22
 \end{array}$$

Ha most a keletkezett átvitelt figyelmen kívül hagyjuk, azaz az eredményből levonunk 16-ot, a különbség 6, tehát helyes eredményt kaptunk. Fentiek alátámasztására tekintsük az alábbi magyarázatot:

$$13 - 7 = 13 - (16-9) = 13 - 16 + 9 = (13+9) - 16$$

Világos, hogy a kettes komplementens használata egyszerűsítette a kivonást, hiszen az eredményből 16-ot levonni igen egyszerű - nem kell figyelembe venni a keletkezett átvitelt.

Ha a kivonás eredménye negatív szám lesz, a fenti eljárás akkor is helyes eredményt ad, csupán a különbséget kapjuk kettes komplementensben:

$$7 - 13 = -6$$

0 1 1 1 ₍₂₎	-	7
1 1 0 1 ₍₂₎	-	13
0 0 1 0 ₍₂₎	-	2 (13 egyes komplemente)
0 0 1 1 ₍₂₎	-	3 (13 kettes komplemente)

Adjuk hozzá 7-hez 13-nak a kettes komplementjét:

0 1 1 1 ₍₂₎	-	7
+ 0 0 1 1 ₍₂₎	-	+ 3
1 0 1 0 ₍₂₎		10

Mivel átvitel nem keletkezett, az eredmény kettes komplementben van, tehát negatív szám. Ha abszolút értékére vagyunk kíváncsiak, vissza kell állítani a kettes komplementből közönséges bináris számmá, vagyis kettes komplementét kell képeznünk:

1 0 1 0 ₍₂₎	-	10
0 1 0 1 ₍₂₎	-	5 (10 egyes komplemente)
0 1 1 0 ₍₂₎	-	6 (10 kettes komplemente)

Fentiek alátámasztására - matematikai bizonyítás nélkül - tekintsük az alábbi magyarázatot:

$$7 - 13 = 7 - (16 - 3) = 7 - 16 + 3 = (7+3) - 16$$

A kettes komplement számábrázolás előnyei tehát a következők:

- a kivonás visszavezethető összeadásra, ha a kivonandót kettes komplementben ábrázoljuk,
- nem kell külön figyelni az eredmény előjelét, mert az esetleges átvitel ezt egyértelműen meghatározza.

Az ESZR számítógépek messzemenően kihasználják a kettes komplement számábrázolás adta lehetőségeket.

2.3.4 BCD ÖSSZEADÁS

Binárisan kódolt decimális számok (Binary Coded Decimals, BCD) összeadását a gép közvetlenül elvégzi. A BCD számok a 10 db decimális számjegyet jelenítik meg 4 bináris helyértéken az alábbi összefüggés szerint:

Decimális számjegy	BCD szám
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

A fenti számábrázolásból következik, hogy minden egyes - bináris számként értelmezett - BCD szám értéke megegyezik a neki megfelelő decimális számjegy értékével (például $6_{(10)} = 0110_{(2)}$ stb.), továbbá az ábrázolható szám-tartomány nem 9-ig, hanem 15-ig terjed. Ebből azonban csak a 0-9 közé eső számok jelentenek decimális számjegyeket, 10, 11, ..., 15 már nem írható le egyetlen decimális számjeggyel. A 10-15 számok tehát ebből a szempontból nem érvényesek, ún. tiltott tartományba esnek. Ha mármost összeadás során valamely decimális helyértéken 9-nél nagyobb részösszeg keletkezik, decimális korrekciót kell végrehajtani. Ennek az a lényege, hogy a 10 és 15 közé eső számokat 0 és 6 közé eső számokká alakítja, a keletkezett átvitelt pedig az eggyel nagyobb helyértéken veszi figyelembe. Mindezt egy példán világítjuk meg. Tekintsük az alábbi összeadást:

$$\begin{array}{r} 39 \\ + 65 \\ \hline 104 \end{array}$$

Ugyanezt BCD számokkal felírva az alábbi alakhoz jutunk:

$$\begin{array}{r}
 0011 \cdot \quad 1001 \\
 + 0110 \quad 0101 \\
 \hline
 1001 \quad 1110 \\
 (9) \quad (14) \\
 \\
 \quad + 0110 \\
 \quad 10100 \\
 \quad (4) \\
 \\
 + \quad 1 \\
 \hline
 1010 \\
 (10) \\
 \\
 + 0110 \\
 \hline
 10000 \\
 (1) \quad (0)
 \end{array}$$

Mivel az alacsonyabb helyértéken 14 áll, korrekcióra van szükség. Adjunk hozzá ehhez a számjegyhez 6-ot (0110)!

A keletkezett átvitelt adjuk hozzá az eggyel nagyobb helyértéken képződött részösszeghez!

Ismét korrigálnunk kell 6-tal, mert 10-et kaptunk.

Most már csak megengedett számjegyek szerepelnek. Az eredmény: 104.

A decimális korrekció alap gondolata az, hogy a kettes számrendszerben a 4 helyértéken ábrázolható számjegyek értéke 16-onként periodikusan ismétlődik (0-tól 15-ig). A 10-es számrendszer hasonló értelemben vett periodicitása viszont 10 (0-tól 9-ig tart). A 9-nél nagyobb számok esetében 6 hozzáadásával mintegy "átlépjük" azt a 6 számjegyet, amely nem tartozik a 10-es számrendszerhez, de 4 bináris helyértéken előfordulhat (1010-tól 1111-ig). A keletkezett átvitelt közönséges átvitelként kezeljük.

A decimális korrekciót szükség esetén a számítógép logikai áramkörei automatikusan végrehajtják, így a műveletek helyes eredményt szolgáltatnak. A korrekcióval a felhasználónak nem kell törődnie.

2.3.5 LOGIKAI MŰVELETEK

A logikai műveletek hasonlóak az aritmetikai műveletekhez, de néhány szempontból jelentős eltérést is mutatnak. A logikai műveletek operandusai nem számértékek, hanem ún. logikai változók. Egy logikai változó mindig egy ese-

ményt jelent, amelynek két lehetséges kimenetele van: vagy teljesül (igaz), vagy nem teljesül (hamis) az állítás. Például jelölje az Y logikai változó azt, hogy

szép idő van.

Rendeljük hozzá Y igaz voltához a bináris 1-et, hamis voltához pedig a bináris 0-t. Tehát

ha szép idő van, $Y = 1$ – az állítás igaz, teljesül,

ha nincs szép idő, $Y = 0$ – az állítás hamis, nem teljesül.

Ily módon tehát egy logikai változó két lehetséges értéke 1 bit információ-tartalommal rendelkezik. Látható, hogy a fenti hozzárendelés mellett nem lenne értelme például $Y = 1,7$ -ről beszélni. Figyeljük meg, hogy a logikai változó két lehetséges értéke (1 és 0) milyen jól alkalmazkodik a digitális számítógép alapáramköreinek lehetséges állapotaihoz. A kapcsolat nem véletlen; a számítógép áramköreit is hasonló, de jóval bonyolultabb logikai feltételek, függvények alapján tervezik.

Ha több logikai változót egymással valamilyen logikai kapcsolatba hozunk, logikai függvényhez jutunk. Maga a logikai függvény is egyetlen logikai változónak tekinthető, amelynek értéke (igaz vagy hamis volta) egy vagy több másik logikai változó értékétől függ. Elemi és összetett logikai függvényeket ismerünk. Minden összetett logikai függvény felépíthető elemi logikai függvényekből, bármilyen bonyolult legyen is.

Elemi logikai függvények

Három elemi logikai függvényt értelmezzünk;

- konjunkció - logikai szorzás, un. ÉS kapcsolat,
- diszjunkció - logikai összeadás, un. VAGY kapcsolat,
- negáció - logikai tagadás.

Itt említjük meg az antivalencia logikai függvényt is - ez az un. KIZÁRÓ VAGY kapcsolat -, bár ez már az előző három elemi függvényből épül fel.

Az elemi logikai függvények igen jól szemléltethetők az ún. igazságtáblázat bevezetésével. Ebben feltüntetjük azokat a logikai változókat, amelyektől a logikai függvény értéke függ, továbbá magát a logikai függvényt. A változó értékek kombinációira a függvény értéke azonnal leolvasható.

a) Konjunkció

A logikai függvény értéke 1, ha valamennyi változójának értéke 1. Más szavakkal: egy logikai feltétel (a függvény) akkor és csak akkor teljesül, ha valamennyi változója egyidejűleg igaz értékű.

Például jelöljük Y -nal azt az állítást, hogy kalapot vásárolunk. Jelentse A azt, hogy van elég pénzünk, B pedig azt, hogy találunk megfelelő kalapot. Világos, hogy vásárlásra csak az A és B feltétel egyidejű teljesülése esetén kerülhet sor. Innen az elnevezés: Y igaz, ha A és B egyszerre igaz.

A konjunkció jelölésére az aritmetikai szorzás jelét használjuk:

$$Y = A \cdot B$$

Alább felrajzoltuk a konjunkció igazságtáblázatát. A táblázatot az A és B logikai változókra értelmeztük, de értelemszerűen kiterjeszthető tetszőleges számú változóra is.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Példa: Számítsuk ki a $8E_{(16)} \cdot BF_{(16)}$ logikai függvény értékét!

Megoldás: A két hexadecimális szám 8-8 független logikai változót képvisel.

Ezeket bitenként ÉS kapcsolatba hozva kapjuk az eredményt:

$$\begin{array}{rcl} 8E & = & 1\ 0\ 0\ 0 \quad 1\ 1\ 1\ 0 \\ BF & = & 1\ 0\ 1\ 1 \quad 1\ 1\ 1\ 1 \\ \hline & & 1\ 0\ 0\ 0 \quad 1\ 1\ 1\ 0 \quad (=8E_{16}) \end{array}$$

b) Diszjunkció

A logikai függvény értéke 1, ha legalább egy változójának értéke 1. Más szavakkal: egy logikai feltétel (a függvény) akkor teljesül, ha legalább egy változója igaz értékű.

Például jelöljük X -szel azt az állítást, hogy színházba megyünk. Jelentse A azt, hogy sikerült jegyet vásárolni, B pedig azt, hogy kapunk tiszteletjegyet. Látható, hogy akár az A , akár a B feltétel teljesül, nincs akadálya X teljesülésének. Ugyanakkor viszont nincs szükség arra, hogy mind A , mind B igaz legyen. Innen az elnevezés: X igaz, ha A vagy B igaz.

A diszjunkció jelölésére az aritmetikai összeadás jelét használjuk:

$$X = A + B$$

Alább felrajzoltuk a logikai diszjunkció igazságtáblázatát. Az elmondottak értelemszerűen igazak kettőnél több változóra is.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Példa: Számítsuk ki a $8E_{(16)} + BF_{(16)}$ logikai függvény értékét!

Megoldás: Az előző példához hasonlóan, bitenként végezzük el a műveletet.

$$\begin{array}{rcl}
 8E & = & 1\ 0\ 0\ 0 \quad 1\ 1\ 1\ 0 \\
 BF & = & 1\ 0\ 1\ 1 \quad 1\ 1\ 1\ 1 \\
 \hline
 & & 1\ 0\ 1\ 1 \quad 1\ 1\ 1\ 1 \quad (=BF_{16})
 \end{array}$$

c) Negáció

A logikai függvény értéke 1, ha változójának értéke 0, és a függvény értéke 0, ha változója 1. Ez az elemi logikai függvény csak egy változóra értelmezett, és a változó invertálásával egyenértékű. Jól jegyezzük meg, hogy egy logikai változó és negáltja egyidejűleg sohasem lehet 1 vagy 0! Alább felrajzoltuk a negáció igazságtábláját. A negálás jelölésére a felülhuzást használjuk: $\overline{A}$.

A	$\overline{A}$
0	1
1	0

Példa: Számítsuk ki $8E_{(16)}$ negáltját!

Megoldás: A negálást bitenként végezve kapjuk az eredményt:

$$\begin{array}{rcl}
 8E & = & 1\ 0\ 0\ 0 \quad 1\ 1\ 1\ 0 \\
 \overline{8E} & = & 0\ 1\ 1\ 1 \quad 0\ 0\ 0\ 1 \quad (=71_{16})
 \end{array}$$

Megjegyezzük, hogy amennyiben a negálni kívánt mennyiséget bináris számnak fogjuk fel, a negálás a szám egyes komplementjét szolgáltatja.

d) Antivalencia

A logikai függvény értéke 1, ha változóinak értéke egyidejűleg különböző. Jelölése: $Z = A \oplus B$. Az antivalencia már nem szorosan vett elemi logikai függvény, ugyanis az előző három elemi függvényből felépíthető. Tekintsük az alábbi igazságtáblázatot:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Látható, hogy $Z = A \cdot \bar{B} + \bar{A} \cdot B$ kifejezés éppen az antivalencia igazságtáblázát írja le.

Példa: Számítsuk ki a $8E_{(16)} \oplus BF_{(16)}$ logikai függvény értékét!

Megoldás: A műveletet bitenként elvégezve az alábbi eredményhez jutunk:

$$\begin{array}{r}
 8E = \quad 1 \ 0 \ 0 \ 0 \quad 1 \ 1 \ 1 \ 0 \\
 BF = \quad 1 \ 0 \ 1 \ 1 \quad 1 \ 1 \ 1 \ 1 \\
 \hline
 \quad 0 \ 0 \ 1 \ 1 \quad 0 \ 0 \ 0 \ 1 \quad (=31_{16})
 \end{array}$$

Az antivalenciát itt két változóra mutattuk be, de általánosítható tetszőleges számú változóra is, erre azonban nem térünk ki.

3. Adatközlés a számítógéppel

3.1 INFORMÁCIÓK ÉS ADATOK

A számítástechnika történetének és az ESZR számítógépeknek az áttekintése során megvilágítottuk, hogy az elektronikus számítógépek is bizonyos mennyiségekkel végeznek műveleteket, és ezeket a mennyiségeket az előzőekben információ, illetve adat névvel jelöltük. Szeretnénk pontosabban is megfogalmazni, mit értünk információn, illetve adaton a számítógép szempontjából.

Információnak tekintjük azt az ismeretet, amely valamely kérdésben fennálló bizonytalanságunkat, tájékoztatlanságunkat szünteti meg. Például, ha a gyalogos bizonytalankodik, hogy a közlekedési szabályok értelmében áthaladhat-e az uttesten, a jelzőlámpa fénye - mint információ - megszünteti e bizonytalanságát: tájékoztatja az áthaladás lehetőségéről, vagy megtiltja számára az átkelést. Az információt a gyalogos számára éppen a fény színe hordozza: piros fény = tilos az átkelés, zöld fény = szabad áthaladás. Ilyenkor azt mondjuk, hogy a fény színének a gyalogos számára információtartalma van. Ugyanígy információnak tekintjük egy vállalat dolgozóinak személyi adatait stb.

Ha már most a vállalati dolgozók személyi adatait például törzslapokra írjuk, akkor az információt rögzítettük, további felhasználás céljára alkalmassá tettük. Adatnak tekintjük tehát a rögzített információt. Világos, hogy az adat az információ egyik megjelenési formája. Ugyanis adatoknak minősülnek például egy kérdőív kérdéseire adott válaszok, a számlák tételei stb.

A vállalati dolgozók törzslapjait, vagy magukat a kérdőíveket, amelyekre az adatokat rögzítettük, az adatmegjelenítés eszközeinek, adathordozóknak nevezzük. Adathordozó lehet valamilyen űrlap, bizonylat, mikrofilm stb., de a számítógép-technikában gyakoribb adathordozó a lyukkártya, lyukszalag, mágnesszalag stb.

Az adatmegjelenítés hagyományos módjai az adatok feldolgozásakor azonban számos nehézséget okoznak. Gondoljunk csak a vállalati dolgozók törzslapjainak kezelésére, ahol számos személynév, helységnév, évszám stb. fordul elő, mint rögzített információ. A többféle jellegű adat (betű, számjegy, írásjelek stb.) egységes rendszerű megjelenítésének céljából vezették be a kódolást. Megállapodás szerint valamennyi betű-, számjegy- és írásjelfajtához adott bináris számjegysorozatot rendelünk hozzá, ezzel biztosítjuk a kölcsönösen egyértelmű megfeleltetést. Például a nagy A betűhöz bizonyos számjegysorozatot rendelünk, és e számjegysorozat mindig és csakis a nagy A betűre lesz jellemző. Így járunk el a többi betű, számjegy és írásjel esetében is. A fenti műveletet kódolásnak nevezzük.

Kódolás = adott szimbólumhoz (betű, számjegy, írásjel stb.) adott szimbólumsorozat (esetünkben számjegysorozat) hozzárendelése.

Magukhoz a számjegyekhez is adott számjegysorozatot rendelünk, így egy kódolt számjegy eredeti számjegy-megfelelőjétől eltérő formában jelenik meg. A kódolás eredményeképpen nyert számjegysorozatot kódnak nevezzük.

Ha kódolt formájú adatok állnak rendelkezésünkre, és azokból a megfelelő eredeti adatokat vissza akarjuk állítani, akkor a dekódolás műveletét végezzük.

Dekódolás = adott szimbólumhoz rendelt adott szimbólumsorozat (kód) alapján az eredeti szimbólum visszaállítása.

Mint látható, a dekódolás a kódolással ellentétes művelet: a kódolt információból a kódolás alapjául szolgáltat információt szolgáltatja. Az egész kódolás-dekódolás szemléletes példája a titkosítás, ahol minden jelnek megfelel valamely más jel vagy jelsorozat, s az eredeti jeleket a titkosítás megfejtése után állíthatjuk elő.

Az ábécé betűit, a számjegyeket, valamint az írásjeleket és speciális jeleket (például százalékjel stb.) összefoglaló néven karaktereknek nevezzük. Az ESZR-ben használt adathordozók és maguk a számítógépek is kódolt formában tartalmazzák, illetve dolgozzák fel az egyes karaktereket; a kód fajtája adathordozónként más és más lehet. A használatos kódrendszereket és ezen belül a leggyakrabban előforduló karakterek kódjait a 3. táblázatban foglaltuk össze. A "kézi" dekódolás például ennek a kódtáblázatnak segítségével végezhető el; ekkor adott kód esetén egyszerűen visszakeressük a neki megfelelő karaktert. A számítógépek a dekódolást, átkódolást (áttérést egyik kódrendszerről a másikra) automatikusan elvégzik.

3.2 ADATHORDOZÓK ÉS KÓDRENDSZEREK

Tekintsük át mindezek ismeretében az ESZR számítógépeken használatos adathordozókat, kódrendszereket, kiemelve sajátos tulajdonságaikat és javasolt alkalmazási területüket.

3.2.1 LYUKKÁRTYÁS ADATRÖGZÍTÉS

a) Lyukkártya

A lyukkártya mint adathordozó szerte a világon igen elterjedt. Előnye, hogy téves lyukasztás vagy adatmódosítás esetén a régi kártya a kártyacsomagból (deck) könnyen kiemelhető és újjal helyettesíthető. Ezzel szemben tárolási helyigénye viszonylag nagy, a levegő páratartalmának változásaira érzékeny, könnyen vetemedik. Mint hagyományos adathordozó, valamennyi ESZR számítógépen használatos.

Az ESZR szabványosított lyukkártyáját a 3. ábra szemlélteti.

A lyukkártya kemény papírból készül, és egyik sarkán megcsonkítják. Erre azért van szükség, hogy a kártyát mindig csak egy meghatározott helyzetben lehessen a lyukkártyás berendezésbe helyezni, tehát kizárjuk a téves pozicionál-



3. ábra. 80 oszlopos szabványos lyukkártya

lást. A lyukkártyán az adatok 80 oszlopban és 12 sorban helyezkednek el, lyukasztási kombinációk formájában. Összesen tehát 960 lyukasztási pozíció van, ennyi az egy kártyán elhelyezhető lyukasztások maximális száma. A karakterek kódjai az oszlopok mentén, keresztben sorakoznak, így egy kártyán maximálisan 80 karakter ábrázolható.

b) Kártyakód

A lyukkártyán az adatokat un. Hollerith-kártyakódban ábrázoljuk. A kód szempontjából a lyukkártya első három sorát (felülről lefelé a 12., 11. és 0. sort) zónarésznek, a többi 9 sort a kód információs részének tekintjük. A számítógépek által használt teljes karakterkészlet több csoportra oszlik. A zónarész feladata éppen a kívánt csoport kijelölése, az információs rész pedig meghatározza az adott karaktert a kijelölt csoporton belül. Alább feltüntetjük a gyakrabban előforduló karaktercsoportokat:

- az angol ábécé nagybetűi A-től I-ig,
- az angol ábécé nagybetűi J-től R-ig,
- az angol ábécé nagybetűi S-től Z-ig,
- a tízes számrendszer számjegyei 0-től 9-ig,
- egyéb karakterek (kisbetűk, írásjelek stb.).

Ennek alapján a Hollerith-kártyakód felépítése a következő:

- a nagybetűknek megfelel egy zóna- és egy információs lyukasztás, tehát minden nagybetűt két lyukasztással ábrázolunk;
- a számjegyeknek megfelel egy információs lyukasztás annyiadik sorban, amennyi az illető számjegy értéke. A számjegyek kódja tehát nem tartalmaz zónalyukasztást, ez alól csak a 0 kivétel;
- a speciális karaktereknek egyéb, a fentiektől eltérő lyukasztási kombináció felel meg. (Nem nyomtatható karaktereknek is van lyukkártyakódja.)

Mivel az angol ábécé 26 betűből áll, az első és második csoportba 9-9, a harmadikba 8 betű került. A 3. ábrán látható, hogy az egyes betűcsoportokon belül az információs lyukasztás annyiadik sorban van, ahányadik az illető betű az adott csoportban. Kivétel ez alól a harmadik csoport; itt a csoport első betűjének (S) információs lyukasztása a második sorban van. Ez azzal függ össze, hogy a harmadik csoportba csak 8 betű tartozik.

3.2.2 LYUKSZALAGOS ADATRÖGZÍTÉS

a) Lyukszalag

A lyukszalag elvileg tetszőleges hosszúságú papírszalag, amelyre a karakterek kódjait keresztirányban rögzítjük lyukasztási kombinációk formájában.

Lyukszalag alkalmazása főleg ott lehet célszerű, ahol valamely folyamat regisztrálása során a szemléletesebb diagram mellett - "melléktermékként" - a további feldolgozásra sokkal inkább alkalmas lyukszalag is keletkezik. Ez a lyukszalag a megfelelő berendezésekben közvetlenül és automatikusan feldol-



4. ábra. Nyolccsatornás lyukszalag

gozható. Elterjedt a lyukszalag még a tudományos-műszaki számításoknál, ahol viszonylag kevés adattal kell bonyolult matematikai műveleteket végezni. Hátránya, hogy a hibás szalag vagy egyáltalán nem, vagy csak nehézkesen javítható (például a hibás karakter leragasztásával és ujralyukasztásával). A szalag hossza tetszőleges, de általában 10 cm és 300 m közé esik.

b) Lyukszalagkód

A lyukszalagkódok kidolgozása nemzetközi szabványajánlások alapján történt, és ehhez igazodik az ESZR is. A 4. ábra egy un. nyolccsatornás lyukszalag elvi felépítését szemlélteti.

Az ESZR-ben lehetőség van öt-, hat-, hét- és nyolccsatornás lyukszalag olvasására, valamint öt- és nyolccsatornás lyukszalag lyukasztására.

Az ábrából látható, hogy a szalagot fizikailag két részre osztották: a "felezővonal" nem középen helyezkedik el. Tőle egyik irányban maximum 3, másik irányban maximum 5 lyukasztás helyezhető el karakterenként. Erre azért volt szükség, hogy a szalagot ne lehessen fordítva, tévesen behelyezni a lyukszalagolvasóba.

Információtartalma csak a nagyobb, un. információs lyukasztásoknak van; egy lyukasztás logikai 1-nek felel meg, a lyukasztás hiánya logikai 0-t képvisel. A szalagot kettéosztó kisebb lyukasztások a mechanikus szalagtovábbításra szolgálnak: egy fogaskerék ezekbe kapaszkodva mozgatja a szalagot. Ezek a lyukasztások alkotják a szinkroncsatornát, az információs lyukasztások pedig hosszirányban információs csatornákat képeznek. Egy-egy információs csatorna mentén levő lyukasztási kombináció tehát nem "értelmes" információ!

3.2.3 MÁGNESSZALAGOS ADATRÖGZÍTÉS

a) Mágnesszalag

A mágnesszalag a magnószalaghoz hasonló, de annál szélesebb, 12,7 mm (1/2 inch) széles celluloid szalag, amelynek egyik felét mágnesezhető réteggel vonják be a gyártás során. A szalag szokásos hossza kb. 750 m. Felcsévéeléskor a mágnesezhető réteg a belső oldalra kerül, így is védjük a külső sérülé-

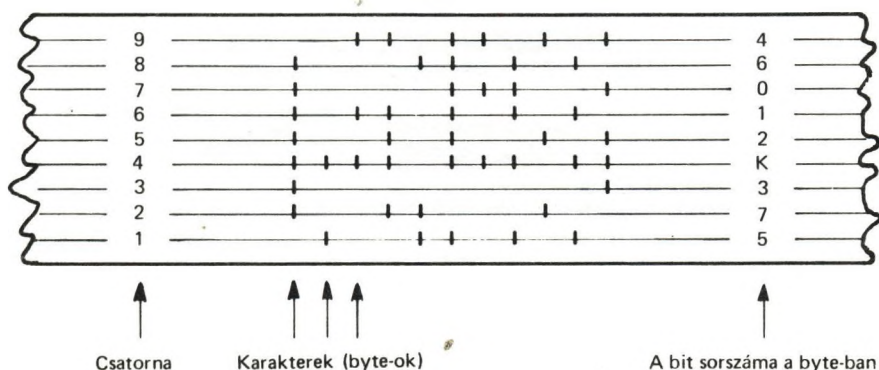
sektől. A mágnesszalag a programozó szempontjából logikailag végtelen nagy mennyiségű adat tárolására képes, hiszen valamely teleirt szalaghoz logikailag további üres szalagokat csatlakoztathatunk, s azokon folytathatjuk az adattárolást. Ennélfogva nincs értelme a mágnesszalag adattároló kapacitásáról beszélni.

b) Mágnesszalagkód

A mágnesszalagon az információ a lyukszalaghoz hasonló módon helyezkedik el: egy karakter kódját (1 byte) a szalag hosszirányára merőlegesen rögzítik. A logikai 1 és 0 állapotot azonban nem lyukasztás, hanem a mágnesezhető rétegben létrehozott rendezett mágnesezettségű állapot képviseli. Az 5. ábra szemlélteti az információ elhelyezkedését a szalagon.

A szalag 9-csatornás; 8 csatorna az információs bitek, a 9. csatorna pedig az ellenőrző bit (K) tárolására szolgál. Figyeljük meg, hogy a csatornák számozása nem esik egybe a bitek sorszámozásával a byte-on belül. A kódolásra vonatkozó egyéb tudnivalókat a mágneslemez kódolásával együtt az 54. oldalon ismertetjük.

A mágnesszalagon használt információsűrűség 8-63 karakter/mm. Az elérhető írássűrűséget a szalag minősége korlátozza; a felhasználó erre vonatkozó utasítással az adatsűrűséget a kívánt értékre változtathatja. Átlagos minőségű szalagoknál az információsűrűség 32 karakter/mm.

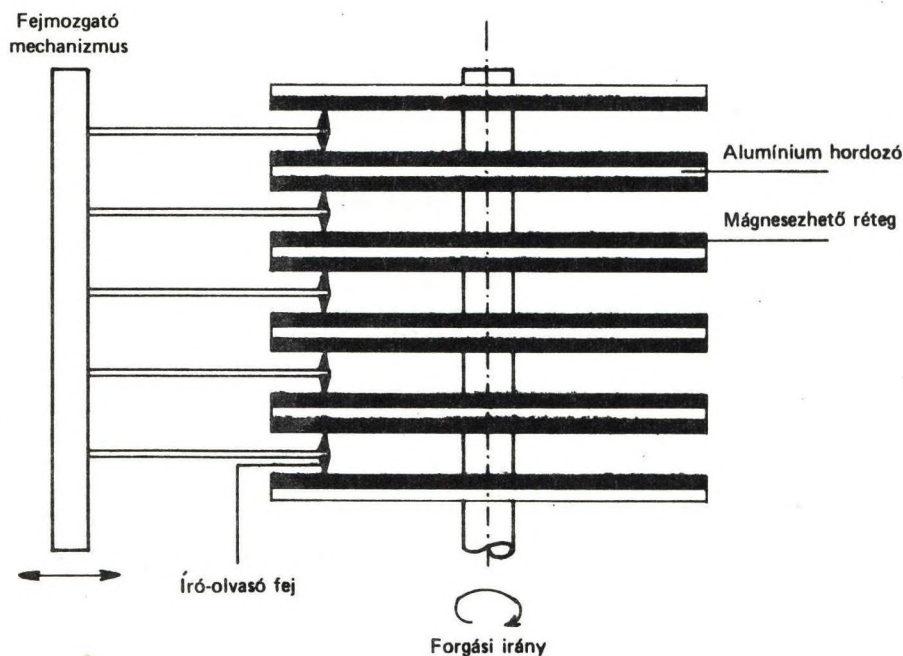


5. ábra. Az információ elhelyezkedése a mágnesszalagon

3.2.4 ADATRÖGZÍTÉS MÁGNESLEMEZEN

a) Mágneslemez (disc)

A mágneslemez igen pontosan megmunkált, kör alakú alumínium lemez, melyet mindkét oldalán mágnesezhető réteggel vonnak be a gyártás során. Az adattároló kapacitás növelése érdekében - az ESZR-ben is - általában nem egy-egy mágneslemezt, hanem több lemezből álló lemezköteget használnak, amely 6 db mágneslemezről áll. A lemezek közös tengelyhez rögzítettek, és működés közben együtt forognak nagy sebességgel. A lemezköteg legfelső és legalsó oldala nem kap mágnesezhető bevonatot, ezekre a felületekre információ nem rögzíthető. Mindez azért indokolt, mert a külső felületek könnyebben szennyeződnek, karcolódnak, és ez az információ sérülését eredményezhetné. A belső lemezoldalak is igen érzékenyek mechanikai hatásokra, ezért kézzel sohasem szabad belenyulni egy lemezköteg belsejébe. A használat közbeni tisztítást a mágneslemezegység automatikusan elvégzi. A 6. ábrán bemutatjuk egy hatlemezes disc-köteg vázlatos rajzát.



6. ábra. Mágneslemezcsomag felépítése

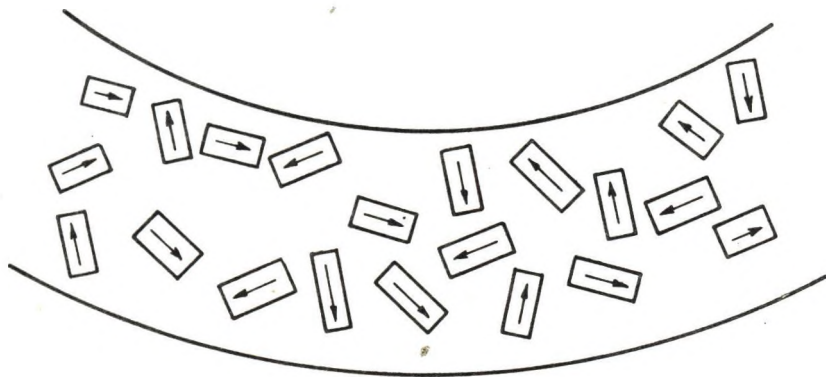
A mágneslemezkeg adattároló képessége igen nagy: kb. 7 250 000 byte. Éppen ezért elsősorban nem input-output célokra használják, hanem az operatív tár kapacitásának bővítésére, un. háttértárolóként. A mágneslemez elengedhetetlen adattároló az ESZR számítógépeknél; rajta foglal helyet többek között az egész gépet üzemeltető programrendszer, az operációs rendszer is.

b) Mágneslemez kód

A lemez felületét is úgy képzelhetjük el, mintha rendkívül sok apró elemi mágnesből állna, amelyek azonban közönséges körülmények között kifelé nem mutatnak mágnesességet, mert É-D mágneses tengelyeik rendezetlenek (lásd 7. ábra).

Az adatrögzítés a következőképpen történik: minden egyes mágnesezhető réteggel bevont lemezoldalt 203 koncentrikus kör alakú sávra osztunk (track). Valamely sáv információtartalmát úgy állítjuk elő, hogy az adott sáv mágnesezhető bevonatában villamos impulzusok segítségével rendezett mágneses állapotot hozunk létre.

Együttal szinkronizáló impulzusokat adunk a lemezre, amelyek meghatározott időközönként követik egymást. Ha a szinkronizáló impulzusok között egyéb impulzus is előfordul, akkor ez logikai 1 jelenlétére utal. Ha két szinkronizáló impulzus között nincs egyéb impulzus, ez logikai 0-t jelöl. Egy sávot tehát byte-onként töltünk ki az információkkal.



7. ábra. Mágnesezettségi irányok rendezetlen mágneses állapotnál

Az ESZR információcserére és adattárolásra szolgáló bináris kódja (EBCDIC) kódtáblázata

Zónacsoport									
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	0	0
2	0	0	1	1	0	0	1	1	0
3	0	1	0	1	0	1	0	1	0

Numerikus csoport									
0	1	2	3	4	5	6	7	8	9
0	ПУС	АР1	Д00	Д16	Проб	&	-	ц	й
1	НЗ	СУ1	Д01	Д17		/		а	ј
2	НТ	СУ2	Д02	СИН				б	к
3	КТ	СУ3	Д03	Д19				с	l
4	Д28	Д29	Д04	Д20				д	м
5	ГТ	Д05	ПС	Д21				е	п
6	Д06	ВШ	КВ	Д22			ю	ф	о
7	ЗВ	Д07	АР2	КП				г	р
8	Д23	АН	Д08	Д24			б	х	у
9	Д13	КН	Д09	Д25			▼	і	г
А	Д14	Д18	Д10	Д26	[	]	:	д	к
В	ВТ	Д15	Д11	Д27	.	×	≠		л
С	ПФ	РФ	Д12	СТП	<	*	@	ф	м
Д	ВК	РГ	КТМ	НЕТ	(	)	—	г	н
Е	РУС	РЗ	ДА	Д30	+	;	=		ж
Ф	ЛАТ	РЭ	ЗВ	ЗМ	і	^	"	л	п
							?		б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш
									з
									ц
									ч
									б
									л
									я
									ш

Általában 200 sáv használható információörögzítés céljára, a maradék sávok tartalékot képeznek (például ide menthető ki egy sérült sáv tartalma stb.).

A mágneses adathordozók az információt közvetlenül a gép belső kódjában tárolják, tehát az operatív tárból kivitt információ átkódolás nélkül kerül a mágnesszalagra, illetve mágneslemezre. A mágneses adathordozók tehát az EBCDIC kódot használják (Extended Binary Coded Decimal Interchange Code - kiterjesztett binárisan kódolt decimális kód), ritkábban az ASCII kódot (American Standard Code of Information Interchange). Az ESZR-ben általános az EBCDIC (DEKOI) kód használata. A kódok felépítését, bináris és hexadecimális leírását a 3. táblázat tartalmazza.

Utmutató az EBCDIC kódtáblázat használatához

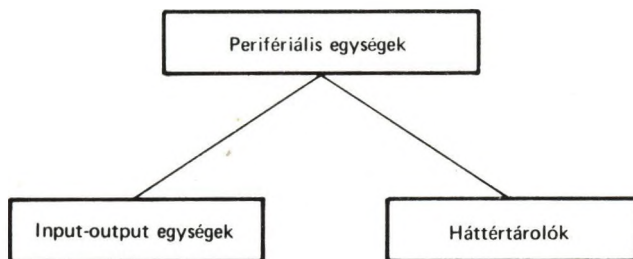
A kódtáblázat egyaránt alkalmas valamely karakter EBCDIC kódjának megállapítására és adott EBCDIC kód alapján a megfelelő karakter visszakeresésére. A táblázat oszlopai a kód zónarészét jelentik (0-3. bit), sorai pedig a kód numerikus részét (4-7. bit). A lehetséges zóna- és numerikus részek értékét binárisan és hexadecimálisan is feltüntettük a könnyebb áttekinthetőség kedvéért. Minden egyes karakter valamely sor és oszlop metszéspontjában található. Kódolásnál az alábbi módon járunk el: megkeressük az illető karaktert a táblázatban, és a szóban forgó oszlop, illetve sor szélén leolvassuk hozzá a kód zóna-, illetve numerikus részét. Például "M" kódja $1101\ 0100_2$ ($D4_{16}$) stb. Dekódolásnál a kód zónarésze és numerikus része kijelöl egy oszlopot, illetve sort a táblázatban; ezek metszéspontjában található a keresett karakter. Például a $1100\ 1000_2$ ($C8_{16}$) kód a "H" betűt jelenti stb. Felhívjuk a figyelmet arra, hogy a táblázatban nemcsak nyomtatható karakterek, hanem ún. vezérlőkarakterek kódjai is szerepelnek. Ezek a kódok a különböző perifériális egységek működését vezérlik; használatukra itt nem térünk ki.

4. ESZR perifériális egységek

4.1 A PERIFÉRIÁK ÉS A SZÁMÍTÓGÉP

A perifériális egységek (röviden: perifériák) feladata a felhasználó és a számítógép központi egysége közötti kapcsolatteremtés. Mivel a felhasználó közvetlenül nem fér hozzá a központi egységhez vagy az operatív tárhoz, olyan közvetítő elemeket kellett létrehozni, amelyek elvégzik a központi egységgel folytatandó információcsere feladatát. Ezek ugyanakkor átalakítják a bevitt adatokat a gép belső kódjának megfelelően, s a kihozott információt az ember által értelmezhető formában jelenítik meg. Megjegyezzük, hogy az operátori kezelőpultból ugyan közvetlenül is beavatkozhatunk a központi egység működésébe, de a központi egység magát a kezelőpultot is perifériának tekinti.

A perifériákat funkciójuk szerint két csoportra oszthatjuk:



Az input-output egységek feladata a központi egység és a felhasználó közötti adatforgalom lebonyolítása, a háttértárolók az operatív tár kapacitását bővítik. Tulajdonképpen ezek is input-output műveleteket végeznek a központi egység szempontjából, de másodlagos tárolóként használjuk őket. Ezzel szemben az input-output eszközöket sohasem használjuk adattárolásra. A fenti felosztás indokoltsága a perifériák megismerése során nyilvánvalóvá válik.

Az ESZR perifériakészlete az alábbi kommersz perifériákat foglalja magában:

Input-output egységek:

- lyukkártyaolvasó,
- lyukkártyalyukasztó,
- lyukszalagolvasó,
- lyukszalaglyukasztó,
- sornyomtató,
- konzolirógép.

Háttértárolók:

- mágnesszalagegység,
- mágneslemezegység.

Használatosak még egyéb perifériák is (rajzgép, display, optikai bizonylatolvasó, távadatfeldolgozási berendezések stb.); ezekre itt nem térünk ki.

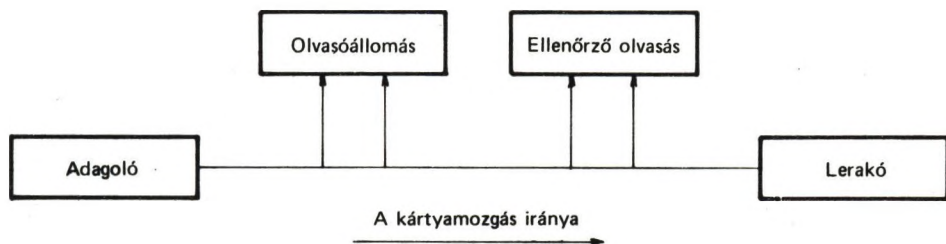
A perifériák működését közvetve a csatornák, közvetlenül pedig a perifériavezérlő egységek irányítják. Minden perifériavezérlő egység szabványos illesztéssel csatlakozik valamelyik csatornához; ez az ún. standard interface. Az interface tulajdonképpen egy kábelköteg, amely adatbyte-ok és vezérlőjelek átvitelére szolgál. A csatorna és a megfelelő perifériavezérlő egység ezen vezérlőjelek (villamos impulzusok, illetve feszültség szintek) segítségével tartja a kapcsolatot az input-output művelet alatt.

4.2 PERIFÉRIATÍPUSOK

A továbbiakban ismertetett ESZR perifériák adatai mindig a megadott típusra vonatkoznak, így más berendezésekben ezektől való eltérések természetesen lehetségesek.

4.2.1 LYUKKÁRTYAOLVASÓ (ESZ-6012)

A lyukkártyaolvasó feladata az, hogy valamilyen lyukasztási kombináció formájában tárolt adatokat villamos jelekké alakítsa át, és azokat a számítógép belső kódjában ábrázolva a központi egység felé továbbítsa. A kártyaolvasás folyamata a 8. ábrán látható.

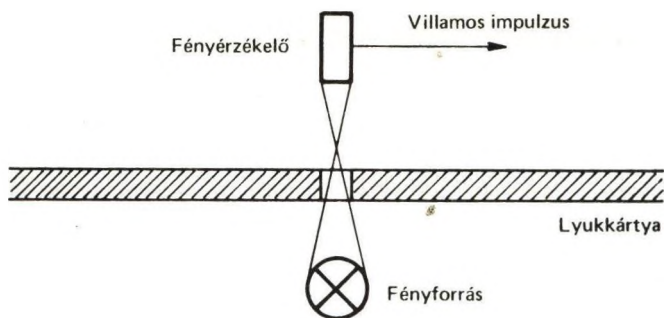


8. ábra. A lyukkártya olvasásának elve

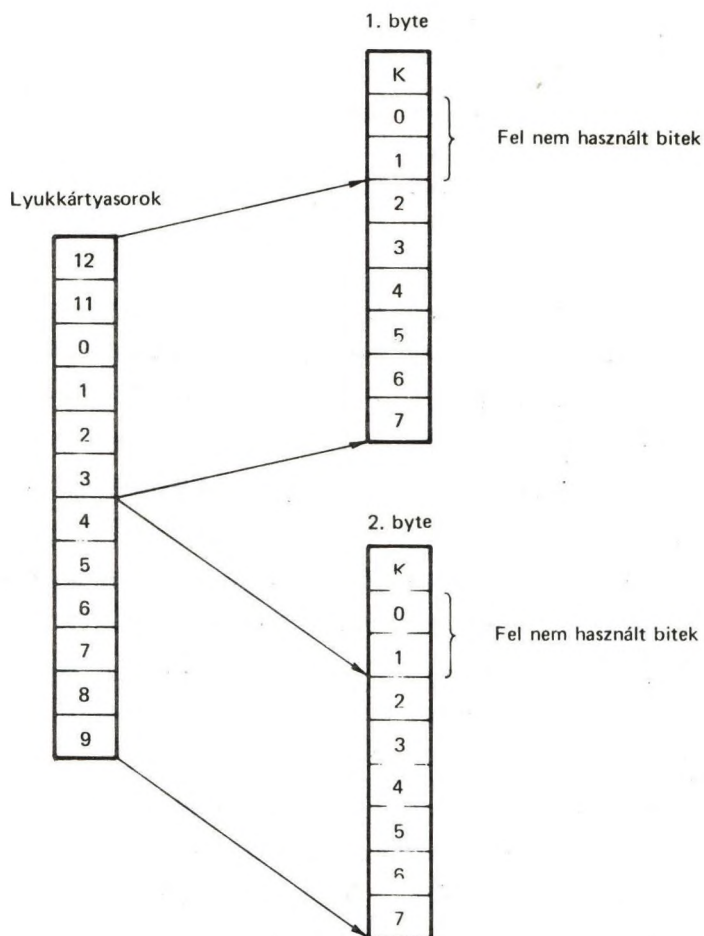
A lyukkártyaolvasó integrált áramkörös kivitelű, 80 és 45 oszlopos lyukkártya olvasására alkalmas. Az olvasandó kártyának megfelelő üzemmódot (80, illetve 45 oszlopos lyukkártya) a berendezés kezelőpultján állíthatjuk be. Mivel a 80 oszlopos lyukkártya az elterjedt, 45 oszlopossal itt nem foglalkozunk.

Az olvasás fotoelektromos úton megy végbe. Az olvasóállomáson a kártya minden sorához tartozik egy-egy izzólámpa és az ahhoz méretezett fényérzékelő. Ahol a kártyán lyukasztás van, az izzó fénye megvilágítja a fényérzékelő elemet, és abban villamos impulzust kelt. Oszloponként annyi sorban keletkezik impulzus, ahány lyukasztás van az adott oszlopban. Így annak alapján, hogy mely fényérzékelőkben keletkezett egyszerre impulzus, azonosítható az olvasott lyukkombináció (lásd a 9. ábrát).

A berendezés minden egyes kártya adagolása előtt ellenőrzi, hogy üzemképesek-e az olvasócsatornák, nem szorult-e be kártya az olvasás során, működőképes-e az adagoló mechanizmus, s végül: nem került-e egyszerre két kártya az olvasóállomásra. Csak a fenti feltételek teljesülése esetén adagol újabb kártyát.



9. ábra. Fotoelektromos olvasási elv



10. ábra. Általános lyukasztási kombináció olvasása

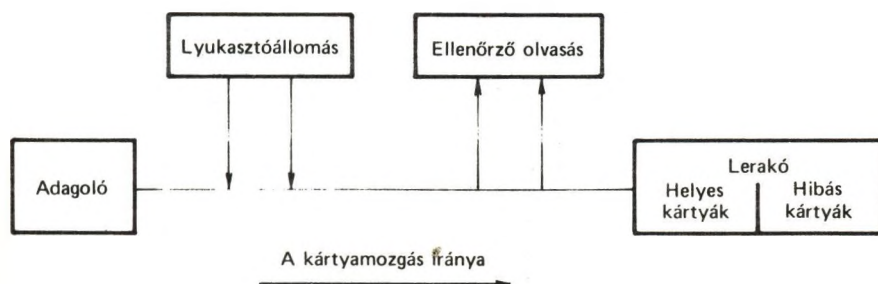
A lyukkártyán az adatok Hollerith-kódban vagy tetszőleges más kódban lehetnek. A kártyaolvasó a Hollerith-kódban ábrázolt információt átalakítja EBCDIC kódba, és ilyen formában továbbítja a központi egység felé. A felhasználónak lehetősége van arra, hogy a Hollerith-kódban lyukasztott információt a berendezéssel ellenőriztesse (ellenőrző olvasás).

A Hollerith-kódban való információ ábrázolásánál a berendezés a következő korlátozást érvényesíti: az 1-7. sorban karakterenként csak egy lyukasztás lehet. A többi sorban tetszőleges lyukkombináció használható. Az ellenőrző olvasás éppen abban áll, hogy felderíti az esetleges, a Hollerith-kód szempontjából érvénytelen lyukkombinációkat, és azokat hibásnak minősíti. Bármely más kódban levő információt oszloponként két byte-ra osztva továbbít, átkódolás nélkül, a 10. ábra szerint.

A kártyaolvasás maximális sebessége 500 kártya/perc; az adagoló és lerakó befogadóképessége 1000-1000 kártya.

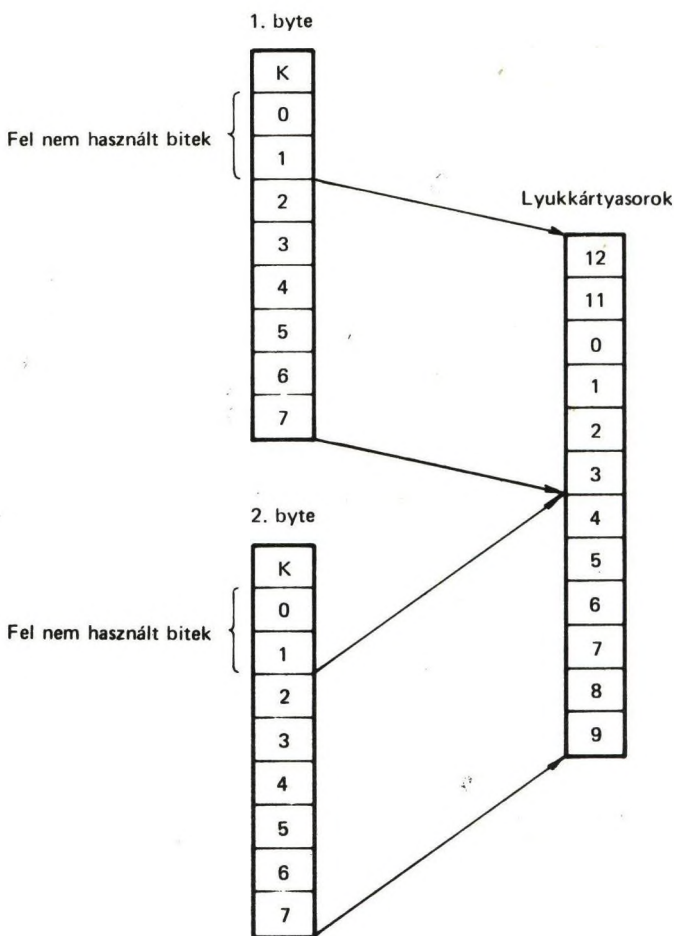
4.2.2 LYUKKÁRTYALYUKASZTÓ (ESZ-7010)

A lyukkártyalyukasztó feladata az, hogy a számítógépből villamos jelek formájában érkező információt lyukasztási kombinációvá alakítsa át, és lyukkártyán megjelenítse. A kártyalyukasztás folyamata a 11. ábrán látható. A kártyalyukasztó 80 oszlopos lyukkártyával dolgozik. A sorok irányában, tehát egy lépésben egy sort lyukaszt. Ily módon egy kártya lyukasztása maximálisan 12 lépést vesz igénybe. A lyukasztást elektromágnesekkel vezérelt kések végzik; a lyukasztás ideje alatt a kártya áll. Külön ellenőrző berendezés gondoskodik arról, hogy a lyukasztásnál a papírhulladék-gyűjtő a helyén legyen.



11. ábra. A lyukkártyalyukasztás elve

Mivel a lyukasztás nem oszloponként, tehát nem "értelmes" információ alapján történik, a számítógépből érkező adatokból össze kell állítani egy kártyának megfelelő mennyiségűt, s csak azután lehet hozzáférni a lyukasztáshoz. Erre az adatgyűjtésre és átmeneti adattárolásra szolgál a kártyalyukasztó helyi puffertárolója, amely 256 byte kapacitású. A helyi puffertárban mindaddig benne tartózkodik az éppen lyukasztandó kártya információtartalma, amíg az ellenőrző olvasás be nem fejeződött. Lyukasztás előtt a berendezés átalakítja a számítógépből érkező, EBCDIC kódban ábrázolt információt Hollerith-kódba. Nem Hollerith-kódu lyukasztás esetén két byte felel meg egy lyukkombinációnak a 12. ábra szerint.



12. ábra. Általános lyukasztási kombináció származtatása

Az ellenőrző olvasás során megszámloljuk a helyi pufferben levő lyukasztandó pozíciókat és a ténylegesen lyukasztott pozíciókat minden egyes kártyán. Ha ezek egyező számúak, a kártya helyesnek minősül, és a "helyes kártyák" lerakóba kerül. Eltérő számú lyukasztás esetén a kártya hibás, és a "hibás kártyák" lerakóba kerül. (A javítási lehetőségekre itt nem térünk ki.) Az ellenőrző olvasás fotoelektromos uton történik, a már megismert módon (lásd lyukkártyaolvasó).

A lyukkártyalyukasztó maximális sebessége több, mint 100 kártya/perc. Az adagoló kapacitása minimálisan 700 kártya, a lerakóké pedig minimálisan 500-500 kártya. A berendezés automatikusan ellenőrzi, hogy az adagolóból nem fogyott-e el a kártya, illetve nincs-e tulcsordulás valamelyik lerakóban.

A kártyaolvasó az egyik legelterjedtebb kommersz bemeneti periféria. A lyukkártyát mindenütt előszeretettel alkalmazzák előnyei miatt. A kártyalyukasztót ott célszerű használni, ahol valamely számítás eredményei további feldolgozás kiindulási adatai lesznek. Ilyen feldolgozás lehet például az adatfeldolgozás lyukkártyás gépen.

4.2.3 LYUKSZALAGOLVASÓ (ESZ-6022)

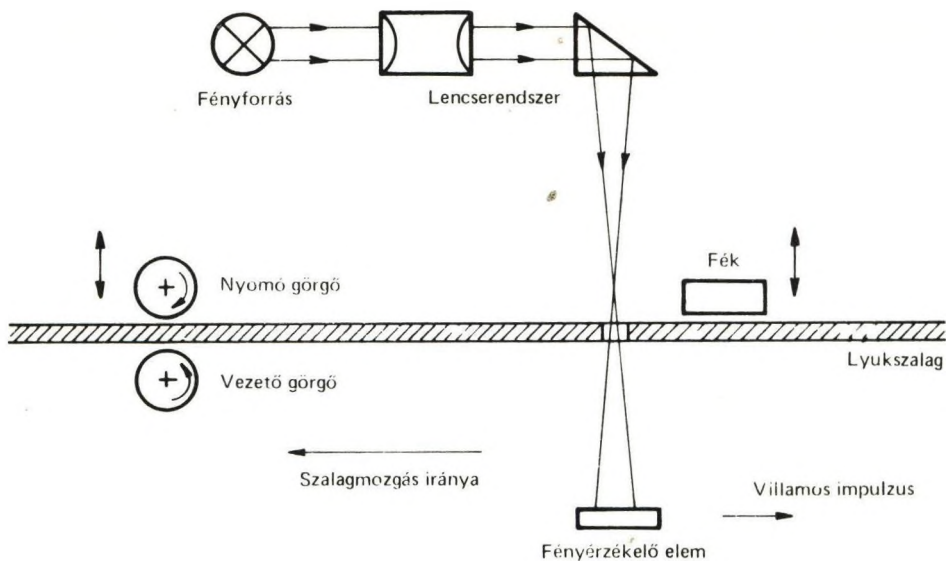
A lyukszalagolvasó berendezés feladata az, hogy a lyukszalagra rögzített információt villamos jelekké alakítsa át, és a számítógép központi egysége felé továbbítsa. Az ESZ-6022 lyukszalagolvasó öt-, hat-, hét- és nyolcsatornás lyukszalag olvasására alkalmas.

Kétféle olvasási üzemmódban dolgozhat:

- másoló, itt az olvasott lyukszalagkódot minden változtatás nélkül továbbítja;
- átkódoló; kizárólag nyolcsatornás lyukszalagnál használatos, és EBCDIC kódba konvertál.

Az olvasott információ helyességének ellenőrzését az operátor állíthatja be az olvasó kezelőpultján. Szükség esetén az ellenőrzés kiiktatható.

A lyukszalag olvasásának folyamatát a 13. ábra szemlélteti.



13. ábra. A lyukszalagolvasás elve

A csatornából érkező olvasási parancs hatására a nyomó görgő a vezető görgőhöz szorítja a szalagot. Mivel a vezető görgő állandóan forog, a szalag mozgásba jön, és elhalad az olvasóállomás alatt. Itt egy sor fotodióda érzékeli a lyukasztásokat, és továbbítja a keletkezett villamos impulzusokat. A fénysugár fókuszálásáról lencserendszer gondoskodik. Az olvasás teljesen hasonló a fotoelektromos rendszerű kártyaolvasáshoz, azzal a különbséggel, hogy a lyukszalag bárhol megállítható. A szalag megállítását elektromágneses fék végzi, és ez képes a maximális sebességgel mozgó szalagot 1,5 mm fékuton belül megállítani.

Az olvasandó lyukszalag hossza elvileg végtelen, gyakorlatilag 10 cm-től 300 m-ig terjed. Ilyen hosszú szalagok elhelyezésére és adagolására külön kazetta szolgál. A berendezés üzemi sebessége 1500 karakter/s. Ha a csatorna ilyen nagy sebességgel nem tudja fogadni az adatokat, az olvasó automatikusan leáll, elkerülve ezzel az adatvesztés lehetőségét.

4.2.4 LYUKSZALAGLYUKASZTÓ (ESZ-7022)

A lyukszalaglyukasztó feladata az, hogy a számítógépből nyert információt megfelelő kódban, lyukszalagon rögzítve megjelenítse. A berendezés öt- és nyolccsatornás lyukszalaggal dolgozik.

A lyukasztás üzemmódjai:

- másoló; tetszőleges kódban ábrázolt adatok lyukasztása átkódolás nélkül (akár öt-, akár nyolccsatornás szalaggal végrehajtható);
- átkódoló; a gép belső kódjából (EBCDIC) valamilyen lyukszalagkódba konvertálja az adatokat, és azután lyukaszt (ebben az üzemmódban csak nyolccsatornás szalaggal dolgozik).

A kívánt üzemmódot az operátor állíthatja be az egység kezelőpultján.

A lyukasztás - hasonlóan a kártyalyukasztáshoz - elektromágneses uton vezérelt késekkel történik; egy lépésben egy karaktert lyukaszt. A lyukasztóba helyezett üres szalag maximális hossza 290 m körül van. Az információs lyukakkal egyszerre történik a szinkroncsatorna lyukasztása is. Az automatikus ellenőrzés kiterjed az információ helyességének, a szalag elszakadásának és a szalagtartalék csökkenésének vizsgálatára is. A berendezés maximális lyukasztási sebessége 150 karakter/s. Ez akkor érhető el, ha az adatokat a csatorna megfelelően gyors ütemben szolgáltatja. Egyébként a sebesség értelemszerűen kisebb.

A lyukszalagos berendezések és általában a lyukszalag viszonylag kis adatmennyiségek tárolása és feldolgozása esetén használható jól; például előnyös lehet tudományos-műszaki számításoknál (ezekre jellemző az általában kis mennyiségű kiinduló adat, melyeket érdemes lyukszalagra rögzíteni. Ezek az adatok lehetnek már valamilyen előző számítás eredményei is). (Gyakran keletkezik lyukszalag regisztráló műszerek melléktermékeként mint másodlagos adathordozó. Az így nyert adatok azonnal feldolgozhatók.)

A 4. táblázat a lyukkártya és lyukszalag összehasonlítását tartalmazza egyes tulajdonságaik alapján. Hogy milyen berendezést célszerű használni, azt a konkrét igények döntenek el.

Lyukkártya és lyukszalag összehasonlítása

Szempont	Lyukkártya	Lyukszalag
Előállítási ár	drágább	olcsóbb
Adatrögzítés ára	drágább	olcsóbb
Input-output sebessége	gyorsabb	lassabb
Input-output periféria árai	drágább	olcsóbb
Melléktermék-e	nem	lehetséges
Program és adatok rögzítésére	előnyösebb	
Rendezhető-e	igen	nem
Helyfoglalás *	nagyobb	kisebb
Javítható-e a program vagy az adatok	kártyacserével igen jól	nehezen vagy egyáltalán nem

* Helyfoglalás: Azonos mennyiségű információ tárolásához szükséges férőhely lyukkártya és lyukszalag esetén.

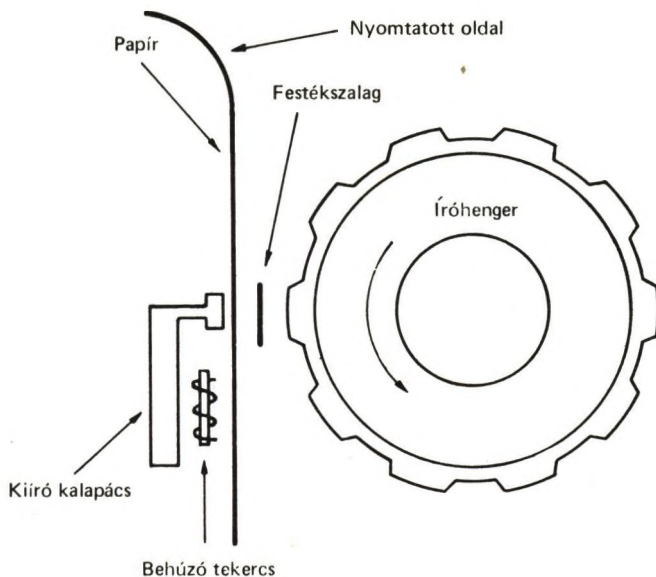
4.2.5 SORNYOMTATÓ (ESZ-7032)

A sornyomtató feladata az, hogy a számítógépből nyert információt a felhasználó számára közvetlenül olvasható formában jelenítse meg, tehát nyomtatott, karakteres szöveget szolgáltatson. A sornyomtató kifejezetten output periféria. Elnevezése onnan ered, hogy egy lépésben egy teljes sort nyomtat szabványos papírra. A papír két szélén perforált; így biztosítjuk a papírtovábbítást, megfelelő fogaskerekekkel. Az egy sorban nyomtatható karakterek maximális száma 128, összesen 64-féle karakter nyomtatható. A nyomtatott szövegről maximálisan két másolat készülhet, speciális papírra.

A sornyomtató elvi működését a 14. ábrán szemléltetjük.

A sornyomtató az alábbi műveleteket képes elvégezni:

- a számítógépből érkező információ vétele és átmeneti tárolása,
- a tárolt információ kiolvasása és nyomtatása,
- papírtovábbítás.



14. ábra. A sornyomtató elvi működése

Az ESZ-7032 sornyomtató ún. dobrendszerű nyomtató. Az íróhengeren (dobon) egy-egy alkotó mentén azonos karakterek helyezkednek el. Az íróhenger hossza azonos a papír szélességével. Egy sorba tehát maximálisan 128 karakter helyezhető el, bár a sorok teljes kitöltése nem kötelező.

A nyomtatás a következőképpen zajlik le: a nyomtatandó sor karaktereinek kódját a sornyomtató fogadja a számítógéptől, és betölti saját átmeneti tárolójába (maximum 128 karakter). Az íróhenger egyenletes sebességgel forog a papír és a festékszalag mögött. Megfelelő időpillanatokban tehát egy-egy karakter áll szemben a papíron egy sor helyével (például csupa Y betű mind a 128 pozíción). Mármint arról kell csupán gondoskodnunk, hogy azokon a pozíciókon, ahová Y betűt akarunk nyomtatni, a papírt odaszorítsuk a festékszalagon keresztül az íróhengerhez. Erre szolgál a 128 db kírókalapács, amelyek elektromágneses működtetésűek, és közülük mindig azok "ütnek", amelyek pozícióján az éppen ott tartózkodó karaktert kell nyomtatni. Hogy az íróhengernek éppen melyik karaktere (geometriailag melyik alkotója) áll szemben a papírral, egy ún. szinkrontárcsa érzékeli. Ez a hengerrel együtt forog. A kírókalapács-

csok vezérlése tehát a nyomtatandó információra hárul, figyelembe véve a szinkrontárcsa helyzetét.

Mindezt egy példán világítjuk meg. Kinyomtatandó a következő karakter-sorozat: 'MUSTANG-741'. Feltételezzük, hogy a karakterek adott sorrendben (nem feltétlenül ábécérendben) helyezkednek el az íróhenger palástján. A karakterek elhelyezkedési sorrendje a következő:

Sorszám	12	33	39	42	46	49	52	63	72	76	79	80
Karakter	'	A	M	T	1	4	7	-	G	N	S	U

A sornyomtató aszinkron vezérlésű, tehát egy sor kinyomtatása mindig akkor kezdődik, amikor a helyi puffer megtelt, és nyomtatási parancs érkezett. Amely karakter nem szerepel a nyomtatandó sorban, annál egyetlen kiírókalapács sem üt rá a papírra. Mindezek alapján a fenti karaktersorozat kinyomtatásának a lépései (feltételeztük, hogy az íróhengeren a következő karakter éppen az aposztróf):

'												
'	•	A										
'	M	A										
'	M	TA										
'	M	TA			1							
'	M	TA			4	1						
'	M	TA			7	4	1					
'	M	TA			-	7	4	1				
'	M	TA	G	-	7	4	1					
'	M	TANG		-	7	4	1					
'	M	STANG		-	7	4	1					
'	M	MUSTANG		-	7	4	1					

Mint látható, egy sor kinyomtatásához annyi lépésre van szükség, ahány különböző fajta karakter fordul elő az adott sorban (esetünkben 12). Mivel az íróhenger egyszeri körülfordulása során az összes karakter sorra kerülhet,

egy sor kinyomtatása maximálisan egy hengerfordulatot igényelhet.

Ha a nyomtatás az íróhenger teljes körülfordulása előtt befejeződik, akkor azonnal új sor nyomtatása következik - feltéve, hogy a nyomtatandó információ a gépből megérkezett, és jelen van az érvényes nyomtatási parancs is. Az aszinkron működés gyors nyomtatást tesz lehetővé, de kissé bonyolultabb vezérlést igényel.

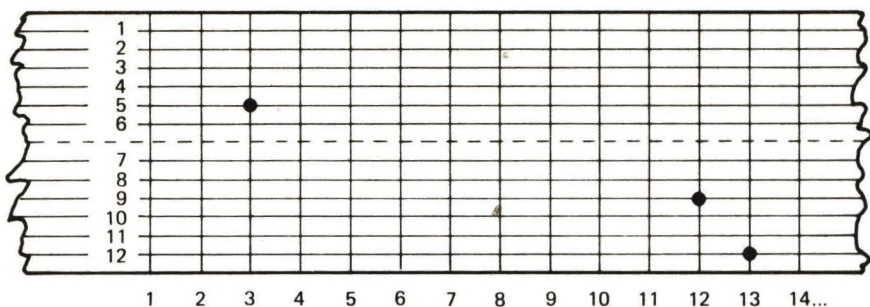
Papirtovábbítás

A papir mozgatása egy erre a célra rendszeresített lyukszalaggal, az ún. carriage- vagy formátumszalaggal történik. Ez egy 12 csatornás lyukszalag, amelynek "logikai" hossza egyenlő egy iv papir hosszával. A formátumszalagot a felhasználó készítheti el, és segítségével a kívánt papirtovábbítást érheti el. A formátumszalag végtelenített; egy körülfordulása alatt 1 iv papir továbbítódik. A 12 csatorna közül háromnak kötött szerepe van:

1. csatorna - a lap kezdete,
9. csatorna - figyelmeztetés a lap végére,
12. csatorna - a lap vége.

A többi csatornán a felhasználó jelölheti ki a lyukasztások helyét, és e lyukakhoz tetszőleges lapmozgatási funkciókat rendelhet hozzá. A hozzárendelés során a futó program megfelelő utasításával kijelöljük a vezérlőcsatornát, valamint megadjuk, hogy a papirtovábbítást nyomtatás előtt vagy után akarjuk végrehajtani. A formátumszalag vázlatos rajza a 15. ábrán látható.

Ha például valahol az 5. csatornán elhelyezünk egy lyukasztást, és utasítást adunk arra, hogy a papirtovábbítás vezérlését az 5. csatorna végezze, addig fog a papir előrehaladni, amíg az olvasóállomás nem érzékeli az 5. csatornán levő lyukasztást. Megfelelő papirhelyzetben kiadott utasítással tehát annyi sort emelhetünk, amennyi a - példánkban az 5. csatornán elhelyezett - lyukasztás



15. ábra. A formátumszalag felépítése

eléréséig még hátra van. A formátumszalag mindig együtt halad a papírral, és egy adott pozíciója papíriveként ciklikusan ismétlődik.

Az ESZ-7032 típusú sornyomtató nyomtatási sebessége kb. 900 sor/perc. A sornyomtató a számítóközpont egyik kulcsfontosságú perifériája; nélküle a számítógép korszerűen ugyszólván nem is üzemeltethető. Igen nagy előnye a többi output perifériával szemben, hogy közvetlenül olvasható, maradandó bizonylatot (például táblázatot) szolgáltat.

4.2.6 KONZOLÍRÓGÉP (ESZ-7070)

A konzolirógép feladata, hogy biztosítsa az információforgalmat az operátor és a számítógép központi egysége között. A konzolirógépet szorosan vett input-output műveletekre nem használjuk! A berendezés közönséges elektromechanikus írógép, rövid üzenetek, illetve operátori parancsok, közlések továbbítására szolgál. Nyomtatási sebessége maximum 10 karakter/s; egy sorban legfeljebb 106 karakter nyomtatható. Az írógép jelkészlete 93 karakterből áll. A papir folytonos, egyszerre több másolat is készíthető (például naplózás céljára).

A konzolirógép - a sornyomtatóval együtt - a korszerű számítóközpont üzemeltetésében nélkülözhetetlen periféria, hiszen ezen keresztül teremthet kapcsolatot az operátor a központi egységgel, illetve az operációs rendszer bizonyos programjaival. Ugyancsak a konzolirógépen közli a számítógép az egyes programok állapotáról, a feldolgozás állásáról szóló információt.

4.2.7 MÁGNESSZALAGEGYSÉG (ESZ-5511)

A mágnesszalagegység igen gyors, háttértárolóként használt mágneses periféria. Adathordozója a már megismert mágnesszalag, amely cserélhető, tehát az egység bármely szabványos mágnesszalaggal tud dolgozni. Elvileg végtelen tárolókapacitása miatt elsősorban felhasználói adatokat őrzünk rajta; törzsadatokat, referenciaadatokat stb. Ugyanakkor átmeneti munkaterületként is használatos: a program futása során éppen fel nem használt információt, részeredményeket, esetleg később futtatandó programrészeket tárolhatunk rajta.

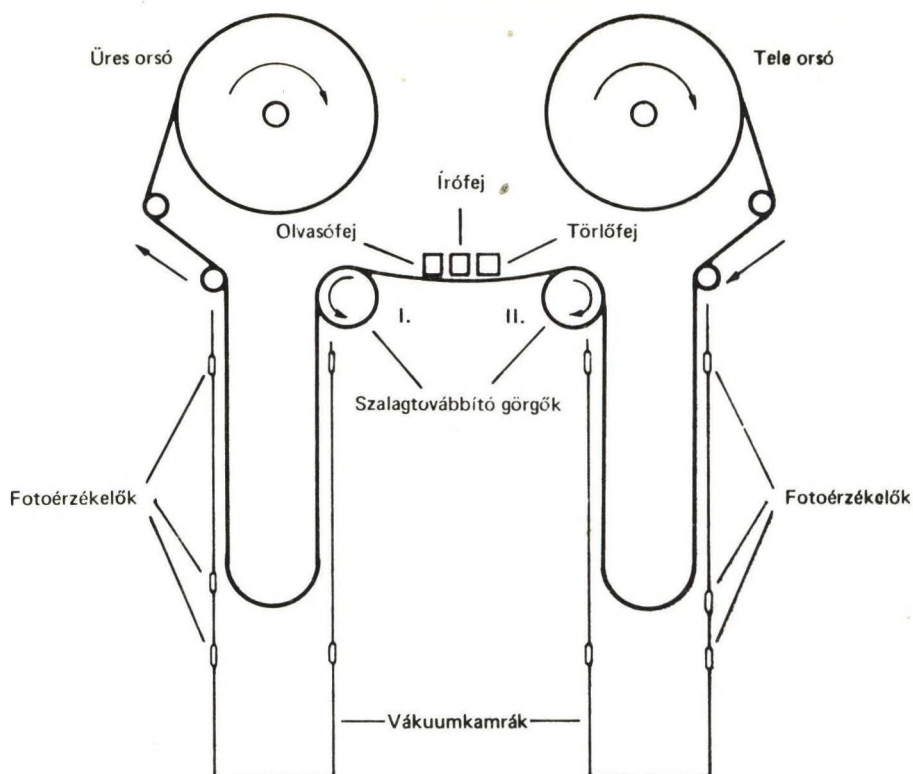
A szalagegységet különálló vezérlőegység irányítja; egy mágnesszalag-vezérlő egységhez maximálisan 8 mágnesszalagegység csatlakoztatható (ennek címzési okai vannak). Általában valamelyik szelektor csatornán keresztül bonyolítja le az adatforgalmat a központi egységgel. Azért, hogy a sokszori befűzés a szalag információtartalmát ne rongálja, az elején és a végén néhány méteres szakaszt fenntartunk a befűzésre. Ezek végén helyezkedik el a szalag nem mágnesezhető (külső) felére ragasztott "szalag eleje", illetve "szalag vége" jel, amely egy közönséges alumínium fólia. Az információkhoz csak e két jel közötti szalagtartományban lehet hozzáférni.

A mágnesszalagegység működése

A szalag befűzése után a készülék bekapcsolható, s ekkor automatikusan beáll a "szalag eleje" jelre. Ezzel alkalmassá tettük a berendezést az adatforgalom lebonyolítására, illetve egyéb parancsok végrehajtására. Alapvetően 4-féle műveletet végezhet:

- előre és hátra csévélés,
- írás előre,
- olvasás előre és hátra,
- egyéb műveletek (törlés, diagnosztika stb.).

A szalagtovábbítás un. vákuumgörgős megoldásu (lásd a 16. ábrát). A szalaggal telt orsót a jobb oldali tengelyre helyezzük, a felvevő üres orsót



16. ábra. Szalagmozgató mechanizmus vázlatos rajza

pedig a bal oldalira. A szalagtovábbító görgők palástján piciny nyílások találhatók; ezeken keresztül a görgő belseje érintkezik a külső levegővel. A szalagtovábbítás úgy folyik, hogy a kívánt szalagmozgással egyező irányban forgó görgő belsejében vákuumot létesítünk, s ez ily módon mintegy magához szippantja és magával ragadja a szalagot. A másik - ellentétesen forgó - görgőbe pedig kis tuynyomást vezetünk, s azon a szalag, mint valami légpárnán, továbbiklik. Ezzel elérjük, hogy a szalag mozgásirányának változtatásakor a görgőknek nem kell visszafelé forogniuk, csupán a nyomás- és vákuumviszonyokat kell megváltoztatnunk. Mindez igen gyors működést tesz lehetővé. Mivel a viszonylag súlyos szalagtároló orsók a hirtelen lassulást, irányváltást nem tudják követni, a feszes szalag elszakadna ilyenkor. Ennek megelőzésére egy-egy vákuumkamra gondoskodik arról, hogy legyen elegendő "tartalék" szalag, amely-

nek hossza változhat, és így kiegyenlíti a mozgási egyenetlenségeket. Ezt a megoldást vákuumkamrás szalagpuffernak nevezzük. A kamrákba leszivott szalag helyzetét fotoérzékelők tartják optimális szinten.

A mágnesszalagegységnél szorosan vett átlagos elérési időről nem beszélhetünk, hiszen a keresett adat éppugy lehet a szalag elején, mint a végén, vagy valahol a közepén. Mivel a teljes szalag áttekerése kb. 6-7 percet vesz igénybe, 3-3,5 perces "átlagos" elérési időről szoktak beszélni. A szalag névleges haladási sebessége 2 m/s. A szokásos írássűrűség 32 karakter/mm, tehát a névleges adatátviteli sebesség 64000 byte/s. Az információ átvitele byte-onként történik. Ajánlatos az adatokat blokkolva tárolni, helyenként üres hézagot hagyva a szalagon. Így könnyebben hozzáférhető valamely "belül fekvő" adathalmaz is.

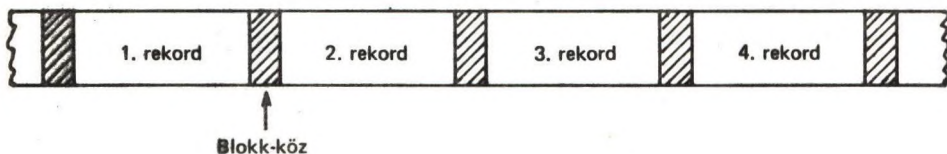
Minden egyes mágnesszalagtekercs védhető az adatok felülírása ellen. Az orsó hátlapján műanyag karika - az ún. írásgyűrű - látja el a védelmi funkciót. Ha az írásgyűrűt levettük, az azt befogadó horony üres marad az orsón, így az abba illeszkedő piciny nyomógomb - melyet a gyűrű nyom le, ha benn van a horonyban - kienged az orsó feltevése után. Ennek a kis kapcsolónak a kiengedett állapota letiltja az elektronikus íróberendezést, tehát a szalag információtartalmát nem tudjuk tönkretenni. Ugyanakkor a helyére tett írásgyűrű benyomva tartja ezt a nyomógombot, és így engedélyezi az írási műveletet.

Adatelrendezés a mágnesszalagon

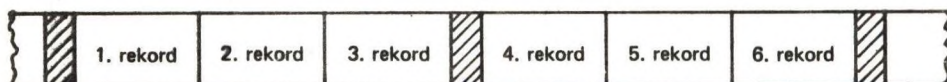
A szalagon az információt ún. blokkok egymásutánjában célszerű csoportosítani. Az egyes blokkokat blokkhézag választja el; ez hasznos információt nem tartalmaz, és arra szolgál, hogy a blokkok feldolgozásakor a szalagot adatvesztés nélkül meg tudjuk állítani. Minden blokk rekordokból áll. Rekord alatt értjük a valamilyen szempontból összetartozó adatok egy meghatározott mennyiségét (például a dolgozók személyi adatai). Több rekordot logikai egységbe fogva file-hoz jutunk (például egy bizonyos üzemegeység dolgozóinak személyi adatait tartalmazó file). Figyeljük meg, hogy a blokk és a file nem rokon fogalom; a file logikailag összetartozó adatokat jelent, a blokk pedig a tényle-

ges fizikai adatrögzítésre utal. Mágnesszalagnál a minimális blokkhossz 18 byte (ennek mechanikai okai vannak). Blokkoltnak tekintjük tehát az adatrögzítést, ha több rekordot fizikailag egységbe - blokkba - fogunk össze, és blokkolatlanak, ha az egyes rekordok fizikailag önállóak (blokk-köz választja el őket). Alább bemutatjuk a gyakoribb adatelrendezéseket:

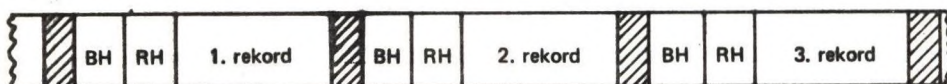
a) Fix hosszúságú, blokkolatlan rekordok



b) Fix hosszúságú, blokkolt rekordok: a rekordhossz állandó, és minden egyes blokk azonos számú rekordból áll.



c) Változó hosszúságú, blokkolatlan rekordok: minden egyes blokk egyetlen változó hosszúságú rekordból áll.



d) Változó hosszúságú, blokkolt rekordok: minden blokk több változó hosszúságú rekordból áll.



BH = blokkhossz (byte-ban)

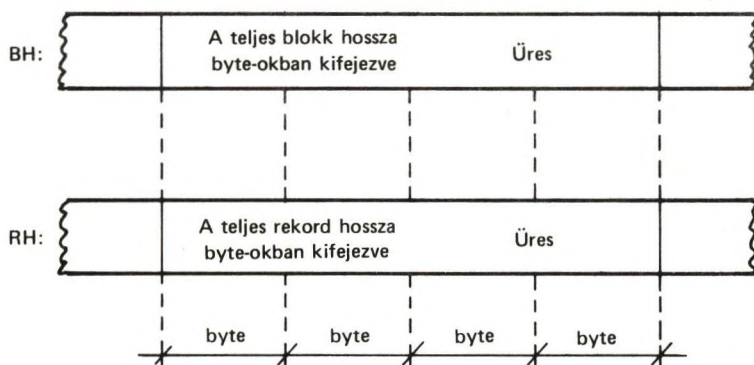
RH = rekordhossz (byte-ban)

e) Definiálatlan rekordok: olyan rekordok tartoznak ebbe a csoportba, amelyek számára egyik formátum sem lenne megfelelő.



17. ábra. Adatelrendezés mágnesszalagon

A blokkhossz-(BH) és rekordhossz-(RH) mezők felépítése az alábbi:



A rekordhossz és blokkhossz kiszámítására az alábbi képletek érvényesek:

$$RH = \text{a rekord adatbyte-jainak száma} + 4,$$

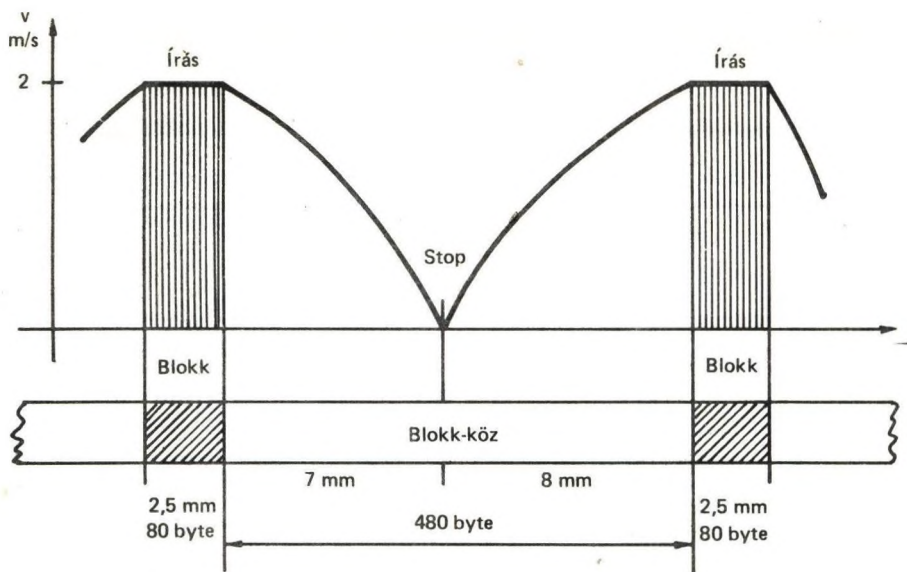
$$BH = \text{a blokkban foglalt rekordok rekordhosszainak összege} + 4.$$

Mint látható, az egyes hosszadatok az RH, illetve BH mezőt is beleszámítják a rekord, illetve blokk tényleges hosszába. Változó hosszúságú rekordok, illetve blokkok esetén a rekordhosszt és blokkhosszt azért kell megadnunk, hogy az operációs rendszer input-output vezérlő programjait tájékoztassuk arról, milyen adatmennyiségre kell számítaniok.

Megjegyzés. Blokkolatlan rekordok esetén igen rossz szalagkihasználás keletkezhet, különösen rövidebb (például 80 byte-os, vagyis 1 kártyányi) rekordok rögzítésekor. A blokk-közök ugyanis aránytalanul nagy helyet foglalhatnak el az információt tartalmazó szalagrészekhez képest. A blokk-köz hossza attól függ, milyen jók a periféria szalagmozgató mechanizmusának dinamikus tulajdonságai (gyorsulás, lassulás, fékezés). A 18. ábra a legkedvezőtlenebb esetet mutatja, 32 karakter/mm írássűrűség esetén, 80 byte-os rekordokra nézve.

Látható, hogy a szalag kihasználtsága igen kedvezőtlen:

$$\frac{\text{Hasznos szalaghossz}}{\text{Teljes szalaghossz}} = \frac{2,5 \text{ mm}}{17,5 \text{ mm}} \cong 14 \%$$



18. ábra. Mágnesszalag kihasználtsága

A hasznos szalaghossz tehát az elfoglalt hosszának 14 %-a, amely igen kedvezőtlen érték. Ha most például nyolckártyányi információt egy blokkba fogunk össze, akkor $8 \cdot 80 = 960$ byte lesz a blokkhossz, a blokk-köz pedig továbbra is 480 byte marad. A szalag kihasználtságát tehát jelentősen javítottuk: 14 %-ról 67 %-ra. Ez egyszersmind a feldolgozási időt is rövidíti.

Mágnesszalagcímkek

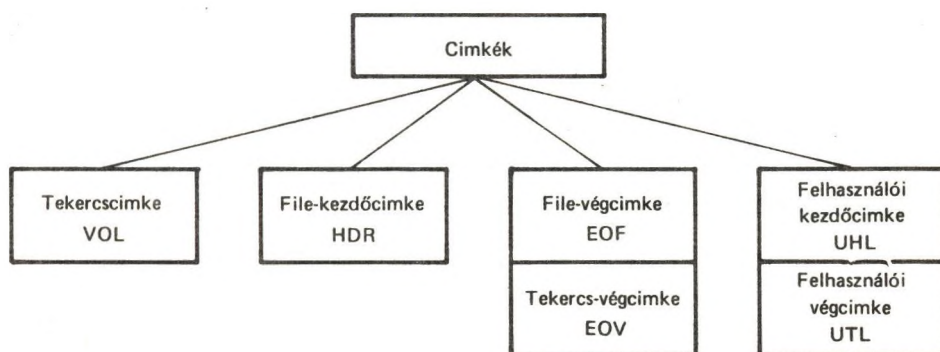
Mivel a számítógéppontban, illetve maguknál a felhasználóknál is tetemes mennyiségű mágnesszalagtekercset kell tárolni, szükség van olyan információs rendszerre, amely lehetővé teszi a pontos tájékozódást az egyes szalagok között. Nemkülönbön fontos az is, hogy valamely felhasználó ne dolgozhasson tévedésből egy másik felhasználó adataival, tehát kiküszöböljük a mágnesszalagok véletlen kicserélését stb. Ezeket a feladatokat oldják meg a szalagcímkek.

Címke = a szalagra mágneses uton felírt azonosító mező

A címkemezőben a felhasználó megadhat minden olyan adatot, amely az ő adatállományát jellemzi. A címkék lehetnek standard és nem standard címkék; alább a standard címkékkal foglalkozunk. Adatfeldolgozáskor az operációs rendszer bizonyos rutinjai leolvassák a szalagcímkeket, és összehasonlítják azok információtartalmát a felhasználó által megadott paraméterekkel. Eltérés esetén a feldolgozás nem kezdődhet el.

A címkék fajtái

A standard mágnesszalagcímkek az alábbi csoportokra oszthatók:



A címkék nevei angol rövidítések. Értelmezésük:

- | | |
|----------------------------|---------------------------|
| - VOL (Volume Label) | - Tekercscímke |
| - HDR (File Header Label) | - File-kezdőcímke |
| - EOF (End of File) | - File-végcímke |
| - EOY (End of Volume) | - Tekercsvégcímke |
| - UHL (User Header Label) | - Felhasználói kezdőcímke |
| - UTL (User Trailer Label) | - Felhasználói végcímke |

Alább részletesen megadjuk az egyes standard mágnesszalagcímkek formátumát. A címkék egységesen 80 byte hosszúak, és felosztásuk kötött. Az egyes címkemezők tartalmát az alábbi táblázatok byte-okra lebontva tartalmazzák.

a) VOL - Volume Label

Pozíció (byte) sorszáma	Cimkemező tartalma
1-3	Cimkeazonosító; esetünkben VOL
4	VOL címkek sorszáma a tekercsen (mindig 1)
5-10	Archiválási szám
11	Tekercsvédelem; a DOS nem használja
12-21	Üres pozíciók; a mágnesszalag nem használja
22-31	Rezervált mező; üres pozíciók
32-41	Rezervált mező; üres pozíciók
42-51	A tulajdonos neve és címe
52-80	Rezervált mező későbbi bővítéshez

Mint látható, a tekercscimke az egyes mágnesszalagtekercsek azonosítására szolgál. Egy szalagon maximálisan 8 VOL címke helyezhető el a szalag elején, de általában csak egyet használnak.

b) HDR - File Header Label

Pozíció (byte) sorszáma	Cimkemező tartalma
1-3	Cimkeazonosító, esetünkben HDR
4	HDR címkek sorszáma a HDR címkecsoportban
5-21	File-azonosító (alfanumerikus jelsorozat)
22-27	Nyilvántartási szám; a file első mágnesszalag-tekercsének sorszáma az archivumban
28-31	Tekercs sorszáma a file-ban (0001-9999)
32-35	File sorszáma a file-csoportban (0001-9999)
36-39	A file kiadási száma, nyilvántartása
40-41	Kiadási verzió száma; az előző mezőt egészíti ki, pontosabbá teszi annak információ-tartalmát

Pozíció (byte) sorszáma	Cimkemező tartalma
42-47	A file keletkezési dátuma (év, nap)
48-53	File-megőrzési határidő (év, nap)
54	File-védelem (a DOS nem használja)
55-60	Blokk számláló (HDR címkében nem használ- latos)
61-73	Az operációs rendszer kódja
74-80	Rezervált mező

A HDR címkéket mindig a VOL címkék után írjuk; pontos leírást adnak a file-ról. Maximálisan 8 HDR címke helyezhető el file-onként a szalagon, de általában csak egyet használnak.

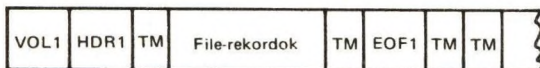
c) EOF és EOVS - End of File, End of Volume

Pozíció (byte) sorszáma	Cimkemező tartalma
1-3	Címkeazonosító, esetünkben EOF, illetve EOVS
4	EOF/EOVS címkék sorszáma az EOF/EOVS cím- kecsoportban
5-80	Azonos a megfelelő HDR címke megfelelő me- zőivel

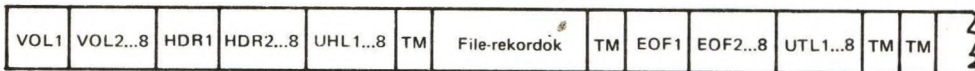
Az EOF címkéket a file utolsó rekordja után helyezzük el, az EOVS címkék a szalag (tekercs) végére kerülnek. Egy csoportban maximálisan 8 EOF, illetve EOVS címke lehet.

Megjegyzés. Az EOF és EOVS címkékben használatos egy blokkszámláló is, melynek értéke a file blokkjainak számával egyenlő, illetve EOVS után a tekercs blokkjainak számát tartalmazza.

Egy file egy tekercsen

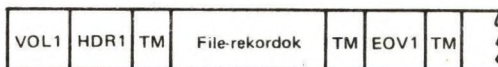


Maximális címkekészlet

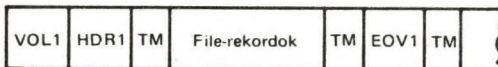


Egy file több tekercsen

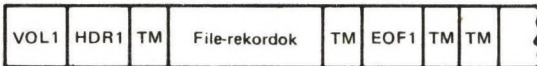
1. szalag



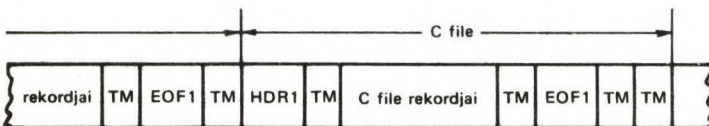
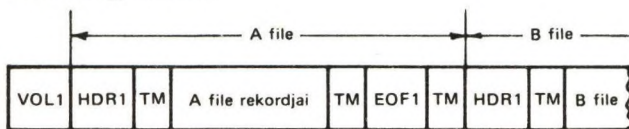
2., 3. stb. szalagok



Utolsó szalag



Több file egy tekercsen



Több file több tekercsen



Megjegyzés. TM = tape mark – szakaszjel. Speciális jel, amely a file adatainak és a címkeknek az elkülönítésére szolgál. Önálló blokkot alkot, de a file blokkjaiba nem számít bele (a blokkszámoló értékét nem befolyásolja). A „tape mark”-ot az operációs rendszer írja a szalagra. Egy-egy „tape mark” szolgál a címkek, file-ok elkülönítésére és a szalag végének jelzésére, ha a file még tovább tart (EOV). Két „tape mark” kerül a szalag végére, ha ez az utolsó vagy egyetlen file-végével egybeesik (lásd az 1–5. szalagvégeket EOF1 után).

19. ábra. Mágnesszalagcímkek elrendezései

d) UHL és UTL - User Header Label, User Trailer Label

Pozíció (byte) sorszáma	Cimkemező tartalma
1-3	Cimkeazonosító, esetünkben UHL, illetve UTL
4	UHL/UTL címkek sorszáma az UHL/UTL címkecsoportban
5-80	Tetszőleges felhasználói információ

Az UHL címkek - ha szerepelnek - mindig a HDR címkek után következnek, az UTL címkek - ha vannak - az EOF, illetve EOY címkeket követik. Felhasználói címkeket csak standard (nem felhasználói) címkek (VOL, HDR, EOF, illetve EOY) jelenlétében lehet elhelyezni, magukban nem állhatnak. A nem felhasználói címkeket az operációs rendszer kezeli, de a felhasználói címkek feldolgozásáról a programozónak kell gondoskodnia. Egy csoportban maximum 8 UHL, illetve UTL címke állhat. Használhatók címke nélküli mágnesszalagok is; címkek használata nem kötelező.

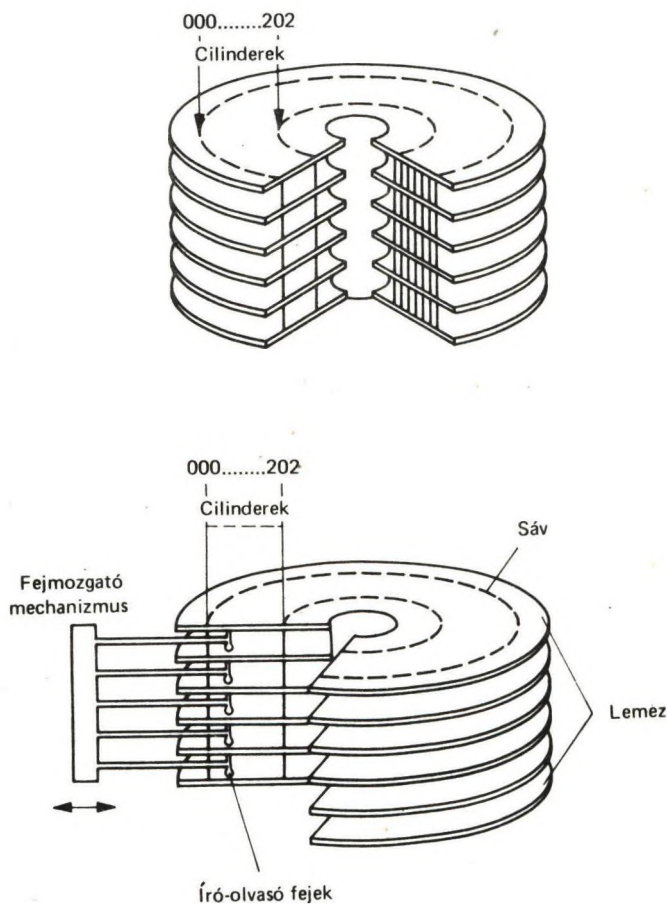
A 19. ábrán bemutatjuk néhány címkével ellátott file elrendezését a mágnesszalagon. Figyeljük meg, hogy egy file több tekercset is elfoglalhat, valamint egy tekercsen több file is elhelyezhető a file méretétől függően. Az ábrák rávilágítanak a címkehasználat fontosságára is.

4.2.8 MÁGNESLEMEZEGYSÉG (ESZ-5052)

A mágneslemezegység a gyors perifériák közé tartozik. Adathordozója a már megismert hatlemezes cserélhető mágneslemezkeg (disc), tehát az egység bármely szabványos lemezkeggyel tud dolgozni. Igen nagy adattároló kapacitása miatt a lemezegységet háttértárolóként üzemeltetjük, s úgy tekintjük, mintha az operatív tár kapacitását bővítené. Általában mágneslemezen tárolódik az operációs rendszer, a különböző programkönyvtárak, de a felhasználó is tárolhatja lemezen adatállományait és programjait.

Mivel a mágneslemezegység bonyolultabb periféria a nem mágneses készüléknél, a mágnesszalaghoz hasonlóan a saját perifériavezérlő egysége nincs vele egybeépítve, hanem külön egységet képez. Egy vezérlőegységhez maximumisan nyolc mágneslemezegység csatlakoztatható. A vezérlőegység ezután szokás szerint valamelyik szelektor csatornán keresztül bonyolítja le az adatforgalmat a központi egységgel.

A mágneslemez minden egyes lemezoldalát 203 koncentrikus kör alakú sávra osztották. Ezek sorszámozása kívülről befelé 000-tól 202-ig terjed. Valamennyi lemezoldal azonos sorszámú sávjait együttevénél cilindernek nevezzük. Egy cylinder tehát tíz sávból áll, és lemezekötegenként 203 cylinder van. A felosztást a 20. ábra szemlélteti.



20. ábra. Mágneslemez felosztása

A készülék névleges fordulatszáma 2400 fordulat/perc. Bekapcsolása után ezt az értéket rövid idő alatt eléri, s ekkor alkalmassá válik adatátvitelre, és készenléti állapotát azonnal jelzi a csatornának. Amennyiben a programban a mágneslemezre vonatkozó input-output utasítás van, a mágneslemez vezérlőegysége fogadja a csatornától az adatátviteli parancsot. Ha ugyanakkor nem foglalt egy másik művelet végrehajtásával, elindítja az adatátvitelt a megcímzett mágneslemezegységgel. Mindezt átviteli parancsok vezérlik, amelyek főbb típusai a következők:

- a) Pozicionálási parancsok – a megfelelő cylinder és író-olvasó fej (adatsáv) kiválasztását vezérlik.
- b) Keresési parancsok – a kiválasztott sávon rekordok keresését végzik például kulcs, identifikátor stb. alapján.
- c) Olvasási parancsok – a megtalált és azonosított rekordok olvasását vezérlik.
- d) Írási parancsok – a kívánt mezők lemezre írását végzik (például azonosító mező + kulcs + adatmező írása stb.).
- e) Egyéb parancsok. Ide tartozik néhány speciális parancs is, ezekkel itt nem foglalkozunk.

Az adatátviteli parancsoknak logikus sorrendben kell követniük egymást a sikeres átvitel érdekében. A pozicionáló parancs elvégzi az író-olvasó fejeknek a kiválasztott adatsávra állítását, a kereső parancsok pedig a kiválasztott sávon kikeresik a kívánt rekordokat. Csak ezután kezdődhet el az adatátvitel. Ha a megjelölt helyen a keresett rekord nem található, adatátvitel nem megy végbe, hanem a csatorna jelzést kap a keresés sikertelenségéről a mágneslemez vezérlőegységétől.

A parancsok egységesen egy byte hosszúságúak. A mágneslemez vezérlőegysége dekódolja ezeket, és az egyes mágneslemezegységek felé már csak vezérlőjeleket továbbít (például fejmozgatás). Egy adott időpontban mindig csak egy író-olvasó fej végezhet adatátvitelt. A teljes adatátviteli művelet alatt fennáll a csatornával a logikai és elektronikus kapcsolat. Az utolsó információs byte átvitele után a vezérlőegység lekapcsolódhat a csatornáról (és az interface-ről), ha a csatorna másként nem intézkedik. Az átvitel végrehajtására vonatkozó információ egy ún. állapotbyte-ban (status byte) rögzítődik; hibás átvitel esetén ez szolgálhat a hibafelderítés alapjául.

Tekintsünk át néhány tájékoztató adatot:

- adattároló kapacitás: 7 250 000 byte,
- átváltási idő két szomszédos sáv között: 20 ms,
- átlagos adatelérési idő: 60 ms,
- maximális adatelérési idő (a legkedvezőtlenebb): 95 ms,
- információsűrűség a 000 számú sávokon: 30 bit/mm,
- információsűrűség a 202 számú sávokon: 44 bit/mm,
- adatátviteli sebesség: 1 250 000 bit/s.

Látható, hogy az igen nagy információsűrűség miatt rendkívül jó minőségű mágnesezhető rétegre van szükség, és az alkatrészek finommechanikai megmunkálása is igen nagy pontosságot igényel. Mindez együttvéve lehetővé teszi, hogy a lemezegység az egyik leggyorsabb periféria legyen, és a központi egységnek ne kelljen viszonylag sokat várakoznia egy-egy adatra. Mivel a mágneslemezen gyakorlatilag minden információ egyszerre "kéznél van", nagyjából az átlagos elérési időn belül hozzáférhető, a mágneslemezt szokás közvetlen elérésű tárolónak nevezni (direct access storage device; DASD), szemben a mágnesszalaggal, amely soros elérésű.

Adatelrendezés a mágneslemezen

A mágneslemez igen nagy tárolókapacitása és gyors adatátviteli tulajdonságai miatt szükség van arra, hogy a felhasználói adatok helyének megállapítását, illetve általában a lemezen tárolt információ megtalálását biztonságosan

tudjuk elvégezni. E célból a lemezre bizonyos kiegészítő információt is fel kell írni, amely valamilyen formában tartalmazza a lemezen található adatok főbb tulajdonságainak leírását. Mindezek alapján a lemez minden egyes adatsávjának fizikai felosztását a 21. ábra mutatja.



21. ábra. Adatelrendezés mágneslemezen

Ugy tűnik, hogy a kiegészítő információ mennyisége felülmulja a hasznos adatokét. Valójában ez nincs így - az azonosító és kulcsmező hosszúsága sokkal kisebb lehet, mint az adatmezőé. Alább ismertetjük az egyes mezők információtartalmát.

a) Sávcím (home address)

Byte sorszáma	Információtartalom
0	Jelző; a sáv állapotát írja le
1	Zérus valamennyi biten
2	A cilinder száma
3	Zérus valamennyi biten
4	Író-olvasó fej száma (sáv száma)
5-6	Ellenőrző byte-ok

A sávcím minden egyes sávra vonatkozólag az adott sáv fizikai elhelyezkedésére (címére) és állapotára nézve nyújt tájékoztatást. Minden sávnak saját sávcíme van, és ez a "Kezdőpont" után következik. A "Kezdőpont"-tól számítjuk valamennyi sáv kezdetét; ezt egy kis bemetszés jelöli a lemezkötegen.

b) Azonosító mező (count)

Byte sorszáma	Információtartalom
0	Jelző; ugyanaz, mint a sávcím
1	Zérus valamennyi biten
2	A cilinder száma
3	Zérus valamennyi biten
4	Író-olvasó fej száma (sáv száma)
5	Rekord száma – rendszerint a rekord sorszámát tartalmazza az adott sávon
6	A kulcsmező hosszúsága (byte-ban)
7-8	Az adatmező hosszúsága (byte-ban)
9-10	Ellenőrző byte-ok

Minden egyes rekordnak saját azonosító mezője van. Ez tartalmazza az összes szükséges leíró információt az adott rekordra vonatkozólag. Az 1-5. mezőt szokás identifikátornak is nevezni; léteznek olyan mágneslemezparancsok, amelyek identifikátor alapján keresik az illető rekordot.

c) Kulcs (key)

A kulcsmezőben tetszőleges alfanumerikus információ helyezhető el, és ez a következő adatmező azonosítására szolgál (például kulcs alapján is lehet rekordot keresni). A kulcsok többnyire növekvő sorrendben követik egymást, amíg a sáv be nem telik. Ezután valamelyik következő sávon folytatódhat az adatrögzítés. Az egyes mezőket hézagok választják el egymástól. Különböző kódokat tartalmaznak, amelyek az adott mágneslemezegységtől függenek. Hossz-

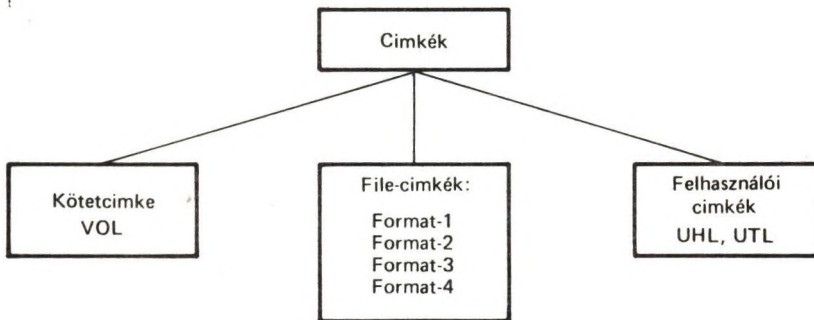
szuságuk különböző. A mágneslemez mechanikai, forgási egyenetlenségeinek következményeit is ezek az elválasztó mezők veszik magukra.

Az R0 rekordnak nincs kulcsmezője. Ez a rekord ún. sávazonosító blokk (track descriptor record), amelyet az operációs rendszer használ pl. tartalék sáv (alternate track) kijelölésére. A felhasználói rekordok R1-gyel kezdődnek.

Mágneslemez címkék

A mágnesszalaghoz hasonlóan, a mágneslemez esetében is feltétlenül szükséges, hogy rendelkezésünkre álljon egy olyan információs rendszer, amely lehetővé teszi az eligazodást a lemezen tárolt adatok között. Mivel a közvetlen elérésű tárolók, tehát a mágneslemez is sokkal bonyolultabb adatelrendezéseket, file-szervezést tesz lehetővé, mint a soros elérésű tárolók, érthető, hogy címkerendszere is bonyolultabb. Alább vázlatosan bemutatjuk a mágneslemez logikai felosztását, a standard mágneslemez címkéket, valamint megadjuk az egyes címkemezők információ tartalmát is.

A standard mágneslemez címkék az alábbi csoportokra oszthatók:



a) Kötet címke - VOL

Minden lemezköteg 000 számú cilinderén a 0-dik sáv harmadik rekordja a VOL címke. Funkciója az illető mágneslemezcsomag azonosítása. Elvileg nyolc db lehet belőle egy VOL címkecsoportban, de a DOS operációs rendszer csak egyet kezel.

Az egyes címkemezők tartalma az alábbi:

Kulcsmező:

Byte sorszáma	Címkemező tartalma
1-3	Címkeazonosító; esetünkben VOL
4	VOL címkék sorszáma a VOL címke- csoportban

Adatmező:

Byte sorszáma	Címkemező tartalma
1-3	Címkeazonosító; esetünkben VOL
4	VOL címkék sorszáma a VOL címke- csoportban
5-10	A kötet sorszáma; az illető mágneslemez- csomag sorszáma az archivumban
11	Kötetvédelem; a DOS nem használja
12-21	VTOC kezdőcíme (lásd alább)
22-31	Rezervált mező
32-41	Rezervált mező
42-51	A tulajdonos neve és címe
52-80	Rezervált mező

b) File-címkék

A file-címkék az egyes file-ok leírását tartalmazzák. Egységesen 140 byte hosszúságúak; 44 byte-os kulcsmezőből és 96 byte-os adatmezőből állnak. A file-címkéket a rendszer egy erre a célra kijelölt helyen, egy speciális táblázatban tárolja. Ez az ún. VTOC maga is file, és saját file-címkével rendelkezik. Követelmény, hogy a VTOC egy cilinderen elférjen, címét a VOL1 címke tartalmazza. A VTOC elhelyezéséről a felhasználó rendelkezik a kötet első használatakor (inicializálásakor). A file-címkék négy különböző formában

fordulhatnak elő, ezeket rendre a Format-1, Format-2, Format-3 és Format-4 jelöléssel láttuk el. Az adott esetben használatos file-cimkekészlet a file típusától függ; a kapcsolatot a táblázatban foglaltuk össze.

Format-1 címke. Ez a címke minden file-nál használatos. A file-hoz tartozó lemezterületeket definiálja, ezek maximális száma 3. A területeknek egyenként összefüggőeknek kell lenniök: a három terület azonban a lemezcsomag tetőszőleges három helyén definiálható. Ha a file-hoz háromnál több független terület tartozik, akkor Format-3 címkét is kell használnunk, és erre utalnunk kell a Format-1 címkében. Az egyes címkemozók értelmezése:

Kulcsmező:

Byte sorszám	Cimkemoző tartalma
1-44	File-azonosító, alfanumerikus kód

Adatmező:

Byte sorszám	Cimkemoző tartalma
45	Formátumazonosító, esetünkben 1 (F1 ₁₆)
46-51	File-nyilvántartási szám, a file/kötet kapcsolatot teszi pontosabbá
52-53	A kötet sorszám, több kötetes file-nál az illető kötetet azonosítja
54-56	A file keletkezési dátuma (év, nap)
57-59	File-megőrzési határidő (év, nap)
60	A file-hoz tartozó területek száma az adott kötetben
61	A DOS nem használja, tartalma szóköz
62	Rezervált mező, tartalma szóköz
63-75	A programozási rendszer kódja (operációs rendszer verziója)
76-82	Rezervált mező, tartalma szóköz

Byte sorszám	Címkemező tartalma
83-84	A file típusa (szekvenciális, direkt stb.)
85	A DOS nem használja
86	Opcionális file-területeket leíró kód (indexelt szekvenciális file-oknál főindex és tulcsordulási területek jelzésére)
87-88	Blokkhossz
89-90	Rekordhossz
91	A kulcs hossza (byte-okban)
92-93	A kulcsmező elhelyezkedése blokkolt rekordok esetén
94	Adatállomány-jelző; több kötetes file ese- tén jelzi, hogy utolsó kötetről van-e szó
95-105	A DOS nem használja
106, 116, 126	File-területek típusát határozza meg*
107, 117, 127	File-terület sorszáma
108-111	Az adott file-terület kezdőcíme
118-121	
128-131	
112-115	Az adott file-terület utolsó címe a mág- neslemezen
122-125	
132-135	
136-140	Az adott file következő címkéjének címére mutat (Format-2; vagy Format-3, ha van), egyébként a tartalma bináris 0

Ebből a vázlatos leírásból is kitűnik, hogy a Format-1 típusu file-címke vala-
mennyi fontos információt tartalmazza az adott file-ra vonatkozólag.

Format-2 címke. Ez a címketípus csak indexelt szekvenciális file-oknál
használatos (lásd a 10. fejezetben). Mindig azon a lemezkötegen található, ame-
lyen a cylinderindex. Nyilvántartást tartalmaz minden egyes file-hozzáfordu-
lásról.

* A 106-115, 116-125, 126-135 mező egy-egy file-terület leírására szolgál.

Kulcsmező:

Byte
sorszáma

Cimkemező tartalma

1	Kulcsazonosító. Tartalma 02 ₍₁₆₎ , és az adott kulcs megkülönböztetésére szolgál a Format-1 címkék kulcsától
2-44	A DOS nem használja

Adatmező:

Byte
sorszáma

Cimkemező tartalma

45	Formátumazonosító, esetünkben 2 (F2 ₁₆)
46	Különböző szintű indexek száma
47	A DOS nem használja
48-50	Az első adatrekord címe valamennyi cilinderen
51-52	Az elsődleges adatterület utolsó adatsávjának címe cilinderenként
53	A DOS nem használja
54	A főindexbejegyzéseket tartalmazó sávok lehetséges legmagasabb rekordsorszáma
55	Az elsődleges adatsávok lehetséges legmagasabb rekordsorszáma
56	A tulcsordulási sávok lehetséges legmagasabb rekordsorszáma
57	Utolsó rekord sorszáma a sávindexszel közösen használt sávon (ha van ilyen)
58	Utolsó rekord (indextétel) sorszáma a sávindex sávjain
59	Rezervált mező; tartalma zérus
60-61	Számláló, logikailag törölt rekordokra vonatkozik
62-64	Számláló, tulcsordulási láncokban levő rekordok eléréséhez szükséges direkt hozzáfordulások számlálására

65-66	A legmagasabb index hossza (byte-okban)
67	A legmagasabb index által elfoglalt sávok száma
68-71	Az elsődleges adatterületre irt rekordok száma
72	A teleirt file-terület állapotát írja le
73-79	A cylinderindex címe
80-86	A (legmagasabb szintű) főindex címe *
87-93	A főindex (ha van) vagy cylinderindex címe
94-101	Az elsődleges adatterület utolsó rekordjának címe
102-106	A sávindex utolsó tételének címe
107-111	A cylinderindex utolsó tételének címe
112-116	A főindex utolsó tételének címe
117-124	Az utolsó rekord címe a független tulcsordulási területen
125-126	A DOS nem használja
127-128	A független tulcsordulási terület még szabad sávjainak száma
129-130	Az összes tulcsordult rekord száma
131-132	Betelt cilindertulcsordulási területek száma
133-140	A DOS nem használja; rezervált mezők

Látható, hogy a Format-2 típusú címke igen részletes leírást ad az indexelt szekvenciális file-ról. Az operációs rendszer OPEN rutinjai ebből nyerik a szükséges információt, illetve CLOSE rutinjai ide írják be az esetleges módosításokat.

* Az ESZR/DOS-ban a főindex címe. Magasabb szintű főindexeket is megengedő operációs rendszereknél a legalacsonyabb szintű főindex címe.

Format-3 címke. Ez a címketípus akkor használatos, ha a file háromnál több területen helyezkedik el, s a Format-1 címkével már nem írható le. A Format-3 címke 13 független terület leírására használható. Csak azokra a lemezkeztegekre kerül rá, amelyeken háromnál több file-terület van. Címét a megfelelő Format-1 címke tartalmazza. A kulcsmező itt egybeolvad az adatmezővel.

Byte sorszáma	Címkemező tartalma
1-4	Kulcsazonosító, byte-onként 03 ₍₁₆₎
45	Formátumazonosító, esetünkben 3 (F3 ₁₆)

Az 5-135. byte-okon a címke 13 db egyenként 10 byte-os mezőt tartalmaz. Ezek felosztása megegyezik a Format-1 címkében levő, file-területeket leíró mezők felosztásával. Végül a 136-140. byte-okon a következő Format-3 címke címe található.

Format-4 címke. Ez a címketípus magát a VTOC-t definiálja. Ez egyben a VTOC első rekordja is, amely a mágneslemez inicializálásakor keletkezik.

Kulcsmező:

Byte sorszáma	Címkemező tartalma
1-44	Kulcsazonosító, esetünkben 04 ₍₁₆₎ byte-onként

Adatmező:

Byte sorszáma	Címkemező tartalma
45	Formátumazonosító, esetünkben 4 (F4 ₁₆)
46-50	A DOS nem használja
51-52	A VTOC szabad rekord helycinek száma
53-56	A következő tartalék sáv címe

Byte
sorszám

Cimkemező tartalma

57-58	Tartalék sávok száma
59	VTOC indikátorok
60	Tartalma 00 ⁽¹⁶⁾
61-62	Rezervált mező. A DOS nem használja
63-76	A használt mágneslemezegység paraméterei
77-105	Rezervált mező. A DOS nem használja
106	01 ⁽¹⁶⁾
107	00 ⁽¹⁶⁾
108-111	A VTOC kezdőcíme
112-115	A VTOC végének címe
116-140	Rezervált mező

c) Felhasználói címkék

Felhasználói címkéket csak szekvenciális és direkt feldolgozásnál használunk. Ezek nem kerülnek be a VTOC-be, de a file-hoz tartozó minden lemez-csomagon megjelennek. Feldolgozásukról a felhasználónak kell gondoskodnia, a rendszer csak a címkék olvasását végzi el.

A felhasználói címkék egységesen 4 byte-os kulcsmezőből és 80 byte-os adatmezőből állnak. Maximálisan 8 felhasználói címke helyezhető el egy ilyen címkecsoportban. Alább megadjuk az egyes mezők tartalmát.

Kulcsmező:

Byte
sorszám

Cimkemező tartalma

1-3	Címkeazonosító, esetünkben UHL vagy UTL
4	Felhasználói címke sorszáma a címkecsoportban

Adatmező:

Byte
sorszáma

Cimkemező tartalma

1-3	Cimkeazonosító, esetünkben UHL vagy UTL
4	Felhasználói címke sorszáma a címkecsoportban
5-80	Tetszőleges felhasználói információ Nyomtatható karakterek vagy szóközők

Az eddigiekben megközelítően pontos leírást adtunk a mágneslemez címkéről. Mivel a lemez igen bonyolult szervezésű tároló, érthető, hogy címke-rendszere is összetettebb, hiszen sokkal több információt kell tartalmaznia, mint a soros elérésű tárolók címkeinek. Az 5. táblázat a lehetséges címke-használat eseteit mutatja.

5. táblázat

Mágneslemez címkék használata

A file felépítése	Cimkék
Szekvenciális	Format-1; Format-3 (1 vagy több)
Indexelt szekvenciális	Format-1; Format-2 (1 db)
Direkt	Format-1; Format-3 (1 db)

5. A csatornarendszer

5.1 INPUT-OUTPUT MŰVELETEK

A perifériális egységek bevezetésével megoldódott a központi egységgel folytatandó adatforgalom lebonyolítása: a perifériák oldaláról hozzáférhetővé vált a központi egység a felhasználó számára. Az egyes perifériákat azonban valahogyan csatlakoztatni kellett a központi egységhez; ezt a feladatot látja el a csatornarendszer, amely egyben az adatátviteli műveletek vezérlését is elvégzi. A csatornák tehát a processzornál eggyel alacsonyabb szintű vezérlőegységeknek tekinthetők, amelyek kizárólag input-output műveletekkel foglalkoznak. Csatornarendszer használatával lehetővé vált az, hogy a program feldolgozása, valamint az egyes input-output műveletek időben egyszerre, egymással párhuzamosan folyjanak. A processzor ugyanis nem foglalkozik a teljes adatátvitel vezérlésével, hanem mintegy átruházza azt a csatornarendszerre. Ennélfogva nincs is tudomása az input-output művelet állapotáról egészen addig, amíg a csatornarendszertől jelzést nem kap az átvitel befejezéséről. Az átviteli művelet befejezésének oka lehet az, hogy megtörtént a teljes adatmennyiség átvitele, vagy olyan hiba lépett fel az input-output rendszerben, amely lehetlenné tette az átvitel folytatását. E fejezet feladata az, hogy betekintést nyújtson a csatornarendszer elvi működésébe, az input-output műveletek végrehajtásába.

A csatorna és a hozzá csatlakoztatott periféria (perifériavezérlő egység, perifériák) között a teljes adatforgalom a szabványos interface-en keresztül zajlik le. Az interface tulajdonképpen egy kábelköteg, amely magába foglalja az információátvitelt végző vezetékeket, valamint az átvitelt vezérlő, kiszolgáló jelek vezetékeit. Az interface szabványosítása lehetővé teszi, hogy az ESZR-en belül bármely periféria bármely csatornához csatlakoztatható, jöllehet bizonyos elrendezések az eltérő sebességviszonyok következtében kerülendők

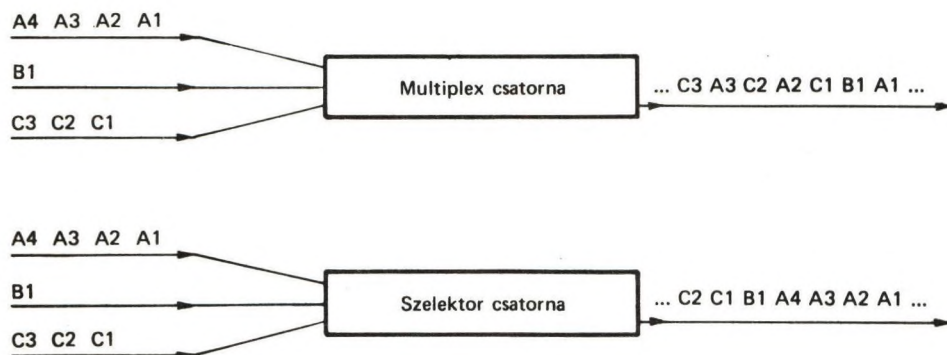
(lásd 99. old.). Az interface-nek a csatorna oldalán és a periféria oldalán is megvan a maga elektronikus hálózata, amely kiadja, fogadja, értelmezi az egyes vezérlőjeleket, valamint előkészíti az átviendő byte-okat az interface számára. Egyszerre csak az egyik irányban lehetséges adatátvitel: az input és output művelet időben kölcsönösen kizárja egymást egy adott csatornára nézve.

5.2 A CSATORNÁK FAJTÁI

A csatornarendszerbe tartozó csatornákat működési elvük alapján két csoportra osztjuk: multiplex és szelektor csatornák. Ezek maximális száma az adott géptípustól függ; az erre vonatkozó adatokat az 1. táblázat tartalmazza. Nagy vonalakban az alábbi módon jellemezhetjük az egyes csatornafajtákat:

- a multiplex csatorna egyszerre több (lassu) perifériával foglalkozik, és egy-egy lépésben egy-egy byte-ot visz át perifériánként;
- a szelektor csatorna egy lépésben a teljes adatblokkot átviszi, és egyszerre csak egy perifériával foglalkozik.

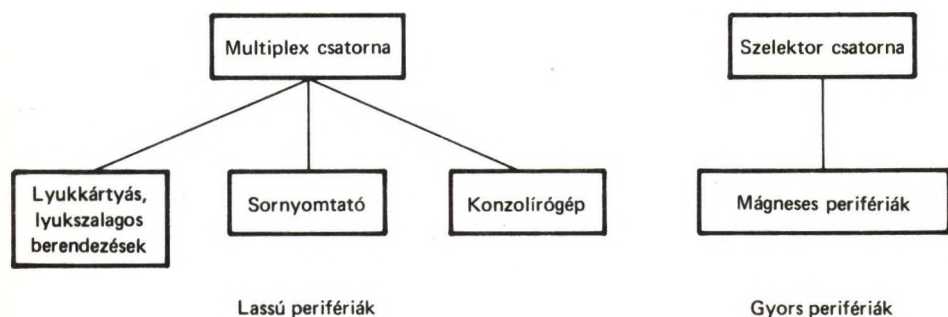
Mindezt igen leegyszerűsített formában a 22. ábra szemlélteti. Az A, B, C jelű perifériákról a jelzett sorrendben érkeznek a rekordok.



22. ábra. Multiplex és szelektor csatorna működési elve

Mint látható, a multiplex csatornának meg kell valósítania azt is, hogy a különböző perifériákról érkező rekordok az operatív tárnak különböző területeire kerüljenek, és a nemkívánatos keveredést elkerüljék. E feltételek teljesítése azzal jár együtt, hogy a multiplex csatornának viszonylag sok "haszontalan", kiszolgáló információt kell kezelnie a sikeres átvitel érdekében. Emiatt a multiplex csatorna lassabb működésű, mint a szelektor csatornák.

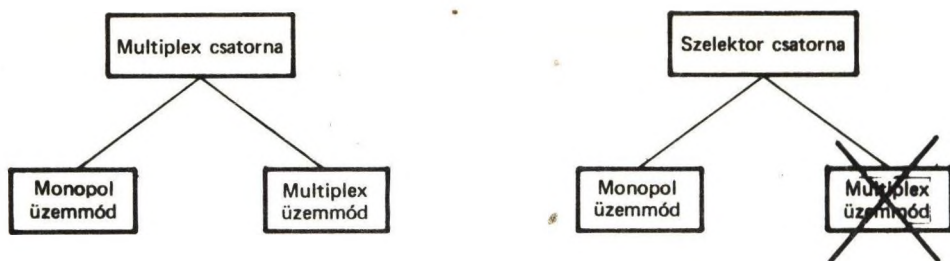
Az 1. táblázatból kitűnik, hogy valamennyi ESZR modell egy multiplex és két vagy több szelektor csatornával tud dolgozni. Ez azzal függ össze, hogy a multiplex csatornára igen sok periféria csatlakoztatható, esetleg több száz is. A szelektor csatornák viszonylag kevesebb perifériát tudnak kiszolgálni, de átviteli sebességük nagyobb. Mindezek alapján a javasolt csatlakoztatás az alábbi:



A fenti elrendezést nem kötelező betartani, de az eltérő sebességviszonyok mellett a rendszer így adja a legkedvezőbb input-output tulajdonságokat a tapasztalatok szerint. Megjegyezzük, hogy a perifériák működési sebessége tekintetében "lassu" perifériáknak minősülnek a viszonylag lassu perifériák, tehát például a lyukkártyaolvasó a mágneslemezhez képest lassu periféria, jóllehet szemmel láthatóan eléggé gyors működésű.

5.2.1 A CSATORNARENDSZER ÜZEMMÓDJAI

A csatornarendszerben kétféle üzemmód fordul elő: monopol és multiplex üzemmód. Az egyes csatornafajták és az üzemmódok lehetséges kapcsolatát a következő rajz szemlélteti.



Monopol üzemmódban a csatorna az adatátviteli művelet teljes időtartama alatt logikai és elektronikus kapcsolatban áll a megfelelő perifériával: a periféria mintegy monopol helyzetben van a csatorna erőforrásainak kihasználásában. A monopol üzemmódban átvihető legkisebb adatmennyiség az egy blokk. Multiplex üzemmódban a csatorna egy byte átvitele után időlegesen felfüggeszti logikai és elektronikus kapcsolatát az adott perifériával, és a sorrendben következő perifériához fordul, ha az input-output szolgáltatkérést jelentett be. Ily módon az egyes perifériákról érkező byte-ok egymás közt keveredve jelennek meg; ezek szétválogatása, osztályozása a csatorna feladata.

A kívánt üzemmódot a periféria szabja meg: multiplex üzemmód esetén egy byte átvitele után lekapcsolódik a csatornáról, monopol üzemmód esetén mindvégig fenntartja a kapcsolatot a csatornával. Mindezek alapján nyilvánvaló, hogy a mágneses perifériák számára csak a monopol üzemmód kedvező: adatátviteli sebességük olyan nagy, hogy a byte-onkénti lekapcsolódást nem tudják adatvesztés nélkül végrehajtani. Többek között ezért javasolják a mágneses perifériák üzemeltetését szelektor csatornáról.

5.3 AZ INPUT-OUTPUT RENDSZER CÍMZÉSE

Mielőtt a központi egység elindíthatná az adatátvitelt valamelyik perifériával, ki kell választania az input-output rendszer azon elemeit, amelyek részt vesznek a műveletben. Ez a kiválasztás a kívánt periféria megcímzésével érhető el; a cím ezután tartalmazza az érintett csatorna, a perifériavezérlő egység és a periféria címét. A kívánt adatátviteli művelet csak akkor kezdődhet

el, ha az illető periféria vagy csatorna nincs elfoglalva egy korábban elkezdődött input-output művelet végrehajtásával. Ellenkező esetben meg kell várni, hogy az előbb elindított adatátvitel véget érjen.

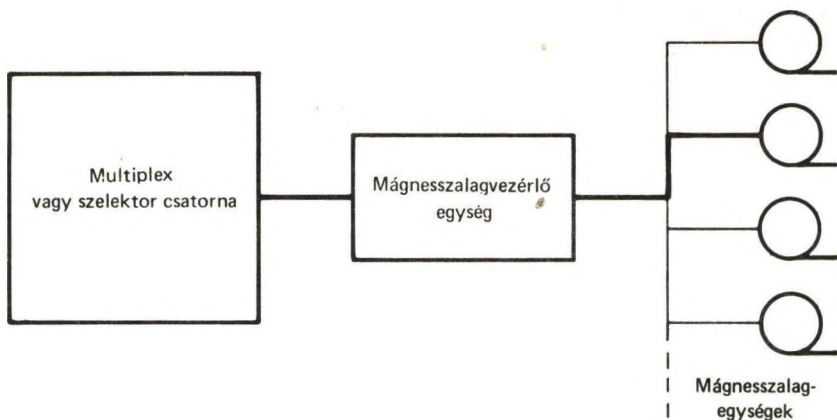
Mivel minden egyes csatornához több periféria csatlakoztatható, szükség van az egyes perifériákkal folytatott input-output műveletek szétválasztására, elkülönítésére. Erre szolgálnak az ún. alcsatornák, amelyek feladata az adatátvitelre vonatkozó összes lényeges információ tárolása. Az alcsatornák száma eltérő a multiplex és szelektor csatorna esetén:

A multiplex csatorna több alcsatornával rendelkezik, a szelektor csatornának csak egy alcsatornája van.

Minden egyes alcsatornához egy vagy több periféria rendelhető. Ennek alapján az alcsatornák lehetnek monoperifériálisak vagy multiperifériálisak. A monoperifériális alcsatorna mindig csak egy és ugyanazon perifériára vonatkozó információt tárol, a multiperifériális alcsatorna minden időpontban annak a perifériának az átviteli információját tárolja, amellyel a csatorna éppen input-output műveletet végez.

Monoperifériális alcsatornát használnak azok a perifériák, amelyek perifériavezérlő egysége is monoperifériális (például a lyukkártyaolvasó vezérlőegysége csak egy lyukkártyaolvasó berendezést irányít stb.). Multiperifériális alcsatornát használhatnak azok a perifériák, amelyek perifériavezérlő egysége is multiperifériális (például a mágnesszalag vezérlőegységére több mágnesszalagegység kapcsolható, és ezek egy közös - multiperifériális - alcsatornával dolgozhatnak). Megjegyezzük, hogy több perifériát irányító vezérlőegység perifériái is használhatnak külön-külön egy-egy monoperifériális alcsatornát.

Felmerül a kérdés: vajon nem keveredik-e össze a multiperifériális alcsatornában az egyes perifériákra vonatkozó átviteli információ. Hogy ez miért nincs így, azt egy példán világítjuk meg. Tekintsük az alábbi elrendezést! Feltetelezzük, hogy mind a négy mágnesszalagegység ugyanazt a multiperifériális alcsatornát használja.



23. ábra. A csatornák szerepe mágnesszalagos átvitelnél

Ha most valamelyik perifériával input-output műveletet kezdünk, az erre vonatkozó átviteli információ bekerül a hozzá rendelt multiperifériális alcsatornába. Mivel azonban a mágnesszalagegység csak monopol üzemmódban tud dolgozni, mindaddig nem kapcsolódik le a csatornáról, amíg egy teljes blokkot át nem vitt. Nem fenyeget tehát az a veszély, hogy az input-output művelet végrehajtása alatt valamelyik másik mágnesszalagegység szintén adatátviteli műveletet kezd, és így felülírja a még szükséges átviteli információt! A multiperifériális alcsatorna erőforrásait tehát időben egymás után használják a hozzá rendelt perifériák, így nem képesek egymás átviteli információját felülírni. Mindez természetesen nem érvényes a multiplex üzemmódban dolgozó perifériákra: ott a teljes átviteli művelet időben szétdarabolódik, byte-onként játszódik le, éppen ezért szükséges az, hogy minden egyes multiplex módban működő periféria saját önálló alcsatornával rendelkezzen. Az alcsatornák az operatív tár erre a célra fenntartott területén vannak, számuk és elhelyezkedésük az adott számítógéptől függ.

Az input-output rendszer címzésénél tehát az alábbi címeket kell feltétlenül megadni:

- a csatorna címe,
- az alcsatorna (perifériavezérlő egység) címe,
- a periféria címe.

Az input-output rendszer minden egyes eleme csak azt a címet kapja meg, amelyre a további kiválasztás során szüksége van. Így például a csatorna a teljes input-output rendszer címét megkapja, a perifériavezérlő egység már csak a saját címét és a periféria címét, maga a periféria csak a saját címét kapja dekódolt formában (vezérlőjelekké alakítva) a 24. ábra szerint:



24. ábra. Az I/O rendszer címzése

Az input-output rendszerre vonatkozó címek kétbyte-osak. Az I/O rendszer címének felépítése:



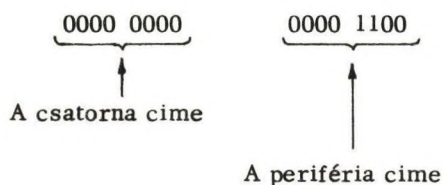
A periféria címe két részre osztható: a perifériához rendelt alcsatorna címére és magára a periféria címére. Az alcsatorna címében a legnagyobb helyértékű bit dönti el, hogy monop perifériális vagy multiperifériális alcsatornáról van-e szó:

monoperifériális alcsatornánál ez a bit 0,

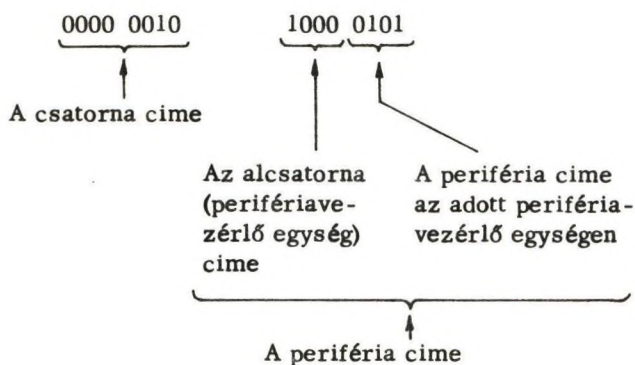
multiperifériális alcsatornánál ez a bit 1.

Monoperifériális alcsatornák esetén az alcsatorna (perifériavezérlő egység) címét együtt értelmezzük a perifériális egység címével, multiperifériális al-

csatornánál külön perifériavezérlő egység címről beszélhetünk. Az elmondottakat megvilágítja az alábbi példa:



(Tulajdonképpen a 0000 0000 című csatorna 0000 alcsatornájának 1100 című perifériájáról van szó; az alcsatorna monop perifériális.)



(Tulajdonképpen a 0000 0010 című csatorna 1000 című periféria vezérlő egységének 0101 című perifériájáról van szó; az alcsatorna multiperifériális.)

A csatornák címzése a következő:

0000 0000	-	multiplex csatorna
0000 0001	-	1. szelektor csatorna
0000 0010	-	2. szelektor csatorna
0000 0011	-	3. szelektor csatorna
0000 0100	-	4. szelektor csatorna
0000 0101	-	5. szelektor csatorna
0000 0110	-	6. szelektor csatorna

A fentiekől eltérő csatornacímként érvénytelennek minősül.

Mivel minden egyes szelektor csatornának csak egy alcsatornája van, egy-egy szelektor csatorna felfogható egy-egy multiperifériális alcsatornaként. Tekintve, hogy a szelektor csatorna csak monopol üzemmódban dolgozik, és a megkezdett input-output műveletet folyamatosan hajtja végre, az átvitel végén alcsatornája felszabadul, így annak korábbi tartalmát egy másik perifériájára vonatkozó átviteli információval minden további nélkül felülírhatjuk.

5.4 INPUT-OUTPUT MŰVELETEK

Az input-output műveletek végrehajtása alapvetően három szinten történik:

- a processzor kiadja a szükséges input-output utasítást,
- a csatornarendszer ennek hatására kiadja a csatornaparancsot, amelyet a perifériavezérlő egység végrehajt,
- a perifériavezérlő egység kiadja a megfelelő vezérlőjeleket a megcímzett perifériának a csatornaparancs dekódolása után.

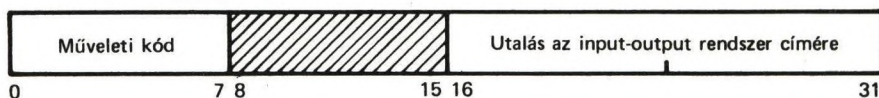
Az input-output utasítás és a csatornaparancsok magában a gépi programban találhatók, a csatornaparancsok alkotják az ún. csatornaprogramot. Ennek megfelelően analógia vonható a gépi program és a csatornaprogram között; mindkettő az operatív tárban helyezkedik el, csak a csatornaprogramban jóval kevesebb fajta művelet fordulhat elő, mint a gépi programban, hiszen az kizárólag az input-outputot vezérli.

Tekintsük át most már részletesebben, milyen információt tartalmaznak az egyes input-output utasítások és csatornaparancsok! Valamennyien kötött (standard) formátumúak, és az egyes byte-oknak, illetve biteknek meghatározott jelentése, funkciója van.

Négyféle input-output utasítást különböztethetünk meg:

- SIO (Start Input-Output) - feladata az adatátviteli művelet elindítása;
- HIO (Halt Input-Output) - feladata az adatátviteli művelet megállítása;
- TIO (Test Input-Output) - az input-output rendszer állapotának vizsgálatára szolgál;
- TCH (Test Channel) - az illető csatorna állapotának vizsgálatára szolgál.

Valamennyi input-output utasítás egységesen négy byte hosszúságú, un. SI formátumu és privilegizált (lásd a 7. fejezetben). Az egyes byte-ok információ-tartalmát a 25. ábra szemlélteti.



25. ábra. Input-output utasítások felépítése

A műveleti kód (0-7. bit) azonosítja az input-output utasítást, vagyis megmondja, hogy SIO, TIO stb. műveletet kell-e végezni. A 8-15. bit tartalma az utasítás szempontjából közömbös, a gép nem veszi figyelembe. A 16-31. biten található az input-output rendszer címe a címzésnél megismert formában. Ahhoz, hogy valamely input-output utasítás végrehajtható legyen, az szükséges, hogy az utasítás érvényes műveleti kóddal rendelkezzen, és a címmezőben megadott érték ténylegesen létező perifériális egységre vonatkozzon. Például a csatorna címe ne legyen nagyobb $06_{(16)}$ -nál, hiszen a rendszerben nem létezik hét szelektor csatorna stb. Az input-output utasítás végrehajtását a gép befejezettnek tekinti, ha a csatornának sikerült kapcsolatot teremtenie a megcímzett perifériával, és az képes a csatornaparancsot végrehajtani. Ellenkező esetben az input-output utasítás végrehajtása megszakad, és un. programmegszakítás jön létre. A megszakítási ok azonosításáról, valamint a programmegszakítás kezeléséről a későbbiekben lesz szó.

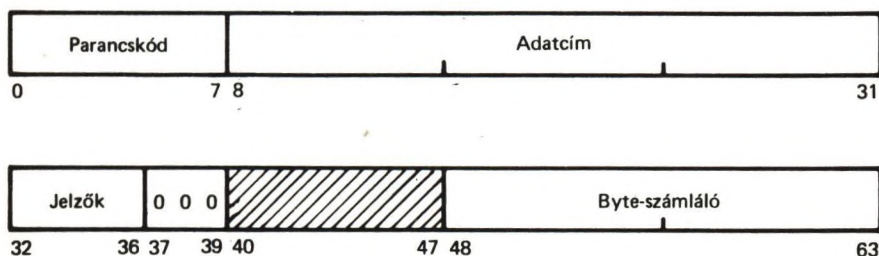
5.4.2 CSATORNAPARANCSONK, CSATORNAPROGRAM

A csatornaprogram egy vagy több, egységesen nyolc byte hosszúságu un. csatornaparancsszóból (Channel Command Word - CCW) áll. Az egy és ugyanazon csatornaprogramhoz tartozó CCW-k az operatív tárban közvetlenül egymás után, duplaszóhatáron helyezkednek el. A csatornaprogram végrehajtása azzal kezdődik, hogy a csatorna megkeresi az operatív tárban az illető input-output műveletet vezérlő első CCW címét, azaz a csatornaprogram kezdőcímét. Ezt az un. csatornacím-szóból nyeri (Channel Address Word - CAW), amely az operatív tár $72_{(10)}$ -es címén található, egységesen valamennyi géptípusra vonatkozólag.

A CCW-k és a CAW formátuma szintén kötött, az egyes CCW mezők helyes kitöltéséért a programozó a felelős. Míg azonban a CCW-k az operatív tárban gyakorlatilag bárhol elhelyezhetők, a CAW-t a csatorna mindig a $72_{(10)}$ -es tárcímen fogja "keresni", azaz minden esetben a $72-75_{(10)}$ rekeszek tartalmát fogja CAW-nek tekinteni!

Alább megadjuk a CCW és CAW formátumát és az egyes bitek jelentését.

CCW



CCW bit sorszáma

Funkció

0-7

Parancskód - a végrehajtandó input-output műveletet azonosítja, például írás, olvasás hátra stb.;

8-31

Adatcím - az input-output műveletben részt vevő operatív tár területének kezdőcíme;

32	Adatláncolás - ha ez a bit 1, akkor a csatorna-program következő CCW-je azonnal végrehajtódik, de a jelenlegi CCW parancskódjával;
33	Parancsláncolás - ha ez a bit 1, akkor a jelenlegi CCW végrehajtása után azonnal elkezdődik a következő CCW végrehajtása a saját parancskódjával;
34	Hosszhiba elnyomása - ha ez a bit 1, akkor nem keletkezik programmegszakítás, ha a byte-számláló értéke és az átvendő byte-ok száma nem egyenlő;
35	Adatátvitel elnyomása - ha ez a bit 1, nem kerül sor inputra;
36	Programvezérelt megszakítás - ha ez a bit 1, a csatorna programmegszakítási feltételt generál;
48-63	Byte-számláló - az átvendő byte-ok számát tartalmazza.

A 37-39. bit értékét 0-ra kell állítani, a 40-47. biteket a gép nem veszi figyelembe (ezek a mezők alkalmasak későbbi rendszerbővítés, változtatás céljára).

Megjegyzések. Az adatcím mindig arra az operatív tárbeli rekeszre (byte-ra) vonatkozik, amelynek tartalma az input-output műveletben részt vesz. Több byte átvitele esetén a csatorna byte-onként eggyel növeli az adatcím értékét, így az egy CCW-vel vezérelt átvitel egy összefüggő operatív tárterületet érint. Kivételt képez a címképzésnél az "olvasás hátra" mágnesszalag-művelet, ahol az adatcím kezdetben a feltöltendő operatív tárterület utolsó byte-jára mutat, majd a csatorna byte-onként eggyel csökkenti az adatcím értékét.

Adatláncolás alkalmazásával érhető el az, hogy az operatív tár nem összefüggő területei is részt vehetnek az adatátvitelben. Ugyanis a jelenlegi CCW parancs kódja lesz érvényben, de az adatcímét stb. a csatorna a következő CCW-ből meríti. Az átvitel folyamatos lesz, hiszen a "jelenlegi" CCW végrehajtása után a csatorna azonnal rátér a következő CCW-re.

CAW



0-3	Az operatív tár védelmi kulcsa - az adott SIO utasításhoz tartozó összes CCW védelmi kulcsa (a tárvédelemről bővebben a 6. fejezetben lesz szó);
4-7	Zérusokat tartalmaz, kitöltése kötelező;
8-31	Parancscím - az adott csatornaprogram első CCW-jének operatív tárbeli címe (duplaszócím!).

5.4.3 VEZÉRLŐJELEK

A vezérlőjeleket a perifériavezérlő egységek hozzák létre, miután a csatornától kapott parancsot dekódolták. A vezérlőjelekkel itt részletesen nem foglalkozunk; ezek mind valamilyen villamos jelek, és funkciójuk, elnevezésük az adott perifériális egységtől függ. A 6. táblázatban összefoglaltuk a lehetséges input-output műveletfajtákat, és feltüntettük azok bináris kódját is (parancskód). A kódban szereplő "m" ugynevezett módosító bit, értéke az adott perifériális egységre vonatkozó parancstól függ.

6. táblázat

Művelet	Bináris kód
Írás	m m m m m m 0 1
Olvasás	m m m m m m 1 0
Olvasás hátra	m m m m 1 1 0 0
Vezérlés	m m m m m m 1 1
Hibajelentés	m m m m 0 1 0 0
Ugrás a csatorna-programban	x x x x 0 0 0 0

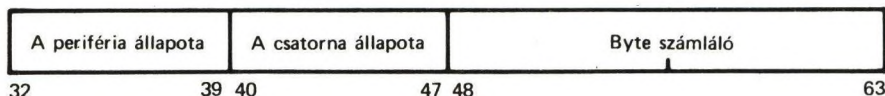
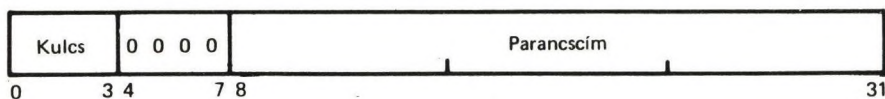
Megjegyzések. Vezérlésen olyan parancsokat értünk, amelyek általában nem járnak együtt a szokásos értelemben vett adatátvitellel, hanem azt mintegy kiszolgálják; például fejmozgatás, a mágneslemez pozicionálása stb.

Hibajelentés parancs hatására a megcímzett periférialvezérlő egység olyan információt továbbít a csatornának, amely jellemző a végrehajtott input-output művelet során bekövetkezett rendellenességekre (ha ilyenek voltak), továbbá az illető periféria állapotára.

Ugrás a csatornaprogramban - hatására az ebben a CCW-ben megadott operatív tárbeli címen levő duplaszón folytatódik a csatornaprogram. Ennek a műveletnek a bevezetésével elérhető, hogy a csatornaprogram ne feltétlenül egymást közvetlenül követő CCW láncból álljon. A x arra utal, hogy az illető bit értéke a parancskód szempontjából közömbös.

5.5 AZ INPUT-OUTPUT RENDSZER ÁLLAPOTÁNAK NYILVÁNTARTÁSA

Az input-output művelet normális befejezéséről, illetve a végrehajtása során keletkezett rendellenességekről, hibákról a periférialvezérlő egység tájékoztatja a csatornát. Ha a hiba olyan természetű, hogy lehetetlenné teszi az input-output művelet folytatását, akkor programmegszakítás keletkezik, és a hiba okát - kódolt formában - a rendszer nyilvántartja, rögzíti. Erre szolgál az ugynevezett csatornaállapot-szó (Channel Status Word - CSW), amely az operatív tár 64⁽¹⁰⁾-es címén kezdődő duplaszó. Felosztása kötött, az egyes biteknek meghatározott jelentése van. Az input-output rendszer állapotát leíró információt a csatorna mindig ebben a duplaszóban rögzíti. Alább megadjuk a CSW formátumát és az egyes bitek jelentését.



CSW bit sorszáma

Funkció

- | | |
|------|---|
| 0-3 | Az operatív tár védelmi kulcsa - az adott csatornaprogramra vonatkozik (lásd még CAW alatt); |
| 4-7 | Zérusokat tartalmaz; |
| 8-31 | Parancscím - ez a mező az utolsóként végrehajtott CCW címének nyolccal megnövelt értékét tartalmazza (tulajdonképpen a csatornaprogram következő CCW-jére mutat); |

32-39	A periféria állapota (állapotbyte);
32	"Figyelem" - konkrét jelentése a perifériától függ;
33	Állapotmódosító - általában a perifériavezérlő egység foglaltságát jelzi;
34	"Perifériavezérlő egység kész" - jelzi, hogy a perifériavezérlő egység felszabadult, jöllehet maga a periféria esetleg még nem;
35	"Foglalt" - a periféria foglalt valamely korábban elkezdett input-output művelet végrehajtásával;
36	"Csatorna kész" - jelzi, hogy az input-output műveletnek az adatátvitellel kapcsolatos része befejeződött, és az alcsatorna felszabadult;
37	"Periféria kész" - jelzi, hogy a periféria befejezte az input-output művelet végrehajtását, felszabadult, és képes a következő parancs végrehajtására;
38	Perifériahiba;
39	Periféria-rendellenesség - olyan eseményre utal, amely nem jelent hibás állapotot, de különbözik a szokásostól, például mágnesszalag-olvasásnál tape mark felfedezése stb.;
40-47	A csatorna állapota (állapotbyte);
40	Programvezérelt megszakítás - a csatorna olyan CCW-t talált, amelyben a 36. bit értéke 1;
41	Hosszhiba - a byte-számláló értéke kezdetben nem volt egyenlő az átviendő byte-ok számával (egyidejűleg a CCW 34. bitjének értéke 0, nincs hosszhiba-elnyomás);
42	Hiba a csatornaprogramban - érvénytelen CCW cím, érvénytelen parancskód stb. fennállását jelzi;
43	Operatív tárvédelem megsértése - a csatorna védett operatív tárterületre hivatkozik, és a védelmi kulcsok nem egyeznek (lásd a 6. fejezetben);
44	Adathiba a csatornában - a csatorna paritáshibát észlelt valamelyik byte-nál;

45	Vezérlési hiba a csatornában - az input-outputot érintő minden géphiba kiválthatja;
46	Hiba az interface-en - a csatorna érvénytelen jelet fedezett fel az interface-en;
47	Láncolási hiba - adatláncolásnál az input sebesség nagyobb, mint amilyen gyorsan a csatorna ki tudná szolgálni a perifériát;
48-63	Byte-számláló - általában az utolsónak végrehajtott CCW byte-számlálójának és a ténylegesen átvitt byte-ok számának különbségét tartalmazza.

Nagy vonalakban tehát egy input-output művelet az alábbi módon megy végbe:

- A vezérlőegység (processzor) értelmezi a gépi programban megjelenő input-output utasítást, és az abban foglalt input-output rendszer címet átadja a csatornának;
- A csatorna kiolvassa az operatív tár $72_{(10)}$ -es címén kezdődő csatorna-címest (CAW), ebből megtudja, hol kezdődik az operatív tárban a végrehajtható csatornaprogram;
- A csatorna kiolvassa az első CCW-t, és végrehajtja azt. Láncolás esetén azonnal rátér az utána következő CCW végrehajtására;
- A csatornaprogram végrehajtása után a csatorna egy programmegszakítási feltételt generál, és a CSW-ben (operatív tár $64_{(10)}$ -es címétől kezdődően) rögzíti az input-output rendszer állapotát. Ezzel az input-output művelet befejezettnek minősül.

Fenti elemzésben nem tértünk ki azokra az esetekre, amikor valamilyen hiba keletkezik az átvitel közben; ez a leírás egy tipikus input-output művelet lefolyását vázolja nagyon leegyszerűsített formában. Mialatt a csatornaprogram fut, természetesen folytatódhat a gépi program feldolgozása is - éppen ez a körülmény teszi nagyon indokolttá a csatorna-(vezérlő-) rendszer alkalmazását: a processzor és a csatornarendszer egymással szimultán dolgozik, ami jelentős futási idő megtakarítást jelent az adott program esetében.

6. Az operatív tár

6.1 AZ OPERATÍV TÁR SZEREPE A SZÁMÍTÓGÉPBN

Az operatív tár feladata az, hogy az éppen feldolgozandó információt fogadja, tárolja, illetve átadja a számítógép többi egységének. A Neumann-féle tárolt program koncepció értelmében az operatív tár egyaránt tárol programokat és adatokat, bináris ábrázolásmódban. Mivel a binárisan kódolt program utasításai és a binárisan ábrázolt adatok (számok, karakterek stb.) között a gép nem tud különbséget tenni, a felhasználónak kell nyilvántartania, hogy melyik operatív tárterületen találhatók a programjai, illetve az adatai. Mint később látni fogjuk, ezt a nyilvántartást magasabb szintű programozásnál (például valamely programozási nyelv használatánál) az operációs rendszer elvégzi a felhasználó helyett. Az operatív tár elnevezés onnan származik, hogy e tárterület rendeltetésszerűen mindig csak azokat a programokat és adatokat tartalmazza, amelyek éppen feldolgozásra kerülnek, vagy éppen feldolgozás alatt állnak (ellentétben a háttértárolókkal, amelyek tetszőleges időpontban feldolgozandó információt tárolnak).

Az operatív tár kapacitása (információtároló képessége Kbyte-ban kifejezve) korlátozott, és géptípusonként változó lehet. A programozónak célszerű mindig szem előtt tartania azt az irányelvet, hogy igyekezzen programjai és adatai számára minél kevesebb operatív tárterületet lefoglalni. Ez részben elérhető a program további futása során már fel nem használt rekeszek tartalmának felülírásával és egyéb, erre irányuló programozási megoldásokkal.

6.1.1 A TÁROLÁS ELVE

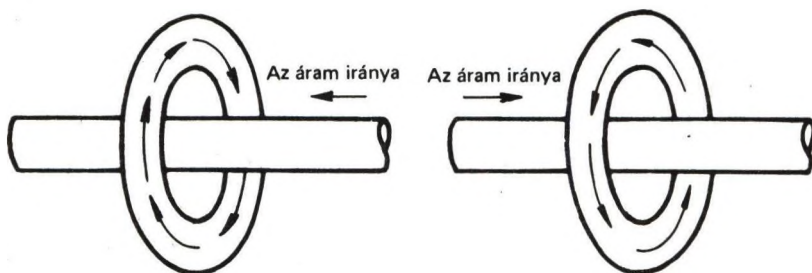
Mint az 1. fejezetben említettük, az információtárolás byte-onként történik. Minden egyes byte-nak sorszáma van, amelyet az illető byte címének tekintünk. A sorszámozás 0-tól kezdődik. A címtől függetlenül minden egyes byte azonos idő alatt érhető el az operatív tárban. Az információtárolás az alábbi elveken alapul:

- ferritgyűrűs tárolási elv,
- elektronikus tárolási elv.

Ferritgyűrűs információtárolásnál a ferritgyűrűk maradandó mágneses tulajdonságait használjuk ki, elektronikus adattárolásnál pedig olyan áramköröket alkalmazunk, amelyek megtartják valamely állapotukat mindaddig, amíg azt módosító információ nem érkezik.

a) Ferritgyűrűs tárolási elv

A ferritgyűrűs információtárolási elv alapja az a fizikai jelenség, hogy egy mágnesezhető közeg tetszőlegesen hosszú ideig megtartja mágneses állapotát, ha csak külső átmágnesező hatás nem éri. A ferritgyűrűs tárolónál ezt jól kihasználhatjuk: minden egyes bitnek megfeleltetünk egy ferritgyűrűt, amely kétféle mágnesezettségi állapotot vehet fel. Ugyanigy egy bit információtartalma is csak 1 vagy 0 lehet. A két állapot éles elkülönítése két ellentétes irányu mágnesezettség alapján lehetséges (lásd a 26. ábrát).

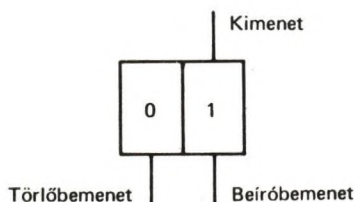


26. ábra. A ferritgyűrűs információtárolás elve

A mágnesezettség beállítását úgy képzelhetjük el, mint a mágnesezhető anyag rendezetlen mágneses állapotának rendezetté való átalakítását. Ezt az átalakítást a ferritgyűrűn keresztül haladó villamos vezetéken folyó áram végzi, saját mágneses tere segítségével. A mágnesező áram megszűnése után a ferritgyűrű továbbra is megtartja beállított mágneses állapotát.

b) Elektronikus tárolási elv

Az elektronikus információ-tárolási elv azon alapul, hogy minden egyes bitnek egy-egy elemi elektronikus áramkört feleltetünk meg, amely egy szabványos feszültségérték formájában tárolja a beírt információt. Például a logikai 1-nek +5 V, a logikai 0-nak 0 V stb. felel meg. Az ilyen áramköröket bistabil multivibrátoroknak, illetve e konkrét alkalmazásban flip-flopoknak nevezzük. Működésüket az alábbi rajz szemlélteti:



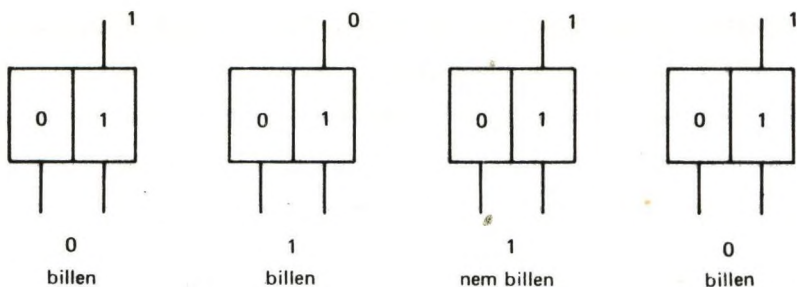
Az áramkör állapota (kimeneti feszültsége) igen egyszerűen határozható meg:

- a beíróbemenetre adott impulzus hatására a kimeneti feszültség szint logikai 1 értéket vesz fel;
- a törlőbemenetre adott impulzus hatására a kimeneti feszültség szint logikai 0 értéket vesz fel. Látható, hogy az áramkör mindenkor állapota nem függ az előző állapotától, hanem csupán a bemenetére adott impulzustól.

Példaként vizsgáljuk meg, milyen lesz az alábbi négybites áramkörrendszer viselkedése, ha a következő információ módosítást akarjuk végrehajtani:

kiindulási információ: 1 0 1 1

beírandó információ : 0 1 1 0



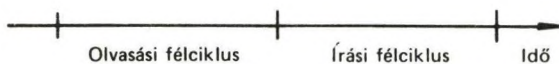
27. ábra. Négybites áramkörrendszer viselkedése

Látható, hogy csak azok az elemi áramkörök (flip-flopok) változtatják meg állapotukat, amelyeknél a beírandó információ nem egyezik meg az aktuális információtartalommal. Az állapotváltozást nem igénylő áramkörök is megkapják a bemeneti billentőimpulzust, de ez hatástalan az áramkör állapotára.

Nem lehet egyértelműen állást foglalni valamelyik tárolási elv mellett vagy ellene. A ferritgyűrűs tár előnye, hogy információtartalma az átmágnesező áram kikapcsolása után is megmarad, hátrányos tulajdonsága a viszonylag lassabb működés. Az elektronikus tár ezzel szemben gyorsabb, de a tápfeszültség kikapcsolása (kimaradása!) esetén információtartalma elvész, hiszen a tároló-áramkörök működtetéséhez állandó villamos energiaforrásra van szükség. Ujbóli bekapcsolás után az elemi áramkörök állapota előre nem becsülhető meg. A két-féle tárolási elv előnyeit kihasználhatjuk, ha a gyakran változó tartalmu tárrészeket elektronikus elemekből építjük fel, a kevesebb hozzáfordulásnak kitett memóriaterületet ferritgyűrűkkel valósítjuk meg.

6.1.2 AZ OPERATÍV TÁR MŰKÖDÉSE

Minden egyes tárhivatkozás alkalmával egy teljes, ún. operatív tárciklus játszódik le. A tárciklus két félciklusból (ütemből) áll, amelyek szigorúan kötött sorrendben követik egymást:



Az olvasási félciklus során a kiválasztott ferritgyűrűkből kiolvassuk az információt, az írási félciklus alatt pedig a kiválasztott ferritgyűrűbe információt írunk be. Az operatív tár működése tehát nagy vonásokban három fázisra bontható:

- a keresett ferritgyűrű kiválasztása,
- olvasási félciklus,
- írási félciklus.

a) A keresett ferritgyűrű kiválasztása

A teljes operatív tár ferritgyűrűkészlete úgy van kialakítva, hogy 9 gyűrű mindig egy logikai csoportba tartozik. Az elrendezés megfelel a byte-orientált információátvitelnek: 8 bit szolgál a hasznos információ tárolására, a 9. bit ellenőrző funkciót lát el (paritásbit). Minden egyes byte-nak saját címe van. Egy adott byte kiválasztása (kikeresése) az illető byte címe alapján történik.

A tárcím megjelenítésére és átmeneti tárolására szolgál az operatív tár címregisztere. Ebbe írjuk be a kiválasztani kívánt byte (byte-ok) címét, amely a teljes tárciklus alatt az aktuális tárcímet tartalmazza. A teljes tárciklus lezajlása előtt a címregiszter tartalmát megváltoztatni nem szabad! Tehát például olvasási félciklus után közvetlenül nem hivatkozhatunk egy másik tárcímrre, újabb hozzáfordulás a tárhoz csak az esedékes írási félciklus megtörténte után kezdődhet. Ez a megkötés azzal függ össze, hogy az olvasási félciklus után a kiolvasott tárrekesz tartalma nem jellemző annak sem korábbi, sem újabb tartalmára, így ezt a határozatlan állapotot meg kell szüntetni (erre szolgál az írási félciklus, lásd a c) pontban.)

A kiválasztás fizikailag a következőképpen megy végbe: minden egyes ferritgyűrűn két villamos kiválasztóvezeték halad keresztül. Az adott gyűrűt akkor és csakis akkor tekintjük kiválasztottnak, ha mindkét vezetékben egyidejűleg jelentkezik kiválasztó áramimpulzus. A módszer az ún. féláramu kiválasztás; egyik vezetékben folyó áram nagysága sem elegendő egyedül a mágátmágnesezéséhez, viszont a két vezeték összeadódó mágneses tere már képes átbillenteni a magot ellentétes mágnesezettségi állapotba. A kiválasztóáramokat szokás x és y irányu áramnak nevezni, feltéve, hogy a ferritgyűrűk mátrixszerűen helyezkednek el egy felületen.

b) Olvasási félciklus

A kiválasztóvezetékek mellett még egy harmadik vezetékek is áthalad valamennyi magon: ez az olvasóvezeték. Az egyes magok átbillenésük során áramimpulzust keltenek az olvasóvezetékben. Ezt használjuk ki az olvasási félciklusban: a kiválasztott ferritmagokat egységesen 0-nak megfelelő mágneses állapotba billentjük. Természetesen amely magok már előzőleg is 0 állapotban voltak, nem tudnak billenni, és ezek esetében hiányzik az áramimpulzus az olvasóvezetékben. Hogy a kiolvasott információt azonosítani tudjuk, most már csak azt kell figyelni, mely magok hoztak létre impulzust az olvasóvezetékben. Ezek információtartalma 1 volt, a nem billent magoké 0. Így állapíthatjuk meg egy memóriaterület (rekesz) információtartalmát. Mivel a ferritgyűrűk továbbra is megtartják mágneses állapotukat a 0-ra billentés után is, jól jegyezzük meg, hogy

az olvasási félciklus végén valamennyi kiválasztott mag logikai 0 állapotban van, függetlenül attól, hogy billent-e az olvasás során.

A fentiekből kitűnik, hogy az olvasási félciklus alatt bizonyos memóriaterület tartalmát megsemmisítettük, hiszen az olvasást csak ennek árán tudtuk megvalósítani. Többek között ezért nem szabad a címregiszter tartalmát megváltoztatni, mielőtt az írási félciklus is befejeződött volna. Ez gondoskodik ugyanis arról, hogy éppen ezt a kiválasztott memóriaterületet újból feltöltsük valamilyen információval.

Az olvasási félciklus alatt a kiolvasott információt az olvasóvezetékéről az operatív tár adatregiszterébe vezetjük, innen vezethetjük el további feldolgozásra például a processzorba stb.

c) Írási félciklus

Az írási félciklus alatt a kiválasztott és éppen csupa 0-t tartalmazó ferritmagokba információt írunk be. A beírt információ kétféle lehet az eredetét nézve:

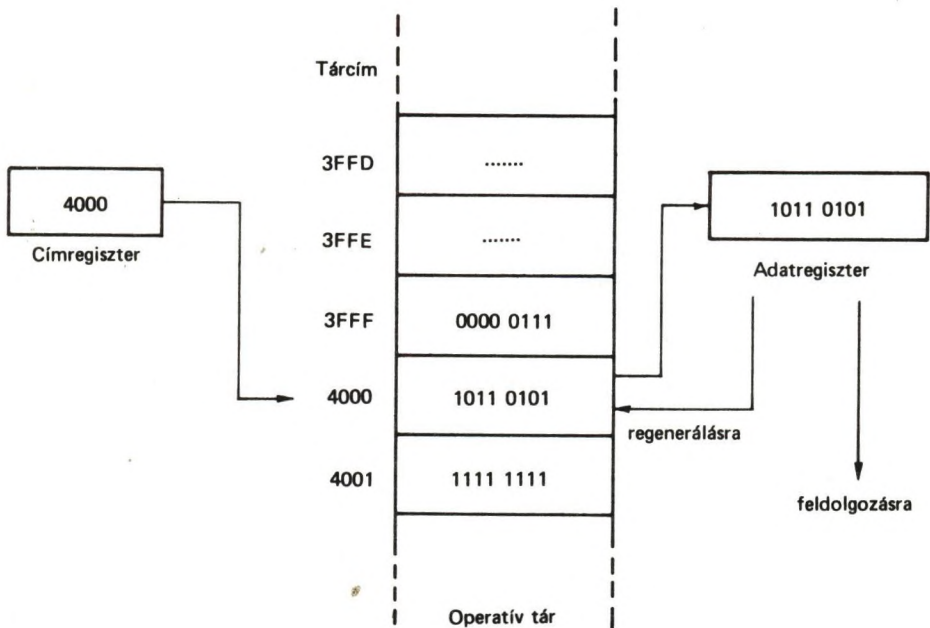
- az olvasási félciklus alatt kiolvasott információ,
- valamilyen új információ.

A magok átbillentését az írási félciklus során úgy oldjuk meg, hogy az 1-be állítandó magok egyidejűleg mind az x, mind az y irányú vezetékben áramimpulzust kapnak, a 0-ban tartandó magok csak x irányból kapnak áramot. Ez pedig nem elegendő a mag átbillentéséhez. Az ismertetett elrendezés az ún. 2, 5 dimenziós társzervezés.

Példaként tekintsünk át néhány jellegzetes tárműveletet!

1. Kiolvasás-regenerálás tárművelet

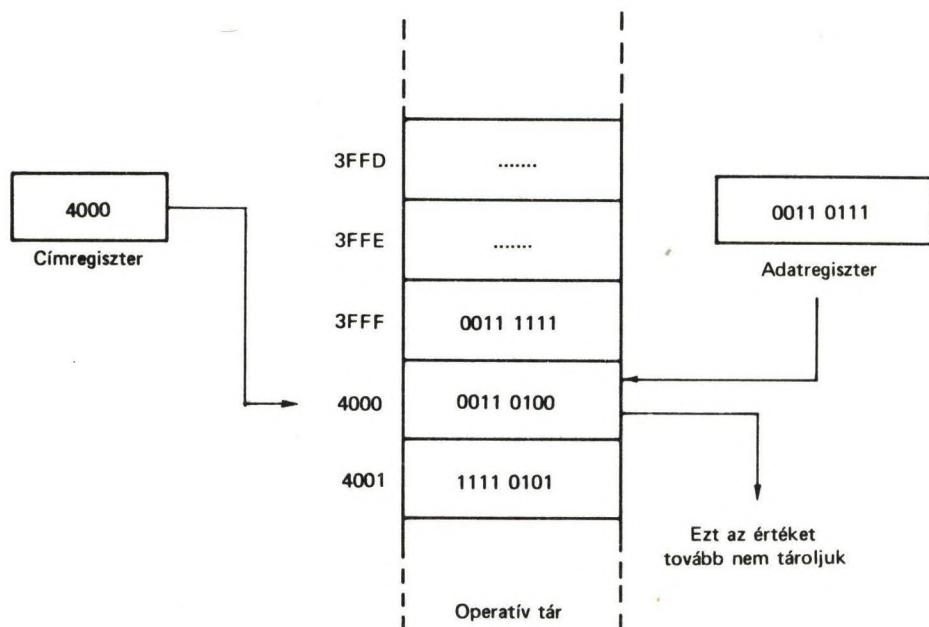
Az olvasási félciklus során a megcímzett rekeszekből kiolvassuk az információt, majd az adatregiszterből ugyanazt az információt az írási félciklus alatt visszairjuk. Ezzel a kiválasztott rekeszek tartalmát visszaállítottuk, regeneráltuk. A teljes tárciklus lezajlása után a rekeszek tartalma ugyanaz, mint a tárciklus előtt volt. A műveletet a 28. ábra szemlélteti.



28. ábra. Kiolvasás-regenerálás tárművelet

2. Törlés-beírás tárművelet

Az olvasási félciklus során a megcímzett rekeszekből kiolvassuk az információt, majd az adatregiszterben előkészített új információt az írási félciklus alatt visszairjuk. Ilyenkor a kiolvasott információt nem tároljuk további feldolgozás céljára. A műveletet a 29. ábra szemlélteti.



29. ábra. Törlés-beírás tárművelet

A 4000 cím új tartalma tehát 0011 0111 lesz, korábbi tartalma (0011 0100) pedig elvész.

6.2 STANDARD TÁRREKESZEK

A felhasználói programok és adatok - mint említettük - elvileg bárhol elhelyezhetők az operatív tárban. Kivétel ez alól néhány rekesz, amelyek funkciója kötött, és információtartalma a programok futásához nélkülözhetetlen. A

csatornarendszerek tárgyalásakor már megismerkedtünk néhány kötött funkció-
ju rekesszel (például CAW, CSW), a többieket a 7. fejezetben részletezzük.

Alább feltüntettük a standard tárrekeszek címét, hosszát és elnevezését.

Tárcim	Hossz	Rendeltetés
0 0000 0000	duplaszó	Kezdeti programbetöltő PSW
8 0000 1000	duplaszó	Kezdeti programbetöltő 1. CCW
16 0001 0000	duplaszó	Kezdeti programbetöltő 2. CCW
24 0001 1000	duplaszó	Külső megszakítás <u>régi</u> PSW-je
32 0010 0000	duplaszó	Supervisor-hívás <u>régi</u> PSW-je
40 0010 1000	duplaszó	Programmegszakítás <u>régi</u> PSW-je
48 0011 0000	duplaszó	Géphiha miatti megszakítás <u>régi</u> PSW-je
56 0011 1000	duplaszó	Input-output megszakítás <u>régi</u> PSW-je
64 0100 0000	duplaszó	CSW
72 0100 1000	szó	CAW
76 0100 1100	szó	Nem használatos
80 0101 0000	szó	Timer
84 0101 0100	szó	Nem használatos
88 0101 1000	duplaszó	Külső megszakítás új PSW-je
96 0110 0000	duplaszó	Supervisor-hívás új PSW-je
104 0110 1000	duplaszó	Programmegszakítás új PSW-je
112 0111 0000	duplaszó	Géphiha miatti megszakítás új PSW-je
120 0111 1000	duplaszó	Input-output megszakítás új PSW-je
128 1000 0000		Diagnosztikai terület *

Az egyes kiegészítő operatív tárterületek nagysága a főmemória^{***} kapacitásának függvényében változhat, továbbá ezek elhelyezkedése is géptípusonként különböző lehet.

* A diagnosztikai terület hossza géptípusonként változó lehet.

*** A felhasználó által címezhető, adatok és programok tárolására szolgáló operatív tárterület.

6.3 A FŐMEMÓRIA INFORMÁCIÓVÉDELME

Mivel a nagyobb ESZR modellek akkor üzemeltethetők csak gazdaságosan, ha a főmemóriában egyszerre több program fut (multiprogramozás, lásd 41. ábra), meg kell akadályozni, hogy az egyes programok egymás memóriaterületének információtartalmát módosíthassák. Hasonlóképpen gondoskodni kell arról is, nehogy valamely felhasználói program "felülírhasssa" a gépet üzemeltető operációs rendszer területét, hiszen ez adott esetben lehetetlenné tenné a programfuttatást. A fenti feladatokat látják el az ún. memóriakulcsok. A memóriakulcsok egységesen ötbitesek, és egy erre a célra szolgáló igen gyors működésű memóriakulcstárban helyezkednek el. A kulcsok felépítése:



A kulcsok bináris számokat tartalmazhatnak, ezenkívül megvan a saját paritás-bitjük (K) is. Az utolsó bit értéke a memóriavédelmi módot határozza meg:

VM = 0 - írásvédelem,

VM = 1 - írás és olvasás elleni védelem.

Látható, hogy megfelelően beállított VM bit segítségével nemcsak egy bizonyos memóriaterület tartalmának felülírását lehet megakadályozni, hanem azt is, hogy valamely program hozzáférjen számára tiltott területen levő információhoz. Csak olvasás elleni védelem nem lehetséges. A kulcsok numerikus része négybites, ez összesen 16 különböző kulcsérték beállítását teszi lehetővé ($2^4 = 16$). Minden program, illetve csatornaprogram saját kulccsal rendelkezik, és ez a kulcs a PSW, CAW, CSW megfelelő bitpozícióin található. A PSW, CAW és CSW azonban nem tárolja a kulcs VM bitjét, csak a numerikus részt (négy bit).

A memóriavédelem működése: Ahányszor a főmemóriához kell fordulni, a gép összehasonlítja a tárhivatkozást végző program kulcsát a tárrekeszhez

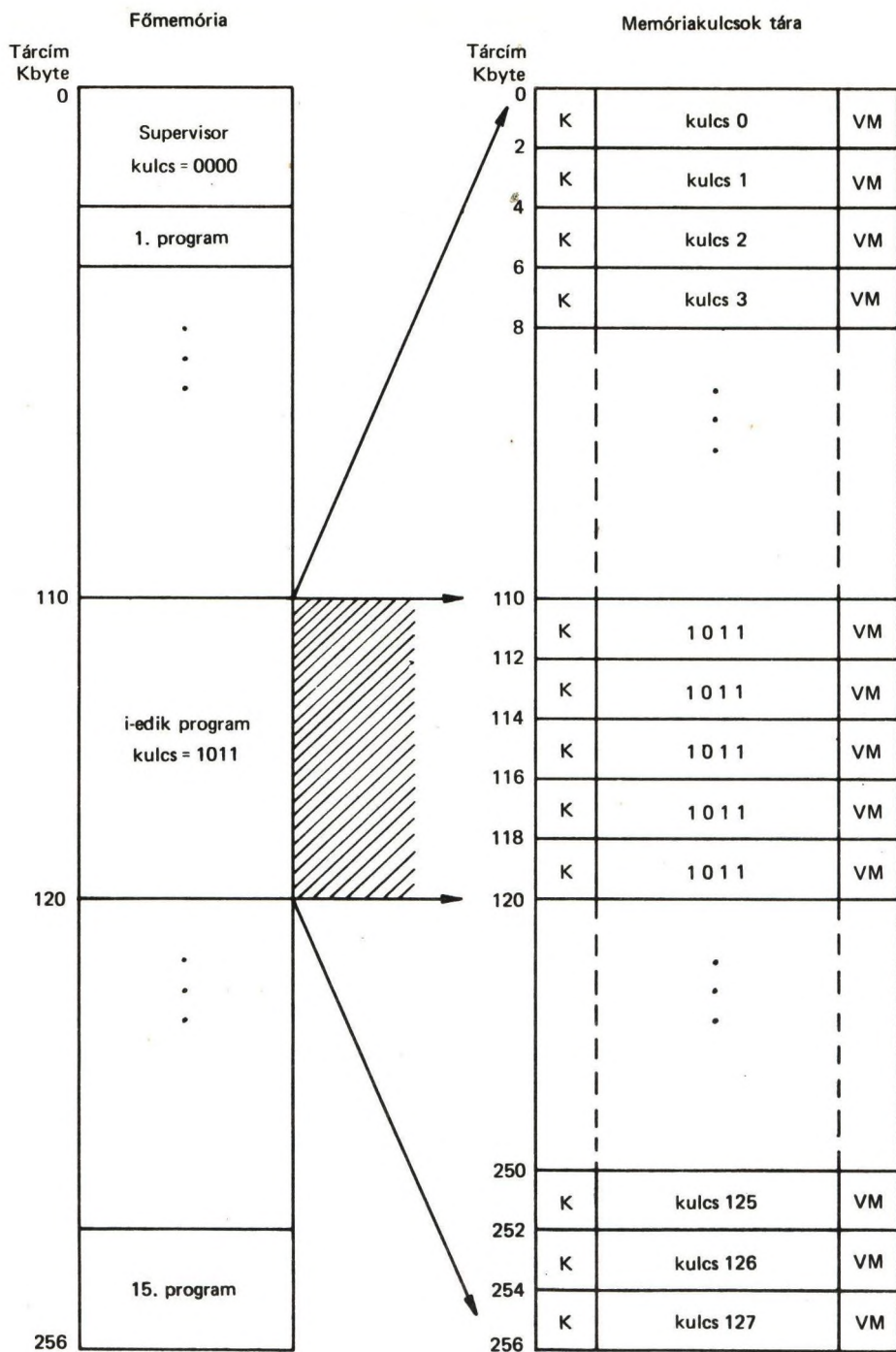
rendelt kulccsal. A numerikus részek nem egyezése bizonyos esetekben programmegszakításhoz vezet, más esetekben nem befolyásolja a tárművelet végrehajtását. Az előforduló eseteket a 7. táblázat tartalmazza.

7. táblázat

VM bit értéke	Numerikus részek	A tárhivatkozás hatása
0	Egyeznek	Az írás-olvasás engedélyezett
0	Nem egyeznek	Az olvasás engedélyezett, az írási kísérlet programmegszakításhoz vezet
1	Egyeznek	Az írás-olvasás engedélyezett
1	Nem egyeznek	Az írási-olvasási kísérlet programmegszakításhoz vezet

A főmemóriakulcsok beállítása speciálisan erre a célra szolgáló gépi utasítással történik. Védelmi szempontból a főmemória 2 Kbyte-os (2048 byte) lapokra oszlik, és minden laphoz külön kulcs rendelhető. Mivel azonban csak 16 különböző értékű kulcsot lehet előállítani, ugyanaz a kulcs több laphoz is rendelhető. Általában minden programnak, amely az operatív tárban tartózkodik, saját külön kulcsa van. Összesen tehát legfeljebb 16 különböző program lehet a tárban, illetve ennyi védhető. Egyes operációs rendszerek korlátozzák a párhuzamosan futó programok számát, ekkor természetesen nem használnak mind a 16-féle kulcs. A programok közötti tárkiosztásról bővebben szólunk a 9. fejezetben.

A 30. ábrán azt az esetet tüntettük fel, amikor az *i*-edik program számára a 110-120 Kbyte főmemória-terület érhető el. Ha ez a program e területen kívül eső tárrekeszre hivatkozik, akkor a VM bit értéke dönti el, mi lesz a tárhivatkozás hatása (lásd a 7. táblázatot). Megjegyezzük, hogy a zérus értékű kulccsal ellátott program valamennyi főmemória-területhez hozzáférhet. Ezért ezt a kulcsot az operációs rendszer felügyelőprogramja, a supervisor használja.



30. ábra. Memóriavédelmi kulcsok használata

6.4 AZ OPERATÍV TÁR FŐ FUNKCIONÁLIS RÉSZEI

Azért, hogy az eddigiekben megismert, a felhasználó által is címezhető operatív tárrészt ne terhelje az a sok egyéb információ, ami például input-output stb. műveletek vezérléséhez szükséges, létrehoztak olyan tárterületeket, amelyek a felhasználó által nem címezhetők, és kötött funkciójuk van. Ezek az operatív tár kibővítésének tekinthetők, elnevezés szempontjából azonban az operatív tárhoz tartozónak minősülnek. Hasonlóképpen külön tárterületet alkotnak a felhasználó által is címezhető regiszterek, amelyek működése egyes modellek esetében lényegesen gyorsabb a főmemóriáénál. Mindezek alapján az operatív tár fő funkcionális részei az alábbiakban foglalhatók össze:

- Főmemória: a felhasználó által címezhető terület. Itt tartózkodnak a futó programok, a hozzájuk tartozó adatok stb. Ugyancsak ebben találhatók a standard tárrekeszek is.
- Multiplex memória: a multiplex csatorna területe, a felhasználó által nem címezhető. Ebben találhatók a multiplex csatorna alcsatornái, különböző pufferek, input-output műveletek vezérlő információja stb.
- Regiszterblokk: a felhasználó által részben címezhető terület. Itt foglal helyet a 16 db általános célú regiszter (számozásuk 0-tól 15-ig), a 4 db lebegőpontos regiszter lebegőpontos operandusok tárolására (számozásuk 0, 2, 4, 6), valamint egyéb regiszterek. Az általános célú és lebegőpontos regiszterekre a programból is lehet hivatkozni. Ugyancsak itt találhatók a szelektor csatornák alcsatornái is.

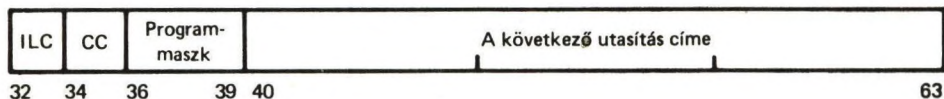
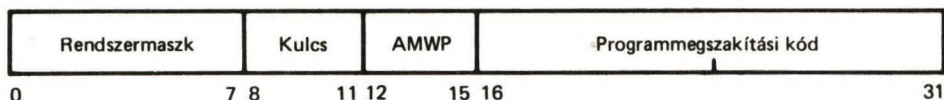
7. A központi egység (processzor)

7.1 A PROCESSZOR FELADATA

A központi egység feladata az egész számítógép működésének irányítása. Elvégzi a programutasítások kiolvasását az operatív tárból, értelmezi, majd végrehajtja őket. Az input-output utasítások kiolvasása után az input-output művelet vezérlését átadja a csatornarendszernek, ő maga a program következő utasításának végrehajtásával foglalkozik. A központi egységben található az aritmetikai-logikai egység is, amely a számítógép tulajdonképpeni művelet-végző egysége. A processzor mindig a hardware-architektúra legmagasabb rendű eleme, a számítógép összes többi egysége ennek van alárendelve. E fejezetben vázlatosan ismertetjük a processzor elvi működését, a feldolgozás menetét.

7.1.1 A PROGRAM ÁLLAPOTÁNAK NYILVÁNTARTÁSA

A futó program mindenkori állapotának nyilvántartására szolgál az ugynevezett programállapotszó (Program Status Word, PSW). Funkciója hasonló a már megismert CSW funkciójához, de azzal ellentétben nem a csatornaprogram és az input-output rendszer állapotáról tartalmaz információt, hanem a processzor által végrehajtandó programról. Az aktuális PSW tartalma állandóan változhat, ezért azt a processzor ún. PSW regiszterében tároljuk (dinamikus tároló!). Ha a programot valamiért meg kell szakítani - például hibás adatok, eredmények stb. miatt - akkor az aktuális PSW-t az operatív tár meghatározott rekeszébe írjuk be, a processzor PSW regiszterébe pedig a futtatandó program (például hibakereső vagy elhárító rutin) PSW-je kerül. Fentiek jobb megvilágítására tekintsük át, milyen információtartalma van a PSW egyes bitjeinek. A PSW formátuma kötött, hasonlóan a többi vezérlőszó formátumához, hossza nyolc byte, tehát duplaszó.



PSW bit sorszáma

Funkció

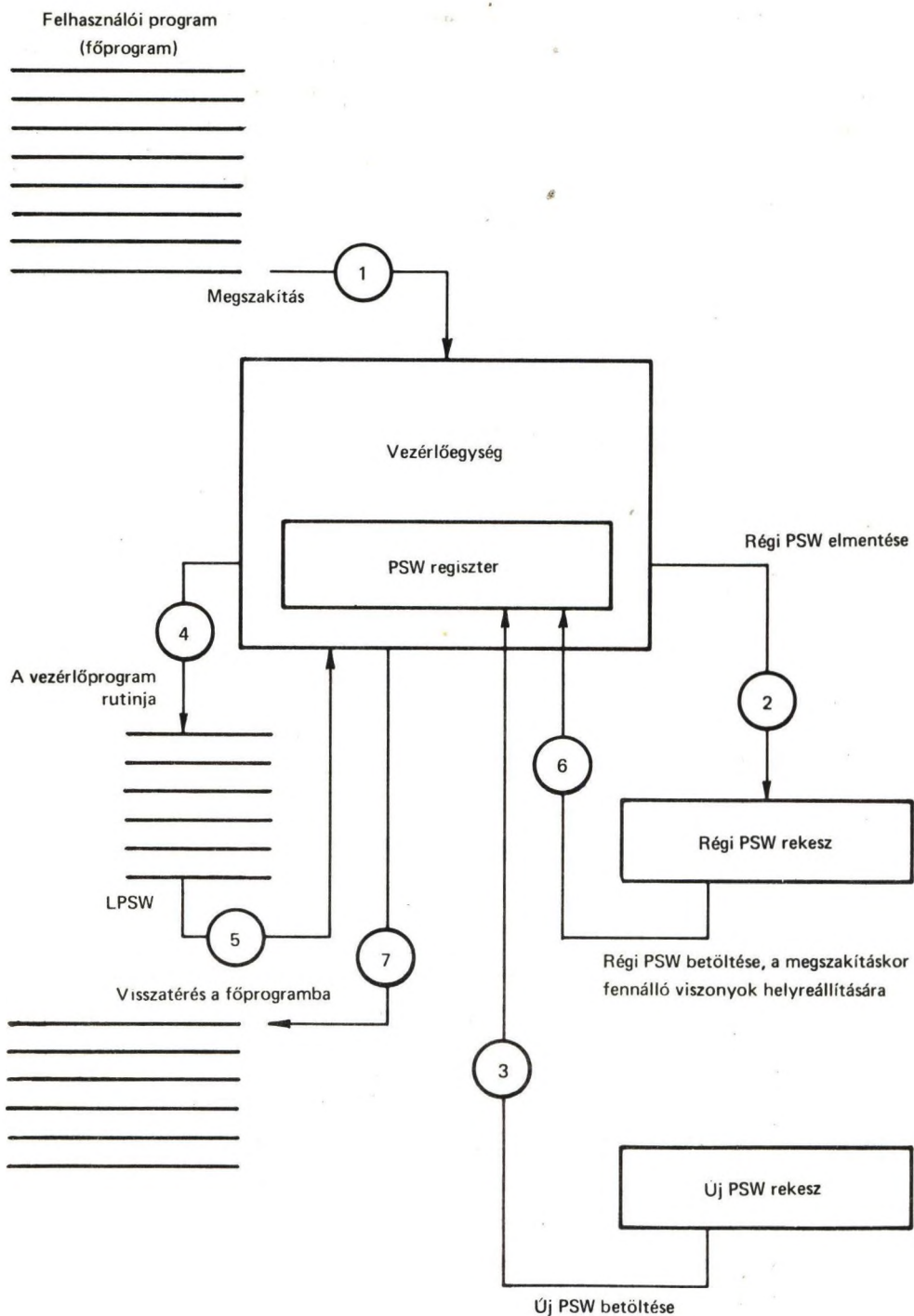
- | | |
|-------|---|
| 0 | Multiplex csatorna megszakítási maszk |
| 1 | 1. szelektor csatorna megszakítási maszk |
| 2 | 2. szelektor csatorna megszakítási maszk |
| 3 | 3. szelektor csatorna megszakítási maszk |
| 4 | 4. szelektor csatorna megszakítási maszk |
| 5 | 5. szelektor csatorna megszakítási maszk |
| 6 | 6. szelektor csatorna megszakítási maszk |
| 7 | Külső megszakítási maszk |
| 8-11 | Az operatív tár védelmi kulcsa |
| 12 | A belső kód kiválasztása. Ha ez a bit 1, akkor a gép belső kódja az ASCII kód, egyébként az EBCDIC kód; |
| 13 | A gépi ellenőrzési rendszer maszkja |
| 14 | A processzor várakozó/működő állapota. Ha ez a bit 1, a processzor várakozási állapotban van; |
| 15 | A rendszer supervisor/program állapota. Ha ez a bit 1, a processzor programállapotban van, ilyenkor egyes - privilegizált - utasítások nem hajthatók végre; |
| 16-31 | Programmegszakítási kód. E bitek tartalmából lehet következtetni arra, milyen eredetű programmegszakítás van érvényben, ha nem fut a program. A megszakítási kódokat a 8. táblázat tartalmazza. |

32-33	Utasításhosszkód. E két bit megadja, hány félszó hosszúsága az éppen végrehajtandó utasítás.
34-35	Feltételkód - egyes aritmetikai és logikai műveletek eredményétől függően változik. A feltételkód értékétől függően programelágaztatást lehet szervezni stb.
36	Fixpontos tulcsordulás maszkja
37	Decimális tulcsordulás maszkja
38	Kitevő alácsordulás maszkja
39	Mantissza nulla maszkja
40-63	A következő utasítás címe.

Megjegyzések a PSW-hez. Maszkolás alatt értjük valamely megszakítás letiltását. Maszkolt a megszakítás, ha a neki megfelelő maszk bit értéke a PSW-ben 0. Tehát ha például a 3. bit értéke 0, ez azt jelenti, hogy a 3. szelektor csatornától érkező programmegszakítás nem juthat érvényre mindaddig, míg ez az adott PSW aktuális. A kulcsmezőben az operatív tár védelmi kulcsának négy bitje található, a védelmi módot a PSW nem tartalmazza.

7.2 A PROGRAMMEGSZAKÍTÁSI RENDSZER

Mint említettük, a processzor PSW regiszterében mindig az éppen futó program PSW-je tartózkodik, ez az aktuális PSW. Ha most ez a program valamilyen ok miatt nem folytatható, mert például hiba keletkezett a rendszer valamely pontján, akkor programmegszakítás jön létre, és az aktuális PSW az operatív tár meghatározott rekeszébe kerül. A programmegszakítás okát is ez a tárolt PSW rögzíti. Az aktuális PSW helyére - a processzor PSW regiszterébe - most annak a programnak a PSW-je kerül, amely program a megszakítás fajtájától függően annak okát elemzi, és rendelkezik a rendszer továbbüzemeltetéséről; megkísérli a hibát kijavítani vagy legalábbis behatárolni stb. Ha a hibavizsgálat eredményeként a program futtatása folytatható, akkor a tárolt korábbi PSW-t kiolvassuk az operatív tár megfelelő rekeszéből,



31. ábra. A programmegszakítás lefolyása

és ismét ez lesz az aktuális PSW. Látható, hogy a következő utasítás címének tárolása igen lényeges: a programba való visszatéréskor a PSW közvetlenül az éppen végrehajtandó utasítás címét tárolja! Mindezt egyértelműen megvilágítja a 31. ábra. A régi PSW rekesz tartalmazza a megszakított programnak a megszakítás pillanatában érvényes aktuális PSW-jét, a megszakítást kezelő program PSW-jét az új PSW rekeszben készítjük elő esetleges betöltésre. Természetesen programmegszakítás nemcsak hiba esetén keletkezhet; az egyes megszakítási okokat a 8. táblázat tartalmazza. A megszakítások öt osztályba sorolhatók.

Felmerül a kérdés, mi történik, ha "egyszerre" több megszakítási feltétel jelentkezik. A gép a megszakítási feltételeket az egyes gépi utasítások végrehajtása között elemzi; az engedélyezett feltételek esetében PSW-t cserél mindaddig, míg a feltételek "elfogynak", a nem engedélyezett feltételeket figyelmen kívül hagyja. A megszakítási feltétel érvényre jutásának engedélyezését bizonyos mértékig a maszkolóással módosíthatjuk. A PSW-cserék sorrendje a megszakítások prioritásának felel meg. Az összes megszakítás négy prioritási szint-hez tartozik, tehát van egy olyan prioritási szint, amelyhez két megszakítási osztály sorolható. A 8. táblázat tartalmazza az egyes megszakítási feltételeket, megszakítási osztály és prioritási szint szerint rangsorolva.

8. táblázat

Megszakítási ok	Megszakítási kód
Input-output, 4. prioritás	
Multiplex csatorna	0000 0000 aaaa aaaa
1. szelektor csatorna	0000 0001 aaaa aaaa
2. szelektor csatorna	0000 0010 aaaa aaaa
3. szelektor csatorna	0000 0011 aaaa aaaa
4. szelektor csatorna	0000 0100 aaaa aaaa
5. szelektor csatorna	0000 0101 aaaa aaaa
6. szelektor csatorna	0000 0110 aaaa aaaa

Megszakítási ok	Megszakítási kód
Programhiba, 2. prioritás	
Érvénytelen műveleti kód	0000 0000 0000 0001
Privilegizált művelet	0000 0000 0000 0010
EX utasítások láncolása	0000 0000 0000 0011
Tárvédelem megsértése	0000 0000 0000 0100
Címzési hiba	0000 0000 0000 0101
Hibás specifikáció	0000 0000 0000 0110
Decimális adathiba	0000 0000 0000 0111
Fixpontos tulcsordulás	0000 0000 0000 1000
Fixpontos osztáshiba	0000 0000 0000 1001
Decimális tulcsordulás	0000 0000 0000 1010
Decimális osztáshiba	0000 0000 0000 1011
Karakterisztika-tulcsordulás	0000 0000 0000 1100
Kitevő-alácsordulás	0000 0000 0000 1101
A mantissza nulla	0000 0000 0000 1110
Lebegőpontos osztáshiba	0000 0000 0000 1111
Supervisor-hívás, 2. prioritás	
SVC utasításbitek	0000 0000 rrrr rrrr
Külső eredetű megszakítások, 3. prioritás	
Timer	0000 0000 1mmm mmmm
"Megszakítás" nyomógomb	0000 0000 m1mm mmmm
2. külső jel	0000 0000 mm1m mmmm
3. külső jel	0000 0000 mmm1 mmmm
4. külső jel	0000 0000 mmmm 1mmm
5. külső jel	0000 0000 mmmm m1mm
6. külső jel	0000 0000 mmmm mm1m
7. külső jel	0000 0000 mmmm mmm1
Gépi hiba miatti megszakítás, 1. prioritás	
Gépi hiba	0000 0000 0000 0000

A kódokban használt betűk jelentése a következő:

- a - az input-output perifériacím bitjei
- r - az SVC utasítás R_1 és R_2 mezőjének bitjei
- m - egyéb külső megszakítási feltételkód.

Mindezek alapján nyilvánvaló, hogy célszerű volt az operatív tárban meghatározott rekeszeket kijelölni az egyes régi és új PSW-k számára, az egyes programok vagy rutinok állapotát leíró információ így nem keveredhet össze, nem veszhet el.

7.3 GÉPI UTASÍTÁSFORMÁTUMOK

Valamennyi nagyobb ESZR modell byte-szervezésű, ún. kétcimes gép. Ez azt jelenti - mint láttuk -, hogy az operatív tár legkisebb címezhető egysége a byte, továbbá egyetlen gépi utasításban két operandussal (adattal) végezhetünk műveletet. A gépi utasítás tartalmazza a műveleti kódot és a két operandus címét, esetleg magát az egyik operandust. A műveleti kód egyértelműen meghatározza az elvégzendő műveletet (például decimális osztás, aritmetikai léptetés balra stb.), az operanduscímek pedig rendre meghatározzák az 1. és 2. operandus helyét az operatív tárban.

Ahhoz, hogy egy gépi utasítás végrehajtható legyen, a végrehajtást megelőzően az operanduscímeknek és az operandusoknak konkrét értékkel kell bírniuk.

Mindezek alapján a gépi utasítások egységes formátuma az alábbi elvi ábrán látható:

Műveleti kód	1. operandus címe	2. operandus címe
--------------	-------------------	-------------------

A gépi utasításoknak öt fő csoportját különböztetjük meg:

- RR típusu utasítás, regiszter - regiszterműveletekhez
- RX típusu utasítás, regiszter - indexelt tárcim műveletekhez
- RS típusu utasítás, regiszter - operatív tár közötti műveletekhez
- SI típusu utasítás, operatív tár - közvetlen operandus közötti műveletekhez
- SS típusu utasítás, operatív tár - operatív tár közötti műveletekhez.

Az utasításokat annak alapján jelöltük, hogy mely memóriaterületen található az egyes megcímzett operandusok. Például az RR típusu utasításban mindkét operandus a regiszterekben van, az RS típusuban az 1. operandus a regiszterben, a 2. operandus pedig az operatív tárban.

A műveleti kód első két bitje és az utasítás hossza, illetve típusa között meghatározott kapcsolat áll fenn. Ezt a 9. táblázat tartalmazza.

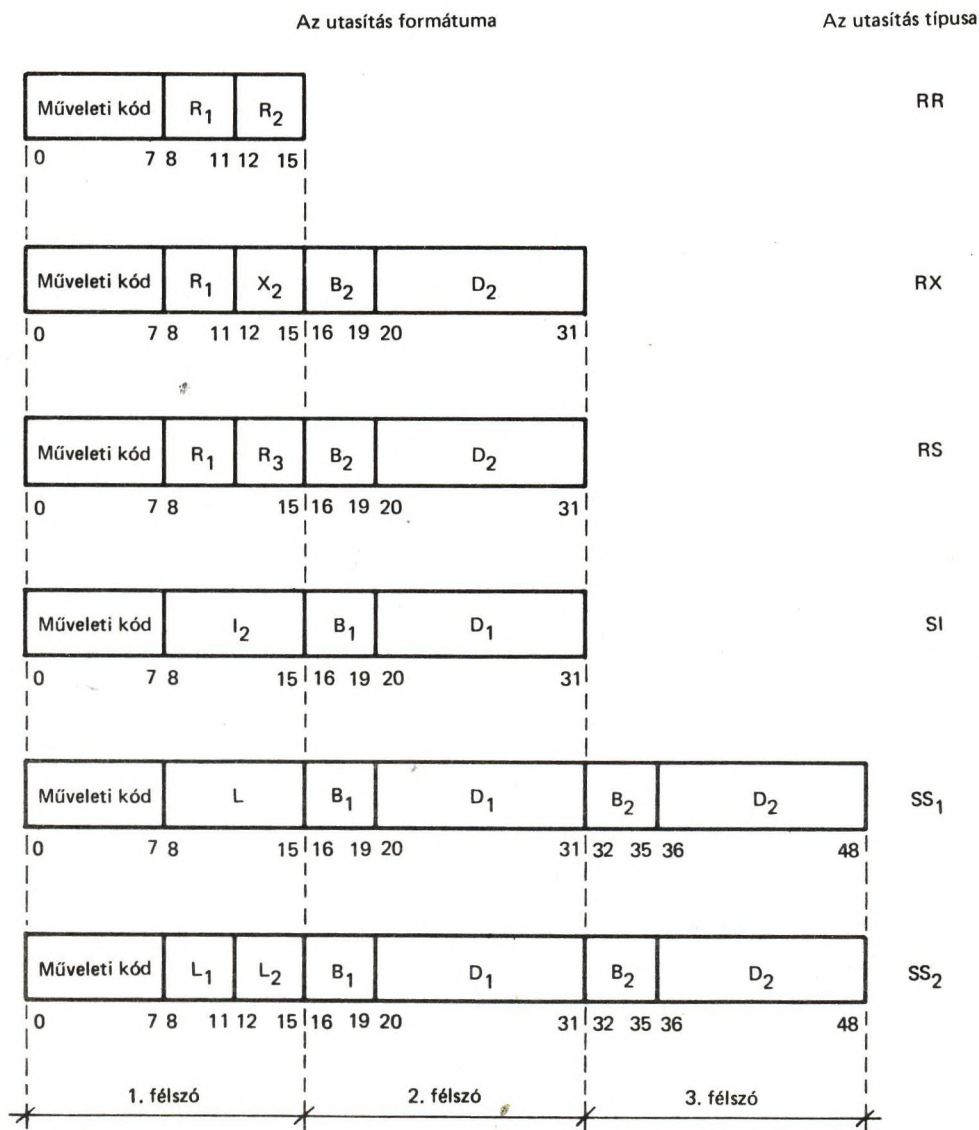
9. táblázat

Az utasítás hossza félszóban	A PSW 32 - 33. bitje	Az utasítás 0-1. bitje	Az előforduló utasítás - típusok
0	00	--	--
1	01	00	RR
2	10	01	RX
2	10	10	RS vagy SI
3	11	11	SS

Az utasításhossz kódolásának igen nagy jelentősége van az utasításcím szempontjából. A végrehajtandó utasítás kiolvasása után ugyanis azonnal be kell állítani a következő utasítás címét a PSW-ben, hogy az esetleg előforduló programmegszakítás után biztosított legyen a visszatérés a programba. A processzor tehát az utasítás első két bitjéből - tulajdonképpen a műveleti kód első két bitjéből - tudja meg, mennyivel növelje a PSW-regiszterben levő ak-

tuális utasításcimét. Például ha a kiolvasott utasítás SS típusu, hossza 3 fél-
szó = 6 byte, és az aktuális utasításcím $00009E_{(16)}$ volt, akkor a következő
utasítás címe $0000A4_{(16)}$ lesz stb.

Az egyes utasításformátumokat és az utasításmezők elnevezését a 32.
ábrán tüntettük fel.



32. ábra. Gépi utasítások formátumai

Az egyes mezők értelmezése a következő:

- Az R_1 és R_2 egy-egy általános célú vagy lebegőpontos regiszter sorszámát tartalmazza, amelyben az első, illetve második operandus található;
- Az R_3 szintén egy regisztert jelöl ki. Van néhány háromcimes utasítás is, ott R_3 a harmadik operandus címét tárolja;
- Az X_2 a második operandus indexregiszterének sorszámát tartalmazza, sohasem mutat lebegőpontos regiszterre;
- A B_1 és B_2 az első, illetve második operandus bázisregiszterének sorszámát tárolja. A bázisregiszter mindig általános célú regiszter;
- A D_1 és D_2 az első, illetve második operandus relatív címe;
- Az I_2 magát az egy byte hosszúságú operandust tárolja;
- Az L az SS_1 típusú utasítás azonos hosszúságú operandusainak byte-okban kifejezett hosszára utal;
- Az L_1 és L_2 az SS_2 típusú utasítás első, illetve második operandusának byte-okban kifejezett hosszára utal.

A fentiek megvilágítására tekintsük át, hogyan végzi el a gép az operandusok címének kiszámítását. Az első (illetve harmadik) operandus címe az utasításból közvetlenül látható: ha valamely R_1 mező tartalma például 0101, akkor az első operandust az 5-ös számú általános célú regiszterben kell keresni. Ha egy regiszter pusztán sorszáma alapján általános célú és lebegőpontos regiszterként is értelmezhető (például a 4-es), akkor a műveleti kód dönti el, hogy lebegőpontos műveletről van-e szó vagy sem. Ennek alapján már azonosítható a regiszter. Ha az első operandus nem regiszterben, hanem az operatív tárban van, címét az alább ismertetendő módon számítjuk. A második operandus vagy valamely regiszterben, vagy az operatív tárban található. Az operatív tárbeli cím számításának menete:

RX típusú utasításnál	2. operandus címe = $(X_2) + (B_2) + D_2$
Egyébként	2. operandus címe = $(B_2) + D_2$
	1. operandus címe = $(B_1) + D_1$

A zárójelek itt azt jelentik, hogy a bennük specifikált regiszter tartalma vesz részt a cím számításában, nem pedig maga a regiszter sorszáma. Például adott az alábbi utasítás:

0111 1100 0100 1000 1100 0000 0001 0100

Állapítsuk meg az utasítás típusát, műveleti kódját, valamint az egyes operanduscímeket! Feltesszük, hogy a 8-as regiszter tartalma = 4000, a 12-es regiszterben pedig 820 van.

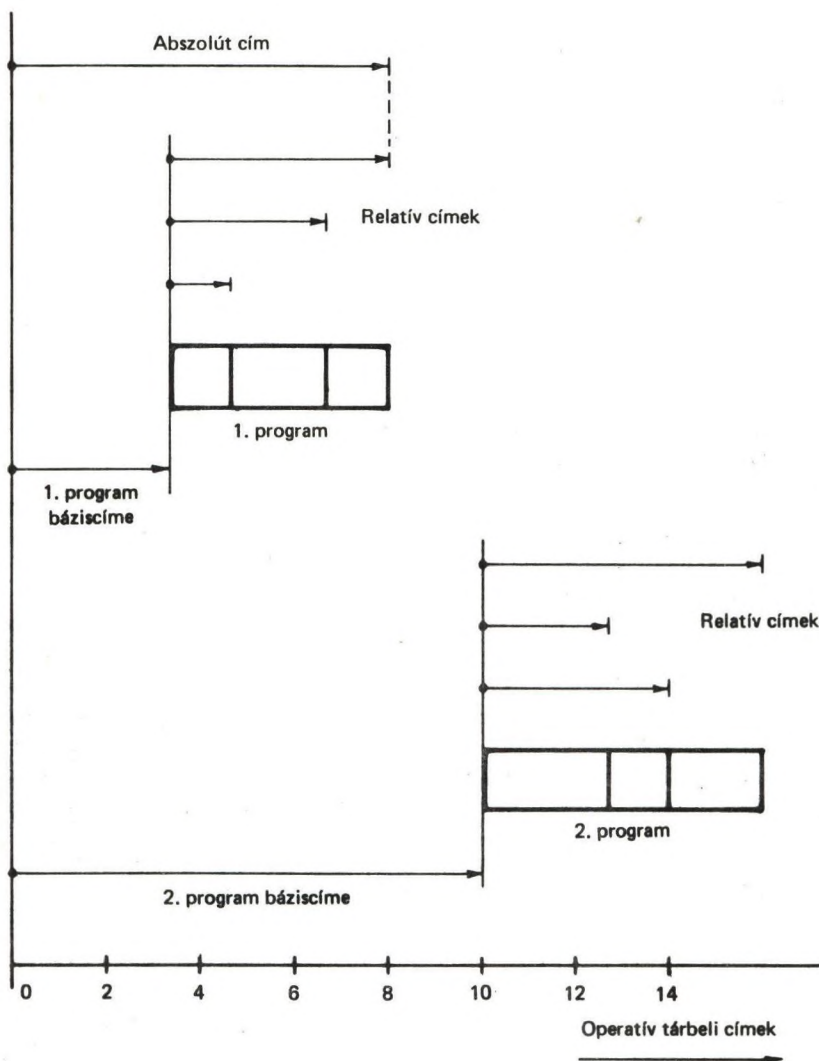
Megoldás: Az utasítás első két bitjéből látható, hogy RX típusu utasításról van szó. A műveleti kód táblázatból kiderül, hogy lebegőpontos szorzásról van szó. Az első operandus a 4-es sorszámu lebegőpontos regiszterben van. A második operandus címét úgy számítjuk, hogy összeadjuk a 8-as és a 12-es regiszter tartalmát, és ehhez még hozzáadjuk az utasításban szereplő D_2 relatív címet: $(X_2) + (B_2) + D_2 = 4000 + 820 + 20 = 4840$. A második operandust tehát az operatív tár 4840-es címén kell keresni.

Megjegyzések: Indexregiszter használata azért célszerű, mert több operandus egymást követő megcímzésénél nem kell minden esetben módosítani magát az utasítást. Elegendő csupán az indexregiszter tartalmát változtatni, hogy előállítsuk az új címet.

A bázisregiszter használata lehetővé teszi a program áthelyezését az operatív tárban anélkül, hogy át kellene írunk az egyes utasításokat. Ha a bázisregiszterbe egy más értéket helyezünk, mint például a program egy korábbi futásakor tettük, akkor valamennyi cím, amelynek kiszámításában ez a bázisregiszter részt vesz, új értéket kap, és a cím tartalma (például egy operandus) más helyre kerül a tárban. Ily módon megvalósítható az ún. dinamikus programbetöltés, vagyis a program az operatív tár bármely területén futtatható, feltéve, hogy elegendő szabad tárterület áll rendelkezésre.

A relatív cím szintén lehetőséget nyújt a címmódosításra, de mivel értéke konstans - nem pedig egy regiszter változó tartalma - az operanduscím megváltoztatásához át kell írni magát az utasítást. A 12 bites relatív címmel egy összefüggő, $2^{12}=4096$ byte hosszúságu operatív tárterület fedhető le.

Az operatív tárbeli cím, a báziscím és a relatív cím kapcsolatát a 33. ábra szemlélteti. L , L_1 , L_2 mezők tartalma azt fejezi ki, hogy az operandusok a megcímzett (bal oldali) byte-on kívül még hány byte-ból állnak. Például $L_1 = 1001$ egy tíz byte-os operandushosszt jelöl stb.



33. ábra. Operatív tárbeli címek szerkezete

7.3.1 OPERANDUSFORMÁTUMOK

Az ESZR számítógépek alapvetően négyféle operandussal tudnak dolgozni.

Ezek:

1. Decimális szám,
2. Fixpontos (bináris) szám,
3. Lebegőpontos (bináris) szám,
4. Logikai mező.

1. Decimális szám

A decimális számok kétféle alakban ábrázolhatók: zónázott és tömör formájú decimális számként. A zónázott decimális szám a következő alaku:

Zóna	Számjegy	Zóna	Számjegy	Zóna	Előjel	Számjegy
Byte		Byte			Byte	

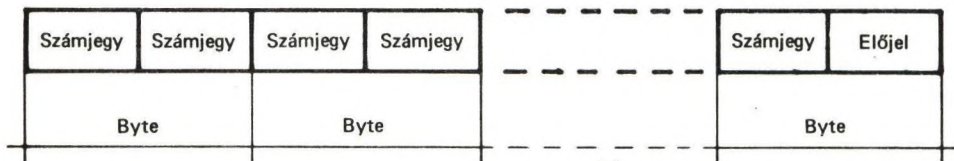
Minden egyes byte-ot két négy bites mezőre (félbyte-ra) osztunk. A bal oldali tetrád a zónarészt, a jobb oldali tetrád a decimális számjegyrészt tárolja binárisan kódolt formában. Az előforduló számjegyek az alábbiak:

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tízről tizenötig a négy biten ábrázolható számok nem tartoznak a decimális számjegyekhez, ebből a szempontból tiltott tartományt alkotnak. A decimális korrekció lényege éppen az, hogy a tíz és tizenöt közé eső számokat átalakít-

sa nulla és hat közé eső érvényes decimális számjegyekké. Zónázott decimális számok főleg input-output műveletekben lépnek fel, ugyanis az adatok beolvasása, illetve kinyomtatása során ez a formátum használatos.

A tömör formájú decimális szám alakja:



A tömör formájú decimális szám minden egyes tetrádján számjegy található, kivéve az utolsó tetrádot, amely az előjelet tárolja. Leegyszerűsítve a dolgot, azt mondhatjuk: egy zónázott decimális számból úgy nyerünk vele azonos értékű tömör formájú decimális számot, hogy a zónázott szám utolsó byte-jában a tetrádokat felcseréljük egymással, majd a zónarészeket kiemeljük a számból, és a megmaradt számjegyeket jobbra "tömörítjük" (innen az elnevezés: tömör formájú szám). E számok hossza maximálisan 16 byte lehet, ami 31 decimális számjegyet jelent. Az ábrázolható legnagyobb abszolút értékű szám $10^{31}-1$. Valamennyi decimális szám egész, a tizedespont figyelembevételéről a programozónak kell gondoskodnia.

Tömör formájú decimális számok főleg decimális aritmetikai műveletekben fordulnak elő. Nyilvánvaló, hogy a zónarészekre nincs szükség műveletvégzéskor, azok csak feleslegesen foglalják a helyet a tárban. Ezért a beolvasott zónázott decimális számokat a decimális aritmetikai műveletek céljára előbb át kell alakítani tömör formájú számokká, és ezeket (például az eredményt) a nyomtatáshoz vissza kell alakítani zónázott formába. Az átalakítások alkalmas gépi utasításokkal végezhetők el.

Zóna- és előjel-konvenció

Attól függően, hogy a gép milyen belső kódban ábrázolja a számot, más és más lesz a zóna, illetve az előjel kódja. Az ábrázolás tulnyomórészt EBCDIC kódban történik. A 10. táblázat feltünteti az egyes kódokat. Megjegyezzük,

hogy a gép negatív előjelként értelmezi a hexadecimális B-t, és pozitív előjelnek tekinti a hexadecimális A, E, F-et is a táblázatban feltüntetett kódokon kívül.

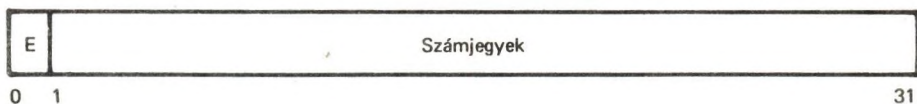
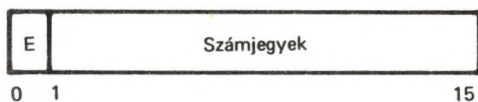
10. táblázat

	EBCDIC	ASCII
Decimális számjegy zónarésze	1111	0101
+ előjel	1100	1010
- előjel	1101	1011

Míndezek alapján a 0000 0110 1000 1101 két byte hosszúságu tömör formájú decimális szám -68-cal ekvivalens stb.

2. Fixpontos szám

Fixpontos formátumban bináris egész számok fordulnak elő félszó- vagy szóhosszuságban. Az egyes számformátumokat a következő rajz szemlélteti:

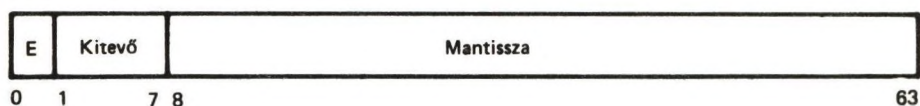
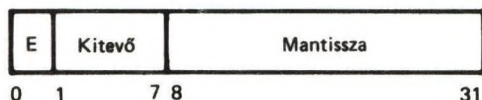


Valamennyi fixpontos szám 0-dik bitje a szám előjelét tartalmazza. 1 jelenti a mínusz, 0 a plusz előjelet. A számok numerikus része bináris számjegyekből áll. A különböző formátumok különböző nagyságu maximális abszolút érték ábrázolását teszik lehetővé. A negatív számokat kettes komplementssükkel ábrázoljuk, ez az előjel kezelése szempontjából igen kényelmes. Itt utalunk arra a körülményre is, hogy a félszóban is elhelyezhető bináris számokat nem célszerű gépi szóval ábrázolni memóriatakarékossági megfontolásokból. A fixpontos számok és számtartományok egyes jellemzőit a 11. táblázat tartalmazza.

	Byte-ok száma	Bitek száma	Legnagyobb pozitív szám	Legnagyobb abszolút értékű negatív szám
Félszó	2	16	$+2^{15}-1$	-2^{15}
Szó	4	32	$+2^{31}-1$	-2^{31}

3. Lebegőpontos szám

A lebegőpontos számok formátumait az alábbi rajz mutatja.



A kétféle lebegőpontos számot szokás rövid (32 bites) és hosszú (64 bites) lebegőpontos számnak is nevezni. Mindkét formátumnál a 0-dik bit a mantissza előjele.

A lebegőpontos számot a következőképpen értelmezzük:

$$N = \begin{matrix} + \\ - \end{matrix} m \cdot 16^k$$

ahol N a lebegőpontos számot,

m a mantisszát,

k a kitevőt jelenti.

A mantissza a rövid lebegőpontos alaknál 6, a hosszúnál 14 hexadecimális jegyet tartalmazó törtszám. A legkedvezőbb számábrázolás akkor érhető el, ha a mantisszában minél több számjegyet figyelembe lehet venni a pontosság nö-

velése érdekében. Célunk tehát az, hogy a mantissza elején a lehető legkevesebb zérus legyen. Azt az alakot, amelyben a mantissza első hexadecimális számjegye zérustól különböző, a szám normalizált alakjának nevezzük.

A normalizált alak hexadecimális számjegyekre vonatkozik, tehát a

$$0.0001....._2$$

alak még normalizáltnak számít, hiszen ez nem más, mint $0.1....._{16}$.

A lebegőpontos műveletek nagy része mind normalizált alaku, mind normalizálatlan operandusokkal elvégezhető. Az eredmény hasonlóképpen lehet normalizált vagy - összeadásnál és kivonásnál - nem normalizált. A lebegőpontos műveletek közül az összeadás és a kivonás csak egyenlő kitevőjű operandusokkal végezhető el. Szorzásnál, illetve osztásnál a kitevőket összeadjuk, illetve kivonjuk egymásból, a mantisszákat pedig összeszorozzuk, illetve elosztjuk egymással:

$$\begin{aligned} N_1 &= m_1 \cdot 16^{k_1} & N_1 \cdot N_2 &= m_1 \cdot m_2 \cdot 16^{(k_1+k_2)} \\ N_2 &= m_2 \cdot 16^{k_2} & \frac{N_1}{N_2} &= \frac{m_1}{m_2} \cdot 16^{(k_1-k_2)} \end{aligned}$$

A kitevő hétbites bináris szám, értéke 0-tól 127-ig terjedhet (2^7-1). Mivel a kitevőnek nincs előjele, valamilyen módon meg kellett oldani a negatív kitevőjű számok ábrázolását is. Ezért a kitevőt a rendszer automatikusan úgy értelmezi, mintha 64-gyel több lenne tényleges értékénél, s minden esetben levon belőle 64-et. Ez az ún. 64-es eltolt kitevőábrázolás lényege. A 0 értékű kitevő ábrázolása tehát:

$$1000000.$$

Ennek értelmében minden 64-nél kisebb kitevőt negatívnak tekintünk, a 64-nél nagyobb kitevők pozitívak lesznek. Az így ábrázolható számtartomány 16-nak -64-edik hatványától +63-adik hatványáig terjedhet.

A lebegőpontos operandusok néhány jellemzőjét a 12. táblázatban foglaltuk össze.

	Byte-ok száma	Bitek száma	Legnagyobb pozitív szám	Legnagyobb abszolút értékű negatív szám
Rövid lebegőpontos szám	4	32	$(1-16^{-6}) \cdot 16^{63}$	16^{-65}
Hosszu lebegőpontos szám	8	64	$(1-16^{-14}) \cdot 16^{63}$	16^{-65}

Lebegőpontos operandusok használata ott célszerű, ahol igen nagy számtartományt kell átfogni, illetve ha nem becsülhető meg előre, hogy milyen nagyságrend körüli eredmények várhatók.

4. Logikai mező

Logikai operandus lehet szó, duplaszó, változó hosszúságú mező vagy egyetlen byte. Minden esetben megengedett valamennyi lehetséges bitkombináció, és a műveletben részt vevő operandusok egyenlő hosszúságúak. Logikai operandusok elhelyezhetők az operatív tár általános célú regisztereiben, magában a főmemóriában és közvetlenül a gépi utasításban is (SI típusú utasítás, kizárólag egy byte hosszúságú logikai operandust tárolhat).

A logikai mezők többnyire karakterek kódolt formáiból állnak. Tetszőleges alfanumerikus karaktersorozat felfogható logikai operandusnak. Egy-egy ilyen karaktersorozat hossza maximálisan 256 byte lehet. A 34. ábra egy általános logikai operandust mutat be, amely karakterekből áll.

Karakter	Karakter	Karakter	-----	Karakter
Byte	Byte	Byte	-----	Byte

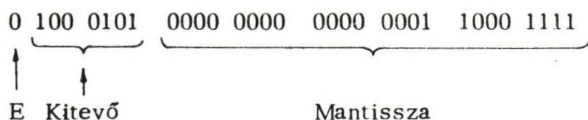
34. ábra. Karaktersorozat mint logikai operandus

Végül szeretnénk rámutatni egy igen fontos körülményre, amely a bináris ábrázolásmódból következik. Nevezetesen arról van szó, hogy a számítógép maga nem tudja megkülönböztetni egymástól az egyes adattípusokat, hiszen valamennyien egységesen bináris formátumúak.

Tekintsük példaként az alábbi négybyte-os, egész szóhatáron elhelyezkedő információt:

4500018F₍₁₆₎

Ha most erre valamilyen lebegőpontos műveletben hivatkozunk, a gép a szó tartalmát rövid lebegőpontos számnak fogja tekinteni, és az alábbi módon értelmezi:



Látható, hogy egy normalizálatlan rövid lebegőpontos számról van szó. Ha ennek értékét keressük, először célszerű a számot normalizálni, majd a már megismert konverziós eljárás segítségével decimális számmá konvertálni:

$$\begin{aligned}
 0.00018F_{(16)} \cdot 16^{+5} &= 0.18F_{(16)} \cdot 16^{+2} = \\
 \left(\frac{1}{16^1} + \frac{8}{16^2} + \frac{15}{16^3} \right) \cdot 16^{+2} &= \\
 = \frac{256 + 128 + 15}{16^3} \cdot 16^{+2} &= \frac{399}{16^3} \cdot 16^{+2} = \frac{399}{16} = 25_{(10)}
 \end{aligned}$$

Semmi akadálya sincs azonban annak, hogy a fenti négybyte-os információra valamilyen fixpontos műveletben hivatkozzunk, s ekkor a gép az érintett szó tartalmát fixpontos számnak fogja tekinteni, amelynek előjele pozitív (0-val kezdődik a szám). Adott esetben még az is megengedett, hogy a szóra deci-

mális műveletben hivatkozzunk, hiszen nem tartalmaz érvénytelen számjegyet, az F-et pedig a gép elfogadja pozitív előjelnek. Decimális számként értelmezve tehát a szó tartalma:

+ 4 500 018

Végül felfoghatjuk a szót valamilyen logikai operandusként is, amely ez esetben nem karaktorsorozat, hanem tetszőleges felhasználói információ (például maszkok sorozata stb.).

Mindezzel arra szerettünk volna rávilágítani, hogy a számítógép szempontjából teljesen közömbös valamely tárterület tartalma, és a programozónak kell tudnia, milyen tárcímekre milyen jellegű információt helyezett el. Természetesen a feldolgozás során már nem mindegy, milyenfajta műveletben hivatkozzunk az illető tárterületre, hiszen például érvénytelen decimális számjegy vagy előjelkód észlelése esetén a gép nem fogadja el az operandust.

7.4 A PROCESSZOR FELÉPÍTÉSE

Valamennyi ESZR számítógép közös jellemzője, hogy un. univerzális számítógép. Ez azt jelenti, hogy rendelkezik mindazokkal a hardware-erőforrásokkal, amelyek lehetővé teszik a decimális, fixpontos (bináris), lebegőpontos műveletek végzését, valamint logikai operandusok kezelését. A központi egység amellet, hogy kiadja a teljes számítógéprendszer számára szükséges vezérlőjeleket, és összehangolja a teljes hardware működését, elvégzi a különböző típusu operandusokon értelmezett aritmetikai-logikai műveleteket is. A standard gépi utasítások végrehajtása során a központi egység az alábbi fő feladatokat végzi el:

- kiolvassa (lehívja) az aktuális utasítást az operatív tárból,
- értelmezi (dekódolja) az utasítás műveleti kódját, vagyis megállapítja az elvégzendő műveletet,
- beállítja az utasításcím (PSW 40-63. bitje) értékét a soron következő utasítás címére, ezzel biztosítja a program folytatását egy esetleges megszakítás után,

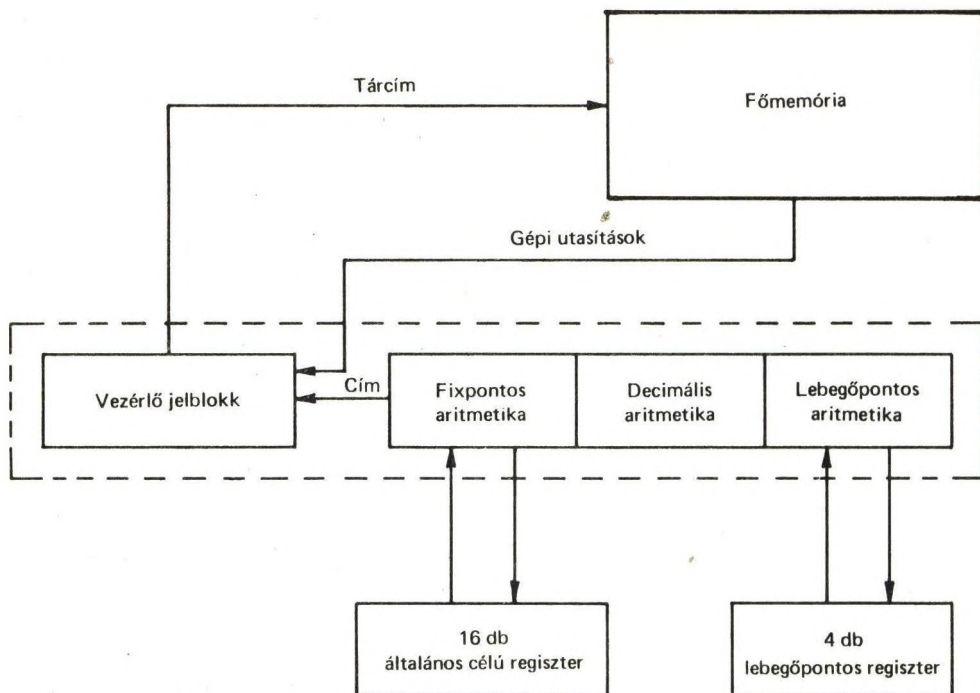
- megkeresi (megcimzi) és kiolvassa a hivatkozott operandusokat az operatív tárból,
- ténylegesen végrehajtja az előírt műveletet,
- az eredményt egy meghatározott címre teszi (ez az esetek döntő többségében az 1. operandus címe).

A felsorolásban nem vettük figyelembe a hibavizsgálatokat, és feltételezzük, hogy az utasítás végrehajtása normálisan megy végbe.

Látható, hogy egyetlen gépi utasítás is több - esetleg igen sok - elemi lépésre bontható, amelyek sorrendje szigorúan kötött. A gépi utasítás így módon felfogható úgy, mintha valamilyen igen alacsony szintű programot (az elemi lépések sorozatát) hajtánánk végre egy-egy gépi utasítás lefuttatása-kor. Ez a program természetesen más és más attól függően, hogy milyen elemi lépésekből épül fel az adott gépi utasítás.

A fent ismertetett alacsony szintű program az ún. mikroprogram, a mikroprogramot alkotó elemi lépések pedig az egyes mikroutasítások. A mikroprogram a számítógép legalacsonyabb programozási szintje, és a felhasználó által általában nem módosítható. A mikroprogram egy erre a célra szolgáló mikroprogramtárban helyezkedik el, amelyből az egyes mikroutasítások a processzor által kiolvashatók. Mikroutasítások végzik például az aritmetikai-logikai egység vezérlését, az operatív tárral folytatott adatforgalom irányítását, a hardware-megszakítások elemzését stb. Jellemét tekintve a mikroprogramtár hasonló az operatív tárhoz, megvan a maga címregisztere és adatregisztere. Alapvető különbség azonban a kettő között, hogy a mikroprogramtárból csak kiolvasni lehet információt, beírni általában nem. Emiatt szokás a mikroprogramtárat állandó memóriának is nevezni (Read-Only Memory, ROM).

Maguk a mikroutasítások is hasonlóak a gépi utasításokhoz annyiban, hogy kötött funkciójú mezőkre vannak osztva, és az egyes mezők tartalmának dekódolása útján jut a processzor a szükséges vezérlőjelekhez, impulzusokhoz. A 35. ábrán feltüntettük a processzor egyszerűsített vázlatát és a hozzá kapcsolódó operatív tárterületeket. Az ábrán szaggatott vonallal határolt terület a processzort foglalja magában.



35. ábra. A központi egység blokkvázlata

Megjegyzés. A többféle aritmetika használatát a különféle jellegű műveletek indokolják. A decimális aritmetika elsősorban adatfeldolgozásra alkalmas, ugyanis elkerüljük a hosszadalmas decimális-bináris konverziót. A lebegőpontos aritmetika tudományos-műszaki számításoknál előnyös, ahol az átfogni kívánt nagyságrendek tartománya igen nagy. Végül a fixpontos aritmetika általános célú, és jóval gyorsabb a decimálisnál. Ott előnyös a használata, ahol nagy műveleti sebességre van szükség, és viszonylag kevés decimális-bináris konverziót kell végezni.

8. A folyamatábra-készítés alapjai

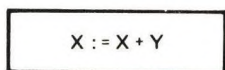
8.1 A FOLYAMATÁBRA MINT ESZKÖZ

Mint láttuk, a felhasználó szempontjából a számítógép egy olyan, nagy bonyolultsági fokú automata, amely képes a számára előre megadott logikai lépéssorozat, műveletssorozat elvégzésére. A gép azonban magától nem lenne képes például egy félbehagyott vagy valamilyen okból megszakított műveletssor értelmes folytatására, mert gondolkodni nem tud. Ezért a feladatot a gép számára mindig a felhasználónak kell pontosan leírnia, megfogalmaznia olyan ki-fejezésformában, amely a gép számára érthető. Valamely feladat megoldására a fentiek alapján megírt utasítássorozatot programnak nevezzük. Látható, hogy a program nem tartozik szervesen a számítógéphez; a gépen egymás után, vagy akár egyszerre több program is futtatható anélkül, hogy a számítógép műszaki tulajdonságai, adottságai közben megváltoznának. A számítógépi programokat alapvetően két fő csoportra oszthatjuk:



A rendszerprogramokat nagyrészt a gyártó cég dolgozza ki, bár a felhasználó is bővítheti a rendszert saját maga készítette programokkal. Ezek részben a számítógép üzemeltetését segítik elő, részben különböző szolgáltatásokat nyújtanak a felhasználói programoknak.

A felhasználói programok csakis a felhasználótól származnak, és valamilyen feladat megoldására szolgálnak (adatfeldolgozás, tudományos-műszaki számítások stb.). Ezek a programok ugyanakkor - mint a számítógéprendszer erőforrásait - használhatják a rendszerprogramokat.



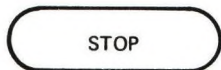
Az elvégzendő művelet jele



Input-output művelet, független az információáramlás irányától



A folyamatábra kezdetét jelzi



A folyamatábra végét jelzi



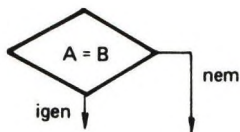
Az információ terjedésének irányai, elágazások jele



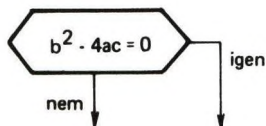
Csatlakozás ugyanazon a lapon



Csatlakozás másik lapon



Döntés, illetve feltételvizsgálat; a feladatmegoldás folytatása a bejelölt feltétel teljesülésétől függ



36. ábra. Folyamatábra-szimbólumok

A programok írását, javítását nagymértékben megkönnyíti a folyamatábrák alkalmazása. A következőkben vázlatosan áttekintjük a folyamatábra-készítés alapjait, sajátosságait. A folyamatábra (más szóval blokkdiagram) valamely feladat megoldásának menetét tükrözi, annak logikai lépéseit írja le grafikus formában. A folyamatábrának van egy kezdőpontja és egy végpontja, a kettő között a haladási irányt az egyes lépések ábráit összekötő nyilak jelzik. Más haladási irány nem képzelhető el; a folyamatábra lépései egymás után logikai sorrendben következnek, tehát az okozat nem előzheti meg magát az okot. Elágazásoknál több folyamatábraág található, fontos tehát tudnunk a továbbhaladás irányát. Ezért a nyilakat mindig pontosan kell feltüntetni. A folyamatábra tehát magát a feladatmegoldás algoritmusát, műveleti sorrendjét és lépéseit adja meg.

A 36. ábrán feltüntettük a leggyakrabban előforduló szabványos folyamatábra-jelöléseket, és megadtuk értelmezésüket. Használatuk nem kötelező, de nemzetközileg is igen elterjedt.

Megjegyzés: Csatlakozási szimbólumokra azért van szükség, mert az esetek többségében a folyamatábra nem fér el egy lapon, s így az ábra bármely pontjáról lehet hivatkozni tetszőleges, másik lapon levő pontra. A "csatlakozás ugyanazon a lapon" szimbólumot akkor célszerű használni, ha az információterjedés irányát jelölő vonalak berajzolása tulságosan kusza, áttekinthetetlen ábrát eredményezne. Mindkét esetben valamilyen alfanumerikus jelet írunk a csatlakozási szimbólumba; ezzel azonosítható a jelölt hely az ábrában.

8.2 A FOLYAMATÁBRÁK FAJTÁI

Alkalmazásuk szerint kétféle folyamatábrát különböztetünk meg:

- makroblokkdiagram,
- mikroblokkdiagram.

A makroblokkdiagram a feladatot főbb vonalakban vázolja, de lépésenként nem elemzi. Csak a fontosabb részfeladatokat nevezi meg, a megoldás módját azonban nem közli. Ott célszerű használni, ahol áttekintést kell adnunk valamely program főbb lépéseiről, állapotairól, s a teljesen részletes folyamatábra zsufolt és bonyolult ábrázolási módja nehezítené az eligazodást. A makroblokk-

diagram éppen ezért program- (feladat-) orientált; nem tünteti fel a gép által végrehajtandó elemi műveleteket, hanem azokat logikai egységbe vonja össze.

A mikroblokkdiagram a feladatot elemi lépésekre bontva ábrázolja, pontosan tükrözi a feladatmegoldás módját úgy, ahogyan azt a gép végzi. Ennélfogva a mikroblokkdiagram gépre orientált; ennek alapján végezhető el a program kódolása is.

8.2.1 A FELADAT FELBONTÁSA

Foglaljuk össze, mit kell tennie a felhasználónak egy feladat kézhezvételétől a megoldás eléréséig, feltéve, hogy a feladatmegoldás számítógéppel történik!

1. A feladat megfogalmazása – pontosan körül kell határolni, mi az elvégzendő feladat;
2. A feladat lépésekre bontása úgy, hogy e lépésekből folyamatábra készülhessen (szükség szerint makro- és mikroblokkdiagram);
3. Folyamatábra készítése;
4. Kódolás - programozás, a folyamatábra lépéseinek átírása valamely programnyelvre;
5. Próba-futtatás, hibajavítás - a kódolt programot mindig célszerű ellenőriztetni próbaadatokkal.

Lássunk néhány példát egyszerűbb feladatok megoldására!

1. példa

Egy kereskedő ötféle termékből összesen tíz tételt adott el egy nap alatt. Az egyes tételek eladása után más-más jutalék illeti meg aszerint, hogy az eladottétel melyik termékcsoportba tartozik. Kérdés: Mekkora a kereskedő haszna egy nap alatt, ha az egyes tételek utáni hasznát a 13. táblázat szerint kell számítani?

13. táblázat

Termékcsoport	Jutalék (Ft)
1	$E \cdot 0,15$
2	$(E - \ddot{O}) \cdot 0,40$
3	$\ddot{O} \cdot 0,10$
4	$10 + \ddot{O} \cdot 0,05$
5	15

E = eladási ár, $\ddot{O}$ = önköltség.

Megoldás: A tíz tétel mindegyikét meg kell vizsgálni, melyik termékcsoportba esik. A jutalékot egy erre a célra kijelölt helyen kell tárolni úgy, hogy minden egyes újabb termék utáni járulék az előző értékhez hozzáadható legyen. Ezt egy ugynevezett munkarekeszben valósíthatjuk meg, amelynek információ-tartalma lépésről lépésre változhat, így mintegy gyűjti a számítás közbeni eredményeit.

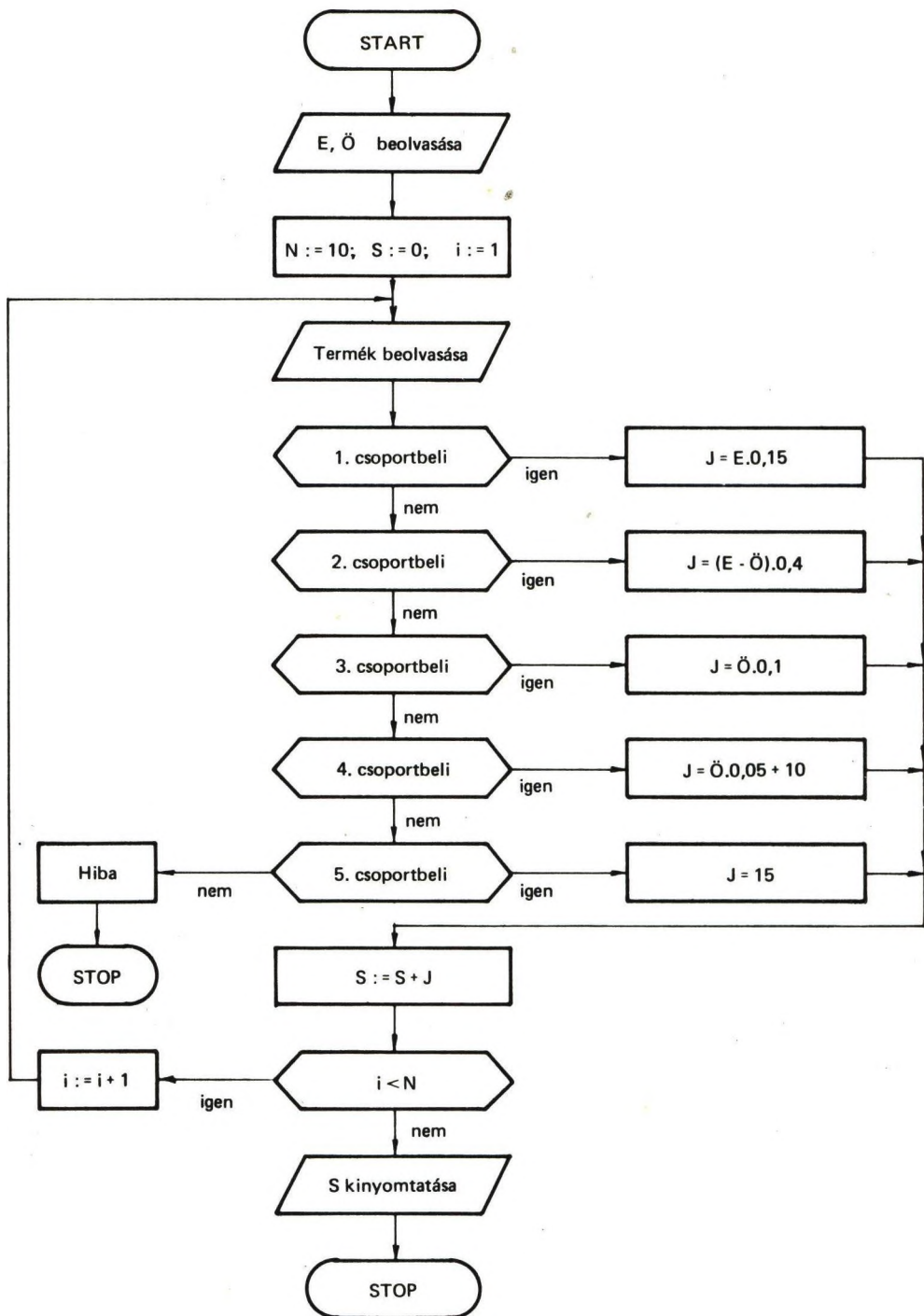
A folyamatábra három funkcionális részre osztható:

- a kiindulási adatok beolvasása (eladási ár, önköltség az egyes termékekre vonatkozólag);
- tényleges jutalékszámítás, a tíz eladott termék utáni jutalék kiszámítása és összegezése;
- az eredmény (a teljes jutalék egy napra) kinyomtatása például sornymotatón.

A folyamatábra a 37. ábrán látható. Figyelembe vettük az esetleges hibás lyukasztás lehetőségét is, erről hibajelzést adunk.

A folyamatábra a következő logikai műveletsorozatot írja le:

- A kiindulási adatok beolvasása, a vizsgálandó terméktételek számának beállítása (N), a teljes haszon tárolására kijelölt rekesz (S) tartalmának zérusra állítása, az aktuális tételsorszám (i) kezdeti értékének beállítása. Mivel összesen tíz tételünk van, a tételvizsgálatot és a jutalékszámítást is pontosan tízszer végezzük el, minden egyes tételre vonatko-



37. ábra. Az 1. példa megoldása

zóna egyszer. Ezt fejezi ki az $N := 10$ értékadás. A tízszer egymás után elvégzett tételvizsgálat és jutalékszámítás egymáshoz teljesen hasonlóan megy végbe, ezért ezt a műveletsort ciklusnak nevezzük, utalva az ismétlődő, ciklikus végrehajtásra. Az i változó feladata éppen annak a nyilvántartása, hányadszor fut le a ciklus. Emiatt az i -t ciklusváltozónak is szokás nevezni.

- A termék beolvasása. A soron következő termék adatait beolvassuk a számítógépbe, majd legfeljebb öt lépésben eldöntjük, melyik termékcsoporthoz tartozik, s ennek megfelelően számítjuk a rá vonatkozó jutalékot (J). Ha az adott termék - tulajdonságainál fogva, például kódja miatt - egyik termékcsoporthoz sem illeszthető be, hibajelzést adunk, és leállítjuk a programot. Ilyen jellegű hiba oka lehet például az, hogy a termék adatait tévesen lyukasztották a lyukkártyára, vagy más, ide nem tartozó termék kártyája keveredett a kártyacsomagba stb.
- A beolvasott termék utáni jutalék értékét az S nevű gyűjtőrekeszben tároljuk, amelynek tartalmát mindig az éppen kiszámított jutalék értékével növeljük. Így S -ben mintegy halmozódik a jutalék értéke, J rekesz tartalmát pedig felülírhatjuk a következő terméktől származó jutalékkal. A módszer előnye, hogy a tizedik termék átfutása után az S rekeszben azonnal hozzáférhető a teljes haszon értéke; az egyes termékcsoporthoz közös J rekeszt használnak, ami az operatív tár felhasználásában megtakarítást eredményez. Figyeljük meg, hogy mennyire lényeges a program helyes működése szempontjából az $S := 0$ értékadás! Ezzel biztosítjuk azt, hogy a gyűjtőrekesz tartalma az első termék után számított jutalék képzése előtt zérus legyen, és ne fordulhasson elő az, hogy S tartalma nem nulláról indul.
- Miután S -ben rögzítettük az aktuális haszon értékét, megvizsgáljuk, hogy eljutottunk-e már a tizedik termékhez, vagyis i értéke kisebb-e N -nél. Ha igen, i értékét eggyel megnöveljük ($i \neq N$ esetén), és a ciklust újra lefuttatjuk, most már a következő ($i + 1$ -edik) termékre nézve. Ha i már nem kisebb, mint N , akkor éppen a tizedik termék adatait dolgoztuk fel, és többször már nem futtatjuk le a ciklust, hanem rátérünk az eredmény kinyomtatására (S aktuális tartalma).

8.2.2 CIKLUSSZERVEZÉS

A fent ismertetett példa igen jól szemlélteti egy tipikus programciklus felépítését. A ciklus egyike azoknak a műveleteknek, amelyek a legközelebb állnak a számítógép adottságaihoz: a nagy műveleti sebesség következtében a gép könnyen és gyorsan végrehajtja az előírt műveletsorozatot akár több ezerszer is egymás után, mindig más és más kiindulási értékekkel. Ciklust használhatunk például egy adott érték egyre pontosabb megközelítésére, itt minden következő ciklusfuttatás valamivel pontosabb eredményt szolgáltat, vagyis "közelebb visz" a pontos megoldáshoz. A ciklusokat és általában a ciklusszervezést a programozási technika lépten-nyomon használja, ezért érdemes bővebben is kitérni rá.

Egy ciklus általában az alábbi fő részekre osztható:

1. Előkészítő rész - a kezdeti értékek és a korlátozó feltételek beállítása, valamint értékadás a ciklusváltozónak. Előbbi példánkban $N:=0$, $S:=0$, $i:=1$.
2. Ciklusmag - legalább egy utasítás, amely többször egymás után végrehajtódik. A ciklusmag tetszőlegesen nagy számú utasításból is állhat. Előző példánkban a ciklusmag a termék beolvasásától a kiszámított jutalék rögzítéséig (S-ben) tart.
3. Döntési rész - a ciklusváltozó értékét egy korlátozó értékhez hasonlítjuk, és az összehasonlítás eredményétől, a kiugrási feltétel teljesülésétől függően vagy kiugrunk a ciklusból, vagy ismét futtatjuk azt. Példánkban az i ciklusváltozó aktuális értékét hasonlítjuk N -hez, és mindaddig ismételjük a ciklust, amíg csak i értéke kisebb N -nél.
4. Módosító rész - a ciklusmag lefutása után a ciklusváltozó (esetünkben tulajdonképpen ciklusszámláló) értékét módosítja. Ezzel tehát beállítjuk a ciklusváltozó értékét a következő ciklusfutás számára. Példánkban ezt az $i:=i+1$ utasítás végzi.
5. Kiugrás a ciklusból - ha a döntési rész olyan eredményt szolgáltat, hogy a kiugrási feltétel teljesül, tehát a ciklust többször már nem kell lefuttatni, ezután a ciklusmagot követő utasítások következnek (a példában $i=N$ esetén S tartalmának kinyomtatása). Figyeljük meg, hogy a ciklusból való kiugrás

nem feltétlenül a kiugrási feltétel teljesülésének eredményeképpen jön létre, hiszen egy olyan termék esetében, amely nem sorolható egyik termékcsoporthoz sem, hibajelzés kíséretében magából a ciklusmagból ugrunk ki, és állítjuk le a programot.

Megjegyzés. A döntési rész és a módosító rész sorrendben felcserélhető, illetve elhelyezhető mind a ciklusmag előtt, mind a ciklusmag után. Ha példánkban az i ciklusváltozó értékét a döntési rész előtt növeltük volna eggyel, akkor a kilencedik termék adatainak feldolgozása után $i=10$ lett volna, és $i=N$ miatt kiugrottunk volna a ciklusból; hamis eredményt kaptunk volna! A probléma áthidalható például úgy, hogy a kiugrási feltételt most így fogalmazzuk meg: " i értéke kisebb vagy egyenlő N -nel". Ekkor ugyanis $i=10$ esetén még egyszer lefut a ciklus, feldolgozva mind a tíz termék adatait.

A ciklusváltozó

Az elmondottakból következik, hogy

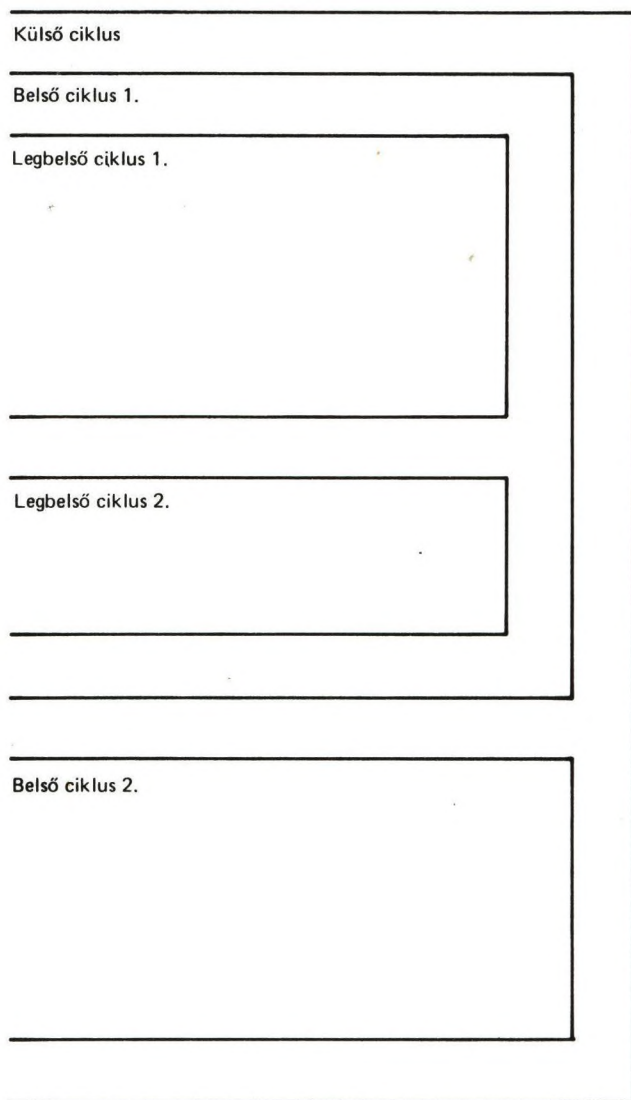
mindig nagyon jól át kell gondolni, mekkora lesz a ciklusváltozó értéke a ciklusból való kiugráskor, és ezzel összhangban kell megválasztani a kiugrási feltételt.

Ugyanakkor természetesen nem kötelező a ciklusváltozó kezdeti értékét $+1$ -re állítani. A ciklusváltozó kezdeti értéke tetszőleges (egész) szám, és a ciklus futásai során értéke mind pozitív, mind negatív irányban változtatható. Az előző feladatot úgy is megoldhattuk volna, hogy $i:=10$ értékadással tizről indulva minden egyes termék adatainak feldolgozása után i értékét eggyel csökkentjük volna (visszaszámlálás). Ekkor $i=0$ lett volna a kiugrási feltétel, és a ciklusváltozó aktuális értéke azt jelezte volna, hogy hány termék feldolgozása van még hátra. Hasonlóképpen nem kötelező a ciklusváltozó értékét eggyel módosítani. A ciklus szervezését mindig a programozó határozza meg az adott feladat sajátosságait figyelembe véve.

Belső ciklusok

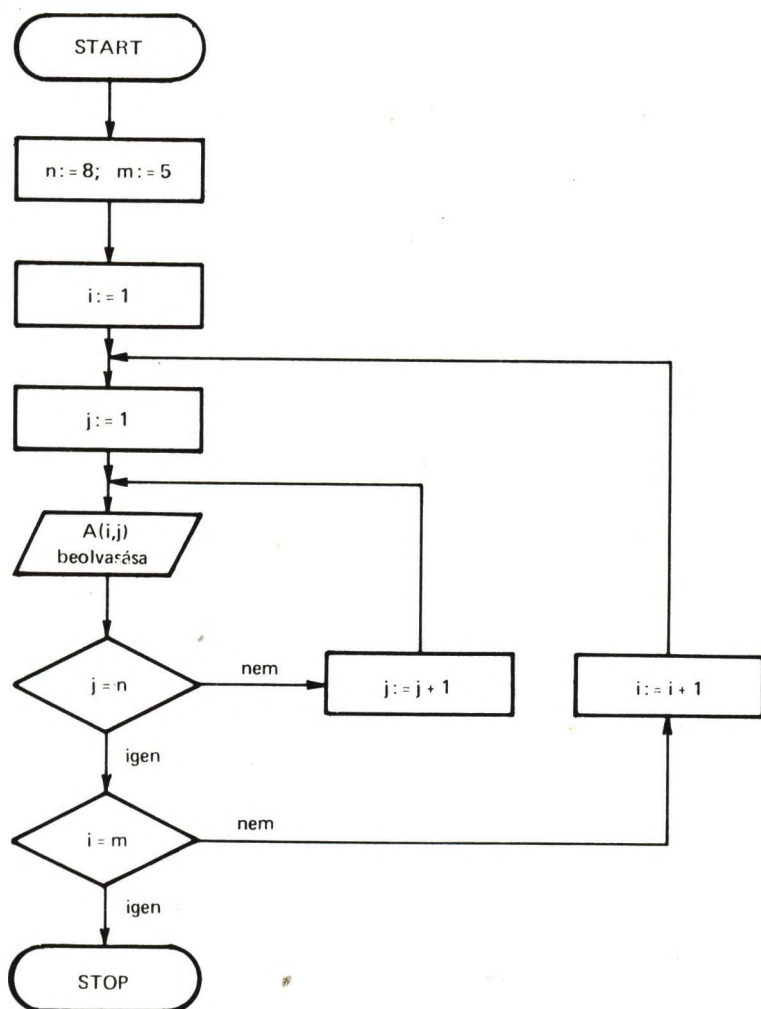
Megengedett több ciklus "egymásba skatulyázása" is. Ekkor a ciklus magja ismét egy teljes ún. belső ciklust is tartalmaz, amely ismét magában foglalhat teljes ciklusokat (lásd a 38. ábrát). Ilyenkor a legbelső ciklus fut egészen addig,

amíg a saját kiugrási feltétele nem teljesül. Utána fut az eggyel kijebb levő ciklus és így tovább. Egyes programozási nyelvek korlátozzák az egymásba skatulyázható ciklusok maximális számát, illetve a ciklusváltozó értékkészletét. Minthogy ez nyelvenként különböző lehet, részletes ismertetésére itt nem térünk ki.



38. ábra. Ciklusok egymásba skatulyázása

A ciklusok egymásba skatulyázását a 39. ábrán egy egyszerű példán mutatjuk be. A feladat az, hogy be kell olvasni az operatív tárba egy kétdimenziós tömb (mátrix) elemeit sorfolytonosan. Álljon az A mátrix öt sorból és nyolc oszlopból, egyes elemeire pedig hivatkozzunk kettős indexszel. Tehát $A(i, j)$ az A mátrixnak azt az elemét jelenti, amely az i -edik sor és a j -edik oszlop metszéspontjában található. Sorfolytonos beolvasás alatt azt értjük, hogy a mátrix elemeit soronként olvassuk be. A sorrend tehát a következő lesz: $A(1, 1), A(1, 2), \dots, A(1, 8), A(2, 1), A(2, 2)$ stb.



39. ábra. Mátrix sorfolytonos beolvasása

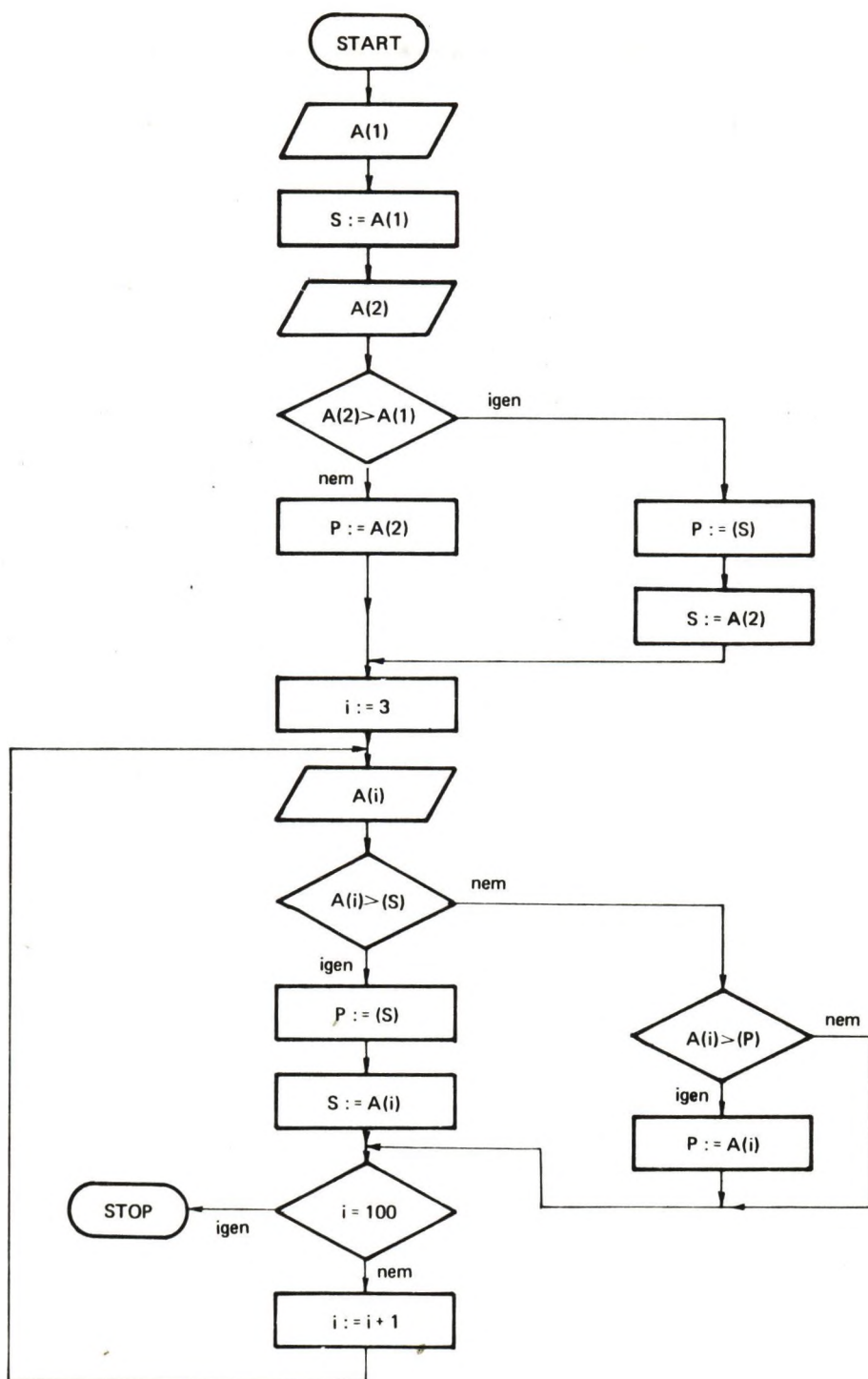
A feladatot kettős ciklussal oldjuk meg. A sorok sorszáma az i , az oszlopok sorszáma a j ciklusváltozó tartalmazza. A folyamatábrából látható, hogy i értéke mindaddig nem változik, amíg a $j=n$ feltétel nem teljesül, vagyis míg egy adott sor mind a nyolc elemét be nem olvastuk (fut a belső ciklus). Ezután i értékét eggyel növeljük, és $j=1$ -től kezdve beolvassuk a mátrix következő sorának elemeit. A belső ciklus tehát jóval többször fut le, mint a külső.

A különböző feltételvizsgálatok, elágazások következtében egyes folyamat-ábrák igen bonyolulttá válhatnak. Alább bemutatunk egy példát összetettebb feladat megoldására.

2. példa

Adott 100 szám, kikeresendő közülük a két legnagyobb. Feltételezzük, hogy a számok mind valósak. A megoldás menetét a 40. ábrán tüntettük fel. A 100 db számot közös egységbe (tömbbe) foglaltuk, amelyet A -nak nevezünk a feladatban. Az egyes számokra - a tömb elemeire - indexekkel hivatkozunk (például $A(78)$ jelenti az A tömb 78. elemét, vagyis a számsorozat 78. tagját). A megoldás végén S -sel jelölt rekesz fogja tárolni a sorozat legnagyobb értékű tagját, P -vel jelölt rekesz pedig a második legnagyobb értékű tagját.

A program működése a következő: Beolvassuk az 1. számot $A(1)$ -et, és elhelyezzük S -ben. Ezután beolvassuk $A(2)$ -t, és megvizsgáljuk, hogy nagyobb-e $A(1)$ -nél. Ha igen, S tartalmát átírjuk P -be, és az $A(2)$ kerül S -be. Ha nem, $A(2)$ P -be kerül. A továbbiakban ciklust szervezünk i ciklusváltozóval, amely egészen 100-ig beolvassa a hátralevő 98 számot, és megvizsgálja, hogyan módosítja az ujonnan beolvasott elem az S és P rekesz tartalmát. Ha $A(i)$ nagyobb S tartalmánál, biztosan nagyobb P tartalmánál is, tehát S tartalmát átírjuk P -be, $A(i)$ pedig S -be kerül. Ha $A(i)$ kisebb S tartalmánál, akkor ismét két eset lehetséges: vagy nagyobb, vagy kisebb P tartalmánál. Ha nagyobb, akkor $A(i)$ -t beírjuk P -be, ha nem nagyobb, akkor mind S , mind P tartalma változatlan marad. A ciklusváltozó értékét megnövelve beolvassuk $A(i+1)$ -et, és hason-



40. ábra. A 2. példa megoldása

lőán folytatjuk az eljárást $i=100$ -ig. Amennyiben a sorozat legnagyobb értékű tagja egynél többször fordul elő a sorozatban, S és P ezt a számot fogja tartalmazni. Ennek az igazolását az olvasóra bizzuk.

A feladat bemutatásával főleg arra akartunk rávilágítani, hogyan lehet egy bonyolultabb feltételrendszert elemi "igen-ném" jellegű döntésekre lebontani. Látható, hogy az összetettebb feltétel kezelése nem jelent járulékos nehézséget, csupán az elemi döntések száma növekszik.

Végül megjegyezzük, hogy egy adott feladat megoldására igen sokféle folyamatra (algoritmus) készíthető, itt csak egy-egy lehetséges megoldást mutatunk be.

9. Bevezetés az ESZR alapsoftware-be

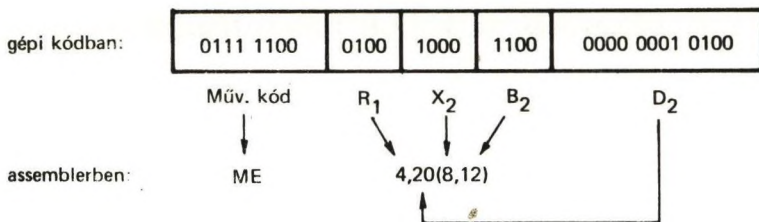
9.1 SOFTWARE

A folyamatábra-készítés tárgyalásakor már esett néhány szó az un. rendszerprogramokról, amelyek a számítógép üzemeltetését végzik. E speciális programok nélkül a korszerű számítógépek gazdaságosan nem üzemeltethetők; a gép software-ellátottsága legalább olyan fontos tényező, mint a gyors, megbízható hardware-felépítés. A software-fejlesztésben még igen sok a kiaknázatlan lehetőség, ezért mind a gyártó cégek, mind a felhasználók a software bővítését, fejlesztését szorgalmazzák a már meglévő géptípusokhoz. A software egyébként a gép árának jelentős részét teszi ki.

9.1.1 AZ ALAPSOFTWARE KIALAKULÁSA

Tekintsük át vázlatosan, hogyan alakultak ki a mai korszerű operációs rendszerek, és hogyan befolyásolta a software fejlődése a felhasználó és a számítógép kapcsolatát.

A régebbi elektronikus számítógépek programozását gépi kódban végezték, és a felhasználó maga kezelte a gépet. Mivel a gépi kód használata rengeteg tévedési lehetőséget rejt magában, a programozás lassu és igen nehézkes volt. Ezért a felhasználó munkájának megkönnyítésére megalkották az un. assembly szintű programnyelveket, az assemblereket (assembly = összeállító program). Az assembly szinten megírt program mindössze annyiban különbözik gépi kódú megfelelőjétől, hogy a műveleti kódokat és az operanduscimeket szimbolikusan fejezi ki. Egyébként minden egyes gépi utasításnak megfelel egy assembler utasítás (kivételek a makroutasítás, lásd alább). Például egy lebegőpontos szorzás alakja



A makroutasítások olyan speciális assembler utasítások, amelyek több gépi utasítást fejeznek ki. Főleg olyan műveletekre célszerű makroutasításokat használni, amelyek igen gyakran fordulnak elő a programban (például input-output utasítások, file-definiáló utasítások stb.). Ezáltal elérhető, hogy a gyakori műveleteket nem kell minden esetben újra programozni, hanem az egyszer már megírt makroutasításra tetszőlegesen sok esetben hivatkozhatunk.

Mivel az assembly szinten megírt program a gép számára érthetetlen szimbólumokat tartalmaz, közvetlenül nem futtatható, előbb át kell alakítani gépi kóddá, és úgy tölteni be az operatív tárba. Ezt a feladatot a program fordításának nevezzük, és az assembler nevű fordítóprogram végzi el. Ugy tűnik, hogy a felhasználó érdekében véghezvitt egyszerűsítés a számítógéprendszert bonyolultabbá tette, hiszen most már fordítóprogramot is ki kellett dolgozni a géphez. Az ellentmondás csupán látszólagos: a fordítóprogramot a gyártó cég egyszer s mindenkorra kidolgozza, és a géppel együtt bocsátja rendelkezésre, s a szimbólumok használata igen nagy mértékben egyszerűsíti a programozói munkát. Assembly szinten a gép felhasználói regiszterei (16 általános célú és 4 lebegőpontos regiszter) mind hozzáférhetők, ezért az assembler nyelvet gépre orientált programnyelvnek tekintjük.

A programfuttatás további automatizálására és az emberi beavatkozás mértékének csökkentésére hozták létre az első vezérlő rendszerprogramokat (system monitor, executive program, master program stb.). Ezek folyamatosan nyilvántartják a felhasználói programok állapotát, hibavizsgálatot (esetleg javítást) végeznek, és nagyrészt önállóan dolgoznak. Az operátornak mindössze elindítania kell a számítógéprendszert, és az esetleges fennakadásokat kell áthidalnia. A vezérlőprogramokból fejlődtek ki a mai bonyolult operációs rendszerek, amelyek lehetővé teszik a számítógéprendszer majdnem teljesen automatikus üzemeltetését.

A számítástechnika előrehaladásával azonban a megoldandó feladatok is mind bonyolultabbá, mind terjedelmesebbé váltak, szükség volt ún. feladat-orientált programnyelvek kidolgozására. Ezek állnak legközelebb a programozó gondolkodásmódjához, s ezáltal számos feladat megfogalmazását egyszerűvé teszik. Fenti tulajdonságaik miatt magas szintű programnyelveknek is nevezik őket. Az ESZR-ben használt magas szintű programnyelvek a következők:

- FORTRAN (FORmula TRANslation - képletfordítás); igen jól használható a sok számítási műveletet igénylő tudományos-műszaki számítások programozására;
- COBOL (Common Business Oriented Language - kereskedelem-orientált nyelv); főleg adatfeldolgozási munkáknál használható, nagy tömegű adaton végzett viszonylag egyszerű számításokhoz;
- PL/I (Programming Language I); univerzális programozási nyelv, egyesíti magában a FORTRAN és COBOL adta lehetőségeket;
- R. P. G. (Report Program Generator); segítségével könnyen és egyszerűen készíthetők jelentések adatállományokról, különféle szempontok szerint, táblázatok összeállításánál is előnyösen használható;
- ALGOL^{*} (ALGOrithmic Language - algoritmikus nyelv); főleg Európában terjedt el, tudományos-műszaki számítások programozására szolgáló magas szintű nyelv.

Természetesen a magas szintű nyelven írt programokat is gépi kódra kell fordítani, hiszen a gép számára érthetetlen szimbólumokat tartalmaznak. Ezt a fordítási feladatot olyan fordítóprogramok végzik, amelyek minden egyes magas szintű nyelven írt utasításnak egy vagy több gépi utasítást feleltetnek meg (kompilálás). Innen az elnevezésük: kompilátorok. Figyeljük meg, hogy ASSEMBLER nyelven írt programok fordításakor nincs szükség kompilálásra, hiszen egy ASSEMBLER utasítás egy gépi utasításnak felel meg!

^{*} Az ESZR/DOS nem tartalmazza az ALGOL fordítóprogramot.

Megjegyzés. A magas szintű programnyelvek szabályait (szintaxis) nemzetközileg rögzítették, így a programokat a számítógéptípustól függetlenül, egységesen lehet írni. A nemzetközileg szabványosított programnyelveket hivatkozási nyelveknek is nevezik. Az érintett gép fordítóprogramjai alkalmazkodnak a nemzetközi szabványokhoz, és a forrásnyelven megírt programokat az adott gép utasításaira kompilálják. Ezzel a programozási munka csaknem teljesen függetlenné vált magától a számítógéptől, hiszen ha a gépnek megvan a kívánt fordítóprogramja (pl. COBOL), akkor a COBOL nyelven írt programokat egységesen értelmezni tudja.

Valamely magas szintű nyelvnek több - azonos géptípusra írt - fordítóprogramja is lehet, ezek különböző mértékben változtatják a hivatkozási nyelvadta lehetőségeket. Például a FORTRAN IV kompilátor lényegesen többet valósít meg a hivatkozási nyelvből, mint a FORTRAN II stb. A forrásnyelvek általában mindig eltérnek többé-kevésbé a hivatkozási nyelvtől, annak lehetőségeit szűkítik, esetleg bővítik.

9.1.2 OPERÁCIÓS RENDSZEREK

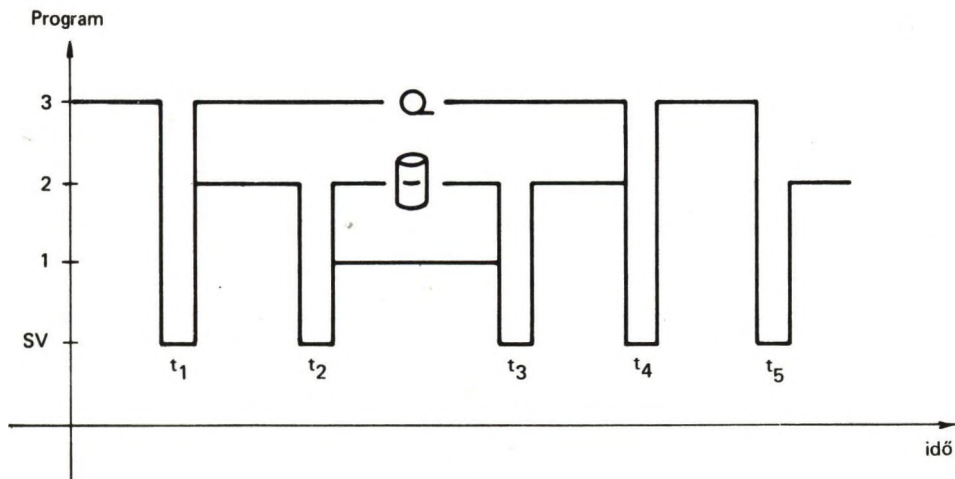
A III. generációs számítógépek megjelenése a központi egység egyes paramétereinek további javulását eredményezte (például műveleti sebesség stb.), a perifériális egységek fejlődése azonban nem tartott lépést ezzel, így tovább nőtt a sebességviszonyok miatti feszültség a központi gép és perifériái között. Meg kellett találni a módját annak, hogy a számítógéprendszer minden eleme egyidejűleg csaknem százszázalékosan kihasználható legyen, ne kelljen például a központi egységnek megvárnia, míg a viszonylag lassu input-output befejeződik stb. Ez a cél úgy vált elérhetővé, hogy az operatív tárban egyszerre több programot helyeztek el, s bármelyik program input-output műveletével egyidőben a többi használhatta a központi egységet. A vázolt elv az ún. multiprogramozás elve, amely összefügg a korszerű operációs rendszerek kialakulásával.

Operációs rendszeren egy olyan programrendszert értünk, amely elvégzi a felhasználói programok teljesen automatizált futtatását multiprogramozási üzemmódban.

Az operációs rendszer használata tehát minimálisra csökkenti az operátori beavatkozás szükségességét, valamely felhasználói program végrehajtása után a rendszer automatikusan áttér a következő program futtatására, és végül: csaknem optimálisan gazdálkodik a számítógérendszer erőforrásaival. Az alábbiakban vázlatosan ismertetjük az operációs rendszer szolgáltatásait.

a) Multiprogramozás

Mivel egy-egy program rendszerint nem használja ki teljesen a számítógép un. erőforrásait, célszerű a gépen egyszerre több programot futtatni úgy, hogy a rendelkezésre álló erőforrásokat* felosztjuk az egyes programok között. A módszer előnye az, hogy valamely program input-output művelete alatt egy másik program központiegyység-időhöz juthat, például számításokat végeztethet az aritmetikai-logikai egységgel. Az egyes programok különböző prioritásuak; a nagyobb prioritású program elsőbbséget élvez a számítógép erőforrásainak igénybevételénél, tehát adott esetben megszakíthatja az alacsonyabb prioritású program futását. A 41. ábrán feltüntettük egy háromszorosan multiprogramozott rendszer idődiagramját input-output műveletekre vonatkozóan.



41. ábra. Multiprogramozás

* Erőforrás mindaz a fizikai vagy a gép alapsoftware-e által nyújtott eszköz, amely az adott számítógépben a programok futásakor szükséges lehet (például memóriaterületek, csatornák, I/O eszközök, vezérlőprogramok stb.).

Az ábrán a 3. program a legmagasabb prioritású. Amint ez a program input-output műveletet akar végezni, rövid időre a supervisorhoz (felügyelő-program, lásd 173. old.) kerül a vezérlés, amely elemzi a megszakítási okot, és elindítja a kijelölt perifériát (t_1). Ezután a 2. program fut mindaddig, míg ez is input-output megszakítást kér (t_2), s a vezérlés most az 1. programhoz kerül. Amint a 2. program befejezte az input-output műveletet, prioritásánál fogva megszakítja az 1. program futását (t_3), majd visszakapja a vezérlést egészen addig, míg a 3. program be nem fejezte input-outputját (t_4). Tegyük fel, hogy t_5 -ben a 3. program befejeződik, így ismét a 2. program futhat.

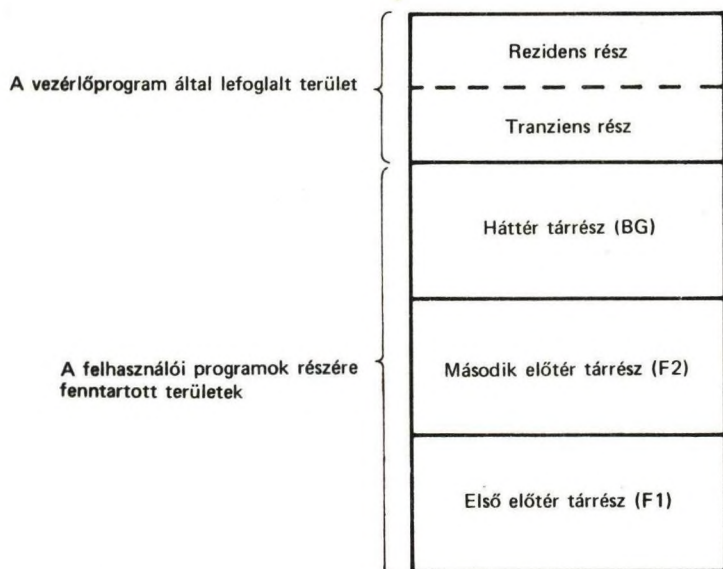
Az ábra egyben igen fontos multiprogramozási elvet szemléltet:

Viszonylag nagy perifériaigényű programokat célszerű magasabb prioritással ellátni.

Példánkban az 1. program szinte nem is igényel input-output műveletet; ha most ez a program lenne a legmagasabb prioritású, akkor ez monopolizálná a számítógép erőforrásait, s a többi programnak végig kellene várnia, amíg az 1. program befejeződik, csak akkor indíthatná el input-outputját. Ezzel szemben a feltüntetett - helyes - szervezés mellett a magasabb prioritású programok input-output műveletei alatt az alacsonyabb prioritásúak használhatják a központi egységet, tehát futhatnak.

Az egyes operációs rendszereknél a multiprogramozhatóság korlátozott. Így például az ESZR/DOS operációs rendszer legfeljebb három programot futtathat egyidőben, ezt a lehetőséget természetesen nem kötelező kihasználni. A program futása alatt az operatív tárban kell lennie az operációs rendszer bizonyos programjainak, amelyek nélkülözhetetlenek a számítógép üzemeltetéséhez. Ezek alkotják az ún. tárrezidens részt, az egyéb rendszerprogramok, amelyekre csak átmenetileg van szükség, az operációs rendszer tranziens részét képezik, és csak átmenetileg tartózkodnak az operatív tárban. Az operációs rendszer tehát állandóan lefoglal egy bizonyos tárterületet rezidens és tranziens része számára. A fennmaradó tárrész felosztható a multiprogramozásban részt vevő programok között.

Az operatív tár egy lehetséges felosztását mutatja a 42. ábra háromszoros multiprogramozás esetén.



BG = background (háttér); F2, F1 = foreground 2, illetve 1 (2. illetve 1. előtér)

42. ábra. Az operatív tár felosztása az ESZR/DOS szempontjából

Az egyes tárrészekben futtatott programok prioritása: F1, F2, BG a prioritás csökkenő sorrendjében. Lehetőség van arra, hogy a teljes felhasználói tárrészt a BG vagy a BG és F2 részére tartsuk fenn, ami természetesen szűkíti a multiprogramozhatóságot. A tárfelosztást, valamint az egyéb paramétereket, amelyekkel az operációs rendszert üzemeltetni kívánjuk, a rendszer generálásakor kell megadni. Minthogy az operációs rendszer általában igen sok programból áll, minden felhasználó maga választhatja ki azokat a programokat, amelyekre szükség van. A teljes operációs rendszer mágnesszalagon található, és rendszergeneráláskor erről a szalagról a továbbiakban szükséges programokat (modulokat) átmásoljuk mágneslemezre. (Maga a rendszergenerálást végző program is rajta van az említett mágnesszalagon.)

Mivel a multiprogramozás elég nagy perifériaigényű lehet, csak a minimális konfigurációt jelentősen meghaladó kiépítettség esetén van értelme.

b) Kötegelt feldolgozás (batch processing)

Minden egyes tárrészben lehetőség van kötegelt feldolgozásra. Ez azt jelenti, hogy az operációs rendszer automatikusan áttér a következő program futtatására, miután az előzőt végrehajtotta. A programfutas indításánál nincs szükség emberi beavatkozásra. Megjegyezzük, hogy a háttér tárrészben csak kötegelt feldolgozás lehetséges, az előtér tárrészek alkalmasak kötegelt és egyedi feldolgozásra is. Azt, hogy az előtér tárrészeket hogyan kívánjuk üzemeltetni, rendszergeneráláskor kell megadni.

c) A felhasználói program végrehajtása

Mivel a felhasználói programokat általában valamilyen forrásnyelven írjuk, futtatás előtt le kell őket fordítani gépi kódra. A programfordítást a már megismert fordítóprogramok végzik. A futtatható programnak minden esetben fix és abszolút operatív tárbeli címekkel kell rendelkeznie, tehát általában nem futtatható a tár tetszőleges területén. Ez a megkötöttség nagyban nehezíti a tár gazdaságos kihasználását, ezért a fordítóprogramok nem közvetlenül futtatható formátumu programot szolgáltatnak a fordítás eredményeként, hanem olyan gépi kódú programot, amelyben a konkrét tárhivatkozások (címek) nincsenek kitöltve. A lefordított programrészek (modulok) előkészítését a futtatásra az ún. szerkesztőprogram végzi el, amely abszolutizálja az előforduló operatív tárbeli címeket, és szükség esetén összekapcsolja a több részből álló program moduljait.

Az operációs rendszer tehát az alábbi három formában előforduló programokat kezel:

- forrásnyelvi programok,
- modulprogramok (tárhivatkozások nélküli modulok),
- fázisprogramok (abszolút címekkel rendelkező programok).

A fenti programok ideiglenes vagy állandó tárolására bizonyos mágneslemez-területek szolgálnak, ezek az ún. programkönyvtárak.

Az előforduló könyvtárak a következők :

- forrásnyelvi könyvtár (elemei könyvek),
- tárgynyelvi könyvtár (elemi modulok),
- abszolút könyvtár (elemei fázisok).

Megjegyezzük, hogy az operációs rendszer működőképes a forrásnyelvi és tárgynyelvi könyvtár nélkül is, de abszolút könyvtárra feltétlenül szükség van. Ebben található az operációs rendszer összes programja (komponense) is. A felhasználó kijelölhet saját, ún. privát könyvtárakat is saját személyes használatára.

9.2 ESZR OPERÁCIÓS RENDSZEREK

Az **ESZR** gépek a gyakorlatban főleg kétféle operációs rendszerrel működnek:

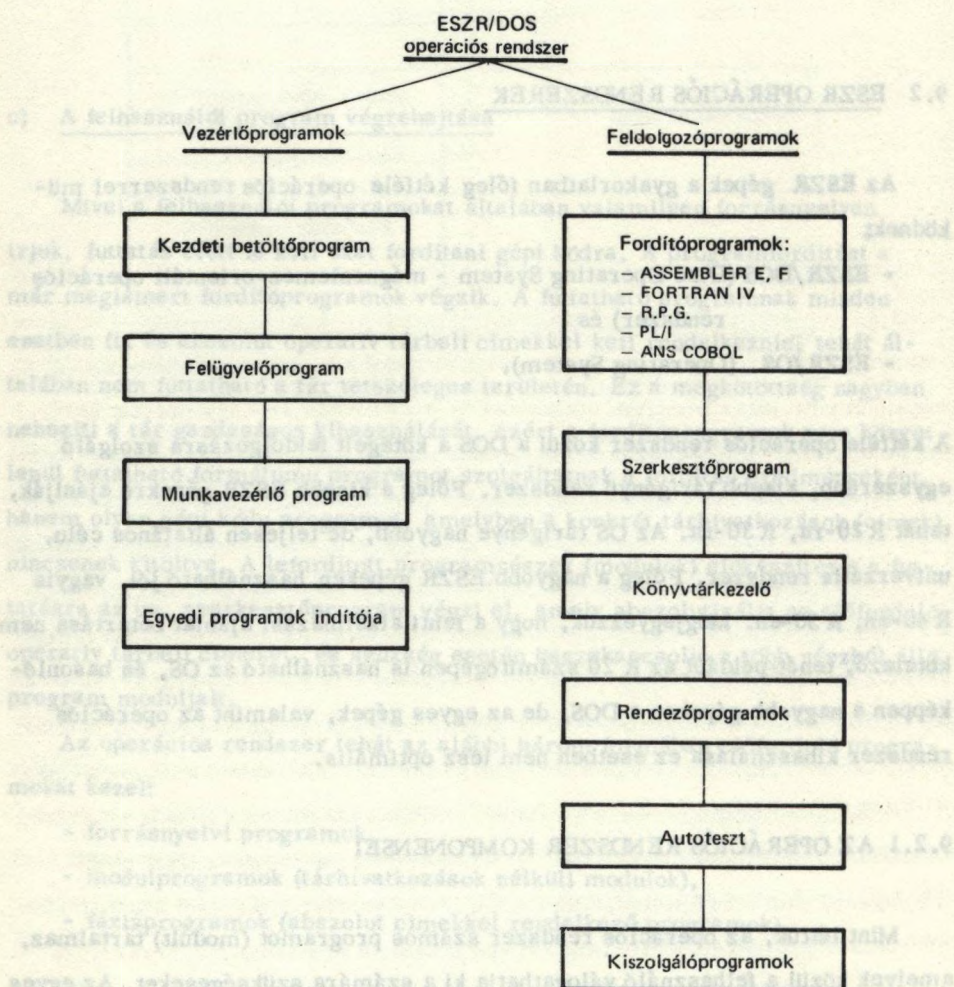
- **ESZR/DOS** (Disc Operating System - mágneslemez-orientált operációs rendszer) és
- **ESZR/OS** (Operating System).

A kétféle operációs rendszer közül a DOS a kötegetelt feldolgozásra szolgáló egyszerűbb, kisebb tárigényű rendszer. Főleg a kisebb **ESZR** gépekre ajánlják, tehát R 20-ra, R 30-ra. Az OS tárigénye nagyobb, de teljesen általános célú, univerzális rendszer. Főleg a nagyobb **ESZR** gépeken használható jól, vagyis R 40-en, R 50-en. Megjegyezzük, hogy a fenti alkalmazási ajánlat betartása nem kötelező, tehát például az R 20 számítógépen is használható az OS, és hasonlóképpen a nagyobb gépeken a DOS, de az egyes gépek, valamint az operációs rendszer kihasználása ez esetben nem lesz optimális.

9.2.1 AZ OPERÁCIÓS RENDSZER KOMPONENSEI

Mint láttuk, az operációs rendszer számos programot (modult) tartalmaz, amelyek közül a felhasználó válogathatja ki a számára szükségeseket. Az egyes operációs rendszerek eltérő terjedelműek, több vagy kevesebb programból áll-

nak. Alább egy igen elterjedt operációs rendszer, az ESZR/DOS felépítését mutatjuk be a fogalmak megvilágítására. Ennek a komponensei alapvetően két csoportra oszthatók: vezérlőprogramokra és feldolgozóprogramokra. A vezérlőprogramok magát a rendszert irányítják, a feldolgozóprogramok pedig különféle szolgáltatásokat nyújtanak a felhasználó számára. Az operációs rendszer vázlatos felépítését a 43. ábra szemlélteti.



43. ábra. Az ESZR/DOS komponensei

1. Kezdeti betöltőprogram

Az operációs rendszer üzembe helyezését szolgálja. A gépkezelő (operátor) beolvastat az operatív tárba 24 byte-nyi információt a rendszert tartalmazó mágneslemezről, ez elegendő arra, hogy behozza a tárba a kezdeti betöltőprogram többi részét. Ez aztán folyamatosan behozza az operációs rendszert, és átadja neki a vezérlést. A programbetöltés kezdetekor egyes rendszerparaméterek állapota módosítható, ha erre szükség van.

2. Felügyelőprogram (supervisor)

A feldolgozás irányítását végzi. Főbb funkciói a következők:

- a megszakítások kezelése;
- az input-output műveletek lebonyolítása, hibavizsgálat;
- programok behívása az abszolút könyvtárból;
- a multiprogramozás szervezése;
- kapcsolattartás az operátorral stb.

A supervisor tárrezidens része állandóan az operatív tárban tartózkodik, tranziens rutinjai (a program bizonyos részei) azonban szükség szerint kerülnek be a tranziens tárterületre (lásd a 42. ábrát).

3. Munkavezérlő program (job control)

Feladata a csoportos feldolgozás megszervezése. Minden egyes program - a forrásnyelvi utasításokon kívül - tartalmaz néhány un. munkavezérlő utasítást, amelyek a munkavezérlő programnak szólnak. Általában lyukkártyára rögzítjük őket, és a forrásnyelvi program kártyái közé illesztjük. Ezek az un. vezérlőkártyák. Kitöltésük és helyük a programban kötött, az egyes job control utasítások formátumát és tartalmát egy külön nyelv, a munkavezérlő nyelv szabályozza (job control language - JCL). Vezérlőkártyákon keresztül vehetők igénybe a munkavezérlő program különféle szolgáltatásai stb. Vezérlőkártyán adjuk meg a kívánt fordítóprogram nevét, és vezérlőkártyákon rendelkezünk a lefordított program további sorsáról is.

4. Egyedi programok indítója (single program initiator - SPI)

A munkavezérlő programhoz hasonló funkciójú, bár lehetőségei korlátozottabbak. Olyan programok indítására használjuk, amelyek valamely nem csoportos feldolgozást végző tárrészben helyezkednek el. Ez a program csak egyetlen felhasználói programot indíthat, segítségével automatikus áttérés a következő programra nem lehetséges.

5. Fordítóprogramok (translator)

Feladatuk az illető forrásnyelven írt program lefordítása az egységes tárgynyelvre (gépi kód, tárhivatkozások nélkül). Az assembler két változata között a különbség főleg sebességben és tárigényben mutatkozik, bár az F változat néhány járulékos szolgáltatást is nyújt az E-vel szemben.

6. Szerkesztőprogram (linkage editor)

Feladata a fordítóprogramok által létrehozott tárgynyelvi modulok összekapcsolása futtatható programmá. Szükség esetén felvéteti a fázisprogramot az abszolút könyvtárba, ahonnan az egyszerűen behívható és utána azonnal futtatható.

7. Könyvtárkezelő

Feladata az operációs rendszer könyvtárainak és az esetlegesen felhasználói, privát könyvtáraknak a karbantartása. Főbb tevékenységei:

- privát könyvtárak létrehozása,
- új elem felvétele a könyvtárba,
- elemek törlése a könyvtárból,
- a könyvtár tartalomjegyzékének kiírása,
- a könyvtár egy elemének átmásolása más adattárolóra,
- a forrásnyelvi könyvtár tartalmának módosítása, javítása,
- elemek átmásolása egyik könyvtárból a másikba,
- a könyvtárak tömörítése (kondenzálás), a megüresedett helyek feltöltése programtologatással.

8. Rendezőprogramok (sort-merge)

Feladatuk különféle adatfile-ok kulcs szerinti rendezése. A rendezés két fő fázisra osztható: az operatív tárban elhelyezkedő adatállomány rendezése (sort), és előzetesen már rendezett file-ok összeállítása (merge). Az ESZR/DOS-ban két független rendezőprogram létezik:

- mágnesszalagos file-ok rendezésére,
- mágnesszalagos vagy mágneslemez file-ok rendezésére.

9. Autoteszt

Feladata a programok javításának, "belövésének" megkönnyítése. Lehetőséget nyújt arra, hogy magát a tárgynyelvi programot javítsuk, ezért a forrásprogramot nem kell minden esetben újra fordítani. Egyben segít a hibakeresésben is.

10. Kiszolgálóprogramok (utility)

Feladatuk adatfile-ok másolása egyik adathordozóról a másikra, illetve néhány speciális, ritkán igényelt tevékenység végrehajtása:

- másolás lyukkártyáról mágneslemezre,
- másolás lyukkártyáról lyukkártyára és sornyomtatóra,
- másolás lyukkártyáról mágnesszalagra stb.,
- mágneslemez inicializálása,
- tartalék sáv beállítása,
- lemezterület törlése stb.

9.2.2 A PROGRAM FUTTATÁSA DOS ALATT

Alább bemutatunk egy tipikus példát egy program futtatására, kiemelve a vezérkártyák szerepét, amelyek révén az operációs rendszer szolgáltatásait igényeljük.


```
// JOB JOBNEVE8 [tetszőleges megjegyzés]
```

```
// OPTION LIST, XREF, SYM, LINK
```

```
// EXEC PL/I
```

```
_____  
_____  
_____  
      ⋮  
_____  
      }
```

Forrásprogramkártyák

```
/*
```

```
// EXEC LNKEDT
```

```
// EXEC
```

```
_____  
_____  
_____  
      ⋮  
_____  
      }
```

Adatokat tartalmazó kártyák

```
/*
```

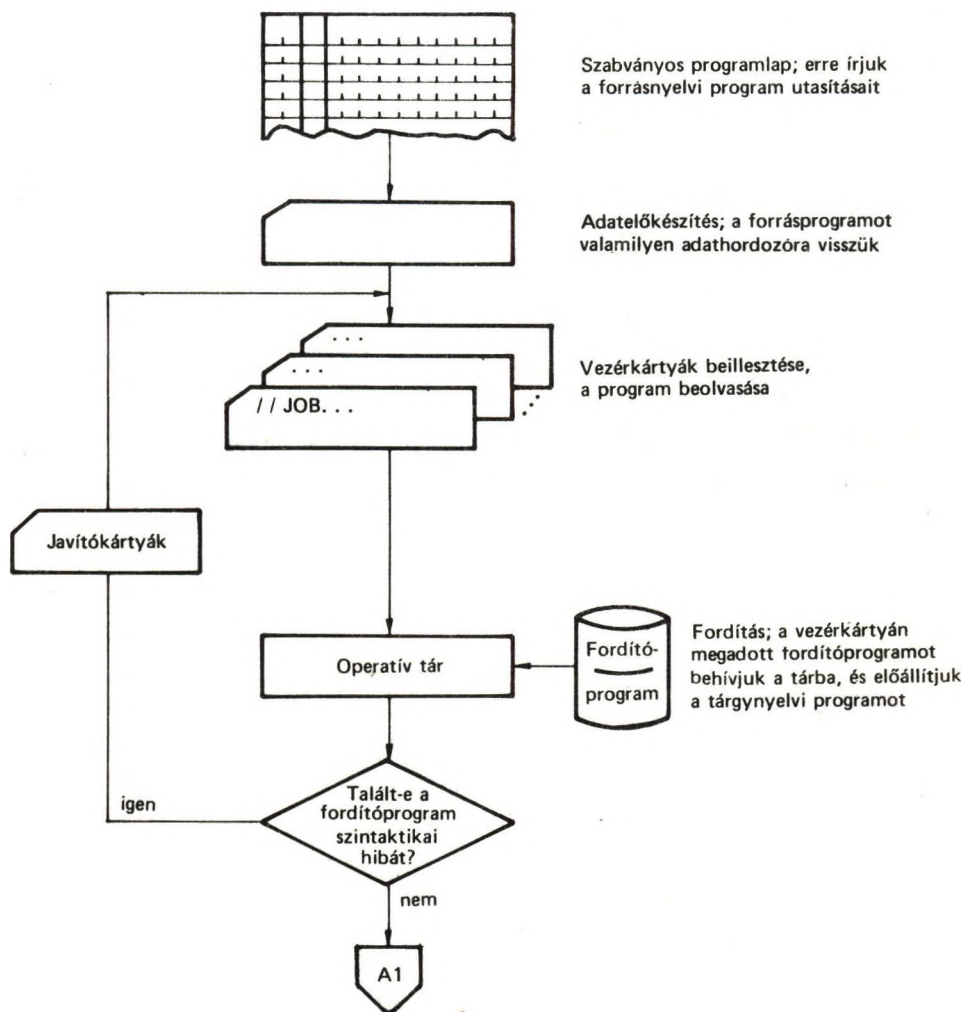
```
/&
```

Mint látható, a vezérkártyák jól megkülönböztethetők a program- és adat-kártyáktól: speciális karakterekkel kezdődnek. Az egyes vezérkártyák szerepe a következő:

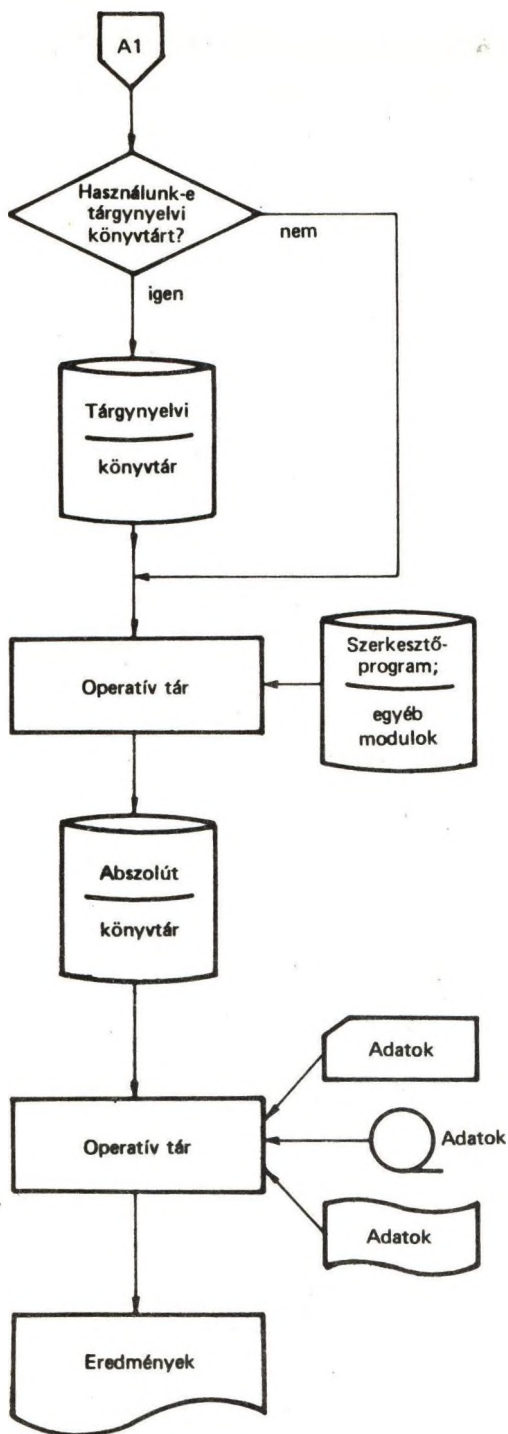
- JOB kártya - a jobot definiálja. A job neve maximálisan 8 karakterből állhat, utána tetszőleges megjegyzés következhet, például a programozó neve, dátum stb.
- OPTION kártya - ezen adható meg a rendszertől igényelt szolgáltatás. Például LIST = listát kérünk, SYM = szimbólumtáblázatot kérünk, LINK = szerkesztést kérünk. Az egyes opciókat a feladat természetétől függően vagy kérjük, vagy nem.
- EXEC kártyák - végrehajtást vezérlő információt tartalmaznak. Ezekkel indíthatjuk el a fordítás, a szerkesztés és a futtatás műveletét.
- /* kártyák - a feldolgozandó kártyák végét jelzik. Például fordítandó kártyák után, adatkártyák után stb.
- /& kártya - a job végét jelzi.

Megjegyezzük, hogy nem kötelező minden jobnak e három lépésből állnia. Egy job elvileg tetszőleges számú lépést tartalmazhat, az egyes lépések kezdetét EXEC kártyák jelzik. Az egyes lépések végrehajtási sorrendje általában nem cserélhető fel, tehát nem indítható el a szerkesztés addig, amíg a program fordítása be nem fejeződött stb.

Végezetül a 44. ábrán feltüntettük egy tipikus program utját a beolvasástól egészen az eredmények kinyomtatásáig.



44. ábra. Programfordítás, szerkesztés és futtatás
Megjegyzés: adatokat nem fordítunk.



Tárgynyelvi könyvtár; itt tároljuk a lefordított program moduljait

Szerkesztés; az önálló modulokból – esetleg egyéb modulok bevonásával – előállítjuk a fázisprogramot

Abszolút könyvtár; itt tároljuk a már futtatható fázisprogramot

Betöltés; a betöltőprogram segítségével a tárból visszük a fázisprogramot, és elindítjuk a futtatást

A futtatás befejeztével a számítási eredmények a sornyomtatón jelennek meg

44. ábra folytatása

10. File-szervezési eljárások

10.1 A SZERVEZÉS KIINDULÓ ADATAI

A file-szervezési módszerek feladata az, hogy megismertessék a felhasználóval az adott esetben leggazdaságosabb adatszervezési lehetőségeket, és így biztosítsák a központi egység idejének lehető legjobb kihasználását. Mindazonáltal a file-szervezés elmélete nem minden esetben ad a felhasználó kezébe bizonyíthatóan bevált módszert, sokszor igen jelentőssé válik a programozói tapasztalat, illetve a megfelelően szervezett próbálkozás. A későbbiekben az ESZR/DOS által támogatott file-szervezési eljárásokat mutatjuk be előbb nagy vonalakban, majd részletesebben. A file-szervezés célja mindenkor a gyors háttértárolók (mágnesszalag, mágneslemez) és a központi egység adatforgalmának optimalizálása. Alább a file-szervezéssel kapcsolatos legfontosabb definíciókat és tudnivalókat foglaljuk össze.

10.1.1 A FILE EGYES JELLEMZŐI

- a) Állandóság - a file bővítésére és rekordok törlésére vonatkozik. Státus a file, ha a rekordmódosítások csak kis százalékát képezik az összes file-hozzáfordulásnak (ilyen például a törzsadatfile, amely rendszerint kevés módosítást szenved). Változó a file, ha a rekordmódosítások (bővítések, törlések) az összes file-hozzáfordulás nagy százalékát teszik ki (ilyen például egy napi valutaárfolyam-file, ahol az egyes rekordok - az egyes valuták relatív értékei - napról napra változnak). Mindez igen jelentős abból a szempontból, hogy egyes file-szervezési eljárásokkal a gyakori rekordmódosítások hatékonyabban kezelhetők, mint másokkal.

- b) Aktivitás - az egyes rekordok feldolgozási (hozzáfordulási) gyakorisága. Célszerű a gyakrabban feldolgozandó rekordokat úgy elhelyezni, hogy elérési idejük minimális legyen.
- c) Terjedelem - a file adott esetben lehet igen nagy vagy igen kis méretű. Nagyon kis méretű file-oknál előfordul, hogy az alkalmazott feldolgozási mód nem befolyásolja a hatékonyságot, nagy terjedelmű file-ok esetében ez azonban meghatározó tényező lehet. A file várható bővülésének mértékét is figyelembe kell venni a file létrehozásakor; rendszerint valamilyen belátható időtartamra tervezzük a file méreteit, beleszámítva az esetlegesen várható bővítéseket/törléseket is. A kezdeti tervezésnek figyelembe kell vennie azt is, hogyan kezelhetők a file méretét befolyásoló időközi bővítések.

10.1.2 A FELDOLGOZÁS EGYES JELLEMZŐI

Szekvenciális feldolgozás esetén az input hozzáfordulásokat a file-nak megmegfelelő sorrendbe kell rakni. Ez a módszer különösen akkor előnyös, ha a file-okat mágnesszalagon tároljuk, de jól használható közvetlen elérésű tárolók esetén is.

Nem szekvenciális feldolgozás (random processing) esetén a file egyes rekordjaira tetszőleges sorrendben hivatkozunk. A közvetlen elérésű tárolók igen hatékony random feldolgozást tesznek lehetővé.

10.1.3 FILE-SZERVEZÉSI MÓDSZEREK

a) Szekvenciális file

A szekvenciális file-ban a rekordok fizikai sorrendben következnek egymás után. Általában növekvő kulcs szerint sorakoznak, de ez nem kötelező. Feldolgozásuk olyan sorrendben történik, ahogyan leolvasásra kerülnek. Például a 100. rekordot csak akkor olvassuk le, ha az előző 99-et már mind leolvastuk.

Az egyes rekordokat itt hosszadalmas lehet megtalálni. Általában nincs mód rekordokat törölni vagy a file-t rekordokkal bővíteni anélkül, hogy az egész file-t át ne szerveznénk. Ezért ez a feldolgozási mód akkor használható jól, ha valamennyi rekord feldolgozásra kerül a file egy hozzáfordulása során.

b) Indexelt szekvenciális file

Felépítése hasonló a szekvenciális file-éhoz, de különböző szintű indexek használatával az egyes rekordok is gyorsan hozzáférhetők, ami hatékony random feldolgozást tesz lehetővé. A file természetesen alkalmas szekvenciális feldolgozásra is. A bővítések egyszerűen megvalósíthatók anélkül, hogy át kellene szervezni a file-t, ugyanis a beszurandó rekordok kulcs szerinti helyükre kerülnek, illetve - ha az elsődleges adatterületen nem férnek el - egy erre a célra fenntartott ún. tulcsordulási területre, és ezt a területet logikailag a file-hoz tartozónak tekintjük. A programozónak igen kevés input-output szervezést kell végeznie, mert az operációs rendszer kezeli a file-t majdnem minden szempontból.

c) Direkt file

A direkt file-ban az egyes rekordok kulcsa és fizikai címe meghatározott kapcsolatban áll egymással. A kapcsolatot a felhasználó definiálja a file létrehozásakor. Ez a módszer főként ott használatos, ahol a file jellemzői nem teszik lehetővé a szekvenciális vagy indexelt szekvenciális szervezést, vagy pedig az egyes rekordok elérési idejét minimumra kell csökkenteni.

A direkt szervezési mód igen rugalmas, bár a felhasználó nagymértékben felelős a kulcs és a rekordcím közötti kapcsolat helyes megszervezéséért és programozásáért. Az operációs rendszer csupán a file-írási -olvasási rutínokat szolgáltatja.

10.1.4 AZ OPERÁCIÓS RENDSZER FUNKCIÓI

Az operációs rendszer magába foglalja az egyes adatelérési rutinokat, amelyek sorba állítják és vezérlik a központi egység és a mágnesszalag/mágneslemez közötti információátvitteket. Az operációs rendszerek lehetővé teszik a programozók számára, hogy erőforrásaikat a rekordok feldolgozására koncentrálják. A felhasználónak csupán a file leírását kell megadnia, és a megfelelő input-output utasítást kiadnia. Alább részletesen ismertetjük a fenti háromféle file szervezését és a javasolt eljárásokat. A file-szervezés célja mindig az, hogy a központi egység számára minimális időn belül elérhetővé tegye a feldolgozandó rekordokat. Mint látni fogjuk, ez csak részben oldható meg, a file igazán hatékony feldolgozási módját sokszor csak próbálgatással, illetve helyes becslés segítségével tudjuk meghatározni.

Megjegyzés. Egy rekord elérése alatt értjük a háttértárolóról leolvasandó rekord helyének (címének) megállapítását és az illető rekord megkeresését, illetve a háttértárolóra irandó rekord számára megfelelő üres hely (háttértároló terület) keresését. A rekordelérési módszerek (access methods) tehát egyaránt vonatkoznak input és output műveletekre.

10.2 A SZEKVENCIAÁLIS FILE FELÉPÍTÉSE

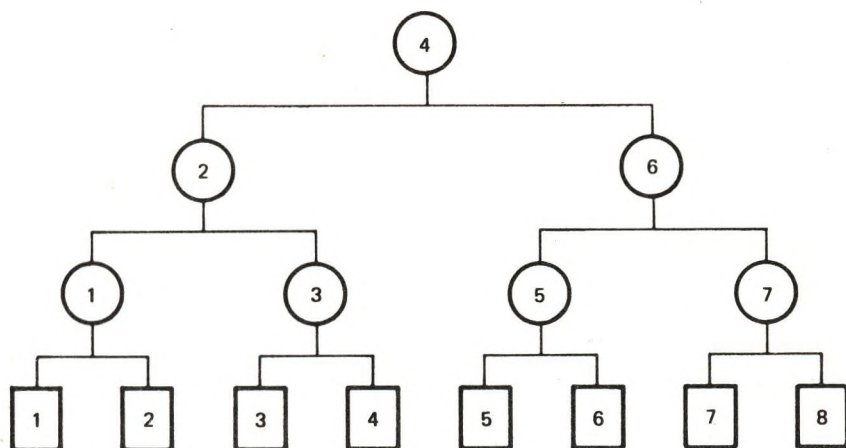
A szekvenciális file-ban a rekordok egymás után helyezkednek el, sávról sávra, cilinderről cylinderre, növekvő cimeken. Rendszerint kulcssorrendben vannak. A rekordok lehetnek fix vagy változó hosszúságúak, blokkoltak, blokkolatlanok vagy definiálatlanok.

Kulcsok használata nem kötelező. Ha a file-t mindig szekvenciálisan dolgozzuk fel, nincs értelme kulcsokat használni. Ha valamilyen okból jelentős mértékű random feldolgozás lehetséges, a kulcsok használata gyorsítja a rekordok elérését. A szekvenciális feldolgozás egy cylinderenkénti keresési műveletből és rekordonként egy olvasási műveletből áll.

A nem szekvenciális feldolgozás elvileg rossz hatásfoku. Ha ritkán fordul elő, nem jelentős. Az egyik lehetséges programozási megoldás az, hogy a re-

kordokat sorban leolvassuk, míg a keresett rekordra nem bukkanunk. Így átlagosan a file felét le kell olvasnunk, bár nem dolgozzuk fel (ugyanis több rekord feldolgozása - ha azok nem egymás után következnek - több lemezfordulatot igényel).

Hatékonyabb keresési módszer az ún. bináris keresés. Ennek során több lépésben meghatározzuk, hogy melyik file-területen (cilinderen) érdemes keresnünk a rekordot. Ezután már csak ezt az egy cilindert kell átvizsgálnunk. Így mintegy behatároljuk a keresett rekordot. A bináris keresést a 45. ábra szemlélteti. Feltételezzük, hogy a file 8 cilindert foglal el, és a rekordok kulccsal rendelkeznek.



45. ábra. Bináris keresés

Először a 4. cylinder utolsó (legnagyobb) kulcsát hasonlítjuk össze a keresett kulccsal. Ha a keresett kulcs ennél kisebb, a 2. cylinderre ugunk, ahol a keresett kulcsot most ennek legnagyobb kulcsához hasonlítjuk. Ellenkező esetben a 6. cylinderen folytatjuk a keresést. Az átvizsgálandó cylinderek száma minden egyes összehasonlítás során felére csökken, míg végül eljutunk ahhoz a cylinderhez, amelyen a keresett rekord megtalálható. File-bővítések és rekordtörlések csak a file átszervezése útján lehetségesek. A szekvenciális file-t elsősorban táblázatok és közbenső tárolás céljára használják.

10.3 AZ INDEXELT SZEKVENCIÁLIS FILE FELÉPÍTÉSE

Az indexelt szekvenciális file-nak három fő területe van:

- az elsődleges adatterület (prime data area),
- az indexterület,
- a túlcsordulási terület (overflow area).

Az elsődleges adatterületre írjuk az egyes rekordokat, amikor a file-t létrehozuk vagy újjászervezzük. Itt a rekordok növekvő kulcs szerinti sorrendben helyezkednek el. A rekordok lehetnek blokkoltak vagy blokkolatlanok. Blokkolt rekordok esetén a blokk kulcsmezője a blokkban előforduló legnagyobb kulcsot tartalmazza.

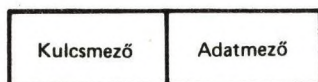
10.3.1 INDEXEK

a) Sávindex (track index)

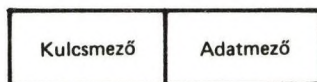
A legalacsonyabb szintű index. A sávindex tételei közvetlenül az egyes rekordokra mutatnak.

Cilinderenként egy sávindex használatos, és az mindig az indexelt cylinder első sávjain található.

Minden egyes sáv indexelésére két sávindextétel használatos: un. normál és túlcsordulási indextétel. Ezek kiindulási tartalma a file létrehozásakor rögzítődik (lásd a 46. ábrát).



Normál tétel



Túlcsordulási tétel

46. ábra. Sávindextételek

A normál indextétel kulcsmezője az indexelt sávon található legmagasabb kulcsot tartalmazza (mivel a file alapvetően szekvenciális, ez a sáv utolsó rekordjának kulcsa). Adatmezője az indexelt sáv fizikai címére mutat. A tulcsordulási indextétel tartalma attól függ, vannak-e az adott sávhoz tartozó tulcsordult rekordok. Ennek alapján

- ha a szóban forgó sávhoz nem tartoznak tulcsordult rekordok, a tulcsordulási indextétel tartalma megegyezik a normál indextétel tartalmával,
- ha az adott sávhoz tulcsordult rekordok tartoznak (amelyek a tulcsordulási területen vannak), akkor a tulcsordulási indextétel kulcsmezője a legmagasabb kulcsu tulcsordult rekord kulcsát tartalmazza, adatmezője pedig a legkisebb kulcsu tulcsordult rekord fizikai címére mutat. Speciális esetben - egyetlen tulcsordult rekord esetén - a tulcsordulási indextétel kulcsmezője és adatmezője egyaránt erre a rekordra vonatkozik. Általános esetben a tulcsordulási indextétel kulcsmezője és adatmezője mintegy közrefogja a sávhoz tartozó tulcsordult rekordokat, és ezzel megkönnyíti a rekordok elérését.

A sávindex utolsó tétele un. üres rekord (dummy entry), amely a sávindex végét jelzi. A sávindex sávjára csak ezután írhatók felhasználói rekordok, ha ott még van hely a számukra.

Valamennyi indextétel formátuma azonos, az egyes indextételek fix hosszúságú, blokkolatlan rekordok. A kulcsmező hosszát a felhasználó specifikálja. Kivétel ez alól az üres rekord, amelynek kulcsmezője csupa 1, adatmezője csupa 0. Az ábrákon az indextételek azonosító mezőjét (count) az egyszerűség kedvéért nem tüntettük fel.

b) Cylinderindex (cylinder index)

A sávindexnél eggyel magasabb szintű index. File-onként egy cylinderindex használatos, és az a közvetlen elérésű tároló valamely erre a célra kijelölt helyén található. Megengedett a cylinderindex elhelyezése az adatterületeket tartalmazó mágneslemezcsomagtól különböző lemezcsomagon is. Ennek azonban az a feltétele, hogy feldolgozáskor a cylinderindexet tartalmazó lemez "on-line" legyen, tehát elérhető legyen valamelyik mágneslemezegységen.

A cylinderindex cilinderenként egy-egy tételt tartalmaz; az utolsóétel itt is üres rekord, és az index végét jelzi. Az egyes tételek hasonlóak a sáv-index normál tételeihez: kulcsmezők az adott cilinderen levő legnagyobb kulcs értékét tartalmazzák, adatmezők pedig az indexelt cilinder sávindexének fizikai címére mutatnak. Például a 00610 0000 cylinderindex-étel értelmezése:

a 00 számú cilinderen levő legnagyobb kulcs = 00610,

a 00 számú cilinder sávindexének címe = 00.

Ha a file nem foglalja el az összes cilindert, akkor az utolsó cylinder-index-tételek inaktív, és felépítésük megegyezik az üres rekord szerkezetével:

kulcsmező = csupa 1,

adatmező = csupa 0.

Az inaktív indextételek lehetővé teszik az eddigieknél magasabb kulcsu rekord hozzáadását a file-höz, vagyis a file kibővítését.

c) Főindex (master index)

Ez a legmagasabb szintű index, és opcionális - nem kötelező. Akkor használjuk, ha a cylinderindex olyan hosszú, hogy végigvizsgálása túlságosan sok időt igényelne. A főindex felállítása akkor javasolt, ha a cylinderindex több mint négy sávot foglal el.

A főindex a cylinderindex sávonkénti eloszlására mutat. Kulcsmezője az adott cylinderindexsávon található legmagasabb kulcsot tartalmazza (például ha az első 50 cilinder cylinderindexe foglal el egy sávot, akkor a főindex első tételének kulcsmezője az 50-edik cilinder legnagyobb kulcsára mutat).

A főindex kötelezően mindig a cylinderindex előtt van, tehát a cylinderindex közvetlenül a főindex utáni sávon kezdődik, vagyis a két index együttesen összefüggő indexterületet alkot. Ez a terület lehet külön lemezcsomag is, ha feldolgozáskor az adott lemezcsomag on-line kapcsolatban van. Ez jelentősen gyorsítja a feldolgozást.

Valamely rekord keresése mindezek alapján a következőképpen megy végbe:

- A főindex átvizsgálása; ebből kiderül, hogy a sávindeksnek melyik sávján érdekes a keresést folytatni. Ha nem használunk főindexet, ez a lépés értelesszerűen elmarad.
- A cilinderindex megfelelő sávjának átvizsgálása; ebből megtudjuk, melyik cilinderen van a keresett rekord. Ha nincs főindex, akkor a cilinderindex átvizsgálása előlről kezdődik.
- A megfelelő cilinder sávindeksének átvizsgálása; ennek eredményeképpen rendelkezésre áll az a sáv, amelyen a keresett rekord található, illetve - ha a keresés tulcsordult rekordra vonatkozik - a sávhoz tartozó legkisebb kulcsu tulcsordult rekord fizikai címe (lásd a sávindeks tulcsordulási tételének adatmezőjét). Ez esetben a keresés a tulcsordulási területen folytatódik, ahol a tulcsordult rekordok közötti eligazodást speciális (un. láncoló) mezők segítik. Erről részletesen a tulcsordulási területeknél lesz szó.

Megjegyzés. Egyes operációs rendszerek (például SIEMENS) magasabb szintű főindexek használatát is megengedik. Általában egy magasabb szintű index bevezetése mindig akkor célszerű, ha a nála eggyel alacsonyabb szintű index túl hosszú, több sávot foglal el, és végigvizsgálása időigényes.

10.3.2 TÚLCSORDULÁSI TERÜLETEK

Kétféle tulcsordulási terület létezik:

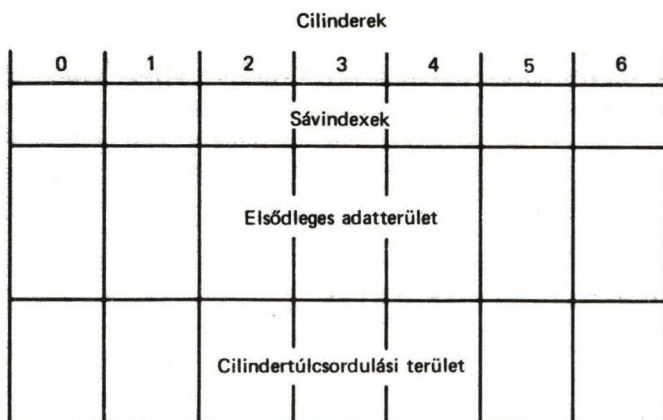
- cilindertulcsordulási terület (cylinder overflow area),
- független tulcsordulási terület (independent overflow area).

A felhasználó választhat, hogy csak az egyiket vagy mindkettőt specifikálja. A tulcsordulási területek fogadják be a file-ba beszurandó rekordokat, ha azok nem férnek el az elsődleges adatterületen. Ugyancsak a tulcsordulási területre kerülhetnek egyes rekordok a file bővítésekor (lásd 192. old.).

a) Cilindertulcsordulási terület

Minden egyes cilinderen kijelölhető azonos számú sáv, amelyet tulcsordult rekordok számára tartalékolunk. Ez esetben az RØ rekord un. cilindertulcsordulást vezérlő rekord (Cylinder Overflow Control Record – COCR), amelyből az operációs rendszer merít információt.

A cilindertulcsordulási terület használatának megvan a maga előnye és hátránya. Kedvező, hogy az adott cilinderről tulcsordult rekord megkereséséhez nincs szükség újabb pozicionálási műveletre (író-olvasó fej mozgatására), hiszen minden egyes cylinder maga tartalmazza saját tulcsordult rekordjait. A hátrány nagyobb méretű file-ok esetén mutatkozik: Ha a tulcsordult rekordok cilinderenkénti eloszlása nem egyenletes, sok üres helyet kell tartalékolnunk a tárolón. A cilindertulcsordulási terület elhelyezkedése a 47. ábrán látható.

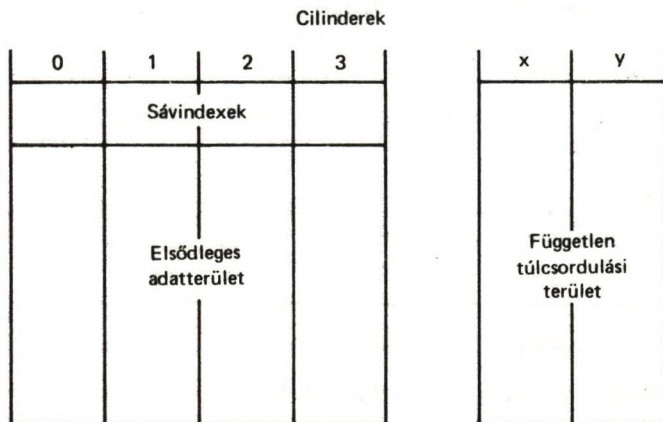


47. ábra. Cilindertulcsordulási terület elhelyezkedése

b) Független tulcsordulási terület

A felhasználó kijelölhet még un. független tulcsordulási területet is, amely a file bármely helyéről származó tulcsordult rekordok befogadására szolgál. A független tulcsordulási terület teljes cilinderekből áll. A módszer előnye,

hogy kevesebb helyet kell tartalékolni a tulcsordult rekordok számára, és a tulcsordulás cylinderenkénti egyenetlen eloszlása nem befolyásolja a tároló kapacitásának kihasználtságát. Hátránya viszont, hogy a tulcsordult rekord megkeresése pozicionálási műveletet igényel, hiszen az író-olvasó fejrendszert át kell helyezni a tulcsordulási területre. Ez elkerülhető, ha a független tulcsordulási területet külön lemezcsoomagon helyezzük el. A független tulcsordulási terület elhelyezkedése a 48. ábrán látható.

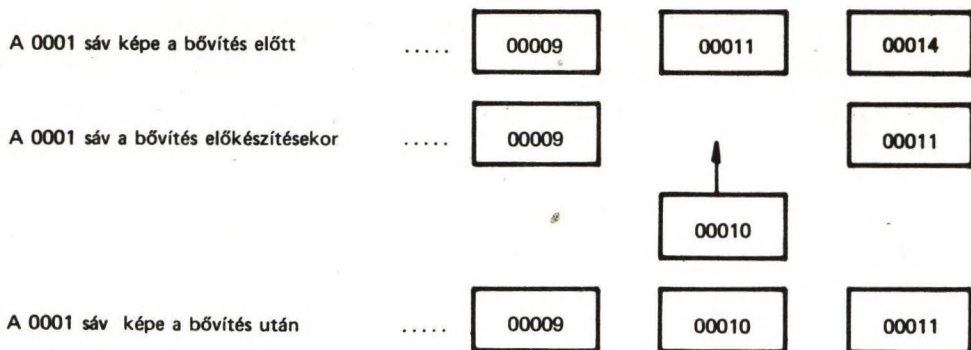


48. ábra. Független tulcsordulási terület elhelyezkedése

A probléma javasolt megközelítése a következő: jelöljünk ki akkora cylindertulcsordulási területet, hogy az elég nagy legyen az átlagos mennyiségű tulcsordult rekordok tárolására. Az ugyancsak kijelölt független tulcsordulási területet arra használjuk, hogy csak a cylindertulcsordulási területről is tulcsordult rekordokat tárolja. Így várhatóan kevesebb lesz a hozzáfordulás a független tulcsordulási területhez, és gyorsabb lesz az átlagos adateltérés.

10.3.3 FILE-BŐVÍTÉS

Tekintsük a 49. ábrát! Ez egy adott file-nak a bővítését mutatja a 00010 kulcsu rekorddal. Kérdés, hogy hova kerüljön ez a rekord.



49. ábra. File-bővítés rekordbeszurással

A 0001 című sávon foglalnak helyet azok a rekordok, amelyek kulcsa 00001-től 00014-ig változik. A 00010-es kulcsnak tehát - növekvő kulcs-sorrendet feltételezve - a 0001-es sávra kell kerülnie. Mivel azonban ez a sáv tele van,

a sáv utolsó rekordját áthelyezzük valamelyik tulcsordulási területre, és a beszurandó rekordot a neki megfelelő helyre tesszük, növekvő kulcssorrend mellett.

Esetünkben ez azt jelenti, hogy a 00009 és 00011 kulcs között valahogyan helyet kell találnunk a 00010 kulcs számára. Tehát a 00011 kulcsu rekordot egy rekordhellyel feljebb toljuk, és ennek korábbi helyére tesszük a 00010 kulcsu beszurandó rekordot.

A példából látható, hogy a 00010 kulcsu rekord kiszorította a korábbi helyéről a 00014 kulcsu rekordot, így az a tulcsordulási területre tolódott. Ha most további rekordok beszurásával a 00011 kulcsu rekord is a tulcsordulási területre kerülne, ott a 00014 kulcsu rekord után következne.

A tulcsordulási területen a rekordok nem szükségképpen növekvő kulcs szerinti sorrendben helyezkednek el, hanem érkezési sorrendben.

Emiatt meg kellett valósítani a tulcsordult rekordok közötti logikai folytonosságot, mégpedig növekvő kulcs szerinti sorrendben. Az ellentmondást a rekordok ún. láncoló mezője (link field) hidalja át; minden egyes tulcsordult rekord láncoló mezője a sorrendben következő nagyobb kulcsu tulcsordult rekordra mutat. Kivétel ez alól az utolsó tulcsordult rekord, amelynek láncoló mezőjében FF₍₁₆₎ van. Ez jelzi, hogy az illető rekord a lánc utolsó tagja, tehát a tulcsordulási terület nem tartalmaz az adott sávhoz tartozó további rekordokat. Példánkban a 00014 kulcsu rekord az egyetlen tulcsordult rekord, ám ha például a 00013 kulcsu rekordot akarnánk beszurni, az a 00014 kulcsu rekord után következne a tulcsordulási területen, és láncoló mezője a 00014 kulcsra mutatna.

Megjegyzés. Bár a cilindertulcsordulási terület a cylinder összes sávjáról fogadja a tulcsordult rekordokat, a független tulcsordulási terület pedig a file összes cilinderének tulcsordult rekordjait tárolja, minden egyes sávnak saját tulcsordulási láncja van.

Az indexelt szekvenciális file indexrendszerét egyértelműen megvilágítja az 50. ábra, amely egyttal a különböző szintű indexek viszonyát is mutatja. A példában feltételeztük, hogy a sávindex sávján már elkezdődik az adatrekordok rögzítése. Az elsődleges adatterület a 7. sávon fejeződik be, a 8. és 9. sáv valamennyi cilinderen cilindertulcsordulási területet alkot. A file felépítése eredetileg az alábbi volt:

- a 0. sávra kerültek a sávindexen kívül az adatrekordok a 76-os kulcsu rekorddal bezárólag,
- az 1. sáv tartalmazta a 100-152-es kulcsu rekordokat,
- a 7. sávon voltak a 900-980-as kulcsu rekordok.

A 130-as, 140-es és 152-es kulcsu rekordok beszúrások következtében a tulcsordulási területre kerültek, a 135-ös kulcsu rekord egyenesen oda került. A 7. sávról - a 973-as kulcsu rekord beszúrása következtében - a 980-as kulcsu rekord a tulcsordulási területre tolódott, és ott mint a 7. sáv egyetlen tulcsordult rekordja szerepel. A tulcsordult rekordokat a rendszer megfelelő láncoló mezőkkel (LM) látta el, a lánc utolsó tagjában a * mutatja a

Sáv Sávinde

0	COCR	76	0. sáv	76	0. sáv	122	1. sáv	152	8. sáv 3. rec.	238	2. sáv	238	2. sáv	973	7. sáv	980	8. sáv 4. rec.	11...1	00...0	76	Adat
---	------	----	--------	----	--------	-----	--------	-----	-------------------	-----	--------	-----	--------	-----	--------	-----	-------------------	--------	--------	----	------

1	100	Adat	103	Adat														117	Adat	122	Adat
---	-----	------	-----	------	--	--	--	--	--	--	--	--	--	--	--	--	--	-----	------	-----	------

2	200	Adat	211	Adat														232	Adat	238	Adat
---	-----	------	-----	------	--	--	--	--	--	--	--	--	--	--	--	--	--	-----	------	-----	------

...

7	900	Adat	920	Adat														955	Adat	973	Adat
---	-----	------	-----	------	--	--	--	--	--	--	--	--	--	--	--	--	--	-----	------	-----	------

Cilindertűcsordulási terület

8	152	LM *	Adat	140	152	Adat	130	135	LM	Adat	980	LM *	Adat	135	140	LM	Adat				
---	-----	---------	------	-----	-----	------	-----	-----	----	------	-----	---------	------	-----	-----	----	------	--	--	--	--

9

Főindex

X	4730	x+1. sáv	8560	x+2. sáv														85610	x+20 sáv	11...1	00...0
---	------	----------	------	----------	--	--	--	--	--	--	--	--	--	--	--	--	--	-------	----------	--------	--------

Cilinderindex

X+1	980	1. cilinder 0. sáv	1850	2. cilinder 0. sáv																4730	10. cilinder 0. sáv
-----	-----	-----------------------	------	-----------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	------	------------------------

X+2

	4800	11. cilinder 0. sáv	4900	12. cilinder 0. sáv																8560	20. cilinder 0. sáv
--	------	------------------------	------	------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	------	------------------------

...

X+20	60511	150. cilinder 0. sáv	71711	151. cilinder 0. sáv														85610	160. cilinder 0. sáv	11...1	00...0
------	-------	-------------------------	-------	-------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	-------	-------------------------	--------	--------

50. ábra. Indexelt szekvenciális file felépítése

sávhoz tartozó tulcsordulási lánc végét. Végeredményben az egyes tulcsordulási láncok az elsődleges adatterület sávjai meghosszabbításának tekinthetők: az adatsávok látszólagos hossza így nagyobb, mint a tényleges fizikai hossz.

10.3.4 A FILE LEMEZTERÜLET-IGÉNYÉNEK BECSLÉSE

Mint hogy a felhasználó ismeri adatainak formátumát (kulcsmező, adatmező hossza stb.), kiszámíthatja, hány rekordot vagy blokkot tud elhelyezni az egyes file-területeken. A rekordok számából ezután következtetni lehet a file helyfoglalására. Alább néhány kapacitásszámítási képletet adunk meg. A képletekben

n - a sávok száma cilinderenként az elsődleges adatterületen,

KL - a kulcsmező hossza byte-okban kifejezve (key length),

DL - az adatmező hossza byte-okban kifejezve (data length).

Sávindex hossza (a kiszámított érték szükség esetén felfelé kerekítendő)

$$S = (2n + 1) (91.49 + 1.049 KL) \text{ byte.}$$

Adatrekordok száma a 0. sávon (az elsődleges adatterületen)

$$A = \left[1 + \frac{3605 - (KL + DL + 1)}{81 + 1.049 (KL + DL)} \right]$$

Adatrekordok száma a többi sávon

$$B = \left[1 + \frac{3605 - (KL + DL)}{81 + 1.049 (KL + DL)} \right]$$

Blokkok száma cilinderenként (az elsődleges adatterületen)

$$K = A + (n-1) B,$$

ahol blokkolt rekordok esetén $DL = \text{blokkolási faktor} \cdot \text{rekordhossz}$

blokkolatlan rekordok esetén $DL = \text{rekordhossz (byte)}$.

Tulcsordult rekordok száma sávonként

$T = B$, ahol B kiszámításában $DL = \text{rekordhossz} + 10 \text{ (byte)}$.

Cilinderindex-(főindex) tételek száma sávonként

$I = B$, ahol B kiszámításában $DL = 10 \text{ (byte)}$.

Megjegyzés. A képletek a maximálisan elhelyezhető rekord-, illetve blokk-számot adják meg.

10.4 A DIREKT FILE FELÉPÍTÉSE

A direkt file esetében az egyes rekordkulcsok és rekordcímek között szoros kapcsolat van. Ez lehetővé teszi az egyes rekordok gyors elérését, jöllehet azok esetleg nem szekvenciálisan helyezkednek el a file-on belül. Szekvenciális feldolgozás esetén célszerű előzetesen a rekordok rendezését elvégezni.

10.4.1 CÍMZÉS

Az operációs rendszer általában megengedi relatív sávcímek használatát, és azokat saját maga konvertálja abszolút címekre (lásd alább, illetve az 1. és 2. példában).

a) Közvetlen címzési mód

Minden lehetséges kulcsot egy-egy külön címre konvertálunk. Ennélfogva bármely rekord megkeresésére elegendő egy kereső- és egy olvasóutasítás. Ügyeljünk arra, hogy minden egyes rekord számára helyet tartalékoljunk, még akkor is, ha az illető rekord helye üres.

A címgenerálásnak két lehetséges módja van:

- címként használt kulcs,
- keresztreferencia táblázat használata.

Ahhoz, hogy valamely kulcsot közvetlenül egy rekord címként használhassunk, szükséges, hogy a rekordok fix hosszúságúak, a kulcsok pedig numerikusak legyenek (csak számjegyeket tartalmazzanak). A címképzés az alábbi összefüggés alapján megy végbe:

$$\frac{\text{Kulcs}}{\text{Sávonkénti rekordok száma}} \Rightarrow Q + R,$$

ahol Q - a hányados (relatív sávcím)

R - a maradék ($R + 1$ = rekord sorszáma az adott sávon).

Keresztreferencia táblázat használata esetén a file minden egyes rekord-jához egy címet rendelünk, és ezt egy ún. keresztreferencia táblázatban tároljuk. A táblázat ki is nyomtatható. Bár az egyes rekordok címük alapján gyorsan elérhetők, bizonyos idő kell a cím kikeresésére a keresztreferencia táblázatban. (Az indexelt szekvenciális file-szervezés ennek a módszernek egy változata.)

b) Közvetett címzési mód

Közvetett címzést általában akkor használunk, ha a teljes kulcstartományban igen nagy százalékban fordulnak elő kihasználatlan kulcsok, és így a közvetlen címzés nem elég hatékony. Például vállalati dolgozók adatait tároljuk kulccsal ellátva; a teljes kulcstartomány 0001-től 9999-ig terjed, azonban a cég csak 3000 dolgozót számlál, s így a kulcsok nagy része kihasználatlan.

Kézenfekvő megoldás kínálkozik a fenti példa alapján: képezzük le a teljes kulcstartományt (0001-9999) a ténylegesen kihasznált kulcstartományra (pl. 0001-3000). Ezáltal a teljes intervallumot mintegy összenyomtuk, összeszűkítettük egy kisebb intervallummá. A leképzés eszköze a címkonverzió. Ezen azt értjük, hogy minden egyes rekord kulcsának megfeleltetünk egy címet a file-ban. A megfeleltetés szabályait, algoritmusát a felhasználó definiálja:

vagy valamely kipróbált, bevált módszert, vagy akár saját maga készítette algoritmust alkalmazva. A cimkonverziós eljárások elkerülhetetlen velejárója, hogy több kulcsot konvertálnak ugyanarra a címre. A megegyező című rekordok az alábbiak:

- saját rekord (home record) - az első, erre a címre kerülő rekord,
- tulcsordult rekord (overflow record) - a többi, erre a címre kerülő rekord.

A saját rekord és a tulcsordult rekordok együttesen szinonimákat alkotnak. Mivel egy adott címre csak egyetlenegy rekord kerülhet, a tulcsordult rekordok címét meg kell változtatni.

Megjegyzések. Valamely cimkonverziós eljárás jóságára jellemző, hogy milyen százalékban generál tulcsordult rekordokat, illetve szinonimákat. A megfelelő cimkonverziós eljárás kiválasztásakor két feltételt kell szem előtt tartanunk:

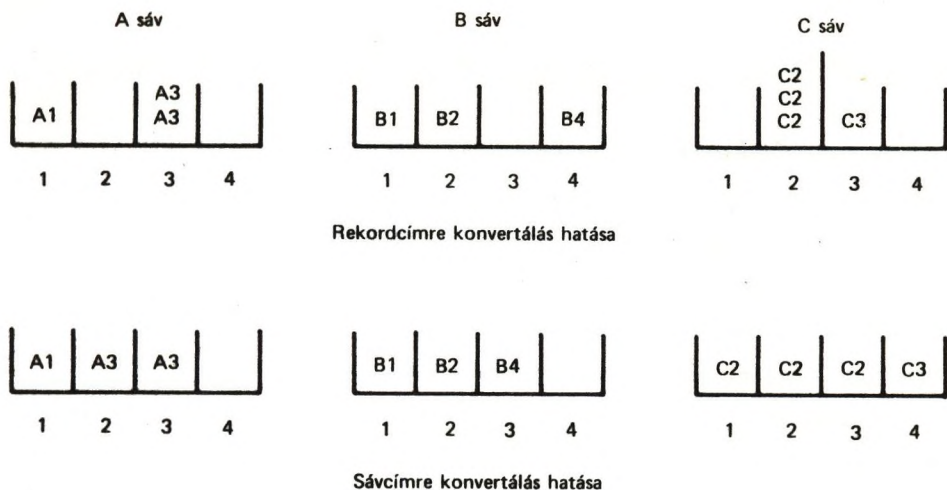
- a file-ban előforduló minden egyes rekord kulcsa a kijelölt címtartományba konvertálódjon;
- a címtartományban a címek eloszlása egyenletes legyen, vagyis a lehető legkevesebb szinonima keletkezzen.

A szinonimák számának minimalizálására egy lehetséges mód az, hogy a file számára tényleges terjedelménél nagyobb tárterületet jelölünk ki. Hogy valamely file milyen arányban - hány százalékban - tölti ki a számára fenntartott tárterületet, arra az ún. kitöltési tényező jellemző:

$$\text{Kitöltési tényező} = \frac{\text{a file felhasznált rekordhelyeinek száma}}{\text{a file összes rekordhelyeinek száma}}$$

Javasolt kiindulási érték a 80-85 %. Például ha egy 8200 rekordból álló file számára 10 000 rekordhelyet tartalékolunk, akkor a kitöltési tényező értéke 82 %.

A tulcsordult rekordok számának minimalizálására egy lehetséges mód az, hogy az egyes rekordkulcsokat nem rekordcímre, hanem sávcímre konvertáljuk. Az 51. ábra mutatja a módszer előnyeit.



51. ábra. A címkonverzió hatása

Rekordcímre konvertálás esetén a rekordok kb. 50 %-a szinonima és kb. 30 %-a tulcsordult rekord. Sávcímre konvertáláskor a rekordok 100 %-a szinonima, de nincs egyetlen tulcsordult rekord sem.

Jól jegyezzük meg, hogy a szinonimák ugyanarra a fajta címre való konvertáláskor keletkeznek, tehát például az A sávra konvertált rekordok valamennyien szinonimák, ha sávcímre való konvertálást tételezünk fel.

Egyébként mind a rekordcímre, mind pedig a sávcímre konvertált rekordok elérési ideje kb. azonos, a sávcímre konvertálás előnye a tulcsordult rekordok kiküszöbölésében áll. Az ESZR/DOS-ban az írási-olvasási műveletek mindig sávcímre vonatkoznak.

10.4.2 CÍMKONVERZIÓS ELJÁRÁSOK

Számos címkonverziós eljárás ismeretes. A legmegfelelőbb kiválasztása többnyire próbálgatás útján történik. Jó kiindulási feltételnek számít az, ha az összes rekordnak nem több, mint 20 %-a szinonima. Alább vázlatosan ismertetünk néhány címkonverziós eljárást.

a) Hányados-maradék módszer

Először mindig ezzel a módszerrel célszerű próbálkozni, mert igen egyszerű, és gyakran ad kielégítő eredményt. Lényege abban áll, hogy a rekordkulcsot elosztjuk egy olyan törzsszámmal, amelynek értéke közel esik a file számára fenntartott rekordhelyek számának értékéhez (például ha 10000 rekordhelyet tartálékoltunk, a 10000-hez közel eső törzsszám a 9973). A maradék fogja jelenteni a rekord-, illetve sávcimet.

Az eljárás különbözik rekordcímre, illetve sávcímre konvertálás esetén. Az alábbi példák szemléltetik a különbségeket:

1. példa

Tárolandó 8000 db 200 byte hosszúságú rekord a mágneslemezen; a konvertálást rekordcímre végezzük.

- Legyen a kitöltési tényező 80 %, így 10000 rekordhelyre van szükség.
- Egy 10000-hez közel eső törzsszám: 9973.
- Osszuk el a kulcsot 9973-mal!
- Osszuk el a maradékot a sávonkénti rekordok számával (feltesszük, hogy sávonként 13 rekord helyezhető el)!
- A hányados megadja a relatív sávcimet, a maradék eggyel növelt értéke pedig a rekord sorszámát az adott sávon.

2. példa

Tárolandó 8000 db 200 byte hosszúságú rekord a mágneslemezen; a konvertálást sávcímre végezzük.

- Legyen a kitöltési tényező 80 %, így 10000 rekordhelyre van szükség.
- Feltesszük, hogy sávonként 13 rekord helyezhető el, tehát összesen 770 sávra van szükség.
- Egy 770-hez közel eső törzsszám: 769.
- Osszuk el a rekordkulcsot 769-cel!
- A maradék megadja a relatív sávcimet (000-768).

Megjegyzés. Azért hivatkozunk a fenti példákban relativ sávcimre, mert ez a cím a file kezdősávjának sávciméhez képest értendő. A relativ sávcim alapján az operációs rendszer kiszámítja az abszolút sávcimet, így megtalálható a keresett rekord.

A hányados-maradék módszer magától kielégíti azt a feltételt, hogy valamennyi kulcsot a kijelölt címtartományba konvertálja. Azt, hogy relative mennyi szinonimát generál, csak próbálgatással lehet megbízhatóan eldönteni.

b) Összekombinálás (folding)

A kulcsot két vagy több részre osztjuk, és a részeket azután összeadjuk. Az összeg - vagy annak egy része - alkotja a relativ címet.

Példa 746298 kulcsra (numerikus kulcs):

- $746 + 298 = 1044$ (felezés),
- $74 + 62 + 98 = 234$ (harmadolás),
- $769 + 428 = 1197$ (váltakozó - páratlan és páros sorszámú - számjegyek).

c) Radixtranszformáció

A kulcsot valamely más számrendszerbeli számmá transzformáljuk. A tulcsorduló számjegyeket figyelmen kívül hagyjuk (például ha a cím négy számjegyű, az ötödik jegyet levágjuk).

Példa $42356_{(11)}$ konvertálására tizes számrendszerben:

$$\begin{aligned} & 4 \cdot 11^4 + 2 \cdot 11^3 + 3 \cdot 11^2 + 5 \cdot 11^1 + 6 \cdot 11^0 = \\ & = 58564 + 2662 + 363 + 55 + 6 = 61650_{(10)}. \end{aligned}$$

Legyen ezek után a relativ sávcim 1650!

A kiválasztott címkonverziós eljárást felhasználása előtt meg kell vizsgálni abból a szempontból is, hogy a keletkezett szinonimák hogyan befolyá-

solják a rekord elérési idejét. Nagyobb szinonimaarány esetén ugyanis több olvasási művelet szükséges az éppen keresett rekord megtalálásához. Jó értékeknek tekintjük az 1,2 átlagos rekord olvasási gyakoriságot; az ennél magasabb értékeket igyekszünk csökkenteni.

10.4.3 A FILE LÉTREHOZÁSA

A file egyes feldolgozásai alkalmával rekordokhoz vagy blokkokhoz szeretnénk hozzáférni a lehető leggyorsabban. A rekord elérési ideje nagymértékben függ attól, hogyan, milyen formában hoztuk létre a file-t annak idején. Alább röviden ismertetünk két javasolt módszert file-ok létrehozására. Ezek:

- a láncolási módszer,
- a progresszív tulcsordulási módszer.

Mindkét módszer feltételezi, hogy a file kielégíti az alábbi követelményeket:

- a rekordok blokkolatlanok,
- a rekordokat kulccsal láttuk el,
- a címkonverzió sávcímre vonatkozik,
- a tulcsordult rekordokat az elsődleges (és egyetlen) adatterület üres rekordhelyeire tesszük.

a) Láncolási módszer

Minden egyes sáv valamely rekordja speciális, ún. láncolási rekord. Ez teremti kapcsolatot a normál és a tulcsordult rekordok között. A tulcsordult rekordokat a következő nagyobb című sávra írjuk. Az 52. ábra egy négy-sávos elrendezést és rekordkeresést szemléltet.

A rekordok betöltési sorrendje: A1, B1, A2, D1, C1, A3, C2, A4 (tulcsordulás), B2, C3, B3 (tulcsordult), A5 (tulcsordult). Ha valamelyik rekordot keressük, a keresés természetesen a saját sávján kezdődik (például A rekord esetén az A sávon). Ha ott nem találjuk, a keresést a láncolási rekordban megadott sávon folytatjuk - esetünkben ez a B sáv lesz. Ha a B sávon sem találjuk a

Sáv	Láncolási rekord	Adatrekordok			
A	B	A1	A2	A3	
B	D	B1	A4	B2	
C		C1	C2	C3	
D		D1	B3	A5	

52. ábra. A láncolási módszer vázlata

rekordot, tovább megyünk a D sávra stb. Ha azonban valamelyik C rekord nem találjuk a C sávon, a keresést nem folytatjuk, mivel a C sáv láncolási rekordja üres, és azt jelzi, hogy nincs több C rekord a file-ban.

b) Progresszív tölcsördulási módszer

A tölcsördult rekordokat itt is a következő magasabb című sávra írjuk. A láncolási módszertől annyiban térünk el, hogy nem létesítünk láncolási kapcsolatot a rekordok saját sávjai és a tölcsördulási sávok között; a rekordok összetartozását egyedül az egymás után következő sávok jelentik.

Mint hogy a direkt file esetében a rekordkulcs és rekordcím között szoros, pontosan meghatározott kapcsolat van, a direkt file általában perifériafüggetlen; a file-t tároló berendezés címzési módjához, annak különböző korlátozásaihoz alkalmazkodik.

INTRODUCTION TO THE ARCHITECTURE OF LARGER UCS MODELS

The architecture of the larger UCS models, from the programmer's point of view is discussed in detail by this lecture note. In addition, it gives some basic knowledges in computer engineering too (numerical systems, machine representation of numbers and other fundamentals). In case of the larger models of the UCS family, the representation of the hardware architecture is taking place on functional level; that is, the reader can get acquainted with the computer to an extent necessary to programming in machine code. Finally, it gives a roughly outlined review of flowcharting, of the operating system and of certain processes of file organization. The aim of the tables in the text is to promote the use of this lecture notes as a manual.

We send this textbook off in the hope that it would efficiently be used both by unexperienced and skilled programmers.

In parallel with the spreading of computers the sphere of the domestic users of UCS machines keeps growing. This publication is intended to them and is equally useable as a curriculum text book and as a tool of individual study.

The notebook discusses in detail the fundamentals of computing essential to programmers, and the architecture of hardware of larger models of the UCS family.

ХЕРБОЙ, Миклош:

ВВЕДЕНИЕ В АРХИТЕКТУРУ БОЛЬШИХ МОДЕЛЕЙ ЕС ЭВМ

Учебник подробно описывает архитектуру больших моделей ЕС ЭВМ с аспекта программирования. Одновременно с этим он дает определенные основные знания в области вычислительной техники /системы представления чисел, машинное изображение чисел, прочие основные понятия/. Относительно больших моделей ЕС ЭВМ архитектура технических средств описывается на функциональном уровне; таким образом, читатель может познакомиться с электронно-вычислительной машиной в глубине необходимой для программирования в машинном коде.

Наконец, учебник дает краткий обзор о составлении блоксхем, об операционных системах, а также о некоторых способах организации файлов. Таблицы, находящиеся в материале учебника способствуют использованию учебника в качестве справочника.

При издании настоящего учебника мы старались оказывать помощь к работе как начинающих, так и опытных программистов.

Параллельно распространению вычислительных машин расширяется и круг отечественных пользователей средств ЕС ЭВМ. Данный учебник составлен для них и он может быть использован в качестве учебного пособия на курсах, а также как вспомогательное средство индивидуальных занятий.

Учебник подробно занимается основными знаниями по вычислительной технике необходимыми для программистов, а также технической архитектурой больших моделей ЕС ЭВМ.

57379



NEMZETKÖZI SZÁMÍTÁSTECHNIKAI OKTATÓ ÉS TÁJÉKOZTATÓ KÖZPONT**LEGÚJABB KIADVÁNYAI**

<i>Dr. Mérey András:</i> ADATSZERKEZETEK	80,— Ft
<i>Széplaki Ágnes:</i> PROGRAMHIBA-KERESÉS, TESZTELÉS, MINŐSGVIZSGÁLAT	60,— Ft
<i>Rugyai Tamás:</i> KERESKEDELMI RENDSZERSZERVEZÉSI ESETTANULMÁNY	70,— Ft
<i>Rammacher Tamás:</i> DOS KOMPONENSEK MEGFELELŐI OS-BEN	70,— Ft
<i>Bárdos Attila:</i> A PROGRAMBIZONYÍTÁS ALAPJAI	70,— Ft
<i>Somogyi Katalin:</i> KERESKEDELMI RENDSZERSZERVEZÉSI ESETTANULMÁNY	62,— Ft
<i>Marschik Iván:</i> MIKROPROCESSOROK, MIKROGÉPEK	78,— Ft
<i>Bana István—Kováts Ágnes:</i> SZÁMÍTÓGÉPES INFORMÁCIÓRENDSZEREK FEJLESZTÉSÉNEK IRÁNYÍTÁSA	69,— Ft
<i>Dr. Szabó Gyula:</i> ON-LINE KOMMUNIKÁCIÓ A SZÁMÍTÓGÉPPAL	47,— Ft
<i>Komáromi Imre:</i> SZÁMÍTÓGÉPES GRAFIKA	50,— Ft
<i>Kiss Zoltán—Schrenpf József:</i> SZÁMÍTÁSTECHNIKA A VÁLLALATIRÁNYÍTÁSBAN	53,— Ft
<i>Halassy Béla:</i> ADATMODELLEZÉS, ADATBÁZIS-TERVEZÉS	72,— Ft
<i>Aszalós János—Erki Irén:</i> BEVEZETÉS A STRUKTURÁLT PROGRAMOZÁSBA	90,— Ft

**A KIADVÁNYOK MEGVÁSÁROLHATÓK:**

BUDAPEST: SZÁMOK Könyvesbolt, XI., Szakasits Árpád út 68. — 1115 • SKV Statisztikai és Számítástechnikai Könyvesbolt, II., Keleti Károly u. 10. — 1024 • ÁKV Műszaki Könyvtárház, VI., Liszt Ferenc tér 9. — 1061 Közgazdasági és Jogi Könyvesbolt, V., Münnich Ferenc u. 8. — 1051 • Technika Könyvesbolt, XI., Bartók Béla út 15. — 1114 • A Művelt Nép Könyvterjesztő Vállalat vidéki boltjaiban: **PÉCS:** Közgazdasági és Jogi Könyvesbolt, Szalai A. u. 6. — 7622 • **SZEGED:** Móra Ferenc Könyvesbolt, Kárász u. 5. — 6701 • **GYŐR:** Műszaki és Idegen nyelvű Könyvesbolt, Lenin út 18. — 9021 • **MISKOLC:** Műszaki és Idegen nyelvű Könyvesbolt, Széchenyi út 54. — 3530 • **DEBRECEN:** Ady Endre Könyvesbolt, Vörös Hadsereg útja 26. — 4024